DISSERTATION

OPTIMAL CHAOS CONTROL THROUGH REINFORCEMENT LEARNING

Submitted by

Sabino Gadaleta

Department of Mathematics

In partial fulfillment of the requirements

for the degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2000

COLORADO STATE UNIVERSITY

October 30, 2000

WE HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER OUR SUPERVISION BY SABINO GADALETA ENTITLED "OPTIMAL CHAOS CONTROL THROUGH REINFORCEMENT LEARNING" BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY.

Committee on Graduate Work

Charles W. Anderson

Michael Kirby

Robert E. Gaines

Gerhard Dangelmayr
Advisor

Rudy Weissmann
Department Head

ABSTRACT OF DISSERTATION

OPTIMAL CHAOS CONTROL THROUGH REINFORCEMENT LEARNING

We present an approach for the control of chaotic systems based on reinforcement learning. Reinforcement learning algorithms provide a solution to the optimal control of systems in situations where system dynamics and analytic information about the desired goal state are not available. We formulate the chaos control problem as a reinforcement learning problem to obtain a general purpose chaos controller which acts globally, allows control in non-stationary and noisy environments, and can be used in parametric or impulsive control applications. To reduce the computational complexity of the reinforcement learning problem, vector quantization techniques are applied and the control policy is approximated in the reduced space. To find minimal sufficient codebooks, we suggest a fixed-point sensitive growing chaos controller which combines a modification of Fritzke's Growing Neural-Gas algorithm with reinforcement learning. We demonstrate the algorithm in a variety of applications including low- and high-dimensional discrete and chaotic systems, logistic coupled map lattices and a multistable rotor.

Sabino Gadaleta
Department of Mathematics
Colorado State University
Fort Collins, Colorado 80523
Fall 2000

ACKNOWLEDGEMENTS

First of all, I wish thank my advisor, Gerhard Dangelmayr, for his constant guidance and support.

I wish to thank Markus Anderle, Charles Anderson, Carlos Bins, Gerhard Dangelmayr, Doug Hundley, Michael Kirby and Aubrey Poore for helpful discussions and explanations. Thanks go to Ulrike Feudel for bringing the rotor to our attention and to the unknown referees of our papers whose "criticism" led to several improvements in the algorithm and in the final version of this dissertation.

I want to thank everybody in the Department of Mathematics at CSU for their hospitality.

A special thank you goes to my parents who always supported me in my plans and studies.

Lastly I want to send my thanks and my love to my wife, Angela, who was always there for me.

To my wife and our little one.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

ab ovo usque ad mala
(from eggs to apples)

Many physical systems are known to exhibit chaotic dynamics in certain parameter regimes, where they display a rich variety of different behaviours. In principle, a chaotic system has access to an unlimited number of states which are, however, unstable and are visited by the system in an unpredictable manner. In many cases of interest, system performance can be improved by controlling the dynamics such that it is constrained to one of these unstable states. This problem has been first addressed by Ott, Grebogi and Yorke [OGY90] and since gained much interest and found many important engineering, physical or medical applications. Today it is believed that chaos can be of advantage in certain systems if we find ways to exploit and control it by using its dynamical characteristics.

Concerning "real world applications" which are influenced by noisy and non-stationary environments and where no analytical description in terms of a map or vector field is available, an ideal, general purpose chaos controller should show the following features: (1) does not require an analytical description of the system or previous knowledge about the location of desired states, (2) apply perturbations globally to allow the system to quickly reach desired neighborhoods (i.e. solve the targeting problem), (3) be robust against noise and non-stationarity, (4) offer the possibility of performance in real time.

This dissertation is concerned with the development of such a general purpose chaos controller on the basis of reinforcement learning. Reinforcement learning has been shown to solve optimal control problems in situations where no analytical system description is available. We will formulate the chaos control problem as a reinforcement learning problem and show through numerical simulations that this allows efficient control of chaotic systems.

The task of every control system is to transform a given initial state of a system into a desired *goal state* which matches certain performance criteria. This is achieved through control signals, consisting of the perturbations of either control parameters or system variables, and chosen according to a *control policy*. In *open-loop* or *non-feedback control*, policies are independent of the system state, while in *closed-loop* or *feedback control* policies prescribe controls as a function of the current system state. The problem of establishing control can be described as the problem of finding a control policy which optimizes a performance measure. In this dissertation we will generally assume no available explicit knowledge on system dynamics in which case the control policy has to be established iteratively through a feedback process. A general such *closed-loop control system* is summarized in Figure 1.1. The components and variables are defined in the next paragraph.

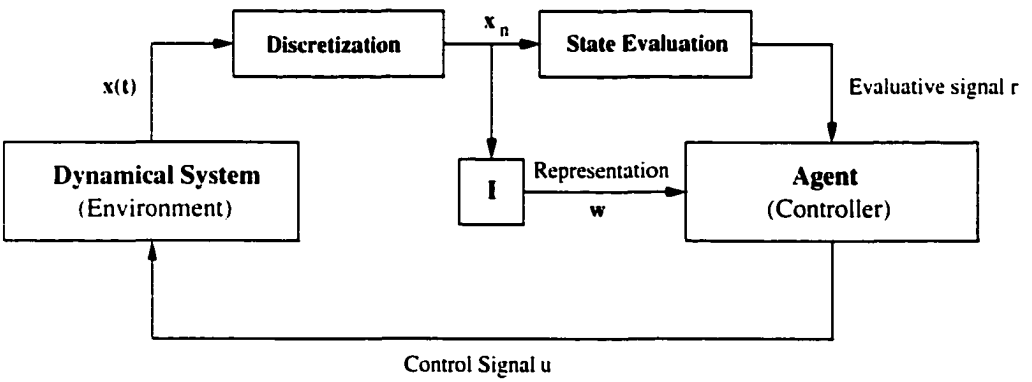


Figure 1.1: The main components of a general closed-loop control system.

The *dynamical system* or *environment* represents the entity to be controlled by the control system. In this work we focus on systems which exhibit complex dynamical behaviour such as chaotic, multistable or spatio-temporal systems. In the finite dimensional case, the state $\mathbf{x}(t)$ of the system is composed only of a finite set of state variables.

In most applications, the true state space will not be accessible, and knowledge about the system is obtained through discrete *measurements*, or *observations*, $h(\mathbf{x}(t_n))$ of the system at times t_n with measurement function h. The measurement process transforms possibly continuous dynamics into discrete dynamics. If the true state space is directly accessible it is often convenient nonetheless to introduce a discrete representation of the system through sampling of the trajectory at discrete time intervals. Throughout most of this dissertation we will assume that the only available knowledge on system dynamics is represented by a discrete set of states $\mathbf{x}_n \in \mathcal{X}$ with dynamics determined by a discrete (unknown) map, $\mathbf{x}_{n+1} = \mathbf{f}(\mathbf{x}_n)$. Although the discrete representation is only an approximation to the true system we will assume that it captures all necessary information and that it is in one-to-one correspondence with the original system. We will make the further assumption that the system states follow a *Markov process*: all the relevant information for the state $\mathbf{x}_{n+1}$ is contained in $\mathbf{x}_n$. In experiments the system dynamics will seldom follow a true Markov process but nonetheless reinforcement learning often produces good results. Among the requirements for convergence of a reinforcement learning algorithm to an optimal control policy (e.g. Markov process, infinite exploration, finite state and action spaces), the Markov assumption is among the less restrictive.

In control applications in which the optimal control policy is not a priori known, an evaluative component is necessary, which, based on the observations $\mathbf{x}_n$, guides the control process such that a measure for control performance is optimized.

In the case of supervised learning, a teacher would take the role of this component and transmit error signals, measuring the difference between current and desired state, to the controlling component. Supervised learning typically requires explicit knowledge on either the optimal control policy, the system dynamics or the goal state. If explicit analytical knowledge is not assumed reinforcement learning can be used to establish the control policy. In this case a *critic* evaluates the current state and provides failure/success reinforcement signals r to the controlling component.

If the *agent* (or *controller, controlling component*) has no control over the dynamics of the environment, the fundamental problem of interest is that of prediction [Sin94]. Prediction requires the approximation of the state space $\mathbb{X}$ on the basis of the set $\mathcal{X}$. In situations in which the agent has the possibility to influence the state of the system, the fundamental problem becomes that of control. This is the situation we are interested in throughout this dissertation.

Assuming Markov states, the control task can be described as a *Markovian decision process* (MDP) [Sin94]. MDPs are discrete time tasks in which at each of a finite number of time steps the agent can choose a control from a finite set of controls $\mathcal{U}$. At time step n, the agent receives the representation $\mathbf{w}_n \in \mathcal{W}$ of $\mathbf{x}_n$ and executes an action $u_n \in \mathcal{U}$. As a result, the system makes a transition to a state $\mathbf{x}_{n+1}$, which closes the loop.

The agent's task is to determine a policy for selecting controls that maximize some cumulative measure of the payoffs received over time. Such a policy is called an *optimal policy*. A policy is a function $\pi : \mathcal{W} \rightarrow \mathcal{U}$ that assigns a control to each state. The agent delivers *control signals, actions* or *perturbations* u to the system based on its current control policy and improves this policy based on the reward signals r received from the evaluative component.

The input function I, which determines how the agent views the environment, is frequently taken to be the identity. However, in applications in which $\mathcal{X}$ is

large, learning has to be performed in a reduced space $\mathcal{W}$. In this dissertation we construct $\mathcal{W}$ through a vector quantization technique.

The main assumptions we will make throughout this dissertation are as follows:

- No explicit analytical knowledge describing the system dynamics or the goal state is available.
- The complete control system is represented by a finite Markov process. This implies a discrete set of states and a discrete set of controls associated with each state.

We are not assuming strict stationary systems. Although the convergence proofs in reinforcement learning rely on this assumption, we will see that the proposed method is suitable in non-stationary environments.

This dissertation is concerned mainly with the control of chaotic systems. In Chapter 2 we will review important properties of chaotic systems. Of special interest in this context are the characteristics of strange attractors, the extraction of unstable periodic orbits from experimental data, and the discrete representation of continuous dynamics.

Different situations require different control strategies and as a result a large number of different approaches to chaos control have been suggested. Chapter 3 is an overview of many of these methods. In particular we will distinguish non-feedback and feedback methods operating continuously or discretely in time.

In this dissertation we will propose a time-discrete feedback chaos control algorithm based on reinforcement learning. The field of reinforcement learning is reviewed in Chapter 4. In particular we will present the temporal-difference approaches of Q- and Sarsa learning.

To reduce the complexity of the reinforcement learning problem we use clustering or vector quantization techniques. Chapter 5 introduces these techniques and focuses on self-organizing networks: the Neural-Gas and the Growing Neural-Gas algorithm.

In Chapter 6 we propose a chaos control algorithm based on reinforcement learning and vector quantization and demonstrate the algorithm in various parametric and impulsive control applications. The idea to use reinforcement learning to control chaotic systems has previously been introduced by Der and Herrmann [DH94] who applied it to the control of the logistic map. Our algorithm can be viewed as a generalization of their work in that it is able to control fixed points in non-stationary environments and allows parametric or impulsive control signals. Furthermore the chaos control of systems other than the logistic map with reinforcement learning techniques has previously not been demonstrated. In Chapter 7 we demonstrate control of a coupled map lattice and in Chapter 8 control of a multistable noisy rotor.

In Chapter 9 improvements are suggested which combine reinforcement learning with the Growing Neural-Gas algorithm. By approximating the exact location of the desired fixed point and a continuous local control policy in a neighborhood of it, a general purpose chaos control algorithm is obtained which uses minimal state representations and allows for vanishing control perturbations in ideal situations. The use of vector quantization techniques to reduce the computational complexity of the reinforcement learning problem has previously been suggested in [DH94] and recently in [FB99, GD99]. However the combination of a grid growing vector quantization algorithm and reinforcement learning into one algorithm has, to our knowledge, previously not been shown.

In Chapter 10 we conclude with a summary and suggestions for future improvements and research.

Chapter 2

DYNAMICAL SYSTEMS AND CHAOTIC ATTRACTORS

> The often repeated statement, that given the initial conditions we know what a deterministic system will do far into the future, is false.
>
> (P. Cvitanović, 1989, p. 3)

The main focus of this dissertation is the control of chaotic systems. In this chapter we review related concepts from dynamical system theory and introduce much of the notation used throughout this dissertation. Of particular importance for the control of chaotic systems are the fundamental properties of chaotic attractors and the existence of an infinite number of unstable periodic orbits. In addition to these concepts we will review the so-called So transformation which allows an elegant estimation of such orbits from experimental data. We refer to experimental data as data recorded from a system from which no analytical equations describing the system are known. Time series data from a measurement is an example of such data.

2.1 Introduction

Poincaré [Poi99], at the beginning of the 20th century, was the first to realize the possibility of deterministic dynamical systems showing unpredictable behaviour, now commonly referred to as chaos. The beginning of the modern study of chaos is typically associated with Lorenz's attempt to model convection in the atmosphere through computer simulations [Lor63]. Due to a slight numerical error in

initial conditions, Lorenz accidentally discovered the first experimental evidence of a system exhibiting sensitive dependence on initial conditions: Two initially close trajectories will diverge from each other at an exponential rate making long term predictions of the system's behaviour impossible. Since this first identification of a chaotic attractor, the phenomenon of chaos has been the subject of intense research and many systems exhibiting chaotic dynamics have been identified. Chaos is now believed to occur in the dynamics of many natural systems such as the atmosphere, the brain or the heart [1] and has been identified in many experimental systems such as lasers, electric circuits or chemical reactions (see [Gle87] for a history and overview).

2.2 Dynamical Systems

Many physical systems are well described in terms of finite-dimensional dynamical systems. The state $\mathbf{x}$ of the system depends then only on a finite set of d_x state variables,

$$\mathbf{x} = [x_1 \ \cdots \ x_{d_x}]^T \in \mathbb{X} \subset \mathbb{R}^{d_x},$$

and the system dynamics can be expressed as a system of autonomous first-order *ordinary differential equations*

$$\mathbf{x}' = \mathbf{F}(\mathbf{x}, \mathbf{p}), \quad \mathbf{x} \in \mathbb{X} \subset \mathbb{R}^{d_x}, \tag{2.1}$$

with a set of parameters $\mathbf{p}$. Given a non-autonomous system we can introduce the state-variable $x = t$ and obtain an autonomous system. The vector field $\mathbf{F}$ describes the time evolution of the states $\mathbf{x}$ and the d_x-dimensional space $\mathbb{R}^{d_x}$

[1] The role of chaos in biological systems is much debated. Although it is unlikely that dynamics of biological organisms can be viewed as low-dimensional chaotic it is believed that chaotic itinerancy, or multistability, plays a fundamental role in the behaviour of biological systems. In such systems chaos occurs as transient behaviour.

spanned by the variables $x_1, \cdots, x_{d_x}$ is referred to as *state space* or *phase space*.

The graph of a particular solution $\mathbf{x}(t)$ in phase space is called a *trajectory* or *phase curve*. An *orbit* specifies the set of all points on a trajectory. The set of all trajectories is described by the *flow*

$$o_t(\mathbf{x}) \; : \; \mathbb{R} \times \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_x},$$

with $\mathbf{x}(t) = o_t(\mathbf{x}(t_0))$. Uniqueness of solutions to the system of equations (2.1) implies that no crossings of trajectories exist in phase space. In this typical case for applications, o is a unique function.

Certain systems are not described by a continuous set of differential equations but rather by a *map* or *difference equation*

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

Given a continuous system, a discrete map can be associated to it either analytically or experimentally in various ways as discussed in Section 2.9. The discrete formulation is often more convenient since the analysis of the system can be performed without integration. Furthermore, a measurement process or an integration scheme naturally results in a discretization of time.

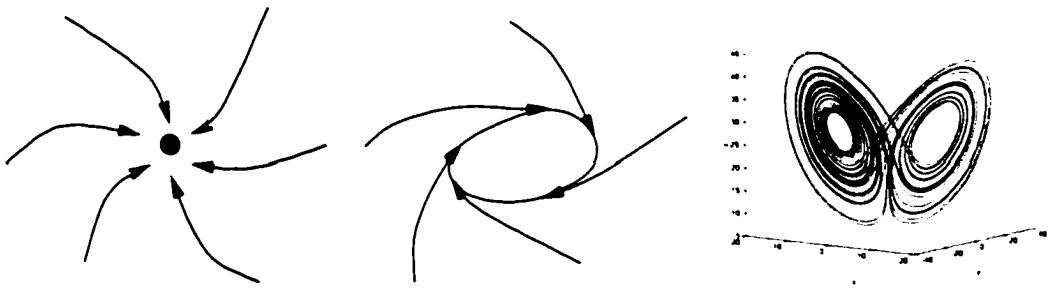
Depending on $\mathbf{F}$ (or $\mathbf{f}$) and initial values, typical trajectories will either go to infinity or stay in a bounded area forever. The set of initial conditions leading to the same asymptotic behaviour of the trajectory is called the *basin of attraction* [KS99]. The systems discussed in this dissertation typically do not only have bounded solutions but are also *dissipative*, which means that on average the phase space volume containing initial conditions is contracted under the dynamics. As a result, trajectories approach an attracting set $\mathcal{A}$ of measure zero which describes the asymptotic behaviour of the system and has the following properties [Wig96]:

(A1) It is a compact set, invariant under the flow: $\phi_t \mathcal{A} = \mathcal{A}$.

(A2) It has an open neighborhood which contracts to $\mathcal{A}$ under the influence of the flow.

(A3) It is topological transitive ($\mathcal{A}$ can not be partitioned into two closed, distinct and invariant sets).

Regular attractors such as limiting cycles, fixed points or tori characterize regular behaviour. Irregular behaviour is characterized by *strange* or *chaotic* attractors, which are fractal subsets of the phase space and can generally be embedded in a smooth invariant manifold of lower dimension than the phase space dimension. In Figure 2.1 examples of regular and chaotic attractors are shown. Systems possessing strange attractors are referred to as chaotic and will be the focus of this dissertation. We will review the properties of such attractors in more detail in the next section.



(a) Fixed point (b) Limit cycle (c) The Lorenz attractor

Figure 2.1: Examples of regular (a),(b) and strange (c) attractors.

2.3 Strange Attractors

Ruelle and Takens [RT71] discovered an area bounded in phase space which was attracting trajectories globally, and therefore behaving like an attractor, but

in which trajectories were locally diverging from each other at an exponential rate. They coined the term *strange attractor* for this type of limit behaviour. The interplay between convergence and divergence of trajectories gives rise to dynamics characterized by sensitivity to initial conditions making long-term predictions impossible.

Strange attractors can be defined in different ways. Wiggins [Wig96] defines a strange attractor $\mathcal{A}$ as an attractor which has the properties (A1) through (A3) and in addition the following property (S1).

(S1) The dynamics in $\mathcal{A}$ are *sensitive dependent on initial conditions*: There is an $\epsilon > 0$ such that, for any $\mathbf{x} \in \mathcal{A}$ and any neighborhood $\mathcal{N}$ of $\mathbf{x}$, there exists a $\mathbf{y} \in \mathcal{N}$ and $t > 0$ such that $|o_t(\mathbf{x}) - o_t(\mathbf{y})| > \epsilon$.

Properties (A1) - (A3) together with (S1) can be considered as defining conditions for a chaotic attractor. Devaney [Dev87] adds a further characteristic property of chaotic attractors:

(S2) The periodic orbits of the dynamics are dense in $\mathcal{A}$: Every ϵ neighborhood of a point $\mathbf{x} \in \mathcal{A}$ contains a point that belongs to a periodic orbit.

(S2) is an important characteristic of a strange attractor, in particular regarding the control of chaotic systems: The attractor contains a dense set of periodic orbits, which are all unstable, because otherwise the attractor would not be strange. A chaotic attractor can be thought of as a combination of an infinite number of *unstable periodic orbits* (UPOs) with dynamics continuously switching from one UPO to another while remaining close to an UPO only for a certain amount of time. If the dynamics are described by a map, an UPO corresponds to a *fixed point*. In Figure 2.2 several different UPOs of the Lorenz attractor (see Figure 2.1c) are shown in a two-dimensional projection. The presence of a strange attractor implies

sensitive dependence on initial conditions, or in other words, chaos. In the next section we will describe some basic mechanisms leading to the formation of a strange attractor.

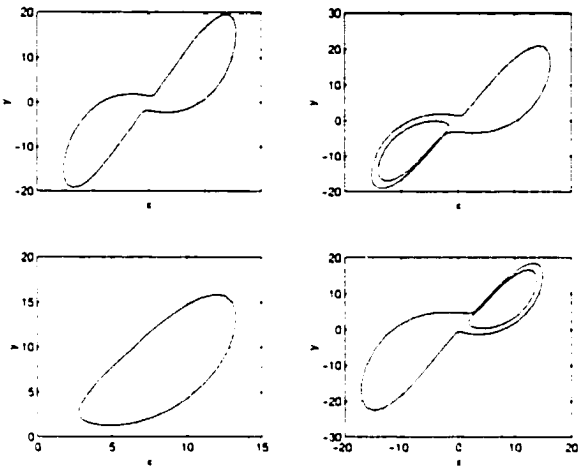


Figure 2.2: Different unstable periodic orbits of the Lorenz attractor.

2.4 The Creation of Chaos

A fixed point of a system $\mathbf{x}' = \mathbf{F}(\mathbf{x})$ is a point $\mathbf{x}_0$ such that $\mathbf{F}(\mathbf{x}_0) = 0$. Important for the creation of chaotic behaviour in a nonlinear system are *saddle* fixed points. In two dimensions, such fixed points possess one stable direction characterized by a negative eigenvalue λ_s of the Jacobian matrix of $\mathbf{F}$ at $\mathbf{x}_0$ with corresponding eigenvector $\mathbf{u}_s$, and one unstable direction characterized by an eigenvalue $\lambda_u > 0$ with eigenvector $\mathbf{u}_u$. Close to a saddle fixed point $\mathbf{x}_0$ trajectories converge exponentially ($\sim e^{\lambda_s t}$) towards $\mathbf{x}_0$ along the stable direction and diverge ($\sim e^{\lambda_u t}$) from $\mathbf{x}_0$ along the unstable direction (see Figure 2.3a). This local behaviour leads to the definition of local stable and unstable manifolds, $W^s_{loc}(\mathbf{x}_0)$ and $W^u_{loc}(\mathbf{x}_0)$, in the neighborhood $\mathcal{N}$ of a saddle fixed point $\mathbf{x}_0$ [AFH94] (see Figure 2.3b):

$$W^s_{loc}(\mathbf{x}_0) := \{\mathbf{x} \in \mathcal{N} : \phi_t(\mathbf{x}) \to \mathbf{x}_0, \text{ for } t \to +\infty \text{ and } \phi_t(\mathbf{x}) \in \mathcal{N}, \forall t \geq 0\}$$

$$W^u_{loc}(\mathbf{x}_0) := \{\mathbf{x} \in \mathcal{N} : \phi_t(\mathbf{x}) \to \mathbf{x}_0, \text{ for } t \to -\infty \text{ and } \phi_t(\mathbf{x}) \in \mathcal{N}, \forall t \leq 0\}.$$

Extending the local manifolds beyond $\mathcal{N}$ one can define global stable and unstable manifolds, W^s and W^u, of the fixed point (see Figure 2.3c).

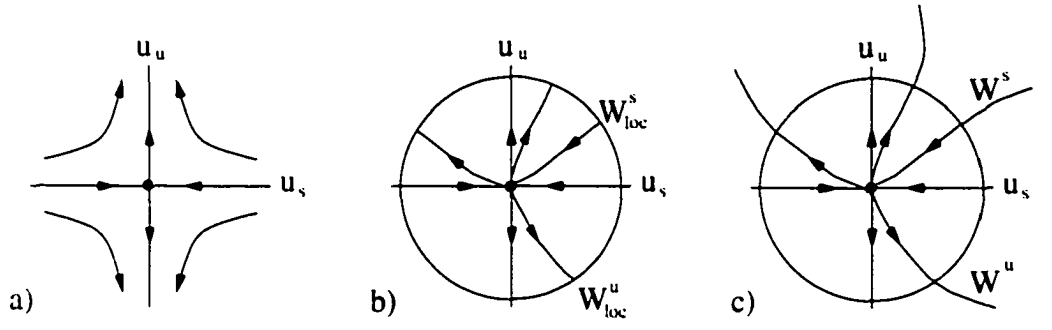


Figure 2.3: a) A saddle fixed point. b) Local stable and unstable manifolds W^s_{loc}, W^u_{loc}. c) Global stable and unstable manifolds W^s, W^u.

Due to uniqueness of solutions, W^s and W^u can never self-intersect, but it is possible that W^s and W^u intersect with each other. If W^s and W^u meet at a point with equal tangent, then the unstable manifold returns to its origin (see Figure 2.4a). The resulting orbit, which goes to the same fixed point for $t \to \pm\infty$ is called a *homoclinic orbit* and is structurally unstable, i.e., small perturbations will destroy it as shown in Figure 2.4b and c.

For two-dimensional maps $\mathbf{x}_{n+1} = \mathbf{f}(\mathbf{x}_n)$ a fixed point $\mathbf{x}_0$, $\mathbf{x}_0 = \mathbf{f}(\mathbf{x}_0)$, is a saddle if the Jacobian has one stable eigenvalue, $|\lambda_s| < 1$, and one unstable eigenvalue, $|\lambda_u| > 1$. The definition of the corresponding stable and unstable manifolds is analogous to the continuous case, but now these manifolds can intersect *transversally* (with nonzero angle) in isolated points as shown in Figure 2.4d.

Such points are called *homoclinic points* and the existence of one homoclinic point implies the existence of infinitely many such points [Wig96] as can be understood from Figure 2.5a. Let x be the point of intersection between unstable and stable manifold, with x' ahead of x on the stable manifold and x'' ahead of x on the unstable manifold. The mappings Tx' and Tx'' must be ahead of the mapping Tx.

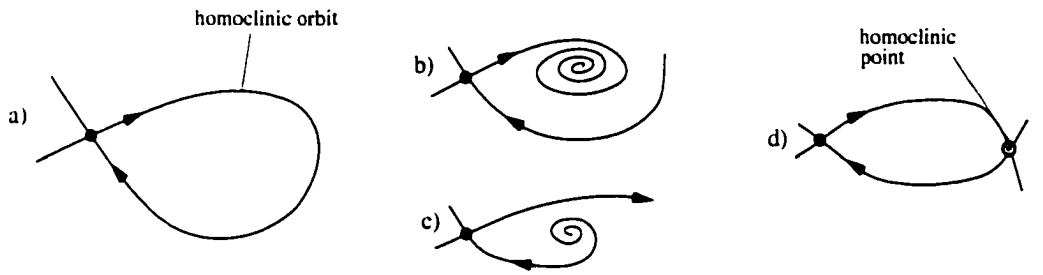


Figure 2.4: A homoclinic orbit (a) and its destruction into regular (b),(c) and chaos creating behaviour (d).

This implies that the unstable manifold loops back and crosses the stable manifold in another homoclinic point Tx. Then another loop must be formed intersecting at another homoclinic point T^2x. Since T^2x is closer to the saddle point than Tx, the distance between T^2x and Tx is less than the distance between Tx and x. Area preservation requires the area of each loop to be the same and implies that the loops must extend further. This further implies an infinite "oscillation" of W^u around W^s which accumulates at $\mathbf{x}_0$. Iterating backward, we find also an oscillation of the stable manifold around the unstable manifold and we obtain a picture as shown in Figure 2.5b. The resulting dynamics is very complex and does not allow long-term predictions. Similar behaviour can be observed at homoclinic points of orbits of higher period.

Typically the existence of homoclinic points leads to transient chaotic dynamics. This type of behaviour can also occur in three (and higher) dimensional continuous systems, such as, for example, the Lorenz equations

$$x' = -\sigma(x-y)\ ,\quad y' = rx - y - xz\ ,\quad z' = -bz + xy. \tag{2.2}$$

Fixing $\sigma = 10$ and $b = 8/3$, and varying r one observes for $r > 1$ a saddle fixed point at the origin. For $1 < r < 13.926...$ the unstable manifolds are caught in two sinks C_1, C_2 as shown in Figure 2.6a. At the critical value $r = 13.926...$ a homoclinic bifurcation occurs (see Figure 2.6b). For larger r values homoclinic

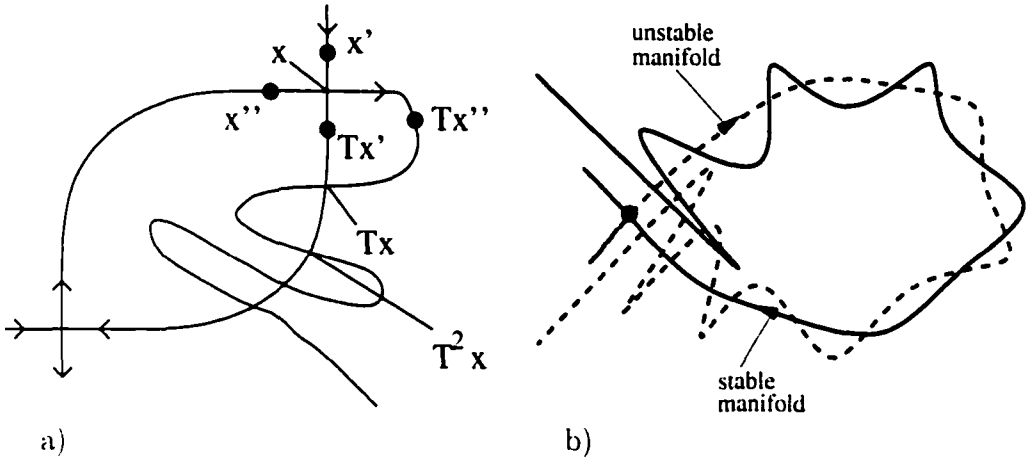


Figure 2.5: Stable and unstable manifold of a saddle fixed point in the case of transversal intersections between the manifolds. The dynamics is sensitive to the initial conditions due to the infinite number of "oscillations" of the manifolds.

points exist, however the chaotic attractor is not observed until $r > 24.74...$. The intermediate behaviour $13.926... < r < 24.74...$ is called *metastable chaotic* [AFH94] and is characterized by chaotic transients. The trajectory eventually approaches one of the two stable fixed points. The effect of the homoclinic points is a "fractalization" of the basin boundaries of the fixed points leading to sensitive dependence on initial conditions: Initially close trajectories can approach different fixed points. At the critical value $r = 24.74...$ the sinks C_1 and C_2 turn into saddles with a two-dimensional unstable and a one-dimensional stable manifold. The familiar chaotic Lorenz attractor (see Figure 2.1c) appears for $r > 24.74...$.

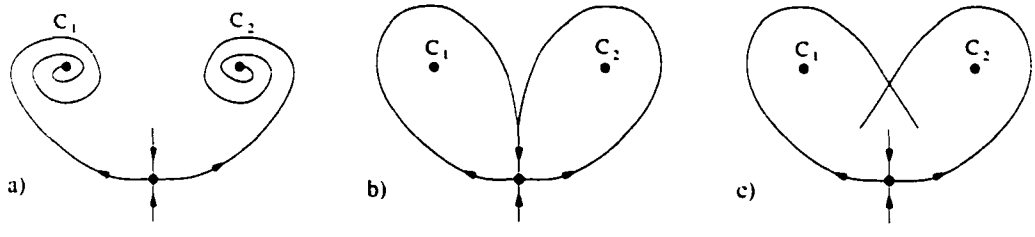


Figure 2.6: Homoclinic bifurcation of the Lorenz attractor. a) $1 < r < 13.926....$ b) $r = 13.926....$ c) $r > 13.926....$

In the next section we will review some basic invariant measures characterizing the behaviour of chaotic dynamics.

2.5 Invariant Measures

A large part of research in dynamical system theory has concentrated on the characterization of strange attractors through *invariant measures.*[2] Typical invariant measures of a strange attractor are its fractal dimensions, the Lyapunov exponents and the topological entropy (see [Far82, ER85]).

2.5.1 Lyapunov Exponents

Lyapunov exponents are a measure for the unpredictability of a chaotic system. Recall that the dynamics close to a saddle fixed point is characterized by an exponential divergence (convergence) along the unstable (stable) direction proportional to $e^{\lambda_u t}$ ($e^{\lambda_s t}$). Trajectories of a chaotic system show divergence and convergence in different directions at exponential rates everywhere on the attractor. The properly averaged exponents of this divergence or convergence are called *Lyapunov exponents* and are characteristic for the system, i.e., they quantify the strength of the chaos. A chaotic attractor must have at least one positive and one negative Lyapunov exponent. Chaotic systems with more than one positive Lyapunov exponent (divergence in at least two directions) are called *hyperchaotic.*

The largest positive Lyapunov exponent provides a measure of the predictability horizon of a chaotic system. Consider two trajectories $\mathbf{x}_1(t)$, $\mathbf{x}_2(t)$ and let $\mathbf{x}(t) = \mathbf{x}_1(t) - \mathbf{x}_2(t)$. Suppose that the initial conditions are close ($\|\mathbf{x}_0\|$ small) and let σ_1 be the largest positive Lyapunov exponent. Then

$$\|\mathbf{x}(t)\| \approx e^{\sigma_1 t}\|\mathbf{x}(0)\|$$

---

[2]Invariant measures must be invariant under the flow and independent of the coordinate system in which they are computed.

is an estimate of the separation of the trajectories. If $\|\mathbf{x}(t)\|$ has reached the size of the attractor, predictability of the system is lost. Let τ be one time unit. An error in an initial condition will grow according to $\exp(\sigma_1 t/\tau)$ and predictability is lost if $\|\mathbf{x}(t)\| \sim |\mathcal{A}|$, where $|\mathcal{A}|$ denotes the characteristic length of the attractor. The length of the *prediction horizon* is therefore

$$t_{hor} \sim \frac{\tau}{\sigma_1} \ln \frac{|\mathcal{A}|}{\|\mathbf{x}(0)\|}.$$

2.5.2 Fractal Dimensions

Fractal dimensions are characteristic measures for the geometric form of an attractor and are related to the distribution of points in phase space. They contain information about how the number of points in a sphere of radius r scales as $r \to 0$. Motivation for this definition is the fact that the volume of a d-dimensional sphere scales as r^d. Analogously a dimension can be defined for a point distribution (a single point has dimension zero) as follows.

Let $n(\mathbf{x}, r)$ be the number of attractor points of a map contained in a sphere of radius r centered at $\mathbf{x}$. Let $\rho(\mathbf{x})d\mathbf{x}$ be the *natural invariant measure* of the attractor, the average time a typical trajectory spends in the phase space element $d\mathbf{x}$. The *correlation integral* of order q is defined as

$$C(q, r) = \int \rho(\mathbf{x}) n(\mathbf{x}, r)^{q-1} d\mathbf{x}.$$

In the limit $r \to 0$ one expects that C scales according to a power law,

$$C(q, r) \sim r^{(q-1)d_q},$$

which defines a fractal dimension d_q. This leads to the following definition of the *generalized fractal dimension* d_q:

$$d_q = \lim_{r \to 0} \frac{\ln C(q, r)}{(q-1) \ln r}.$$

If the limit exists we have $d_{q'} \geq d_q$ for $q' < q$: the larger q, the more weight is put on regions which are visited more frequently. d_0 is the *box-counting dimension* , d_1 the *information dimension*, and d_2 the *correlation dimension*.

Many methods exist to determine Lyapunov exponents and fractal dimensions from experimental data (see e.g. [KS99]). In [GOY88, LK89] the invariant measures were linked to the UPOs of the attractor. Thus besides of their importance for control of chaos the determination of UPOs of a chaotic attractor is also important to characterize their behaviour. We will briefly discuss methods to determine UPOs in the next section.

2.6 Determining Unstable Periodic Orbits

Methods to identify the location of UPOs can be divided into methods which assume experimental data and methods which require an analytical description of the system dynamics. The simplest method to identify UPOs from experimental data relies on finding points of close return (*recurrent points*) in phase space [LK89]. Assuming that dynamics are described by a map, a point $\mathbf{x}_n$ satisfying $\|\mathbf{x}_{n+k} - \mathbf{x}_n\| < \epsilon$ is called a (k, ϵ) recurrent point and corresponds to a fixed point of period k.[3] A disadvantage of this method is that a large number of data points is necessary in order to accurately identify UPOs of large period. Furthermore, the ball size ϵ has to be given in advance. If ϵ is too large different UPOs might not be resolved. However if ϵ is too small, not enough recurrent points might occur in reasonable time to allow estimation of the UPO. This *method of close returns* was successfully applied to data from the Belousov-Zhabotinskii reaction [LK89] and to the detection of UPOs in human cardiac rhythms [NGG98]. An improvement of

[3] A fixed point of period smaller k can also satisfy $\|\mathbf{x}_{n+k} - \mathbf{x}_n\| < \epsilon$. For this reason we use a modified measure for recurrence in Chapter 9.

this method was presented by So *et al.* and will be discussed in the next subsection. We will apply the So method in our control algorithm in Chapter 9.

2.6.1 The So Transformation

So *et al.* [SOS+96, SOS+97] proposed a transformation of experimental data such that the transformed data is concentrated on the periodic orbits.

Consider a map $x_{n+1} = f(x_n)$ and assume that we want to estimate a fixed point x_f such that $x_f = f(x_f)$. So *et al.* suggested the transformation

$$x_n \longrightarrow \hat{x}_n = [x_{n+1} - s_n(k)x_n] / [1 - s_n(k)].$$

with

$$s_n(k) = (x_{n+2} - x_{n+1})/(x_{n+1} - x_n) + k(x_{n+1} - x_n).$$

The density function for $\hat{x}$ can be shown [SOS+96] to have singularities $\sim |\hat{x} - x_f|^{-1/2}$ at x_f which implies that the histogramm of $\hat{x}$ will have sharp peaks at x_f. Without requiring the definition of a ball size ϵ, all points that lie in the linear region of x_f are transformed into the vicinity of x_f. Consider as an example the logistic map $f(x) = 3.8x(1-x)$ which has a fixed point at $x_f = 1 - 1/3.8 \approx 0.7368$. In Figure 2.7a a histogram of the untransformed data is shown. Figure 2.7b shows the transformed data $\hat{x}$ with $k = 0$. We see that a sharp peak appears at the location of the fixed point. For fixed k the histogram typically will contain spurious peaks not due to fixed points which depend on k. Averaging the transformed data over different k will lead to a single peak at the desired location (see Figure 2.7c).

In [SOS+96, SOS+97] So *et al.* present the transformations for general high-dimensional systems and data in the form of embedding vectors (see next section). This method has been applied to interburst data from a rat hippocampal slice [SOS+97] and to data from human epileptic activity [QMAV97].

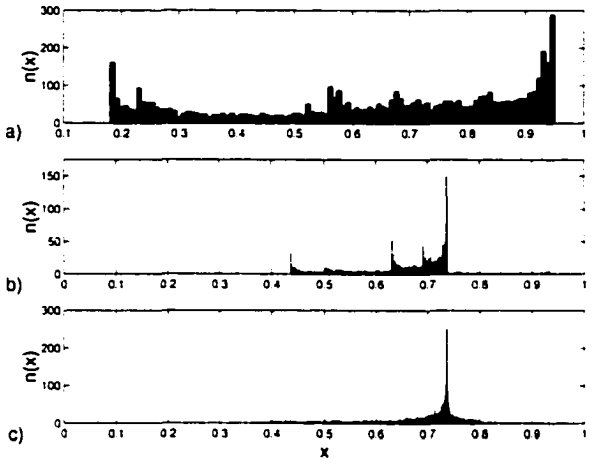


Figure 2.7: Histogram of a) the logistic map $x_{n+1} = 3.8x_n(1 - x_n)$, b) the transformed data $\hat{x}$ with $k = 0$, and c) averaged over different k.

2.6.2 Root Finding Methods

The methods based on close returns and its modifications do not assume knowledge about system dynamics. If such knowledge can either be extracted from the observed data (see [KS99] for an overview of reconstruction methods) or if it is actually available, then root-finding methods can be applied which rely on finding roots of the function

$$\mathbf{g}(\mathbf{x}) = \mathbf{f}^{(k)}(\mathbf{x}) - \mathbf{x}.$$

with k denoting the period of the desired fixed point. The root is typically found through an iterative scheme and an obvious choice would be a Newton-type method (see [PFTV86]) which calculates updates according to [DL99a]

$$-\mathbf{J}(\mathbf{x})\Delta\mathbf{x} = \mathbf{g}(\mathbf{x}).$$

Newton methods however require a good initial guess and as a result a very fine grid has to be used. Schmelcher and Diakonos [SD98] suggested a universal set of linear transformations which transforms the unstable orbits into stable ones and

proposed updates of the form

$$\Delta \mathbf{x} = \lambda \mathbf{C}\mathbf{x}.$$

With an appropriate choice of $\mathbf{C}$ and λ this method can find any periodic orbit of a chaotic system. A more efficient method based on a similar principle was derived in [DL99a].

As mentioned before one distinguishes methods assuming system knowledge and methods which do not. Although a model can in principle be established from experimental data, the approximation of this model suffers from the same problems as the methods of close returns: given a complicated system, to approximate a model which is good enough to allow accurate approximation of UPOs would typically require a large amount of data. As a result, experimentalists usually only determine low-period orbits. Fortunately it was shown in [HO96] that generically low-period orbits give the major contribution to invariant measures and can be considered more important than high-period orbits.

2.7 Reconstruction of Phase Space

In many practical applications the observed dynamical system is unknown and the only information available is obtained through discrete time measurements of certain observable quantities. For simplicity we assume that only one quantity is observed. The result of a measurement with sampling time τ is a time series

$$\{s(1),\ s(2), \cdots\},\ \ s(n) = h(\mathbf{x}(n\tau)),\ n \in \mathbb{N}^+. \tag{2.3}$$

with the measurement function $h : \mathbb{X} \to \mathbb{R}$. An important result for experimentalists was obtained by Takens 1981 [Tak81]. Takens showed that the measurement of a single time series allows the reconstruction of a space which is equivalent to the original phase space of the system in the sense that the invariant measure of the

attractor can be estimated in the reconstructed phase space. From the time series so-called *embedding vectors* are constructed which form an *embedding space* that is in one-to-one correspondence with the true phase space as far as the embedding of the attractor is concerned.

2.7.1 Embedding Vectors

The possibility to reconstruct the phase space and its embedded attractor, or in other words "all relevant degrees of freedom", from a measured time series relies solely on the nonlinearity of the underlying dynamical system. As an example consider the Lorenz equations (2.2). The nonlinear terms xz and xy couple all variables of the system such that the change of one variable is influenced by all other variables. As a result, the observation of just one variable allows to reconstruct the dynamics created by the complete system.

To reconstruct the phase space based on the time series (2.3), generally *time delay embedding vectors*

$$\mathbf{s}_n = [s(n),\ s(n+T),\ \cdots,\ s(n+(d_e-1)T)]^T,\quad n = 1,\ \cdots,\ \mathcal{N}(d_e),\quad \mathbf{s} \in \mathcal{S}, \tag{2.4}$$

are constructed, with *time delay* T and embedding dimension d_e. The number $\mathcal{N}(d_e)$ of available embedding points depends on the embedding dimension d_e. For correctly chosen T and d_e, the geometrical object formed by the $\mathbf{s}$ will be in one-to-one correspondence with the original trajectory. We refer to $\mathcal{S}$ as the *reconstructed phase space*. In Figure 2.8 we used the Lorenz equations (2.2) to illustrate the reconstruction process. In Figure 2.8a the attractor $\mathcal{A}$ in the true phase space is shown. The x-coordinate is measured and the resulting time series is shown in Figure 2.8b. The time series is embedded with $T = 5$ and $d_e = 3$ and the geometrical object $E(\mathcal{A})$ obtained in this way is shown in Figure 2.8c.

In the next subsections we introduce briefly the most important embedding theorems and methods to determine the embedding parameters.

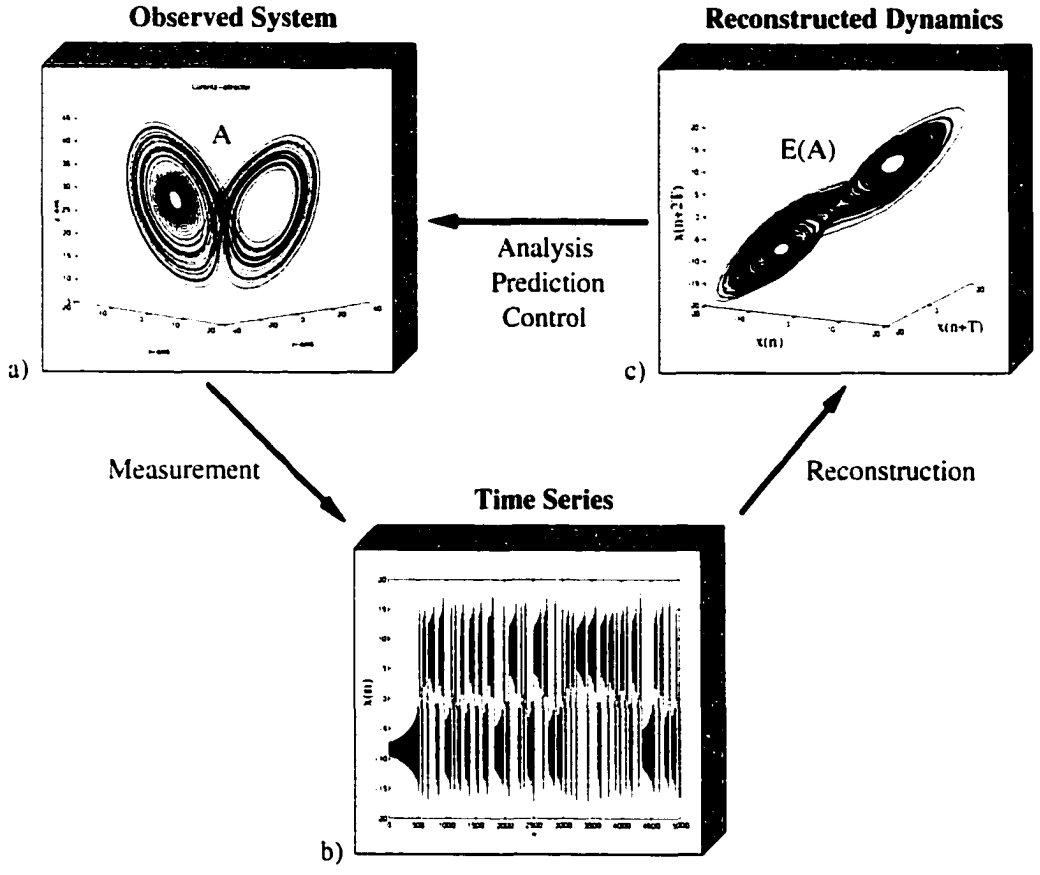


Figure 2.8: Reconstruction of the Lorenz attractor through a time series embedding. a) The Lorenz attractor in the true phase space spanned by x, y, z. b) Recording of the x-coordinate. c) Embedding vectors with $T = 5$ and $d_e = 3$.

2.7.2 Embedding Theorems

Suppose we are given a dynamical system with attractor $\mathcal{A} \subset \mathbb{X}$ and that information about the system is available through an observation process which in general can be described as a map $E : \mathcal{A} \rightarrow E(\mathcal{A})$, from the attractor to its image $E(\mathcal{A})$. An *embedding* is a diffeomorphism[4] from $\mathcal{A}$ onto $E(\mathcal{A})$ [SYC91]. The embedding theorems below describe under what general conditions on $\mathcal{A}$ and E the

[4]A smooth one-to-one map which has a smooth inverse.

map E is an embedding. The first embedding theorem was given 1936 by Whitney [Whi36]:

Theorem 2.1 (Whitney's Genericity Embedding Theorem)
A d-dimensional smooth manifold can be embedded in $\mathbb{R}^{2d+1}$. The set of embedding maps is dense and open in C^1.

To illustrate this theorem, assume we are given a 1-dimensional manifold ($d = 1$) in a 3-dimensional space, e.g. a circle as shown in Figure 2.9a. The topology of the circle is preserved if we embed it in two dimensions. However, if the circle is deformed, the embedding in two dimensions might not be sufficient and an embedding in $2d + 1 = 3$ dimensions can be necessary (see Figure 2.9b). The false embedding (here a projection) is characterized by a self-intersection of the image which is not present in the original manifold. Elimination of these false intersections is the basic idea of the false-nearest neighbor method (see Section 2.8.2) to estimate a correct embedding dimension.

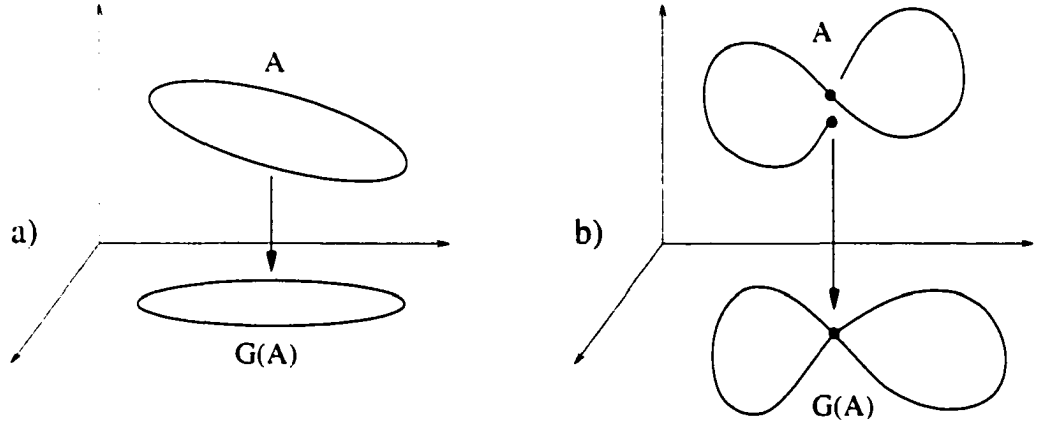


Figure 2.9: Embedding of a 1-d manifold in two dimensions. a) For a true circle the 2-dimensional embedding is sufficient. b) The deformed circle can not be embedded in a two-dimensional space.

Whitney's theorem is not sufficient for constructing embeddings of attractors from measurements for two reasons: 1) it assumes smooth manifolds of integer

dimension and is as such not adequate for strange attractors, and 2) it only guarantees that the set of embeddings is dense, which means that every map is only arbitrarily close to an embedding. However one would like to conclude that a given map is an embedding with probability one and denseness alone can not guarantee this since open dense sets of small Lebesgue measure exist [SYC91]. Two generalizations of this theorem were presented in 1991 by Sauer, Yorke and Casdagli [SYC91]. The first theorem guarantees that a map is an embedding with probability one.

Theorem 2.2 (Whitney Embedding Prevalence Theorem)
A d-dimensional compact smooth manifold $\mathcal{A}$ can be embedded in $\mathbb{R}^{2d+1}$. Almost every[5] *smooth map $\mathcal{A} \to \mathbb{R}^{2d+1}$ is an embedding of $\mathcal{A}$.*

The second theorem extends the statement to fractal sets.

Theorem 2.3 (Fractal Whitney Embedding Prevalence Theorem)
Let $\mathcal{A}$ be a compact subset of $\mathbb{R}^k$ of box-counting dimension d_0, and let n be an integer greater than $2d_0$. For almost every smooth map $E : \mathbb{R}^k \to \mathbb{R}^n$,

1. *E is one-to-one on $\mathcal{A}$.*
2. *E is an immersion*[6] *on each compact subset C of a smooth manifold contained in $\mathcal{A}$.*

The last theorem guarantees that it is sufficient to observe $2d_0 + 1$ independent quantities simultaneously in order to reconstruct an attractor of box-counting dimension d_0. In experimental situations however it is in general not possible to

---

[5] *Almost every* in the sense of prevalence as defined in [SYC91]: A Borel subset S of a normed linear space V is *prevalent* if there is a finite-dimensional subspace E of V such that for each $v \in V$, $v + e$ belongs to S for (Lebesgue) almost every $e \in E$.

[6] The Jacobian of E is one-to-one at every point on $\mathcal{A}$.

observe a large number of independent observables and the question is how to form a map which allows a reconstruction of the phase space from just one observable. To this end Taken's [Tak81] introduced the time delay embedding vectors (2.4) and proved the following theorem [Tak81, KS99].

Theorem 2.4 (Takens' Delay Map Embedding Theorem)
It is a generic property[7] *that a delay map of dimension* $2d + 1$ *is an embedding of a compact manifold with dimension* d *if the measurement function* h *is* C^2 *and couples all degrees of freedom.*

Takens' theorem is for delay embeddings what Whitney's theorem is for general embeddings. Again Sauer *et al.* [SYC91] generalized Takens' theorem:

Theorem 2.5 (Fractal Delay Embedding Prevalence Theorem)
A delay map with integer embedding dimension $d_e > 2d_0$ *is an embedding of an attractor with box-counting dimension* d_0 *for almost all (in the sense of prevalence) smooth measurement functions* h *and sampling times* $\tau > 0$, *if there are no periodic orbits of the system with period* τ *or* 2τ *and only a finite number of periodic orbits with period* $k\tau, k > 0$.

The important consequence of the embedding theorems is that an experimenter can claim to have found an embedding with probability one if d_e is chosen correctly. An embedding preserves the invariant measures of the original system and gives a reconstructed attractor which is in one-to-one correspondence with the original attractor. It turns out however that it is not always necessary to have one-to-one correspondence in order to preserve all possible invariant measures [DGOS93].

[7] A property is generic if the set of maps having this property is open and dense. Thus "generic" is weaker than "almost all".

For certain applications $d_e < 2d_0 + 1$ still leads to good results. In [SSOY98] for example it was shown that the prediction of a time series is possible in certain cases even if $E(\mathcal{A})$ shows self-intersections.

The theorems assume that one has access to infinitely many exact measurements and do not make any statement about the choice of the time delay. In practice however one is limited to a finite number of noisy measurements and the choice of a correct delay is critical. While the theorems provide a fundamental theoretical result on how to choose d_e from the dimension of the manifold, they give no information on how to find this dimension. To estimate the dimension of the attractor, we need a reconstruction of the phase space in the first place. Many different methods have been, and are currently being, developed to estimate correct embedding parameters from experimental data. We give a brief overview of these methods in the next subsection.

2.8 Determining Embedding Parameters from Experimental Data

2.8.1 Determining the Time Delay

Since we are attempting to extract as much information about the original attractor as possible through the embedding process, most methods apply information theoretical approaches to determine T, guided by the following two principles [CEFG91, Aba95]:

- *Redundancy*: If T is too small, two consecutive embedding vectors will be redundant and not contribute new information. For example, measuring sunspots in terms of seconds will produce large amounts of redundant data.
- *Irrelevance*: If T is too large, then two consecutive embedding vectors will be statistically uncorrelated and not reveal new information about the dynamics. For example, measuring heart beats in terms of hours will not reveal information on possible arrhythmia.

From the two examples we see that the time delay is directly related to the measurement sampling time. If the sampling time is not chosen properly, even an optimal embedding will not reveal information on the dynamical behaviour of the system.

The simplest method to determine an adequate time delay is to embed the attractor for different T and to use the eye to decide for which T the attractor appears optimally unfolded. If T is too small (see Figure 2.10a), the attractor will only cover a small area of phase space and be distributed along the identity line (redundancy). If T is too large (see Figure 2.10c), then the geometrical structure of the attractor will become distorted (irrelevance). Naturally this method will only work if the attractor can be embedded in small dimensions. Also if the dynamics are very noisy or complex, this method will fail.

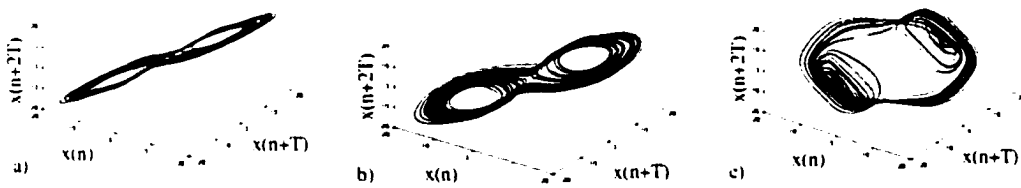


Figure 2.10: Embedding of the Lorenz attractor for different time delay. a) T=1, b) T=3, c) T=10.

A number of methods have been proposed which quantify this geometrical idea [KS99]: The *wavering product* [LS89] measures near neighbor distances, the *fill factor* [BP92] measures the phase space volume covered by data and the method of *displacement from the diagonal* [RCL94].

Methods not based on geometrical ideas are typically based on information theory and measure the dependence of two successive embedding vectors. A naive

choice, requiring linear independence, is to choose the value for which the auto-correlation function

$$\frac{\langle s(n)s(n-T)\rangle - \mu^2}{\sigma^2}.$$

with $\mu = \langle s(n)\rangle$, $\sigma^2 = \langle (s(n)-\mu)^2\rangle$ and $\langle\ .\ \rangle$ denoting expectation value, first passes through zero [FS86]. A more appropriate measure for nonlinear systems is the *mutual information* which is based on Shannon's information theory [Sha48]. It was introduced in 1968 by Gallagher [Gal68] and measures the general dependence of two events. It is defined as the sum of the entropies of the two isolated events minus the entropy of the joint event.

Let $p(s(n))$ denote the probability that the nth measurement gives $s(n)$. The average information, or entropy, gained by this measurement is defined as

$$S_n = -\sum_{n=1}^{N} p(s(n)) \log_2 p(s(n)).$$

From this the mutual information is formed as

$$I(T) = S_n + S_{n+T} - S_{n|n+T} = 2S_n - S_{n|n+T}.$$

If T is zero, then $S_{n|n+T} = S_n$ and $I(T) = S_n$. If T is too large, no information will be gained from the joint measurement and $S_{n|n+T} = 0$ giving $I(T) = 2S_n$. Note that $S_n \geq S_{n|n+T} \geq 0$. Although $I(T)$ does not necessarily have a minimum, the hope is that it has and Shaw [FS86] suggested to use the first minimum of $I(T)$ to determine T. Since S_n does not depend on T in practice the first minimum of the *average mutual information* [Aba95],

$$S(T) = \sum_{n,n+T} p(s(n), s(n+T)) \log_2 \frac{p(s(n), s(n+T))}{p(s(n))p(s(n+T))}.$$

is determined. Practical algorithms to compute the mutual information are based on estimation of the probabilities based on binning of the state space [FS86, Fra89,

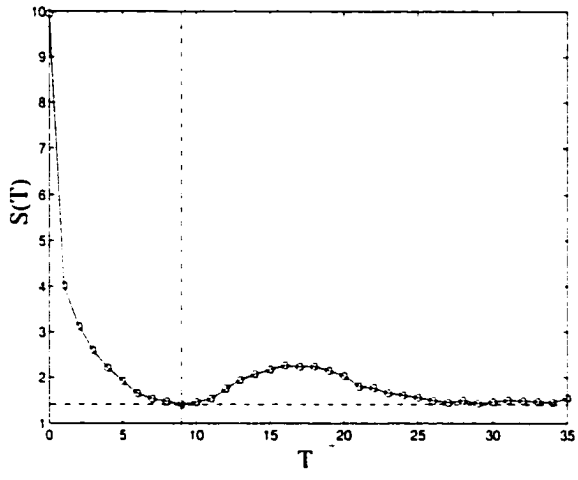


Figure 2.11: Average mutual information $S(T)$ of embedded Lorenz data.

PS94]. See Figure 2.11 for an example of $S(T)$ for the embedded Lorenz data, computed with an implementation of Fraser and Swinney's [FS86] algorithm. The first minimum occurs at $T = 9$.

The computation of the mutual information is computationally expensive and requires large data sets [KES99]. Liebert and Schuster [LS89] suggested to use the first minimum of the *generalized correlation integral*

$$C(d_e, T, N, r) = \frac{2}{N(N-1)} \sum_{1 \le i < j \le N} \Theta(r - \|\mathbf{s}_i - \mathbf{s}_j\|), \quad r > 0,$$

where

$$\Theta(a) = 0, \quad \text{if } a < 0, \text{ and } \Theta(a) = 1 \text{ otherwise.}$$

Kim *et al.* [KES99] introduced the efficient *C-C method* to determine optimal delay times based on computation of the correlation integral.

2.8.2 Determining the Embedding Dimension

The embedding theorems imply that it is possible to determine an euclidean space $\mathbb{R}^{d_e}$ large enough to allow a unique unfolding of a set of points of fractal

dimension d_0. The requirement $d_e > 2d_0$ is only sufficient but not necessary. For the Lorenz attractor $d_0 \approx 2.06$ and the embedding theorem would require $d_e = 5$. But it turns out that this attractor can be uniquely reconstructed with $d_e = 3$. In theory it is not problematic to work with unnecessary large embedding dimensions but in practice computational requirements to compute the invariant measures increase with d_e. Furthermore, unnecessary dimensions amplify noise [ABST93]. Therefore even if the box-counting dimension is known, it is of value to compute the lowest necessary embedding dimension. Three basic ideas can be used to choose minimum embedding dimensions: Computation of invariant measures, principal component analysis and detection of false nearest neighbors.

Computing Invariant Attractor Measures.

By computing an invariant measure for increasing embedding dimension one can identify a necessary embedding dimension from the embedding dimension at which the value of the measure begins to converge. The Grassberger-Procaccia algorithm [GP83] and the method by Pineda and Sommerer [PS94] implement this idea. As noted in [Cao97], this approach is time-consuming, requires large datasets and often does not allow clear interpretation of results. In Figure 2.12a we show the generalized dimensions d_1 and d_2 of the embedded Lorenz data computed in different embedding dimensions. We can conclude $d_e = 4$. It is important to note that invariants can be computed at dimensions which do not completely unfold the attractor [DGOS93, SSOY98] such that this method is not adequate in applications in which a one-to-one correspondence with the original attractor is required.

Principal Component Analysis.

A principal component, singular-value [BK86] or Karhunen-Loève decomposition is based on computation of the *covariance matrix* with components

$$C_{ij} = \frac{1}{N - d_e + 1} \sum_{n=1}^{N-d_e+1} \mathbf{s}_{n-d_e+i}^T \mathbf{s}_{n-d_e+j}.$$

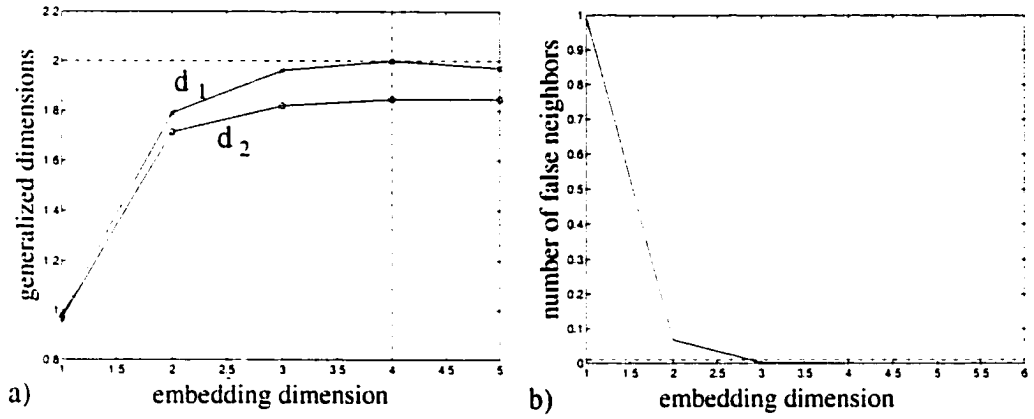


Figure 2.12: Estimation of embedding dimension. a) Generalized dimensions d_1 and d_2 for the embedded Lorenz data computed at different embedding dimensions. b) Number of false nearest neighbors as a function of the embedding dimension.

This matrix is symmetric with real eigenvalues. The size of the largest eigenvalue is characteristic for the dimension of the space capturing the important characteristics of the dynamics. Eigenvalues belonging to extra, unnecessary, dimensions capturing noise should all be equally small and form a so-called "noise-floor" [ABST93]. The number of eigenvectors not belonging to the noise-floor can be taken as embedding dimension. As pointed out in [ABST93] and [Cao97] this method is hard to interpret on occasion and the number of large singular values may depend on the details of the embedding and data as much as they do on the dynamics of the system.

False Nearest Neighbors

Today the most popular methods for detecting embedding dimensions rely on finding false nearest neighbors in phase space and attack directly the fundamental problem of finding embedding dimensions which unfold the attractor such that no false crossings of trajectories exist which are due only to an embedding into a too low dimensional space. Given a time delay T, an embedding vector in embedding

dimension d can be written as

$$\mathbf{s}_n(d) = [s(n),\ s(n+T),\ \cdots,\ s(n+(d-1)T)]^T.$$

A pair of nearest neighbors in dimension d is a pair of true neighbors if they are still neighbors in dimension $d+1$, otherwise the neighbors are called false neighbors. For a correct embedding dimension the number of false neighbors will be negligible such that this number can be used to detect correct embedding dimensions. Methods differ by the criterion used to detect false neighbors.

Let $\mathbf{s}_m(d)$ denote the (euclidean) nearest neighbor of $\mathbf{s}_n(d)$. The *false nearest neighbor* algorithm of Kennel [KBA92, AK93, KA95] detects a false neighbor by determining if the quantity

$$a_{FNN}(n,d) = \frac{|s(n+dT) - s(m+dT)|}{\|\mathbf{s}_n(d) - \mathbf{s}_m(d)\|}$$

is above a certain threshold. While this seems to be the most popular method, it is subjective to the choice of threshold [Cao97]. Cao [Cao97] proposed the measure

$$a_{Cao}(n,d) = \frac{\|\mathbf{s}_n(d+1) - \mathbf{s}_m(d+1)\|}{\|\mathbf{s}_n(d) - \mathbf{s}_m(d)\|}$$

and suggested to use the average measure

$$e1(d) = \frac{e(d+1)}{e(d)},\ \text{with}\ e(d) = \frac{1}{N}\sum_{n=1}^{N} a_{Cao}(n,d),$$

instead of a threshold parameter, to detect the embedding dimension. $e1$ will converge at a value of d which can be used as minimum embedding dimension. In Figure 2.12b we have shown the number of false nearest neighbors of the embedded Lorenz data as a function of the embedding dimension. We can conclude $d_e = 3$.

The false nearest neighbor algorithm returns embedding dimensions even for random data in certain cases [Cao97]. Cao proposed the additional measure

$$e2(d) = \frac{e^*(d+1)}{e^*(d)},\ e(d) = \frac{1}{N}\sum_{n=1}^{N} |s(n+dT) - s(m+dT)|$$

to distinguish deterministic from stochastic signals (tested for random colored noise). For random data $e2$ will always be equal to one, while for deterministic data $e2$ depends on d. This method was shown to be more efficient and less subjective than the false nearest neighbor algorithm.

2.9 Discrete Representation of Continuous Motion

In experimental situations, the measurement process observes the continuous dynamics only at discrete times τ. In simulations where a system is integrated, the time step of the integration process takes the role of the sampling time.[8]

The observation of a continuous process at discrete time intervals is a so-called *stroboscobic mapping* of the continuous system. This map can be expressed as

$$\mathbf{x}(t_{n+1}) = \mathbf{f}(\mathbf{x}(t_n)), \quad t_n = t_0 + n\tau, \quad n = 1, 2, \cdots .$$

Another type of map is the *Lorenz map* [Lor63] which was introduced by Lorenz to extract order from chaos. Given a continuous variable $x(t)$, the idea is to construct a map $x_{n+1} = f(x_n)$ which expresses the $(n+1)$th local maximum of x in terms of the nth local maximum. It is only useful to construct this map for nearly two-dimensional attractors [Str94].

Another possibility to introduce a discrete description is the so-called *Poincaré map* illustrated in Figure 2.13. This mapping is important because it reduces the dimension of the system by one and often allows a good interpretation of the continuous dynamics, especially close to periodic orbits. The idea is to record the continuous trajectory only if it intersects a certain surface-of-section Σ, which is generally a hyperplane of dimension $d_x - 1$.

---

[8] It is not recommended to use adaptive integration methods in this case since this would simulate a measurement process with varying sampling times and make computation of time delays more complicated.

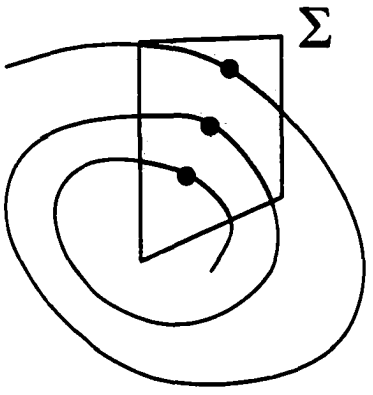


Figure 2.13: Illustration of a Poincaré map. The continuous trajectory is only recorded if it intersects a certain surface-of-section.

We require that Σ is transversal to the flow, $\mathbf{F}^T\mathbf{n} \neq 0$, where $\mathbf{n}$ denotes the surface normal of Σ. The Poincaré map $\mathbf{f}$ is a mapping from Σ to Σ obtained by following trajectories from one intersection with Σ to the next [Str94]. With $\mathbf{x}_n$ denoting the nth intersection with Σ, the Poincaré map is defined by

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

Nowadays virtually any discrete time system that is associated with an ordinary differential equation is referred to as a Poincaré map [Wig96] although the maps not necessarily are defined through a surface-of-section.

2.10 Conclusions

In this chapter we introduced the main concepts concerning chaotic systems necessary for the understanding of the following chapters. Of particular importance for the following is the fact that a chaotic attractor possesses unstable periodic orbits. If the continuous motion is viewed in a discrete space the periodic orbits correspond to fixed points. These fixed points can be detected by the So transformation. In many applications data is measured in the form of a time series and the attractor can be reconstructed by means of a time delay embedding. Although

we will not discuss such a reconstruction here further we included the topic of reconstruction of the phase space in this chapter for completeness.

Chapter 3

CHAOS CONTROL

> Instead of engineering to avoid chaos, as has traditionally been the custom, it may be more productive to design systems to operate chaotically and exploit the order contained within chaos.
>
> (D. Peak and M. Frame, 1994, p. 239)

We now turn to control and introduce the problem of chaos control, give an overview over existing methods, mention their advantages and weaknesses, demonstrate examples and motivate the chaos control method proposed in Chapter 6.

3.1 Introduction

Originally chaos was regarded as undesirable due to its intrinsic instability and unpredictability. In the engineering sciences systems were designed such that chaotic dynamics would not occur. A new approach was suggested by Ott, Grebogi and Yorke [OGY90] who proposed to control chaos by using the characteristics of chaos itself and showed that chaotic systems could be stabilized by applying small perturbations to a parameter. This idea of chaos control is fundamentally different from the engineering approach because the system stays in the chaotic regime during control but is stabilized at one of its unstable periodic orbits. Through small perturbations the undesired unpredictable dynamics are converted into desired predictable behaviour without requiring a fundamental change of operating conditions. Ott, Grebogi and Yorke's method, now known simply as OGY control, led to a rethinking of the role of chaos and to many interesting applications such as

control of atrial fibrillation [LBM+99, WMV+00] and epilepsy [GNN+96]. Today chaos is no longer considered a nuisance and applications focus on the exploitation of the characteristics of chaos: In many applications chaos can be considered desirable since it allows to produce an infinite number of periodic or non-periodic behaviours using the same chaotic system, merely by applying tiny, properly chosen perturbations to either a control parameter or a system variable.

An extension of these ideas has led to the more general idea of targeting general states on a chaotic attractor and to the idea of anti-chaoscontrol with the goal to sustain chaos. The observation that simple feedback mechanisms allow the synchronization of chaotic systems resulted in many applications, e.g., for secure communication [PC90]. Since 1990 more than 1000 papers appeared on the subject of chaos control (see [Che] for a bibliography with 800 references and [Kap96, BGL+00] for recent reviews). Today the focus has shifted from the control of low-dimensional systems to the control of high-dimensional, spatially extended systems exhibiting spatio-temporal or other complex behaviours such as multistability. The control of these systems will be the topic of Chapters 7 and 8.

In the next sections we introduce the chaos control problem, give a sample of experimental systems for which chaos control is of importance, and review the main approaches to chaos control, divided into non-feedback and feedback methods.

3.2 The Chaos Control Problem

We consider either a continuous system described by a nonlinear differential equation of the form

$$\mathbf{x}' = \mathbf{F}(\mathbf{x}, \mathbf{p}), \tag{3.1}$$

where $\mathbf{p}$ is a vector of control parameters, or a discrete system described by a map

$$\mathbf{x}_{n+1} = \mathbf{f}(\mathbf{x}_n, \mathbf{p}). \tag{3.2}$$

A discrete system representation can be obtained from the continuous system for example through a Poincaré map as discussed in Section 2.9.

The long-term behaviour of the solutions can be distinguished into regular (fixed points, periodic or quasiperiodic orbits) and irregular (chaotic) behaviour. Typically, by varying one or more parameters $p_i \in \mathbf{p}$, the system dynamics changes from regular to chaotic behaviour. We will assume throughout this dissertation that $\mathbf{p}$ is such that the unperturbed system behaves chaotically.

The problem of chaos control consists of changing the unpredictable chaotic behaviour into a desired predictable behaviour.

The obvious question to answer before attempting to find efficient chaos control methods is to ask, why chaos control is necessary? Is it not possible to simply design systems showing no chaotic dynamics? This is indeed not the case whenever a system is intrinsically chaotic, has to be operated in the chaotic regime to run efficiently, or is subject to unpredictable and uncontrollable influences which cause it to operate chaotically. In general, for systems with the possibility of nonlinear and complex behaviour chaos control can become an issue. We will give now a brief summary of experimental systems where chaos control is of great value.

- Stabilization of undesirable chaotic vibrations.

  In many systems which are subject to unpredictable external forces chaotic vibrations can occur which are harmful to the system. Examples are chatter of airplane wings, bridges subject to wind, buildings in earthquakes or ships subject to waves. Control of these undesired vibrations can greatly reduce stress on the system. Indeed one of the first experimental applications of the OGY method [DRS90] dealt with the control of chaotic vibrations in a mechanical ribbon.

- Increase of outputs of nonlinear systems.

  Complicated systems such as lasers often give maximum output and efficiency only if operated in a chaotic regime. Then, due to coupling of modes, undesired frequencies persist. Roy *et al.* [RMMG92] controlled the chaotic oscillations of a chaotic multi-mode laser and showed a 15-fold increase in power output if a chaos control method is applied. A related application is the synchronization of chaotic systems through chaos control. Besides greatly enhancing the efficiency of coupled systems such as coupled laser arrays [TJD+99] this led to applications to digital communication such as secret [CO93] and fast [WR98] transmission of data.

- Control of biological excitable systems.

  Chaos has been associated with the malfunctioning of certain biological systems such as the heart where it is believed that heart rhythms undergo transitions from periodic to chaotic behaviour during cardiac arrest. [1] On the other hand chaos seems to be an important ingredient for the efficient information processing capabilities of the brain. It is believed that during epileptic seizures a transition from chaotic to stable periodic behaviour occurs [2] which temporarily prevents effective information processing. Some of the first applications of chaos control methods concentrated on these medical applications which eventually could lead to the development of implantable medical devices. Garfinkel *et al.* applied chaos control first to a rabbit

---

[1] It is disputed if dynamics observed in biological systems are truly chaotic. Nonetheless, chaos control techniques have been succesfully applied to control of human atrial fibrillation [WMV+00]. Many chaos control technique can also be applied to a wider range of systems such as stochastic system [CC95].

[2] Or more general from a less ordered to a state of higher order.

heart [GSDW92] and later to human patients [LBM+99, WMV+00]. Control of brain activity based on chaos control techniques was first reported in [SJD+94] and later applied to the suppression of epileptic activity [GNN+96].

- Increasing mixing rates in fluids and efficiency of chemical reactions.

  The mixing rates of fluids and the efficiency of chemical reactions often are optimal if the system is stabilized at an UPO in the chaotic regime. The stabilization further allows better prediction of the outcome of the process. Peng *et al.* [PPS91] showed first the possibility of chaos control in chemical reactions while Singer *et al.* [SWB91] reported the first controlled experiment of a convective fluid.

- Targeting space-crafts to the moon.

  The three-body system consisting of the earth, the moon and a space-craft shows chaotic dynamics. As a result, a space-craft would eventually visit the neighborhood of the moon if left unperturbed. This offers the interesting opportunity to apply small perturbations at certain locations in space in order to target a craft to the moon (or to other planets) in a fast manner while not requiring large amounts of fuel. This targeting problem was numerically investigated in [BM95] and more recently in [SO97].

- Understanding of neural information processing.

  Much experimental evidence points to the theory that chaos is playing a fundamental role in neural information processing [SF87] and that it is of importance to understand how chaos is controlled in order to process, store and retrieve information. In this context, control mechanisms must be investigated which are based on ideas derived from artificial intelligence learning or biologically motivated neural network models. In particular reinforcement learning

is believed to play a role in the conditioning of neural connections [MDPS96]. This not only could lead to a better understanding of the brain and possible medical applications but also to powerful information processing devices. For investigations in this context see e.g. [SB93, LB94, Hof94, KPSJ97].

In summary there are two major motivations to study the control of chaos: 1) efficiently transform unpredictable chaotic into predictable periodic behaviour, and 2) design systems to operate chaotically and utilize the power of chaos: A chaotic system has a large number of readily accessible distinct states in the form of UPOs.

The methods of chaos control can be divided into non-feedback and feedback methods. Non-feedback methods apply state-independent perturbations (open-loop control) while feedback methods select perturbations based upon a knowledge of the state of the system (closed-loop control). To perturb the system we can either perturb a control parameter $p_i \in \mathbf{p}$ (parametric control) or perturb directly a system variable $x_i \in \mathbf{x}$ (state control). We can further distinguish between methods applying perturbations discrete in time or continuous in time. We will give an overview of existing approaches in the next sections. Keeping in mind that "Claiming that one method is superior to another would be senseless, much depends on the specific problem one has to tackle" [LP98] we will briefly summarize the main characteristics of the presented methods.

3.3 Non-feedback Chaos Control

Non-feedback methods exploit some property or knowledge of the system and change the controlled system for example by permanently shifting a control parameter or adding an external force such that the system behaviour changes from the chaotic attractor to a periodic state [Kap96]. The first approach consists of designing the chaos out of the system and we refer to it as *engineering chaos control*. The second approach generally considers external state-independent periodic or stochastic perturbations and is referred to as *taming chaos*.

3.3.1 Engineering Chaos Control

Engineering chaos control relies on a *bifurcation analysis* of the system to be controlled. Bifurcations are changes in the topological structure of the attracting set resulting from a change of one or more control parameters. A bifurcation analysis investigates the topological structure of the attracting set in dependence of a control parameter. Consider for example the logistic map.

$$x_{n+1} = px_n(1 - x_n),$$

with control parameter p. It is well known that the dynamics undergoes period-doubling bifurcations if p is changed from initially p=0 to p=4. This can be summarized in a bifurcation diagram as shown in Figure 3.1. The diagram shows x_{1000} as a function of p. For $p > p_\infty \approx 3.57$ the dynamics are chaotic. On the basis of the bifurcation diagram control is performed by setting system parameters such that the system exhibits the desired dynamics. For example if a period two dynamics of the logistic map is desired one could set $3 < p < 3.449...$.

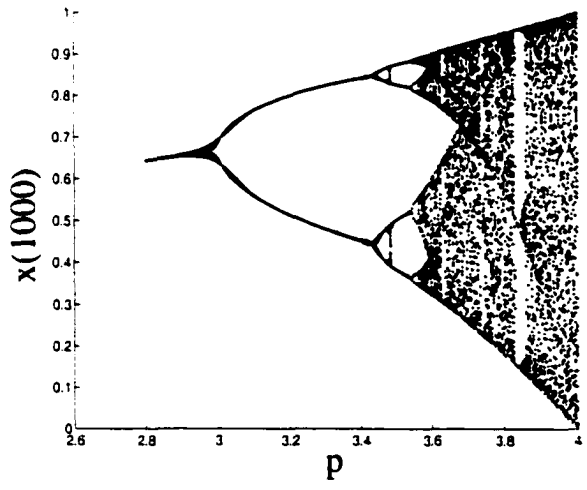
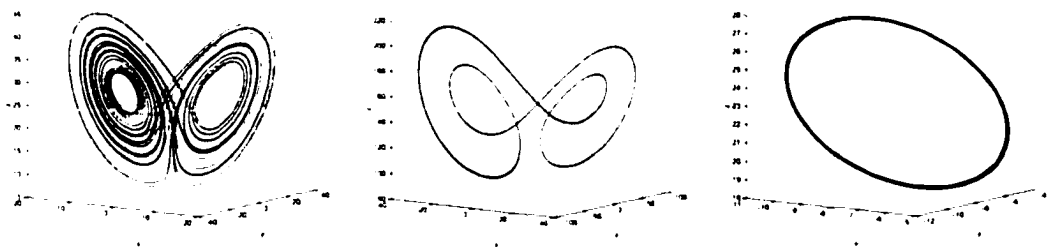


Figure 3.1: The bifurcation diagram of the logistic equation $x_{n+1} = px_n(1 - x_n)$. Shown is x_{1000} with x_0 randomly chosen from $[0, 1]$.

As a further example consider the well studied Lorenz system (2.2) which exhibits chaotic behaviour for $r = 28$ (see Figure 3.2a). A detailed analysis of the Lorenz system with r as control parameter shows that there are parameter windows where the system exhibits non-chaotic behaviour [Spa82]. For example for $r = 151.32$ the attractor is a 2-cycle (see Figure 3.2b). Also, since chaotic behaviour occurs only for $r > 24.74...$ the chaotic system can be changed into a regular one by choosing $r < 24.74...$ (see Figure 3.2c). One can observe that the engineering approach fundamentally changes the system since the controlled states are far from the original attractor, and that parameter changes generally have to be large to achieve control.



(a) $r = 28$ (b) $r = 151.32$ (c) $r = 24.1$

Figure 3.2: The Lorenz attractor $x' = 10(y - x)$, $y' = rx - xz$, $z' = -(8/3)z + xy$.

3.3.2 Taming Chaos

Taming chaos refers to the control of chaos through state-independent periodic perturbations or the addition of random noise. In principle this results in a modification of the original system (3.1) into the system

$$\mathbf{x}'(t) = \mathbf{F}\big(\mathbf{x}(t), \mathbf{p}_0 + \mathbf{u}_1(\omega_1, \gamma_1, t), \rho(t)\big) + \mathbf{u}_2(\omega_2, \gamma_2, t). \tag{3.3}$$

with perturbation $\mathbf{u}_1$ of a system parameter and/or external force $\mathbf{u}_2$.

For certain values of the parameters ω_i, γ_i the system can show periodic or other desired motion. Adequate values for the parameters can be found through a systematic stability analysis of the system with the additional parameters ω_i, γ_i or through trial-and-error. Many different mechanisms of this type of non-feedback control have been suggested: periodic perturbation of an internal system parameter [KAS87, KRB94, CB96, PKC98], external periodic perturbation [BG91, USOK98, SCH99] or external stochastic perturbation (chaos control through noise) [Her88]. These approaches, however, are in general limited by the fact that their action is not goal oriented and the final state cannot easily be decided by the operator [BGL+00]. If an analytical description of the goal state is available *entrainment* techniques [Bre94, Jac97] can be applied where the desired goal dynamics are used as external perturbation to a system variable in order to drive the system to this input state. This open-loop control is related to the Pyragas-type closed-loop methods which we will discuss in Subsection 3.4.3.

3.3.3 Characteristics of Non-feedback Methods

The main advantage of non-feedback techniques is their speed. Once adequate control parameters for either the perturbations or the system parameters are found, control can be achieved fast since no on-line monitoring of the system dynamics or processing of a control policy is necessary. These methods are thus promising for controlling fast systems such as electro-optical systems or superconducting Josephson junctions [CB96]. On the other hand these methods do not use the characteristics of chaos for control but create a new system with the desired properties. The stabilized orbit does generally not exist for the unperturbed system but in many applications this is undesirable or even impossible to realize since for example only small changes to system parameters are allowed. Furthermore, if trial-and-error methods do not find suitable parameters an often extensive analysis is necessary in order to establish control.

3.4 Feedback Methods

Feedback methods select a perturbation based upon knowledge of the state of the system. In many situation the dynamical equations describing a system are unknown or inaccurate and control methods will only be useful if they are *model independent*, i.e., do not require analytical knowledge on system dynamics. Instead, a learning phase is applied in which the knowledge necessary to choose successful perturbations is extracted from observations of the system dynamics. The goal in practically all feedback chaos control methods is to stabilize the system at one of the embedded UPOs through small perturbations. We will distinguish parametric feedback methods which perturb a certain system parameter and impulsive methods which perturb directly a system variable. These methods are typically time-discrete control methods and apply control signals at discrete times, for example, when the system crosses a surface-of-section. The third category of feedback method are time-continuous methods. In addition, we will review two important improvements of parametric control methods: *targeting*, which refers to the general problem of bringing a system from any initial state as fast as possible to a prescribed desired state, and *tracking*, which refers to methods able to stabilize a system in a non-stationary environment.

In the ideal case of feedback chaos control aiming for stabilization of embedded UPOs, and in a noise free environment, small perturbations suffice and the applied perturbation will vanish once the desired state is reached. The possibility to control the system through small perturbations is one of the major advantages of feedback chaos control methods. For example, in medical applications concerning control of heart arrhythmia, the control through small, almost vanishing, perturbations offers the possibility of far less invasive control than common pacing methods [CC97b].

3.4.1 Parametric Chaos Control

The OGY method

In the case of chaotic behaviour, the attractor of (3.1) is a *strange attractor* and has embedded within a dense set of UPOs (see Section 2.3). The chaotic dynamics can be seen as *shadowing* periodic behaviour through certain time intervals [BGL+00]. Chaotic dynamics are further characterized by a sensitive dependence on initial conditions leading to exponential divergence of initially close trajectories and an unpredictability of the system. This can be interpreted differently: Due to the sensitive dependence, a tiny perturbation will give rise to a large response over time. Ott, Grebogi and Yorke suggested in a seminal paper [OGY90] to exploit this sensitivity in order to stabilize a chaotic system at one of its embedded UPOs through small, properly chosen perturbations and thus revealing the shadowed periodic behaviour. The main advantage of this type of control is that it exploits the characteristics of chaos and does not fundamentally change the controlled system. Once the desired state is reached the perturbations almost vanish in the ideal case and small perturbations allow the switching between a large arsenal of dynamics provided by the infinite number of UPOs.

The OGY Control Formula

The OGY method [OGY90] applies small time-dependent perturbations Δp to a system parameter p_0 in the neighborhood of a previously extracted UPO. The extraction of UPOs from experimental data was discussed in Section 2.6. The perturbation is chosen by means of a reconstruction of the local linear properties of the dynamics around the desired UPO. If no analytical knowledge is available, experimental data can be used to extract the necessary information. The OGY method is therefore model independent. Assuming that the dynamics can be described by

a two-dimensional map,

$$\mathbf{x}_n = f(\mathbf{x}_n, p_0 + \Delta p_n), \quad \mathbf{x} \in \mathbb{R}^2, p_0 \in \mathbb{R},$$

the following *OGY formula* can be derived for the perturbation Δp_n of a system parameter p_0 [OGY90, BGL+00]:

$$\Delta p_n = \frac{\lambda_u \mathbf{f}_u}{(\lambda_u - 1)\mathbf{f}_u \cdot \mathbf{g}} (\mathbf{x}_n - \mathbf{x}_f(p_0)). \tag{3.4}$$

In (3.4) $\mathbf{x}_f(p_0)$ denotes the location of the desired fixed point of the unperturbed system. The *gain* $\mathbf{g}$ describes how $\mathbf{x}_f$ changes with p,

$$\mathbf{x}_f(p) \approx \mathbf{x}_f(p_0) + \mathbf{g}\,\Delta p, \text{ such that } \mathbf{g} \approx \frac{\mathbf{x}_f(p) - \mathbf{x}_f(p_0)}{\Delta p}.$$

λ_u denotes the unstable eigenvalue associated with the unstable eigenvector $\mathbf{e}_u$ of the Jacobian matrix $\mathbf{J}(\mathbf{x}_f(p_0))$ of the map f evaluated at $\mathbf{x}_f(p_0)$. $\mathbf{e}_s$ represents the stable eigenvector. $\mathbf{f}_u$ denotes the adjoint unstable eigenvector defined through $< \mathbf{f}_u, \mathbf{e}_u >= 1$. $< \mathbf{f}_u, \mathbf{e}_s >= 0$. It is important to note that all necessary quantities can be obtained from the observation of experimental data. The idea of the OGY method can be understood from Figure 3.3. Without control a trajectory would approach the fixed point along the stable direction and depart from it along the unstable direction (Figure 3.3a). If the parameter is perturbed according to the OGY formula, the stable direction is slightly shifted such that the trajectory approaches the stable direction of the unperturbed system (Figure 3.3b). In the ideal case, after perturbation, the state of the system lies exactly on the stable direction of the unperturbed system and without requiring further perturbations, the dynamics will approach the desired fixed point (Figure 3.3c).

The OGY control scheme is only local, i.e. control is only turned on after the state of the system, driven by the unperturbed dynamics, has reached a neighborhood, C, of the desired UPO. Due to the chaotic nature of the attractor containing

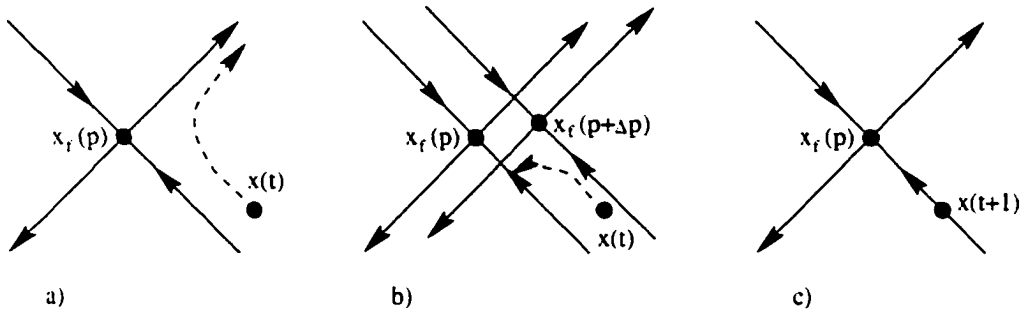


Figure 3.3: The idea of the OGY method. See text for details.

the UPOs it is guaranteed that every trajectory eventually enters C. The region C in which the OGY control formula is valid can be computed from the local properties of the dynamics around the fixed point as shown in [OGY90]. The local character of the control scheme results in an initial waiting period before control can be turned on. This time is called *transient time* τ. Letting $\mu(C)$ denote the rate at which random orbits fall into the region C, it follows that the average transient time is $\langle\tau\rangle = 1/\mu(C)$ [BGL+00]. The size of the control region C depends for the OGY method on the maximum allowed perturbation δ ($|\Delta p| < \delta$) and the local dynamics of $\mathbf{f}$ in C. For one-dimensional maps one can show $\langle\tau_{OGY}\rangle \sim \delta^{-1}$ and for two-dimensional maps [BGL+00]

$$\langle\tau_{OGY}\rangle \sim \delta^{-\gamma}, \quad \gamma = 1 + \frac{\ln|\lambda_u|}{2\ln(1/|\lambda_s|)}.$$

For example for the logistic map $x_{n+1} = (p_0 + \Delta p)x_n(1 - x_n)$ with $p_0 = 3.8$ and $\delta = 0.1$ we would expect $\langle\tau\rangle \sim 100$. For the Hénon map $x_{n+1} = y_n + 1 - 1.29x_n^2$, $y_{n+1} = 0.3x_n$ at the fixed point (0.8385.... 0.8385...) we would expect $\langle\tau\rangle \sim \delta^{-2.225}$, or $\langle\tau\rangle \sim 3669$ for $\delta = 0.025$. The transients can become quite long and represent a problem which can be overcome by global control methods or targeting procedures (see Subsection 3.4.2).

Characteristics of the OGY Method

The main advantage of the OGY control scheme is that it is model independent. All necessary quantities describing the local dynamics close to the desired state can be approximated from experimental data. However, to compute the control formula, the desired state must be determined a priori from experimental data. Any accessible system parameter can be used as control parameter.

Given an accurate control formula, small noise can destabilize the chaotic orbit and induce occasional chaotic bursts as discussed in [OGY90, RGOD91]. In real applications, the determination of the desired UPO and in particular the estimation of the linearized dynamics will become less accurate if noise is present. Thus the influence of noise is twofold: it can kick dynamics out of the controllable region and it affects the accuracy of the control formula. In attempting to control chaos in a real electronic device the method was found to be sensitive to the inevitable noise of the circuit elements and OGY control was impossible [Kap96]. In view of this, we will distinguish in Chapters 6 through 8 two different types of control in noisy environments: 1) We establish the control policy in the noise free environment and investigate its quality under the influence of noise. 2) We establish the control policy in the noisy environment.

To apply the OGY method, the location of the fixed point has to be estimated in advance. In the case of non-stationarity this location will shift and the OGY control formula will become inaccurate. At some point a recomputation of the required quantities will become necessary. A solution to this problem is provided by *tracking* methods which will be discussed among other improvements in the next paragraph.

As mentioned above, the local character of the OGY method can result in long transient times. One solution to overcome this problem is to use nonlinear

manifold approximations (see the next paragraph for references). The more general solution to this problem through targeting is discussed in Subsection 3.4.2.

Despite its drawbacks, the OGY method remains one of the most popular chaos control method and was successfully applied to many experimental systems such as a magneto-elastic ribbon [DRS90], a rabbit heart [GSDW92], a driven pendulum [HDM+94], a driven glow discharge [WKW95] and a rat brain [SJD+94].

Modifications of the OGY Method

Many extensions and improvements of the OGY method have been proposed. A real-time variant for effectively one-dimensional systems was proposed in [CC97b] and the case of multi-parameter control was investigated in [BG95]. The application of the OGY control method to experimental data reconstructed from a time series using time delay embeddings was discussed in [DN92, SO95]. The OGY formula has to be modified in order to be applicable to this type of data. The extension to high-dimensional systems was the topic of [RGOD91, HJJY96, DYI+96, GL97] and is known as *pole placement*. To enlarge the size of the controllable region it was suggested to apply perturbations with varying magnitude [ED98] and to use nonlinear approximations for the quantities in the OGY formula [YU98]. Transients can be further reduced by targeting methods (see Subsection 3.4.2). To cope with relatively large noise levels, it was suggested to use multiple sections and to adjust system parameters more frequently [BX96].

A further popular variant of the OGY method applies perturbations of the form [SBGS97] $\Delta p_i = -k(x_i - x_f(p_0))\theta(\tau_i)$, where $\theta(\tau_i)$ denotes a square wave function of duration τ_i. This *occasional proportional feedback* (OPF) method [Hun91] and variants [CS94, SN96] feed deviations of the chaotic variable from an a priori fixed point back into the system as parameter perturbation whenever the system variable falls within a specified window. The free parameters such as τ and γ or additional parameters in the variants typically are adjusted experimentally by varying

them over wide ranges until desired behaviours are observed. The method is particularly popular for experimental systems where the parameters can be changed fast such as lasers and diode resonators [Hun91, NAGK94, CS95, SN96]. It is rather surprising that this method often works well even for high dimensional systems although typically only a single state variable is used as feedback signal. However, due to the fact that parameter perturbations are allowed to be large, stabilized orbits do not necessarily exist for the unperturbed system. It has further similarities to non-feedback methods in that the desired orbit can not be chosen beforehand [KS99].

A drawback of the OGY scheme in the case of non-stationary systems is that it requires the location of the desired UPO to be known in advance. In non-stationary environments this location will change and the OGY formula will become invalid, making a recomputation of the relevant parameters necessary. The problem of stabilizing an UPO in a non-stationary environment is generally called *tracking control of UPOs*. A controller able to track UPOs must have the ability to estimate locations of fixed points and update its control formula on-line as data is received. Schwartz *et al.* [CTSP92, ST92, SCT97] suggested a prediction-correction scheme based on the OGY method to continuously estimate the error in the fixed point which is proportional to the mean in the fluctuations of Δp. Minimizing the error as a function of the fixed point location gives predicted new estimates of it and allows to correct the control formula correspondingly. Bielawski *et al.* [BDG93] modified the control formula directly to use the difference between consecutive iterates of the map instead of the difference between each iterate and the fixed point as in the OGY formula. This makes the method independent of the fixed point location and allows to stabilize the fixed point if parameters change slowly. This method can be considered as a discrete parameter version of the Pyragas scheme, which we will review in Subsection 3.4.3. In a later work Bielawski *et al.* [BDG94] extended

the time-discrete version to time-continuous parameter perturbations. Both the discrete and the continuous version of these authors were only demonstrated for systems with an effectively one-dimensional Poincaré map and for the control of period one fixed points. It is questionable that this approach will be successful for more general cases.

For completeness we also want to mention *adaptive parameter control* methods [SG98, PKK98]. Assuming the system with control parameter p shows chaotic behaviour for $p = p_0$, the parameter p_0 is updated according to the rule $p_{n+1} = p_n + \gamma(x_f - x_n)$. The result of this type of control is a permanent shift of the parameter p_0 to a value p_{lim}, for which the system naturally shows the desired behaviour. Thus these type of methods effectively achieve engineering control of chaos and do not follow the OGY philosophy of chaos control through small perturbations. The adaptive parameter methods play an important role for the control of spatially extended systems (see Subsection 7.3.1) since they are easier to realize than OGY type methods.

We summarize with the words of Bayly and Virgin [BV94]: "The control scheme devised by Ott, Grebogi, and Yorke, and modified by subsequent authors, is a remarkable application of nonlinear dynamical systems theory. In practice, current versions will work better on systems whose Poincaré sections are nearly one-dimensional, with weakly nonlinear and weakly unstable local dynamics."

3.4.2 Targeting

The problem of targeting is twofold. The first problem is that of steering a system from a given initial point to a desired target point in the smallest possible time by applying only small perturbations to the system. Related is the problem of targeting a certain attractor in a system possessing multiple coexisting attractors.

Different approaches to targeting have been suggested based on the idea of applying small perturbations to switch from long to short trajectories. Shinbrot

et al. [SOGY90] applied one initial perturbation to place the system from an initially long trajectory to a neighboring short one. Bollt *et al.* and Schroer [SO97] generalized this *pass targeting* method by allowing controls in general switching regions which connect short trajectories to the target (see Figure 3.4).

A different approach was suggested by Kostelich *et al.* [KGOY93] and Baptista *et al.* [BC98] who allowed small perturbations at each iteration to target the fixed point along a hierarchy of paths leading to the target. This *tree targeting* can be understood by going backwards from the target state $\mathbf{x}_f$ as shown in Figure 3.4. Assume that we are targeting a control region C around $\mathbf{x}_f$. Iterating this region back using the unperturbed dynamics we will get the light gray regions labeled C_1 shown in Figure 3.4. Allowing small perturbations if the system is close to these regions and designed such that the system targets C, the regions C_1 get effectively larger as illustrated by the dark regions in Figure 3.4. Extending this over many iterations a global tree can be constructed through which the target state can be reached quickly. Kostelich *et al.* allowed perturbations from a continuous set while Baptista *et al.* restricted them to a discrete set which makes the method computationally more effective and more realistic for some real systems where an exact perturbation can not be achieved [BC98].

A method which is in principle similar to the Kostelich approach was introduced by Funke *et al.* [FHD97] who generate a set of control neurons through a Kohonen self-organizing network which targets a previously estimated fixed point along a quasi-optimal path. In addition to targeting their method was able to stabilize and extract a fixed point without requiring analytical knowledge on system dynamics.

Targeting and Optimal Control

In theory the optimal solution to the targeting problem can be found by formulating the problem in terms of stochastic optimal control theory [Ber87]: find

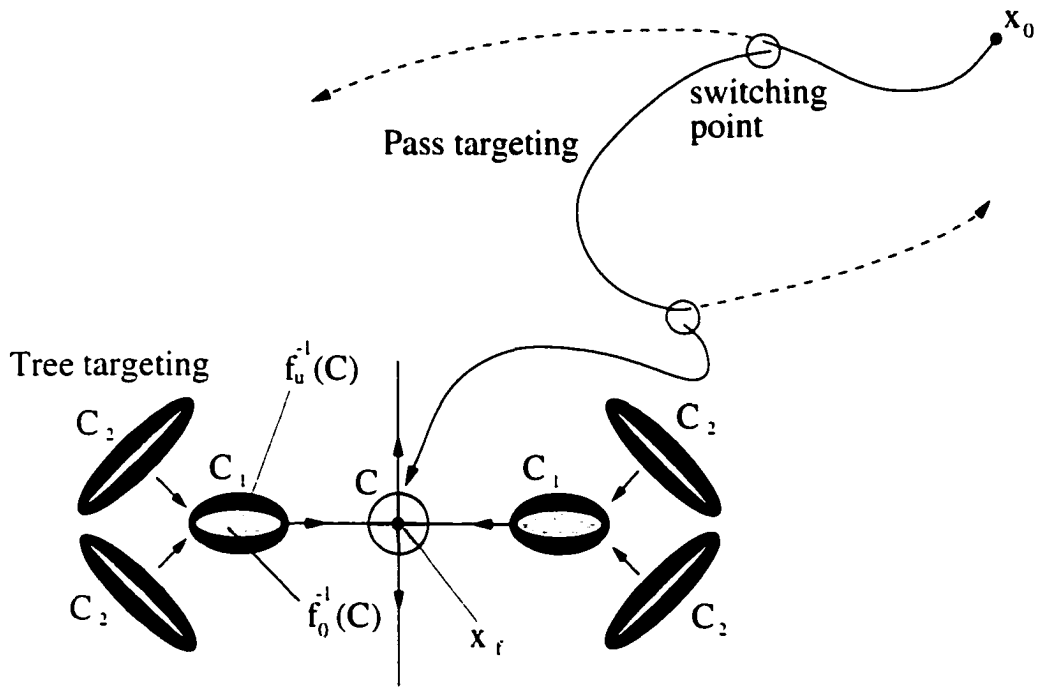


Figure 3.4: Two approaches to targeting: pass targeting and tree targeting. See text for details.

an optimal control policy $\pi(\mathbf{x})$ on the basis of which state dependent perturbations $u(\mathbf{x})$ are chosen. The optimal policy is such that an error functional or performance index is minimized. The goal of targeting is to minimize the distance between $\mathbf{x}_n$ and a target $\mathbf{x}_f$ such that $\|\mathbf{x}_n - \mathbf{x}_f\| < \epsilon$ after a minimum number of iterations n from any initial condition $\mathbf{x}_0$ by applying small maximal parameter or system state perturbations. This optimality approach was followed by several authors [PMT95, BRR95, FV97] by introducing suitable cost functionals and applying optimal control theory methods with Lagrangian relaxation schemes. While finding the optimal control policy, the drawback of the methods [PMT95, BRR95, FV97] is that they generally require accurate analytical knowledge on the system dynamics. In some cases this knowledge can be approximated through local models [Hun98] to desired accuracy but in general the exact approximation of the dynamics of nonlinear systems is not straightforward and can be computationally and data

expensive. Even if analytical knowledge is available the optimal control methods are often mathematically complicated, computationally expensive and difficult to implement.

Stochastic optimal control suffers from two major problems [BT96]: Bellman's *curse of dimensionality* (computational requirements increase exponentially with the dimension of the problem) and the *curse of modeling* (an explicit system model is needed). The field of reinforcement learning, also known as neuro-dynamic programming, claims to deal effectively with these two problems by 1) computing an approximation of the policy, and 2) using simulation to receive reinforcements on the success of controls [BT96]. We will give a basic review over the field of reinforcement learning in Chapter 4.

The idea to use reinforcement learning to control a chaotic system was first suggested by Der *et al.* [DH94] who applied the method to target a known fixed point of the logistic map. We generalized their approach such that it solves the tracking problem and allows for perturbations of either parameter or state variables, and applied it to different systems including a hyperchaotic system [GD99], a coupled map lattice [GD00a] and a multistable rotor [GD00b]. These results, extensions and background material are the topic of the following chapters.

Transient Time of Targeting Methods

Let C_t denote the size (diameter) of a small target region on an attractor. Without targeting the average transient time to naturally reach this region is given by [KGOY93] $\tau \sim C_t^{-d_L}$, where d_L denotes the Lyapunov dimension of the attractor defined as $d_L = 1 + \sigma_u/|\sigma_s|$ in two dimensions (see [AFH94] for the general definition), with σ_u and σ_s being the positive and negative Lyapunov exponents, respectively. In any case we have $d_L \geq 1$ such that a lower bound for the average transient time without targeting is given by $\tau_u \sim 1/C_t$ for $C_t \leq 1$.

How does targeting improve the transient time? Assume that we apply a control signal already at the start of the systems iteration and that the initial condition is located in a small phase space region of size C_0.[3] Iterating forward in time, the size of this region will increase at exponential rate $C(\tau) \sim C_0 e^{\sigma_u \tau}$ along the unstable direction, corresponding to σ_u, the largest positive Lyapunov exponent of the system. If $C(\tau)$ has reached the size of the attractor ($C(\tau) \sim 1$), then it is safe to assume that an initial condition from C_0 will have entered the finite target region C_t. The task of most targeting methods is to find the shortest such trajectory. From $C(\tau_t) \sim 1$ we can compute the transient τ_t for a targeting method as $\tau_t \sim (\ln C_0^{-1})/\sigma_u$. Assuming that $C_0 \approx C_t$ we obtain the estimate

$$\tau_t \sim \frac{\ln \tau_u}{\sigma_u}. \tag{3.5}$$

Different estimates specific to the particular targeting methods where derived in [SOGY90, SO97] but the important result is that targeting results in a logarithmic shortening of transient times.

Targeting methods can be considered as global chaos control methods if they can stabilize the desired state in addition to bringing dynamics globally close to the neighborhood of this state. Local chaos control methods make use of the saddle-type character of the fixed points corresponding to UPOs to trap the trajectory in the neighborhood of the desired fixed point. Global methods exploit the exponential divergence of trajectories and achieve a logarithmic shortening of transient times.

In Chapter 6 we will provide theoretical and numerical justification that the transient time of our chaos control method based on reinforcement learning scales similar to (3.5).

---

[3]The size of the initial element depends on the local dynamics at the initial condition.

3.4.3 Time-continuous Additive Chaos Control

Experimental situations may arise in which no control parameter is easily accessible. For this case, Pyragas [Pyr92] developed a chaos control method applying self-controlling feedbacks directly to a system variable.

Consider a dynamical system $\mathbf{x}'(t) = \mathbf{F}(\mathbf{x})$ where some scalar variable $y \in \mathbf{x}$ can be measured as a system output. Furthermore, suppose that the system has an input $g(t)$ available as external force which only acts on y. We can write the system in the form

$$\begin{pmatrix} y(t) \\ \hat{\mathbf{x}}(t) \end{pmatrix}' = \begin{pmatrix} F_1(y, \hat{\mathbf{x}}) + g(t) \\ \hat{\mathbf{f}}(y, \hat{\mathbf{x}}) \end{pmatrix}.$$

For $g \equiv 0$ the system is assumed to behave chaotically. In the taming chaos methods mentioned in Section 3.3.2 the external force $g(t)$ is chosen independently from the current state of the system, thus the control is generally not goal oriented and non-vanishing. In entrainment control $g(t)$ corresponds to a general state of the system. Pyragas [Pyr92] suggested to apply state dependent feedback signals and introduced two forms for $g(t)$:

1. External force control: $g(t) = \gamma\,[y_i(t) - y(t)]$.

   Here $y_i(t)$ is a signal from an external oscillator which is proportional to the signal of a desired UPO. This signal has to be extracted in advance, which can be done from experimental data as discussed in Section 2.6. γ is a positive, experimentally adjustable weight of the perturbation.

2. Time-delayed feedback: $g(t) = \gamma\,[y(t - T) - y(t)]$.

   Here T is the period of the desired UPO and has to be determined in advance.

In both cases, the effect of $g(t)$ is such that if $g(t) > 0$ (< 0), then $y'(t)$ is increased (decreased) with the result that y approaches the desired UPO specified either by a

complete example of the UPO (external force control) or by its period (time-delayed feedback). Eventually the system will be in phase with the desired UPO and g will vanish. The stabilization in both methods is achieved through additional degrees of freedom introduced in the system by the perturbation which results in changes of the Lyapunov exponents of the UPO such that they become negative and the UPO is stabilized [Pyr92]. An example of external force control of the period one UPO of the Rössler attractor is shown in Figure 3.5. The external force method uses complete information of the desired UPO and is consequently more efficient and robust than the time-delayed feedback method. The time-delayed feedback approach on the other hand is much easier to implement in certain, especially electronic, systems. More general forcing functions $g(t)$ have been suggested such as nonlinear feedbacks with delay [dSVL96], feedbacks incorporating information from farther in the past [SBGS97] and half-period delayed feedbacks suitable for systems with symmetries [NU98].

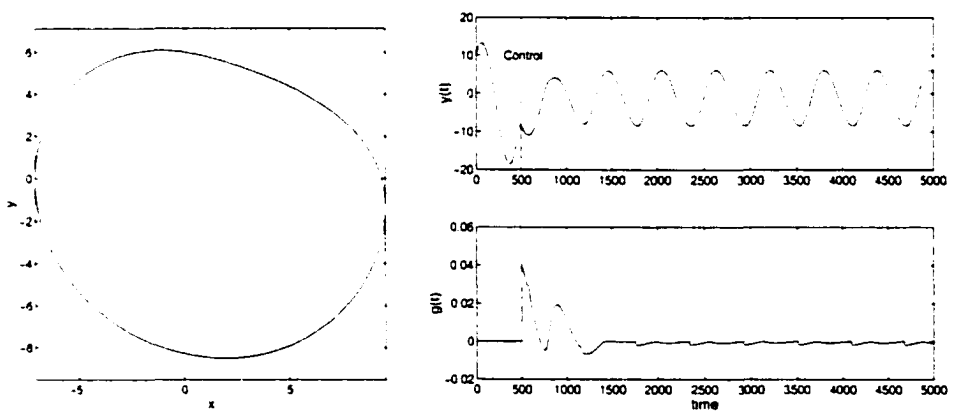


Figure 3.5: Control of the Rössler attractor $x' = -y - z$, $y' = x + 0.2y + g(t)$, $z' = 0.2 + z(x - 5.7)$, through external force control with $g(t) = k(y_i - y(t))$ and $k = 0.02$. a) The stabilized period one orbit. b) The variable $y(t)$ (upper graph) and the force $g(t)$ (lower graph).

More general, the dynamical system controlled by time-continuous feedback perturbation of the state can be written in the form

$$\mathbf{x}' = \mathbf{F}(\mathbf{x}) + \mathbf{K}\,(\mathbf{g}(t) - \mathbf{x}), \tag{3.6}$$

where $\mathbf{K}$ is called the *feedback matrix*, and $\mathbf{g}(t)$ denotes any solution on the attractor (e.g. an UPO or a time-delayed trajectory or more general solutions) and can typically be found from analytical computation or experimental observation. Thus the major task lies in the correct computation of $\mathbf{K}$. $\mathbf{K}$ has to be chosen such that the desired solution $\mathbf{g}(t)$ is stable. Barrett [Bar96] showed the important result that for every solution $\mathbf{g}(t)$ of the unperturbed system a constant matrix $\mathbf{K}$ exists such that $\mathbf{g}(t)$ becomes an asymptotically stable solution of (3.6). Furthermore he gave a method of finding $\mathbf{K}$ which involves solving a constrained optimization problem and relies on a priori knowledge of the governing system equations $\mathbf{F}$. If this analytical knowledge is not available (which is the case we focus on in this work) then a *controllability range* for successful $\mathbf{K}$ must be estimated experimentally. This can be done systematically by observing the dominant largest Lyapunov exponent as a function of the parameters of $\mathbf{K}$ or simply through trial-and-error. Obviously, if more than one variable is perturbed such that $\mathbf{K}$ is not a scalar, then this stability analysis can become quite expensive.

A recent improvement [AB97] offers an adaptive solution to this problem for the time-delayed version. Writing the controlled system in the form

$$\mathbf{x}' = \mathbf{F}(\mathbf{x}) + \mathbf{g}(t),$$

each component of $\mathbf{g}$ is now given by

$$g_i(t) = \gamma_i(t)(x_i(t - T) - x_i(t)).$$

The adaptive feedback parameters γ_i are updated according to

$$\gamma_i(t_n) = \frac{\gamma_i(t_{n-1})}{1 - \tanh(\sigma\lambda_i(t_n))}.$$

with

$$\lambda_i(t_n) = \gamma_i(t_{n-1}) \log \left| \frac{x_i(t_n - T) - x_i(t_n)}{x_i(t_{n-1} - T) - x_i(t_{n-1})} \right|,$$

each time t_n a new observation arrives. The sensitivity $\sigma > 0$ prevents γ from going to infinity and has to be chosen beforehand. The strength of the perturbation now depends adaptively on the local dynamics of the system.

Characteristics of Pyragas-type Control Methods

Certainly the greatest advantages of the time-continuous control methods as suggested by Pyragas is that they do not require any computational effort once the UPO or its period is extracted from the system and adequate feedback parameters are found. Since the control is applied continuously, stabilization of the desired state can be achieved almost instantly. However the stability analysis required for determining the feedback gain is generally a difficult task. The adaptive method [AB97] was designed to choose the feedback parameters adaptively but has yet to be verified experimentally. In general, the Pyragas-type control method is quite sensitive to the parameters $\mathbf{K}$ and $\mathbf{g}(t)$. If they are not sufficiently exact control often results in non-vanishing controls and a stabilized orbit which does not exist for the uncontrolled system.

The Pyragas method has been experimentally applied to many different systems including a laser system [BDG94] and a fast diode resonator [SBGS97]. Due to its practical simplicity the Pyragas method has become popular for the control of coupled or spatially extended systems (see Chapter 7 for references). In situations which allow to perturb a system variable continuously in time there is currently probably no better chaos control method available. In a recent paper Tsui and Jones [TJ00] concluded that "a rigorous analysis of Pyragas's method is long overdue". We want to stress that the control method we are proposing in this dissertation must be viewed as a time-discrete control method and can therefore not be compared to time-continuous control methods.

3.4.4 Time-discrete Additive Chaos Control

In certain experimental situations the only possible types of perturbations are time-discrete perturbations of a state variable.

Then, as first option, a discrete version of the Pyragas control can be applied to a system variable, e.g. when a trajectory hits a Poincaré section. For the control of a fixed point of period p, the discrete version of the continuous time-delayed feedback perturbation is given by [Gad98]

$$\mathbf{x}_{n+1} = \mathbf{f}(\mathbf{x}_n) + \mathbf{K}\,(\mathbf{x}_n - \mathbf{x}_{n-p}).$$

This a popular approach for the control of coupled map lattices (see Chapter 7 for references).

An approach similar to a discrete version of Pyragas external force control, but derived from the OGY method was suggested by Garfinkel *et al.* [GSDW92]. They attempted to control an irregular cardiac rhythm in a rabbit heart. No system parameter was readily accessible and they developed a method now known as *proportional perturbation feedback* which forces the system's state by placing it directly onto the stable manifold of the desired UPO. In principle this is similar to the mechanics of a pacemaker. The important difference is that the perturbations can be applied much less frequent since they are only necessary if the state leaves the neighborhood of the UPO. A computationally easier variant was introduced by Christini and Collins [CC97a]. The ultimate goal of these investigations are implantable devices for the control of heart arrhythmia [GSDW92, CC96, LBM+99, WLL+98, WMV+00] or other diseases like epilepsy [SJD+94, GNN+96].

A different type of system perturbation was proposed by Güémez and Matías [GM93] who suggested to stabilize one-dimensional maps by a series of proportional feedbacks.

$$x_n \to x_n(1+\gamma), \tag{3.7}$$

on a system variable, performed every Δn iterations. Δn is related to the period of the stabilized orbit and γ has to be found experimentally. The method was extended to continuous systems in [MG94, MG96, Cha98]. The authors point out similarities to non-feedback perturbation methods (see Subsection 3.3.2) since the control signal is often non-vanishing and the stabilized orbit does not necessarily exist in the uncontrolled system. The result of the proportional control (3.7) is that a different dynamical system with the additional parameters γ and Δn is created. In practice, fixing Δn and varying γ, a large number of orbits can be stabilized and a stability analysis can be performed to find good values for γ [Cha97]. Güémez and Matías' method is a multiplicative *impulsive control method*. Another possibility are additive impulse generating laws [YYY97, ZS97]. The impulsive control methods have the advantage in common with the taming chaos methods that no knowledge on system dynamics or goal state is necessary. Successful control can be established simply by trial-and-error. Furthermore since control is applied to state variables instead of system parameters this method is popular for the control of biological or chemical systems where control parameters are not readily identified or perturbed and time-continuous control is infeasible [MG96].

3.5 Artificial Intelligence Chaos Control

In this section we will review chaos control methods which are based on *artificial intelligence* (AI) *learning algorithms* such as multilayer perceptrons, genetic algorithms, radial basis functions, reinforcement learning or other algorithms generally designed to perform "cognitive tasks, at which humans are currently better" [Hay99]. These methods are generally data-based which allows them to learn desired outputs from given inputs without analytical knowledge on system dynamics.

3.5.1 Local Control: Backpropagation Multilayer Perceptrons

A multilayer perceptron trained with the back-propagation algorithm (BMLP) can be viewed as a general function approximator [Hay99]. Given an unknown mapping $\mathbf{y} = \mathbf{f}(\mathbf{x})$ and feeding the BMLP with input-output pairs $[\mathbf{y}, \mathbf{x}]$ in theory any mapping $\mathbf{f}$ can be approximated to any desired accuracy. In practice we will always have a finite set of (possibly noisy) data. This leads to the problem of over-fitting which typically is avoided by cross-validation [Hay99]. In any case, this type of neural network can be considered an universal function approximator which uses supervised learning with given input-output pairs.

Maybe the first chaos control application of a BMLP and an artificial learning method in general was demonstrated by Alsing *et al.* [AGK94]. The OGY control formula can be considered a mapping

$$\Delta p = \mathbf{g}(\mathbf{x}_n, \mathbf{x}_f).$$

Alsing *et al.* approximated this mapping by presenting input-output pairs $[\mathbf{x}_n, \mathbf{x}_{n-1}]$ and using the difference between Δp produced from the BMLP and Δp_{OGY}, the optimal one produced by an OGY controller, as error feedback signal. If this error is small, the BMLP has approximated the OGY control formula. A similar method was suggested by Mulpur *et al.* [MMJ95] where a feedback controller was applied as teacher for the BMLP. While demonstrating that neural networks can control chaotic systems these methods are not of much practical interest since they require an existing chaos controller as supervising teacher. Because of this deficiency Otawara and Fan [OF95] proposed a BMLP architecture which approximates the system dynamics of the perturbed system close to the fixed point and uses the deviation between the predicted state value at the next iteration and the desired fixed point as error feedback signal. Similar methods were applied in [CD95, LMS95]. The main advantage of these methods over the OGY method lies

in the nonlinear approximation of the control formula which extends the controllable region. However this approach is still local in the sense that controls are only designed to stabilize the fixed point when the system naturally approaches its neighborhood.

The methods [OF95, CD95, LMS95] require a priori knowledge of the location of the desired UPO and are thus not appropriate in non-stationary environments. Weeks and Burgess [WB97] used a genetic algorithm to produce a neural controller. The fitness function of the genetic algorithm measures the quality of the solution produced by the controller in terms of several criteria. One criteria measures $\|\mathbf{x}_n - \mathbf{x}_{n-1}\|$ and allows the algorithm to estimate $\mathbf{x}_f$. The algorithm is, however, slow and was not proven to be effective in a non-stationary environment. Konishi and Kokame [KK95, KK97, KK98] developed a neural controller using a local watcher activating the controller whenever $\|\mathbf{x}_n - \mathbf{x}_{n-1}\| < \epsilon$. When activated the controller applied perturbations to all state variables. With error signals based on $\|\mathbf{x}_{n+1} - \mathbf{x}_n\|$ and the size of the perturbation, the controller eventually learns to stabilize the fixed point. Since learning is only performed locally, the created controller is only of local nature. The algorithm was numerically proven to be successful in non-stationary environments.

To our knowledge no BMLP based algorithm has been suggested for targeting, i.e., global chaos control.

3.5.2 Global Control: Reinforcement Learning

All algorithms described in the last subsection share the fact that they are local in the sense that the approximated control law is only designed to stabilize the system at an UPO once the system naturally approaches this state.

We have already mentioned in Subsection 3.4.2 the method suggested by Funke *et al.* [FHD97] which uses a Kohonen network to generate global control neurons

achieving targeting. Der *et al.* [DH94] suggested to target an UPO through reinforcement learning. Recently Lin *et al.* [LJ99] applied an actor-critic neural network to the problem of chaos control. As the method of Funke *et al.*, this method is able to learn a global optimal control policy which can solve the targeting problem and stabilize an UPO.

All the methods introduced in [DH94, FHD97, LJ99] do not require analytical knowledge on system dynamics or previous knowledge on location of the UPO. They have, however, only been tested for the logistic map (and the Hénon map: [LJ99]), and it was not demonstrated that they are applicable in non-stationary environments. Moreover, they all are based on parameter perturbations. The methods of [DH94, FHD97] extract the location of the UPO before the targeting is learned and are therefore expected to be inadequate in non-stationary environments. The method of Lin *et al.* [LJ99] is extremely slow since it attempts to find a globally optimal control policy allowing continuous perturbations everywhere. To control the fixed point of the logistic map an estimated number of 10^{10} iterations was necessary. The applicability of this method in real-time situations is therefore questionable.

3.6 Characteristics of a General Time-Discrete Chaos Controller

The goal of this dissertation is to introduce a general time-discrete chaos controller with potential for "Black-Box" control. In general such a controller should require the least possible input and allow application in as many situations as possible. We can summarize the requirements for a general purpose time-discrete controller as follows:

(1) It must be model independent.

(2) It acts globally and can solve the targeting problem.

(3) It does not require a priori analytical knowledge about the location of the desired state and can solve the tracking problem.

(4) It can be applied to high-dimensional, spatially extended and other complex systems.

(5) It allows to perturb either control parameters or system variables.

(6) It is robust against noise.

(7) It can be applied in real-time situations.

To our knowledge no method currently satisfies all these requirements. The time-continuous Pyragas methods are probably the best candidates for "Black-Box" controllers but require time-continuous system perturbations and do not solve the tracking problem. Concerning time-discrete methods. the OGY method certainly violates (2).(3) and (5) and its applicability to spatially extended systems still has to be thoroughly investigated. Coupled with targeting methods it can satisfy (2) and using a tracking extension it can satisfy (3). No method based on OGY has been suggested which efficiently combines tracking and targeting ideas. Experimental implementations like the OPF method seem good candidates but are not goal oriented and can result in stabilized orbits not present in the unperturbed system. Impulsive control methods are specifically designed for certain experimental situations where no other control is possible and typically have similar problems as the OPF method.

The BMLP based methods described in the preceding subsection are all local in nature and thus violate (2). The reinforcement learning based methods discussed in Subsection 3.5.2 do not satisfy (3) and (5). Both the BMLP and the reinforcement learning methods have not been applied to high-dimensional or spatially extended systems and often need long learning periods.

In the following we will introduce a control algorithm based on reinforcement learning which is a generalization of the method introduced by Der *et al.* [DH94]. In the next chapter we will review the basic concepts of reinforcement learning. To deal with the curse of dimensionality we attempt to approximate the optimal control policy in a reduced space and propose to use vector quantization techniques for this purpose. These will be reviewed in Chapter 5. The proposed algorithm is discussed and exemplified in Chapter 6. applied to spatially extended systems in Chapter 7. and to a multistable rotor in Chapter 8. This version of the algorithm. as published in [GD99]. has the disadvantage of non-vanishing perturbations. An extension is suggested in Chapter 9 which incorporates the So transformation and a BMLP in a final phase to produce a chaos control method which shows all the characteristics (1)-(7).

3.7 Conclusions

In this chapter we attempted to give an overview over existing approaches to chaos control. Due to the large amount of literature in this area such an overview can not be complete. The approaches can generally be classified as feedback and non-feedback methods and one can further distinguish time-continuous and time-discrete methods. Methods applying perturbations to control parameters are referred to as parametric control methods while impulsive methods perturb state variables directly. Different approaches will be suitable for different types of applications. Concerning time-continuous chaos control with feedback on a state variable the Pyragas method is most suitable and successful. We summarized the characteristics of the presented methods and compared them to the characteristics desired for a general time-discrete chaos controller. We believe that currently no general purpose time-discrete chaos controller exists and propose in this dissertation a general time-discrete chaos control algorithm based on reinforcement learning.

Chapter 4

REINFORCEMENT LEARNING

> The greater the satisfaction or discomfort, the greater the strengthening or weakening of the bond.
> (Thorndike, 1911, p. 244)

In this chapter we review the basic concepts of reinforcement learning. In Section 4.1 an overview of the history and the general ideas of reinforcement learning are presented. Section 4.2 introduces the mathematical concepts behind reinforcement learning. Here the temporal-difference learning algorithm known as Q-learning will be at the core of the proposed chaos control algorithm. In Section 4.3 methods to reduce the computational complexity of the reinforcement learning problem are discussed.

4.1 History and General Ideas of Reinforcement Learning

Two major paths led to the development of reinforcement learning (RL) [SB98]: the *trial-and-error theory* of learning and *optimal control theory*. A second major influence from learning theory, the notion of a *secondary reinforcer* led to the development of temporal-difference learning, in particular Watkin's Q-learning [Wat89], which unified the different paths and represents the most widely used approach to reinforcement learning today, see Figure 4.1.

4.1.1 Trial-and-error Learning

A major part of psychology research has focused on the nature and theory of learning. One theory, the *trial-and-error learning* theory, first formulated by E.

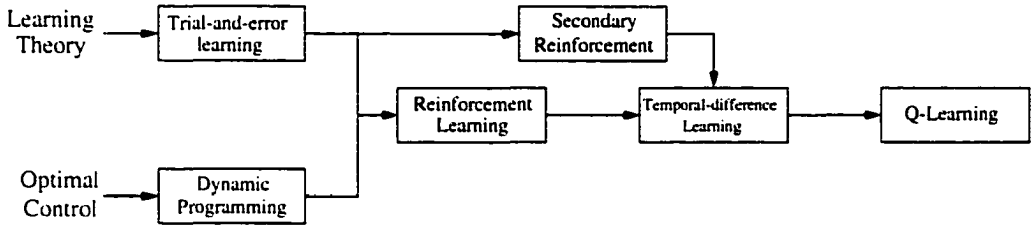


Figure 4.1: The development of reinforcement learning.

Thorndike [Tho11], represents the idea that the probability to choose an action will be enlarged if this action resulted in a good outcome, while it will be decreased if it resulted in a bad outcome. Over time, using trial-and error, a good strategy to achieve a certain goal is established using rewards received through interaction with the surrounding environment. The idea of learning through rewards is the essence of reinforcement learning.

In general, one distinguishes three types of learning [Gul90]: Unsupervised learning, supervised learning and reinforcement learning.

- **Unsupervised learning** represents learning, in which a learning procedure is built into the learning algorithm itself. Vector quantization techniques or self-organizing networks like the Kohonen map [Koh89] are typical examples for unsupervised learning algorithms. In general, given a set of inputs, unsupervised learning algorithms approximate the probability distribution of the input set and encode the input set through a set of reference vectors.

- **Supervised learning** represents learning through an external teacher which provides correct inputs and outputs to guide the learning procedure. Learning minimizes the error between the actual and the desired outputs of the system. Function approximation techniques such as least-squares or back-propagation neural networks [Hay99] can be viewed as examples of supervised learning algorithms.

- **Reinforcement learning** represents learning through an external critic which evaluates an action and provides guidance through evaluative (good-/bad action) feedbacks.

Since reinforcement learning involves an external critic, it is also frequently called *actor-critic learning*. Note the fundamental difference between supervised and reinforcement learning, reflected by the different roles teacher and critic play to guide the learning process. The teacher in supervised learning has access to desired outputs corresponding to input signals and uses these to guide the learning process. The critic in reinforcement learning on the other hand, has no access to desired outputs and merely evaluates if the current state matches certain goal characteristics or not. Thus reinforcement learning can be used to learn in environments in which neither analytical information on system dynamics nor on goal locations is available.

4.1.2 Optimal Control theory

Optimal control theory describes the application of forces to a system for the purpose of maximizing some measure of performance or minimizing a cost functional [Ste94]. If random disturbances influence either the control or the dynamical system to be controlled, the expected value of the performance criterion is to be optimized given assumptions about the statistics of the random disturbances. This leads to *stochastic optimal control theory*.

Optimal control theory can be traced back to the calculus of variations and the brachistochrone problem of finding the shape of a wire such that a bead sliding along it traverses the distance between the two ends in minimum time [Bry96]. One approach to optimal control is today known as *dynamic programming* and is based on the Hamilton-Jacobi equation. Dynamic programming was developed by Bellman and his coworkers in the 1950s [Bel57] and uses the concepts of a

dynamical system's state and a value function to define a functional equation, the *Bellman equation*, which the optimal value function has to satisfy. The dynamic programming approach captures the tradeoff between the desire for low present cost and the undesirability of high future costs [Ber87] by ranking decisions based on the sum of present cost and expected future costs. As a result, the Bellman equation can be solved in a recursive fashion iterating back from the terminal state. Although dynamic programming suffers from the *curse of dimensionality* with computational requirements increasing exponentially with the number of state variables, it is widely considered the only feasible way of solving general stochastic optimal control problems [SB98].

Almost all optimal control methods require complete knowledge of the system to be controlled [SB98]. The application of reinforcement learning to the optimal control problem relieves this requirement and allows the solution of control problems in unknown environments.

4.1.3 Temporal-difference Learning

Temporal-difference learning first introduced the idea of assigning reinforcements based on the difference between temporally successive predictions [Sut88]. Its origins lie in the notion of secondary reinforcer from the field of psychology, which are paired with a primary reinforcer such as food and have similar reinforcing properties [SB98]. A primary reinforcer (instinct) is a stimuli that animals react to without training such as pain or food. A secondary reinforcer is a stimuli that animals react to only after learning about them. In the famous experiments on learning by Pavlov dogs got watery mouths from the hearing of a bell because the ringing of a bell was trained to be associated with food. The basic idea of temporal-difference learning as motivated by learning theory can be summarized as follows [Mob]: "If an environmental event which is neutral with respect to the

animal's survival, is found by experience to precede another event which has direct survival impact, such as the availability of food, then the animal forms a lasting association between those events and can use the prior event to predict the occurrence of the second event."

A particularly important temporal-difference learning scheme is Watkins's Q-learning [Wat89]. Watkins introduced the notion of a state-action value function and derived an update rule for the state-action value function which can be proven to approximate the optimal state-action value function satisfying the Bellman equation under certain conditions. The update is based on instantaneous rewards received from a critic and does not require any explicit knowledge on the system.

Today, reinforcement learning has become one of the major areas in artificial intelligence research and has been applied to a large variety of problems (see [SB98] for references).

4.2 Theory of Reinforcement Learning

Reinforcement learning methods offer a solution to the problem of learning from interaction of a decision maker, called agent, with a controlled environment[1] to achieve a goal. At each discrete time steps $n = 1, 2, \cdots$ the agent receives a representation $\mathbf{w}_n \in \mathcal{W}$ of the environment's state $\mathbf{x}_n \in \mathcal{X} \subset \mathbb{X}$. $\mathcal{X}$ represents the (finite) set of observed environment states, $\mathbb{X}$ the (possibly infinite) set of accessible environment states and $\mathcal{W}$ is the (finite) set of state representations. In certain situations, $\mathcal{W}$ will be equal to $\mathcal{X}$. However if $\mathbb{X}$ is large or even infinite then the

[1] The term environment is used in two ways: it can represent the system the agent is attempting to control or the outside world in which the agent is attempting to achieve a certain task. In the first case the agent perceives information from the environment and in return changes the environments state. In the second case the agent only perceives environment information in order to change its own behaviour but leaves the environment unchanged.

set $\mathcal{W}$ will need to be a finite approximation of $\mathcal{X}$. For the purpose of state-space approximation we introduce an input function $I : \mathcal{X} \rightarrow \mathcal{W}$ which we will construct through a vector quantization technique.

On the basis of $\mathbf{w}_n$, the agent selects a control (or action) $u_n \in \mathcal{U}(\mathbf{w}_n)$, with $\mathcal{U}(\mathbf{w}_n)$ representing the set of all available actions in state $\mathbf{w}_n$. The control is selected according to a policy π, which is described by a probability distribution $\pi_n(\mathbf{w}, u)$ of choosing $u_n = u$ if $\mathbf{w}_n = \mathbf{w}$. One time step later, as a consequence of the control u_n, the agent receives a numerical reward r_{n+1} and a new state $\mathbf{w}_{n+1}$. Formally the model consists of:

- A set of system states $\mathbf{x} \in \mathcal{X} \subset \mathbb{X}$.
- An input function $I : \mathcal{X} \rightarrow \mathcal{W}$ and a discrete set of representation states $\mathbf{w} \in \mathcal{W}$.

  The input function will play an important role in the reduction of the computational requirement of the learning problem. In particular we will represent the set $\mathcal{X}$ by a small set of reference vectors $\mathbf{w} \in \mathcal{W}$ constructed by means of a vector quantization technique. The computed policy $\pi(\mathbf{w}, u)$ is then only an approximation to the optimal policy $\pi(\mathbf{x}, u)$.
- A discrete set of controls $u \in \mathcal{U}(\mathbf{w})$.
- A set of scalar reward signals $r \in \mathbb{R}$.

The goal is to find the *optimal policy* which maximizes the accumulated rewards or reinforcements r. In the general case of the reinforcement learning problem, the agent's actions determine not only its immediate rewards, but also the next state of the environment [KLM96]. As a result, when taking actions, the agent has to take the future into account. This fact is represented by the use of

delayed reinforcements. Typically one uses *infinite horizon models* and seeks to maximize the expectation value of the *discounted return*

$$\hat{r}_n = \sum_{k=0}^{\infty} \gamma^k r_{n+k+1} \quad , \; 0 \leq \gamma \leq 1, \tag{4.1}$$

with the *discount rate* γ. The sum will only converge for $\gamma < 1$ and bounded reward sequence r_n [SB98]. $\gamma = 1$ is suitable for control problems which have a well defined terminal state such that the sum will be finite, while discounting is necessary for continuing tasks where a terminal state is not reached.

Problems with delayed reinforcements and finite representation spaces are well modeled as finite Markov decision processes (MDPs). A finite MDP consists of:

- A finite set of states $\mathbf{w} \in \mathcal{W}$.
- A finite set of controls $u \in \mathcal{U}$.
- A reward function $R(\mathbf{w}, u) : \mathcal{W} \times \mathcal{U} \rightarrow \mathbb{R}$ specifying the expected instantaneous reward of the current state-action pair.
- A state transition function $P^u(\mathbf{w}', \mathbf{w})$ denoting the probability of a transition from state $\mathbf{w}$ to $\mathbf{w}'$ using control u.

The model is Markovian if the state transitions depend only on the current state and control.

Knowledge of the model requires knowledge of R and P. Given an accurate model, the optimal policy can be found through dynamic programming. Otherwise reinforcement learning can be used.

4.2.1 Dynamic Programming

Most dynamic programming algorithms are based on estimation of a state value function

$$V^{\pi}(\mathbf{w}) = E_{\pi}\{\hat{r}_n \mid \mathbf{w}_n = \mathbf{w}\}, \tag{4.2}$$

which gives the value of a state $\mathbf{w}$ under a certain policy π. An optimal value function $V^*(\mathbf{w})$ is defined, for given $\mathbf{w}$, as the maximum over the set of all policies π.

$$V^*(\mathbf{w}) = \max_{\pi} V^{\pi}(\mathbf{w}). \tag{4.3}$$

The optimal value function is unique and satisfies the Bellman equation for infinite horizon problems [Bel57, Ber87]:

$$V^*(\mathbf{w}) = \max_{u \in \mathcal{U}} \left(R(\mathbf{w}, u) + \gamma \sum_{\mathbf{w}' \in \mathcal{W}} P^u(\mathbf{w}', \mathbf{w}) V^*(\mathbf{w}') \right). \tag{4.4}$$

which represents a recurrence relation. Policy and value iteration are two widely used methods which allow the computation of V^* and π^* under the assumption of model knowledge [SB98]. Given an optimal value function $V^*(\mathbf{w}, u)$, the optimal policy π^* can be found from the equation

$$\pi^*(\mathbf{w}) = \arg\max_{u \in \mathcal{U}} \left(R(\mathbf{w}, u) + \gamma \sum_{\mathbf{w}' \in \mathcal{W}} P^u(\mathbf{w}', \mathbf{w}) V^*(\mathbf{w}') \right).$$

Alternative efficient optimization techniques apply Lagrangian relaxation methods. See the books by Bertsekas [Ber87, Ber95, Ber99] for the theory of optimal control and optimization. All these methods typically require analytical knowledge on the system dynamics (the curse of modeling). In this dissertation we assume that no such knowledge is available.

4.2.2 Reinforcement Learning

The reinforcement learning algorithms discussed here are based on estimation of a state-action value function (in comparison to actor-critic methods [BSA83] where the policy is represented independent of the value function)

$$Q^{\pi}(\mathbf{w}, u) = E_{\pi}\{\hat{r}_n \mid \mathbf{w}_n = \mathbf{w}, u_n = u\}, \tag{4.5}$$

which gives the value of a state-action pair under a certain policy. The optimal state-action value function $Q^{\pi}(\mathbf{w}, u)$ is defined as the expected discounted reward of taking action u in state $\mathbf{w}$, then continuing by choosing actions optimally [KLM96]. It follows that

$$V^{*}(\mathbf{w}) = \max_{u \in \mathcal{U}} Q^{*}(\mathbf{w}, u),$$

and from (4.4),

$$Q^{*}(\mathbf{w}, u) = R(\mathbf{w}, u) + \gamma \sum_{\mathbf{w}' \in \mathcal{W}} P^{u}(\mathbf{w}', \mathbf{w}) \max_{u'} Q^{*}(\mathbf{w}', u'). \tag{4.6}$$

Given Q^{*}, choosing in any state $\mathbf{w}$ the action u^{*} with associated maximum value (*greedy control*),

$$u^{*}(\mathbf{w}) = \arg\max_{u \in \mathcal{U}} Q^{*}(\mathbf{w}, u). \tag{4.7}$$

leads to an optimal control strategy. Here $\arg\max_u$ denotes the value of u at which the expression that follows is maximized.

Denoting the operator on the r.h.s of (4.6) by $H(Q)$, i.e., $Q^{*} = H(Q^{*})$, it can be shown [Sin94, BT96] that H is a contraction operator and that the root Q^{*} can be found using the stochastic Robbins Monroe algorithm [RM51].

$$Q_{n+1} = Q_n + \beta_n [H_n(Q_n) - Q_n] \equiv Q_n + \Delta Q_n. \tag{4.8}$$

The $\{\beta_n\}$ are positive and have to satisfy $\sum \beta_n^2 < \infty$, $\sum \beta_n = \infty$ to guarantee convergence with probability one, and a possible choice is $\beta_n = 1/n$.

To apply iteration (4.8) requires knowledge of the operator

$$H_n(Q_n) = R(\mathbf{w}_n, u_n) + \gamma \sum_{\mathbf{w}_{n+1} \in \mathcal{W}} P^{u_n}(\mathbf{w}_{n+1}, \mathbf{w}_n) \max_{u} Q_n(\mathbf{w}_{n+1}, u).$$

which requires knowledge of the expected rewards $R(\mathbf{w}_n, u_n)$ and state-transition probabilities $P^{u_n}(\mathbf{w}_{n+1}, \mathbf{w}_n)$. If the cost structure R and the model probabilities

P are not known, then an approximation of $H_n(Q_n)$ has to be used instead. Fortunately the stochastic iteration (4.8) can be proven to converge for cases where $H_n(Q_n)$ is not accurately known [BT96].

In *temporal-difference learning* methods [Sut88] the estimate of $H_n(Q_n)$ is based on observations of the present state $(\mathbf{w}_n, u_n)$ and the next state $(\mathbf{w}_{n+1}, u_{n+1})$ together with the immediate reward r_{n+1} which estimates $R(\mathbf{w}_n, u_n)$. Only the value of Q_n at $(\mathbf{w}_n, u_n)$ is updated, i.e., $\Delta Q_n(\mathbf{w}, u) = 0$ if $(\mathbf{w}, u) \neq (\mathbf{w}_n, u_n)$. The update at $(\mathbf{w}_n, u_n)$ depends on the choice of estimate for the sum

$$\sum_{\mathbf{w} \in \mathcal{W}} P^{u_n}(\mathbf{w}, \mathbf{w}_n) \max_u Q_n(\mathbf{w}, u). \tag{4.9}$$

leading to different temporal-difference learning architectures. Estimating (4.9) by $\max_u Q(\mathbf{w}_{n+1}, u)$ leads to Watkins *Q-learning* [Wat89].

$$\Delta Q(\mathbf{w}_n, u_n) = \beta_n [r_{n+1} + \gamma \max_u Q(\mathbf{w}_{n+1}, u) - Q(\mathbf{w}_n, u_n)]. \tag{4.10}$$

This procedure represents an *off-policy method*: the Q-values are updated independent of the policy the controller uses to select controls. General convergence results concerning Q-learning can be found in [JJS94].

Estimating (4.9) by $Q(\mathbf{w}_{n+1}, u_{n+1})$ leads to the *Sarsa-learning* algorithm [RN94, Sut96], which represents an *on-policy method*.

$$\Delta Q(\mathbf{w}_n, u_n) = \beta_n [r_{n+1} + \gamma Q(\mathbf{w}_{n+1}, u_{n+1}) - Q(\mathbf{w}_n, u_n)]. \tag{4.11}$$

A convergence proof of this version of Sarsa-learning (one-step tabular Sarsa) can be found in [SJLS98].

For both algorithms, convergence to the optimal value function requires that

1. The state space $\mathcal{W}$ and the action spaces $\mathcal{U}(\mathbf{w})$ are finite.

2. Every state-action pair in $(\mathcal{W} \times \mathcal{U})$ is updated infinitely often.

3. $\sum_{n=0}^{\infty} \beta_n = \infty$ and $\sum_{n=0}^{\infty} \beta_n^2 < \infty$.

4. $\mathrm{Var}(r_n) < \infty \ \forall n$.

5. If $\gamma = 1$ all policies lead to a cost free terminal state with probability one.

Although it is clearly not possible to satisfy the second condition, in practice, as noted in [Sin94], one is not interested in the optimal state-action value function Q^* but in the optimal policy associated with it. It is possible that the policy greedy with respect to the estimated function Q becomes optimal after only a finite number of visits of the action-state space. Unfortunately no general stopping criterion exists to detect when the policy is optimal. In real applications, in particular in situations with changing environments, it is generally impossible to find the optimal policy, and one must be content with policies which achieve the goal and considerably improve performance over the uncontrolled system. In non-stationary environments it is convenient to set $\beta_n = \beta = \text{const.}$ with $0 < \beta < 1$ [SB98]. Complete convergence can then not be guaranteed even for infinite action-state space visits, but the controller continues to receive rewards and is able to adjust to changing environments.

We have summarized Q-and Sarsa-learning in Figure 4.2.

4.2.3 Epsilon-greedy Policies and Episode Learning

The convergence requirements imply that the whole state-action space is explored. To ensure this, control actions are chosen from the corresponding Q values according to a specified policy which initially chooses actions stochastically and is slowly frozen into a deterministic policy. This can for example be achieved through *ϵ-greedy policies* π_ϵ, where an action $u \in \mathcal{U}(\mathbf{w})$ which is different from the greedy action u^* is chosen with probability $\pi_\epsilon(\mathbf{w}, u) = \epsilon/|\mathcal{U}|$, where ϵ satisfies initially

Initialize $Q(\mathbf{w}, u)$ arbitrarily
Repeat (for each episode)
  Initialize **x**, observe **w**
  Sarsa-learning
   Choose u from **w** using policy derived from Q
  Repeat (for each step of episode)
    Q-learning
     Choose u from **w** using policy derived from Q
     Apply control u, observe r, $\mathbf{w}'$
     $\Delta Q(\mathbf{w}, u) = \beta\left(r + \gamma \max_u Q(\mathbf{w}', u) - Q(\mathbf{w}, u)\right)$
     $\mathbf{w} \leftarrow \mathbf{w}'$
    Sarsa-learning
     Apply control u, observe r, $\mathbf{w}'$
     Choose u' from $\mathbf{w}'$ using policy derived from Q
     $\Delta Q(\mathbf{w}, u) = \beta(r + \gamma Q(\mathbf{w}', u') - Q(\mathbf{w}, u))$
     $\mathbf{w} \leftarrow \mathbf{w}'$; $u \leftarrow u'$
  until **w** is terminal
until policy approximation terminated

Figure 4.2: Pseudo-code of Q- and Sarsa-learning.

$0 < \epsilon \leq 1$ and is slowly frozen to zero.

$$\pi_\epsilon(\mathbf{w}, u) = \begin{cases} 1 - \epsilon + \epsilon/|\mathcal{U}| & \text{if} \quad u = u^* \\ \epsilon/|\mathcal{U}| & \text{if} \quad u \neq u^*. \end{cases} \tag{4.12}$$

In (4.12), $|\mathcal{U}|$ denotes the cardinality of the set $\mathcal{U}$, i.e. the number of elements contained in $\mathcal{U}$. We call a policy greedy or deterministic, if exclusively the action u^* is chosen ($\epsilon \equiv 0$).

In applications where a well defined terminal state exists, ϵ-greedy learning is typically performed in *episodes*. An episode is defined as an iteration of the system from a random initial condition until the terminal state is reached.

4.2.4 Rewards

The temporal-difference learning algorithms offer a way of solving optimal control problems through experience gained from interaction with the environment. No analytical knowledge about the environment or the goal state is necessary. The choice of the instantaneous rewards r reflects the desired form of the cost functional in an optimal control problem (negative rewards represent a cost). Typically, simple scalar signals r are used such as $r_{n+1} = 1$ if $\mathbf{w}_{n+1}$ represents the goal state and $r_{n+1} = 0$ otherwise. We will discuss this further in the context of our chaos control algorithm in Chapter 6.

4.2.5 An Example

As an illustrative example of how temporal-difference learning works, consider Figure 4.3. Assume an artificial mouse in a limited environment has the task to find a certain location with food as fast as possible from anywhere in the environment. We equip the mouse with the ability to store a Q-value function and update the values with the Q-learning rule

$$\Delta Q(\mathbf{w}_n, u) = r_{n+1} + \max_u Q(\mathbf{w}_{n+1}, u) - Q(\mathbf{w}_n, u),$$

setting $\beta = \gamma = 1$. Initially all Q-values are set to zero. The possible states $\mathbf{w}_n$ are grid locations and the possible actions u_n are to move either north, south, east or west on the grid. As reward $r_{n+1} = -1$ is received if $\mathbf{w}_{n+1}$ does not represent the goal location. Otherwise the episode is terminated and learning continued from a new initial condition. Figure 4.3 shows three different paths through the environment to the goal and the resulting Q-values. Also shown is the eventually approximated optimal policy to which the algorithm will converge if ϵ-greedy learning is performed. Using the optimal policy by taking in each state actions with associated maximum value the mouse will reach the goal location from

anywhere in the environment in a minimum number of steps. It is interesting to note that the policy will stabilize from the goal location outwards. This is similar to the principle of dynamic programming.

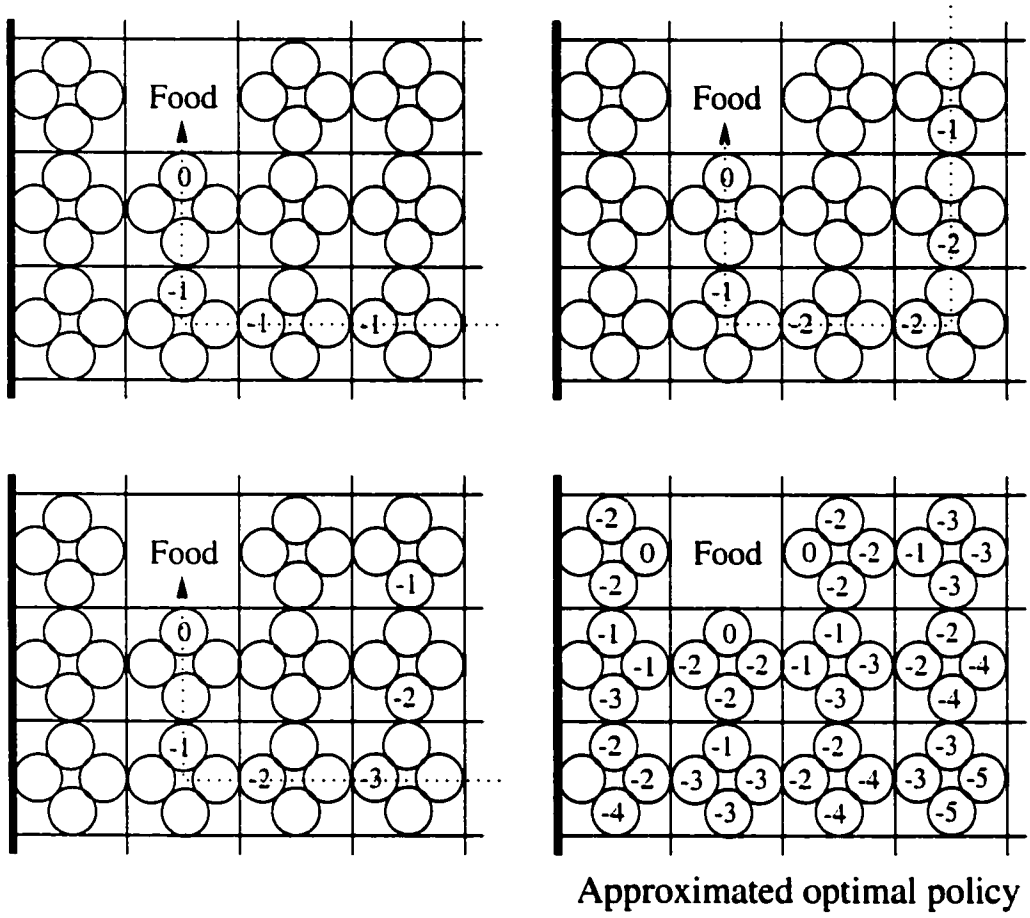


Figure 4.3: An illustrative example of Q-learning. See text for details.

4.3 Reduction of the Learning Task

Most reinforcement learning algorithms assume that the task is a Markov decision process, with discrete states and discrete actions. The convergence proofs for an optimal state-action value function Q^* require that every state-action pair in $(\mathcal{W} \times \mathcal{U})$ is updated infinitely often. Clearly this can never be achieved. Nonethe-

less, to even come close to this requirement we should at least update every state-action pair a finite number of times. To this end the idea of ϵ-greedy policies was introduced which guarantee that actions of low value are taken with nonzero probability.

Let $\mathcal{X} \subset \mathbb{R}^{d_x}, \mathcal{W} \subset \mathbb{R}^{d_w}$ and $\mathcal{U} \subset \mathbb{R}$. Assuming that the input function $I : \mathcal{X} \rightarrow \mathcal{W}$ is the identity, the computational complexity of the learning problem is approximately of the order $|\mathcal{U}||\mathcal{X}|^{d_x}$. This represents the curse of dimensionality also present in dynamic programming problems [Ber95].

The role of the input function $I : \mathcal{X} \rightarrow \mathcal{W}$ is to effectively reduce the computational complexity to the order $|\mathcal{U}||\mathcal{W}|^{d_w}$.

The set $\mathcal{W} \subset \mathbb{R}^{d_w}$ approximates the true observed phase space $\mathcal{X} \subset \mathbb{R}^{d_x}$. If the dimension d_x is large, a dimension reduction technique like a principal components analysis (also known as Karhunen-Loève transformation) [Hay99] can be used to reduce d_x to a dimension $d_w < d_x$. Examples of this type of reduction in the context of chaos control will not be considered in this dissertation but left for future research. In the following we will then assume $d_w = d_x$.

We will however consider effective reduction from $|\mathcal{X}|$ to $|\mathcal{W}| << |\mathcal{X}|$. The systems we consider in this dissertation exhibit complex dynamics. As a consequence, to capture the full behaviour of the system formally requires an infinite number of observations $\mathbf{x}_n$. In particular chaotic dynamics are characterized by a never repeating dynamical state evolution, i.e., every state observation leads to a state that was never observed before, and the controller is faced with the task to control states never experienced before. This is revealed in the fact that the set $\mathcal{X}$ represents a chaotic attractor embedded in a continuous phase space and as such has uncountably infinite cardinality. Since on the other hand the discussed reinforcement learning algorithms require a finite number of states, it is unavoidable to use an input function different from the identity which maps the input set $\mathcal{X}$ to

a finite set $\mathcal{W}$. In essence this means approximation of the deterministic, continuous phase space dynamics by a finite state Markov process. The input function I must then serve for two purposes: 1) data compression by reducing the number of states. 2) generalization by associating a dense neighborhood of environment states to a sparse discrete set of representative states. The reinforcement learning will then approximate an optimal action value function $Q^*(\mathbf{w}, u)$ which represents an approximation of $Q^*(\mathbf{x}, u)$.

Different approaches have been proposed in the reinforcement learning literature to implement the mapping I. Conventional approaches discretize the entire input space into equal-sized boxes or more general *tiles*. This approach is known as *tile coding* [SB98]. Of particular interest are *adaptive resolution methods* where during the course of learning a partition is constructed that is appropriate to the environment. An example is Moore's PartiGame algorithm [Moo94] which adaptively divides the environment into cells until appropriate actions to achieve the goal are established.

More general approaches consider continuous state and action spaces. Simple approaches achieve generalization by quantizing the state and action spaces into a finite number of cells and aggregate all states and actions within each cell [SSR98, Sut96]. We follow this idea and apply a vector quantization technique to find a suitable quantization of the state space. For the most part of this dissertation the action space associated with each state will be equal and consist of a small number of finite actions. Thus we do not consider generalization over actions. In the last chapter we will allow state dependent control sets.

Alternative techniques have been suggested for generalization such as feature mapping and decision trees [SB98, UV98]. Other types avoid quantization by using general function approximators such as neural networks, radial basis functions

(see references in [SP99, SO99]) or different types of sparse coarse-coded function approximators [SSR98].

To find a suitable state quantization, the goal is to find an optimal construction of $\mathcal{W}$ which approximates the probability distribution of the set $\mathcal{X}$ and is small enough to allow an efficient computation of $Q^*(\mathbf{w}, u)$.

The problem of constructing an optimal partitioning is generally solved by clustering techniques and it is rather surprising that these methods have not been much used in the context of reinforcement learning. To the author's knowledge the first who suggested to combine a vector quantization technique (specifically Kohonen's self-organizing network) and reinforcement learning were Der and Herrmann [DH94] in their approach to the control of a chaotic system. More recently Fernández and Borrajo [FB99] applied the LBG algorithm and we [GD99] suggested to use the Neural-Gas algorithm. The application of vector quantization techniques to dynamical systems can be traced back to the work of Walter *et al.* [WRS90] who applied it to the prediction of nonlinear dynamical systems. We have applied the Neural-Gas algorithm to the approximation of the Markov transition probabilities and the prediction of chaotic systems [DSHK99]. Since reinforcement learning relies on the assumption of a Markov decision process, the above successes suggest that the application of a vector quantization technique to reinforcement learning is promising, especially for the control of chaotic systems. The attractor of a chaotic system typically has a topologically complex structure and only visits a small fraction of the phase space in which it is embedded. It follows that a reduction technique is desired which preserves the topological properties of the attractor. We will review such vector quantization techniques in the next chapter.

4.4 Conclusion

In this chapter we gave a brief overview over the field of reinforcement learning and introduced temporal-difference learning. Many different temporal-difference

learning algorithms have been suggested (see e.g. [SB98]) and although in some situations a different algorithm might lead to better results, we will almost exclusively use Q-learning in this dissertation. To find a discrete state space representation we will apply vector quantization and we briefly mentioned alternative representation techniques in this chapter but did not apply these in this dissertation.

Chapter 5

DATA CLUSTERING

> When optimizing a very large and complex system, instead of always going downhill, try to go downhill most of the time.
>
> (S. Haykin, 1999, p. 559)

In this chapter we review basic vector quantization techniques. Parts of this introduction follow closely the overview of Fritzke [Fri97b].

5.1 Introduction

Clustering broadly describes the task of partitioning an input space $\mathbb{X}$ into a set of M units according to a certain *clustering criterion*. Clustering has many applications, such as data reduction, pattern recognition, control, robotics and hypothesis testing [Koh95], to name a few. As a result, many different types of clustering algorithms have been developed with different characteristics (see [TK99] for an overview). We will concentrate here on *competitive learning algorithms* or *self-organizing networks*. The basic principle underlying competitive learning is *vector quantization*.

5.2 Vector Quantization

The main use of vector quantization is data compression (see [Ger79, LBG80, Gra84, GG92]). Most vector quantization methods approximate the probability distribution function $P(\mathbf{x})$ of an input set of vector variables (features) $\mathbf{x} \in \mathbb{X} \subset \mathbb{R}^{d_x}$ of dimension d_x, by a finite set of reference vectors $\mathbf{w} \in \mathcal{W} \subset \mathbb{R}^{d_x}$. The set $\mathcal{W}$

is commonly referred to as *codebook* and its dimension is equal to the dimension of $\mathbb{X}$. The set of $\mathbf{w}$ partitions the input space into a set $\mathcal{M}$ of $M := |\mathcal{M}| = |\mathcal{W}|$ *units* or *cells*

$$\mathcal{M} = \{m_1, m_2, \cdots, m_M\}.$$

Each unit $m \in \mathcal{M}$ has a reference vector $\mathbf{w}_m \in \mathcal{W}$ associated indicating its position or *receptive field center* in the input space. The units together with the reference vectors form a *network*.

An input vector $\mathbf{x}$ is associated to a reference vector through an input function

$$I : \mathbb{X} \rightarrow \mathcal{W}.$$

Nearest neighbor quantizers associate an input vector $\mathbf{x}$ to its nearest reference vector:

$$\mathbf{w}(\mathbf{x}) := I(\mathbf{x}) = \arg \min_{\mathbf{w} \in \mathcal{W}} d(\mathbf{x}, \mathbf{w}).$$

where d is the distance measure used in the association process, usually euclidean distance

$$d(\mathbf{x}, \mathbf{w}) = \|\mathbf{x} - \mathbf{w}\|.$$

but other measures which are better suited for particular applications have been used as well (see [TK99]). The association process $\mathbf{w}(\mathbf{x})$ results in an information loss which is known as *distortion*. The major goal in vector quantization is to distribute the $\mathbf{w}_i$ such that the distortion is minimized. Nearest neighbor classification is not necessarily the best solution in terms of distortion but more efficient than algorithms including more neighbors. As a result, nearest neighbor classification is typically used in classification tasks. We will discuss different types of classification algorithms based on nearest neighbor classification in the next section.

Using nearest neighbor association, a given codebook of size M partitions the input space $\mathbb{X}$ into M cells which are called *Voronoi regions*. The Voronoi region $\mathbb{V}_w$ of a vector $\mathbf{w}$ is defined as the set of all points in $\mathbb{X}$ for which $\mathbf{w}$ is the closest reference vector:

$$\mathbb{V}_w = \{\mathbf{x} \in \mathbb{X} | \mathbf{w} = \mathbf{w}(\mathbf{x})\}.$$

In the case of a finite input data set $\mathcal{X}$ the Voronoi region becomes a finite *Voronoi set* $\mathcal{V}_w$. The partition of $\mathbb{X}$ formed by all Voronoi regions is called a *Voronoi tessellation*. Every point $\mathbf{x}$ is uniquely associated to a Voronoi region unless it is exactly on the border of two regions, in which case it will be associated randomly to one of the regions.

Connecting all pairs of points for which the respective Voronoi regions share an edge results in the *Delaunay triangulation*. A subgraph of the Delaunay triangulation, the *induced Delaunay triangulation*, can be constructed from a dataset by connecting the two closest reference vectors to an input vector with an edge [Fri95a]. The induced Delaunay triangulation is limited to areas of the input space with nonzero probability $p(\mathbf{x})$ and shown to optimally preserve the topology of the input space in a general sense [Mar93]. The approximated topology of the input space is represented by a (possibly empty) symmetric *connectivity set*

$$\mathcal{C} = \mathcal{M} \times \mathcal{M}$$

of *neighborhood connections*. Some competitive learning algorithms like the Topology Representing Network [MS94] and the Growing Neural-Gas algorithm [Fri95a] approximate the codebook together with the connectivity set. For other algorithms, this set can be constructed through the induced Delaunay triangulation after a codebook has been distributed over the input space. Unless an algorithm explicitly uses the connectivity set, there is no advantage of creating it during

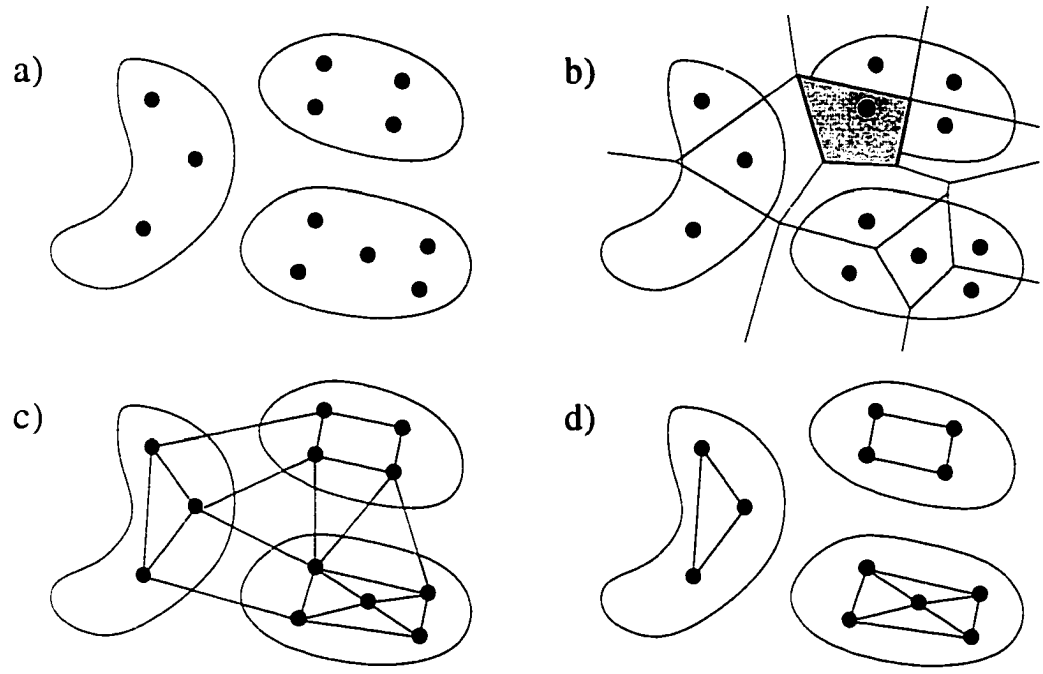


Figure 5.1: a) The light gray area represents the distribution of an input set $\mathcal{X} \subset \mathbb{R}^2$ and is characterized by $p(\mathbf{x}) > 0$. The dots represent the reference vectors of a possible vector quantization. b) Sketch of the Voronoi tessellation of the codebook. The dark gray area is an example of a Voronoi region. c) Delaunay triangulation of the codebook. d) Induced Delaunay triangulation. Two centers are only connected if the common borders of their Voronoi regions are at least partially inside the shaded area.

the construction of the codebook. See Figure 5.1 for examples of the concepts discussed in this section.

In order to find a "good" codebook, algorithms typically attempt to minimize certain performance measures or cost functions. This will be the topic of the next section.

5.3 Error Minimization and Learning

An often used optimality criterion for a codebook is the *expected squared distortion error* (or just *distortion error*). For the continuous case with the input

signal distribution $P(\mathbf{x})$ this becomes

$$E[P(\mathbf{x}), \mathcal{W}] = \sum_{\mathbf{w} \in \mathcal{W}} \int_{\mathcal{V}_w} P(\mathbf{x}) d(\mathbf{x}, \mathbf{w})^2 d\mathbf{x}. \tag{5.1}$$

In the finite case with finite input set $\mathcal{X}$ it is

$$E[\mathcal{X}, \mathcal{W}] = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{w} \in \mathcal{W}} \sum_{\mathbf{x} \in \mathcal{V}_w} d(\mathbf{x}, \mathbf{w})^2. \tag{5.2}$$

where $|\mathcal{X}|$ denotes the size of the input set, $|\mathcal{X}| =: N$. Many different error functions exist in the literature (see [TK99]), but we will only consider algorithms here which attempt to minimize the distortion error.

It can be shown (see [Zad82]) that an optimal selection of a codebook approximates the probability distribution of a sufficiently large input set to $(p(\mathbf{x}))^{M/(M+2)}$ for large M, where $M := |\mathcal{W}|$. Therefore, for M sufficiently large, an optimal codebook approximates the probability distribution of the input set.

Many different algorithms exist which attempt to find an optimal codebook based on minimization of the error function (5.1) or (5.2) . These algorithms differ in the way the features are presented and in the way the feature information is used to update the codebook. Further characteristics are for example the use of a fixed or variable codebook size or the use or construction of a connectivity structure during learning. Learning is said to be *on-line* if the codebook is updated each time a feature is presented. This type of learning is particularly useful in "real-time" applications where constantly new features arrive. In *batch learning* all features are presented to the algorithm at the time the codebook is constructed. Most learning methods apply feature information in a competitive manner to update the reference vectors. When a vector $\mathbf{x}$ is presented to the algorithm using *competitive learning*, all reference vectors compete with each other and desire to approach the presented state. The winner of this competition is the representative that lies closest to $\mathbf{x}$ and is updated so as to move toward $\mathbf{x}$, while the losers either remain unchanged or

are updated toward $\mathbf{x}$ at a slower rate [TK99]. One distinguishes *hard competitive* and *soft competitive* learning algorithms as discussed in the next two sections.[1]

5.4 Hard Competitive Learning

Hard competitive learning represents methods where the information of an input state affects only the update of its nearest reference vector such that learning in these *winner-take-all* methods is only local.

These algorithms typically minimize the error functions (5.1) and (5.2) directly and face two problems as a result of this: local minima and dead units.

The minimization process of the error function can be viewed as the problem of finding the global minimum of a complicated surface with many local minima (see Figure 5.2). In hard competitive learning, the units behave stiff and are easily caught in a local minimum (Figure 5.2a). A possible solution to this problem is the use of soft competitive learning, where the winner is updated together with some of its neighbors. One can think of the units as being linked by an elastic band. The stronger the link, the easier units will be able to escape a local minimum (Figure 5.2b). To remain in a global minimum, the strength of the link has to slowly approach zero. Competitive learning algorithms can be interpreted in terms of statistical mechanics, where one introduces a "temperature". The higher the temperature, the more elastic a unit. Lowering the temperature slowly, freezes the units eventually in the global minimum. Soft competitive algorithms will be discussed in the next section.

A second problem of hard competitive learning is that these algorithms are sensitive to the initial distribution of the codebook vectors. For example, due

---

[1]Some authors (e.g. [TK99]) consider competitive learning as an on-line procedure. Fritzke [Fri97b] however distinguishes on-line and batch competitive learning algorithms. In general, the categorization of algorithms into different types is not clear distinguished in the literature. We follow here the categorization of Fritzke [Fri97b].

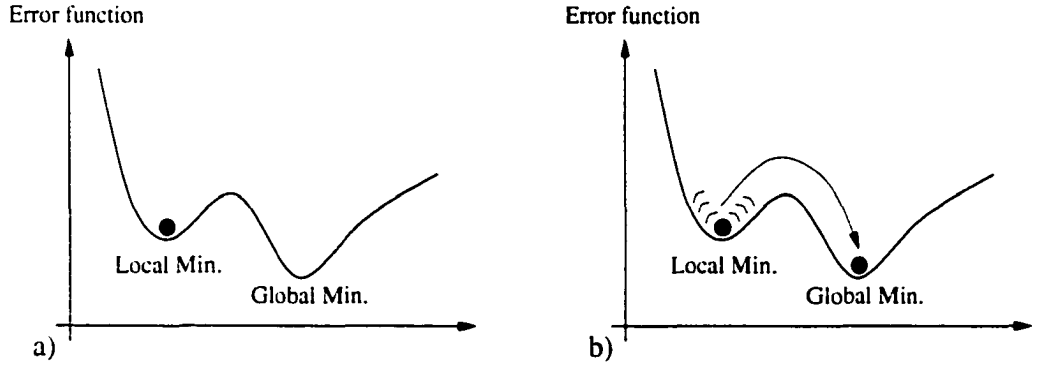


Figure 5.2: Illustration of the difference between hard and soft learning. a) In hard learning, the units behave stiff and can not escape a local minimum. b) In soft learning units are more elastic and can "bounce" out of local minima.

to a wrong initialization, units might never be updated and not contribute to the minimization process. These units are so-called *dead units*. Soft competitive learning also offers a solution to this problem. Another type of solution is to frequently delete unproductive units and insert new ones (grid growing).

An advantage of hard competitive algorithms is that they are generally faster than soft competitive algorithms. We present two hard competitive algorithms in the next subsections: the LBG algorithm, a batch version, and the k-means algorithm, an on-line version.

5.4.1 The LBG Algorithm

A necessary condition for a codebook to minimize the distortion error of a finite set is that each reference vector $\mathbf{w}$ fulfills the *centroid condition* [GG92]

$$\mathbf{w} = \frac{1}{|\mathcal{V}_w|} \sum_{\mathbf{x} \in \mathcal{V}_w} \mathbf{x}.$$

The LBG algorithm of Linde, Buzo and Gray [LBG80] implements this condition by repeatedly moving all reference vectors to the arithmetic mean of the Voronoi cells. The complete LBG algorithm is summarized in Figure 5.3.

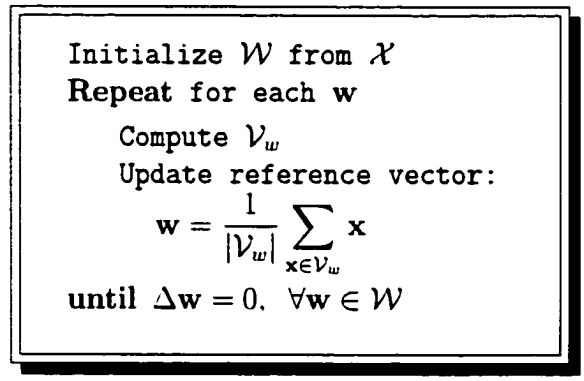


Figure 5.3: Pseudo-code of the LBG algorithm.

The codebook vectors are initialized to distinct vectors randomly chosen from $\mathcal{X}$. The main loop forms a so-called *Lloyd iteration* [Llo82], which is guaranteed to decrease the distortion error or leave it unchanged. As a result, LBG is guaranteed to converge in a finite number of steps to a local minimum [Fri97b]. An example is shown in the last section of this chapter.

Extensions like the LBG-U method [Fri97a] improve upon a converged LBG codebook by moving units of low *utility*. Due to the global character of the batch update, the LBG algorithm can become slow for large datasets. The use of efficient tree structures [Hun98, TK99] or a recently introduced local version [AK99] strongly improve the computational efficiency of the algorithm.

5.4.2 The k-means Algorithm

Batch methods are not suitable for situations in which new data is arriving in real time or on-line. Furthermore, the dataset $\mathcal{X}$ might be too large to be stored. Then on-line clustering methods are necessary. A popular algorithm is the *k-means* (or *isodata*) algorithm [Mac67].[2] Hard on-line algorithms update the

[2]Frequently, e.g. in [TK99], k-means and LBG are referred to as the same algorithm.

winner reference vector according to the rule

$$\Delta \mathbf{w}(\mathbf{x}) = \epsilon \, (\mathbf{x} - \mathbf{w}(\mathbf{x})).$$

where the *learning rate* ϵ determines the extent by which the winner is moved towards the input signal. In k-means ϵ is depending on the unit and decreases according to the harmonic series $\epsilon(k_w) = 1/k_w$, where k_w stands for the number of input signals for which the unit corresponding to $\mathbf{w}$ has been the winner so far [Fri97b]. The k-means algorithm is summarized in Figure 5.4.

```
Initialize W from X
Initialize k_w = 0  ∀w ∈ W
Repeat
    Choose x randomly from X
    Find winner w(x)
    k_w(x) = k_w(x) + 1
    Δw(x) = (1/k_w(x))(x − w)
until stopping criterion reached
```

Figure 5.4: Pseudo-code of the k-means algorithm.

The k-means algorithm bears its name from the fact that each reference vector is always the exact arithmetic mean of the input signals it has been winner for so far. It can be shown [Mac65] that each reference vector converges such that its location satisfies the continuous centroid condition

$$E\left[\mathbf{x} | \mathbf{x} \in \mathcal{V}_w\right] = \int_{\mathcal{V}_w} P(\mathbf{x}) \mathbf{x} \, d\mathbf{x}.$$

On-line methods are generally slower than batch methods since batch methods allow efficient computation through the use of tree structures [Hun98, TK99]. On-line methods on the other hand are easier to implement, can adapt to non-stationary environments and use additional information as it becomes available. The choice of method depends on the situation in which it is intended to be used.

5.5 Soft Competitive Learning

Soft competitive or *fuzzy* clustering algorithms associate a state $\mathbf{x}$ to a reference vector via a *membership function* $h_\eta(\mathbf{x}, \mathbf{w})$ which describes the relation of $\mathbf{x}$ to a reference vector $\mathbf{w}$ and describes the size of the neighborhood of a winner unit $\mathbf{w}(\mathbf{x})$ which takes part in the update process or the strength by which neighboring units are updated.[3] The parameter η is called *fuzzifier* and regulates the strength of the membership relation during the learning process. In hard competitive learning the membership function is

$$h_0(\mathbf{x}, \mathbf{w}) = \begin{cases} 1 & \text{if } \mathbf{w} = \mathbf{w}(\mathbf{x}) \\ 0 & \text{otherwise.} \end{cases}$$

Soft competitive learning is related to the stochastic relaxation scheme known as *simulated annealing* [KCGV83]. The primary objective of simulated annealing is to find the global minimum of a cost function that characterizes a complex system [Hay99]. The annealing process adaptively approximates the global minimum by slowly lowering a temperature. The fine details of the final state appear slowly with decreasing temperature. This process of decreasing the temperature is equivalent to a decrease of the neighborhood association described by the membership function.

A distortion function measuring the expected squared distance can in general be expressed as

$$E_\eta = \frac{1}{c_\eta} \sum_{\mathbf{w} \in \mathcal{W}} \sum_{\mathbf{x} \in \mathcal{X}} h_\eta(\mathbf{x}, \mathbf{w})\, d(\mathbf{x}, \mathbf{w})^2,$$

with a normalization constant c_η typically introduced for convenience. The elements h form a $N \times M$ matrix, which, for the case of hard competitive learning,

---

[3]The resulting vector quantizer can still be a nearest neighbor quantizer if the membership association is only used during the codebook approximation.

becomes a matrix with only one element in each row. For soft competitive learning this matrix will have more entries and algorithms differ in the particular choice of membership relation h and distortion measure d. Typically E_η will be such that E_∞ is convex, with no local minima, and E_0 equal to the hard competitive error function, which has the global minimum to be approximated. By decreasing η slowly, the algorithm is less likely to get stuck in a local minimum during the optimization process.

The *Neural-Gas* algorithm [MBS93] employs the membership function

$$h_\eta\left(k(\mathbf{x}, \mathbf{w}_i)\right) = e^{-k(\mathbf{x}, \mathbf{w}_i)/\eta(t)}.$$

where k is such that the set $\mathcal{W} \setminus \{\mathbf{w}_i\}$ (the codebook without $\mathbf{w}_i$) contains exactly k elements $\mathbf{w}$ satisfying $d(\mathbf{x}, \mathbf{w}) < d(\mathbf{x}, \mathbf{w}_i)$. This realizes the membership function based on a neighborhood ranking of a reference vector with respect to the input signal. For example $k(\mathbf{x}, \mathbf{w}_i) = 3$ would mean that $\mathbf{w}_i$ is the third closest reference vector to $\mathbf{x}$. The algorithm represents an on-line method and is summarized in Figure 5.5. In this figure t_{max} denotes the maximum number of iterations. The functions $\eta(t)$ and $\epsilon(t)$ are given by the exponentially decaying functions

$$\eta(t) = \eta_i \left(\frac{\eta_f}{\eta_i}\right)^{t/t_{max}}, \quad \epsilon(t) = \epsilon_i \left(\frac{\epsilon_f}{\epsilon_i}\right)^{t/t_{max}}.$$

and in [MBS93] the parameters were chosen as

$$\eta_i = 10, \ \eta_f = 0.01, \quad \epsilon_i = 0.5, \ \epsilon_f = 0.005.$$

As shown in [MBS93], the Neural-Gas algorithm performs stochastic gradient descent on the cost function

$$E_{ng} = \frac{1}{c_\eta} \sum_{\mathbf{w} \in \mathcal{W}} \int_{\mathbf{X}} P(\mathbf{x}) h_\eta(k(\mathbf{x}, \mathbf{w})) d(\mathbf{x}, \mathbf{w})^2 d\mathbf{x}.$$

with euclidean measure $d(\mathbf{x}, \mathbf{w})$. For $\eta \to 0$ this cost function becomes equal to the cost function (5.1) which we are attempting to minimize but which has many

```
Initialize W from X
For t = 0 to t_max
    Choose x randomly from X
    For all w
        Δw = ε(t) h_η(t) (k(x, w)) (x − w)
end
```

Figure 5.5: Pseudo-code of the Neural-Gas algorithm.

local minima. For $\eta \rightarrow \infty$ on the other hand E_{ng} becomes parabolic with no local minima. Thus decreasing η slowly should allow convergence to smaller values of the distortion error in comparison to k-means. This was supported by results in [MBS93] which found better performance of the Neural-Gas algorithm in comparison with the k-means algorithm and the two competitive learning schemes. maximum-entropy [RGF90] and the Kohonen map [Koh95]. Furthermore it was shown in [MBS93] that the computational complexity of the Neural-Gas algorithm is comparable to k-means if computing is performed in parallel. We will see examples of Neural-Gas clusterings at the end of this chapter.

5.6 Growing Codebooks

The previously mentioned algorithms have the disadvantage that they require the size of the codebook to be chosen in advance. In many applications however it is of advantage to have a codebook which is as small as possible. Then grid growing algorithms can be used. which generate the codebook by starting from an initially small set of reference vectors and successively enlarge the codebook during the learning process. As a stopping criterion typically a small value of the distortion error is used which leads to a codebook of minimal size for the respective distortion error. Examples of different grid growing algorithms are the algorithms by Fritzke [Fri95a. Fri95c] and Bruske and Sommer [BS95].

The Growing Neural-Gas algorithm of Fritzke combines a growing cell structure algorithm [Fri94] with competitive Hebbian learning [Mar93]. The resulting algorithm creates a growing codebook and a connectivity matrix which approximates the topology of the input space in an optimal way. During the adaptation process new cells are inserted in areas which possess a large accumulated local error. Dead units are avoided through an aging of existing connections of cells.

The Growing Neural-Gas algorithm is summarized in Figure 5.6 [Fri95a]. In this figure $\mathcal{T}(\mathbf{w})$ denotes the topological neighborhood of $\mathbf{w}$ and contains all units which are directly connected to $\mathbf{w}$, except $\mathbf{w}$ itself:

$$\mathcal{T}(\mathbf{w}_i) = \{ \mathbf{w} \mid c(\mathbf{w}_i, \mathbf{w}) = 1 \}.$$

The values $c(\mathbf{w}_1, \mathbf{w}_2) \in \mathcal{C}$ describe the connectivity structure between existing cells and are 0 if no connection exists between $\mathbf{w}_1$ and $\mathbf{w}_2$ and 1 otherwise. Function $a(\mathbf{w}_1, \mathbf{w}_2)$ denotes the age of the connection and $e(\mathbf{w})$ the accumulated local error of a cell. The function $\mod(i, \tau)$ represents the remainder after division of i with τ. Thus cells are only added every τ iterations.

As a stopping criterion one can choose for example a desired minimal size of the distortion error, a maximal codebook size or a maximal number of iterations. However, if the codebook size is used as stopping criterion, then the Growing Neural-Gas algorithm will not lead to an optimal codebook in terms of distortion error. The reason is that in the update of the weights constant factors ϵ_w and ϵ_n are used. To this end we propose to add an annealing phase to the Growing Neural-Gas algorithm after the stopping criterion is achieved (this first phase will be called growing phase). During the annealing phase, the factors ϵ_w and ϵ_n are modified to

$$\epsilon_w(i_a) = \epsilon_w \left(\frac{\epsilon_w^f}{\epsilon_w} \right)^{i_a/t_a}, \quad \epsilon_n(i_a) = \epsilon_n \left(\frac{\epsilon_n^f}{\epsilon_n} \right)^{i_a/t_a}.$$

where t_a is the total number of iterations and i_a the current iteration of the annealing phase. ϵ_w, ϵ_n are the constants from the growing phase and $\epsilon_w^f, \epsilon_n^f$ the terminal values of $\epsilon_w(i_a)$ and $\epsilon_n(i_a)$ respectively. Typical values for the parameters are

$$\epsilon_w = 0.3,\ \epsilon_w^f = 0.01,\ \epsilon_n = 0.06,\ \epsilon_n^f = 0.00006,$$

$$\tau = 300,\ a_{max} = 50,\ \alpha = 0.5,\ \beta = 0.995.$$

The Annealing Growing Neural-Gas combines the versatility of the Growing Neural-Gas with the quality of the Neural-Gas algorithm and is especially useful in applications where the desired size of a codebook is not know but a maximal practical limit exists (e.g. due to processing speed).

Initialize $\mathcal{W}$ with $|\mathcal{W}| = 2$
Initialize $c(\mathbf{w}_1, \mathbf{w}_2) = e(\mathbf{w}_1) = e(\mathbf{w}_2) = 0$, $i = 0$
Repeat
  Choose $\mathbf{x}$ randomly from $\mathcal{X}$, $\Delta i = 1$
  $\mathbf{w}_1 = \arg\min_{\mathbf{w} \in \mathcal{W}} \|\mathbf{x} - \mathbf{w}\|$, $\mathbf{w}_2 = \arg\min_{\mathbf{w} \in \mathcal{W} \setminus \{\mathbf{w}_1\}} \|\mathbf{x} - \mathbf{w}\|$
  Cell.Update.Connectivity($\mathbf{w}_1, \mathcal{T}(\mathbf{w}_1)$)
  Cell.Update.Weights($\mathbf{w}_1, \mathcal{T}(\mathbf{w}_1, \mathbf{x})$)
  Cell.Remove($\mathcal{W}$)
  If mod $(i, \tau) = 0$ **then** Cell.Add($\mathcal{W}$)
  $e(\mathbf{w}) = \beta e(\mathbf{w})$ $\forall \mathbf{w} \in \mathcal{W}$
Until Stopping criterion satisfied

Function Cell.Update.Connectivity($\mathbf{w}_1, \mathcal{T}(\mathbf{w}_1)$)
  $\Delta a(\mathbf{w}_1, \mathbf{w}) = 1$ $\forall \mathbf{w} \in \mathcal{T}(\mathbf{w}_1)$
  $a(\mathbf{w}_1, \mathbf{w}_2) = 0$, $c(\mathbf{w}_1, \mathbf{w}_2) = 1$
  $\forall \mathbf{w} \in \mathcal{T}(\mathbf{w}_1)$: **If** $a(\mathbf{w}_1, \mathbf{w}) > a_{max}$ **then** $c(\mathbf{w}_1, \mathbf{w}) = 0$

Function Cell.Update.Weights($\mathbf{w}_1, \mathcal{T}(\mathbf{w}_1)$)
  $\Delta e(\mathbf{w}_1) = \|\mathbf{w}_1 - \mathbf{x}\|^2$
  $\Delta \mathbf{w}_1 = \epsilon_w(\mathbf{x} - \mathbf{w}_1)$
  $\Delta \mathbf{w} = \epsilon_n(\mathbf{x} - \mathbf{w})$ $\forall \mathbf{w} \in \mathcal{T}(\mathbf{w}_1)$

Function Cell.Remove($\mathcal{W}$)
  $\forall \mathbf{w} \in \mathcal{T}(\mathbf{w}_1))$: **If** $|\mathcal{T}(\mathbf{w})| = 0$ **then** $\mathcal{W} = \mathcal{W} \setminus \{\mathbf{w}\}$
  (remove a reference vector which has no edges)

Function Cell.Add($\mathcal{W}$)
  $\mathbf{w}_{e1} = \arg\max_{\mathbf{w} \in \mathcal{W}} e(\mathbf{w})$, $\mathbf{w}_{e2} = \arg\max_{\mathbf{w} \in \mathcal{T}(\mathbf{w}_{e1})} e(\mathbf{w})$
  $\mathbf{w}_{new} = \frac{1}{2}(\mathbf{w}_{e1} + \mathbf{w}_{e2})$, $\mathcal{W} = \mathcal{W} \cup \{\mathbf{w}_{new}\}$
  $c(\mathbf{w}_{e1}, \mathbf{w}_{e2}) = 0$, $c(\mathbf{w}_{e1}, \mathbf{w}_{new}) = c(\mathbf{w}_{e2}, \mathbf{w}_{new}) = 1$
  $a(\mathbf{w}_{e1}, \mathbf{w}_{new}) = a(\mathbf{w}_{e2}, \mathbf{w}_{new}) = 0$
  $e(\mathbf{w}_{new}) = e(\mathbf{w}_{e1}) = \alpha e(\mathbf{w}_{e1})$, $e(\mathbf{w}_{e2}) = \alpha e(\mathbf{w}_{e2})$

Figure 5.6: Pseudo-code of the Growing Neural-Gas algorithm.

5.6.1 Augmented Data Clustering

In this dissertation we will use clustering techniques to reduce the dimension of datasets representing the attracting set of a dynamical system. In the continuous time case, in addition to the spatial information $\mathbf{x} \in \mathcal{X}$ we have temporal information $\dot{\mathbf{x}} \in \dot{\mathcal{X}}$ available to construct an optimal codebook. Depending on the task for which the codebook is used the utilization of this additional information might improve performance. As an example consider Figure 5.7 which shows a region in phase space where diverging trajectories are close in space. Using only spatial information for clustering, points belonging to trajectories moving to separate directions are possibly associated to the same reference vector (see Figure 5.7a). If the codebook is used for the approximation of temporal as well as spatial information, such a clustering is inappropriate. In time series prediction for example, local maps from current to future data can be constructed based on data clustering techniques (see e.g. [MBS93], [DSHK99]). Typically these methods construct the predictions based on the points included in a Voronoi cell, and using clustering based on spatial information alone might lead to suboptimal results. The problem can be resolved by including temporal information into the codebook approximation procedure (see Figure 5.7b).

The idea of spatio-temporal clustering was originally introduced by D. Hundley [Hun98] and applied to improve codebooks used to reconstruct the flow of dynamical systems. To generalize the idea of spatio-temporal clustering, let us assume that for each input vector $\mathbf{x} \in \mathcal{X}$ a vector $\mathbf{c} \in \mathcal{C}_x$ is available, which contains additional information characterizing the state $\mathbf{x}$. This could for example be the derivative of $\mathbf{x}$, in which case $\mathbf{c} = \dot{\mathbf{x}}$. We will call the information contained in the set $\mathcal{C}_x$ *class information*. If it is nonempty, the set $\mathcal{C}_x$ is assumed to be of the same size as the set $\mathcal{X}$, but of possibly smaller or larger dimension. If additional

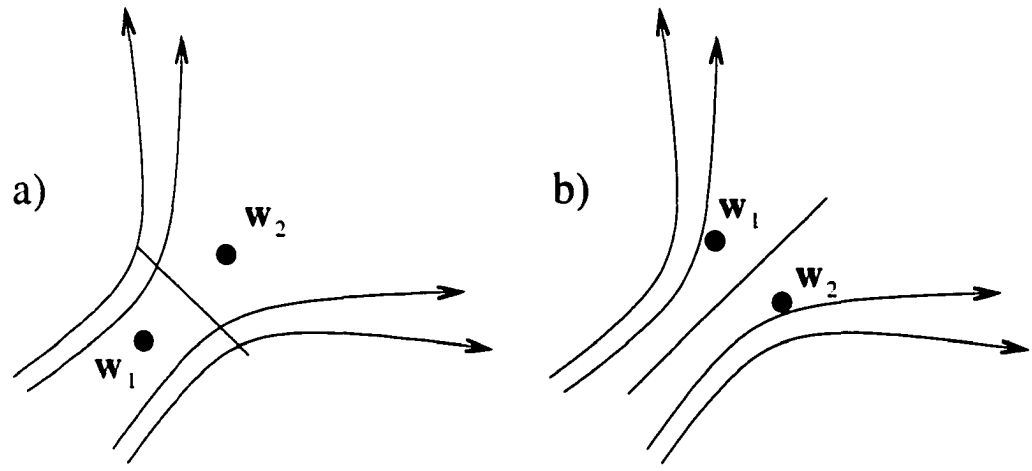


Figure 5.7: Comparison of clustering of spatially close trajectories with different temporal flow. a) Using only spatial information can lead to clusters which contain diverging trajectories. b) Including the temporal information can lead to clusters which better resolve the temporal information.

information is available and it is desired to construct a codebook which represents this additional information. this can be achieved by clustering the *augmented data*

$$\hat{\mathbf{x}} = \begin{pmatrix} \mathbf{x} \\ \gamma \mathbf{c} \end{pmatrix}, \quad \hat{\mathbf{x}} \in \hat{\mathcal{X}} = \mathcal{X} \times \mathcal{C}_x.$$

This set is clustered by an augmented set of reference vectors

$$\hat{\mathbf{w}} = \begin{pmatrix} \mathbf{w} \\ \mathbf{w}_c \end{pmatrix}, \quad \hat{\mathbf{w}} \in \hat{\mathcal{W}} = \mathcal{W} \times \mathcal{W}_c,$$

with $\mathcal{W}$ approximating $\mathcal{X}$ and $\mathcal{W}_c$ approximating $\mathcal{C}_x$. The parameter γ is problem dependent and controls the strength with which the class information influences the placing of the clusters. For $\gamma = 0$, clustering the augmented data is equivalent to clustering the set $\mathcal{W}$ without the use of class information.

Using the augmented data all clustering algorithms can be used to produce a codebook $\hat{\mathcal{W}} = \mathcal{W} \times \mathcal{W}_c$ approximating $\hat{\mathcal{X}} = \mathcal{X} \times \mathcal{C}_x$. If the set $\mathcal{C}_x$ is nonempty and γ appropriately chosen. the codebook $\mathcal{W}$ will resolve the class information and can represent an improvement over a codebook which was approximated without using the available class information.

Consider the example dataset consisting of the four points

$$\mathbf{x}_1 = (0,0),\ \mathbf{x}_2 = (1,0),\ \mathbf{x}_3 = (0,1),\ \mathbf{x}_4 = (1,1),$$

shown in Figure 5.8. Clustering these four points by two reference vectors based on spatial information alone will result in two possible, equal likely codebooks as shown in Figure 5.8a. If the points have additional characteristics, like the direction in which an attached vector points, then the data can be augmented. If $c = 1$ denotes a vector pointing to the right and $c = -1$ a vector pointing to the left, then the augmented dataset becomes

$$\hat{\mathbf{x}}_1 = (0,0,\gamma),\ \hat{\mathbf{x}}_2 = (1,0,\gamma),\ \hat{\mathbf{x}}_3 = (0,1,-\gamma),\ \hat{\mathbf{x}}_4 = (1,1,-\gamma).$$

A clustering of the augmented dataset with $\gamma \neq 0$ will result in the codebook shown in Figure 5.8b with 100% probability.

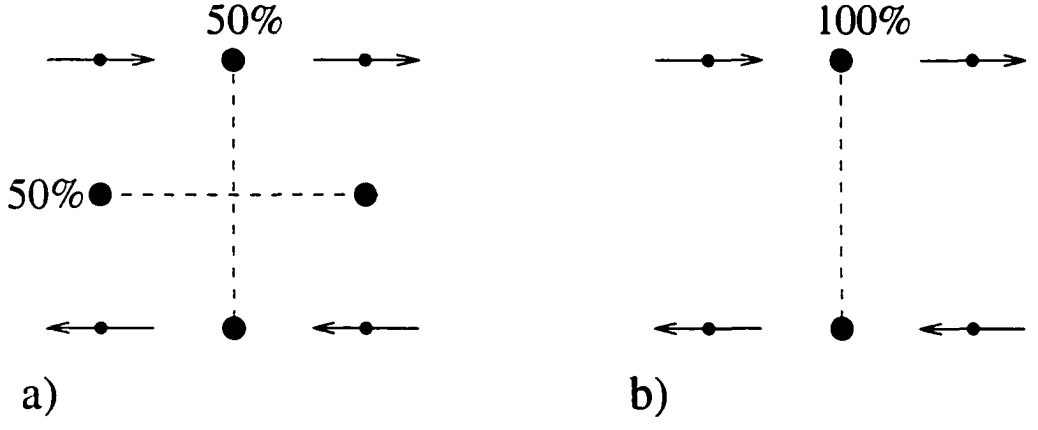


Figure 5.8: Resolution of class information through augmented data clustering. The small dots represent data points, the large dots reference vectors. a) Four points are clustered through 2 reference vectors based on spatial information alone. Codebooks with clusters along the horizontal or vertical line are both equal likely. b) If class information is included, the upper two points are distinguishable from the lower two points and the clustering will result in the codebook along the vertical line with 100% probability.

5.7 Comparisons

In this section we show comparisons of the discussed algorithms on the basis of several different datasets. We measure the performance on the basis of the distortion error. For the Neural-Gas (NG) and k-means algorithm, 40,000 iterations were performed. The Growing Neural-Gas (GNG) algorithm was terminated if the desired codebook size was reached. The annealing phase of the Annealing Growing Neural-Gas (AGNG) algorithm was set to $t_a = 40,000$ iterations.

5.7.1 Dataset 1

The first dataset consists of 1000 data points distributed uniformly over a circle of radius 1. The result of clustering with the LBG algorithm and 10 reference vectors is shown in Figure 5.9. All distortion errors are summarized in Table 5.1. The LBG, k-means, NG and AGNG algorithms lead to similar results. Terminating the GNG algorithm at $|\mathcal{W}| = 10$ leads to a suboptimal codebook.

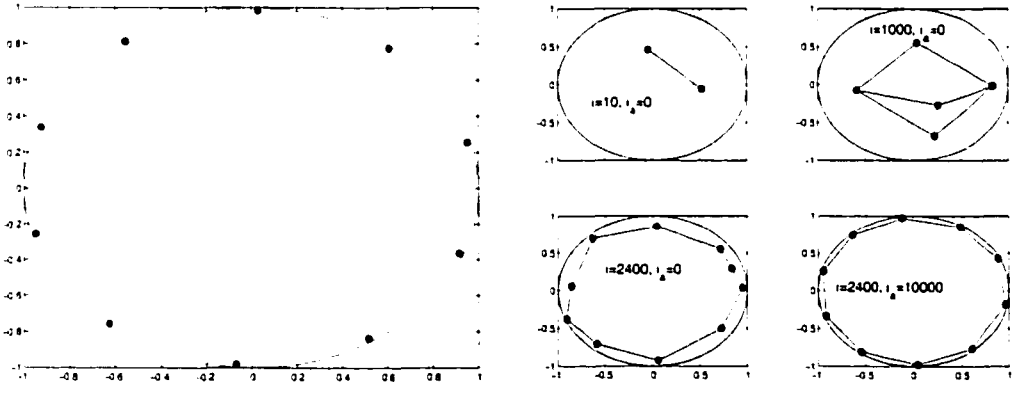


Figure 5.9: The dataset 1 consists of 1000 points randomly distributed over a circle of radius 1. a) 10 reference vectors distributed with the LBG algorithm. b) Growing phase of AGNG codebook. The codebook of the AGNG at size 10 before annealing ($i = 2400, i_a = 0$) is equivalent to the codebook produced by GNG with $|\mathcal{W}| = 10$ as stopping criterion and suboptimal. The annealing phase improves the codebook ($i = 2400, i_a = 10000$).

Dataset	$\|\mathcal{W}\|$	E_{LBG}	E_k	E_{NG}	E_{GNG}	E_{AGNG}
1	10	0.1574	0.1582	0.1577	0.1946	0.1578
2	100	0.4847	0.4782	0.4296	0.5270	0.4295
3	100	0.0106	0.0114	0.0091	0.0113	0.0093
3	10	0.0605	0.0616	0.0606	0.0637	0.0604
3 (ST)	10	0.1719	0.1682	0.1699	0.2192	0.0954
3 (W)	10	0.0779	0.0820	0.1011	0.1639	0.0890

Table 5.1: Summary of the distortion errors E for the different datasets and codebooks constructed with different algorithms. $|\mathcal{W}|$ denotes the number of reference vectors. (ST) denotes augmented data. (W) denotes that the error is only with respect to the spatial part of the augmented data.

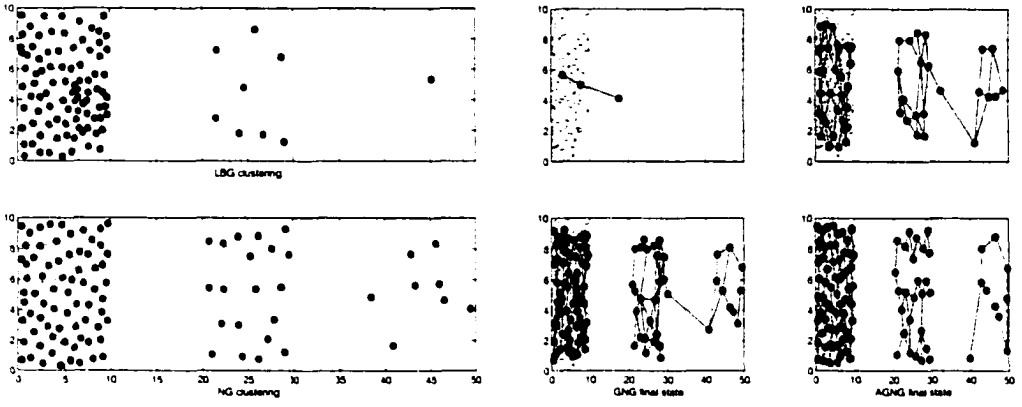
5.7.2 Dataset 2

The second dataset consists of 1110 points distributed over three distinct square regions as shown in Figure 5.10. The first region contains 1000 points, the second 100 and the third 10. The complete dataset was clustered with 100 reference vectors. The codebook obtained with the LBG algorithm is shown in the upper part and the one obtained with the NG algorithm in the lower part of Figure 5.10a. Several stages of the AGNG algorithm are shown in Figure 5.10b. The LBG approximates the local density of the distribution well by placing 90 reference vectors in the first, 9 in the second and 1 in the third region. The Neural-Gas type algorithms on the other hand tend to place more clusters in regions of low density. As seen from Table 5.1, the NG and AGNG approximate codebooks with distortion errors considerably lower than LBG or k-means. Again, the AGNG algorithm reduces the distortion error in comparison to the GNG algorithm.

5.7.3 Dataset 3

The third dataset consists of the dataset

$$\mathbf{x}_n = \begin{pmatrix} \sin(4n\pi/N) \\ (1/20)\cos(4n\pi/N) \end{pmatrix}, \quad n = 0, \cdots, N. \tag{5.3}$$



a) b)

Figure 5.10: Comparison of the algorithms on dataset 2 consisting of 1110 data points. 1000 points are randomly distributed in the first region $0 \leq x \leq 10$. 100 over the region $20 \leq x \leq 30$ and 10 over the region $40 \leq x \leq 50$. a) Upper figure: the LBG algorithm approximates the local density of the probability distribution by placing approximately 91 out of 100 reference vectors in the first, 8 in the second and 1 in the third region. Lower figure: The NG algorithm places more clusters in regions of lower density. b) Several states of the AGNG algorithm, including the final state of the GNG algorithm. AGNG improves the GNG distribution of reference vectors.

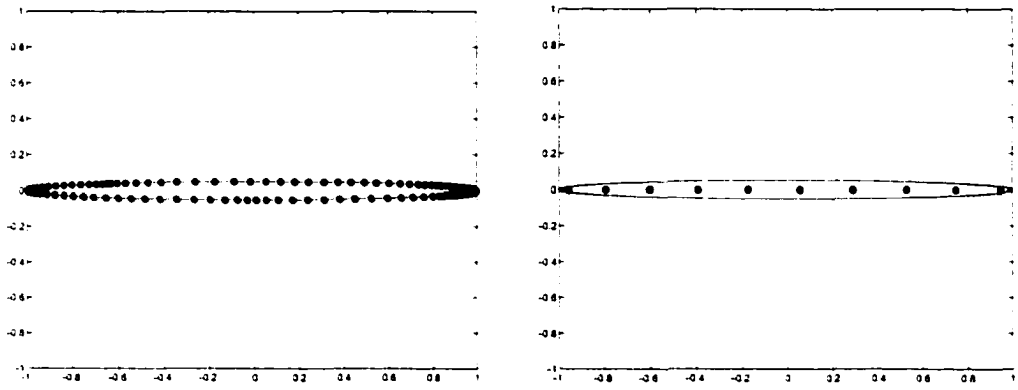
with $N = 1000$, representing an ellipse with length of the large axis being 20 times that of the small axis.

Spatial Clustering

First 100 reference vectors were distributed over this dataset with all algorithms using only spatial information (see Figure 5.11a). All codebooks lead to similar distortion errors. However, by using only 10 reference vectors, all algorithms fail to resolve the elliptical structure of the dataset (see Figure 5.11b).

Spatio-temporal Clustering

The failure to resolve the elliptical structure might have undesirable consequences in certain applications. For example if the codebook is intended to be an approximation to the flow of a dynamical system with the intention to use it for



a) b)

Figure 5.11: An ellipsoid dataset consisting of 1000 points. a) 100 and b) 10 reference vectors were distributed with the LBG algorithm. The elliptical shape of the dataset can not be resolved with a small number of reference vectors.

time series prediction or control, the approximated codebook from Figure 5.11b will not lead to optimal results. Including temporal information describing the local flow of the trajectory, can resolve this problem.[4] To this end, the dataset (5.3) was extended to the set

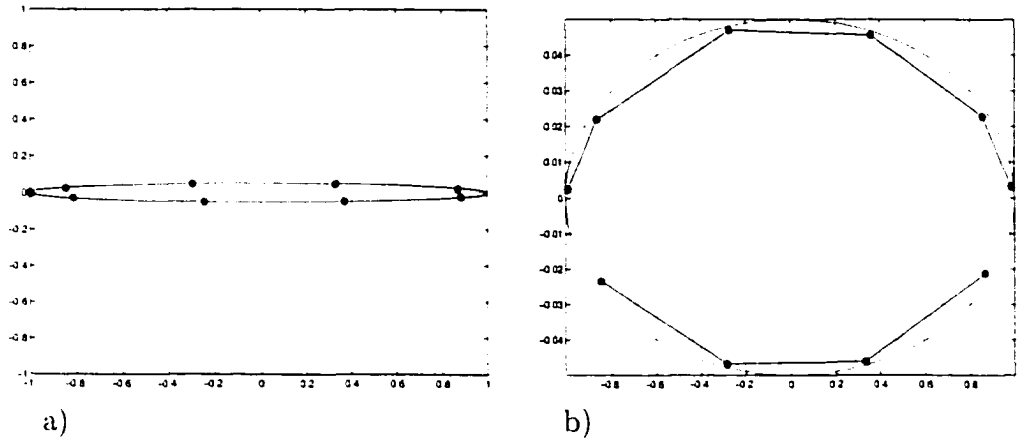
$$\begin{pmatrix} \mathbf{x}_n \\ \dot{\mathbf{x}}_n / \|\dot{\mathbf{x}}_n\| \end{pmatrix}, \quad n = 0, 2, \cdots, N-1$$

with $\dot{\mathbf{x}}_n = \mathbf{x}_{n+1} - \mathbf{x}_n$. The clustering of this dataset with the LBG algorithm is shown in Figure 5.12a and with the AGNG algorithm in Figure 5.12b. We see that the elliptical structure of the dataset can be resolved if temporal information is included into the clustering procedure.

5.8 Conclusions

In this chapter we reviewed four different vector quantization algorithms: the LBG, k-means, Neural-Gas and the Growing Neural-Gas algorithm. All of them

[4]We confirmed this statement by applying the time series prediction method of Martinetz *et al.* [MBS93] based on the NG algorithm to the prediction of dataset 3 and Lorenz data and found that spatio-temporal clustering leads to a considerably lower mean-squared prediction error in both cases. A detailed investigation will be presented elsewhere.



a) b)

Figure 5.12: Resolving of the elliptical structure by including temporal information into the clustering algorithm. a) LBG clustering. b) AGNG clustering with 10 reference vectors. Note the separation of the classes represented by the disconnection of the connectivity structure.

have different advantages and disadvantages and will be preferable in different situations. Throughout the remaining chapters of this dissertation. we will use the Neural-Gas and the Growing Neural-Gas algorithms for the purpose of vector quantization. The Neural-Gas algorithm was shown to lead to smaller quantization errors as the other three algorithms. Furthermore. it can be considered topology preserving if a Delaunay triangulation is induced after the codebook is approximated [Fri95a]. The Neural-Gas algorithm is certainly slow compared to other vector quantization techniques and in applications where speed is of concern other algorithms might be preferred. Over the widely applied self-organizing Kohonen map [Koh95] it offers the advantage of being applicable in any dimension.

The control algorithm discussed in the next chapter. which treats vector quantization and control in separate steps, does not heavily rely on the quality of the codebook and our choice of vector quantization algorithms is more due to personal preference than anything else. In the last chapter of this dissertation we will combine vector quantization and reinforcement learning with the goal to find a minimum sufficient codebook for the control task. In this context grid growing

algorithms are necessary. The important feature of these algorithms is that they allow to place new clusters in desired regions.

As mentioned before, codebooks can sometimes be improved if additional information is used to augment the data. Although we only briefly commented on time series prediction in this context and do not apply class clustering in the remainder of this dissertation, we are convinced that these ideas might be relevant in certain applications of our control algorithm. In principle, for dynamical data, whenever regions in phase space occur with high variance of trajectory slopes, spatio-temporal clustering can lead to codebooks more suitable for applications such as time series prediction or control. As an example of such a situation consider the Lorenz attractor and the region between the wings (see Figure 2.1c) which is better resolved through spatio-temporal clustering as demonstrated in [Hun98].

Chapter 6

OPTIMAL CHAOS CONTROL THROUGH REINFORCEMENT LEARNING

> ... and those states, where none of the available control actions is appropriate to reach the target, shall be directed towards another state which is more close to the target.
>
> (R. Der and M. Herrmann, 1994, p. 2474)

In this chapter we will introduce a general purpose time-discrete chaos control algorithm combining reinforcement learning and vector quantization. We will present parametric and impulsive control applications to low-dimensional chaotic systems and show its application to the targeting and tracking problem.

6.1 Introduction

Although the system dynamics can be continuous, the controller introduced in this thesis works discretely in time. At time n information $\mathbf{x}_n$ about the systems state is fed into the controller which will apply a control signal on the basis of this information. If the dynamics are continuous, $\mathbf{x}_n$ can represent e.g. observations in a surface-of-section as discussed in Section 5.1. The observed system dynamics are then formally described by an (unknown) map

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

with a vector of system parameters $\mathbf{p}$.

In terms of optimal control theory, as discussed in Subsection 4.1.2, we can summarize the scheme of a general purpose chaos controller as follows: in state

$\mathbf{x}_n \in \mathcal{X}$ the controller chooses control $u_n \in \mathcal{U}$ according to a certain control policy $\pi(\mathbf{x}_n)$. The optimal control policy $\pi^*(\mathbf{x}_n)$ maximizes a reward function. For the purpose of chaos control the optimal policy is expected to (a) stabilize a desired state $\mathbf{x}_f$ on the attractor $\mathcal{A}$, (b) reach a neighborhood of $\mathbf{x}_f$ from anywhere on $\mathcal{A}$ in a minimum number of iterations, and (c) apply only control signals leading to small system perturbations. Furthermore, if $\mathbf{x}_f$ corresponds to an embedded UPO of $\mathcal{A}$ and if no noise is present, then the perturbations should vanish once $\mathbf{x}_f$ is stabilized.

For a general purpose chaos controller, it is in addition required that the optimal policy can be established (a) model independent, (b) under noisy and non-stationary conditions, (c) if either parametric or impulsive control is desired, and (d) if systems are high-dimensional or spatially extended.

If analytical knowledge on system dynamics would be available then methods from optimal control theory [Ber87] could be used to find the optimal control policy. Since in real world applications this is often not the case, we require the algorithm to use no other information besides the states $\mathbf{x}_n$ and, in case of parametric control, the value of $\mathbf{p}$. Reinforcement learning, reviewed in Chapter 4, has been shown to solve the optimal control problem [Sin94, BT96]. Der and Herrmann [DH94] were the first to suggest to apply reinforcement learning to chaos control. They used Q-learning to approximate a control policy which targets an UPO along a quasi-optimal path. The policy was approximated in a reduced space constructed by a Kohonen self-organizing network. The drawbacks of the approach of Der and Herrmann are that exact knowledge of an UPO is required and that the algorithm operates only through parameter perturbations. Moreover the algorithm was tested only for the logistic map.

The control methods introduced in this and the following chapters also rely on the idea of approximating an optimal control policy in a reduced phase space

through reinforcement learning, but generalize and improve the approach of Der and Herrmann in several aspects [GD99]. In particular the UPO is not required to be known and it is applicable for parametric as well as impulsive control. Although the UPO can be determined by approximating the system dynamics, in order to solve the tracking problem it is desirable to formulate the algorithm such that it also estimates the UPO while the policy is approximated. We will demonstrate performance of the method for a variety of low-dimensional systems and a high-dimensional hyperchaotic Lorenz system. In the following chapters we will apply it to a spatially extended system [GD00a] and a multistable rotor [GD00b, GD01]. In this chapter we treat vector quantization, reviewed in Chapter 5, and reinforcement learning as separate parts of the algorithm and set the control set $\mathcal{U}(\mathbf{w})$ associated with a reduced state $\mathbf{w}$ equal to some fixed $|\mathcal{U}|$ for all $\mathbf{w}$. In the last chapter these steps are combined and variable control sets are allowed, which will lead to an improvement of the algorithm in certain applications.

6.2 Learning Chaos Control

In order to formulate the chaos control problem as a reinforcement learning problem we need to introduce (a) an input function $I : \mathcal{X} \to \mathcal{W}$, (b) a discrete set of control actions $u \in \mathcal{U}$, and (c) a set of scalar reinforcement signals $r \in \mathbb{R}$.

6.2.1 The Input Function

Due to the chaotic nature of the systems considered, any unperturbed orbit $\{\mathbf{x}_n, n \in \mathbb{N}\}$ will be dense in $\mathcal{X}$. Since the temporal-difference learning algorithms described in Subsection 4.1.3 require formally a finite state Markov process, a discretized version $\mathcal{W}$ of the state space $\mathcal{X}$ is needed. If $\mathcal{X}$ is a one-dimensional interval this can simply be achieved by covering $\mathcal{X}$ uniformly by small intervals of which the midpoints $\mathbf{w}^{(i)}$ serve as finite representations of $\mathcal{X}$. If $\mathcal{X}$ is higher-dimensional, a vector quantization is performed based on observations of $\mathbf{x}_n$ over

a time window in which the system behaves chaotically. Specifically we employ the Neural-Gas vector quantization technique (see [MBS93] and Chapter 5) which extracts a finite set of reference vectors $\mathbf{w} \in \mathcal{W} = \{\mathbf{w}^{(1)}, \cdots, \mathbf{w}^{(N)}\}$ representing the state space in their Voronoi cells. The advantage of this technique is that the distributions of the $\mathbf{w}^{(i)}$ are matched optimally to the invariant measure of the attractor [MS94]. Alternative techniques to obtain a finite representation of $\mathcal{X}$, as reviewed in Chapter 5, can also be used. Which technique is used is not relevant for the control algorithm as long as the set $\mathcal{W}$ is sufficiently fine.[1] In any case, the projection $I : \mathcal{X} \rightarrow \mathcal{W}$ of $\mathcal{X}$ onto the representation space $\mathcal{W}$ is defined by

$$\mathbf{w}(\mathbf{x}) := I(\mathbf{x}) = \arg\min_{\mathbf{w} \in \mathcal{W}} ||\mathbf{w} - \mathbf{x}||. \tag{6.1}$$

Based on the representation $\mathcal{W}$ we introduce a state-action value function $Q(\mathbf{w}, u)$ which reduces the reinforcement learning problem to the approximation of the optimal state-action value function $Q^*(\mathbf{w}, u)$.

6.2.2 The Control Set

Concerning the control set U we choose a fixed set of $m + 1$ real values $\mathcal{U} = \{0, u_1, \cdots, u_m\}$, $u_i \in \mathbb{R}$. In each state $\mathbf{w} \in \mathcal{W}$ one of the controls $u \in \mathcal{U}$ can be applied. Given $\mathbf{w}_n$, a control u_n will be chosen from $\mathcal{U}$ according to the current policy (e.g. ϵ-greedy or greedy as described in Chapter 4) based on the values $Q(\mathbf{w}_n, u)$. In this and the following two chapters the set $\mathcal{U}$ will be fixed and equal for all $\mathbf{w}$. Only in the last chapter we will consider state dependent control sets $\mathcal{U}(\mathbf{w})$.

---

[1] In the last chapter we will apply a growing grid VQ to find a minimal sufficient representation.

6.2.3 Formulation of Goal

Observations of the states $\mathbf{x}_n$ result in a sequence $\mathbf{w}_n = \mathbf{w}(\mathbf{x}_n) \in \mathcal{W}$. To allow the control algorithm to solve the tracking problem as discussed in Chapter 3, we formulate the goal of the control problem in terms of this sequence. If a fixed point is to be stabilized, the criterion which tests if the goal was achieved is $\mathbf{w}_n = \mathbf{w}_{n+1}$. For a period two UPO it is $\mathbf{w}_n = \mathbf{w}_{n+2}$, $\mathbf{w}_n \neq \mathbf{w}_{n+1}$. In this case one formally may consider the Cartesian product $\mathcal{W} \times \mathcal{W}$ as a Markov state space and interpret $(\mathbf{w}_{n-1}, \mathbf{w}_n)$ as a single state. One proceeds analogously if higher order UPOs are to be stabilized. Note that no knowledge about the location of the fixed point or UPO is required. The control algorithm will first identify locations of UPOs and then approximate an optimal strategy for reaching them in an optimal way through updating of the associated state-action values $Q(\mathbf{w}_n, u_n)$. For targeting the goal is simply $\mathbf{w}_n = \mathbf{w}^*$, where $\mathbf{w}^*$ is the reference vector closest to the target state $\mathbf{x}^*$. In the form presented here, the algorithm can only distinguish orbits based on their period. If more orbits of a prescribed period are present the algorithm is able to identify these different orbits, from which a desired orbit can be selected and stabilized by defining the corresponding weight vector as the goal location. Without intervention the algorithm will eventually stabilize the UPO which is "easiest" to control.

6.2.4 Choice of Reinforcement Signals

Reinforcement learning guides the learning process through immediate reinforcement signals (or rewards) on which the state-action value function updates depend. The agent seeks to maximize its rewards over time and the rewards have to be chosen such that maximizing them will achieve the goal of the control application [SB98].

For the purpose of chaos control, a first choice for a reward function might be $r_{n+1} = -\|\mathbf{w}_{n+1} - \mathbf{w}_n\|$, if control of a period one UPO is desired. Although this led to successful control of period one UPOs, generalizing this special type of reward function to higher order UPOs causes some problems.

In reinforcement learning, motivated by learning theory, one typically provides scalar good/bad/neutral rewards in the form of +1/-1/0 signals. Der and Herrmann [DH94] suggested to use

$$r_{n+1} = \begin{cases} 1 & \text{if } \mathbf{x}_{n+1} = \mathbf{x}_f \\ 0 & \text{otherwise.} \end{cases} \tag{R1}$$

This type of rewards requires previous knowledge about the desired state. To allow tracking of UPOs we can modify (R1) to

$$r_{n+1} = \begin{cases} 1 & \text{if goal achieved} \\ 0 & \text{otherwise.} \end{cases} \tag{R2}$$

and define the goal through the set of reference vectors as described in the last subsection. The reinforcement signals (R1) and (R2) do not punish unsuccessful actions and did not perform well in non-stationary environments. Therefore instead we suggest

$$r_{n+1} = \begin{cases} 1 & \text{if goal achieved} \\ -r_p & \text{otherwise.} \end{cases} \tag{R3}$$

with $0 < r_p < 1$. With the reward function (R3) the algorithm performed successful in stationary and non-stationary environments. We typically observed best results for $r_p < 0.5$. We also found the following strategy to be successful in all applications:

$$\begin{aligned} &\text{If } \mathbf{w}_{n+1} \text{ is not a goal state: } r_{n+1} = -1 \text{ and update } Q(\mathbf{w}_n, u_n), \\ &\text{otherwise set } Q(\mathbf{w}_n, u_n) = 0. \end{aligned} \tag{R4}$$

In other words, we only punish unsuccessful control actions, while we give successful actions the highest value. This has the advantage that no additional parameter r_p comes into play.

In the presented applications we mainly used the reward functions (R3) with $r_p = 0.5$ and (R4) and obtained similar results for both of these functions. We will formulate the algorithm only for (R3), the extension to (R4) or any other choice of reward function is straightforward. In the last chapter we will modify (R3) and (R4) by punishing with $-r_p - |u_n|$ instead of $-r_p$ to encourage the agent to apply the smallest successful perturbation.

6.2.5 The Controlled Dynamics

Given an optimal state-action value function $Q^*(\mathbf{w}, u)$ and a set of reference vectors $\mathcal{W}$, the controlled dynamics can be written as,

$$\begin{aligned} \mathbf{w}_n &= \mathbf{w}(\mathbf{x}_n)\ ,\ u_n = \arg\max_{u \in \mathcal{U}} Q^*(\mathbf{w}_n, u) \\ \mathbf{x}_{n+1} &= \mathbf{f}(\mathbf{x}_n + u_n \Delta\mathbf{x}, \mathbf{p}_0 + u_n \Delta\mathbf{p}). \end{aligned} \tag{6.2}$$

The arguments in $\mathbf{f}$ reflect the specific form of the control action. If only parameter perturbations are applied (parametric control) we set $\Delta\mathbf{x} = 0$, while $\Delta\mathbf{p}$ is the coordinate vector that selects the particular component of $\mathbf{p}$ chosen to be perturbed. Analogously, if proportional pulses are applied to system variables (impulsive control), $\Delta\mathbf{p} = \mathbf{0}$ and $\Delta\mathbf{x}$ is the coordinate vector of the perturbed component of $\mathbf{x}$. Clearly one may consider also combinations of these approaches or more general perturbations described by matrix operations as well as multi-parameter control through the introduction of additional value functions, however, in this dissertation we examine only the above two simplest types of control perturbations. We emphasize that only the iterated states $\mathbf{x}_n$ are used in the control algorithm, knowledge of $\mathbf{f}$ is not required. In numerical experiments as described in the following, the $\mathbf{x}_n$ are of course computed from a given $\mathbf{f}$, however in "real world applications" they would be observed experimentally.

The optimal state-action value function is approximated using Q- and Sarsa-learning through interaction of the controller with the system. All possible controls

u are tested and their values in terms of rewards are used as feedback to update the corresponding state-action values $Q(\mathbf{w}(\mathbf{x}), u)$. In the case studies of Section 6.3 it turned out that Q-learning typically performed slightly better than Sarsa-learning. In the following chapters we will therefore exclusively apply Q-learning.

We distinguish between *on-line* and *off-line control.* In off-line control, learning is performed ϵ-greedy over many episodes and after termination of each episode the system state is reset to a new, randomly chosen initial value.[2] This enables the controller to find a good approximation to the globally optimal policy Q^*. In situations where one is interested in controlling a running system as quickly as possible one does not need a globally optimal policy. In these situations we perform control on-line. In on-line control, learning is performed deterministic and the system is not reset to a new state after termination of an episode. Q-values are updated *on the fly*, until the desired state is stabilized. In on-line control the controller will find a policy with local character, allowing stabilization of the desired state but not necessarily as quickly as possible from any initial condition.

However we will frequently see that an on-line approximated policy considerably reduces transient times compared to an uncontrolled system and that the off-line approximated policy does not considerably improve performance. The reason for this is two-fold. First, a chaotic system has the natural property to explore the whole state space quickly. Thus for systems not easily controlled, the whole state space will be sufficiently explored until on-line control is established. Second, we punish unsuccessful controls and represent the phase space by a finite discrete

[2]For a chaotic system this can simply be achieved by turning the controller off for a certain time. We note that the term off-line control is often used to refer to control applications where the optimal policy is found from the analytical equations describing the system dynamics. We use the term off-line control in a different context: off-line control here refers to situations where the controller is frequently turned off (or taken off-line) in order to start control from a new initial condition.

set. Thus even when using deterministic learning we can expect that a good control associated with a state $\mathbf{w}$ might be unsuccessful with small probability in some situations, specifically if noise is present. In other words, before a successful policy is found the controller will typically explore much of the phase space and action space even if deterministic on-line learning is performed. In particular for the spatially extended systems investigated in the following chapter and the rotor with considerable noise considered in Chapter 8 we apply only on-line control.

The complete chaos control algorithm based on either Q- or Sarsa learning is summarized in Figure 6.1 for on-line and off-line control of a fixed point and in Figure 6.2 for control of a period two fixed point. The generalization to the control of higher-order fixed points is straightforward.

Determine $I : \mathcal{X} \rightarrow \mathcal{W}$
Fix control set $\mathcal{U}$
Initialize $Q(\mathbf{w}, u)$: $Q(\mathbf{w}, u) = 0 \ \forall(\mathbf{w}, u), \mathbf{w} \in \mathcal{W}, u \in \mathcal{U}$
Initialize $\mathbf{x}$ randomly from $\mathcal{X}$
Find $\mathbf{w} := \mathbf{w}(\mathbf{x})$, $u \leftarrow Q(\mathbf{w}, \cdot)$

Repeat
$r' = 0$
For off-line control:
Initialize $\mathbf{x}$ randomly from $\mathcal{X}$
Find $\mathbf{w} := \mathbf{w}(\mathbf{x})$, $u \leftarrow Q(\mathbf{w}, \cdot)$
Decrease ϵ for ϵ-greedy policies

Repeat (per episode)
Apply u and observe $\mathbf{x}'$
Find $\mathbf{w}' := \mathbf{w}(\mathbf{x}')$, $u' \leftarrow Q(\mathbf{w}', \cdot)$

$$r' = \begin{cases} 1 & \text{if } \mathbf{w} = \mathbf{w}' \\ -r_p & \text{otherwise} \end{cases}$$

Sarsa-learning:
$\Delta Q(\mathbf{w}, u) = \beta[r' + \gamma Q(\mathbf{w}', u') - Q(\mathbf{w}, u)]$

Q-learning:
$\Delta Q(\mathbf{w}, u) = \beta[r' + \gamma \max_u Q(\mathbf{w}', u) - Q(\mathbf{w}, u)]$

$\mathbf{x} \leftarrow \mathbf{x}' : u \leftarrow u' : \mathbf{w} \leftarrow \mathbf{w}'$

until $r' = 1$

until goal stabilized (on-line) or $\epsilon = 0$ (off-line)

Figure 6.1: The chaos control algorithm for on-line and off-line control of a fixed point.

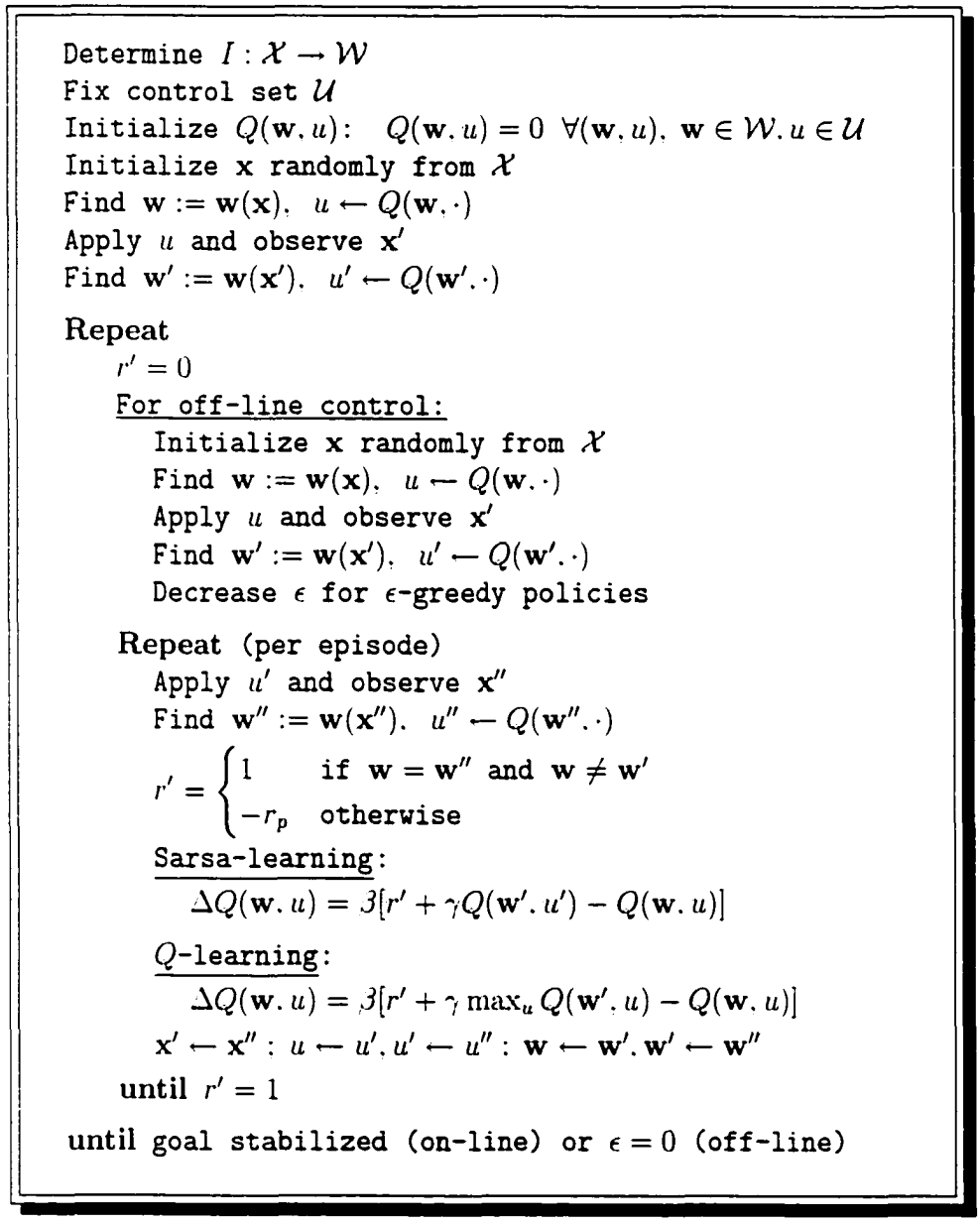


Figure 6.2: The chaos control algorithm for on-line and off-line control of a period two fixed point.

6.2.6 Transient Time

In Subsection 3.4.2 we argued that the transient time τ_t of a targeting method scales approximately as

$$\tau_t \sim \frac{\ln \tau_u}{\sigma_u}, \tag{6.3}$$

compared to a local control scheme with transient time τ_u and effective unstable Lyapunov exponent σ_u. In this subsection we show that an optimal control policy for a discrete control set $\mathcal{U}$ will scale similarly. To this end consider Figure 6.3. which is based on the ideas introduced by Schroer *et al.* [SO97] to prove their targeting scheme. Assume we start from a random source and want to reach a target in a region C. Without applying perturbations to the system outside C the average transient time of the system to reach C is assumed to be τ_u. Allowing small initial perturbations we would obtain a small control segment of length Δx centered at the source. If the perturbations are allowed to be continuous, the whole segment of length Δx would be initially accessible for control. The length of the control segment increases with n, the number of iterations, along the unstable direction with order $\sim e^{\sigma_u n}$ and will eventually reach the size of the attractor at which point we can with certainty find a trajectory from the initial control segment into the target region. This consideration led us in Subsection 3.4.2 to the estimate (6.3). In the reinforcement learning chaos control algorithm we do not apply continuous but only discrete perturbations from a set $\mathcal{U}$ of typically small size $|\mathcal{U}|$. If we would only allow controls at the source we could not reduce the transient time effectively. Instead the algorithm has the option to apply perturbations at every iteration. At the source, the possible perturbations $u \in \mathcal{U}$ allow the creation of $|\mathcal{U}|-1$ new source points with average distance $\delta x := \Delta x/(|\mathcal{U}| - 1)$ between neighboring points. At each iteration the size of δx will increase with order $\sim e^{\sigma_u}$, but at each iteration

the number of accessible points increases by a factor $|\mathcal{U}|$. As a result, the length of δx scales as

$$\delta x \sim \frac{e^{\sigma_u n}}{|\mathcal{U}|^n} = \left(\frac{e^{\sigma_u}}{|\mathcal{U}|}\right)^n.$$

We can conclude that the distance δx between neighboring points at each iteration will be of the order $\Delta x/2$ of the initial control segment or smaller, as long as $e^{\sigma_u} \leq |\mathcal{U}|$. Then, after τ_t iterations, when Δx has reached the size of the attractor, and assuming $\delta x < |C|$, we can be sure to have at least one accessible point contained in the region C. The complete set of points represents all possible accessible states by applying all possible $u \in \mathcal{U}$ at each iteration. The algorithm is designed to find a path which links the accessible point in C to the source point and we can expect the transient time of the algorithm to scale similar to (6.3).

A practical way to estimate transient times τ is to compute the average number of iterations λ from a random initial condition on the attractor $\mathcal{A}$ to the region C. λ_u will represent the transient estimate for the uncontrolled system, while λ_Q represents the estimate of τ for the system controlled through a greedy policy based on the state-action values Q. From the considerations above we therefore would expect

$$\lambda_Q \sim \frac{\ln \lambda_u}{\sigma_u}. \tag{6.4}$$

for an optimal control policy. However in (6.4) we ignored the fact that the set of created accessible points also leads to faster expansion of Δx over the attractor, i.e. Δx is not just stretched to $\Delta x e^{\sigma_u n}$, but has expanded to $\Delta x \sum_{i=0}^{n} e^{\sigma_u i}$ after n iterations. Using this we can find the estimate

$$\lambda_Q \sim \frac{1}{\sigma_u} \ln\left(1 - \lambda_u(1 - e^{\sigma_u})\right) - 1. \tag{6.5}$$

Note however, that in practice we are merely approximating the optimal policy and we must be content if $\lambda_Q \sim \ln \lambda_u$.

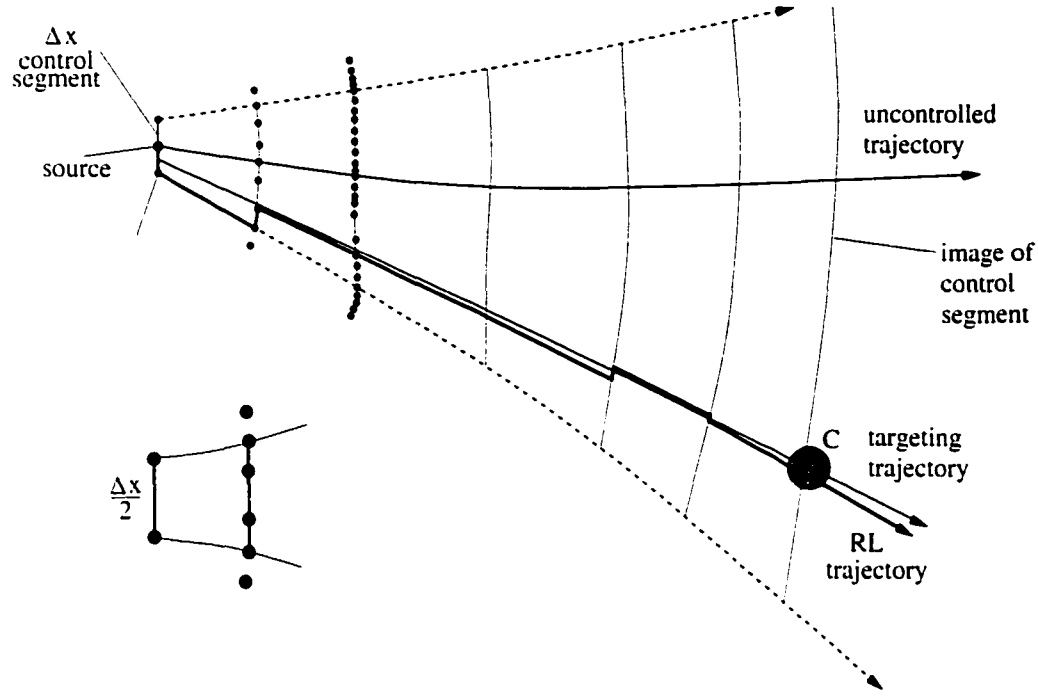


Figure 6.3: Transient time of reinforcement learning chaos control. See text for details.

6.3 Learning Parametric Control

In this section we present various results concerning the application of the algorithm introduced in the previous sections to the control and targeting of low-dimensional chaotic systems.

6.3.1 Off-line Control of the Logistic Map and the Hénon Map

We first present applications of our chaos control algorithm to low-dimensional chaotic maps. We consider the logistic map

$$x_{n+1} = p_0 \cdot x_n(1 - x_n). \tag{6.6}$$

which exhibits chaotic behaviour for $3.5694... < p_0 < 4$. The condition $x_{n+1} = x_n$ yields the nontrivial fixed point $x_f = 1 - 1/p_0$ ($x_f \approx 0.7368$ for $p_0 = 3.8$). The

second map we consider is the two-dimensional Hénon map

$$x_{n+1} = p_0 + b_0 x_{n-1} - x_n^2, \tag{6.7}$$

which behaves chaotically in a large range of the parameters b_0 and p_0 [Hén76]. We choose $b_0 = 0.3$, then the map exhibits chaotic behaviour in the range $1.1 \leq p_0 \leq 1.41$. Setting $\mathbf{x} = (x_n, x_{n-1})$, for $b_0 = 0.3$, the fixed points are

$$\mathbf{x}_{1,2} = \left\{ \frac{-0.7 \pm \sqrt{0.7^2 + 4p_0}}{2}, \frac{-0.7 \pm \sqrt{0.7^2 + 4p_0}}{2} \right\},$$

yielding $\mathbf{x}_{f1} \approx (0.8385, 0.8385)$ and $\mathbf{x}_{f2} \approx (-1.5385, -1.5385)$ when $p_0 = 1.29$.

We apply parametric perturbations $u \in \mathcal{U}$ with $\mathcal{U}_{log} = \{0, 0.1, -0.1\}$ for control of the logistic map and $\mathcal{U}_{Hen} = \{0, 0.025, -0.025\}$ for control of the Hénon map. Given an optimal state-action value function $Q^*(w, u)$ the controlled logistic map dynamics can be written as:

$$x_{n+1} = (p_0 + u_n) \cdot x_n(1 - x_n),$$
$$u_n = \arg\max_u Q^*(w(x_n), u),$$

and similarly for the Hénon map.

Concerning the set of reference vectors for control of the logistic map, since the data are one-dimensional and confined between 0 and 1, we choose 100 points uniformly distributed between 0 and 1 as set of reference points.

$$w \in \mathcal{W}_{log} = \{0.01, 0.02, \cdots, 1\}. \tag{6.8}$$

For the control of the Hénon map a set $\mathcal{W}_{Hen}$ of 400 reference vectors as shown in Figure 6.4 was used. The set was obtained through a Neural-Gas vector quantization of 5.000 data points ($p_0 = 1.29, b_0 = 0.3$).

The goal in this section is to stabilize a period-one fixed point of the maps. As a criterion for reaching a fixed point the condition $w(x_{n+1}) = w(x_n)$ ($\mathbf{w}(\mathbf{x}_{n+1}) = \mathbf{w}(\mathbf{x}_n)$) is used for control of the logistic (Hénon) map.

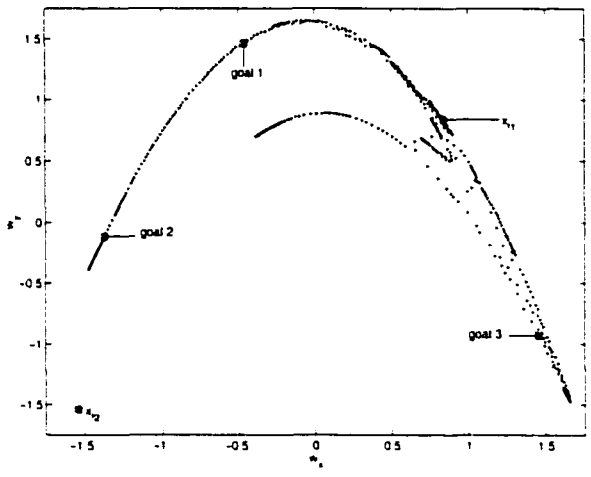


Figure 6.4: The set $\mathcal{W}_{Hen}$ of 400 reference vectors $\mathbf{w}$ used for the control of the Hénon map. The locations of the fixed points $\mathbf{x}_{f1}$ and $\mathbf{x}_{f2}$ as well as the goals for the targeting application of Subsection 6.3.3 are shown.

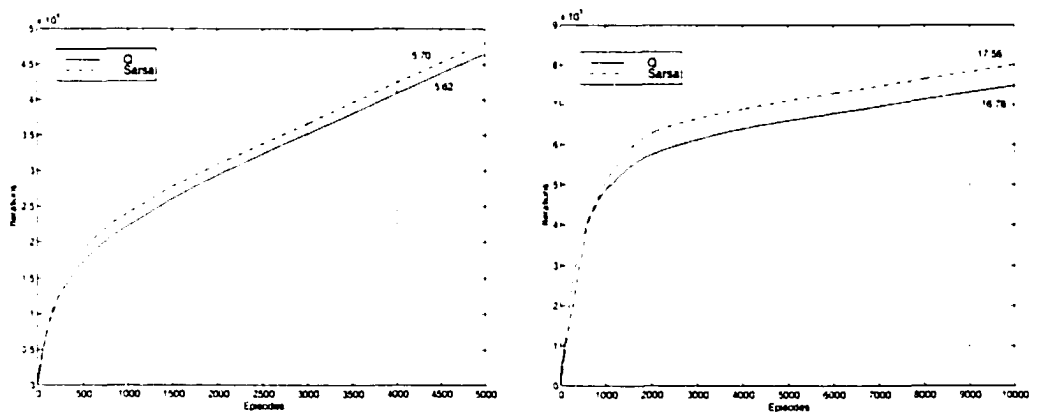
The optimal state-action value function $Q^*(\mathbf{w}, a)$ is approximated through ϵ-greedy off-line Q- and Sarsa-learning. The learning curves illustrating performance of the algorithms on the control task are shown in Figure 6.5 which shows the total number of iterations of the map as a function of the episodes. For control of the logistic map ϵ was slowly decreased to zero over a period of 4.000 episodes (9.000 episodes for control of the Hénon map). The decreasing slope of the graph shows that the goal is reached more and more quickly in each new episode as ϵ is decreased and demonstrates convergence of the policy. For the last 1.000 episodes we set $\epsilon = 0$. The slope of the graph above 4000 episodes is then a measure for λ_{Q^*}, the average number of iterations to reach the goal with the approximated policy Q^*. For control of the logistic map we obtained for Q-learning $\lambda^Q_{Q^*} = 5.62$ and for Sarsa-learning $\lambda^S_{Q^*} = 5.70$. while for the Hénon map $\lambda^Q_{Q^*} = 16.78$ and $\lambda^S_{Q^*} = 17.56$. For both maps, Q-learning led to better results and faster convergence.

To better compare λ for the controlled and the uncontrolled system, λ was averaged over 2.000 episodes starting from random initial values. For the logistic map an episode was terminated if $|x_n - x_f| < 0.005$. Choosing actions greedy

according to the approximated state-action value functions Q^*, we obtained $\lambda^Q_{Q\bullet} = 5.61$ and $\lambda^S_{Q\bullet} = 5.87$, compared to $\lambda_u = 151.34$ for the uncontrolled system.

For the Hénon map an episode was terminated if $||\mathbf{x}_n - \mathbf{x}_f|| < 0.02$ and we obtained $\lambda^Q_{Q\bullet} = 14.78$ and $\lambda^S_{Q\bullet} = 15.35$, while for the uncontrolled system $\lambda_u = 1414$.

The numerically observed transients are in good agreement with our estimates (6.4) or (6.5). For the logistic map with the chosen parameters it is $\sigma_u \approx 0.45$ [AFH94] and we would expect from (6.4) $\lambda_{Q\bullet} \sim 11.15$ and from (6.5) $\lambda_{Q\bullet} \sim 8.92$. For the Hénon map it is $\sigma_u \approx 0.42$ and we would expect $\lambda_{Q\bullet} \sim 17.27$ from (6.4) and $\lambda_{Q\bullet} \sim 14.72$ from (6.5).

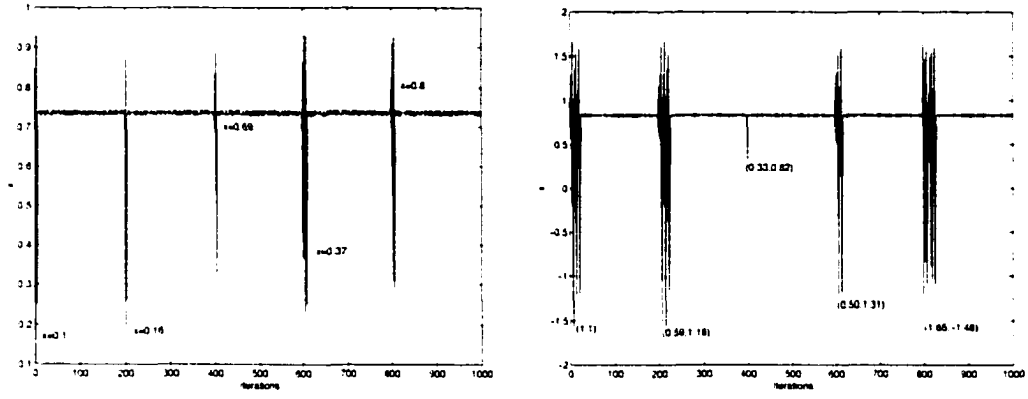


a) b)

Figure 6.5: a) The result of performing ϵ-greedy off-line Q- and Sarsa-learning for the control of a) the logistic map, and b) the Hénon map. The solid line shows the learning curve for the Q-learning algorithm, the dashed line for the Sarsa algorithm. Shown are the total number of iterations from all presented episodes as a function of the presented episodes. ϵ was decreased to zero from initially one and set to zero over the last 1,000 episodes. The number close to the graph is the limiting slope of the corresponding curve and is a measure for the estimate $\lambda_{Q\bullet}$ of the transient of the controlled dynamics.

Figure 6.6a shows controlled logistic dynamics and Figure 6.6b the controlled Hénon dynamics using the policy Q^* approximated through Q-learning. Starting from a random initial condition, the system was randomly perturbed into a new

state every 200 iterations. We see that control is reestablished quickly after each system perturbation.



a) b)

Figure 6.6: The controlled dynamics of a) the logistic map and b) the Hénon map using the policy Q^* obtained through Q-learning. Starting from a random initial state, the system is randomly kicked every 200 iterations into a new state as shown. Control of the nontrivial fixed point is reestablished quickly after each perturbation.

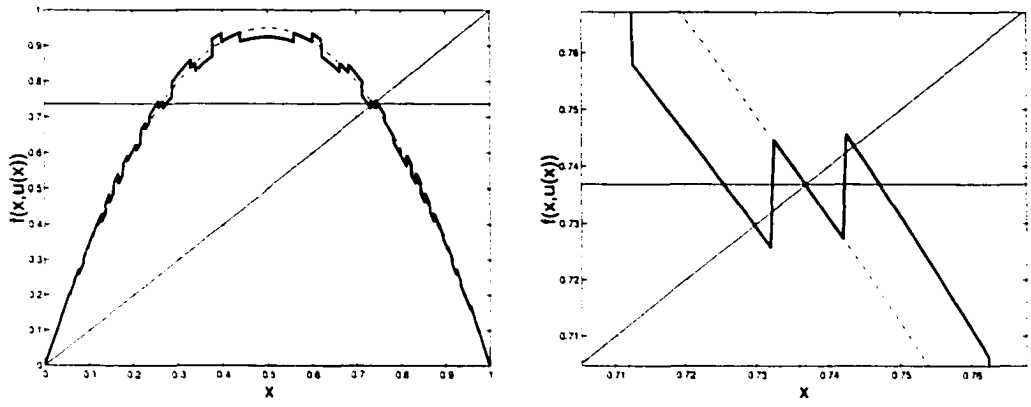
Analysis of the Approximated Optimal Policy

To get an idea of how the approximated control policy Q^* realizes control of a fixed point, we visualize the policy for the logistic map in Figure 6.7. Figure 6.7a shows the effective map $(p_0 + u_n(x_n))x_n(1 - x_n)$ for control of the logistic map. In Figure 6.7b the region close to the fixed point is shown in more detail.

The global action of the control policy can be better understood from Figure 6.8, which shows for each reference vector the associated control with maximal Q-value. Iterating the logistic map backwards from the fixed point x_f:

$$x_n = (1/2)(1 \pm \sqrt{1 - 4x_{n+1}/p_0}). \tag{6.9}$$

locations are obtained which are visited before iterations to the fixed point. The vertical lines in Figure 6.8 correspond to the locations of backward iterated points.



a) b)

Figure 6.7: a) Effective map $(p_0 + u_n(x_n))x_n(1-x_n)$ for control of the logistic map. b) A closer look at the region around the fixed point.

A vertical line labeled with integer i denotes a location from which the fixed point x_f can be reached in i iterations of the uncontrolled logistic map. Comparing the location of these lines and the controls u it becomes apparent that the controller learns to bring the dynamics globally close to these backward iterated locations, thus approximating a globally optimal policy. Close to x_f the controller performs discrete controls which trap the dynamics close to x_f, while globally it brings the dynamics closer to points from which x_f can be reached in a minimum number of iterations. Thus the algorithm targets the UPO along a quasi-optimal path [DH94].

6.3.2 Noise Dependence

In this subsection we investigate the influence of noise on the control performance. Two questions are interesting in this context. Having established an optimal control policy in a noise free environment, how well does the policy do if noise is added to the system? The second, probably more important question is, how well a policy will perform if we approximate it in a noisy environment.

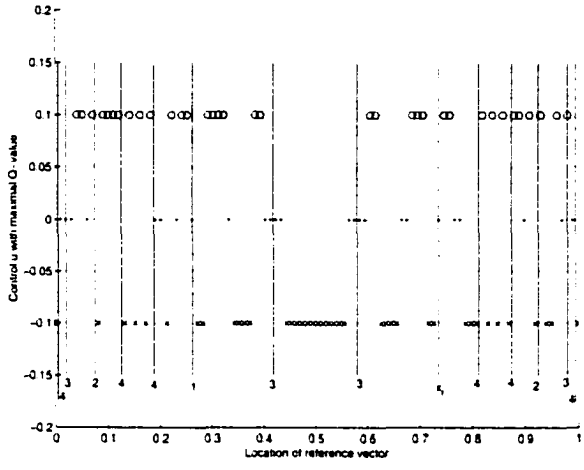
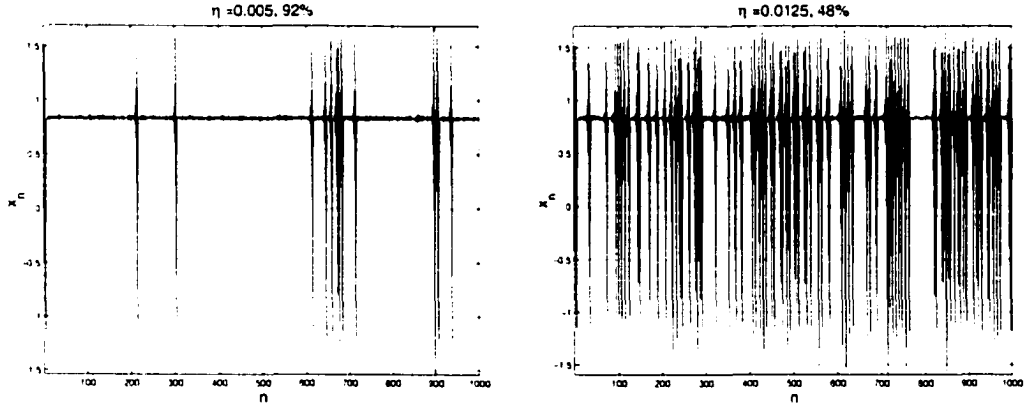


Figure 6.8: The approximated policy Q^* (o for $u = 0.1$, x for $u = -0.1$, . for $u = 0$) as a function of the reference vectors. The vertical lines correspond to locations of points iterated backward from the fixed point x_f of the logistic map.

Using an Optimal Policy under Noisy Conditions

For local control methods control is typically successful under the influence of noise which is small enough to not kick trajectories out of the controllable region. In [OGY90] Ott *et al.* demonstrated that the OGY method was able to stabilize the Hénon system if 20% normal distributed noise (the size of the maximal control perturbation is 5 times η, the size of the noise) where added. To see the performance of our algorithm for the control of the Hénon map under the influence of normal distributed noise, we added a noise term $\eta\,\delta$ to the right hand side of (6.7), where $\eta > 0$ regulates the size of the noise term, and δ is a Gaussian distributed random variable with zero mean and variance one. Using the optimal policy Q^* obtained as described above, we would expect our algorithm to establish control if $\eta \approx 0.2|u|_{max} = 0.005$ for the chosen control set $\mathcal{U}_{Hen}$. The controlled dynamics for $\eta = 0.005$ is shown in Figure 6.9a. We see that control is lost only a few times due to tails in the Gaussian perturbations. The trajectory stays close to the goal for 92% of the control time. For higher noise levels, trajectories will be kicked out of the controllable region with higher probability. As a result, local

control methods will have to wait to reestablish control until the system naturally reenters the controllable region. For global methods this is not the case. Since the controls attempt to bring trajectories into the controllable region from anywhere on the attractor we would expect global methods to be more effective against noise. The controlled dynamics for $\eta = 0.0125$ (50% normal noise) are shown in Figure 6.9b. Although control is frequently lost, trajectories are close to the goal 48% of the control time which shows that trajectories are brought back to the goal quickly. For 100% normal noise we found that trajectories are close to the goal for 20% of the control time.



a) b)

Figure 6.9: Controlled dynamics of the Hénon map using Q^* obtained as discussed above, for a) $\eta = 0.005$, and b) $\eta = 0.0125$.

Approximating a Policy under Noisy Conditions

To demonstrate resistance of the algorithm against noise we added a noise term η_n to the dynamics at each iteration, i.e. for the logistic map:

$$x_{n+1} = p_0 x_n(1 - x_n) + \eta_n,$$

where η_n is now chosen uniformly from the set $(-\alpha/2, \alpha/2)$.[3] We compared λ_Q to λ_u by averaging over 2,000 episodes, where an episode was terminated when the system comes close to the fixed point. For the logistic map we used, as before, $|x_n - x_f| < 0.005$ and for the Hénon map $||\mathbf{x} - \mathbf{x}_f|| < 0.02$. After completion of one episode the system is set into a new random initial state. Figure 6.10 shows λ_Q/λ_u as a function of α/u_{max}. For the logistic map the maximal control u_{max} was 0.1, for the Hénon map 0.025. The policies were learned in a noisy environment with the given noise level. All policies were obtained off-line through ϵ-greedy Q-learning. We see that the controller is able to reduce λ_Q below λ_u for considerable noise levels, even when the added noise is much higher than the applied control. It must be noted however that control of the fixed point is lost when $\alpha/u_{max} \approx 1$ for the logistic map and $\alpha/u_{max} \approx 2$ for the Hénon map. Nonetheless, for targeting applications, where we are only interested in reducing λ_Q as far as possible, the algorithm is resistant to high noise levels. We see that a good control policy can be established even under the influence of considerable noise. In most investigations such as [OGY90] it was not discussed how the influence of noise will influence the computations of the necessary quantities. In some real experiments where noise is always present, the OGY was successfully used. In [Kap96] it was mentioned however that the control of an electronic system was not possible with the OGY method due to the intrinsic noise of the system. A detailed analysis and comparison of the performance of chaos control methods under the influence of noise has not been done yet and would certainly be an interesting topic for future investigations. In general we would expect that global methods or combinations of local methods with targeting are more robust to noise than local methods.

---

[3]For Gaussian noise the dynamics often become unstable, i.e. go to infinity.

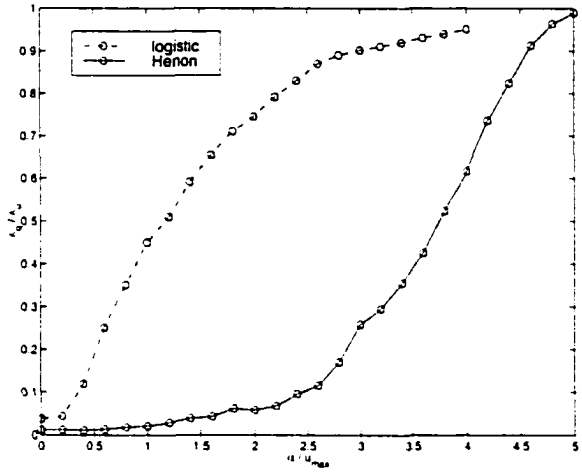


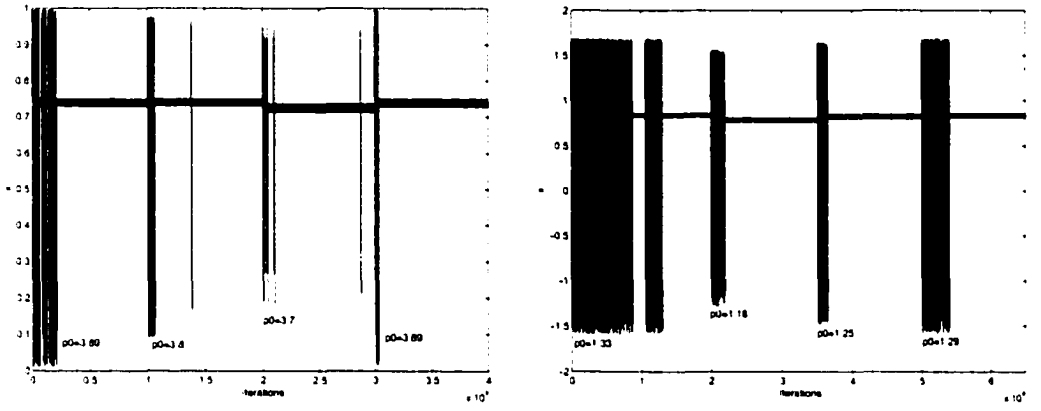
Figure 6.10: Noise dependence of the algorithm. The dashed line gives λ_Q/λ_u as a function of α/u_{max} for the logistic map, the solid line for the Hénon map.

6.3.3 On-line Control

Control in Non-stationary Environments

An advantage of the presented control algorithm is that it can be used in on-line control applications and in non-stationary environments. To demonstrate this, Figure 6.11 shows the result of a control performed on-line. The system was set in a random initial state before the controller was turned on and the algorithm started to update the Q-values with the goal to stabilize the system at a fixed point. Learning was performed deterministically using Q-learning. Figure 6.11a shows on-line control of the logistic map. The parameter p_0 was initially set to 3.89. After approximately 2,500 iterations the controller establishes control of the fixed point x_f. After 10,000 iterations p_0 was set to 3.8. Control is lost for a short time, until the controller is able to reestablish a control strategy and to stabilize the system at the changed fixed point location. After 20,000 iterations p_0 was set to 3.7 and finally after 30,000 iterations back to 3.89. The on-line control of a non-stationary Hénon map is shown in Figure 6.11b. Here p_0 was initially set to 1.33 and then changed after 20,000 iterations to $p_0 = 1.18$, after 35,000 iterations

to 1.25 and then finally after 50,000 iterations to 1.29. We see that the controller is able to control the non-stationary systems and adapt quickly to changes in the system parameter p_0. Note that for the whole control process the set of reference vectors shown in Figure 6.4 was used. It was not necessary to introduce new centers or change center locations.



a) b)

Figure 6.11: On-line control of a non-stationary a) logistic map, and b) Hénon map. In a) p_0 was changed every 10.000 iterations from initially 3.89 to 3.8, 3.7 and back to 3.89, and in b) p_0 was changed from initially 1.33 to 1.18, 1.25 and 1.29.

A different tracking application is shown in Figure 6.12 where the on-line control of the fixed point of the logistic map is shown with p_0 slowly changing from $p_0 = 3.6$ to $p_0 = 3.9$. The results of this subsection demonstrate that the proposed control algorithm is suited for *on the fly* control and the control of non-stationary systems. On-line performance is fast compared to off-line control since the controller only has to establish control starting from the particular initial value. We also see that ϵ-greedy learning is not necessary to establish control in such applications.

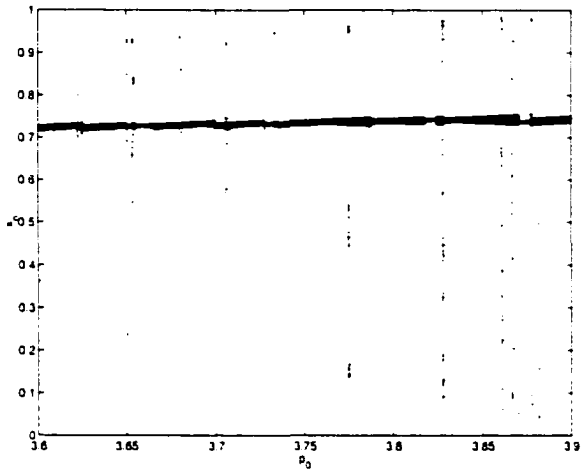
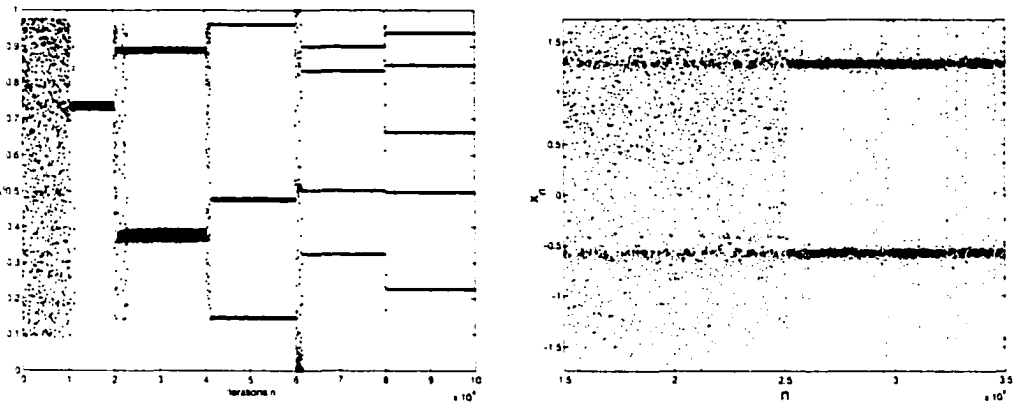


Figure 6.12: Tracking of the period-one UPO of the logistic map through on-line control as p_0 is slowly changed over 40.000 iterations from $p_0 = 3.6$ to $p_0 = 3.9$. An initial phase of 2.000 iterations is not shown in which on-line control of the UPO for $p_0 = 3.6$ was established.

Control of Fixed Points of Higher Period

To demonstrate that the control algorithm is able to detect and stabilize higher order fixed points, we attempted to stabilize fixed points of the logistic map up to period five through a deterministic on-line control with the previously mentioned control set $\mathcal{U}_{log}$. The result is shown in Figure 6.13a. Initially the goal was to stabilize a fixed point of period one. This was achieved after approximately 10.000 iterations of the logistic map. After 20.000 iterations, the goal was switched to control a period two fixed point, after 40.000 iterations to a period three and so on. In each case, locations of the fixed points were quickly identified and control established. In Figure 6.13b on-line control of the period-two fixed point of the Hénon map is shown. For this control the control set was $\mathcal{U}_{Hen} = \{0, 0.06, -0.06\}$. The numerical control of higher-period orbits of the Hénon map is complicated by the fact that the dynamics become unstable if perturbations with $|u| > 0.06$ are applied globally.



a) b)

Figure 6.13: a) Control of various fixed points of the logistic map through deterministic on-line control. The control goal was changed every 20.000 iterations. b) On-line control of period-two fixed point of Hénon map.

Targeting

To demonstrate that the algorithm can be applied to the more general problem of targeting general states we arbitrarily choose three different target locations for both the logistic map and the Hénon map. For the logistic map we choose $x_{g1} = 0.1$, $x_{g2} = 0.4$ and $x_{g3} = 0.8$, for the Hénon map $\mathbf{x}_{g1} = (-0.47, 1.46)$, $\mathbf{x}_{g2} = (-1.38, -0.11)$ and $\mathbf{x}_{g3} = (1.46, -0.93)$ (see Figure 6.4). We then learned optimal control policies using ϵ-greedy off-line Q-learning presenting 10.000 episodes. An episode was terminated if the reference vector closest to the goal was reached. Using these learned control policies, we compared the average number of iterations needed to come close to the goal ($|x_n - x_{gi}| < 0.005$ for the logistic map and $||\mathbf{x}_n - \mathbf{x}_{gi}|| < 0.02$ for the Hénon map) with applied control perturbations, λ_Q, and the average number of iterations until the uncontrolled system comes close to the target location, λ_u. The applied controls and used sets of reference vectors were as described in the preceding sections. The results are shown in Table 6.1. We see that λ is considerably reduced in all cases.

	λ_Q	λ_u	$\ln \lambda_u$
log. map, goal 1	7.9	∞	∞
goal 2	12.6	178.6	5.19
goal 3	8.8	76.2	4.33
Hén. map, goal 1	30.0	498.9	6.21
goal 2	13.9	240.8	5.48
goal 3	195.6	1,142.9	7.04

Table 6.1: Targeting of three different states for the logistic and the Hénon map.

6.4 Impulsive Control

In this section we demonstrate several applications of the proposed chaos control algorithm to impulsive control problems (see Subsection 3.4.4). Here a system variable is directly perturbed at discrete time intervals. These type of perturbations are especially relevant in medical applications and we will discuss the impulsive control of a relaxation oscillator. Furthermore we will demonstrate the control of a hyperchaotic nine-dimensional Lorenz system.

6.4.1 Rössler Attractor

To show that the method can be applied in situations where impulsive control is desired, we control the chaotic Rössler system:

$$x' = -y - z,\ y' = x + 0.2y,\ z' = 0.2 + z(x - 5.7).$$

Letting $\mathbf{x} = [x\ y\ z]^T$, to control this system we apply periodic proportional pulses (see [MG94, Cha97, Cha98] and Subsection 3.4.4)

$$\mathbf{x}(t_n) \rightarrow u \cdot \mathbf{x}(t_n),$$

applied at regular time intervals, whenever the systems visits a surface-of-section. In particular we apply the same principle as proposed in [Cha98], where a formula for u was derived requiring a functional representation of the Poincaré map and

its derivative which is based on the governing equations. Since we do not want to make use of any analytical description of system dynamics, we allow three a priori fixed values for u and use reinforcement learning to find an optimal control strategy.

The control strategy we use is the following. As mentioned in [Cha98], the Rössler system has a well-defined Poincaré section defined by the points where $x(t)$ reaches a local maximum. First we record and store a certain amount of uncontrolled data $\mathbf{x}_{store}$ corresponding to points in the Poincaré section, which gives a set $\mathcal{X}_{store}$. To control the system, we kick the x-coordinate whenever it reaches the Poincaré section: $x \rightarrow u \cdot x$, where u can be any of the three numbers $\{1, 0.95, 1.05\}$. Using this new x value and the unchanged y and z values, we find the closest point from the set $\mathcal{X}_{store}$ and reset the system state to this state.

To formulate a goal and to compute an optimal state-action value function $Q(w(x), u)$, we distributed a set of 150 reference vectors uniformly over the interval $[0, 15]$ which represents the possible values of x at the intersection with the Poincaré section. Based on the intersections of x with the section we obtain a sequence of representation vectors w_n which allows to detect fixed points of desired period. Based on the sequence w_n we will give reinforcements to the state-action pair $(w(x), u)$ and update the Q-values using Q-learning as discussed above.

Figure 6.14 shows the result of an on-line control of the 2-cycle (Figure 6.14a,b), the 3-cycle (Figure 6.14c,d) and the 4-cycle (Figure 6.14e,f). To establish control, only a few hundred visits of the Poincaré section where needed. In principle this control strategy is similar to the impulsive control methods discussed in Subsection 3.4.4. The advantage over these methods is that no trial-and-error is necessary and that a global policy is established. The algorithm systematically finds the best policy for the application of the kicks. Moreover no analytical knowledge is required. With a simple reformulation of the control goal the algorithm can target more general states as discussed in the next paragraph.

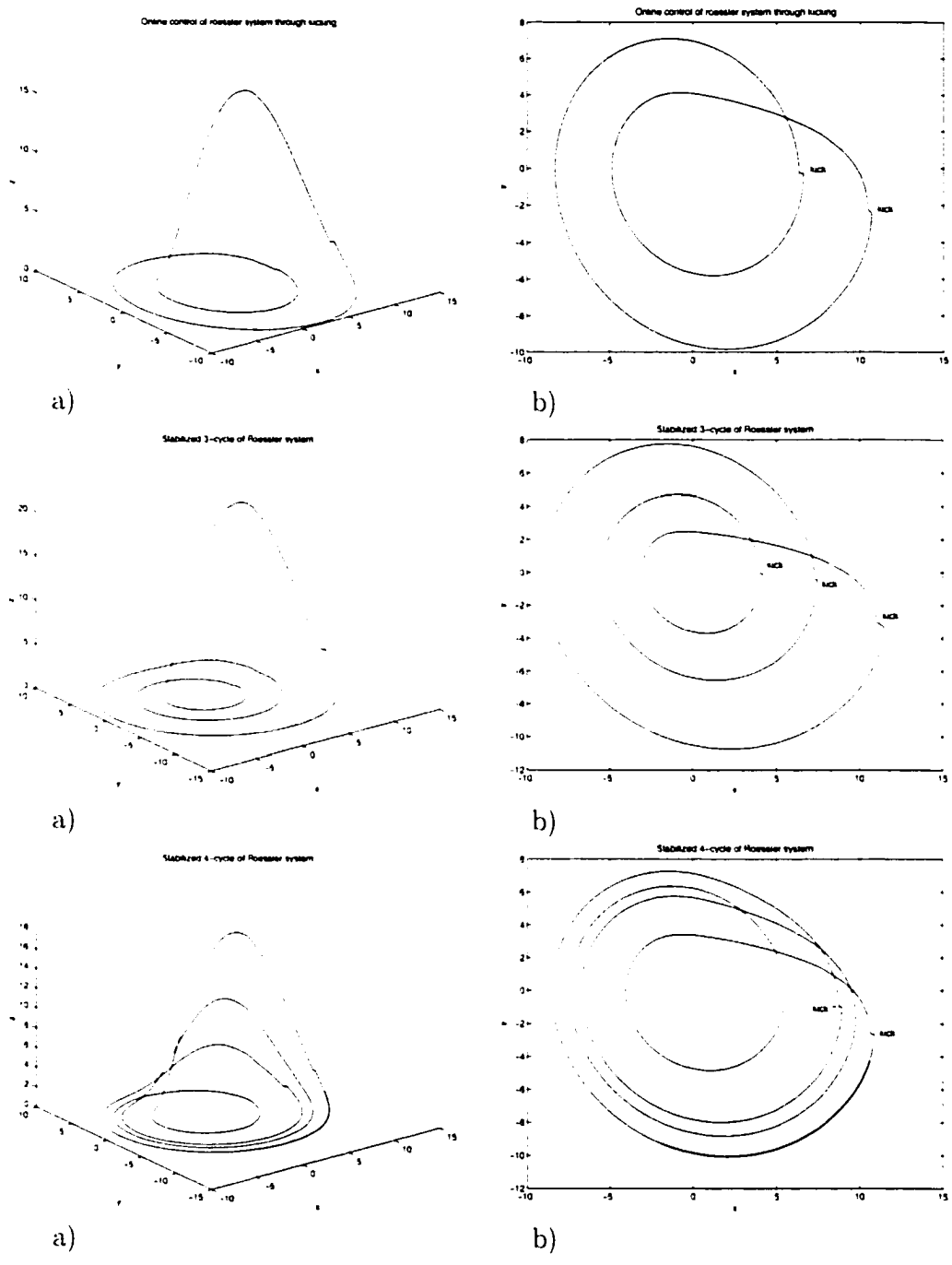


Figure 6.14: Control of different periodic orbits of the Rössler system through periodic proportional pulses. a). b) period 2 orbit. c). d) period 3 orbit. e). f) period 4 orbit.

A Targeting Application

In this paragraph, the task of the controller is to direct the system from an arbitrary initial condition to a previously specified target location through small proportional pulses. Following the same control strategy as described above, kicking the system whenever it reaches the surface-of-section, we give the controller the goal to intersect the Poincaré section at $x = 6.9$ (we chose this value arbitrarily). Thus the goal of the control algorithm is achieved if $w(x_n) = w(6.9)$. An optimal control policy was approximated off-line over 5.000 episodes. An example of targeting with the approximated policy is shown in Figure 6.15. Starting from the initial condition $\mathbf{x}_0 = [-8.5407,\ 1.3441,\ 0.0314]^T$, the system will naturally reach the vicinity of the goal state ($w(x) \approx w(6.9)$) only after 83 visits of the Poincaré section (see Figure 6.15a). Kicking the system when it reaches the section, according to the approximated policy, the system will visit the desired state after only one visit of the section. Thus only one small system perturbation is needed to dramatically reduce the time from initial to goal state. On average, from any possible initial conditions, the results will not be as good as for the shown example. Picking 1.000 random initial conditions on the Rössler attractor and measuring the average number of visits in the section until the desired neighborhood is reached, resulted in $\lambda_u \approx 67$ for the uncontrolled system and $\lambda_{Q^*} \approx 11$ for the controlled system.

The results here offer interesting perspectives for the targeting problem of directing spacecraft to the moon as discussed in [BM95, SO97] and it would be interesting to see how an optimal policy computed through our algorithm compares to the result obtained in [BM95, SO97].

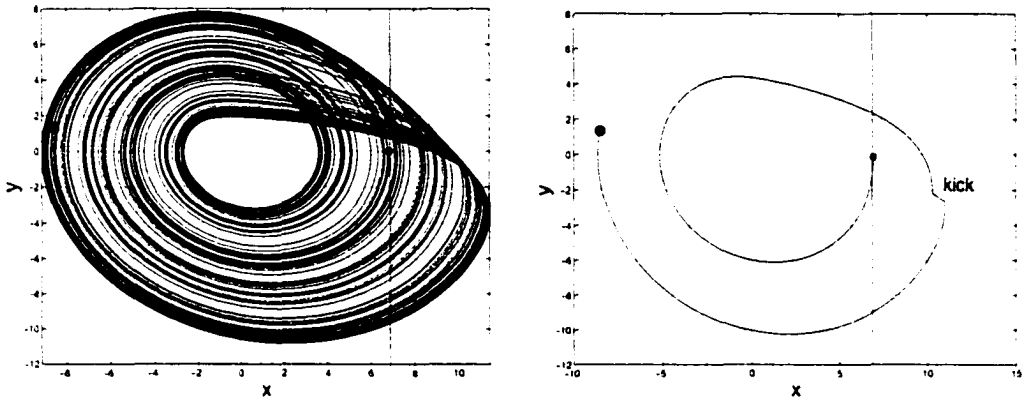


Figure 6.15: Targeting of the Rössler attractor. The goal is to intersect the Poincaré section at $x = 6.9$. This goal is represented by the vertical line at the graph, where the local maximum of the trajectory should occur. a) The uncontrolled trajectory intersects a small neighborhood ($w(x_n) = w(6.9)$) only after 83 visits of the section. b) With control the goal is visited after only one intersection, or one perturbation of the trajectory.

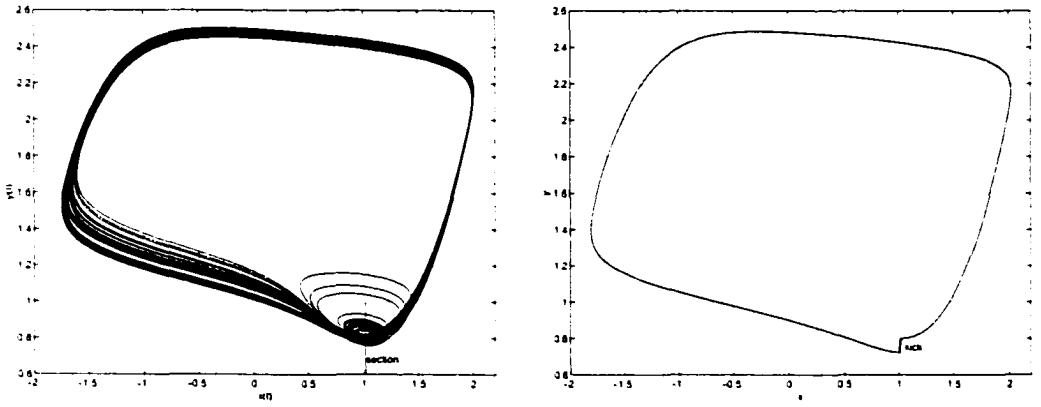
6.4.2 Relaxation Oscillator and Heartbeat

As a further example for the impulsive control of continuous systems from a time series we chose a system originally proposed by Rössler et al. [RRL78], who observed a variety of sub-harmonic responses for the periodically forced system

$$\begin{aligned} x' &= (y - 1.5) + (x - x^3/3) - A\sin(0.2t) \\ y' &= -(x - 0.467 + 0.8(y - 1.5))/9. \end{aligned} \tag{6.10}$$

The sub-harmonic responses are similar to arrhythmia observed in malfunctioning hearts [TS86]. For $A = 0.045$ the system shows chaotic behaviour. The trajectory for $A = 0.045$ in the (x, y)-plane is shown in Figure 6.16a.

To clear the system from the sub-harmonic responses we attempted to control the dynamics on a period one unstable periodic orbit through proportional pulses applied to the y-coordinate. We follow a strategy similar to the one taken for control of the Rössler system but we do not reset kicked states into recorded states in the section. As a criterion for successful control we chose that subsequent minima



a) b)

Figure 6.16: a) Trajectory of system (6.10) in the (x, y)-plane. The experimental section used for the control is also shown. b) Phase portrait of the controlled dynamics. The controller is able to clear the system from the sub-harmonic responses through kicking the y-coordinate whenever y reaches a local minimum.

of the y-coordinate have to occur at close values. Thus the only information needed to control the system is contained in the time series of the y-coordinate. To control the system we applied proportional pulses (or kicks) to the y-coordinate whenever y reaches a local minimum.

$$y \rightarrow u \cdot y \quad \text{if } y \text{ is at a local minima.} \tag{6.11}$$

Control u was taken here from the set $\{0.9, 0.95, 1, 1.05, 1.1\}$. As reference vectors we choose the set of 41 points $\mathcal{W} = \{0.6, 0.61, \cdots, 1\}$.

We applied on-line deterministic Q-learning to this control problem. After approximately 140 visits of the section the controller was able to transform the arrhythmic signal into a periodic one. Figure 6.17 shows two parts of a plot of the signal $y(t)$ subject to the control perturbations. Figure 6.17a shows the signal for the first 50 visits of the section. In certain regions the signal becomes strongly perturbed while the controller evaluates possible actions. Figure 6.17b shows the signal in the interval where the controller finds a successful strategy (after 140 visits of the section). The trajectory of the controlled system is shown in Figure

6.16b. When a local minimum in the y-coordinate is detected, y is perturbed from $y \approx 0.8$ to $y \approx 0.72$.

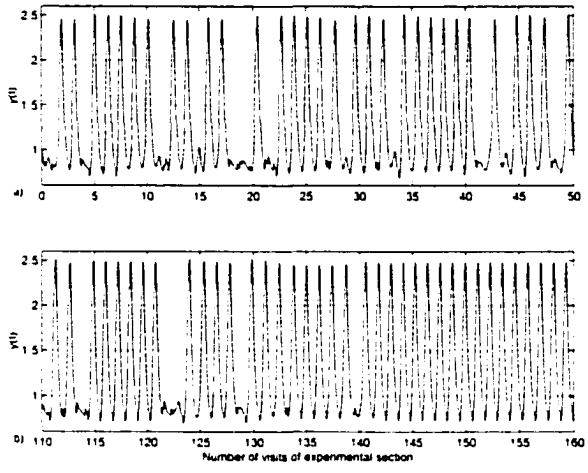


Figure 6.17: The signal $y(t)$ subject to control perturbations while the Q-values are updated. part a) shows the first part, part b) the last part, just before successful control was established.

6.4.3 Controlling Hyperchaos

To investigate if the proposed control method can be applied to the control of high-dimensional chaos, in particular systems with two positive Lyapunov exponents, we attempted to control a nine-dimensional Lorenz system proposed by Reiterer *et al.* [RLS+98]. They proposed the following system for the study of high-dimensional chaos:

$$\dot{C}_1 = -\sigma b_1 C_1 - C_2 C_4 + b_4 C_4^2 + b_3 C_3 C_5 - \sigma b_2 C_7$$

$$\dot{C}_2 = -\sigma C_2 + C_1 C_4 - C_2 C_5 + C_4 C_5 - \sigma C_9/2$$

$$\dot{C}_3 = -\sigma b_1 C_3 + C_2 C_4 - b_4 C_2^2 - b_3 C_1 C_5 + \sigma b_2 C_8$$

$$\dot{C}_4 = -\sigma C_4 - C_2 C_3 - C_2 C_5 + C_4 C_5 + \sigma C_9/2$$

$$\dot{C}_5 = -\sigma b_5 C_5 + C_2^2/2 - C_4^2/2$$

$$\dot{C}_6 = -b_6 C_6 + C_2 C_9 - C_4 C_9$$

$$\dot{C}_7 = -b_1 C_7 - rC_1 + 2C_5 C_8 - C_4 C_9$$

$$\dot{C}_8 = -b_1 C_8 + rC_3 - 2C_5 C_7 + C_2 C_9$$

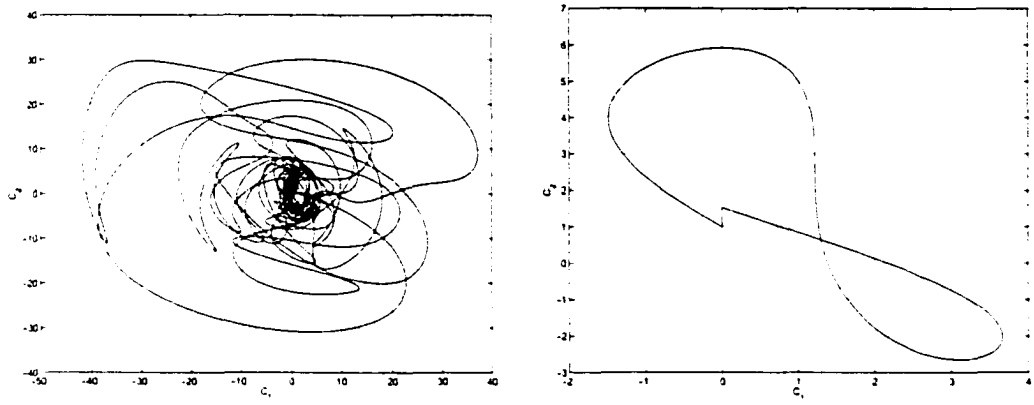
$$\dot{C}_9 = -C_9 - rC_2 + rC_4 - 2C_2 C_6 + 2C_4 C_6 + C_4 C_7 - C_2 C_8.$$

The b_i are defined in terms of a single parameter a as

$$b_1 := 4\frac{1+a^2}{1+2a^2}, \quad b_2 := \frac{1+2a^2}{2(1+a^2)}, \quad b_3 := 2\frac{1-a^2}{1+a^2}$$
$$b_4 := \frac{a^2}{1+a^2}, \quad b_5 := \frac{8a^2}{1+2a^2}, \quad b_6 := \frac{4}{1+2a^2}.$$

Reiterer *et al.* set $a = \sigma = 1/2$ and showed that the system exhibits a variety of different behaviours depending on the value of r. For $r > 43.3$ the system possesses two positive Lyapunov exponents which leads to hyperchaotic behaviour. A projection of the dynamics generated by the system for the values $\sigma = a = 0.5, r = 44.0$ on the $C_1 - C_2$ plane is shown in Figure 6.18a.

By using our method we were able to control the system in the hyperchaotic regime, for $\sigma = a = 0.5, r = 44.0$, onto a periodic orbit of period one. Here perturbations have been applied to the complete set of state variables whenever the solution crosses the section defined by $C_1 = 0$, $\dot{C}_1 > 0$. A set $\mathcal{W}$ of 200 reference vectors $\mathbf{w}$ was constructed through a Neural-Gas vector quantization of 1000 recorded crossings of the unperturbed system. Control was applied by using the method of periodic proportional pulses [Cha98] in the section in the following way. If the orbit crosses the section in a state $\mathbf{C}$, C_2 is perturbed into the new state uC_2 with u chosen from $u \in \mathcal{U} = \{1, 1.05, 1.1, -1.05, -1.1\}$, which brings the system into a state $\hat{\mathbf{C}}$. If u is different from one the complete state is set into the new state $\mathbf{C} = \mathbf{w}_n$, where $\mathbf{w}_n$ is the reference vector closest to $\hat{\mathbf{C}}$. We applied on-line deterministic Q-learning to this control problem. After approximately 100 visits of the section the controller was able to stabilize a period one dynamics. The stabilized orbit is shown in Figure 6.18b.



a) b)

Figure 6.18: a) Projection on the $C_1 - C_2$ plane of the attractor generated by the 9D Lorenz system for $\sigma = a = 0.5$ and $r = 44.0$. The asymptotic behaviour is characterized by two positive Lyapunov exponents. b) The stabilized period one orbit.

6.5 Conclusions

In this chapter we applied reinforcement learning to the problem of chaos control. The control strategy is based on learning state-action value functions for specific tasks. We considered in particular the task of stabilization of unstable fixed points or periodic orbits embedded in a chaotic attractor using small system perturbations. The method does not require any analytical description of the underlying dynamical system nor do we need to know the location of the fixed points in advance. Roughly speaking, the algorithm first explores the reduced state space to determine an orbit that visits a neighborhood of a fixed point of desired period and once this neighborhood has been reached, a control policy is learned which traps the dynamics in the neighborhood of the desired state while globally directing the dynamics as fast as possible into the identified neighborhood. Besides the stabilization of UPOs we also demonstrated that the same algorithm, with suitably formulated goals, is capable of solving the targeting problem. It can be used for parametric or impulsive control, and since it also is suitable under noisy

as well as non-stationary conditions it offers promising perspectives for a general purpose black-box controller.

To satisfy the requirement of a finite Markov decision process, we first pursued a vector quantization to decompose the state space into a finite space, represented by a large number of Voronoi cells. The corresponding reference vectors have been determined by means of the Neural-Gas learning algorithm. The vector quantization reduces the complexity of the reinforcement learning problem. Although the Neural-Gas vector quantization yields an optimal approximation of the invariant measure of the chaotic attractor, the resulting quantization of the state space may not be optimal for the specific control task under consideration. In the last chapter of this dissertation we will present a modification of Fritzke's Growing Neural-Gas algorithm and combine the task of vector quantization and learning of the optimal control strategy.

Our results for the heartbeat model suggest that the control method could possibly be applied to atrial defibrillation. Atrial fibrillation is the most common sustained cardiac arrhythmia [KASM82]. Recently first results with an implantable defibrillator restoring periodic rhythm through low-energy shocks were reported [WLL+98]. In addition the OGY method has recently been applied to the control of atrial fibrillation in human patients [LBM+99, WMV+00]. Since our algorithm is applicable in non-stationary environments, it could be suited for designing "intelligent" implantable atrial defibrillators.

The results for the nine dimensional Lorenz system in its hyperchaotic regime demonstrate the relevance of the presented control method to the control of more complex systems. In the next chapter we will apply the proposed control method to the control of spatially extended coupled logistic map lattices.

Chapter 7

CONTROL OF SPATIALLY EXTENDED SYSTEMS

> Embedding microscopic sensors, computers and actuators into materials allows physical systems to actively monitor and respond to their environments in precisely controlled ways.
>
> (O. Guenther *et al.*, 1997)

In this chapter we apply the proposed chaos control algorithm to the stabilization of a logistic coupled map lattices serving as a model for a system exhibiting spatio-temporal chaotic behaviour.

7.1 Introduction

If the physical variables describing a system depend on more than one independent variable, then a system must typically be described in terms of *partial differential equations* (PDEs) instead of ordinary differential equations. This is the case for many spatially extended or *spatio-temporal systems* such as fluids where system variables are changing over time and space. An example PDE is the *Navier-Stokes equation*

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{u} + \frac{1}{\rho} \mathbf{F},$$

which represents a nonlinear PDE and is the basic dynamical equation expressing Newton's second law of motion for a fluid of constant density [Tri88]. ν is the kinematic viscosity of the fluid and ρ the fluid density which is assumed to be

constant (incompressible fluid). $\mathbf{u} = \mathbf{u}(\mathbf{x}, t)$ represents the velocity of the fluid at a particular point in space and time. $\mathbf{F}$ represents the contribution of external forces such as gravity on the fluid and p is pressure.

Spatio-temporal systems show interesting behaviour if the system is operated outside of its equilibrium[1]. The system can be brought away from equilibrium through the change of a control parameter. At some critical value of this parameter the equilibrium will become unstable and the system undergoes a *bifurcation* leading to new types of behaviour: *non-equilibrium spatial patterns* [CH93]. One distinguishes three types of regular, i.e. periodic patterns:

I_s Patterns periodic in space and stationary in time.

III_0 Patterns uniform in space and periodic in time.

I_0 Patterns periodic in space and time.

Far from equilibrium systems often undergo transitions from spatial patterns to chaotic or turbulent behaviour.

Different mechanisms exist which create the instabilities. One mechanism arises from the existence of constraints and conservation laws [CH93]. *Rayleigh-Bénard convection* is a typical example for this phenomenon [LM82]. A fluid is placed between flat infinite horizontal plates which are perfect heat conductors. The fluid is driven by maintaining the lower plate at a higher temperature. The temperature gradient introduces a density gradient which leads to the desire of the fluid near the lower plate to rise. Due to conservation of mass and the constraints resulting from the plates the fluid can not rise as a whole. At a certain critical value of the temperature gradient the forcing overcomes the dissipative effects of

[1] Equilibrium in the sense of energy, i.e. a fluid is at thermal equilibrium if it has a constant equal temperature anywhere.

thermal conduction and viscosity and the transition to a spatial roll pattern occurs. Increasing the temperature gradient further leads to doubling bifurcations and eventually to turbulence or chaos [AFH94]. A second type of mechanism is provided by competing interactions between elementary units [CH93]. An example are neural networks where the principle of local excitation combined with lateral inhibition is a fundamental ingredient in neural information processing and associative memory tasks [Des94]. The existence of competition between different forces is also the mechanism which leads to pattern formation in chemical reaction-diffusion systems such as the Belousov-Zhabotinskii reaction [HM81].

Since governing equations of spatio-temporal systems in the form of PDEs are generally difficult to analyze either theoretically or numerically [CH93], one often studies *model equations* or *model systems* whose pattern formation properties are designed to be as close as possible to those of the original system, but which are easier to analyze or simulate. Two classes of models can be distinguished: reduced PDEs and discrete model equations.

1. **Reduction to simpler PDEs.**

   Immediately above the threshold, a weakly nonlinear perturbation analysis typically leads to simpler equations in terms of *amplitude equations*. An example is the *Ginzburg-Landau* equation

   $$\frac{\partial A(\mathbf{x}, t)}{\partial t} = (a + ib)\nabla^2 A(\mathbf{x}, t) + f\left(|A|^2\right) A(\mathbf{x}, t).$$

   where A represents a complex amplitude and f an arbitrary complex function. This equation is often studied as a model equation for spatio-temporal behaviour. Further away from equilibrium a perturbation expansion of a fully nonlinear ideal periodic solution sometimes leads to simpler *phase equations* (see [CH93, DK98] and references therein for details). In other cases, PDEs

can be simplified by assuming idealizing system or boundary conditions. An example is the *Boussinesq-Oberbeck approximation* which reduces the Navier-Stokes equations to a simpler set of PDEs [AFH94].

2. **Discrete models.**

   - Discretize space: Coupled ODEs.

     In certain cases PDEs can be viewed as ODEs with an infinite dimensional phase space or be reduced to a finite (low-dimensional) system of coupled ODEs. The reduced system of ODEs can be obtained from the PDE through different finite dimensional approximation techniques such as the *Galërkin approximation* [Chi99]. A famous example is Lorenz's reduction of flow equations to the three dimensional system of coupled ODEs (2.2) [Lor63].

   - Discretize space and time: Coupled maps.

     Discretizing space and time allows to derive discrete coupled maps which are easier to simulate and can reproduce many of the spatial patterns found in the original system. Discretizing is a necessary step in order to numerically integrate a PDE. To this end all partial derivatives are replaced by *finite-difference* quotients. The resulting algebraic equation is called a *difference equation.* The finite-difference approach requires a discretization of space through an appropriate grid or mesh (see [And95] for an elementary introduction). The resulting difference equation on the grid can be interpreted as a coupled map lattice. Coupled map lattices have been intensively studied as models of spatio-temporal systems (see [Kan92] and other articles in the same journal). We will discuss coupled maps further in the next section.

- Discretize space, time and the field: Cellular automata.

  Cellular automata can be derived from coupled maps by discretizing the state variable. These models are computationally and mathematically the simplest and often allow a complete description of their dynamical behaviour. They are studied in particular in the context of biological pattern formation and the behaviour of populations but are also useful as models of neural networks and spin systems (see [Wol86]).

The study of coupled ODEs and coupled maps is important not only because of their role as approximate models for systems governed by PDEs but has also gained much interest due to their speculated role in neural-information processing [SGK90, Erm94, HI97]. Furthermore, many nonlinear systems such as series arrays of Josephson junctions [AGK91, ODG96, Wie99], electric circuits [HXCP98], lasers [ATJR98, TJD+99], power systems and ecology [DL99b], networks of neurons [HI97], reaction-diffusion systems [Mur93] and cardiac pacemaker cells [Win90] can be modeled in terms of coupled ODEs or maps. General hierarchical complex systems can be interpreted in the context of coupled cell systems [BP99]. Recently the possibility of *smart matter* is being investigated in this context [GHH97, HH98]. In many of these applications the question of feasible control mechanisms is important. In this chapter we will discuss the control of coupled map lattices through reinforcement learning and propose a general scheme for the control of coupled systems.

7.2 Complex Systems

When studying and modeling *complex systems*[2] composed of interacting parts one usually decomposes the system into interacting subsystems. The modeling

[2]We refer to a system as a complex system if it can show nontrivial dynamics and is composed of interacting parts.

process consists of finding suitable equations governing the subsystems and their interactions and often relies on a combination of empirical evidence with physical principles and symmetry properties. Coupled systems can exhibit a variety of complex dynamical behaviours such as multistability or spatio-temporal chaos, even if the interrelated subsystems behave simple in isolation. It is essential to view and study the system as a whole in order to understand how the interactions of possibly simple subsystems can lead to complex behaviour.

A class of coupled systems that has attracted much interest in mathematics, physics and biology is provided by *networks of coupled cells*, where each cell is described by a *dynamical unit* showing regular (stationary or periodic) or low dimensional chaotic dynamics. Coupling such units in a network can result in highly complex spatio-temporal behaviour. Moreover, in such systems cooperative phenomena are observed that are emergent properties of the interactions and are expected to enable the design of new devices for applications in pattern recognition, signal analysis, associative memory and system control. Exploring these capabilities is the subject of ongoing research in the area of artificial and biologically oriented neural networks. In artificial neural networks the goal is to control and design networks composed of physical devices, which are often called *neurons*, for specific information processing tasks whereas the focus of biologically oriented research aims at an understanding of information processing in real neural systems.

7.2.1 Networks of Coupled Cells

Consider a system composed of different cells and assume that the state of each cell is described by a state variable $\mathbf{x}^i(t), i \in \mathcal{L}$, where $\mathcal{L}$ denotes a set of indices. The temporal evolution of a general *network of coupled cells* in the case of a continuous time evolution can be expressed by a system of coupled ODEs in

the form

$$\frac{d\mathbf{x}^i(t)}{dt} = \mathbf{f}^i\left(\mathbf{x}^i(t), \mathbf{u}^i(t), \mathbf{p}\right) + \epsilon\, \mathbf{h}^i\left(\left\{\mathbf{x}^j(t), \mathbf{u}^j(t)\right\}_{j\in\mathcal{L}'}, \mathbf{p}\right), \tag{7.1}$$

where $\mathbf{u}^i$ represents an external input on cell i. $\mathbf{p}$ represents a set of adjustable parameters. In the case of discrete time evolution the dynamics can be described by the coupled maps

$$\mathbf{x}^i_{n+1} = \mathbf{f}^i\left(\mathbf{x}^i_n, \mathbf{u}^i_n, \mathbf{p}\right) + \epsilon\, \mathbf{h}^i\left(\left\{\mathbf{x}^j_n, \mathbf{u}^j_n\right\}_{j\in\mathcal{L}'}, \mathbf{p}\right). \tag{7.2}$$

The function $\mathbf{f}^i$ describes the internal dynamics of the cell i. One often investigates networks of identical units, in which case $\mathbf{f}^i \equiv \mathbf{f}^1 := \mathbf{f}\ \ \forall i \in \mathcal{L}$. One refers to *coupled oscillators* if the dynamics of the single cells in isolation are capable of oscillations.

The function $\mathbf{h}$ describes the coupling of unit i to other units with indices $j \in \mathcal{L}'$. Typically one distinguishes *globally coupled networks*, *nearest neighbor networks* and *hierarchical networks*. In globally coupled networks it is $\mathcal{L} = \mathcal{L}'$ such that every cell experiences feedback from the entire system. For nearest neighbor networks it is $\mathcal{L}' = \mathcal{N}$, where the set $\mathcal{N}$ includes only nearest neighbors. In this case the units are usually fixed to some d dimensional grid and $\mathcal{N}$ contains all nearest neighbors on this hypercube lattice. For nearest neighbor networks the definition of boundary conditions, which specify the inputs on units at the lattice boundaries, is important. In more general cases the set $\mathcal{L}'$ will depend on the unit i. An example are hierarchical networks where units are connected in a tree-like structure and one unit plays the role of a *master unit* which *slaves* other units. If the set $\mathcal{L}'$ includes the index i itself, then one speaks of self-controlling feedback or self-regulation which is often observed in biological systems. See Figure 7.1 for illustrations of the different types of networks. If ϵ is small, such that the influence from $\mathbf{h}$ is small compared to $\mathbf{f}$, then one speaks of *weakly coupled networks*.

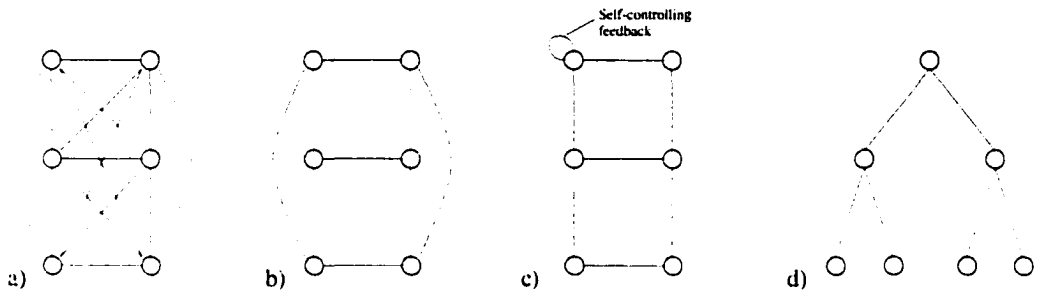


Figure 7.1: Different architectures of coupled neural cells. a) A globally coupled network. b) A nearest neighbor network with periodic boundary conditions. c) A nearest neighbor network with no-flow boundary conditions. Also shown is a cell with self-controlling feedback. d) A hierarchical network.

7.2.2 Applications of Coupled Cell Systems

Equations (7.1) and (7.2) are the prototype of many different systems such as micro-mechanical devices (MEMS) [HI00], Josephson junctions [ODG96], semiconductors [AD89], electrical power systems [WA92], neural networks and neural oscillators [Hop82, HI97], and play an important role in the modeling of neural information processing [SGK90, SB93, Des94, LB94] and the design of *smart matter* [GHH97, HH98].

Coupled Oscillators and Neural Information Processing

The classical work of Hopfield [Hop82, Hop84] initiated much research activity investigating the information processing abilities of artificial as well as biologically motivated neural networks. The original Hopfield networks were mainly used as associative memory for the retrieval of stored patterns and later also for the storage of temporal sequences of patterns [GDG90, GD91]. While classical Hopfield nets are of gradient type and hence are more or less static (but with many stationary states), more recent research in this area focuses on networks composed of 'neural maps' and 'neural oscillators' [HI97] which show much richer dynamics. This research was stimulated by the observation of rhythmic behaviour in

various biological networks such as neocortex, hippocampus and olfactory bulb [Fre86, Gra94, SF87], suggesting that synchronous oscillations are used to coordinate information processing in the nervous system. Based on the observation of both periodic and chaotic states in the olfactory bulb of rabbits, Skarda and Freeman [SF87] argued that periodic states encode odor information while a chaotic state serves as ground state from which all periodic states could be accessed. The idea of a control system, triggered by external stimuli, that stabilizes periodic states in a chaotic environment is therefore suggestive as possible mechanism underlying neural information processing [SGK90, Des94, SB93, LB94]. Thus the study of how oscillatory activity and transitions between different oscillatory states can be coordinated and controlled in networks of coupled cells appears to be of high relevance for understanding information processing in neural systems.

Smart Matter

Besides their biological relevance, the investigation of networks of coupled oscillators and other types of units, and in particular the question of controllability and design of such nets, is of importance for technological smart matter applications as proposed in [GHH97, HH98]. In a general context, smart matter can be viewed as a coupled cell system. Smart matter refers to materials whose properties can be changed through embedded microscopic control systems. The major challenge to realize such materials lies in the development of appropriate control programs [GHH97]. Guenther *et al.* [GHH97] proposed a multiagent control system using simple learning mechanisms to adapt their behaviour. We propose to base such control programs on reinforcement learning which should result in stable and adaptive control algorithms.

7.3 Control of Coupled Cells Systems

Consider a general complex system of coupled cells. In order to actively monitor the system, sensors are embedded in the system at various locations (see Figure 7.2). They can either monitor certain unit variables or the interactions between units. Associated with the sensor devices are *computational agents* or control devices which compute appropriate control signals on the basis of the state information provided by the sensors.

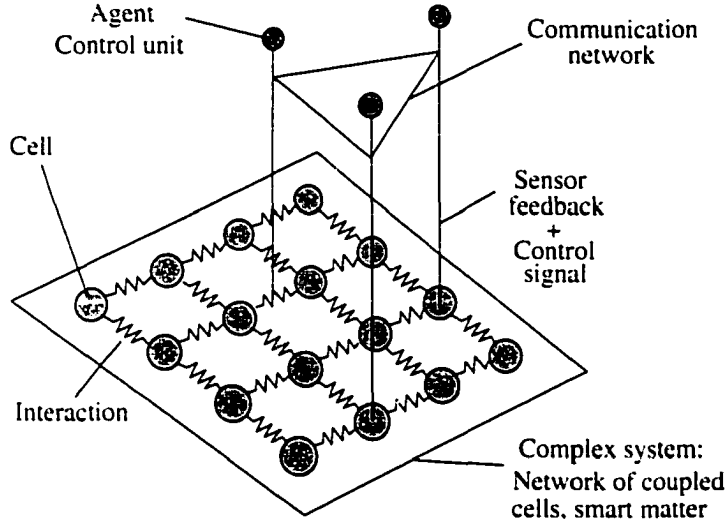


Figure 7.2: Model of a complex system in terms of coupled cells. Distributed agents apply control signals on the basis of state information provided by distributed sensors.

The sensors are possibly connected through a communication network that allows an agent to access sensor information from other sensors [GHH97]. If each agent can independently decide over its actions, then this architecture describes a *multiagent control system* which attempts to control the globally coupled system through cooperative control of single cells. If a certain agent takes the role of a *master control unit* which slaves other control units one obtains a *master-slave* control architecture. One further distinguishes *global control* where every single

unit has a control unit attached, and *distributed*, or *pinning control*, , where control is only applied to a few selected units. If agents receive sensor information only from a small neighborhood we refer to *control through local feedback*. If the agents process information about the global state of the system we refer to *control through global feedback*. The global approach requires rapid access to the full state of the extended system and detailed knowledge of its behaviour and is typically only feasible if global state variables such as an electric field or a density exist which can be easily measured. In the context of control of complex systems the distributed agent approach allows more flexibility and is believed to be more robust [GHH97].

Distributed control of extended systems involves three major questions: 1) How can we find the optimal distribution of the agents? 2) How can we find the optimal structure of the communication network for a given set of agents? 3) How can we find optimal control policies for the agents. The first problem was addressed in [KHL99, GCS97] where the optimal pinning density for the control of coupled map lattices was investigated. The second question was addressed in [GHH97], where a simple learning scheme was introduced to find an optimal communication network. Concerning the third question control signals to date are almost exclusively based on time-delayed feedback signals. This typically requires the possibility to directly perturb state variables. In the following we will demonstrate the control of coupled map lattices through agents which update their policies on the basis of immediate rewards through temporal-difference learning. This approach can be applied to time-discrete parametric or impulsive control as discussed in the previous chapter.

7.3.1 Literature Review

In this section we give a brief overview over existing approaches for the control of spatio-temporal systems. Practically all methods for the control of spatio-temporal chaos are based on either additive time-continuous feedback methods as

proposed by Pyragas (see Subsection 3.4.3) or adaptive parameter perturbation methods (see Paragraph 3.4.1).

Control of Coupled Maps

Most numerical investigations have dealt with the control of coupled map lattices (CML). Spatio-temporal chaos is often created by local nonlinear dynamics and spatial diffusion [Kan92]. Prototypes for this type of spatio-temporal chaos are 1-D *diffusively coupled map lattices*

$$x_{n+1}^{i} = (1-\epsilon) f\left(x_n^i\right) + \frac{\epsilon}{2}\left[f\left(x_n^{i-1}\right) + f\left(x_n^{i+1}\right)\right]. \tag{7.3}$$

or the corresponding 2-D version

$$x_{n+1}^{i,j} = (1-\epsilon) f\left(x_n^{i,j}\right) + \frac{\epsilon}{4}\left[f\left(x_n^{i-1,j}\right) + f\left(x_n^{i+1,j}\right) + f\left(x_n^{i,j-1}\right) + f\left(x_n^{i,j+1}\right)\right]. \tag{7.4}$$

where the integers i, j mark discrete spatial locations. If the local map $f(x)$ is chosen to be the logistic map $f(x) = p_0 x(1-x)$ one obtains a diffuse *logistic coupled map lattice* (LCML). The stability of spatio-temporal patterns occurring in the dynamics of these equations was analyzed in [ZG94b, ZG94a, XH97]. The patterns are either of I_s, III_0 or I_0 type. The diffuse LCML were controlled with the aim state as additive feedback for 1-D LCML in [GZ94, KHL99, PHE97, PPS98] and 2-D LCML in [JAPE97]. Time-delayed additive feedback control was considered in [PHE97, Gad98, KHK98, KK99] for the LCML and in [KH99] for a coupled map car-following model. Adaptive parameter control on the basis of knowledge about the aim state was considered in [Aue94, SG98, PKK98]. Astakho *et al.* [AAS95] applied the OGY method to a 1-D diffuse LCML, analytically computing the necessary parameter perturbations. Optimal pinning densities were considered in [KHL99, GCS97]. The related topic of synchronization of such CMLs was discussed in [ZL97, JP98, JGT+98].

A different type of CML is often studied in the context of biological or physical systems such as biological networks, Josephson junctions arrays or ecological systems: the *globally coupled map lattice*

$$x_{n+1}^i = (1-\epsilon) f\left(x_n^i\right) + \frac{\epsilon}{N} \sum_{j=1}^{N} f\left(x_n^j\right). \tag{7.5}$$

where N denotes the number of coupled maps. In these systems the phenomenon of clustering can be observed. Elements split into cluster, with all elements in a cluster oscillating in synchronization. Dynamically, the system spontaneously switches among different ordered states. Between the ordered phases typically disordered phases are observed [Kan89]. The switching process is known as chaotic itinerancy over attractor ruins [Kan92] or more general as *multistability* and will be discussed for a simple mechanical kicked rotor in the next chapter. The dynamics of a globally coupled map lattice was investigated in [Kan89] where it was also demonstrated how switching between states can be induced through single site perturbations. The spatio-temporal control of a globally coupled neural network on the basis of partial knowledge of the target states was demonstrated in [KLO96] as a model for an associative memory mechanism in the brain.

Control of Coupled ODEs

To the authors knowledge the first work on the control of chaos in coupled networks of oscillators was an article by Sepulchre and Babloyantz [SB93], who extended the OGY control technique to the stabilization of unstable orbits in a 9×9 network of oscillators. However, instead of parametric control they followed the ideas of [GSDW92] and applied impulsive control. They point out important implications of the possibility of chaos control for the explanation of cognitive processes and neural information processing. In a later work, Lourenço and Babloyantz [LB94] applied time-delayed feedback control to a network of excitatory and inhibitory neurons, describing cortical activity. The control of coupled

nonlinear oscillators through distributed control was first investigated by Kocarev *et al.* [KPSJ97]. Controllers located periodically in space applied periodic control signals which forced state variables to new values which were projections of the aim state. Hu *et al.* [HXCP98] applied distributed feedback pinnings to a model of two-dimensional arrays of Chua's circuits. As feedback signal the aim state was fed back into the system. The investigated array has a spiral wave attractor which is believed to play a role as a cause of cardiac disease. They were able to migrate the system from the spiral wave state to a coherent oscillation by only pinning a small number of cells. In [HYL98] the dependence of the controllability on pinning density and couplings was investigated in more detail.

Control of PDEs

The control of strongly anisotropic one-dimensional open-flow systems was demonstrated in [GK93] through time-delayed feedback control by pinning a single unit. Aranson *et al.* [AnLT94] stabilized a structurally unstable topological defect, whose analytical expression is known, by adding an extra term in the complex Ginzburg-Landau equation (CGLE). Stabilization of a plane wave extended through the entire system has been achieved by adding time-delayed feedback terms to the CGLE either through global control based on local information [BS96] or based on global information [BM96]. Instead of adding a time-delayed feedback term, Montagne and Colet [MC97] demonstrated control of the CGLE by adding a nonlinear diffusion term. More recently, pinning time-delayed feedback control, based on global information has been applied to the control of wave instabilities in cardiac tissue [RFK99], and based on local information to an El Niño prediction model [TS97]. Global time-delayed feedback control has been applied to a glob-

ally coupled reaction-diffusion system describing charge transport in a bistable semiconductor [FBS99].

In summary we can say that practically none of the existing approaches for the control of spatially extended systems have considered parametric control as proposed in the original OGY scheme, i.e. control through small parameter perturbations. It is fair to assume that in many practically applications the time-continuous perturbation of system variables will not be feasible. Then either parametric of impulsive control is a necessity. In the following section we discuss the parametric control of LCMLs through the chaos control method introduced in the previous chapter.

7.4 Learning Control of Coupled Map Lattices

In this section we present the control of diffuse LCMLs through reinforcement learning. Part of these results were previously presented in [GD00a].

7.4.1 The Control Method

In Chapter 6 we showed the control of a single logistic map through reinforcement learning. Here we extend this method to the control of the 1-D and 2-D diffuse LCML (7.3) and (7.4) with $f(x) = p_0 x(1-x)$ and $p_0 = 3.8$. Our approach to control the system is based on small discrete state dependent perturbations to either the coupling strength ϵ or the system parameter p_0. Here we apply control to every lattice site, such that the equation for the controlled 1-D lattice can be written as

$$x_{n+1}^i = \left(1 - \epsilon\left(x_n^i\right)\right) f\left(x_n^i\right) + \frac{\epsilon\left(x_n^i\right)}{2}\left[f\left(x_n^{i-1}\right) + f\left(x_n^{i+1}\right)\right], \quad i = 1, \cdots, i_d,$$

where now $\epsilon(x) = \epsilon_0 + \epsilon_u u_\epsilon(x)$ and $f(x) = (p_0 + a_u u_a(x))x(1-x)$. Here, $u_a(x)$ or $u_\epsilon(x)$ represent the discrete control perturbations. We assume periodic boundary

conditions. Distributed control would be obtained by applying control signals only to a subset of sites. We restrict the set of allowed values for the control to $u_\epsilon \in \mathcal{U}_\epsilon = \{0, \Delta\epsilon, -\Delta\epsilon\}$ and $u_a \in \mathcal{U}_a = \{0, \Delta a, -\Delta a\}$, where $\Delta\epsilon$ and Δa are the "amplitudes" of the control signal and ϵ_u and a_u determine which parameter is controlled: $(\epsilon_u, a_u) = (0, 1)$ or $(1, 0)$ (we could also choose to perturb both parameters). Although we consider only two small, nonzero perturbations of one of the parameters, the algorithm leads to a successful control performance. Which value for u ($= u_\epsilon$ or u_a) is chosen depends on the current state x^i and is determined from x^i according to a certain control policy $\pi(x^i)$, which associates a control u to x^i.

In principle each control unit could establish its own "optimal" control policy independently, here however we use one "master" policy for all cells.

7.4.2 Establishing the Master Policy

The master policy is established by monitoring the local neighborhood of one chosen unit. The other units perform perturbations by using this master policy to choose controls from the current local state, but do not change the master policy. Only one "master" unit is responsible for updating the globally used policy. Thus we apply a master-slave control architecture with global control through local feedback.

To establish the master policy, we first discretize the state space $\mathcal{X} = [0, 1]$ of a single logistic map. As before we chose a set of 100 reference points w distributed uniformly over $\mathcal{X}$ which form the reduced state space $\mathcal{W}$. To every $x_n \in \mathcal{X}$ we define $w_n := w(x_n) \in \mathcal{W}$ to be the reference point in $\mathcal{W}$ which is closest to x. The optimal control policy $\pi(x)$ is then approximated in the discretized state space $\mathcal{W}$ as $\hat{\pi}(x) \approx \pi(w(x))$. To compute $\pi(w)$, we associate to each possible state-action pair (w_μ, u_ν) a state-action value $Q_{\mu\nu} = Q(w_\mu, u_\nu)$ and establish the

optimal Q^* through Q-learning as discussed in the previous chapter. The control goal represents only the general characteristics of the target state and is defined on the basis of the local information from the chosen master unit with lattice coordinates (i_m, j_m) and possibly its nearest neighbors. Let T_k denote a pattern with temporal period k and S_n a pattern with spatial period n. We attempted to stabilize three different patterns.

(T_1S_1) This corresponds to a fixed point pattern where all units are stabilized at a fixed point. To stabilize this pattern no information from neighboring units is necessary. The goal state can be defined through the associated reference vectors in the form $w(x_n^{i_m}) = w(x_{n+1}^{i_m})$.

(T_2S_1) This corresponds to a synchronous time period 2 state where all units oscillate in phase. To stabilize this pattern we used the criterion $w(x_n^{i_m}) = w(x_{n+2}^{i_m})$, $w(x_n^{i_m}) \neq w(x_{n+1}^{i_m})$, $w(x_n^{i_m}) = w(x_n^{i_m-1})$.

(T_2S_2) This pattern corresponds to the asynchronous time period 2 pattern where two neighboring units oscillate in anti-phase. The goal criterion for this pattern was set to be $w(x_n^{i_m}) = w(x_{n+2}^{i_m})$, $w(x_n^{i_m}) \neq w(x_{n+1}^{i_m})$.

We used two approaches to control large systems of coupled logistic maps. In the first approach we learn the policy for a small system and use the resulting state-action value function as policy for all units in a larger system. In the second approach we work directly with the larger system, but establish the globally used policy only on the basis of local information from a chosen master unit. In all cases the master unit has the coordinate $i_m = 2$ in the 1-D case and $i_m = j_m = 2$ in the 2-D case.

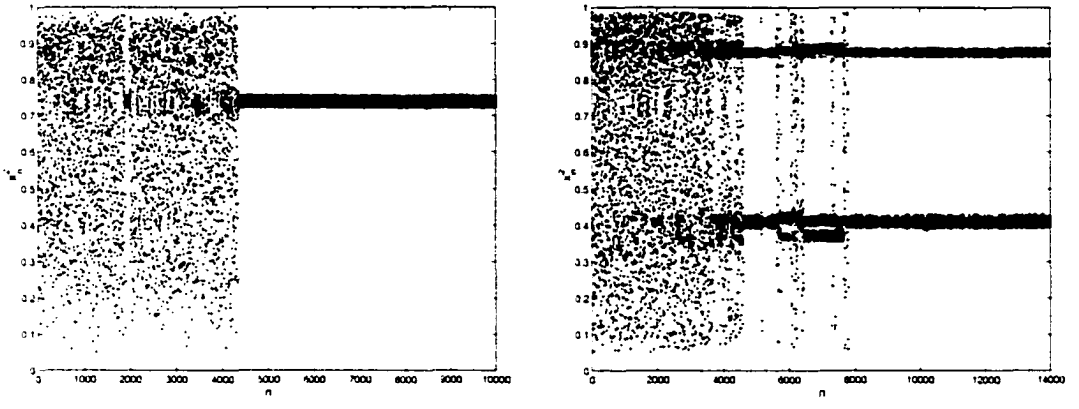
7.4.3 Simulation Results

In all simulations described below, we set $p_0 = 3.8$ and pick ϵ such that the pattern to be controlled is unstable (see [ZG94b, ZG94a, XH97]) (we discuss only weak couplings here). The parameters β, γ, r_{n+1} of the control algorithm are chosen as in the previous chapter. Initially the Q values are all set to zero.

Control of 1-D LCML

Controlling a Small Lattice. First we established a successful control policy by controlling a small 1-D LCML with $i_d = 3$ and $j_d = 1$ through an on-line approach. We established control of the T_1S_1, T_2S_1 and T_2S_2 patterns. The resulting state-action value functions are denoted Q_1, Q_{s2} and Q_{a2}, respectively. In Figure 7.3 a) we show the dynamics using on-line control for the "master" map, x_n^2. The system was started in a random initial condition. After approximately 4,500 iterations a successful policy Q_1 is established. For the 1-D pattern we set $\epsilon = 0.1$. The control parameters are $a_u = 1$, $\Delta a = 0.15$ and $\epsilon_u = 0$. Figure 7.3 b) shows x_n^2 for on-line control of an asynchronous period 2 pattern. For the period 2 pattern we set $\epsilon = 0.05$ (for larger ϵ the period 2 pattern becomes attractive [XH97]). To stabilize the period 2 pattern we chose $a_u = 1$, $\Delta a = 0.15$ and $\epsilon_u = 0$. The asynchronous period 2 pattern could also be stabilized with the parameters $\epsilon_u = 1$, $\Delta\epsilon = 0.025$ and $a_u = 0$.

Controlling a Large Lattice by Learning in the Large Environment. In a similar fashion, a larger 1-D LCML ($i_d = 100$) could be controlled. The map with $i = 2$ was chosen as the master system. The globally used policy was updated on the basis of the information from this unit only (and its neighboring unit for the stabilization of the asynchronous period 2 pattern). Figure 7.4 shows on-line control of this large LCML onto the period 1 pattern. We see that the number of



(a) T_1S_1 pattern (b) T_2S_2 pattern

Figure 7.3: On-line control of a 1-D LCML with three elements. a) Stabilization of a period 1 pattern. b) Stabilization of the asynchronous period 2 pattern. The parameters are as stated in the text.

iterations until the target state is stabilized is about the same as for control of the small LCML. In a similar fashion the other two patterns could be stabilized in the large lattice.

Controlling a Large Lattice by Using a Policy Approximated from a Small Lattice. It is interesting to see if the control policies Q_1, Q_{s2} and Q_{a2}, which were approximated by interaction with a small lattice, are also able to control a large lattice. This might be relevant in hierarchical coupled or growing systems where a control policy might be learned in a small subsystem which takes the role of a "master" cell slaving other units after a growth process of the network occurred. To this end we used the policies Q_1, Q_{s2} and Q_{a2} to control a 1-D lattice with $i_d = 100$. During control, the policies were not updated further. Control onto the T_1S_1 pattern is shown in Figure 7.5. Starting from a random initial condition, the system is stabilized into a global fixed pattern after approximately 200 iterations. It is apparent that once a successful policy is established, control is

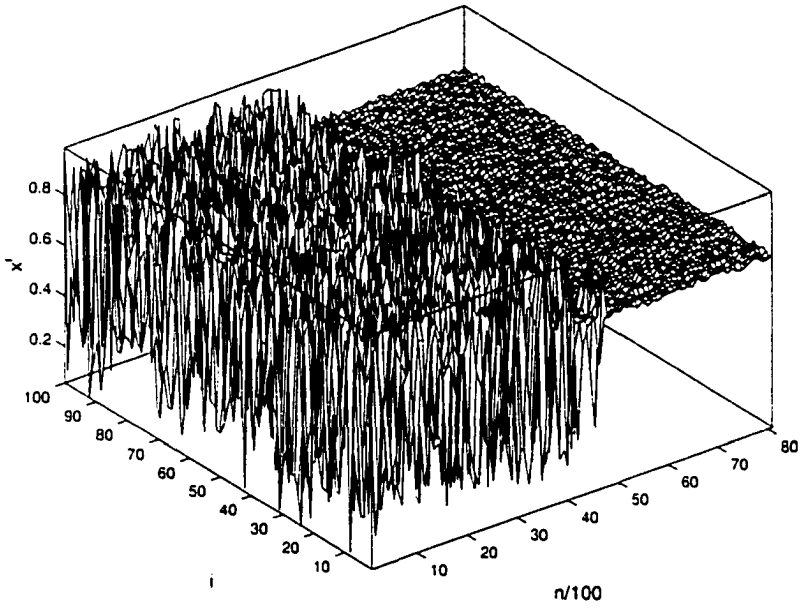


Figure 7.4: On-line control of a 1-D LCML with $N = 100$ elements.

achieved quickly. Although we had used on-line control to establish this policy, it is acting globally as indicated by the fast convergence to the desired target state. Off-line control could further reduce this transient but not significantly. We observed this frequently in cases where the system visits most of the state space before on-line control is established. Similar results were obtained by using the policies Q_{s2} and Q_{a2} for the synchronous and asynchronous period 2 patterns, respectively, as shown in Figure 7.6.

Control of 2-D LCML

The 2-D LCML could be controlled in a similar fashion as the 1-D LCML. We used the previously established policies Q_1, Q_{s2} and Q_{a2} and a 2-D LCML with $i_d = j_d = 20$. In Figure 7.7 different time steps of the control of this 2-D lattice into a synchronous period 2 pattern are shown. The system settles quickly into a global period 2 pattern. However, it takes longer to reach a globally stable synchronous pattern. The asynchronous state on the other hand is easier to stabilize. The state

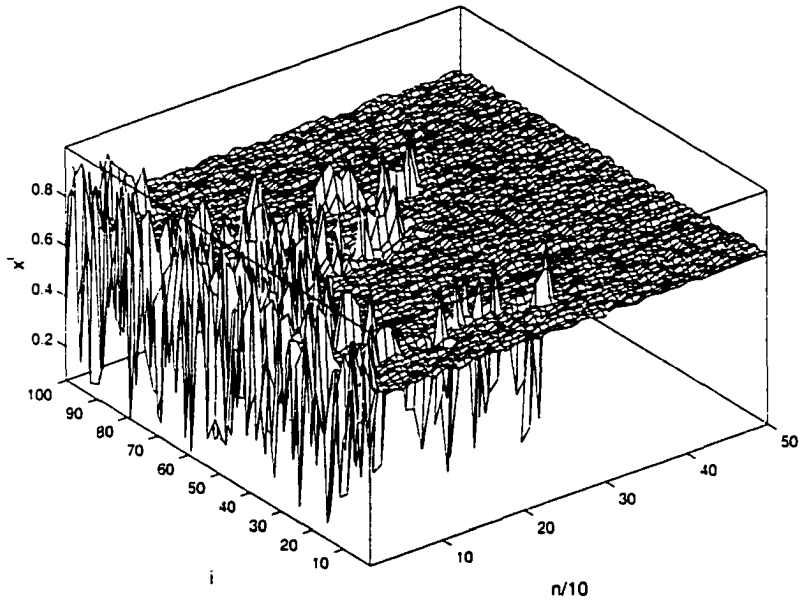
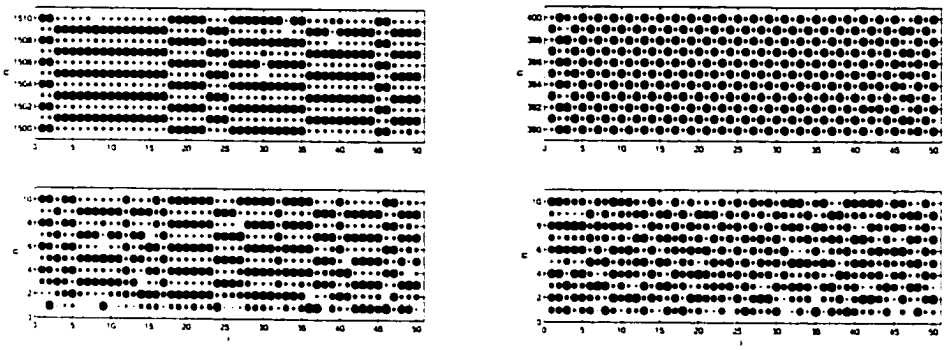


Figure 7.5: Control of a 1-D LCML with $i_d = 100$ elements using the policy Q_1 established through control of a LCML with $i_d = 3$.

for $n = 1000$ is shown in Figure 7.8 where the policy Q_{a2} was applied to control the 2-D LCML.

Various modifications of our control strategy are possible. For example, different subsystems may use different policies to establish control of a variety of patterns. To illustrate this, we show in Figure 7.9 the control of a 2-D lattice into a combination of a period 1 and period 2 pattern for $\epsilon = 0.05$. The subsystems with $i \leq 10$ were controlled according to policy Q_1, while the subsystems with $i > 10$ were controlled according to policy Q_{s2}. We see that after 1,400 iterations the system has reached a combination of a period 1 and a period 2 pattern.

To check performance under noisy conditions we added uniformly distributed noise $\eta_n \in \{-\alpha/2, \alpha/2\}$ to x^i at each iteration step and found that control could be established up to noise levels of $\alpha \approx 0.06$ if p was perturbed. With ϵ as control parameter we found successful control up to $\alpha \approx 0.02$.



(a) T_2S_1 pattern

(b) T_2S_2 psttern

Figure 7.6: Control of 1-D LCML with $i_d = 50$ to a) synchronous period 2 pattern using policy Q_{s2}, and b) asynchronous period 2 pattern using policy Q_{a2}. The iterations n go from bottom to top. The x-axis corresponds to the different cells. The size of the dots is proportional to the value of x_n^i (largest for $x = 1$).

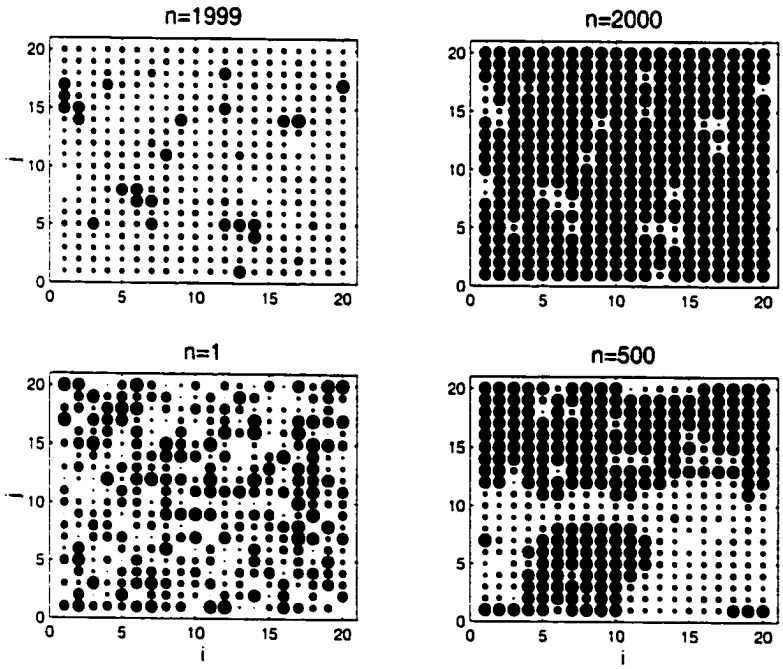


Figure 7.7: Control of a 2-D LCML with 20×20 elements into a synchronous period 2 pattern by using the policy Q_{s2}. Shown are snapshots at $n = 1$, $n = 500$, $n = 1999$ and $n = 2000$. The size of the dots is proportional to the value of $x_n^{i,j}$ (largest for $x = 1$).

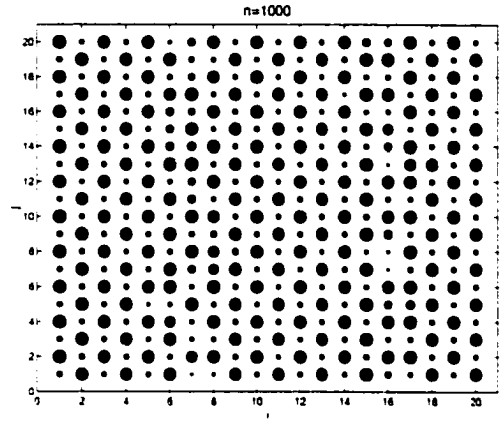


Figure 7.8: Control of a 2-D LCML with 20×20 elements into an asynchronous period 2 pattern by using the policy Q_{a2}. Shown is a snapshot of the controlled system after 1000 iterations. This target state is stabilized much faster than the T_2S_1 pattern.

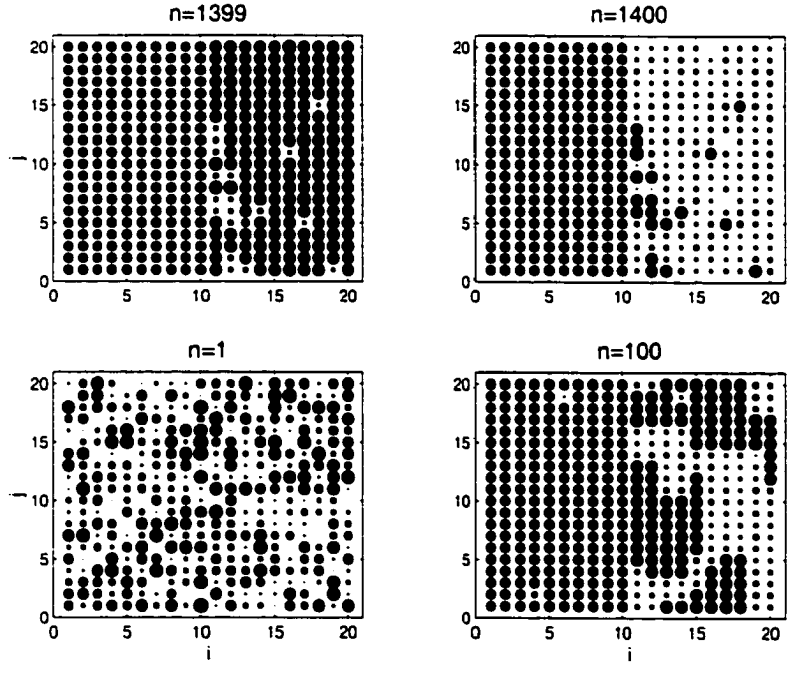


Figure 7.9: Control of a 2-D LCML with 20×20 elements into a combination of period 1 and period 2 patterns.

7.5 Conclusions

In this chapter we demonstrated the control of 1-D and 2-D LCML using a method based on reinforcement learning. We were able to successfully stabilize different period 1 and period 2 patterns for small coupling strengths for which the desired pattern is unstable. The control was fed into the system by choosing an optimal state dependent perturbation of a control parameter in a discrete set of a priori fixed small values. A successful control policy could be established either through interaction with a large LCML or a small LCML. The fact that our method allows a successful control policy of a large system through interaction with a small subsystem might lead to interesting perspectives for information processing, in particular for biological systems. Specifically we have shown that the combination of different policies in different spatial regions can trigger the formation of new patterns and hence may open promising directions for *smart matter applications* (see [GHH97]).

Chapter 8

CONTROL OF MULTISTABLE SYSTEMS

> In conclusion, we see that the myriad of possible behaviours in a complex system is of great utility if we are able to harness it.
>
> (L. Poon and C. Grebogi, 1995, p. 4026)

In this chapter we present the control of a simple model for a multistable system, a kicked mechanical rotor, with the control method discussed in Chapter 6. After describing and deriving the model equations we show how control of selected states of the rotor can be achieved through reinforcement learning even under the influence of considerable noise.

8.1 Introduction

The long-term behaviour of nonlinear dynamical systems is generally classified as either stationary, periodic, quasi-periodic or chaotic. These types of behaviours and their control are well studied and understood if the available states are well separated and their dimension rather low. We discussed the control of such systems in their chaotic regime in Chapters 3 and 6. In recent years the attention has shifted to systems exhibiting spatio-temporal behaviour and systems with many coexisting attracting states. In the previous chapter we discussed the control of spatio-temporal chaotic systems. In this chapter we discuss systems with coexisting attracting states. In general the term "complexity" has been coined to denote systems that have both elements of order and elements of randomness [PG95]. Such systems typically, but not necessarily, have many degrees of freedom, are

composed of many complicated interrelated parts and possess competing attracting sets.

In general, each attractor has associated with it a *basin of attraction*, i.e. the set of initial states converging to the attractor. Complex systems with coexisting attractors possess complexly interwoven basins of attraction such as *fractal boundary basins* [PG95], *riddled basins* [AKYY92] or *intermingled basins* [AHKY96]. Given multiple coexisting attractors and their respective basins of attraction, a basin is riddled if any open set intersecting the basin intersects the complement of the basin in a set of positive Lebesgue measure. The basin is intermingled if the open set intersects each of the other basins in a set of positive measure [AHKY96]. For riddled and intermingled basins each basin point is a basin boundary.

In systems with unriddled basins we can find neighborhoods of the basin in which the probability of finding trajectories leading to a different final state is zero. These basins possess a subset of points forming the basin boundary. Often the basin boundary will be fractal and have embedded an infinite number of unstable invariant sets. For systems with fractal basin boundaries separating multiple attracting states, due to the nontrivial relationship between the coexisting states and their basins of attraction, a final state depends crucially on the initial conditions [KFG99]. This behaviour is called *multistability* and was first studied experimentally in [AMPT82] and since then was observed in a variety of systems from different areas such as physics [PWS94, BBL+91, BLP+91], chemistry [MKH91] and neuroscience [FMM97]. Adding noise to a multistable system will generate complex behaviour and induce competition between the attractiveness towards regular motion in the neighborhood of an attracting state and the transitions between basins of attractions induced by noise [KFG99]. The dynamics is then characterized by a large number of periodic attractors "embedded" in a sea of transient chaos [PG95]. The time the system spends in an attracting state

corresponds to its "ordered" phase, and the transient time to its "random" phase. Added noise can prevent the system from settling down into an ordered phase.

Besides their importance for specific applications, a further motivation to study the dynamics and control of such complex systems is their possible role as information processing devices in neural information processing [FLMM96]. Complex systems are characterized by a large number of coexisting states and, with adequate noise, the system can rapidly access these ordered states. The control of such systems, in particular under noisy conditions, would offer the opportunity to utilize this multistable behaviour for the processing and storage of information, i.e. different ordered states are identified with different "memorized" pieces of information. External input can be thought of as triggering a certain control mechanism that stabilizes a selected ordered state which would be associated with the given input.

The simplest prototype of a complex multistable system is provided by the model equations of a periodically kicked mechanical rotor [FGHY96, FG97, KFG99] whose quantum mechanical counterpart plays an important role in the study of quantum chaos [Cas96]. Until now control was achieved for low noise levels through a simple feedback mechanism [FG97] which perturbs directly all system variables and requires computation of the Jacobian of the map close to the desired state. Moreover, this control technique is only local, i.e. the control is usually switched on only if the system is close to the desired state. In [FG97] the Jacobian was computed from the model equations. In many real world applications this information will not be available and specifically in the context of neural information processing it is unrealistic to base control methods on the basis of analytical knowledge of governing system equations. In some cases, the Jacobian can be estimated from observed data as suggested in [OGY90]. In the presence of noise however, this estimation can become difficult.

In the control of chaotic systems the goal is to stabilize unstable states embedded in the chaotic attractor. Neighborhoods of the desired state are visited with probability one due to ergodicity and global control is designed to reduce transient times. To control multistable systems the desired state is typically chosen to be one of the many existing attracting states. These states are *metastable* (stable only for a finite time) above a certain noise level ($\delta > 0.06$ for the rotor described below). Here, the control must stabilize dynamics against noise. Noise however is a necessary ingredient to allow the system to hop from attractor to attractor. A good control strategy globally acts to destabilize undesired states by increasing the probability that the system leaves these states, while locally stabilizing the desired state against noise.

We apply the method discussed in Chapter 6 to the parametric control of a periodically kicked rotor and we will see that the method developed for chaotic systems is also suited for the control of complex multistable systems in the presence of significant noise levels.

8.2 Basic Equations

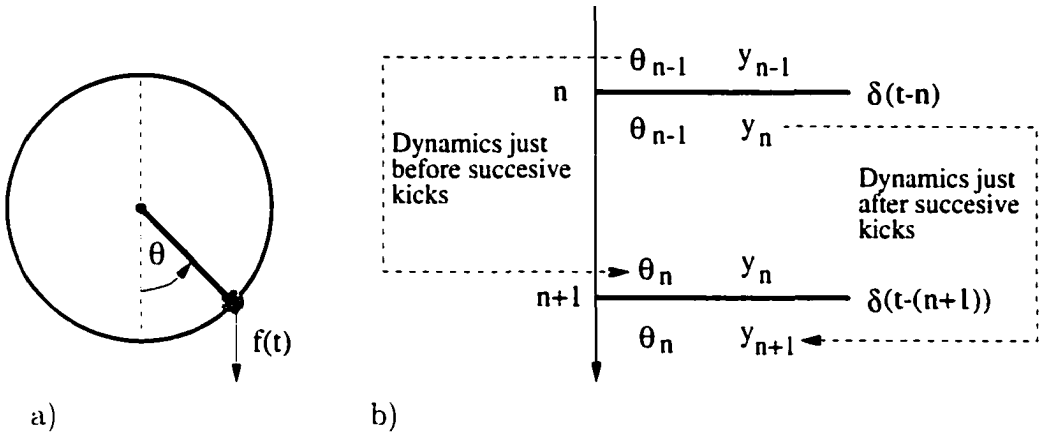
The differential equation describing the temporal evolution of the phase angle θ of a forced damped pendulum with forcing $f(t)$ and damping d (see Figure 8.1a) is given by

$$\theta'' + d\,\theta' = f(t)\sin\theta. \tag{8.1}$$

If the external force acts periodically and impulsively on the rotor,

$$f(t) = f_1 \sum_n \delta(t-n), \tag{8.2}$$

the dynamics is most conveniently described by its return map. Let $\theta(t)$ be the solution at time t and let $\theta_n = \theta(n)$ be its value at the nth kick. Due to the



a) b)

Figure 8.1: a) Example of a mechanical rotor: a weight is attached by a rod to a pivot. b) Illustration of the two descriptions (8.9), describing the rotor just before, and (8.10), describing it just after two successive kicks.

δ-forcing the velocity $\theta'(t)$ shows a discontinuity at $t = n$.

$$\theta'(n+0) - \theta'(n-0) = f_1 \sin\theta_n. \tag{8.3}$$

whereas $\theta(t)$ is continuous. The solution between two successive kicks is then given by

$$\theta(t) = \theta_n - \frac{l_n}{d}\left(e^{-d(t-n)} - 1\right), \quad n \le t \le n+1, \quad l_n := \theta'(n-0) + f_1 \sin\theta_n. \tag{8.4}$$

It follows

$$\theta_{n+1} = \theta_n - \frac{l_n}{d}\left(e^{-d} - 1\right), \tag{8.5}$$

and

$$l_n = l_{n-1}e^{-d} + f_1 \sin\theta_n. \tag{8.6}$$

For simplicity we set $c = 1 - e^{-d}$ $(0 < c \le 1)$. Equation (8.5) with n replaced by $n-1$ yields $u_{n-1} = (d/c)(\theta_n - \theta_{n-1})$ and from (8.6) we obtain

$$l_n = \frac{d}{c}(1-c)(\theta_n - \theta_{n-1}) + f_1 \sin\theta_n. \tag{8.7}$$

which, when substituted back into (8.5), leads to a "finite-difference form" for the phase angle θ_n.

$$\theta_{n+1} - 2\theta_n + \theta_{n-1} + c(\theta_n - \theta_{n-1}) = f_0 \sin\theta_n, \tag{8.8}$$

where $f_0 = cf_1/d$. By introducing the new variable $y_n = \theta_n - \theta_{n-1}$ we obtain from (8.8) the *dissipative standard map*

$$\begin{aligned} y_{n+1} &= (1-c)y_n + f_0 \sin\theta_n. \\ \theta_{n+1} &= \theta_n + y_{n+1} \pmod{2\pi}. \end{aligned} \tag{8.9}$$

which is related to the system just before successive kicks (see Figure 8.1b). Introducing the variable $v_n = y_{n+1}$ we can rewrite this map in the form

$$\begin{aligned} \theta_{n+1} &= \theta_n + v_n \pmod{2\pi} \\ v_{n+1} &= (1-c)v_n + f_0 \sin(\theta_{n+1}). \end{aligned} \tag{8.10}$$

which describes the state of the system just after two successive kicks. The map (8.10) was extensively studied in [FGHY96]. For $c = 0$ it results in the *Chirikov standard map* [Chi79].

In this undamped, Hamiltonian limit the state space consists of a chaotic sea interspersed with regular motion represented by stability islands of stable periodic orbits. The largest regions of regular motion are made up by the islands of the primary periodic orbits. These are the fixed points (period 1, $\theta = \pi, v = 2m\pi$, $m = 0, \pm 1, \cdots$) and higher period periodic orbits that are present for $f_0 = 0$. Further, secondary stable periodic orbits occur for $f_0 \neq 0$. Their islands are grouped around the primary islands and are in general much smaller, inducing a hierarchical organization of the stable periodic orbits [FGHY96, LL92]. We note that in the undamped case the range of the velocity v can also be identified with a circle (v mod2π), i.e. the infinitely extended cylindrical phase space is compactified to a torus. On the torus the infinite family of primary fixed points is represented

by a single point, but the number of all periodic orbits is still assumed to be infinite [LL92].

On the other hand, for very strong damping ($c \approx 1$) one obtains the one-dimensional *circle map* with a zero phase shift,

$$v_{n+1} = v_n + f_0 \sin v_n, \tag{8.11}$$

which exhibits the usual Feigenbaum scenario for the transition to chaos. In particular this map possesses only one attractor in large regions of the phase space [LL92].

When a small amount of dissipation ($0 < c \ll 1$) is added to the undamped system, stable periodic orbits turn into sinks or disappear. The parameter dependence of the sinks and their basins of attraction have been studied numerically by Feudel *et al.* [FGHY96]. We shortly summarize some of the main results of this study. For $c \neq 0$ the range of v can no longer be identified with a circle. The phase space is now an infinitely extended cylinder on which we find an infinite number of stable periodic orbits in the limit $c \to 0$, in particular the primary family of fixed points, denoted P_1. For $c > 0$ all trajectories are eventually trapped in the finite cylinder $|v| \leq f_0/c$ that contains all attractors. The number of sinks is now finite but can be made arbitrarily large by choosing c sufficiently small. Specifically only a finite number of the formerly infinite P_1 family can be present. These fixed points still have $v = 2\pi m$ but the phases are now different and the size of the attraction basins quickly decreases with increasing $|m|$. The main fixed point, P_1^0 ($m = 0$), has the largest basin. In addition, when f_0 is varied one finds births of sinks in saddle node bifurcations and their disappearance in period doubling sequences, but the resulting chaotic parameter regimes are extremely small.

Concerning basin boundaries, these appear all to be fractalized giving rise to chaotic transients along chaotic saddles and hence to uncertainty in initial conditions. The question to which extent the basins are riddled has only partly been

addressed in [FGHY96]. The basins of the P_1-family contain full (though small in size for larger m) neighborhoods of the fixed points and therefore cannot be fully riddled, but partial riddling as defined in [AT00] is not excluded. The basins of other periodic orbits are smaller and full riddling might occur in some cases, but this requires further investigation. In summary, the kicked rotor with small dissipation serves as an example of a multistable system characterized by a complicated coexistence of many periodic sinks with sensitive dependence on initial conditions.

The complexity of multistable systems can further be enhanced through the introduction of noise which leads to unpredictable transitions between different attracting states revealed by almost periodic motion interspersed by random bursts [KFG99]. In addition Kraut *et al.* [KFG99] observed a decrease in the number of accessible states. Their results indicate that the noise induces a preference of certain attractors.

In this chapter we show that the multistable rotor can be stabilized at a desired attracting state through the method discussed in Chapter 6. We show that control can be achieved up to noise levels as high as $\delta = 0.4$ (see equation (8.12) below). As control parameter we choose the forcing f_0. The allowed control actions consist of small discrete perturbations of f_0.

8.3 The Noisy Uncontrolled Rotor

The system investigated by Kraut *et al.* [KFG99] has the form

$$\begin{aligned} \theta_{n+1} &= \theta_n + v_n + \delta_\theta \pmod{2\pi} \\ v_{n+1} &= (1-c)v_n + f_0 \sin(\theta_{n+1}) + \delta_v, \end{aligned} \tag{8.12}$$

where δ_θ and δ_v are the components of the uniformly and independently distributed noise vector with bounded norm: $\sqrt{\delta_\theta^2 + \delta_v^2} \leq \delta$. Throughout this chapter we set the unperturbed forcing to $f_0 = 3.5$ and the damping to $c = 0.02$. For these parameter values Kraut *et al.* [KFG99] found numerically 111 stable periodic orbits

in the noiseless limit. Most of these orbits belong to the P_1-family ($\theta = \theta_m, v = 2m\pi$) and some of them have period 3. Only 0.01% of all found orbits have other periods than 1 and 3, so these orbits do not play an important role. With noise added, Kraut *et al.* [KFG99] observed three different types of behavior. For small noise level ($\delta \lessapprox 0.05$) the trajectory may be trapped in the open neighborhood of an attractor forever. For intermediate noise level ($0.05 \lessapprox \delta \lessapprox 0.1$) attracting periodic orbits still could be identified. However, when an attracted region around such an orbit is reached, the noise will eventually drive the system to the basin boundary where it then follows for a while a chaotic saddle until it reaches a neighborhood of another attractor. The resulting dynamics is characterized by almost periodic motion interrupted by bursts, and as smaller the noise is as larger are the regular phases. This behavior is referred to as attractor hopping and may also be considered as noise-induced chaos. In a sense, sinks of the noiseless system turn into saddles of a "noise-induced chaotic attractor". Controlling a periodic orbit in this regime resembles the control of an unstable periodic orbit embedded in a chaotic attractor of a deterministic system. However, while in the deterministic case unstable directions have to be avoided by forcing the system to remain on the stable manifold, in the noisy case random pushes towards the basin boundary have to be suppressed. It is therefore questionable whether OGY-type methods work in the noisy case because stable and unstable manifolds can hardly be identified.

Which of the periodic orbits of the noiseless system are observed when noise is present depends on the sizes of their basins and the noise level. For example, for $\delta = 0.09$ we observe hopping between fixed points of the primary family P_1, but no period 3 orbits could be identified, see Figure 8.2. When the noise level is reduced, period 3 orbits occur in addition to the primary fixed points. In Figure 8.3 we show the probability density $p(v, \theta)$ in the rectangle $[-7\pi, 7\pi] \times [0, 2\pi]$ for (a) $\delta = 0.02$ and (b) $\delta = 0.09$. Covering the rectangle with an 80×80 grid the density

was numerically generated by iterating 1,000 initial conditions for 1,000 iterations and counting the visits to each grid cell. In Figure 8.3a we clearly see small peaks corresponding to period 3 orbits that surround the large peaks corresponding to the primary fixed points. In Figure 8.3b the small peaks have disappeared and the large peaks are broadened, suggesting that the attraction basins of the period 3 orbits are now absorbed in the basin boundaries of the primary fixed points.

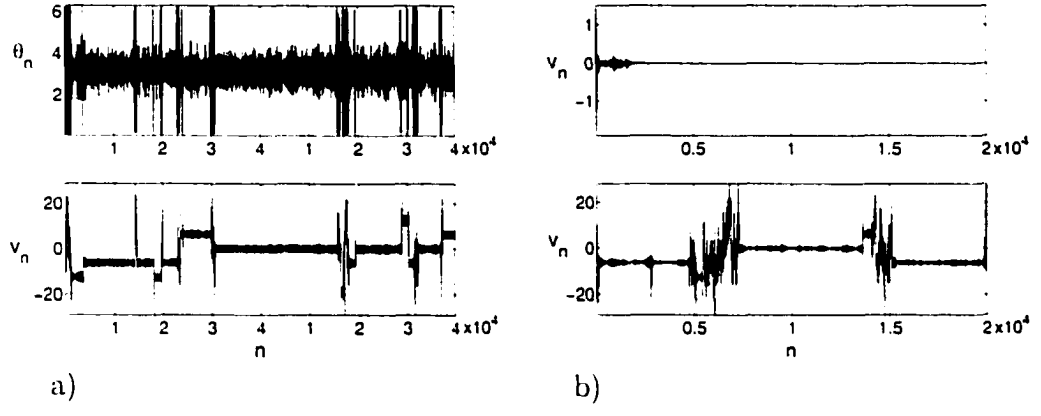


Figure 8.2: a) Dynamics of the noisy kicked rotor for $f_0 = 3.5$, $c = 0.02$ and $\delta = 0.09$. b) The effect of a noisy forcing term $f_n = f_0 + u_n$, with u_n chosen randomly from $\mathcal{U} = \{0, 0.2, -0.2\}$, on the dynamics with $\delta = 0$ (upper part) and $\delta = 0.02$ (lower part).

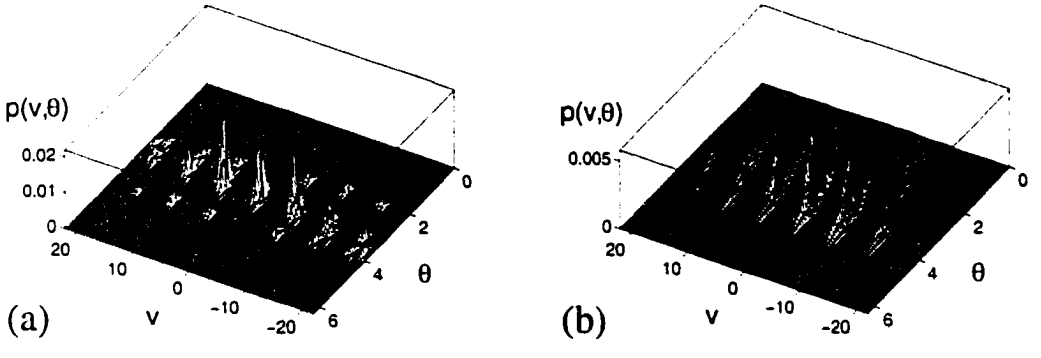


Figure 8.3: Probability density $p(v, \theta)$ in the rectangle $[-7\pi, 7\pi] \times [0, 2\pi]$ for (a) $\delta = 0.02$, and (b) $\delta = 0.09$.

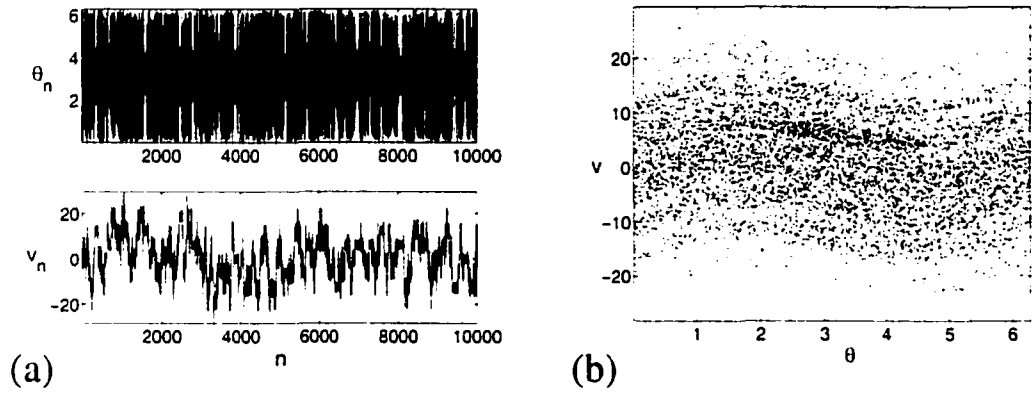


Figure 8.4: a) Dynamics of the noisy kicked rotor for $\delta = 0.3$. b) Corresponding phase plane representation of the dynamics.

For increasing noise level δ the jumps become more and more frequent up to a point where the system does not remain in a stable state for more than a few iterations. Kraut *et al.* refer to this type of behavior as predominance of diffusion due to noise, the system behaves now truly stochastically. The transition from hopping to diffusion is marked by a sharp increase of the exponent of the autocorrelation function of a noisy trajectory which occurs in a range about $\delta = 0.1$ [KFG99]. In the diffusion dominated regime the fine structure of the basins is blurred by noise, but some of this structure is still present preventing the motion from being a pure (δ-correlated) random process. A typical time series in the diffusive regime is shown in Figure 8.4a for $\delta = 0.3$. Figure 8.4b shows the corresponding phase plot.

In the following we will use the notation $\mathbf{x}_n = (\theta_n, v_n) \in X = T^1 \times \mathbb{R}$ to denote the state of the rotor at the n-th iteration. We will attempt to control the system (8.12) through small discrete perturbations u_n of f_0.

$$f_n = f_0 + u_n. \tag{8.13}$$

If u_n is a random perturbation applied to the forcing at time n, then the perturbation (8.13) has similar effect as the external noise δ. Consider Figure 8.2b, which

shows 20.000 iterations for $\delta = 0$ (upper part) and b) $\delta = 0.02$ (lower part), and f_0 replaced by (8.13). At each iteration u_n was chosen randomly from the discrete set $\mathcal{U} = \{0, 0.2, -0.2\}$. Comparing the lower part of Figure 8.2b and Figure 8.2a shows that the dynamics with small δ and intrinsic noise (8.13) lead to similar dynamics as obtained for much larger δ. This was confirmed more rigorously in [KFG99] where it was shown that both types of noise lead to similar behaviour in the system.

The previous observation motivates to apply perturbations of the form (8.13) to control the noisy rotor. As discussed in the next section, instead of applying the perturbations randomly we will determine them from a suitable approximated control policy which will be established through reinforcement learning.

8.4 The Control Algorithm

In this section we describe the control algorithm used to stabilize the noisy rotor at a prescribed state. The algorithm is explained in detail in Chapter 6 where it was used to control chaotic systems. Here we demonstrate that the same method can be used to control the non-chaotic, but multistable and noisy rotor. As mentioned above we attempt to control the noisy rotor through small discrete state-dependent perturbations of the applied forcing f_0. The dynamics of the controlled system can then be written in the form

$$\begin{aligned} \theta_{n+1} &= \theta_n + v_n + \delta_\theta \pmod{2\pi} \\ v_{n+1} &= (1-c)v_n + (f_0 + u_n)\sin(\theta_{n+1}) + \delta_v, \end{aligned} \tag{8.14}$$

where $u_n = u(\mathbf{x}_n)$ represents the state dependent control perturbation applied to the external forcing f_0 at the nth iteration step. To establish control, u_n is chosen from a discrete set $\mathcal{U}$ of allowed controls according to a certain, possibly ϵ-greedy, control policy $\pi_\epsilon(\mathbf{x}, u)$ which associates with every state $\mathbf{x}$ a control u. The main task is to find a control policy π such that a prescribed control goal is achieved

in an optimal way. We use reinforcement learning to establish such an optimal control policy as described in Chapter 6. The necessary discrete representation $\mathcal{W}$ of the state space $\mathcal{X}$ in terms of a finite set of reference vectors $\mathbf{w} \in \mathcal{W}$ is determined by means of a Neural-Gas vector quantization [MBS93] with $N = 100$ reference vectors of a set of 10,000 data points $\mathbf{x}$ obtained from simulations with $\delta = 0.3$ (see Figure 8.4b). The resulting centers are shown in Figure 8.5 together with their Voronoi cells.

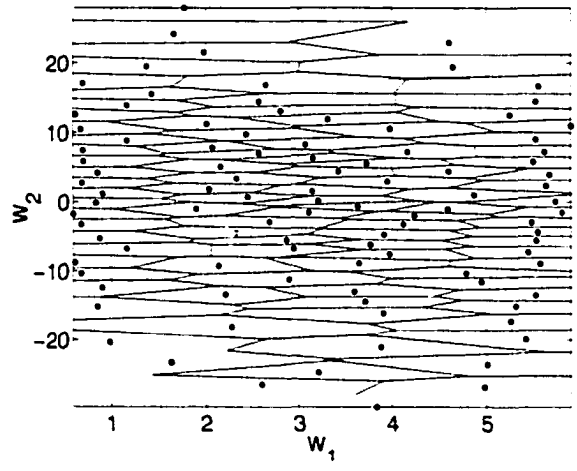


Figure 8.5: The set $\mathcal{W}$ of 100 reference vectors $\mathbf{w}$ and their corresponding Voronoi cells obtained through a Neural-Gas vector quantization of 50,000 data points of the uncontrolled dynamics with $\delta = 0.3$.

To every reduced state $\mathbf{w}$ we associate an allowed set of controls $\mathcal{U}(\mathbf{w})$. To each possible pair of reduced state $\mathbf{w}$ and allowed control signal $u \in \mathcal{U}(\mathbf{w})$ we associate a state-action value $Q(\mathbf{w}, u)$ representing the value of performing control u when the system is in state $\mathbf{x}$, such that $\mathbf{w} = \mathbf{w}(\mathbf{x})$. Whenever the system is in a state $\mathbf{x}$, its corresponding reference vector $\mathbf{w}(\mathbf{x})$ is identified and a control $u(\mathbf{x})$ is chosen from the set $\mathcal{U}(\mathbf{w}(\mathbf{x}))$ according to the policy $\pi(\mathbf{x}, u)$ now defined through the values $Q(\mathbf{w}(\mathbf{x}), u)$. As before, the optimal state-action value function $Q^*(\mathbf{w}, u)$ can be approximated through Q-learning.

8.5 The Noisy Controlled Rotor

In this section we present results obtained by applying the method discussed in the previous section to stabilize the rotor at the fixed points $\mathbf{x}_g = (\theta_g, g)$, with $g = 0,\ 2\pi$, and 4π and $\theta_g \approx \pi$. Unless otherwise stated, the control set U was restricted to $\mathcal{U} = \{0,\ u_{max},\ -u_{max}\}$ with $u_{max} = 0.2$ and the noise level was set to $\delta = 0.09$. As stated in [KFG99], this value of δ marks the transition from a deterministic to a stochastic system. In previous works [PG95, FG97] control could be established only up to very low noise levels ($\delta = 0.01$ in [FG97]). The parameters of the Q-learning update rule were set to the constant values $\beta = 0.5$ and $\gamma = 0.9$ but their particular choice does not influence results much. To measure the quality of an approximated policy defined through the values Q we introduce the quantity λ_Q which measures the average number of iterations per episode, where an episode is an iteration of the system starting at a random initial condition until the criterion for termination of an episode is met. λ_u denotes the same quantity for the uncontrolled system. Unless otherwise noted, we terminate an episode if the condition $\|\mathbf{x}_g - \mathbf{x}_n\| < 1$ is met for 200 consecutive iterations.

8.5.1 On-line Control

On-line control refers to learning in one episode. During system dynamics, starting from a random initial condition with all Q values set to zero, control perturbations, chosen greedy from the current set of values Q, are applied to the forcing function and the control algorithm updates the Q-function from immediate rewards r_n, where $r_n = -0.5$ if $\|\mathbf{x}_g - \mathbf{x}_n\| > 1$ and $r_n = 1$ otherwise. Eventually the controller will find a successful control strategy and keep the system stabilized in the desired state. Figure 8.6 shows on-line control of the noisy rotor with $\delta = 0.09$. Initially the control goal was to stabilize the system in the state $\mathbf{x}_0$. After 30.000 (60.000) iterations, Q was reset to zero and the control goal changed

to the stabilization of $\mathbf{x}_{2\pi}$ ($\mathbf{x}_{4\pi}$). We see that the controller is able to stabilize the rotor at the desired location after only a small number of iterations in all three cases. In a similar fashion we were able to stabilize all fixed points $(\theta_g, \pm m2\pi)$ for m up to 4.

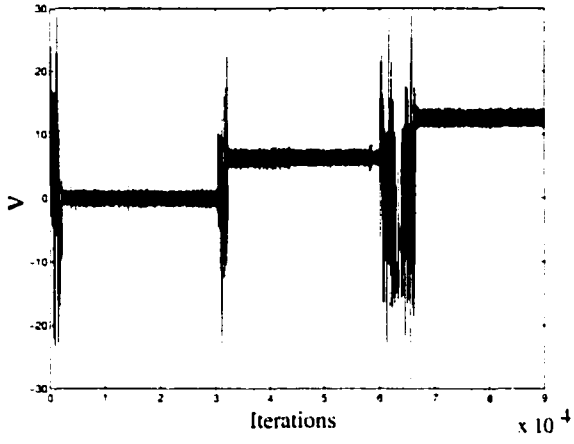


Figure 8.6: On-line control of the rotor dynamics with $\delta = 0.09$ and $\mathcal{U} = \{0.\ 0.2.\ -0.2\}$. Initially the control goal is $\mathbf{x}_0$. After 30,000 iterations the control goal is switched to $\mathbf{x}_{2\pi}$ and after 60,000 iterations to $\mathbf{x}_{4\pi}$.

In Table 8.1 we summarize the performance of the approximated policies for stabilizing the different goals $\mathbf{x}_g$. λ was averaged over 2,000 terminating episodes, where an episode was counted as terminating if the condition $\|\mathbf{x}_g - \mathbf{x}_n\| < 2$ was met for 200 consecutive iterations.

As additional performance criterion we introduce the probability $P_T(Q)$ that a policy Q will succeed. To determine this probability, we count the number λ_{nt} of episodes which did not terminate before a total of 2,000 terminating episodes occurred. An episode was counted as unterminated if it did not terminate after 10,000 iterations. $P_T(Q)$ is then $100 \cdot 2,000/(2,000 + \lambda_{nt})$. $P_T(u)$ denotes the probability of satisfying the termination criterion without control. A lower limit λ_Q^l for the average number of iterations for control with policy Q is given by $\lambda_Q^l = P_T\lambda_Q + 10,000(1 - P_T)$. A good policy Q should satisfy $P_T(Q) >> P_T(u)$ and $\lambda_Q^l << \lambda_u$.

Goal	λ_u	$P_T(u)$	λ_u^l	λ_Q	$P_T(Q)$	λ_Q^l	λ_{Q^*}	$P_T(Q^*)$	$\lambda_{Q^*}^l$
0	524	27%	7441	590	46%	5671	398	98%	590
2π	582	22 %	7928	557	48%	5467	417	99%	512
4π	1.700	10%	9170	516	56%	4688	579	98%	767

Table 8.1: Comparison of on-line (Q) and off-line (Q^*) controlled systems with the uncontrolled system.

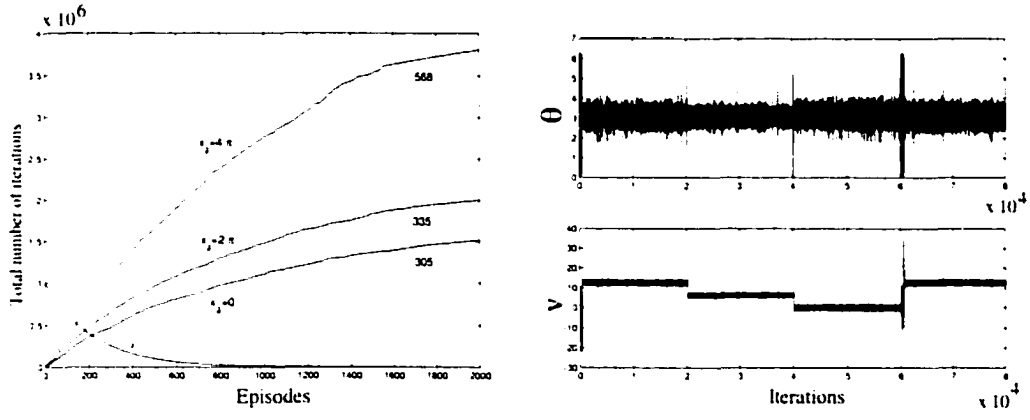
These performance measures are shown in Table 8.1 for on-line (Q) and off-line (Q^*) (see next subsection) approximated policies for the three goals. Use of the on-line approximated policy improves performance considerably over the uncontrolled system, but the policy has low termination probability. To approximate a policy with higher termination probability off-line control can be used.

8.5.2 Off-line Control

To better satisfy the requirements of convergence to an optimal policy, off-line control has to be performed. In off-line control, learning is performed ϵ-greedy in many episodes where each episode is started from a new randomly chosen initial condition.[1] For the learning process, an episode was terminated if the condition $\|\mathbf{x}_g - \mathbf{x}_n\| < 1$ was met for 5 consecutive iterations. Learning was performed over 2.000 episodes and ϵ was decreased from initially one, over 1.500 episodes, to zero. During the last 500 episodes ϵ was set to zero. Figure 8.7a shows the total number of iterations as a function of the episodes for establishing optimal control policies $Q^*_{\mathbf{x}_0}$, $Q^*_{\mathbf{x}_{2\pi}}$ and $Q^*_{\mathbf{x}_{4\pi}}$ for stabilizing the goals $\mathbf{x}_0$, $\mathbf{x}_{2\pi}$ and $\mathbf{x}_{4\pi}$, respectively. A decrease in slope indicates convergence of the corresponding policy. We see convergence in all three cases. The slope of the curves during the last episodes is an estimate for the quality λ^* of the corresponding policy. From the slopes during

[1] As noted earlier, for a chaotic system off-line can be performed on-line by turning the controller repeatedly on and off.

the last 200 episodes we get $\lambda_0^* \approx 305$, $\lambda_{2\pi}^* \approx 335$ and $\lambda_{4\pi}^* \approx 568$. $\lambda_{2\pi}^* \approx 335$ for example means that on average, by using the off-line approximated policy $Q_{\mathbf{x}_{2\pi}}^*$ the dynamics of the controlled rotor will be stabilized at the goal $\mathbf{x}_{2\pi}$ after 335 iterations. In Figure 8.7b we show the use of these global policies for a sample control problem. During 80.000 iterations the applied control policy was switched every 20.000 iterations as described in the figure caption. We see that in all cases, control is established almost instantaneously and not lost during the interval of control.



a) b)

Figure 8.7: a) Off-line control of the rotor dynamics with $\delta = 0.09$ and $\mathcal{U} = \{0,\ 0.2,\ -0.2\}$. The curves labeled $x_g = 0.x_g = 2\pi$ and $x_g = 4\pi$ represent the total number of iterations during the learning process for control of the goal $\mathbf{x}_{x_g}$. ϵ is slowly frozen from initially one to zero during learning as shown in the graph labeled ϵ (rescaled vertical units). With increasing number of episodes and decreasing ϵ a decrease in the slope of the curve shows convergence of the control policy. During the last 500 episodes ϵ was set to zero. The limiting slope (number below the curves) is an estimate of λ for the particular policy. b) Off-line control of the rotor dynamics with $\delta = 0.09$ over 80.000 iterations. The control policy is reset every 20.000 iterations from initially $Q_{\mathbf{s}_{4\pi}}^*$ to $Q_{\mathbf{s}_{2\pi}}^*$, $Q_{\mathbf{s}_0}^*$ and back to the initial policy $Q_{\mathbf{s}_{4\pi}}^*$.

The performance criteria introduced before are shown in Table 8.1 where the subscript Q^* indicates the measures for control with the off-line approximated policy Q^*. We can see a considerable improvement in the performance of the

off-line approximated policies over the on-line approximated policies. Note that $\lambda_{Q^*} = 579$ is larger than $\lambda_Q = 516$ since we average λ only over terminating episodes. Comparing λ'_Q and λ'_{Q^*} we clearly see the improvement offered by the off-line approximated policy.

The global character of the approximated policies becomes clearly apparent by looking at the probability density of the dynamics controlled with the policies Q^*. Figure 8.8 shows the probability density, approximated as discussed above, of the dynamics controlled with the policy $Q^*_{\mathbf{x}_{4\pi}}$. We see that the probability of states not in the neighborhood of $\mathbf{x}_{4\pi}$ is negligible.

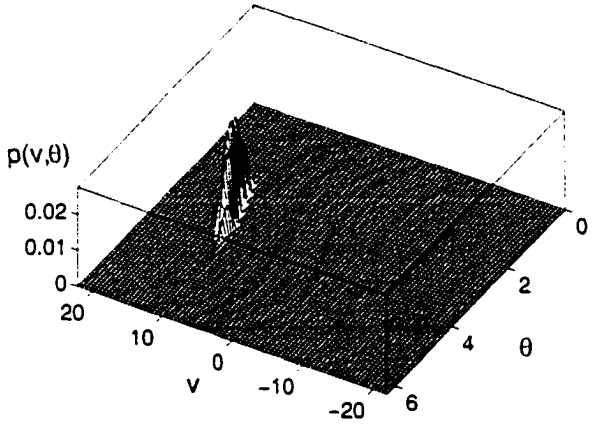


Figure 8.8: Probability density $p(v, \theta)$ in the rectangle $[-7\pi, 7\pi] \times [0, 2\pi]$.

Analysis of the Approximated Policy

The approximated global policy $Q^*_{\mathbf{x}_{4\pi}}$ can be visualized as in Figure 8.9. The dark (light) areas correspond to regions where the control $u_n = -0.2$ ($u_n = 0.2$) is associated with a maximal state-action value. The white areas correspond to regions where no control, $u_n = 0$, is applied. The effect of this policy is that points in the neighborhood of the state $\mathbf{x}_{4\pi}$ are trapped in this region while points in the neighborhood of other attracting states are more probable to leave their neighborhood. This can be seen from Figure 8.10, where 1000 random initial conditions

from a small rectangle around $\mathbf{x}_{4\pi}$ (Figure 8.10a,b) and $\mathbf{x}_0$ (Figure 8.10c,d) were iterated. In Figure 8.10a,c the states for uncontrolled dynamics are shown after 10 iterations, while Figure 8.10b,d shows the states after 10 iterations for the controlled dynamics. The parameters are as in the last subsection. Comparing Figure 8.10a and Figure 8.10b we see that application of the control signals prevents dynamics from leaving the neighborhood of $\mathbf{x}_{4\pi}$. As seen from Figure 8.9 control in this region is equivalent to a slight decrease of the forcing. In the neighborhood of other attracting states the controller learned to slightly increase the forcing which increases the probability of leaving undesired neighborhoods.

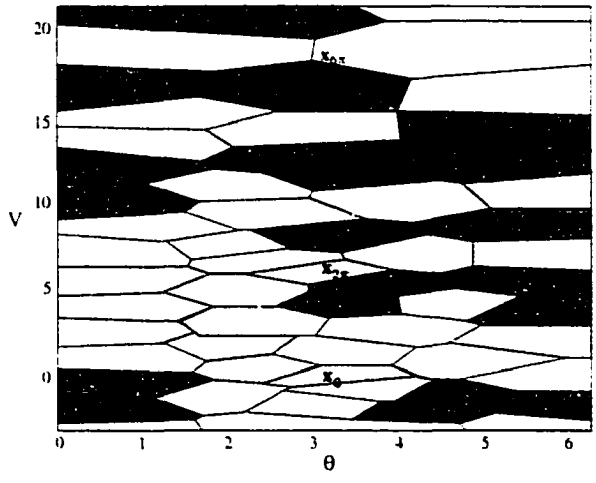


Figure 8.9: Visualization of the policy $Q^*_{\mathbf{x}_{4\pi}}$. The dark (light) grey areas correspond to regions where the control $u_n = -0.2$ ($u_n = 0.2$) is associated with a maximal state-action value. In the white areas no control will be applied.

8.5.3 Control for Higher Noise Levels

Using a Policy Learned for Low Noise

We tested the control performance of the policy $Q^*_{4\pi}$ approximated for $\delta = 0.09$ as described in Section 8.5.2 for larger values of δ. As in the preceding subsection we measure λ by averaging the number of iterations per episode over 2.000 terminating

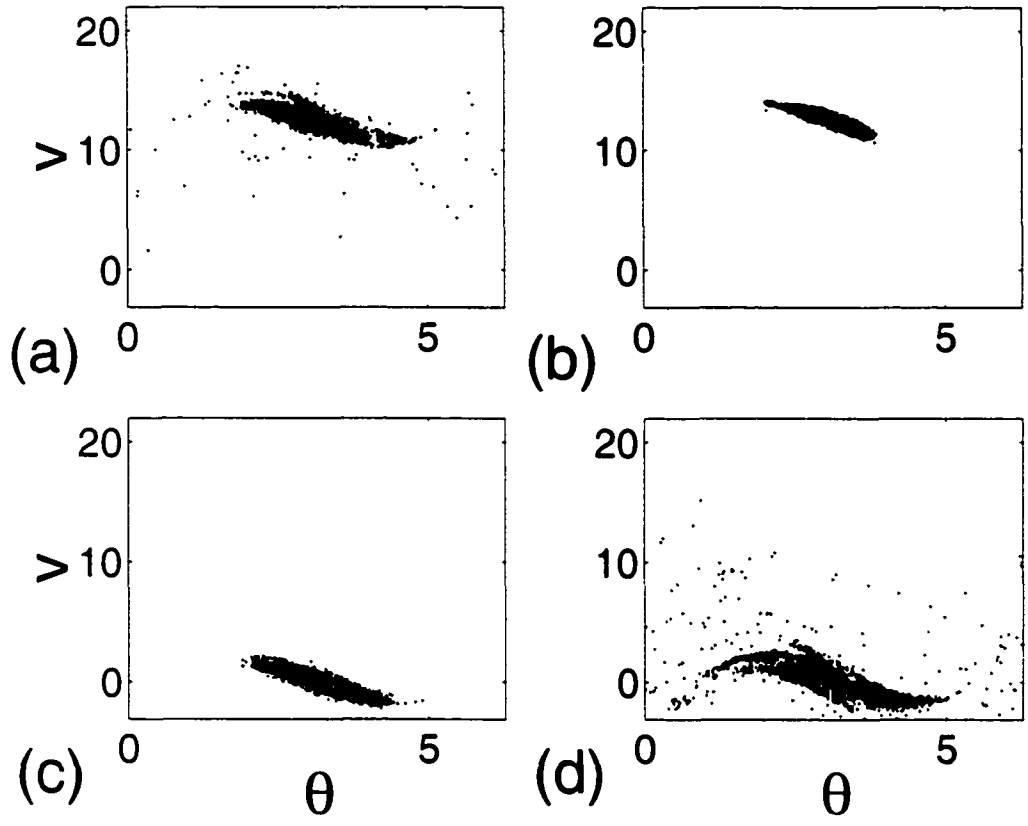
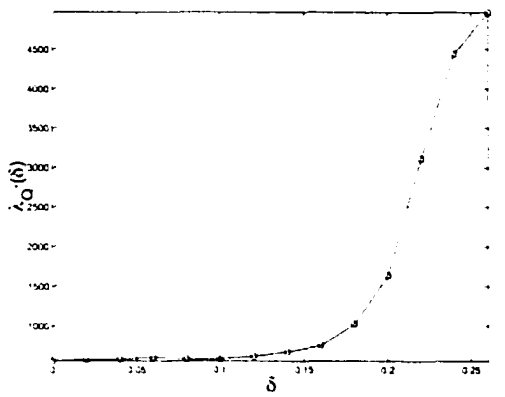


Figure 8.10: Local performance of the policy $Q^*_{\mathbf{x}_{4\pi}}$. Shown are the points obtained after 10 iterations ($\delta = 0.09$) with 1000 initial conditions (θ, v) randomly chosen from a small rectangle (as shown) around (a),(b) $\mathbf{x}_{4\pi}$ and (c),(d) $\mathbf{x}_0$. (a),(c) shows uncontrolled dynamics and (b),(d) shows dynamics controlled according to the policy $Q^*_{\mathbf{x}_{4\pi}}$.

episodes, but here we terminate an episode if $\|\mathbf{x}_g - \mathbf{x}_n\| < 2$ for 200 consecutive episodes.

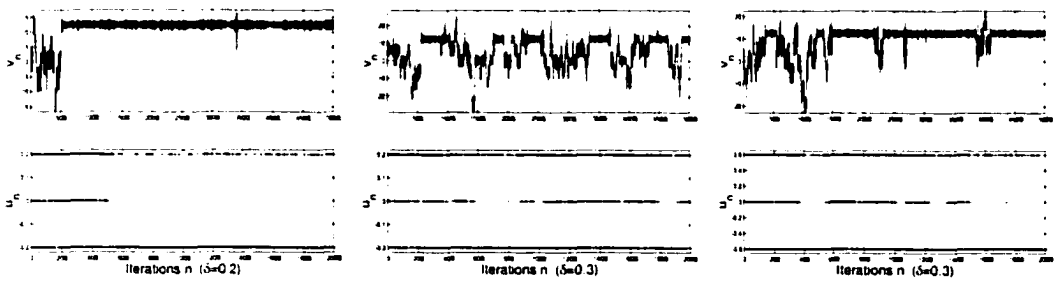
Figure 8.11a shows $\lambda_{Q^*}(\delta)$ and Figure 8.11b $P_T(\delta)$ for a range of noise levels, with λ and P_T defined as before. The policy $Q^*_{4\pi}$ approximated at $\delta = 0.09$ could successfully stabilize $\mathbf{x}_{4\pi}$ over a wide range of noise levels up to $\delta \approx 0.2$, at which point P_T begins to drop sharply below 100%. This can be confirmed from Figure 8.12 where $\|\mathbf{x}_n\|$ for the controlled system at a) $\delta = 0.2$ and b) at $\delta = 0.3$ is shown. For $\delta = 0.2$ control with $u_{max} = 0.2$ is achieved, while for $\delta = 0.3$ the noise often kicks the system out of the desired region. However, allowing a larger maximal

perturbation u_{max} for the same policy, control can be established for $\delta = 0.3$ as can be seen from Figure 8.12c, where $u_{max} = 0.6$.



a) b)

Figure 8.11: Using the policy $Q^*_{4\pi}$ learned for low noise to control the rotor for higher noise levels. Shown is the angular velocity v_n (upper graph) and the applied control signal u_n (lower graph). a) $\lambda(\delta)$, b) $P_T(\delta)$. At $\delta \approx 0.2$ a sharp drop in P_T indicates that control with the policy approximated for low noise levels is unable to stabilize $\mathbf{x}_{4\pi}$.



a) b) c)

Figure 8.12: Using the policy $Q^*_{4\pi}$ learned for low noise to control the rotor for higher noise levels. a) $\delta = 0.2, u_{max} = 0.2$, b) $\delta = 0.3, u_{max} = 0.2$, c) $\delta = 0.3, u_{max} = 0.6$

Learning Control in the Presence of Large Noise

Next we investigated if a control policy stabilizing $\mathbf{x}_{4\pi}$ can be approximated in the presence of large noise. To this end we determined the smallest value $u_{max}(\delta)$ of u_{max} for which, for a given δ, $\mathbf{x}_{4\pi}$ can be stabilized through on-line control. Figure 8.13 shows the ratio $u_{max}(\delta)/\delta$ as function of δ. This ratio approaches a limiting value of approximately 1.8 and can be used to choose the control parameter for different noise levels. For small δ the ratio is getting larger, since comparably larger perturbations are needed to enter the desired state space region due to longer transients. In Figure 8.14 we show the on-line control of the rotor for different noise levels and different sizes of the perturbations. In Figure 8.14a it is $\delta = 0.2$ and $u_{max} = 0.35$, in Figure 8.14b $\delta = 0.3$ and $u_{max} = 0.6$, and in Figure 8.14c it is $\delta = 0.4$ and $u_{max} = 0.85$. Thus it is possible to stabilize the system under the influence of large noise although much larger system perturbations are needed in this case.

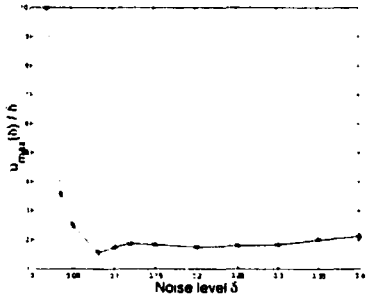
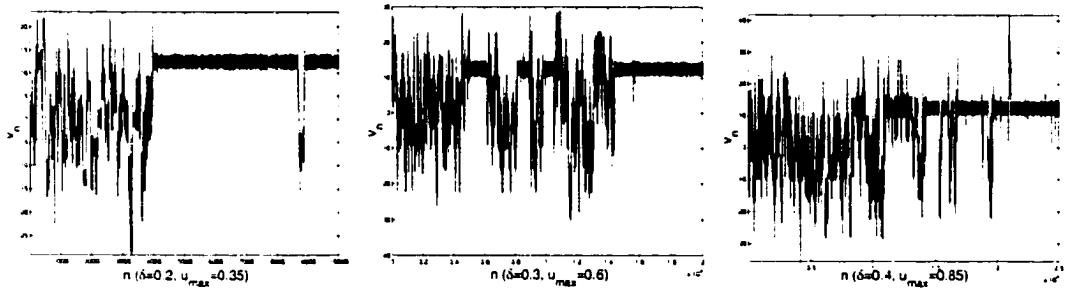


Figure 8.13: Learning Control in the Presence of Large Noise. The graph shows $u_{max}(\delta)/\delta$, where $u_{max}(\delta)$ denotes the minimum perturbation which achieves on-line control for a given δ.

8.5.4 Control of More General States

We briefly mention the control of more general goals: the stabilization of fixed points of period three and a destabilization of attracting states. To describe



a) b) c)

Figure 8.14: On-line control of rotor for different noise levels and with different perturbations. a) $\delta = 0.2$, $u_{max} = 0.35$, b) $\delta = 0.3$, $u_{max} = 0.6$, and c) $\delta = 0.4$, $u_{max} = 0.85$.

a suitable measure for detection if the goal was achieved we introduce the quantity

$$d_p(\mathbf{x}_n) = \frac{\|\mathbf{x}_n - \mathbf{x}_{n-p}\|}{\exp\left(\sum_{i=1}^{p-1} \ln \|\mathbf{x}_n - \mathbf{x}_{n-i}\|\right)}, \quad p > 1,$$

which is minimal only for fixed points of period p (see also next chapter). To stabilize a fixed point of period three we applied the reward $r_{n+1} = 1$ if $d_p(\mathbf{x}_n) < 0.02$ and $r_{n+1} = -0.5$ otherwise. This choice of reward rewards a stabilization of any period three fixed point. Since the rotor has several of these, if a particular period three orbit is desired, we must give positive rewards only if the desired orbit was stabilized. We successfully used this approach to stabilize the period three fixed point in the neighborhood of $\mathbf{x}_{2\pi}$ using on-line control but needed a set of reference vectors of size $|W| = 200$. Figure 8.15a shows the probability density of the dynamics controlled with respect to this on-line approximated policy. The density is concentrated at the desired state.

The other control goal we investigated is related to the problem of maintenance of chaos [IMDS95]. The goal here is to prevent the system from a transition of the chaotic state into a stable attracting state. For the rotor a "maintenance" of the chaotic state means that the system should be kept from visiting the neighborhoods of the stable period one or period three fixed points. This could certainly be

done by adding large noise to the system. Instead here we will demonstrate that the proposed algorithm is able to "maintain" chaos in the rotor through small perturbations to the external forcing by a suitable formulation of the reward function. We chose the reward function $r_{n+1} = 1$ if $d_1 > 2.5$ and $d_3 > 5$ and $r_{n+1} = -0.5$ otherwise. This punishes all actions leading to states close to period one or three fixed points. We approximated a maintenance control policy Q^*_{mtn} off-line. The noise was set to $\delta = 0.02$, the control set was $U = \{0, 0.2, -0.2\}$, and we used the set W of size $|W| = 100$ mentioned above. Figure 8.15b shows the negative probability density $p(v, \theta)$ in the rectangle $[-5\pi, 5\pi] \times [0, 2\pi]$. We see that the regions in the neighborhoods of the attracting states have negligible probability of being visited.

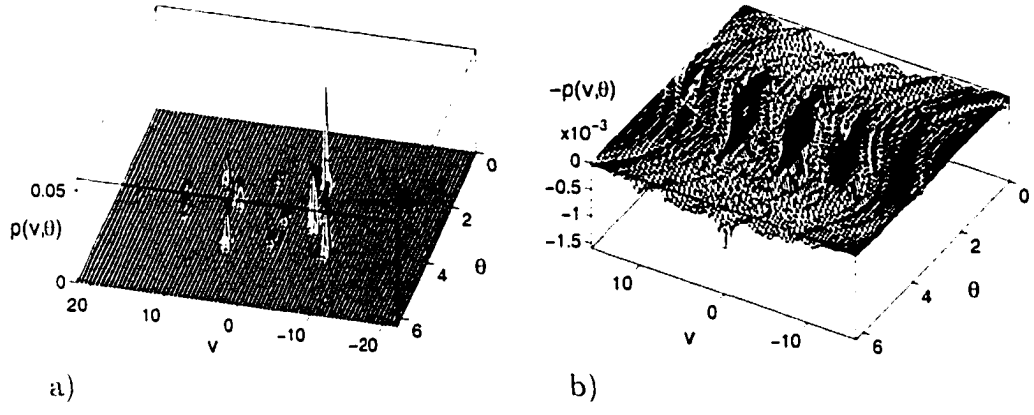


Figure 8.15: (a) Probability density $p(v, \theta)$ in the rectangle $[-7\pi, 7\pi] \times [0, 2\pi]$ for stabilization of period three fixed point. It is $\delta = 0.02$ and $U = \{0, 0.2, -0.2\}$. (b) Negative probability density $-p(v, \theta)$ in the rectangle $[-5\pi, 5\pi] \times [0, 2\pi]$. It is $\delta = 0.02$ and $U = \{0, 0.2, -0.2\}$ with perturbations to f_0 chosen from U according to the policy Q^*_{mtn}. The black areas belong to densities with $p < 2 \cdot 10^{-5}$.

8.6 Conclusions

In this chapter we have demonstrated the control of a "simple" complex system, the kicked mechanical rotor, under the influence of considerable noise, through

the method introduced in Chapter 6. Although the method was introduced for control of chaotic systems it can also be used to control the multistable non-chaotic rotor discussed in this chapter. It is interesting to note that control could be established even in regions in which the system's dynamics are characterized as stochastic. Furthermore the algorithm was able to establish control under the influence of large noise levels up to $\delta = 0.4$. Alternative control methods which require the estimation of dynamical quantities like the Jacobian will certainly be hard to apply under the influence of such large noise levels.

The presented approach has not yet been applied to systems with riddled basins or more complexly structured phase spaces, which often occur in applications, and it has to be investigated how to deal with systems evolving in an infinite phase space. A combination with a dimension reduction technique such as a Karhunen-Loeve decomposition might allow to reduce the system to a finite dimensional system in certain situations. Furthermore, the control of more complicated dynamics such as quasiperiodic orbits (invariant circles) or chaotic saddles still has to be investigated. Our results on "maintenance of chaos" provide a first step in this direction.

Chapter 9

A GROWING NEURAL-GAS CHAOS CONTROLLER

> ... advantages are: The possibility to use problem dependent error measures to determine where units are inserted (insertion where necessary).
>
> (B. Fritzke, 1995, p. 72)

In this chapter we combine a modification of Fritzke's Growing Neural-Gas algorithm described in Chapter 5 with the chaos control algorithm proposed in Chapter 6. The Growing Neural-Gas algorithm is modified in order to obtain codebooks which place more clusters in neighborhoods of fixed points. Combining the codebook generation phase with the chaos control algorithm we will obtain an algorithm which can achieve the desired control goal with minimal computational effort. By associating larger control sets with reference vectors in neighborhoods of UPOs we obtain control policies which allow stabilization with minimal perturbations. Adding a final phase in which the location of the desired fixed point and a local continuous control policy is approximated with high accuracy, the fixed point can be stabilized with practically vanishing perturbations in noise free environments.

9.1 Fixed-Point Sensitive Growing Neural-Gas

The chaos control algorithm introduced in Chapter 6 requires a given set of reference vectors $\mathbf{w} \in \mathcal{W}$ and a given set of controls $\mathcal{U}(\mathbf{w})$ associated with each reference vector. The computational complexity of the algorithm is approximately

of order $|\mathcal{W}|^{d_w}|\mathcal{U}|$. To reduce computational requirements it is important to choose $|\mathcal{W}|$ and $|\mathcal{U}|$ as small as possible. Until know we had generally computed the set $\mathcal{W}$ in advance from recorded data and chosen $|\mathcal{W}|$ sufficiently large. However, to control a fixed point of lower period, it is not necessary to have a fine representation of $\mathcal{X}$ globally. Instead, a fine representation of $\mathcal{X}$ only locally, close to the desired fixed point should suffice. To this end we introduce in this section a modification of Fritzke's Growing Neural-Gas algorithm which places clusters close to fixed points of desired period with increased probability.

Fritzke's Growing Neural-Gas algorithm is described in Section 5.6 and summarized in Figure 5.6. The algorithm allows to insert new units in desired regions through the function `Cell.Add` of Figure 5.6. We modify this function slightly such that new units are placed into the neighborhoods of fixed points of desired period according to a given probability as described in the following section.

9.1.1 The FSGNG Algorithm

As described in 2.6, points of close returns are indicators for fixed points. Thus, assuming a discrete system dynamics

$$\mathbf{x}_n = \mathbf{f}(\mathbf{x}_{n-1}).$$

with unknown map $\mathbf{f}$ fixed points of period one occur at zeros of $\|\mathbf{x}_n - \mathbf{x}_{n-1}\|$ (see Figure 9.1a). In principle this can be extended to fixed points of higher periods, e.g. fixed points of period p occur at minima of $\|\mathbf{x}_n - \mathbf{x}_{n-p}\|$, however a problem arises here, since fixed points of period less than p also often minimize this quantity (see Figure 9.1b). Instead, to detect neighborhoods of fixed points[1] of period p we

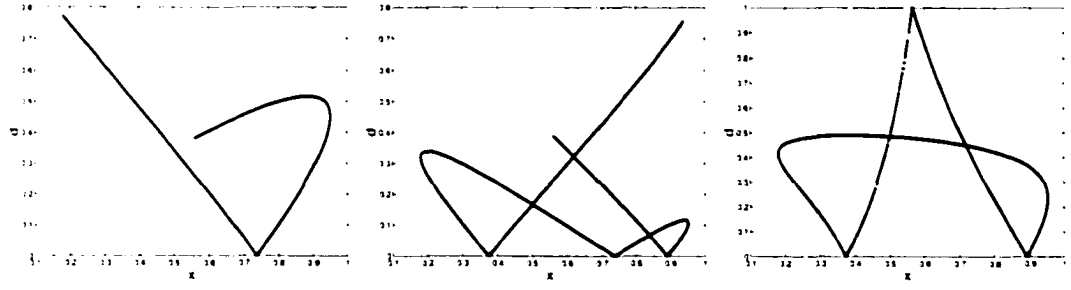
[1]Note that we are not attempting to approximate the exact location of fixed points here, but merely to find their approximate neighborhoods.

will use the quantity,

$$d_p(\mathbf{x}_n) = \frac{\|\mathbf{x}_n - \mathbf{x}_{n-p}\|}{\exp\left(\sum_{i=1}^{p-1} \ln \|\mathbf{x}_n - \mathbf{x}_{n-i}\|\right)},$$

which is minimal only for fixed points of period p (see Figure 9.1c).

To obtain codebooks which place clusters with higher probability into neighborhoods of fixed points we modify the function `Cell.Add` of the Growing Neural-Gas algorithm (see Figure 5.6) such that with a certain probability new units are placed into regions where a $\mathbf{x}_n$ with $d_p(\mathbf{x}_n) < \epsilon$ occurred. This fixed point sensitive Growing Neural-Gas algorithm (FSGNG) is summarized in Figure 9.2. Instead of choosing $\mathbf{x}$ randomly from $\mathcal{X}$ at each iteration, we present the on-line version where $\mathbf{x}$ is obtained from system dynamics. If an $\mathbf{x}_n$ is detected which satisfies $d_p(\mathbf{x}_n) < \epsilon$, we record this state and add it to a set of fixed point neighbors $\mathcal{F}$. If a new unit is added, we place this unit with a certain probability given by p_f into the neighborhood of a random state $\mathbf{x} \in \mathcal{F}$.



a) b) c)

Figure 9.1: Determining neighborhoods of fixed points of period p of the logistic map $x_n = 3.8x_{n-1}(1 - x_{n-1})$. a) $d_1(x_n) = |x_n - x_{n-1}|$, b) $|x_n - x_{n-2}|$, c) $d_2(x_n)$.

```
Initialize W with |W| = 2, F with |F| = 0
Initialize c(w_1, w_2) = e(w_1) = e(w_2) = 0, i = 0
Repeat
    x_n ← x_{n-1}.  Δi = 1
    w_1 = arg min_{w∈W} ||x_n − w||,   w_2 = arg min_{w∈W\{w_1}} ||x_n − w||
    If d_p(x_n) < ε then F = F ∪ {x_n}
    Cell.Update.Connectivity(w_1, T(w_1))
    Cell.Update.Weights(w_1, T(w_1))
    Cell.Remove(W)
    If mod (i, τ) = 0 then Cell.Add.FSGNG(W)
    e(w) = βe(w)  ∀w ∈ W
Until Stopping criterion satisfied
```

```
Function Cell.Add.FSGNG(W)
    Choose z randomly from [0, 1]
    If z > p_f then  w_e1 = arg max_{w∈W} e(w)
    else set w_e1 to a random x ∈ F
    w_e2 = arg max_{w∈T(w_e1)} e(w)
    w_new = ½(w_e1 + w_e2),  W = W ∪ {w_new}
    c(w_e1, w_e2) = 0,  c(w_e1, w_new) = c(w_e2, w_new) = 1
    a(w_e1, w_new) = a(w_e2, w_new) = 0
    e(w_new) = e(w_e1) = αe(w_e1),  e(w_e2) = αe(w_e2)
```

Figure 9.2: Pseudo-code of the fixed point sensitive Growing Neural-Gas algorithm. Only the parts are shown which differ from the Growing Neural-Gas algorithm of Figure 5.6.

We applied the FSGNG algorithm to clustering of the logistic and the Hénon map. In Figure 9.3 we have shown several stages of the FSGNG clustering of the Hénon map with $p = 1$, $p_f = 0.5$ and $\epsilon < 0.1$. The algorithm can be conveniently stopped if an unproportional large number of reference vectors is appearing in the region of the neighborhood. Initially clusters placed in the neighborhood of the fixed points will quickly be distributed over the complete set $\mathcal{X}$ (see Figure

9.3a,b). Only if the set $\mathcal{X}$ is approximated well, clusters placed in the neighborhood of fixed points will remain there. At this point an increase in the histogram in regions corresponding to this neighborhood will occur and the growing phase of the algorithm can be stopped (see Figure 9.3c,d). The final codebook $\mathcal{W}_{Hen}$ with $|\mathcal{W}_{Hen}| = 70$ is shown in Figure 9.3c. In similar fashion a codebook with size $|\mathcal{W}_{log}| = 45$ was obtained for the logistic map.

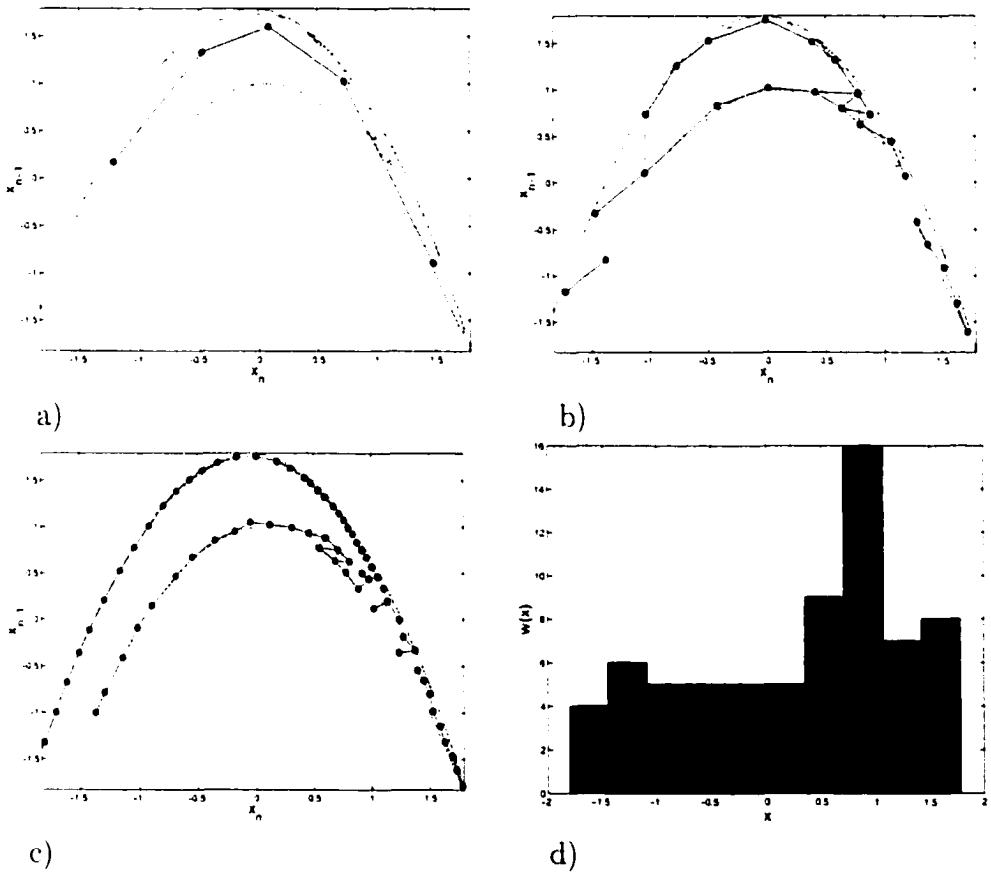


Figure 9.3: FSGNG vector quantization of the Hénon attractor. We set $p = 1, p_f = 0.5$ and $\epsilon = 0.1$. a)-c) Different stages of the clustering. The final codebook is shown in c). d) Histogram of the final codebook.

On-line Chaos Control with the FSGNG Codebook

The obtained codebooks should be suited as reduced space representation for the control algorithm discussed in Chapter 6. In the applications of Chapter 6 we

had the control set $\mathcal{U}$ fixed with $|\mathcal{U}| = 3$. As a result we obtained non-vanishing perturbations. In situations where no considerable noise is present in the environment it is possible to stabilize the desired state with practically vanishing perturbations. To this end we must generally have the ability to choose perturbations from a continuous set $\mathcal{U}$. The algorithm as described in Chapter 6 is not able to handle continuous control sets.

An approximate solution to overcome this problem is to increase the size of the control set $\mathcal{U}$ while keeping the maximal size of the perturbation unchanged. As an example consider the on-line control of the period one fixed point of the logistic map with the control set

$$\mathcal{U}_{log}(\mathbf{w}) = \{0, 0.02, -0.02, 0.04, -0.04, 0.06, -0.06, 0.08, -0.08, 0.1, -0.1\}$$

for all $\mathbf{w}$ and a uniformly distributed codebook of size $|\mathcal{W}_{log}| = 50$. (in Section 6.3 we used $\mathcal{U}_{log} = \{0, 0.1, -0.1\}$ and $|\mathcal{W}_{log}| = 100$). To enforce the algorithm to choose small controls we use the following reward function for control of a fixed point:

$$r_{n+1} = \begin{cases} 1 & \text{if } \mathbf{w}_{n+1} = \mathbf{w}_n \\ -0.5 - |u_n| & \text{otherwise.} \end{cases}$$

The result of an on-line control of the logistic map is shown if Figure 9.4a. As expected, the algorithm takes longer to stabilize the fixed point as in Section 6.3 since the complexity of the learning problem is now greater due to the increased size of $\mathcal{U}_{log}$. However, as we can see from the lower part of Figure 9.4a, which shows u_n, the established policy does not choose a small perturbation. The reason is that small perturbations will only be adequate in a small neighborhood of the fixed point and a uniform partitioning does not guarantee that we have reference vectors close to the fixed point, unless we partition very fine. Instead consider the on-line control shown in Figure 9.4b, where we use the codebook approximated by

the FSGNG algorithm. The algorithm now chooses smaller perturbations but still takes a considerable amount of time until control is established.

To reduce the complexity of the learning problem we allow the size of the set $\mathcal{U}(\mathbf{w})$ to vary with $\mathbf{w}$.

Let $n(\mathbf{w}) = (|\mathcal{U}(\mathbf{w})| - 1)/2$ and consider the finite control set

$$\mathcal{U}_{u_{max}}^{n(\mathbf{w})}(\mathbf{w}) = \left\{0, \pm\frac{u_{max}}{n}, \pm\frac{2u_{max}}{n}, \cdots, \pm\frac{(n-1)u_{max}}{n}\right\}.$$

In this notation, the control set

$$\mathcal{U}(\mathbf{w}) = \{0, 0.02, -0.02, 0.04, -0.04, 0.06, -0.06, 0.08, -0.08, 0.1, -0.1\}$$

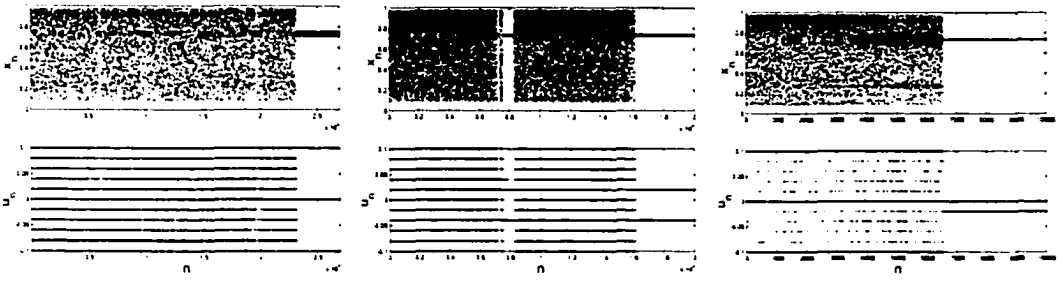
would be conveniently denoted by $\mathcal{U}_{0.1}^{5}(\mathbf{w})$. Control sets associated with different states $\mathbf{w}$ may vary in size but will all have an equal maximal control u_{max}. To reduce the complexity of the learning problem we will set $n(\mathbf{w}) > 1$ only for $\mathbf{w}$ in the neighborhood of a desired state while we set $n(\mathbf{w}) = 1$ for all other $\mathbf{w}$.

In Figure 9.4c an on-line control of the logistic map with the FSGNG codebook is shown where $n(\mathbf{w}) = 5$ for $\mathbf{w}(\mathbf{x})$ with $\mathbf{x} \in \mathcal{F}$ and $n(\mathbf{w}) = 1$ for the remaining reference vectors. The set $\mathcal{F}$ denotes the set of points close to the fixed point satisfying $d_1(x) < \epsilon$ with $\epsilon = 0.1$. We see that control is now established quickly and a minimal perturbation is chosen.

We have demonstrated in this subsection that the FSGNG algorithm produces codebooks suited for the chaos control algorithm as proposed in Chapter 6 but still computed the codebook before the control policy was approximated. In the next section we will combine these two steps.

9.2 Combining the FSGNG Algorithm and Reinforcement Learning

As seen in the last section, the FSGNG algorithm produces codebooks suited for the control of fixed points of chaotic systems. If in addition control sets are



a) b) c)

Figure 9.4: On-line control of the logistic map with a) uniform set $\mathcal{W}$ and fixed $\mathcal{U}$ with $n(\mathbf{w}) = 5\ \forall \mathbf{w}$. b) $\mathcal{W}$ from FSGNG and fixed $\mathcal{U}$ with $n(\mathbf{w}) = 5$, and c) $\mathcal{W}$ from FSGNG and $n(\mathbf{w}) = 5$ for $\mathbf{w}$ close to $\mathbf{x}_f$, and $n(\mathbf{w}) = 1$ otherwise. In all cases we had $u_{max} = 0.1$.

allowed to vary in size with larger sizes chosen for reference vectors close to fixed points, while keeping the allowed maximal perturbation fixed, then the approximated control policy allows effective stabilization of fixed points with minimal perturbations.

Since the FSGNG algorithm is an on-line algorithm, we can easily combine it with the proposed chaos control algorithm and thus eliminate the need to construct a set of reference vectors in advance. In addition, the combination of vector quantization and control can result in codebooks which are effectively smaller than codebooks obtained from uncontrolled data.

The combination is performed as summarized in Figure 9.5. First we perform a growing phase of the codebook during which also an on-line control is performed. Initially we associate with each created reference vector a control set $\mathcal{U}(\mathbf{w})$ of size $|\mathcal{U}(\mathbf{w})| = 3$ and associate state-action values $Q(\mathbf{w}, u)$ with each possible pair $(\mathbf{w}, u)$, where as before $Q = 0$ initially.

At each iteration the reference vectors are updated and possibly new vectors are created according to the FSGNG algorithm. At the same time the controller will receive rewards and update the state-action values as discussed in Chapter

6. During the growing phase actions are exclusively chosen deterministic from the associated Q-values and to all new created reference vectors we will associate a minimal control set of size $|\mathcal{U}| = 3$ ($n(\mathbf{w}) = 1$). During this first phase we will choose a small probability to place reference vectors close to fixed points ($p_f \approx 0$). After a certain time the controller will stabilize the desired state without loosing control which can easily be detected from the received rewards. Once the controller was able to stabilize the desired state for a certain number of iterations, we will allow larger control sets and enforce the controller to choose smaller perturbations as discussed in the last section. To this end we will increase the probability p_f of setting new reference vectors in regions of fixed points ($p_f = 0.5$ in the example below) and whenever a new unit is inserted close to a fixed point we will associate control sets with larger size $|\mathcal{U}| >> 3$ with these units (but as before, the maximal allowed perturbation will remain fixed for all units). In addition we will start an annealing phase by decreasing the reference vector update parameters ϵ_w and ϵ_n and increasing τ to add units less frequently. The growing phase can be terminated once the regions with fixed points are covered rather dense with reference vectors. This can be detected from histogram plots as discussed in the last section.

In Figure 9.6a we show the dynamics of an on-line controlled logistic map during the growing phase. After approximately 7,000 iterations the codebook has grown to $|\mathcal{W}| = 23$ and the desired state is stabilized for more the 200 consecutive iterations. This is used as an indicator to start the annealing phase, to increase p_f and to insert units with $|\mathcal{U}| = 11$ close to the fixed point. We see that the applied perturbations are decreasing (the variation around the fixed point is decreasing) and after 3,000 more iterations the growing phase is terminated. Result of the growing phase is a state-action value function $Q(\mathbf{w}, u)$ and a codebook $\mathcal{W}$ sufficient to stabilize the desired state with small perturbations. A histogram of the final codebook with 29 reference vectors is shown in Figure 9.6b.

1. **Growing Phase**

   (a) On-line learning with fixed $\mathcal{U}$ and $0 \leq p_f << 1$:
   Combine on-line reinforcement learning chaos controller with FSGNG vector quantization algorithm. Initially we start with control sets of minimal size: $n(\mathbf{w}) = 1\ \forall \mathbf{w}$.

   (b) Increase $n(\mathbf{w})$ and set $0 << p_f \leq 1$:
   Once the UPO of desired period is stabilized we increase the size of control sets associated to units created close to fixed points: increase $n(\mathbf{w})$ of units $\mathbf{w}(\mathbf{x})$ with $\mathbf{x} \in \mathcal{F}$ and start annealing phase (add units less frequently and decrease ϵ_w, ϵ_n).

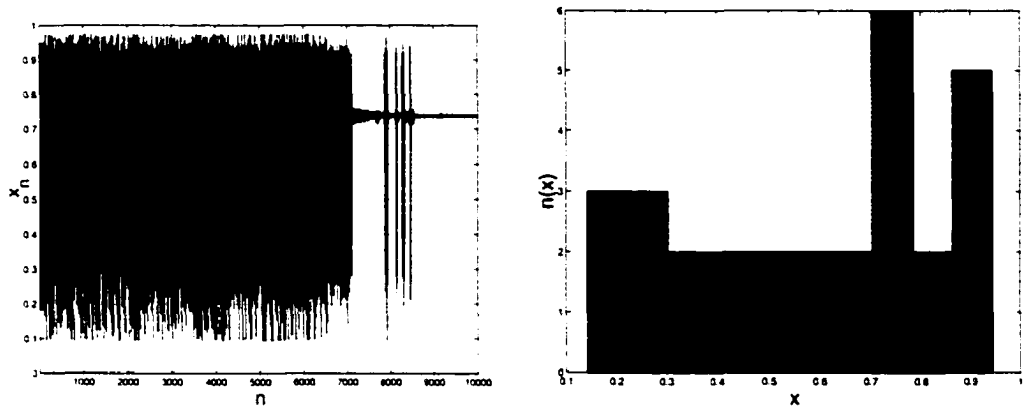
2. **Global Learning**

   We terminate the growing phase if the number of reference vectors close to the fixed point appears sufficient and start an off-line learning phase to improve the policy.

Figure 9.5: Growing chaos controller.

As discussed in Chapter 6, the policy Q obtained through on-line learning is not necessarily globally optimal. To improve the policy, episode learning or off-line control can be used. To this end we add a global learning phase after the growing phase where learning is performed ϵ-greedy over many episodes with ϵ decreasing from initially one to zero. During the off-line control phase we improve upon the policy Q obtained through the growing phase. The codebook is left unchanged during this improvement phase. Result of the global learning phase is a state-action value function Q^* which stabilizes the desired state in a minimum number of iterations from any initial state.

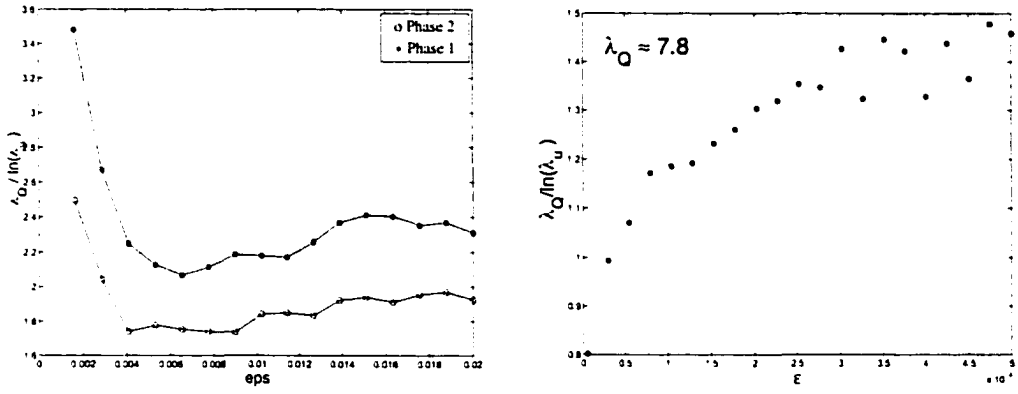
To obtain a policy Q^* for control of the fixed point x_f of the logistic map a global learning phase over 1.000 episodes is performed. To compare the performance of the policies Q and Q^* for control of the logistic map, we average λ_Q and λ_u over 1.000 episodes, terminating an episode if $|x_n - x_f| < \epsilon$, and compute



a) b)

Figure 9.6: Growing phase of the growing chaos controller for control of the fixed point of the logistic map. a) On-line controlled dynamics during the growing phase. The annealing phase is started after approximately 7.000 iterations. b) Histogram of the codebook resulting from the growing phase.

$\lambda_Q / \ln \lambda_u$. Figure 9.7a shows this quotient as a function of ϵ for the policy Q (phase 1) and the policy Q^* (phase 2). We can see the logarithmic scaling of the transient time compared to the transient of the uncontrolled system and observe that the policy Q^* performs better than the policy Q. For $\epsilon = 0.004$ we found $\lambda_Q \approx 10.1, \lambda_{Q^*} \approx 7.8$, and $\lambda_u \approx 88.5$. We see that the policy from the growing phase already offers good performance. In Figure 9.7a we observe a decrease in the performance as $\epsilon < 0.004$. This is due to the fact that we only allow discrete controls. Although we have $|\mathcal{U}| = 11$ for reference vectors close to the fixed point, we still have a finite smallest possible value $|u|_{min} = 0.02$ for the non-zero control signal. As a result the discrete control policy will not allow the system to come arbitrarily close to the fixed point. Increasing $|\mathcal{U}|$ further would bring the system closer to x_f but to come arbitrarily close we must allow continuous control signals close to the desired states. To this end we add a final phase to the algorithm in which a continuous control policy is computed. This is discussed in the next section.



a) b)

Figure 9.7: Control of the logistic map by different policies. $\lambda_Q / \ln \lambda_u$ for different policies Q averaged over many episodes as a function of ϵ. An episode was terminated if $|x_n - x_f| < \epsilon$. a) The policy Q from the growing phase (phase 1) and the policy Q^* from the episode learning phase (phase 2). c) Performance of the control rule which globally uses the policy Q^* and locally switches to a continuous control rule.

9.3 Computing a Locally Continuous Control Policy

As discussed in the last section, it is necessary to find a continuous control policy for states close to $\mathbf{x}_f$ if vanishing control perturbations are desired. We note that this is only an issue in noise free environments. Close to the desired state we can assume dynamics to be linear such that the dynamics $\mathbf{x}_{n+1} = \mathbf{f}(\mathbf{x}_n, u_n)$ are approximately described by $\mathbf{x}_{n+1} \approx \mathbf{A}\mathbf{x}_n + \mathbf{g}u_n$. It follows that in this neighborhood $u_n \approx \psi(\mathbf{x}_{n+1}, \mathbf{x}_n)$. Thus, if we can approximate ψ and the state $\mathbf{x}_f$, then a local continuous control policy is given by $u_n \approx \psi(\mathbf{x}_f, \mathbf{x}_n)$, which will allow us to bring the system dynamics onto $\mathbf{x}_f$ with high accuracy and then allow vanishing perturbations. The approximation of ψ and $\mathbf{x}_f$ is straightforward once the off-line control phase is completed.

9.3.1 Approximation of $\mathbf{x}_f$ Through So Transformation

To approximate $\mathbf{x}_f$ to high accuracy we can conveniently use the So transformation [SOS+96, SOS+97] as discussed in Subsection 2.6.1. All we need to do is to

collect data points $\mathbf{x}_n, \mathbf{x}_{n-1}, \mathbf{x}_{n-2}$ at iterations n where $\mathbf{x}_n$ is in the neighborhood of a fixed point. Since we want to approximate the fixed point of the uncontrolled system, it is important to collect data points where no perturbation was applied. This can be most conveniently performed after the off-line approximation phase by frequently switching the controller off for a few iterations whenever the desired state is stabilized. Since the approximated policy Q^* is globally optimal and restabilizes the system quickly, the data collection phase will be short. We collected 100 points to estimate x_f of the logistic map with $p_0 = 3.8$ and transformed the data with the So transformation[2] obtaining $x_f = 1 - 1/p_0$ to within seven digits of accuracy. As discussed in [SOS+97], the So transformation can be used to estimate fixed points of higher dimensional systems and of higher period.

Approximation of ψ

We approximate ψ through supervised learning by presenting input-output pairs. As inputs we use $(\mathbf{x}_{n+1}, \mathbf{x}_n)$ and as corresponding output u_n. We only approximate ψ from data satisfying $d_p(\mathbf{x}_n) < \epsilon$. This data is collected during the on-line and off-line phase. Any supervised learning scheme can be used to approximate ψ, and this approximation is not difficult since we only need a good approximation of ψ close to the fixed point. Only for convenience[3] we used a back-propagation multi-layer perceptron (BMLP) for this purpose. Although we consider the approximation of ψ as a final phase in the algorithm this approximation can be performed in parallel with the on-line and off-line control phase by sending the training data to a separate computational entity.

---

[2]Since we collect only data close to x_f it is not necessary to average over random k. We can set $k = 0$ in the So transformation.

[3]We used the BMLP implementation of Matlab's neural network toolbox.

Having approximated Q^*, ψ, and $\mathbf{x}_f$, the final controller applies the following perturbations to control a fixed point of period p:

$$u_n = \begin{cases} \arg\max_u Q^*(\mathbf{w}_n, u) & \text{if } d_p(\mathbf{x}_n) > \epsilon \\ \psi(\mathbf{x}_f, \mathbf{x}_n) & \text{otherwise.} \end{cases}$$

In Figure 9.8 we show the control of the logistic map with the above control rule. ψ was approximated with a BMLP with 10 hidden neurons and we set $\epsilon = 0.01$. x_f was approximated as mentioned in the last paragraph through the So transformation. The system was started in a random initial condition and kicked in a random state every 50 iterations. We see that control is quickly established after all perturbations. The control signals practically vanish once the fixed point is reached (they are of order 10^{-6}). Figure 9.7b shows the performance measure $\lambda_Q / \ln \lambda_u$ as explained in the last section for this control policy combining discrete and continuous perturbations. Now the transient time to reach a neighborhood of the fixed point is not increasing if we decrease the size of the neighborhood beyond $\epsilon = 0.004$. The average transient time to reach an arbitrarily small neighborhood of x_f remains at approximately 7.8 for $\epsilon \approx 10^{-6}$ for the controlled system, while the natural transient time increases dramatically.

9.4 Conclusions

In this chapter we combined several ideas to form an efficient general purpose chaos controller. The complete controller consists of 1) a growing and on-line control phase, 2) an off-line control phase, and 3) a local continuous policy approximation phase. Depending on the situation and the system to be controlled, not all three phases might be necessary or helpful. If the system is influenced by considerable noise then phase 3) will not improve performance since the noise prevents exact control of $\mathbf{x}_f$. Furthermore if considerable noise is present or the system is rather difficult to stabilize, then the controller will explore enough of the

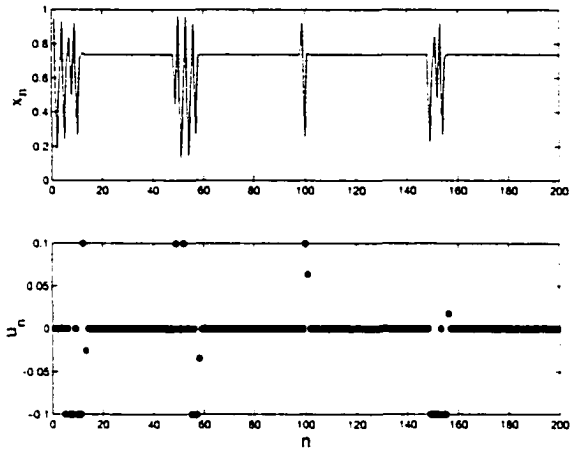


Figure 9.8: Control of the logistic map using the final control policy combining discrete controls approximated through reinforcement learning and a continuous local policy approximated through a BMLP. The system was started from a random initial condition and kicked into a random state every 50 iterations. Control is quickly reestablished and control signals practically vanish once the fixed point is reached.

space $\mathcal{W} \times \mathcal{U}$ in the on-line phase to approximate a global policy. We have seen in the last section that the on-line phase alone resulted in a policy which was close in performance to the off-line approximated policy. In summary we suggest to always start with an on-line control phase, either combined with the FSGNG algorithm or using a codebook approximated with some other vector quantization procedure. The on-line phase allows to confirm if control of the desired state is feasible with the chosen type and size of perturbations. After the on-line control phase an off-line control phase can be performed to improve the global character of the control policy and to allow locally smaller discrete perturbations. If the situation allows stabilization of the desired state through vanishing control signals, a continuous local control policy can be established in a final phase.

Chapter 10

CONCLUSIONS

mihi cura futuri
(my concern is the future)

In this dissertation we introduced a chaos control algorithm based on reinforcement learning. To satisfy the requirement of a finite Markov decision process and to reduce the complexity of the reinforcement learning problem, we applied a vector quantization method to decompose the state space into a finite space, represented by the reference vectors and their Voronoi cells. The control strategy is based on approximating state-action value functions in the reduced space for specific tasks. We considered in particular the task of stabilization of unstable fixed points or unstable periodic orbits embedded in a chaotic attractor and the targeting of more general states by small system perturbations. The algorithm can be used for parametric or impulsive control and the extension to multiparameter control, or combinations of parameter and impulsive control through the use of separate state-action value functions for respective perturbed parameters is straight forward.

The method does not require any analytical description of the underlying dynamical system nor is the exact location of the fixed points required in advance. All information required to find an optimal policy is obtained through interaction with the environment. This does, however, not mean that we suggest not to use information about the system when this is available. Reinforcement learning is

flexible and allows to include additional constraints in the algorithm, as we did when selecting specific fixed points or period 3 orbits in Chapter 8.

Generally speaking, the algorithm initially explores the state-action space to find the neighborhood of a state which shows the desired characteristics. Once this neighborhood has been identified a control policy is learned which traps the dynamics in the neighborhood of the goal state while globally directing the dynamics in a minimum number of iterations into this neighborhood. The learning process is guided by immediate reward signals describing if the current state matches general characteristics of the goal state. The computation of the reward signals is based on the reduced representation and allows the algorithm to perform in non-stationary environments since it does not require analytical knowledge of the goal state.

We demonstrated the algorithm for the logistic and the Hénon map, the Rössler system and a relaxation oscillator serving as a model for an arrhythmic heartbeat. Medical treatments have recently been proposed for the cure of epilepsy [GNN+96] and atrial fibrillation [LBM+99, WMV+00] which are based on the OGY control scheme. Ultimately the goal is here the development of less invasive implantable medical devices. Since our algorithm does not require analytical knowledge and is applicable in non-stationary environments it could be well suited for the design of such chaos control devices. In this context future research must focus on the combination of the control algorithm with model approximation techniques such that the controller interacts in part with a simulated environment instead of the real system. This could greatly reduce the amount of interaction of the controller with the real system and lead to less invasive control. One possibility here is to approximate local linear models as proposed by Martinetz *et al.* [MBS93] who used the approximated models to predict the dynamics of the system.

Of current interest is the control of high-dimensional chaotic or spatially extended systems. We demonstrated an impulsive control of a nine-dimensional hy-

perchaotic Lorenz system and the parametric control of 1-D and 2-D logistic coupled map lattices. These results motivate to investigate the reinforcement learning control of spatially extended systems in more detail in future work. Specifically so-called smart matter applications are interesting testing grounds for multi-agent control systems based on reinforcement learning.

The proposed algorithm was also applied to the control of a noisy mechanical rotor serving as a model for a multistable system. In these applications we could think of the construction of a pattern recognition device in which input activates specific control policies which quickly stabilize a certain state which represents the output associated with the input. Chaos, multistability and the control of these dynamical behaviours have been suggested to play a fundamental role for neural information processing [SF87]. Recent neurophysiological evidence on the activities of cells in nuclei in the monkey brain shows that reward learning can model the activities resulting from a conditioning experiment [MDPS95, MDPS96]. Our results supports the suspected role of reinforcement learning and chaos in neural information processing since they show the feasibility of efficient chaos control by reward learning. Future work must focus on an extension of the introduced ideas to realistic models of the brain.

The method requires a discrete state representation which was computed in advance from sample data with the Neural-Gas algorithm. Many different vector quantization techniques exist in the literature and can be used instead. The question of how to obtain an optimal discrete representation must still be investigated. An initial attempt combined the approximation of the finite representation with the policy approximation and led to representations suited for the particular control problem. Clearly the idea of combining growing vector quantizers and reinforcement learning is not restricted to the control of chaotic systems and it should

be interesting to apply similar ideas to general reinforcement learning control problems. Furthermore, the application to systems with a reconstructed phase space such as a space of embedding vectors [ABST93] must be investigated to make the approach more suitable for realistic applications.

Allowing smaller perturbations close to the fixed point resulted in smaller control signals necessary to stabilize the desired state. The algorithm based solely on reinforcement learning as proposed in Chapter 6 allows only discrete perturbations. In certain situations it can be desirable to allow locally continuous perturbations and we described how this can easily be done once the discrete policy has been established. Although it is technically possible to compute globally continuous policies this is currently practically infeasible and does not considerably decrease transient times.

Current research in the reinforcement learning literature is focusing on the development of algorithms suitable in continuous phase and action spaces and on the application of reinforcement learning to high-dimensional systems. Successes in the field of reinforcement learning could be applied to the control of complex systems. On the other hand, the control of complex systems provides an interesting set of problems for the testing of new reinforcement learning control algorithms.

In summary, we suggested an efficient general purpose chaos controller which requires a minimum of knowledge and has promise for a "black box" chaos controller. Unfortunately all results were obtained through numerical simulations and such simulations certainly do not prove the practical use of a method. It is our hope that future experimental implementations will reveal the strengths and the shortcomings of the proposed method.

Bibliography

[AAS95] V. Astakhov, V. Anishchenko, and A. Shabunin. Controlling spatiotemporal chaos in a chain of the coupled logistic maps. *IEEE Trans. on Circuits and Systems - I: Fundamental Theory and Applications.* **42**:352–357, 1995.

[AB97] F. T. Arecchi and S. Boccaletti. Adaptive strategies for recognition, noise filtering, control, synchronization and targeting of chaos. *Chaos.* **7**:621–634, 1997.

[Aba95] H. Abarbanel. *Analysis of observed chaotic data.* Institute for Nonlinear Science, Springer, 1995.

[ABST93] H. Abarbanel, R. Brown, J. Sidorowich, and L. Tsimring. The analysis of observed chaotic data in physical systems. *Reviews of Modern Physics.* **65**:1331–1392, 1993.

[AD89] D. Armbruster and G. Dangelmayr. Circuit induced oscillator of SNDC semiconductors. *Physics Letters A.* **138**:46–50, 1989.

[AFH94] J. Argyris, G. Faust, and M. Haase. *An Exploration of Chaos.* Elsevier Science, Amsterdam, 1994.

[AGK91] D. Aronson, M. Golubitsky, and M. Krupa. Coupled arrays of Josephson junctions and bifurcation of maps with s_n symmetry. *Nonlinearity.* **4**:861–902, 1991.

[AGK94] P. Alsing, A. Gavrielides, and V. Kovanis. Using neural networks for controlling chaos. *Physica Review E.* **49**:1225–1231, 1994.

[AHKY96] J. Alexander, B. Hunt, I. Kan, and J. Yorke. Intermingled basins for the triangle map. *Ergodic Theory and Dyn. Sys.*, **16**:651, 1996.

[AK93] H. Abarbanel and M. Kennel. Local false nearest neighbors and dynamical dimensions from observed chaotic data. *Physical Review E.* **47**:3057–3068, 1993.

[AK99] M. Anderle and M. Kirby. Adaptive clustering based on local neighborhood interactions. In *Proc. SPIE Vol. 3807, Advanced Signal Processing Algorithms, Architectures, and Implementations IX*, pages 288–297, 1999.

[AKYY92] J. Alexander, I. Kan, J. Yorke, and Z. You. Riddled basins. *Int. J. Bif. Chaos*, **2**:795–813, 1992.

[AMPT82] F. T. Arecchi, R. Meucci, G. Puccioni, and J. Tredicce. Experimental evidence of sub-harmonic bifurcations, multistability, and turbulence in a Q-switched gas-laser. *Physical Review Letters*, **49**:1217–1220, 1982.

[And95] J. Anderson. *Computational Fluid Dynamics*. McGraw-Hill, New York, NY, 1995.

[AnLT94] I. Aranson and H. Levine nd L. Tsimring. Controlling spatiotemporal chaos. *Physical Review Letters*, **72**:2561–2564, 1994.

[AT00] P. Ashwin and J. Terry. On riddling and weak attractors. *Physica D*, **142**:87–100, 2000.

[ATJR98] P. Ashwin, J. Terry, K.S. Thornburg Jr. and R. Roy. Blowout bifurcation in a system of coupled chaotic lasers. *Phys. Rev. E*, **6**:7186–7189, 1998.

[Aue94] D. Auerbach. Controlling extended systems of chaotic elements. *Physical Review Letters*, **72**:1184–1187, 1994.

[Bar96] M. Barrett. Continuous control of chaos. *Physica D*, **91**:340–348, 1996.

[BBL+91] M. Brambilla, F. Battipede, L. Lugiato, V. Penna, F. Prati, C. Tamm, and C. Weiss. Transverse laser patterns. I. Phase singularity crystals. *Physical Review A*, **43**:5090–5113, 1991.

[BC98] M. Baptista and I. Caldas. Easy-to-implement method to target nonlinear systems. *Chaos*, **8**:290–299, 1998.

[BDG93] S. Bielawski, D. Derozier, and P. Glorieux. Experimental characterization of unstable periodic orbits by controlling chaos. *Physical Review A*, **47**:R2492–R2495, 1993.

[BDG94] S. Bielawski, D. Derozier, and P. Glorieux. Controlling unstable periodic orbits by a delayed continuous feedback. *Physical Review E*, **49**:R971–R974, 1994.

[Bel57] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[Ber87] D. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models.* Prentice-Hall, Englewood Cliffs, NJ, 1987.

[Ber95] D. Bertsekas. *Dynamic Programming and Optimal Control,* volume 1 and 2. Athena Scientific, Belmont, Massachusetts, 1995.

[Ber99] D. Bertsekas. *Nonlinear Programming.* Athena Scientific, Belmont, MA, 1999.

[BG91] Y. Braiman and I. Goldhirsch. Taming chaotic dynamics with weak periodic perturbations. *Physical Review Letters,* **66**:2545–2548, 1991.

[BG95] E. Barreto and C. Grebogi. Multiparameter control of chaos. *Physical Review E.* **52**:3553–3557, 1995.

[BGL+00] S. Boccaletti, C. Grebogi, Y.-C. Lai, H. Mancini, and D. Maza. The control of chaos: Theory and applications. *Physics Reports.* **329**:103–197, 2000.

[BK86] D. Broomhead and G. King. Extracting qualitative dynamics from experimental data. *Physica D.* **20**:217, 1986.

[BLP+91] M. Brambilla, L. Lugiato, V. Penna, F. Prati, C. Tamm, and C. Weiss. Transverse laser patterns. II. Variational principle for pattern selection, spatial multistability, and laser hydrodynamics. *Physical Review A.* **43**:5114–5120, 1991.

[BM95] E. Bollt and J. Meiss. Targeting chaotic orbits to the moon through recurrence. *Physics Letters A,* **204**:373–378, 1995.

[BM96] D. Battogtokh and A. Mikhailov. Controlling turbulence in the complex ginzburg-landau equation. *Physica D.* **90**:84–95, 1996.

[BP92] T. Buzug and G. Pfister. Comparison of algorithms calculating optimal parameters for delay time coordinates. *Physica D.* 58:127, 1992.

[BP99] R. Badii and A. Politi, editors. *Complexity: Hierarchical Structures and Scaling in Physics.* Cambridge University Press, 1999.

[Bre94] J. Breeden. Open-loop control of nonlinear systems. *Physics Letters A.* **190**:264–272, 1994.

[BRR95] J. Botina, H. Rabitz, and N. Rahman. Optimal control of chaotic Hamiltonian dymamics. *Physical Review A.* **51**:923–933, 1995.

[Bry96] A. Bryson. Optimal control - 1950 to 1985. *IEEE Control Systems.* **16**(3):26–33, 1996.

[BS95] J. Bruske and G. Sommer. Dynamic cell structure learns perfectly topology preserving map. *Neural Computation*, **7**:845–865, 1995.

[BS96] M. Bleich and J. Socolar. Controlling spatiotemporal dynamics with time-delay feedback. *Physical Review E*, **54**:R17–R20, 1996.

[BSA83] A. Barto, G. Sutton, and C. Anderson. Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, **13**:835–846, 1983.

[BT96] D. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

[BV94] P. Bayly and L. Virgin. Practical considerations in the control of chaos. *Physical Review E*, **50**:604–607, 1994.

[BX96] S. Bishop and D. Xu. Control of chaos in noisy flows. *Physical Review E*, **54**:3204–3210, 1996.

[Cao97] L. Cao. Practical method for determining the minimum embedding dimensions of a scalar time series. *Physica D*, 110:43–50, 1997.

[Cas96] G. Casati. Quantum chaos. *Chaos*, **6**:391–398, 1996.

[CB96] P. Colet and Y. Braiman. Control of chaos in multimode solid state lasers by the use of small periodic perturbations. *Physical Review E*, **53**:200–206, 1996.

[CC95] D. Christini and J. Collins. Controlling neuronal noise using chaos control. *Preprint: http://xxx.lanl.gov/find*, 1995.

[CC96] D. Christini and J. Collins. Using chaos control and tracking to suppress a pathological nonchaotic rhythm in a cardiac model. *Physical Review E*, **53**:R49–R52, 1996.

[CC97a] D. Christini and J. Collins. Control of chaos in excitable physiological systems: A geometric analysis. *Chaos*, **7**:544–549, 1997.

[CC97b] D. Christini and J. Collins. Real-time, adaptive, model-independent control of low-dimensional chaotic and nonchaotic dynamical systems. *IEEE Transactions on Circuits and Systems*, **44**:1027–1030, 1997.

[CD95] G. Chen and X. Dong. Identification and control of chaotic systems: An artificial neural network approach. In *Proc. of the IEEE Int. Conf. on Neural Networks*, pages 1177–1182, 1995.

[CEFG91] M.C. Casdagli, S. Eubank, D. Farmer, and J. Gibson. State space reconstruction in the presence of noise. *Physica D*, 51:52–98, 1991.

[CH93] M. Cross and P. Hohenberg. Pattern formation outside of equilibrium. *Reviews of Modern Physics*, **65**:851–1112, 1993.

[Cha97] N. Chau. Controlling chaos by periodic proportional pulses. *Physical Letters A*, **234**:193–197, 1997.

[Cha98] N. Chau. Controlling continuous chaotic dynamics by periodic proportional pulses. *Physical Review E*, **57**:378–380, 1998.

[Che] G. Chen. Control and synchronization of chaotic systems (bibliography). ECE Dept., Univ. of Houston, TX. ftp://ftp.egr.uh.edu/pub/TeX/chaos.tex.

[Chi79] B. V. Chirikov. Universal instability of many-dimensional oscillator systems. *Phys. Rep.*, **52**:263–379, 1979.

[Chi99] C. Chicone. *Ordinary differential equations with applications*. Springer-Verlag, New York, 1999.

[CO93] K. Cuomo and A. Oppenheim. Circuit implementation of synchronized chaos with applications to communications. *Physical Review Letters*, **71**:65–68, 1993.

[CS94] T. Carr and I. Schwartz. Controlling unstable steady states using system parameter variation and control duration. *Physica Review E*, **50**:3410–3415, 1994.

[CS95] T. Carr and I. Schwartz. Controlling the unstable steady state in a multimode laser. *Physical Review E*, **51**:5109–5111, 1995.

[CTSP92] T. Carroll, I. Triandaf, I. Schwartz, and L. Pecora. Tracking unstable orbits in an experiment. *Physica Review A*, **46**:6189–6192, 1992.

[Cvi89] P. Cvitanović, editor. *Universality in Chaos*. Adam Hilger, Bristol, 1989.

[Des94] A. Destexhe. Oscillations, complex spatiotemporal behaviour, and information transport in networks of excitatory and inhibitory neurons. *Physical Review E*, **50**:1594–1606, 1994.

[Dev87] R.L. Devaney. *An Introduction to Chaotic Dynamical Systems*. Addison-Wesley, 1987.

[DGOS93] M. Ding, C. Grebogi, E. Ott, and T. Sauer. Plateau onset for correlation dimension - when does it occur? *Physical Review Letters*, **70**:3872–3875, 1993.

[DH94] R. Der and M. Herrmann. Q-learning chaos controller. *1994 IEEE International Conference on Neural Networks*, **4**:2472–2475, 1994.

[DK98] G. Dangelmayr and L. Kramer. Mathematical tools for pattern formation. In F. H. Busse and S. C. Mueller, editors, *Evolution of Spontaneous Structures in Dissipative Continuous Systems*, pages 1–85. Springer, 1998.

[DL99a] R. Davidchack and Y.-C. Lai. Efficient algorithm for detecting unstable periodic orbits in chaotic systems. *Physical Review E*, **60**:6172–6175, 1999.

[DL99b] M. Dhamala and Y.-C. Lai. Controlling transient chaos in deterministic flows with applications to electrical power systems and ecology. *Phys. Rev. E*, 59:1646–1655, 1999.

[DN92] U. Dressler and G. Nitsche. Controlling chaos using time delay coordinates. *Physical Review Letters*, **68**:1–4, 1992.

[DRS90] W. Ditto, S. Rauseo, and M. Spano. Experimental control of chaos. *Pysical Review Letters*, **65**:3211–3214, 1990.

[DSHK99] G. Dangelmayr, S.Gadaleta, D. Hundley, and M. Kirby. Time series prediction by estimating markov probabilities through topology preserving maps. In *Proc. SPIE Vol. 3812, Applications and Science of Neural Networks, Fuzzy Systems, and Evolutionary Computation II*, pages 86–93, 1999. Preprint: http://www.math.colostate.edu/-~sabino/publications.html.

[dSVL96] M. de Sousa Vieira and A. Lichtenberg. Controlling chaos using nonlinear feedback with delay. *Physical Review E*, **54**:1200–1207, 1996.

[DYT+96] M. Ding, W. Yang, V. In, W. Ditto, M. Spano, and B. Gluckman. Controlling chaos in high dimensions: Theory and experiment. *Physical Review E*, **53**:4334–4344, 1996.

[ED98] B. Epureanu and E. Dowell. On the optimality of the Ott-Grebogi-Yorke control scheme. *Physica D*, **116**:1–7, 1998.

[ER85] J.P. Eckmann and D. Ruelle. Ergodic theory of chaos and strange attractors. *Rev. Mod. Phys.*, 57:617–56, 1985.

[Erm94] G. Ermentrout. An introduction to neural oscillators. In F. Ventriglia, editor, *Neural Modeling and Neural Networks*, pages 79–110. Pergamon Press, Oxford, 1994.

[Far82] J.D. Farmer. Chaotic attractors of an infinite-dimensional dynamical system. *Physica D*, 4:366–393, 1982.

[FB99] F. Fernández and D. Borrajo. Vector quantization applied to reinforcement learning. *Preprint*, 1999. http://grial.uc3m.es/~ffernand/-publicationes/ijcai99/index.html.

[FBS99] G. Franceschini, S. Bose, and E. Schöll. Control of chaotic spatiotemporal spiking by time-delay autosynchronization. *Physical Review E.* **60**:5426-5434, 1999.

[FG97] U. Feudel and C. Grebogi. Multistability and the control of complexity. *Chaos.* **7**:597-604, 1997.

[FGHY96] U. Feudel, C. Grebogi, B. Hunt, and J. Yorke. Map with more than 100 coexisting low-period periodic attractors. *Physical Review E.* **54**:71-81. 1996.

[FHD97] M. Funke, M. Herrmann, and R. Der. Controlling low-dimensional chaos: Determination and stabilization of unstable periodic orbits by kohonen neural nets. *Int. Journal of Adaptive Control and Signal Processing.* **2**:489-499. 1997.

[FLMM96] J. Foss, A. Longtin, B. Mensour, and J. Milton. Multistability and delayed recurrent loops. *Physical Review Letters.* **76**:708-711. 1996.

[FMM97] J. Foss, F. Moss, and J. Milton. Noise, multistability, and delayed recurrent loops. *Physical Review E.* **55**:4536-4543. 1997.

[Fra89] A. Fraser. Information and entropy in strange attractors. *IEEE Transactions on Information Theory.* **35**:245-262. 1989.

[Fre86] W. Freeman. Petit mal seizure spikes in olfactory bulb and cortex caused by runaway inhibition after exhaustion of excitation. *Brain Research Reviews.* **11**:259-284. 1986.

[Fri94] B. Fritzke. Growing cell structures - a self-organizing network for unsupervised and supervised learning. *Neural Networks.* **7 (9)**:1441-1460. 1994.

[Fri95a] B. Fritzke. A growing neural gas network learns topologies. In G. Tesauro, D. Touretzky, and T. Leen, editors. *Advances in Neural Information Processing Systems 7.* pages 625-632. MIT Press, Cambridge MA. 1995.

[Fri95b] B. Fritzke. Growing self-organizing networks - why? In M. Verleysen, editor. *ESANN'96: European Symposium on Artificial Neural Networks*, pages 61-72. D-Facto Publishers, Brussels, 1995.

[Fri95c] B. Fritzke. Incremental learning of local linear maps. In F. Fogelman and P. Gallinari, editors, *ICANN'95: Proceedings of the International Conference on Artificial Neural Networks, Paris, France*, pages 217–222. EC2 & Cie, 1995.

[Fri97a] B. Fritzke. The LBG-U method for vector quantization - an improvement over LBG inspired from neural networks. *Neural Processing Letters*, **5(1)**, 1997.

[Fri97b] B. Fritzke. Some competitive learning methods, 1997. unpublished report. http://pikas.inf.tu-dresden.de/~fritzke.

[FS86] A. Fraser and H. Swinney. Independent coordinates for strange attractors from mutual information. *Physical Review A*, **33**:1134–1140, 1986.

[FV97] A. Fouladi and J. Valdivia. Period control of chaotic systems by optimization. *Physical Review E*, **55**:1315–1320, 1997.

[Gad98] P. Gade. Feedback control in coupled map lattices. *Physical Review E*, **57**:7309–7312, 1998.

[Gal68] R. Gallagher. *Information theory and reliable communication*. Wiley, New York, 1968.

[GCS97] R. Grigoriev, M. Cross, and H. Schuster. Pinning control of spatiotemporal chaos. *Physical Review Letters*, **79**:2795–2798, 1997.

[GD91] T. Gencic and G. Dangelmayr. Synchronized states in systems of neural oscillators. In K. Goser, U. Ramacher, and U. Rueckert, editors, *Proc. 1st Int. Workshop on Microelectronics for Neural Networks*, pages 171–176. University of Dortmund, 1991.

[GD99] S. Gadaleta and G. Dangelmayr. Optimal chaos control through reinforcement learning. *Chaos*, **9**:775–788, 1999.

[GD00a] S. Gadaleta and G. Dangelmayr. Control of 1-D and 2-D coupled map lattices through reinforcement learning. In *Proceedings of Second Int. Conf. "Control of Oscillations and Chaos"*, volume **1**, pages 109–112. St. Petersburg, Russia, 2000. ISBN 0-7803-6434-1.

[GD00b] S. Gadaleta and G. Dangelmayr. Control of a noisy mechanical rotor (Invited paper). In *Problems in Modern Applied Mathematics*, pages 216–221. World Scientific Engineering Society Press, 2000.

[GD01] S. Gadaleta and G. Dangelmayr. Learning to control a complex multistable system (accepted for publication). *Physical Review E*, 2001.

[GDG90] T. Gencic, G. Dangelmayr, and W. Guettinger. Storing cycles in analog neural networks. In R. Eckmiller, G. Hartmann, and G. Hauske, editors, *Parallel processing in neural systems and computers*, pages 445–450. North-Holland, 1990.

[Ger79] A. Gersho. Asymptotically optimal block quantizations. *IEEE Transactions on Information Theory*, **25(4)**:373–380, 1979.

[GG92] A. Gersho and R.M. Gray. *Vector Quantization and Signal Compression*. Kluwer Publishers, 1992.

[GHH97] O. Guenther, T. Hogg, and B.A. Huberman. Learning in multiagent control of smart matter. *American Association for Artificial Intelligence Workshop on Multiagent Learning*, pages 1–5, 1997.

[GK93] H. Gang and H. Kaifen. Controlling chaos in systems described by partial differential equations. *Physical Review Letters*, **71**:3794–3797, 1993.

[GL97] C. Grebogi and Y.-C. Lai. Controlling chaos in high dimensions. *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications*, **44**:971–975, 1997.

[Gle87] J. Gleick. *Chaos: Making a new science*. Penguin Books, USA, 1987.

[GM93] J. Güémez and M. Matías. Control of chaos in unidimensional maps. *Physics Letters A*, **181**:29–32, 1993.

[GNN+96] B. Gluckman, E. Neel, T. Netoff, W. Ditto, M. Spano, and S. Schiff. Electric field suppression of epileptiform activity in hippocampal slices. *Journal of Neurophysiology*, **76**:4202–4205, 1996.

[GOY88] C. Grebogi, E. Ott, and J.A. Yorke. Unstable periodic orbits and the dimension of multifractal chaotic attractors. *Physical Review A*, 37(5):1711–1724, 1988.

[GP83] P. Grassberger and I. Procaccia. Characterization of strange attractors. *Physical Review Letters*, 50:346–349, 1983.

[Gra84] R.M. Gray. Vector quantization. *IEEE ASSP Magazine*, **1**:4–29, 1984.

[Gra94] C.M. Gray. Synchronous oscillations in neuronal systems: mechanism and functions. *Journal of Computational Neuroscience*, **1**:11–38, 1994.

[GSDW92] A. Garfinkel, M. Spano, W. Ditto, and J. Weiss. Controlling cardiac chaos. *Science*, **257**:1230–1235, 1992.

[Gul90] V. Gullapalli. A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks*, **3**:671–692, 1990.

[GZ94] H. Gang and Q. Zhilin. Controlling spatiotemporal chaos in coupled map lattice systems. *Physical Review Letters*, **72**:68–71, 1994.

[Hay99] S. Haykin. *Neural Networks: A comprehensive foundation.* Prentice-Hall, Inc., Upper Saddle River, New Jersey, 1999.

[HDM+94] B. Hübinger, R. Doerner, W. Martienssen, M. Herdering, R. Pitka, and U. Dressler. Controlling chaos experimentally in systems exhibiting large effective Lyapunov exponents. *Pysical Review E*, **50**:932–948, 1994.

[Hén76] M. Hénon. A two-dimensional mapping with a strange attractor. *Commun. Math. Phys.*, **50**:69, 1976.

[Her88] H. Herzel. Stabilization of chaotic orbits by random noise. *ZAMM*, **68**:582–583, 1988.

[HH98] Tad Hogg and Bernardo A. Huberman. Controlling smart matter. *Smart Materials and Structures*, 7:R1–R14, 1998.

[HI97] F. Hoppensteadt and E. Izhikevich. *Weakly connected neural networks.* Springer-Verlag, New York, 1997.

[HI00] F. Hoppensteadt and E. Izhikevich. Synchronization of mems resonators and mechanical neurocomputing. *submitted to IEEE Transactions On Circuits and Systems I.*, 2000.

[HJJY96] Z. Hong, Y. Jie, W. Jiao, and W. Yinghai. General method of controlling chaos. *Physical Review E*, **53**:299–306, 1996.

[HM81] J. Hudson and J. Mankin. Chaos in the Belousov-Zhabotinskii reaction. *Journal of Chemical Physics*, **74**:6171–6177, 1981.

[HO96] B. Hunt and E. Ott. Optimal periodic orbits of chaotic systems occur at low period. *Physical Review E*, **54**:328–337, 1996.

[Hof94] A. Hoff. Chaos control and neural classification. *Z. Naturforsch.*, **49a**:589–593, 1994.

[Hop82] J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of National Academy of Sciences USA*, **79**:2554–2558, 1982.

[Hop84] J.J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Natl. Acad. Sci. USA*, **81**:3088–3092, 1984.

[Hun91] E. Hunt. Stabilizing high-period orbits in a chaotic system: The diode resonator. *Physical Review Letters*, **67**:1953–1955, 1991.

[Hun98] D. Hundley. *Local Nonlinear Modeling via Neural Charts.* PhD dissertation, Colorado State University, Department of Mathematics, 1998.

[HXCP98] G. Hu, J. Xiao, L. Chua, and L. Pivka. Controlling spiral waves in a model of two-dimensional arrays of Chua's circuits. *Physical Review Letters*, **80**:1884–1887, 1998.

[HYL98] G. Hu, J. Yang, and W. Liu. Instability and controllability of linearly coupled oscillators: Eigenvalue analysis. *Physical Review E*, **58**:4440–4453, 1998.

[IMDS95] V. In, S. Mahan, W. Ditto, and M. Spano. Experimental maintenance of chaos. *Physical Review Letters*, **74**:4420–4423, 1995.

[Jac97] A. Jackson. The OPCL control method for entrainment, model-resonance, and migration actions on multiple-attractor systems. *Chaos*, **7**:550–558, 1997.

[JAPE97] Y. Jiang, A. Antillón, P. Parmananda, and J. Escalona. Selection and stabilization of spatiotemporal patterns in two-dimensional coupled map lattices. *Physical Review E*, **56**:2568–2572, 1997.

[JGT+98] W. Jinlan, C. Guangzhi, Q. Tuanfa, N. Wansun, and W. Xuming. Synchronizing spatiotemporal chaos in coupled map lattices via active-passive decomposition. *Physical Review E*, **58**:3017–3021, 1998.

[JJS94] T. Jaakkola, M. Jordan, and S. Singh. On the convergence of stochastic dynamic programming algorithms. *Neural Computation*, **6**:1185–1201, 1994.

[JP98] Y. Jiang and P. Parmananda. Synchronization of spatiotemporal chaos in asymmetrically coupled map lattices. *Physical Review E*, **57**:4135–4139, 1998.

[KA95] M. Kennel and H. Abarbanel. False neighbors and false strands: A reliable minimum embedding dimension algorithm. *submitted to Phys. Rev. E*, 1995.

[Kan89] K. Kaneko. Chaotic but regular posi-nega switch among coded attractors by cluster-size variation. *Physical Review Letters*, **63**:219–223, 1989.

[Kan92] K. Kaneko. Overview of coupled map lattices. *Chaos*, **2**:279–282, 1992.

[Kap96] T. Kapitaniak. *Controlling Chaos.* Academic Press, San Diego, CA 92101, 1996.

[KAS87] T. Kapitaniak, J. Awrejcewicz, and W.-H. Steeb. Chaotic behaviour of anharmonic oscillator with almost periodic excitation. *J. Phys. A*, **20**:L355–359, 1987.

[KASM82] W. Kennel, R. Abbott, D. Savage, and P. McNamara. Epidemiologic features of atrial fibrillation: the framingham study. *N. England J. Medicine*, **306**:1018–1022, 1982.

[KBA92] M. Kennel, R. Brown, and H. Abarbanel. Determining minimum embedding dimension using a geometrical construction. *Physical Review A*. **45**:3403–3411, 1992.

[KCGV83] S. Kirkpatrick, Jr. C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, **220**:671–680, 1983.

[KES99] H. Kim, R. Eykholt, and J. Salas. Nonlinear dynamics, delay times, and embedding windows. *Physica D*, **127**:48–60, 1999.

[KFG99] S. Kraut, U. Feudel, and C. Grebogi. Preference of attractors in multistable systems. *Physical Review E*, **59**:5253–5260, 1999.

[KGOY93] E. Kostelich, C. Grebogi, E. Ott, and J. Yorke. Higher-dimensional targeting. *Physical Review E*. **47**:305–310, 1993.

[KH99] K. Konishi and H. Kokame K. Hirata. Coupled map car-following model and its delayed-feedback control. *Physica Review E*, **60**:4000–4007, 1999.

[KHK98] K. Konishi, M. Hirai, and H. Kokame. Decentralized delayed-feedback control of a coupled map model for open flow. *Physica Review E*. **58**:3055–3059, 1998.

[KHL99] Y. Kwon, S. Ham, and K. Lee. Analysis of minimal pinning density for controlling spatiotemporal chaos of coupled map lattices. *Physical Review E*, **55**:2009–2012, 1999.

[KK95] K. Konishi and H. Kokame. Stabilizing and tracking unstable focus points in chaotic systems using a neural network. *Physics Letters A*, **206**:203–210, 1995.

[KK97] K. Konishi and H. Kokame. Control of chaotic systems using an on-line trained linear neural controller. *Physica D*, **100**:423–438, 1997.

[KK98] K. Konishi and H. Kokame. Learning control of time-delay chaotic systems and its applications. *International Journal of Bifurcation and Chaos*, **8**:2457–2465, 1998.

[KK99] K. Konishi and H. Kokame. Decentralized delayed-feedback control of a one-way coupled ring map lattice. *Physica D*, **127**:1–12, 1999.

[KLM96] L. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: A survey. *Journal of Artifical Intelligence Research*, **4**:237–285, 1996.

[KLO96] M. Kushibe, Y. Liu, and J. Ohtsubo. Associative memory with spatiotemporal chaos control. *Physical Review E*, **53**:4502–4508, 1996.

[Koh89] T. Kohonen. *Self-Organization and Associative Memory, 3rd Ed.* Springer, 1989.

[Koh95] T. Kohonen. *Self-Organizing Maps.* Springer, 1995.

[KPSJ97] L. Kocarev, U. Parlitz, T. Stojanovski, and P. Janjić. Controlling spatiotemporal chaos in coupled nonlinear oscillators. *Physical Review E*, **56**:1238–1241, 1997.

[KRB94] Y. Kivshar, F. Rödelsperger, and H. Benner. Suppresion of chaos by nonresonant parametric perturbations. *Physical Review E*, **49**:319–324, 1994.

[KS99] H. Kantz and T. Schreiber. *Nonlinear time series analysis.* Cambridge University Press, Cambridge, UK, 1999.

[LB94] C. Lourenco and A. Babloyantz. Control of chaos in networks with delay: A model for synchronization of cortical tissue. *Neural Computation*, **6**:1141–1154, 1994.

[LBG80] Y. Linde, A. Buzo, and R.M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, **28**:84–95, 1980.

[LBM+99] J.J. Langberg, A. Bolmann, K. McTeague, M.L. Spano, V. In, J. Neff, B. Meadows, and W.L. Ditto. Control of human atrial fibrillation. Technical report, Applied Chaos Lab at Georgia Tech, http://www.physics.gatech.edu/chaos/, 1999.

[LJ99] C. Lin and C. Jou. Controlling chaos by GA-based reinforcement learning neural network. *IEEE Transactions on Neural Networks*, **10**:846–859, 1999.

[LK89] D. Lathrop and E. Kostelich. Characterization of an experimental strange attractor by periodic orbits. *Physical Review A*, 40:4028–4031, 1989.

[LL92] A. Lichtenberg and M. Lieberman. *Regular and Chaotic Dynamics.* Springer-Verlag, New York, 1992.

[Llo82] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, **28(2)**:129–137, 1982.

[LM82] A. Libchaber and J. Maurer. A Rayleigh Bénard experiment: Helium in a small box. In T. Rista, editor, *Nonlinear Phenomena at Phase Transitions and Instabilities*, pages 259–286. Plenum, New York, 1982.

[LMS95] D. Lebender, J. Müller, and F. Schneider. Control of chemical chaos and noise: A nonlinear neural net based algorithm. *J. Phys. Chem.*, **99**:4992–5000, 1995.

[Lor63] E.N. Lorenz. Deterministic nonperiodic flow. *J. Atmos. Sci.*, 20:130–141, 1963.

[LP98] R. Lima and M. Pettini. Parametric resonant control of chaos. *International Journal of Bifurcation and Chaos*, **8**:1675–1684, 1998.

[LS89] W. Liebert and H. Schuster. Proper choice of the time delays for the analysis of chaotic time series. *Phys. Lett. A*, 142:107, 1989.

[Mac65] J. MacQueen. On convergence of k-means and partitions with minimum average variance. *Ann. Math. Statist.*, **36**:1084, 1965. abstract.

[Mac67] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical statistics and probability*, pages 281–297. University of California Press, 1967.

[Mar93] T. Martinetz. Competitive hebbian learning rule forms perfectly topology preserving maps. In *ICANN'93: International Conference on Artificial Neural Networks, Amsterdam*, pages 427–434. Springer, 1993.

[MBS93] T. Martinetz, S. Berkovich, and K. Schulten. "Neural-Gas" Network for Vector Quantization and its application to Time-series Prediction. *IEEE Transactions on Neural Networks*, **4**:558–569, 1993.

[MC97] R. Montagne and P. Colet. Nonlinear diffusion control of spatiotemporal chaos in the complex Ginzburg-Landau equation. *Physical Review E*, **56**:4017–4024, 1997.

[MDPS95] P. Montague, P. Dayan, C. Person, and T. Sejnowski. Bee foraging in uncertain environments using predictive hebbian learning. *Nature*, **377**:725–728, 1995.

[MDPS96] P. Montague, P. Dayan, C. Person, and T. Sejnowski. A framework for mesencephalic dopamine systems based on predictive hebbian learning. *Journal of Neuroscience*, **16**:1936–1947, 1996.

[MG94] M. Matías and J. Güémez. Stabilization of chaos by proportional pulses in the system variables. *Physical Review Letters*, **72**:1455–1462, 1994.

[MG96] M. Matías and J. Güémez. Chaos suppression in flows using proportional pulses in the system variables. *Physical Review E*, **54**:198–209, 1996.

[MKH91] P. Marmillot, M. Kaufmann, and J.-F. Hervagault. Multiple steady states and dissipative structures in a circular and linear array of three cells: Numerical and experimental approaches. *The Journal of Chemical Physics*, **95**:1206–1214, 1991.

[MMJ95] A. Mulpur, S. Mulpur, and B. Jang. Control of chaotic oscillations using neural networks. In *Proc. of the 4th IEEE Conf. on Control Applications*, pages 560–565, 1995.

[Mob] http://cs.wwu.edu/faculty/mobus/.

[Moo94] A. Moore. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. In J. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 711–718. Morgan Kaufmann, San Mateo, CA, 1994.

[MS94] T. Martinetz and K. Schulten. Topology representing networks. *Neural Networks*, **7**:507–522, 1994.

[Mur93] J. Murray. *Mathematical Biology*. Springer-Verlag, Berlin, 1993.

[NAGK94] T. Newell, P. Alsing, A. Gavrielides, and V. Kovanis. Synchronization of chaotic diode resonators by occasional proportional feedback. *Physical Review Letters*, **72**:1647–1650, 1994.

[NGG98] K. Narayanan, R. Govindan, and M. Gopinathan. Unstable periodic orbits in human cardiac rhythms. *Physical Review E*, 57:4594–4603, 1998.

[NU98] H. Nakajima and Y. Ueda. Half-period delayed feedback control for dynamical systems with symmetries. *Physical Review E*, **58**:1757–1763, 1998.

[ODG96] J. Oppenlaender, G. Dangelmayr, and W. Guettinger. Nonlinear pattern dynamics in Josephson-junction arrays. *Phys. Rev. B*, **54**:1213–1227, 1996.

[OF95] K. Otawara and L. Fan. Controlling chaos with an artificial neural network. In *Proc. IEEE Int. Joint Conf Fuzzy Syst., vol. 4*, pages 1943–1948. 1995.

[OGY90] E. Ott, C. Grebogi, and J.A. Yorke. Controlling chaos. *Physical Review Letters*, **64**:1196–1199, 1990.

[PC90] L. Pecora and T. Carroll. Synchronization in chaotic systems. *Physical Review Letters*, **64**:821–824, 1990.

[PF94] D. Peak and M. Frame. *Chaos under Control. The Art and Science of Complexity*. W. H. Freeman and Company, New York, 1994.

[PFTV86] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes*. Cambridge University Press, New York, New Rochelle, Melbourne, Sydney, 1986.

[PG95] L. Poon and C. Grebogi. Controlling complexity. *Physical Review Letters*, **75**:4023–4026, 1995.

[PHE97] P. Parmananda, M. Hildebrand, and M. Eiswirth. Controlling turbulence in coupled map lattice systems using feedback techniques. *Physical Review E*, **56**:239–244, 1997.

[PKC98] A. Pisarchik, B. Kuntsevich, and R. Corbalán. Stabilizing unstable orbits by slow modulation of a control parameter in a dissipative dynamic system. *Physical Review E*, **57**:4046–4053, 1998.

[PKK98] N. Parekh, R. Kumar, and D. Kulkarni. Synchronization and control of spatiotemporal chaos using time-series data from local regions. *Chaos*, **8**:300–305, 1998.

[PMT95] M. Paskota, A. Mees, and K. Teo. Directing orbits of chaotic dynamical systems. *Int. J. of Bifurcation and Chaos*, **5**:573–583, 1995.

[Poi99] H. Poincaré. *Les méthodes nouvelles de la mécanique céleste*. Gauthier-Villars, Paris, 1899.

[PPS91] B. Peng, V. Petrov, and K. Showalter. Controlling chemical chaos. *J. Phys. Chem. US*, **95**:4957–4959, 1991.

[PPS98] N. Parekh, S. Parthasarathy, and S. Sinha. Global and local control of spatiotemporal chaos in coupled map lattices. *Physical Review Letters*, **81**:1401–1404, 1998.

[PS94] F. Pineda and J. Sommerer. Estimating generalized dimensions and choosing time delays: A fast algorithm. In A. Weigend and N. Gershenfeld, editors, *Time Series Prediction: Forecasting the Future and Understanding the Past*, pages 367–385. Addison-Wesley, Reading, 1994.

[PWS94] F. Prengel, A. Wacker, and E. Schöll. Simple model for multistability and domain formation in semiconductor superlattices. *Physical Review B*. **50**:1705–1712, 1994.

[Pyr92] K. Pyragas. Continuous control of chaos by self-controlling feedback. *Physics Letters A*, **170**:421–428, 1992.

[QMAV97] M. Le Van Quyen, J. Martinerie, C. Adam, and F. Varela. Unstable periodic orbits in human epileptic activity. *Physical Review E*. **56**:3401–3411, 1997.

[RCL94] M. Rosenstein, J. Collins, and C. De Luca. Reconstruction expansion as a geometry-based framework for choosing proper delay times. *Physica D*, 73:82, 1994.

[RFK99] W. Rappel, F. Fenton, and A. Karma. Spatiotemporal control of wave instabilities in cardiac tissue. *Physical Review Letters*, **83**:456–459, 1999.

[RGF90] K. Rose, F. Gurewitz, and G. Fox. Statistical mechanics and phase transitions in clustering. *Physical Review Letters*, **65 (8)**:945–948, 1990.

[RGOD91] F. Romeiras, C. Grebogi, E. Ott, and W. Dayawansa. Controlling chaotic dynamical systems. *Physica D*, **58**:165–192, 1991.

[RLS+98] P. Reiterer, C. Lainscsek, T. Schürrer, C. Letellier, and J. Maquet. A nine-dimensional Lorenz system to study high-dimensional chaos. *J. Phys. A: Math. Gen.*, **31**:7121–7139, 1998.

[RM51] H. Robbins and S. Monro. A stochastic approximation method. *Ann. Math. Stat.*, **22**:400–407, 1951.

[RMMG92] R. Roy, T. Murphy, T. Maier, and Z. Gills. Dynamical control of a chaotic laser: Experimental stabilization of a globally coupled system. *Physical Review Letters*, **68**:1259–1262, 1992.

[RN94] G. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical report, Technical Report CUED/F-INFENG/TR 166, Engineering Department, Cambrige University, 1994.

[RRL78] O. Rössler, R. Rössler, and H. Landahl. Arrythmia in a periodically forced excitable system. *Sixth Int. Biophysics Congress, Kyoto, Abstracts*, page 296, 1978.

[RT71] D. Ruelle and F. Takens. On the nature of turbulence. *Commun. Math. Phys.*, 20:167–192, 1971.

[SB93] J. Sepulchre and A. Babloyantz. Controlling chaos in a network of oscillators. *Physical Review E*, **48**:945–950, 1993.

[SB98] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction.* Bradford, MIT press, Cambridge, Massachusetts, 1998.

[SBGS97] D. Sukow, M. Bleich, D. Gauthier, and J. Socolar. Controlling chaos in a fast diode resonator using extended time-delay atosynchronization: Experimental observations and theoretical analysis. *Chaos*, **7**:560–576, 1997.

[SCH99] H.-T. Su, Y.-H. Chen, and R.-R. Hsu. Master-slave scheme and controlling chaos in the Braiman-Goldhirsch method. *Physical Review E*. **59**:4687–4690, 1999.

[SCT97] I. Schwartz, T. Carr, and I. Triandaf. Tracking controlled chaos: Theoretical foundations and applications. *Chaos*. **7**:664–679, 1997.

[SD98] P. Schmelcher and F. Diakonos. General approach to the localization of unstable periodic orbits in chaotic dynamical systems. *Physical Review E*. **57**:2739–2746, 1998.

[SF87] C. Skarda and W. Freeman. How brains make chaos in order to make sense of the world. *Behav. Brain. Sci.*, **10**:161, 1987.

[SG98] S. Sinha and N. Gupte. Adaptive control of spatially extended coupled systems: Targeting spatiotemporal patterns and chaos. *Physical Review E*. **58**:R5221–R5224, 1998.

[SGK90] H. Sompolinsky, D. Golomb, and D. Kleinfeld. Global processing of visual stimuli in a neural network of coupled oscillators. *Proc. Natl. Acad. Sci. USA*, **87**:7200–7204, 1990.

[Sha48] C. Shannon. The mathematical theory of communication. *Bell Syst. Tech. Journal*, **27**:379–423,623–656, 1948.

[Sin94] S. Singh. *Learning to Solve Markovian Decision Processes.* PhD thesis, University of Massachusetts, Department of Computer Science, 1994.

[SJD+94] S. Schiff, K. Jerger, D. Duong, T. Chang, M. Spano, and W. Ditto. Controlling chaos in the brain. *Nature*, **370**:615–620, 1994.

[SJLS98] S. Singh, T. Jaakkola, M. Littman, and C. Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. To appear in *Machine Learning journal*. 1998.

[SN96] H. Schuster and E. Niebur. Parametric feedback resonance in chaotic systems. *Physical Review Letters*, **76**:400–403, 1996.

[SO95] P. So and E. Ott. Controlling chaos using time delay coordinates via stabilization of periodic orbits. *Physical Review E*, **51**:2955–2962, 1995.

[SO97] C. Schroer and E. Ott. Targeting in hamiltonian systems that have mixed regular/chaotic phase spaces. *Chaos*, **7**:512–519, 1997.

[SO99] K. Samejima and T. Omori. Adaptive internal state space construction method for reinforcement learning of a real-world agent. *Neural Networks*, **12**:1143–1155, 1999.

[SOGY90] T. Shinbrot, E. Ott, C. Grebogi, and J. Yorke. Using chaos to direct trajectories to targets. *Physical Review Letters*, **65**:3215–3218, 1990.

[SOS+96] P. So, E. Ott, S. Schiff, D. Kaplan, T. Sauer, and C. Grebogi. Detecting unstable periodic orbits in chaotic experimental data. *Physical Review Letters*, **76**:4705–4708, 1996.

[SOS+97] P. So, E. Ott, T. Sauer, B. Gluckman, C. Grebogi, and S. Schiff. Extracting unstable periodic orbits from chaotic time series data. *Physical Review E*, **55**:5398–5417, 1997.

[SP99] R. Sun and T. Peterson. Multi-agent reinforcement learning: weighting and partitioning. *Neural Networks*, **12**:727–753, 1999.

[Spa82] C. Sparrow. *The Lorenz equations: Bifurcations, chaos, and strange attractors*. Springer, New York, 1982.

[SSOY98] C. Schroer, T. Sauer, E. Ott, and J. Yorke. Predicting chaos most of the time from embeddings with self-intersections. *Physical Review Letters*, **80**:1410–1413, 1998.

[SSR98] J. Santamaría, R. Sutton, and A. Ram. Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive Behaviour*, **6**:163–218, 1998.

[ST92] I. Schwartz and I. Triandaf. Tracking unstable orbits in experiments. *Physica Review A*, **46**:7439–7444, 1992.

[Ste94] R. Stengel. *Optimal Control and Estimation*. Dover Publications, Inc., Mineola, N.Y., 1994.

[Str94] S. Strogatz. *Nonlinear Dynamics and Chaos*. Addison–Wesly, Reading, MA, 1994.

[Sut88] R. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, **3**:9–44, 1988.

[Sut96] R. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *In: D. Touretzky, M. Mozer and M. Hasselmo (eds.), Advances in Neural Information Processing Systems: Proceedings of the 1995 Conference.*, pages 1038–1044. 1996.

[SWB91] J. Singer, Y.-Z. Wang, and H. Bau. Controlling a chaotic system. *Physical Review Letters*, **66**:1123–1125, 1991.

[SYC91] T. Sauer, J. Yorke, and M. Casdagli. Embedology. *J. Stat. Phys.*, **65**:579, 1991.

[Tak81] F. Takens. Detecting strange attractors in turbulence. In D. Rand and L.-S. Young, editors, *Dynamical Systems and Turbulence, Lecture Notes in Math. Vol. 898*, pages 366–381. Springer, Heidelberg-New York, 1981.

[Tho11] E. Thorndike. *Animal Intelligence.* Hafner, Darien, CT, 1911.

[TJ00] A. Tsui and A. Jones. The control of higher dimensional chaos: comparative results for the chaotic satellite attitude control problem. *Physica D.* **135**:41–62, 2000.

[TJD+99] J.R. Terry, K.S. Thornburg Jr., D.J. DeShazer, G.D. Van Wiggeren, R. Roy, P. Ashwin, and S. Zhu. Synchronization of chaos in an array of three lasers. *Phys. Rev. E*, **59**:4036–4043, 1999.

[TK99] S. Theodoridis and K. Koutroumbas. *Pattern Recognition.* Academic Press, 1999.

[Tri88] D. Tritton. *Physical Fluid Dynamics.* St Edmundsbury Press Ltd. Bury St Edmunds, Suffolk, 1988.

[TS86] J. Thompson and H. Stewart. *Nonlinear dynamics and Chaos.* John Wiley, Chichester, Great Britain, 1986.

[TS97] E. Tziperman and Harvey Scher. Controlling spatiotemporal chaos in a realistic El Niño prediction model. *Physical Review Letters*, **79**:1034–1037, 1997.

[USOK98] A. Uchida, T. Sato, T. Ogawa, and F. Kannari. Nonfeedback control of chaos in a microchip solid-state laser by internal frequency resonance. *Physical Review E*, **58**:7249–7255, 1998.

[UV98] W. Uther and M. Veloso. Tree based discretization for continuous state space reinforcement learning, 1998.

[WA92] H. Wang and E. Abed. Bifurcation control of chaotic dynamical systems. In M. Fliess, editor, *Proceedings of the NOL-COS'92: Nonlinear Control System Design Symposium, Bordeaux, France*, pages 283–288. Pergamon, Oxford, UK, 1992.

[Wat89] C. Watkins. *Learning from delayed rewards.* PhD thesis, University of Cambridge, England, 1989.

[WB97] E. Weeks and J. Burgess. Evolving artifical neural networks to control chaotic systems. *Physical Review E*, **56**:1531–1540, August 1997.

[Whi36] H. Whitney. Differentiable manifolds. *Ann. Math.*, **37**:645, 1936.

[Wie99] K. Wiesenfeld. Josephson junction arrays: Puzzles and prospects. In M. Golubitsky, D. Luss, and S. Strogatz, editors, *Pattern Formation in Continuous and Coupled Systems*, pages 303–309. Springer-Verlag, New-York, 1999.

[Wig96] S. Wiggins. *Introduction to Applied Nonlinear Dynamical Systems and Chaos.* Springer, Berlin u.a., 1996.

[Win90] A. Winfree. *The geometry of biological time.* Springer-Verlag, Berlin, 1990.

[WKW95] K.-D. Weltmann, T. Klinger, and C. Wilke. Experimental control of chaos in a periodically driven glow discharge. *Pysical Review E*, **52**:2106–2109, 1995.

[WLL+98] H. Wellens, C. Lau, B. Lüderitz, M. Akhtar, A. Waldo, J. Camm, C. Timmermans, H. Tse, W. Jung, L. Jordaens, and G. Ayers. Atrioverter: An implantable device for the treatment of atrial fibrillation. *American Heart Association. Journal Circulation*, **98**:1651–1656, 1998.

[WMV+00] Ditto WL, Spano ML, In V, Neff J, Meadows B, Langberg JJ, Bolmann A, and McTeague K. Control of human atrial fibrillation. *Int. J. of Bifurcation and Chaos*, **10**:593–601, 2000.

[Wol86] S. Wolfram. *Theory and Applications of Cellular Automata.* World Scientific, Singapore, 1986.

[WR98] G. Van Wiggeren and R. Roy. Optical communication with chaotic waveforms. *Physical Review Letters*, **81**:3547–3550, 1998.

[WRS90] J. Walter, H. Ritter, and K. Schulten. Nonlinear prediction with self-organizing maps. In *IJCNN-90 Conf. Proc. vol.III, San Diego, CA*, pages 589–594, 1990.

[XH97] F. Xie and G. Hu. Spatiotemporal periodic and chaotic patterns in a two-dimensional coupled map lattice system. *Physical Review E*, **55**:79–86, 1997.

[YU98] K. Yagasaki and T. Uozumi. A new approach for controlling chaotic dynamical systems. *Physics Letters A*, **238**:349–357, 1998.

[YYY97] T. Yang, C. Yang, and L. Yang. Control of rössler system to periodic motions using impulsive control methods. *Physics Letters A*, **97**:356–361, 1997.

[Zad82] P. Zador. Asymptotic quantization-error of continuous signals and the quantization dimension. *IEEE Transactions on Information Theory*, **28**:139–149, 1982.

[ZG94a] Q. Zhilin and H. Gang. Spatiotemporal periodic patterns in symmetrically coupled map lattices. *Physical Review E*, **50**:163–170, 1994.

[ZG94b] Q. Zhilin and H. Gang. Spatiotemporal periodic states, periodic windows, and intermittency in coupled-map lattices. *Physical Review E*, **49**:1099–1108, 1994.

[ZL97] M. Zochowski and L. Liebovitch. Synchronization of the trajectory as a way to control the dynamics o a coupled system. *Physical Review E*, **56**:3701–3704, 1997.

[ZS97] L. Zonghua and C. Shigang. Control of chaos in conservative flows. *Physical Review E*, **56**:168–171, 1997.

Index

W