INCREMENTAL SYNTHESIS OF OPTIMAL CONTROL LAWS USING LEARNING ALGORITHMS

Stephen C. Atkins

S.B. Aeronautics and Astronautics
Massachusetts Institute of Technology
(1991)

SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE
in
AERONAUTICS AND ASTRONAUTICS
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
May, 1993

Signature of Author [illegible]
Department of Aeronautics and Astronautics
May 7, 1993

Approved by [illegible]
Walter L. Baker
Technical Supervisor, C. S. Draper Laboratory

Certified by [illegible]
Professor Wallace E. Vander Velde
Thesis Supervisor

Accepted by [illegible]
Professor Harold Y. Wachman
Chairman, Department Graduate Committee

INCREMENTAL SYNTHESIS OF OPTIMAL CONTROL LAWS USING LEARNING ALGORITHMS

Stephen C. Atkins

Submitted to the Department of Aeronautics and Astronautics
at the Massachusetts Institute of Technology
on May 7, 1993 in partial fulfillment of the requirements for the
degree of Master of Science in Aeronautics and Astronautics

ABSTRACT

Learning systems represent an approach to optimal control law design for situations where initial model uncertainty precludes the use of robust, fixed control laws. This thesis analyzes a variety of techniques for the incremental synthesis of optimal control laws, where the descriptor *incremental* implies that an on-line implementation filters the information acquired through real-time interactions with the plant and the operating environment. A *direct/indirect* framework is proposed as a means of classifying approaches to learning optimal control laws. Within this framework, relationships among existing direct algorithms are examined, and a specific class of indirect control laws is developed.

Direct learning control implies that the feedback loop that motivates the learning process is closed around system performance. Reinforcement learning is a type of direct learning technique with origins in the prediction of animal learning phenomena that is largely restricted to discrete input and output spaces. Three algorithms that employ the concept of reinforcement learning are presented: the Associative Control Process, Q learning, and the Adaptive Heuristic Critic.

Indirect learning control denotes a class of incremental control law synthesis methods for which the learning loop is closed around the system model. The approach discussed in this thesis integrates information from a learned mapping of the initially unmodeled dynamics into finite horizon optimal control laws. Therefore, the derivation of the control law structure as well as the closed-loop performance remain largely external to the learning process. Selection of a method to approximate the nonlinear function that represents the initially unmodeled dynamics is a separate issue not explicitly addressed in this thesis.

Dynamic programming and differential dynamic programming are reviewed to illustrate how learning methods relate to these classical approaches to optimal control design.

The aeroelastic oscillator is a two state mass-spring-dashpot system excited by a nonlinear lift force. Several learning control algorithms are applied to the aeroelastic oscillator to either regulate the mass position about a commanded point or to track a position reference trajectory; the advantages and disadvantages of these algorithms are discussed.

Thesis Supervisor:	Professor Wallace E. Vander Velde
Title:	Department of Aeronautics and Astronautics
Technical Supervisor:	Walter L. Baker
Title:	Senior Member of Technical Staff, C. S. Draper Laboratory

Acknowledgments

Love and appreciation to Mom and Dad (proper nouns?) for a lifetime of attention, support, encouragement, and no pressure — well, maybe a little pressure in the beginning. I don't think I can express how grateful I am, and I probably will not fully realize the importance of your commitment for many years, but I think you know how I feel.

To brother Bob... thanks for leaving me alone as an undergraduate so that I could learn to stop following in your footsteps. I've learned a lot from you and I hope that I've taught you something, too. Congratulations on finally getting out of this place. Remember, getting everything you want in life is more important than winning!

Heather — thanks for years of emotional stability. I wished we were closer, but the *friendship* was great; and now after $7\frac{1}{2}$ years, what a misguided thing to let happen. Have a wonderful life.

Walt Baker, I am obliged to you for the frequent guidance and technical assistance — even though we seem to think about everything a bit differently. You have been an excellent supervisor, football defensive lineman, and friend.

I appreciate the efforts of Professor Vander Velde, a very busy man, for carefully reviewing my work.

Additionally, Pete Millington enthusiastically offered to give me some advice if he ever has time... Pete, when we did talk, you always painted clear pictures Pete is cool of what was important, thanks.

I acknowledge Harry Klopf, Jim Morgan, and Leemon Baird for giving me a good place whence to start.

I thank Mario Santarelli and the rest of the *SimLab* team for opening the door (twice).

Equally significant were the distractions, I mean the friends, who seldom allowed the past six years to get boring. Noel – I miss Kathy's cooking as well as the competition of Nerf basketball games. I hope you get the aircraft you want. Thanks to everyone for making Draper a fun place to tool: Ruth, Dean, Tom(s), Kortney, Bill, Ching, Roger, Eugene, Steve, Torsten, Dan, Mitch, Dino, etc... and the same to the crowd who never understood why I was always at Draper: Jim, Kim, Jane, and Christine.

Jabin — 8.01 was so long ago! I never thought that you would get married before me; best wishes for the future. Thanks for all the random conversations and I hope you get a chance to put a hockey stick in orbit someday.

To *J3/DC* (Ojas, Ajit, Pat, Adrian, Hugo, Geno, and Trini), thanks for the camaraderie. King — I'll see you in Sydney. Dave, Deb, Sanjeev, Tamal, Art, Pete, Chuck, Jason, Brent, Steve — thanks for always being there.

This thesis was prepared at the Charles Stark Draper Laboratory, Inc. with support provided by the U. S. Air Force Wright Laboratory under Contract F33615-88-C-1740. Publication of this report does not constitute approval by C. S. Draper Laboratory or the sponsoring agency of the findings or conclusions contained herein. This thesis is published for the exchange and stimulation of ideas.

[illegible]

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Statement

The primary objective of this thesis is to incrementally synthesize a nonlinear optimal control law, through real-time, closed-loop interactions between the dynamic system, its environment, and a learning system, when substantial initial model uncertainty exists. The dynamic system is assumed to be nonlinear, time-invariant, and of known state dimension, but otherwise only inaccurately described by an a priori model. The problem, therefore, requires either explicit or implicit system identification. No disturbances, noise, or other time varying dynamics exist. The optimal control law is assumed to extremize an evaluation of the state trajectory and the control sequence, for any initial condition.

1.2 Thesis Overview

One objective of this thesis is to present an investigation of several approaches for incrementally synthesizing (on-line) an optimal control law. A second objective is to propose a *direct/indirect* framework, with which to distinguish learning algorithms. This framework subsumes concepts such as supervised/unsupervised learning and reinforcement learning, which are not directly related to control law synthesis. This thesis unifies a variety of concepts from control theory and behavioral science (where the learning process has been considered extensively) by presenting two different learning algorithms applied to the same control problem: the Associative Control Process (ACP) algorithm [14], which was initially developed to predict animal behavior, and Q learning [16], which derives from the mathematical theory of value iteration.

The aeroelastic oscillator (§2), a two-state physical system that exhibits interesting nonlinear dynamics, is used throughout the thesis to evaluate different control algorithms which incorporate learning. The algorithms that are explored in §3, §4, and §5 do not explicitly employ dynamic models of the system and, therefore, may be categorized as direct methods of learning an optimal control law. In contrast, §6 develops an indirect, model-based, approach to learning an optimal control law.

The Associative Control Process is a specific reinforcement learning algorithm applied to optimal control, and a description of the ACP in §3 introduces the concept of direct learning of an optimal control law. The ACP, which includes a prominent network architecture, originated in the studies of animal behavior. The Q learning algorithm, which derives from the mathematical theorems of policy iteration and value iteration, is a simple reinforcement learning rule independent of a network architecture and of biological origins. Interestingly, Klopf's ACP [14] may be reduced so that the resulting system accomplishes Watkins' Q learning algorithm [16]. Sutton's theory of the temporal difference methods [15], presented in §5, subsumes the ACP and Q learning algorithms by generalizing the reinforcement learning paradigm applied to optimal control.

Several control laws that are optimal with respect to various finite horizon cost functionals are derived in §6 to introduce the indirect approach to learning optimal controls. The structure of the control laws with and without learning augmentation appears for several cost functionals, to illustrate the manner in which learning may augment a fixed parameter control design.

Finally, dynamic programming (DP) and differential dynamic programming (DDP) are reviewed in Appendix A as classical, alternative methods for synthesizing optimal controls. DDP is not restricted to operations in a discrete input space and discrete output space. The DP and DDP algorithms are model-based and, therefore, learning may be introduced by explicitly improving the a priori model, resulting in an indirect learning optimal controller. However, neither DP nor DDP is easily implemented on-line. Additionally, DDP does not address the problem of synthesizing a control law over the full state space.

1.3 Concepts

The primary job of an automatic controller is to manipulate the inputs of a dynamic system so that the system's behavior satisfies the stability and performance specifications which constitute the control objective. The design of such a control law may involve numerous difficulties, including multivariable, nonlinear, and time varying dynamics, with many degrees of freedom. Further design challenges arise from the existence of model uncertainty, disturbances and noise, complex objective functions, operational constraints, and the possibility of component failure. An examination of the literature reveals that control design methodologies typically address a subset of these issues while making simplifying assumptions to satisfy the remainder – a norm to which this thesis conforms.

This section is intended to introduce the reader to some of the relevant issues by previewing concepts that appear throughout the thesis and are peculiar to learning systems and control law development. Additionally, this section motivates the importance of learning control research.

1.3.1 Optimal Control

This thesis examines methods for synthesizing optimal control laws, the objective of which is to extremize a scalar functional evaluation of the state trajectory and control history. The solution of an optimal control problem generally requires the solution of a constrained optimization problem; the calculus of variations and dynamic programming address this issue. However, an optimal control rule may be evaluated by these methods only if an accurate model of the dynamics is available.

In the absence of a complete and accurate a priori model, these approaches may be applied to a model that is derived through observed objective function evaluations and state transitions; this constitutes indirect learning control. Alternatively, in environments with substantial initial uncertainty, direct learning control can be considered to perform incremental dynamic programming without explicitly estimating a system model [1].

1.3.2 Fixed and Adjustable Control

Most control laws may be classified into one of two broad categories: fixed or adjustable. The constant parameters of fixed control designs are selected using an a priori model of the plant dynamics. As a result, stability robustness to modeling uncertainty is potentially traded against performance; the attainable performance of the closed-loop system is limited by the accuracy of the a priori description of the equations of motion and statistical descriptions of noise and disturbances. Adjustable control laws incorporate real-time data to reduce, either explicitly or implicitly, model uncertainty, with the intention of improving the closed-loop response.

An adjustable control design becomes necessary in environments where the controller must operate in uncertain conditions or when a fixed parameter control law that performs sufficiently well cannot be designed from the limited a priori information. The two main classes of adjustable control are adaptation and learning; both reduce the level of uncertainty by filtering empirical data that is gained experientially [2,3].

1.3.3 Adaptive Control

Noise and disturbances, which are present in all real systems, represent the unpredictable, time dependent features of the dynamics. Nonlinearities and coupled dynamics, which are predictable, spatial functions, constitute the remaining model errors.[1] Adaptive control techniques react to dynamics that appear to be time varying, while learning controllers progressively acquire spatially dependent knowledge about unmodeled dynamics. This fundamental difference in focus allows learning systems to avoid several deficiencies exhibited by adaptive algorithms in accommodating model errors. Whenever the plant operating condition changes, a new region of the nonlinear dynamics may be encountered. A memoryless adaptive control method must reactively adjust the control law parameters after observing the system behavior for the current condition, even if that operating condition has been previously experienced. The transient effects of frequently adapting control parameters may degrade closed-loop performance. A learning system, which utilizes memory to recall the appropriate control parameters as a function of the operating condition or state of the system, may be characterized as predictive rather than reactive.

Adaptive control exists in two flavors: direct and indirect. Indirect adaptive control methods calculate control actions from an explicit model of the system, which is enhanced with respect to the a priori description through a system identification procedure. Direct adaptive control methods modify the control system parameters without explicitly developing improvements in the initial system model.

[1] The term *spatial* implies a function that does not explicitly depend on time.

While direct adaptive techniques to perform regulation and tracking are well established, adaptive optimal controllers are primarily indirect.

1.3.4 Learning Control

A learning control system is characterized by the automatic synthesis of a functional mapping through the filtering of information acquired during previous real-time interactions with the plant and operating environment [2]. With the availability of additional experience, the mapping of appropriate control actions as a function of state or the mapping of unmodeled dynamics as a function of state and control, is incrementally improved. A learning system, which is implemented using a general function approximation scheme, may either augment traditional fixed or adaptive control designs, or may operate independently.

1.3.5 Generalization and Locality in Learning

Generalization in a parameterized, continuous mapping implies that each adjustable parameter influences the mapping over a region of non-zero measure [4]. The effect of generalization in function synthesis is to provide automatic interpolation between training data. If the plant dynamics are continuous functions of time and state, then the control law will also be continuous. Therefore, the validity of generalization follows directly from the continuity of the dynamics and the desired control law [2].

The concept of locality of learning is related to generalization, but differs in scope. Locality of learning implies that a change in any single adjustable parameter will only alter the mapped function over a localized region of the input space. For non-localized learning, extensive training in a restricted region of the input space, which might occur when a system is regulated about a trim condition, can corrupt the previously acquired mapping for other regions. Therefore, on-line learning for which training samples may be concentrated in a specific region of the input space, requires the locality attribute [2,3,4].

1.3.6 Supervised and Unsupervised Learning

Learning procedures may be distinguished as supervised or unsupervised according to the type of instructional information provided by the environment. A supervised learning controller requires both a teacher that provides the desired system response and the cost functional which depends on the system output error [5]. Supervised learning control systems often form the error signal by comparing measured system characteristics with predictions generated by an internal model. The supervised learning process evaluates how each adjustable parameter, within the internal model, influences the error signal.

The class of unsupervised control designs learns through a scalar evaluative feedback signal, such as the measure generated by a cost function, that is less informative than the gradient vector of the cost with respect to each adjustable parameter. This type of learning is also referred to as learning with a critic. The scalar evaluation which accrues from performing an action in a state does not indicate the cost to perform any other action in that state. Therefore, even in the case of only two possible actions, an evaluative learning signal contains significantly less information than the feedback required by a supervised learning algorithm [6].

1.3.7 Direct and Indirect Learning

The classifiers *direct* and *indirect* learning are borrowed from the concept of direct versus indirect adaptive control. Direct learning control implies the feedback loop that motivates the learning process is closed around system performance. Indirect learning control denotes that the learning loop is closed around the system model. Whereas in §3 – §5 the learning process is closed around system performance, in §6, the learning loop is closed around system model improvement, leaving the control law derivation and resulting system performance "open-loop."

Direct learning approaches to optimal control law synthesis, which employ reinforcement learning techniques, are not readily applicable to the reference model tracking problem. The adjustable parameters in a reinforcement learning method encode an evaluation of the cost to complete the objective. In a tracking environment, the command objective changes and future values may not be known. Therefore, the cost to complete the objective changes and the application of methods from §3 – §5 is restricted to regulation.

Indirect learning approaches to optimal control primarily employ supervised learning algorithms. In contrast, direct learning methods for optimal control law synthesis principally employ unsupervised learning algorithms.

1.3.8 Reinforcement Learning

Originally conceived in the study of animal learning phenomena, reinforcement learning is a type of unsupervised learning that responds to a performance measure feedback signal referred to as the reinforcement, which may represent a reward or a cost. At each discrete time step, the controller observes the current state, selects and applies an action, observes the subsequent state, and receives reinforcement; the control objective is to maximize the expected sum of discounted future reinforcement. The probability of choosing an action that yields a large discounted future reinforcement should be increased; actions that lead to small discounted future reinforcement should be selected less frequently [1]. Reinforcement learning methods often acquire successful action sequences by constructing two complimentary functions: a *policy* function maps the states into appropriate control actions and an *evaluation* function maps the states into expectations of the discounted future reinforcement.

As the study of connectionist learning methods has evolved from research in behavioral sciences to theories founded in the established disciplines of function approximation and optimization, reinforcement learning has been demonstrated to be a viable technique for solving some stable, nonlinear, optimal control problems [4,7].

Reinforcement learning addresses the credit assignment problem, which refers to the necessity of determining which actions in a sequence are "responsible" for an assessment of reinforcement. The problem is most severe in environments where evaluative feedback occurs infrequently. Additionally, the reinforcement learning process highlights the compromise between passive and active learning. Passive learning strategies are opportunistic and exploit any information that becomes available during the operation of the closed-loop system. In contrast, a control system using an active learning scheme explicitly seeks to gain information in regions where insufficient learning has occurred [4]. For on-line applications, that each action has an information collecting role implies a tradeoff between the expected gain of information, which is related to *future* performance, and the immediate reinforcement, which measures the *current* system performance [8].

1.3.9 BOXES

BOXES [8] is a simple implementation of a learning controller. The state space is discretized into disjoint regions, and the learning algorithm maintains an estimate of the appropriate control action for each region. Associated with any approach using a discrete input space is an exponential growth in the number of bins, as the state dimension or the number of quantization levels per state variable increases [3]. Therefore, quantization of the state space is seldom an efficient mapping technique and a learning algorithm that uses this strategy can generally only represent only a course approximation to a continuous control law. Although this lookup table technique facilitates some aspects of implementation, any parameterized function approximation scheme capable of representing continuous functions will be more efficient with respect to the necessary number of free parameters. Additionally, generalization is inherent to such continuous mappings [3]. A BOXES approach exhibits locality in learning, but does not generalize information across bin boundaries.

Chapter 2

The Aeroelastic Oscillator

2.1 General Description

A simple aeroelastic oscillator (AEO) may be modeled as a classical mass-spring-dashpot system with the addition of two external forces: an aerodynamic lift force and a control force (Figure 2.1). The mass, a rectangular block exposed to a steady wind, is constrained to translate in the direction normal to the vector of the incident wind and in the plane of the page in Figure 2.1. Specifications of the AEO plant are borrowed from Parkinson and Smith [9] as well as from Thompson and Stewart [10]. The low dimensionality of the dynamic state, which consists of the position $x(t)$ and the velocity of the mass, reduces the complexity of computer simulations and allows the system dynamics to be easily viewed in a two-dimensional phase plane. The AEO exhibits a combination of interesting nonlinear dynamics, generated by the nonlinear aerodynamic lift, and parameter uncertainty that constitute a good context in which to study learning as a method of incrementally synthesizing an optimal control law. The control objective may be either regulating the state near the origin of the phase plane or tracking a reference trajectory.

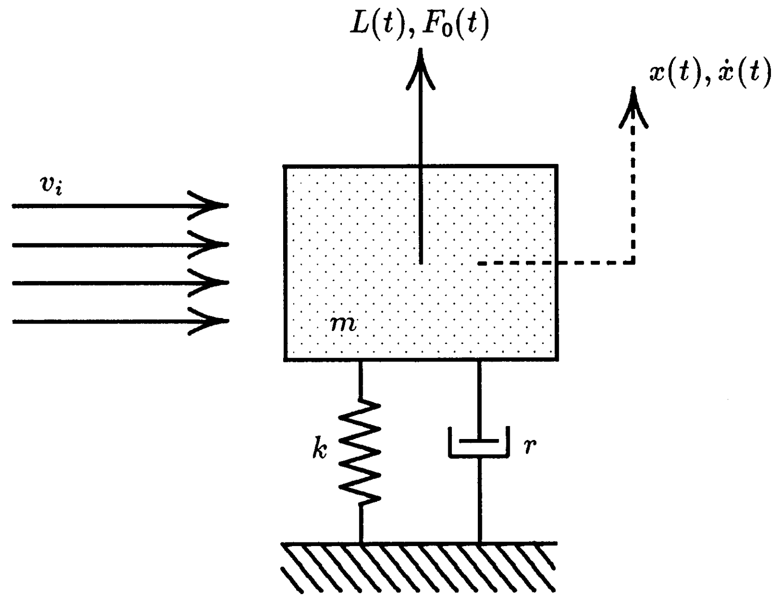

Figure 2.1. The aeroelastic oscillator.

2.2 The Equations of Motion

To investigate the AEO dynamics, the block is modeled as a point mass at which all forces act. The homogeneous equation of motion for the aeroelastic oscillator is a second-order, linear, differential equation with constant coefficients. This equation accurately represents the physical system for zero incident windspeed, in the range of block position and velocity for which the spring and dashpot respond linearly.

$$m\frac{d^2x}{dt^2} + r\frac{dx}{dt} + kx = 0 \tag{2.1}$$

Table 2.1. Physical variable definitions.

Physical Property	**Symbol**
Block Position	$x(t)$
Block Mass	m
Damping Coefficient	r
Spring Coefficient	k

For the undriven system, the block position may be described as a function of time by a weighted sum of exponentials whose powers are the roots of the characteristic equation.

$$x(t) = c_1 e^{s_1} + c_2 e^{s_2} \tag{2.2}$$

$$s_1, s_2 = \frac{-r \pm \sqrt{r^2 - 4mk}}{2m} = \frac{-r}{2m} \pm \frac{1}{2m}\sqrt{r^2 - 4mk} \tag{2.3}$$

$$k \geq 0 \quad \text{and} \quad r, m > 0 \Rightarrow \Re[s_1, s_2] \leq 0 \tag{2.4}$$

The condition that the dashpot coefficient is positive and the spring coefficient is non-negative, implies that the position and velocity transients will decay exponentially. This unforced system possesses a stable equilibrium at the origin of the phase plane.

The aerodynamic lift $L(t)$ and control force $F_0(t)$ constitute the driving component of the equation of motion. Including these forces, the equation of motion becomes a non-homogeneous, second-order, differential equation with constant coefficients.

$$m\frac{d^2x}{dt^2} + r\frac{dx}{dt} + kx = L + F_0 \tag{2.5}$$

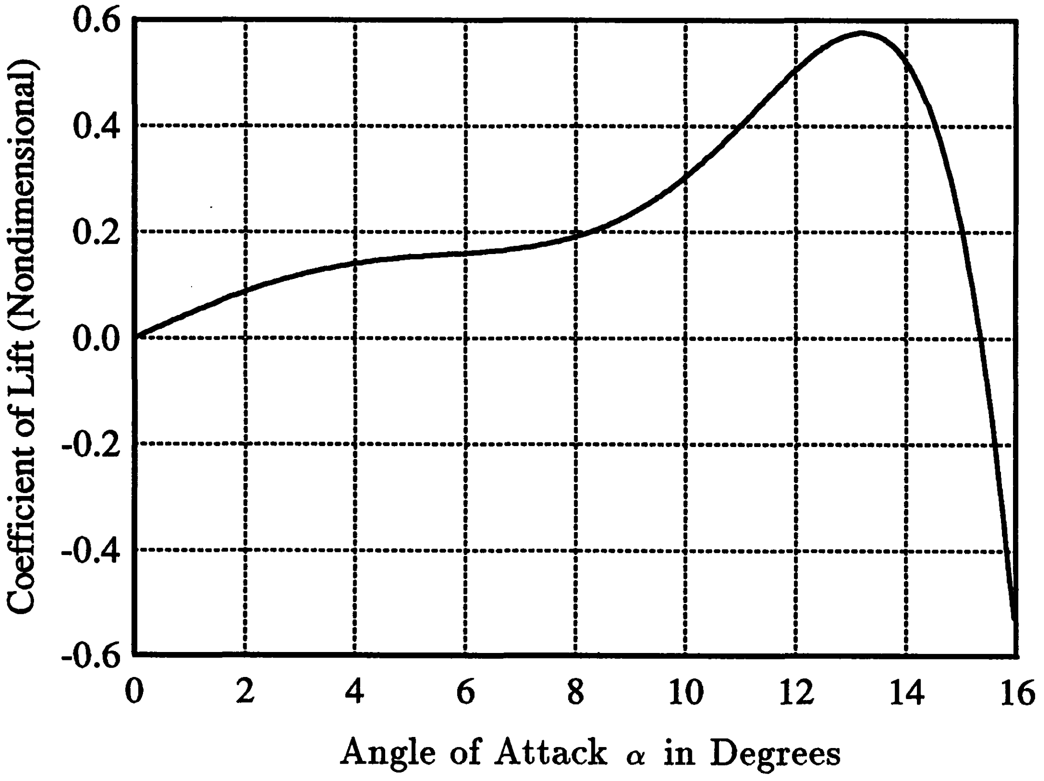

Figure 2.2. The aeroelastic oscillator nonlinear coefficient of lift.

The lift force is a nonlinear function of the effective angle of attack of the mass block with respect to the incident air flow. No current aerodynamic theory provides an analytic method for predicting the flow around an excited rectangular block. Therefore, the coefficient of lift is approximated, using empirical data, as a seventh-order polynomial in the tangent of the effective angle of attack α (Figure 2.2) [9,10]. This approximation to the empirical data is valid for a range of angles of attack near zero degrees, $|\alpha| \leq 18°$.

$$L = \frac{1}{2}\rho V^2 h l C_L \tag{2.6}$$

$$C_L = A_1\left(\frac{\dot{x}}{V}\right) - A_3\left(\frac{\dot{x}}{V}\right)^3 + A_5\left(\frac{\dot{x}}{V}\right)^5 - A_7\left(\frac{\dot{x}}{V}\right)^7 \tag{2.7}$$

$$tan(\alpha) = \frac{\dot{x}}{V} \tag{2.8}$$

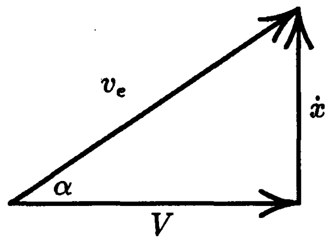

Figure 2.3. The total velocity vector v_e and the effective angle of attack α.

Table 2.2. Additional physical variable definitions.

Physical Property	Symbol
Density of Air	ρ
Velocity of Incident Wind	V
Area of Cross-section of Mass Block	hl
Coefficient of Lift	C_L

Following from the absence of even powers of $\frac{\dot{x}}{V}$ in the polynomial (2.7), the coefficient of lift is an odd symmetric function of the angle of attack, which, given the geometry of the AEO, seems physically intuitive. The definition of the effective angle of attack is most apparent from the perspective that the AEO is moving through a stationary fluid. The total velocity v_e equals the sum of two orthogonal components: the velocity associated with the oscillator as a unit translating through the medium (i.e. the incident flow V), and the velocity $\dot{x}$ associated with the mass block vibrating with respect to the fixed terminals of the spring and dashpot (Figure 2.3). This total velocity vector will form an effective angle of attack α with respect to the incident flow vector.

The dimensional equation of motion (2.5) can be nondimensionalized by dividing through by $k\,h$ and applying the rules listed in Table 2.3. The resulting equation of motion may be written as (2.9) or equivalently (2.10).

$$\frac{d^2X'}{d\tau^2} + 2\beta\frac{dX'}{d\tau} + X' = nA_1U\frac{dX'}{d\tau} - \left(\frac{nA_3}{U}\right)\left(\frac{dX'}{d\tau}\right)^3 + \left(\frac{nA_5}{U^3}\right)\left(\frac{dX'}{d\tau}\right)^5 - \left(\frac{nA_7}{U^5}\right)\left(\frac{dX'}{d\tau}\right)^7 + F' \tag{2.9}$$

$$\frac{d^2X'}{d\tau^2} + X' = nA_1\left[\left(U - \left(\frac{2\beta}{nA_1}\right)\right)\frac{dX'}{d\tau} - \left(\frac{A_3}{A_1U}\right)\left(\frac{dX'}{d\tau}\right)^3 + \left(\frac{A_5}{A_1U^3}\right)\left(\frac{dX'}{d\tau}\right)^5 - \left(\frac{A_7}{A_1U^5}\right)\left(\frac{dX'}{d\tau}\right)^7\right] + F' \tag{2.10}$$

The coefficient of lift is parameterized by the following four empirically determined constants: $A_1 = 2.69$, $A_3 = 168$, $A_5 = 6270$, $A_7 = 59900$ [9,10]. The other nondimensional system parameters were selected to provide interesting nonlinear dynamics: $n = 4.3 \cdot 10^{-4}$, $\beta = 1.0$, and $\frac{U}{U_c} = 1.6$. These parameters define $U_c = 1729.06$ and $U = 2766.5$, where the nondimensional critical windspeed U_c is defined in §2.3. The nondimensional time is expressed in radians.

Table 2.3. Required changes of variables.

New Variables	Relationships
Reduced displacement	$X' = \frac{x}{h}$
Mass parameter	$n = \frac{\rho h^2 l}{2m}$
Natural frequency	$\omega = \sqrt{\frac{k}{m}}$
Reduced incident windspeed	$U = \frac{V}{\omega h}$
Damping parameter	$\beta = \frac{r}{2m\omega}$
Reduced time (radians)	$\tau = \omega t$
Nondimensional Control Force	$F' = \frac{1}{kh} F_0$

The transformation from nondimensional parameters (n, β, and $\frac{U}{U_c}$) to dimensional parameters (ρ, h, l, m, V, r, and k) is not unique. Moreover, the nondimensional parameters that appear above will not transform to any physically realistic set of dimensional parameters. However, this set of nondimensional parameters creates fast dynamics which facilitates the analysis of learning techniques.

An additional change of variables scales the maximum amplitudes of the block's displacement and velocity to approximately unity in order of magnitude. The dynamics that are used throughout this thesis for experiments with the aeroelastic oscillator appear in (2.12).

$$X = \frac{X'}{1000} \qquad F = \frac{F'}{1000} \tag{2.11}$$

$$\frac{d^2X}{d\tau^2} + 2\beta\frac{dX}{d\tau} + X = \frac{1}{1000}\left[1000nA_1U\frac{dX}{d\tau} - \left(\frac{nA_3}{U}\right)\left(1000\frac{dX}{d\tau}\right)^3 + \left(\frac{nA_5}{U^3}\right)\left(1000\frac{dX}{d\tau}\right)^5 - \left(\frac{nA_7}{U^5}\right)\left(1000\frac{dX}{d\tau}\right)^7\right] + F \tag{2.12}$$

Equation (2.12) may be further rewritten as a pair of first-order differential equations in a state space realization. Although in the dimensional form $\dot{x} = \frac{dx}{dt}$, in the nondimensional form, $\dot{X} = \frac{dX}{d\tau}$.

$$x_1 = X \qquad x_2 = \frac{dX}{d\tau} \qquad \dot{x}_1 = x_2 \qquad \dot{x}_2 = \frac{d^2X}{d\tau^2} \tag{2.13}$$

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}0 & 1\\ -1 & nA_1U - 2\beta\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}0\\ 1\end{bmatrix}F + \begin{bmatrix}0\\ f(x_2)\end{bmatrix} \tag{2.14a}$$

$$f(x_2) = \frac{1}{1000}\left[-\frac{nA_3}{U}(1000x_2)^3 + \frac{nA_5}{U^3}(1000x_2)^5 - \frac{nA_7}{U^5}(1000x_2)^7\right] \tag{2.14b}$$

2.3 The Open-loop Dynamics

The reduced critical windspeed U_c, which depends on the nondimensional mass parameter, the damping parameter, and the first-order coefficient in the coefficient of lift polynomial, is the value of the incident windspeed at which the negative linear aerodynamic damping exceeds the positive structural damping.[1]

$$U_c = \frac{2\beta}{nA_1} \tag{2.15}$$

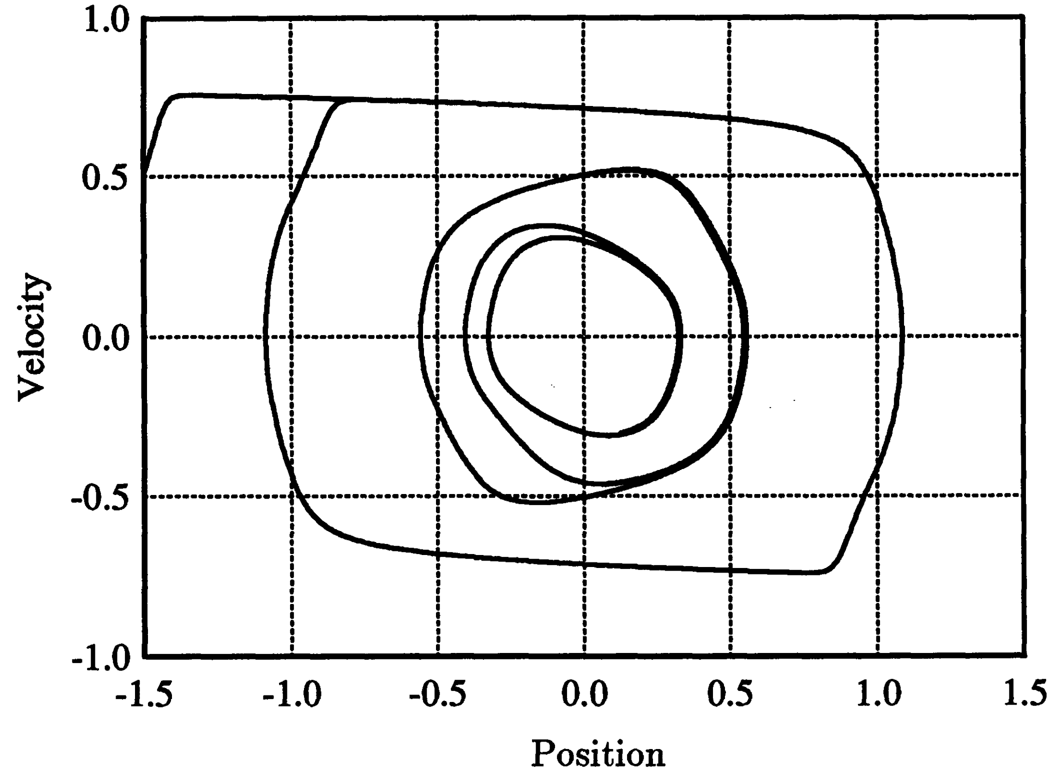

Figure 2.4. The aeroelastic oscillator open-loop dynamics. An outer stable limit cycle surrounds an unstable limit cycle that in this picture decays inward to an inner stable limit cycle.

The nature of the open-loop dynamics is strongly dependent on the ratio of the reduced incident windspeed to the reduced critical windspeed. At values of the incident windspeed below the critical value, the focus of the phase plane is stable

[1] The term *reduced* is synonymous with nondimensional.

and the state of the oscillator will return to the origin from any perturbed initial condition. For windspeeds greater than the critical value, the focus of the two dimensional state space is locally unstable; the system will oscillate, following a stable limit cycle clockwise around the phase plane. The aeroelastic oscillator is globally stable, in a bounded sense, for all U.[2] The existence of global stability is suggested by the coefficient of lift curve (Figure 2.2); the coefficient of lift curve predicts zero lift ($C_L = 0$) for $\alpha = \pm 15.3^\circ$ and a restoring lift force for larger $|\alpha|$. That the aeroelastic oscillator is globally open-loop stable eliminates the necessity for a feedback loop to provide nominal stability during learning experiments. For incident windspeeds greater than U_c, a limit cycle is generated at a stable Hopf bifurcation. In this simplest form of dynamic bifurcation, a stable focus bifurcates into an unstable focus surrounded by a stable limit cycle under the variation of a single independent parameter, U_c. For a range of incident wind velocity, two stable limit cycles, separated by an unstable limit cycle, characterize the dynamics (Figure 2.4). Figure 2.4 was produced by a 200Hz simulation in continuous time of the AEO equations of motion, using a fourth-order Runge-Kutta integration algorithm. An analysis of the open-loop dynamics appears in Appendix B.

2.4 Benchmark Controllers

A simulation of the AEO equations of motion in continuous time was implemented in the *NetSim* environment. *NetSim* is a general purpose simulation and

[2] Each state trajectory is a member of L_∞ (i.e. $||\underline{x}(t)||_\infty$ is finite) for all perturbations δ with bounded Euclidean norms, $||\delta||_2$.

design software package developed at the C. S. Draper Laboratory [11]. Ten *NetSim* cycles were completed for each nondimensional time unit while the equations of motion were integrated over twenty steps using a fourth-order Runge Kutta algorithm for each *NetSim* cycle.

Two simple control laws, based on a linearization of the AEO equations of motion, will serve as benchmarks for the learning controllers of §3, §4 and §5.

2.4.1 Linear Dynamics

From (2.14a), the linear dynamics about the origin may be expressed by (2.16) where A and $\underline{B}$ are given in (2.17).

$$\dot{\underline{x}}(\tau) = A\underline{x}(\tau) + \underline{B}u(\tau) \tag{2.16}$$

$$A = \begin{bmatrix} 0 & 1 \\ -1 & nA_1U - 2\beta \end{bmatrix} \qquad \underline{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2.17}$$

This linearization may be derived by defining a set of perturbation variables, $\underline{x}(\tau) = \underline{x}_0 + \underline{\delta x}(\tau)$ and $u(\tau) = u_0 + \delta u(\tau)$, which must satisfy the differential equations. Notice that $\dot{\underline{\delta x}}(\tau) = \dot{\underline{x}}(\tau)$. The expansion of $\dot{\underline{\delta x}}(\tau)$ in a Taylor series about $(\underline{x}_0, u_0)$ yields (2.18).

$$\begin{aligned} \dot{\underline{\delta x}}(\tau) &= \underline{f}\left[\underline{x}_0 + \underline{\delta x}(\tau), u_0 + \delta u(\tau)\right] \\ &= \underline{f}(\underline{x}_0, u_0) + \left.\frac{\partial \underline{f}}{\partial \underline{x}}\right|_{\underline{x}_0, u_0} \underline{\delta x}(\tau) + \left.\frac{\partial \underline{f}}{\partial u}\right|_{\underline{x}_0, u_0} \delta u(\tau) + \cdots \end{aligned} \tag{2.18}$$

If the pair $(\underline{x}_0, u_0)$ represents an equilibrium of the dynamics, then $\underline{f}(\underline{x}_0, u_0) = 0$ by definition. Equation (2.16) is achieved by discarding the nonlinear terms of

(2.18) and applying (2.19), where A and $\underline{B}$ are the Jacobian matrices.

$$A = \left.\frac{\partial \underline{f}}{\partial \underline{x}}\right|_{\underline{x}_0, u_0} \qquad \underline{B} = \left.\frac{\partial \underline{f}}{\partial u}\right|_{\underline{x}_0, u_0} \tag{2.19}$$

2.4.2 The Linear Quadratic Regulator

The LQR solution minimizes a cost functional J that is an infinite time horizon integral of a quadratic expression in state and control. The system dynamics must be linear. The optimal control is given by (2.21)

$$J = \int_0^\infty \left[\underline{x}(\tau)^T \underline{x}(\tau) + u^2(\tau)\right] d\tau \tag{2.20}$$

$$u'(\tau) = -\underline{G}^T \underline{x}(\tau) \tag{2.21}$$

$$\underline{G} = \begin{bmatrix} 0.4142 \\ 3.0079 \end{bmatrix} \tag{2.22}$$

The actuators which apply the control force to the AEO are assumed to saturate at ± 0.5 nondimensional force units. Therefore, the control law tested in this section was written as

$$u(\tau) = f\left(-\begin{bmatrix}0.4142 & 3.0079\end{bmatrix} \underline{x}(\tau)\right). \tag{2.23}$$

$$f(x) = \begin{cases} 0.5, & \text{if } x > 0.5 \\ -0.5, & \text{if } x < -0.5 \\ x, & \text{otherwise.} \end{cases} \tag{2.24}$$

The state trajectory which resulted from applying the control law (2.23) to the AEO, for the initial conditions $\{-1.0, 0.5\}$, appears in Figure 2.5. The controller applied the maximum force until the state approached the origin, where the dynamics are nearly linear (Figure 2.6). Therefore, the presence of the nonlinearity in the dynamics did not strongly influence the performance of this control law.

If the linear dynamics were modeled perfectly (as above) and the magnitude of the control were not limited, the LQR solution would perform extremely well. Model uncertainty was introduced into the a priori model by designing the LQR controller assuming the open-loop poles were $0.2 \pm 1.8j$.

$$A' = \begin{bmatrix} 0 & 1 \\ -3.28 & 0.4 \end{bmatrix} \tag{2.26}$$

The LQR solution of (2.20) using A' is $\underline{G}^T = [0.1491, 1.6075]$. This control law applied to the AEO, when the magnitude of the applied force was limited at 0.5, produced the results shown in Figures 2.7 and 2.8. The closed-loop system was significantly under-damped.

2.4.3 Bang-bang Controller

The bang-bang controller was restricted to two control actions, a maximum positive force (0.5 nondimensional units) and a maximum negative force (-0.5); this limitation will also be imposed on the initial direct learning experiments. The control law is derived from the LQR solution and is non-optimal for the AEO system. In the half of the state space where the LQR solution specifies a positive force, the bang-bang control law (2.25) applies the maximum positive force. Similarly, in the half of the state space where the LQR solution specifies a negative force, the bang-bang control law applies the maximum negative force. The switching line which

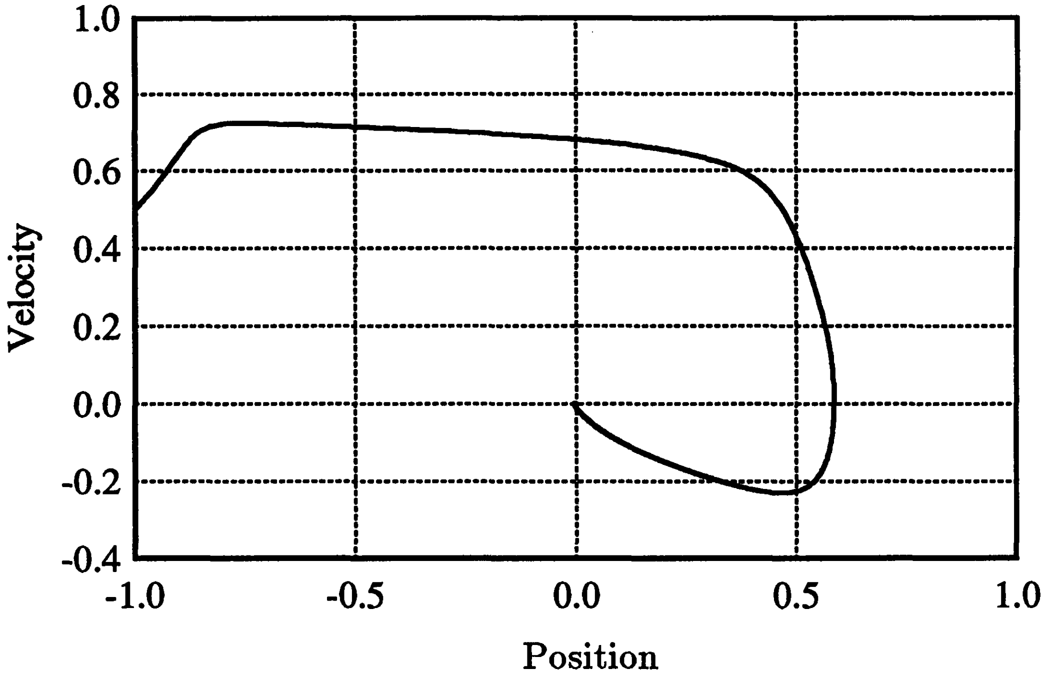

Figure 2.5. The AEO state trajectory achieved by a magnitude limited LQR control law.

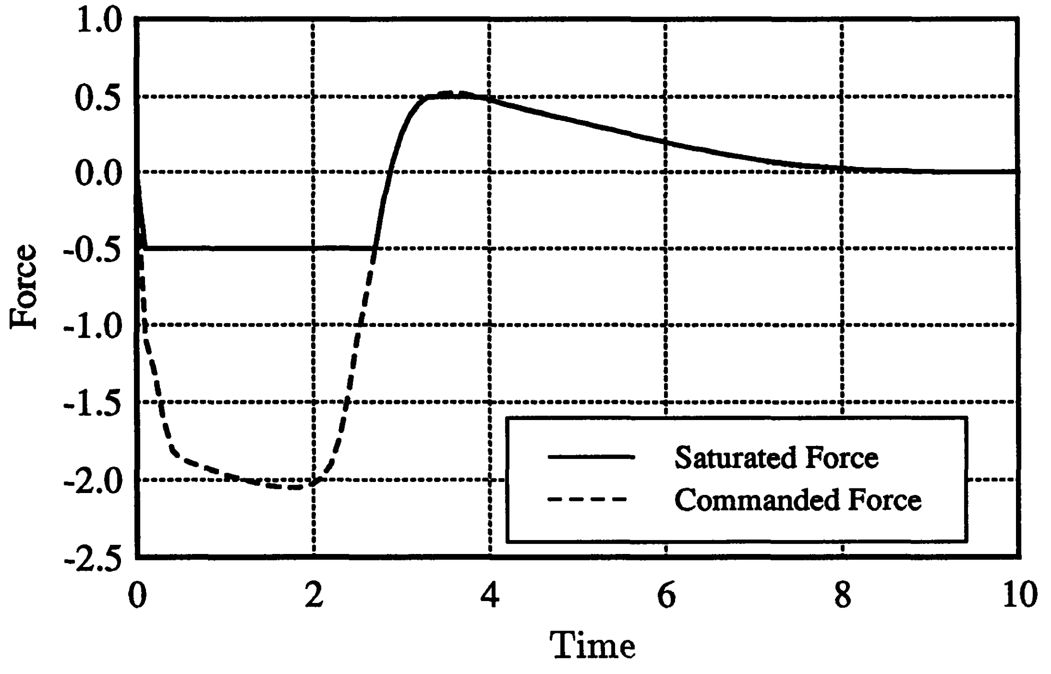

Figure 2.6. The LQR control history and the limited force which yields Figure 2.5.

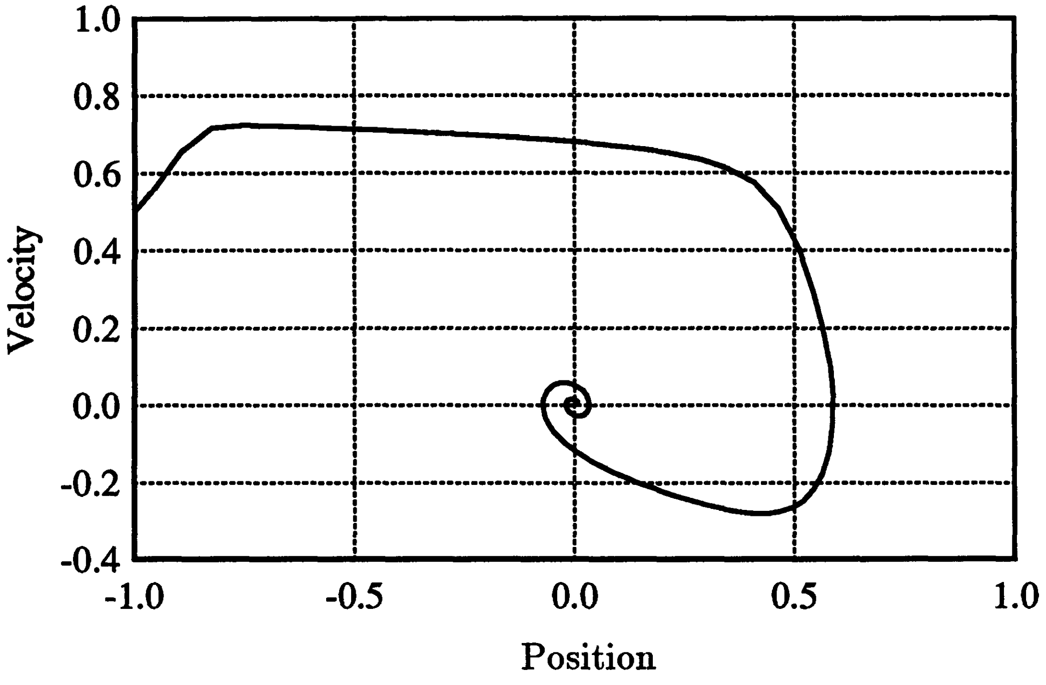

Figure 2.7. The AEO state trajectory achieved by a LQR solution which was derived from a model with error in the linear dynamics. $\underline{x}_0 = \{-1.0, 0.5\}$.

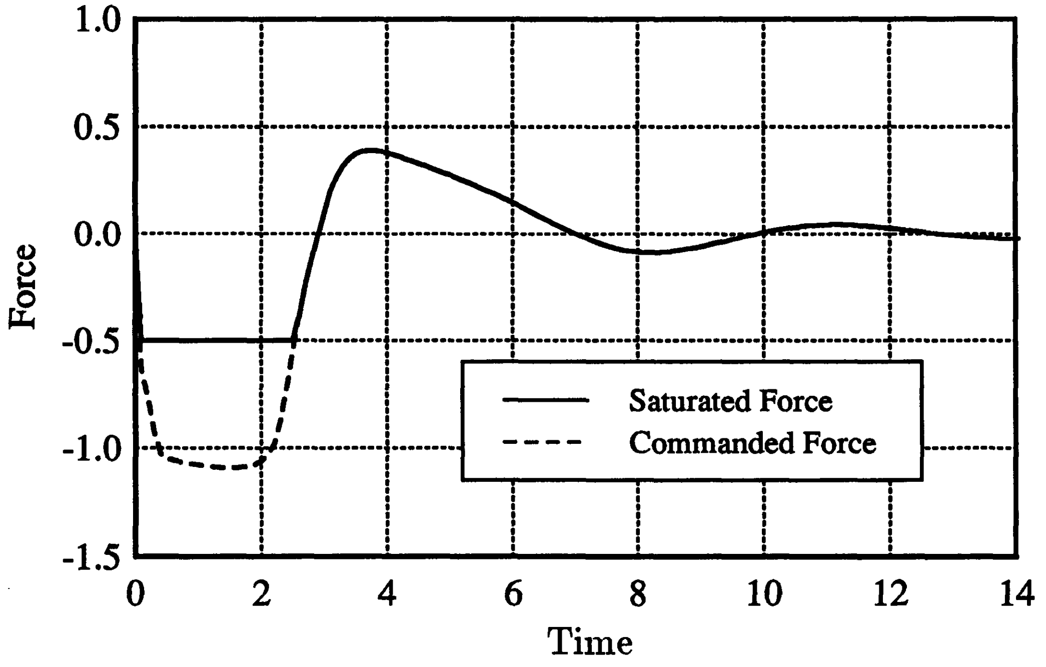

Figure 2.8. The control history for the LQR solution which was derived from a model of the AEO with error in the linear dynamics.

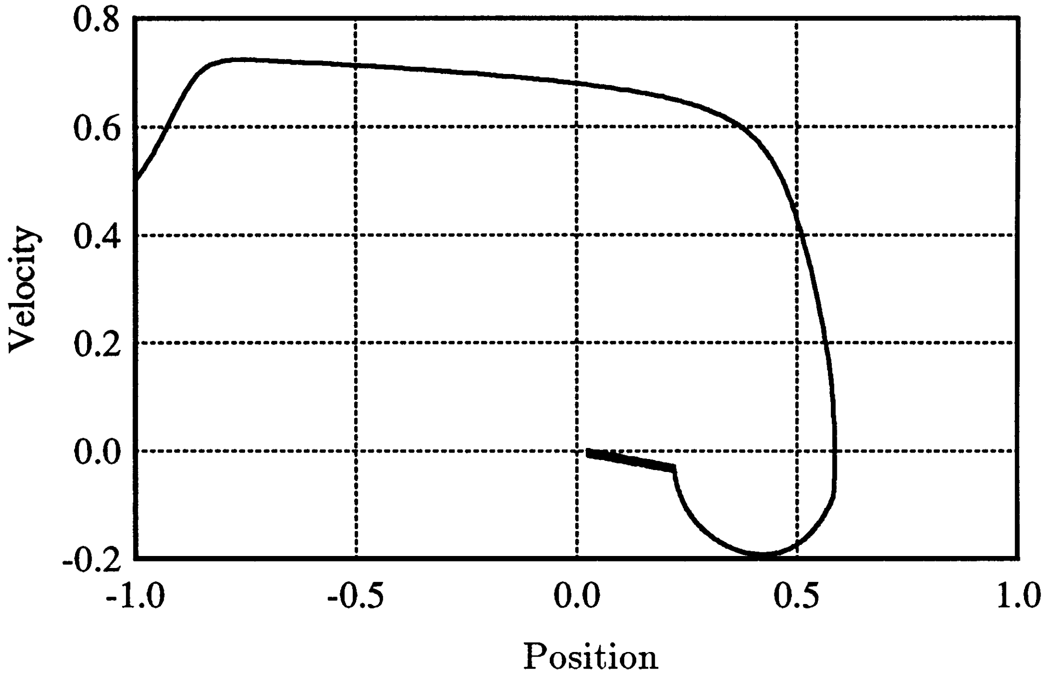

Figure 2.9. The AEO state trajectory achieved by a bang-bang control law derived from the LQR solution.

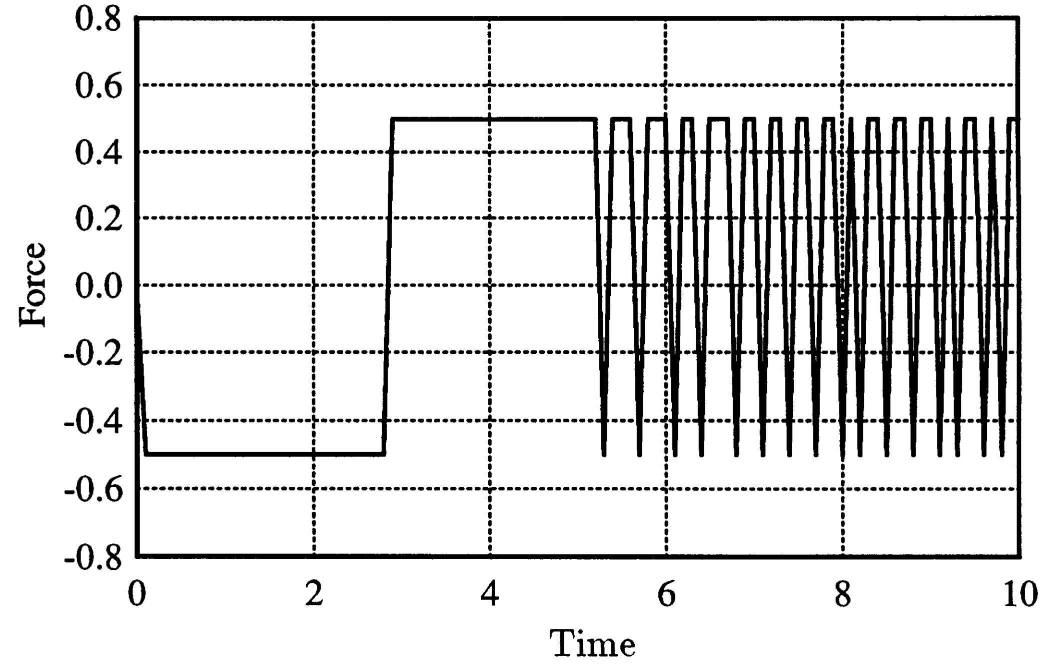

Figure 2.10. The control history of a bang-bang controller derived from the LQR solution, which yields Figure 2.9.

divides the state space passes through the origin with slope -0.138; this is the line of zero force in the LQR solution.

$$u(\tau) = \begin{cases} 0.5, & \text{if } -\underline{G}^T \underline{x}(\tau) > 0 \\ -0.5, & \text{otherwise.} \end{cases} \tag{2.25}$$

The result of applying this bang-bang control law to the AEO with initial conditions $\{-1.0, 0.5\}$ appears in Figure 2.7. The trajectory initially traces the trajectory in Figure 2.5 because the LQR solution was saturated at -0.5. However, the trajectory slowly converges toward the origin along the line which divides the positive and negative control regions, while rapidly alternating between exerting the maximum positive force and maximum negative force (Figure 2.8). Generally, this would represent unacceptable performance. The bang-bang control law represents a two-action, linear control policy and will serve as a non-optimal benchmark with which to compare the direct learning control laws. The optimal two-action control law cannot be written from only a brief inspection of the nonlinear dynamics.

Chapter 3

The Associative Control Process

The Associative Control Process (ACP) network [12,14] models certain fundamental aspects of the animal nervous system, accounting for numerous classical and instrumental conditioning phenomena.[1] The original ACP network was intended to model limbic system, hypothalamic, and sensorimotor function as well as to provide a general framework within which to relate animal learning psychology and control theory. Through real-time, closed-loop, goal seeking interactions between the learning system and the environment, the ACP algorithm can achieve solutions to spatial and temporal credit assignment problems. This capability suggests that the ACP algorithm, which accomplishes reinforcement or self-supervised learning, may offer solutions to difficult optimal control problems.

[1] Animal learning phenomena are investigated through two classes of laboratory conditioning procedures. *Classical conditioning* is an open-loop process in which the experience of the animal is independent of the behavior of the animal. The experience of the animal in closed-loop *instrumental conditioning* or *operant conditioning* experiments is contingent on the animal's behavior [12].

This chapter constitutes a thorough description of the ACP network, viewed from the perspective of applying the architecture and process as a controller for dynamic systems.[2] A detailed description of the architecture and functionality of the original ACP network (§3.1) serves as a foundation from which to describe two levels of modification, intended to improve the applicability of the Associative Control Process to optimal control problems. Initial modifications to the original ACP specifications retain a two-layer network structure (§3.2); several difficulties in this *modified* ACP motivate the development of a single layer architecture. A single layer formulation of the ACP network abandons the biologically motivated network structure while, preserving the mathematical basis of the modified ACP (§3.4). This minimal representation of an Associative Control Process performs an incremental value-iteration procedure similar to Q learning and is guaranteed to converge to the optimal policy in the infinite horizon optimal control problem under certain conditions [13]. This chapter concludes with a summary of the application of the modified and single layer ACP methods to the regulation of the aeroelastic oscillator (§3.5 and §3.6).

3.1 The Original Associative Control Process

The definition of the original ACP is derived from Klopf [12], Klopf, Morgan, and Weaver [14], as well as Baird and Klopf [13]. Although originally introduced in the literature as a model to predict a variety of animal learning results from classical

[2] This context is in contrast to the perspective that an ACP network models aspects of biological systems.

and instrumental conditioning experiments, a recast version of the ACP network has been shown to be capable of learning to optimally control any non-absorbing, finite-state, finite-action, discrete time Markov decision process [13]. Although the original form of the ACP may be incompatible with infinite time horizon optimal control problems, as an introduction to the ACP derivatives, the original ACP appears here with an accent toward applying the learning system to the optimal control of dynamic systems. Where appropriate, analogies to animal learning results motivate the presence of those features of the original ACP architecture which emanate from a biological origin. Although the output and learning equations are central in formalizing the ACP system, to eliminate ambiguities concerning the interconnection and functionality of network elements, substantial textual description of rules is required.

The ACP network consists of five distinct elements: *acquired drive sensors*, *motor centers*, *reinforcement centers*, *primary drive sensors*, and *effectors* (Figure 3.1). In the classical conditioning nomenclature, the acquired drive sensors represent the conditioned stimuli; in the context of a control problem, the acquired drive sensors encode the sensor measurements and will be used to identify the discrete dynamic state. The ACP requires an interface with the environment that contains a finite set of states. Therefore, for the application of the ACP to a control problem, the state space of a dynamic system is quantized into a set of m disjoint, non-uniform bins which fill the entire state space.[3] The ACP learning system operates in discrete time. At any stage in discrete time, the state of the dynamic system

[3] A sufficient condition is for the bins to fill the entire operational envelope, i.e. the region of the state space that the state may enter.

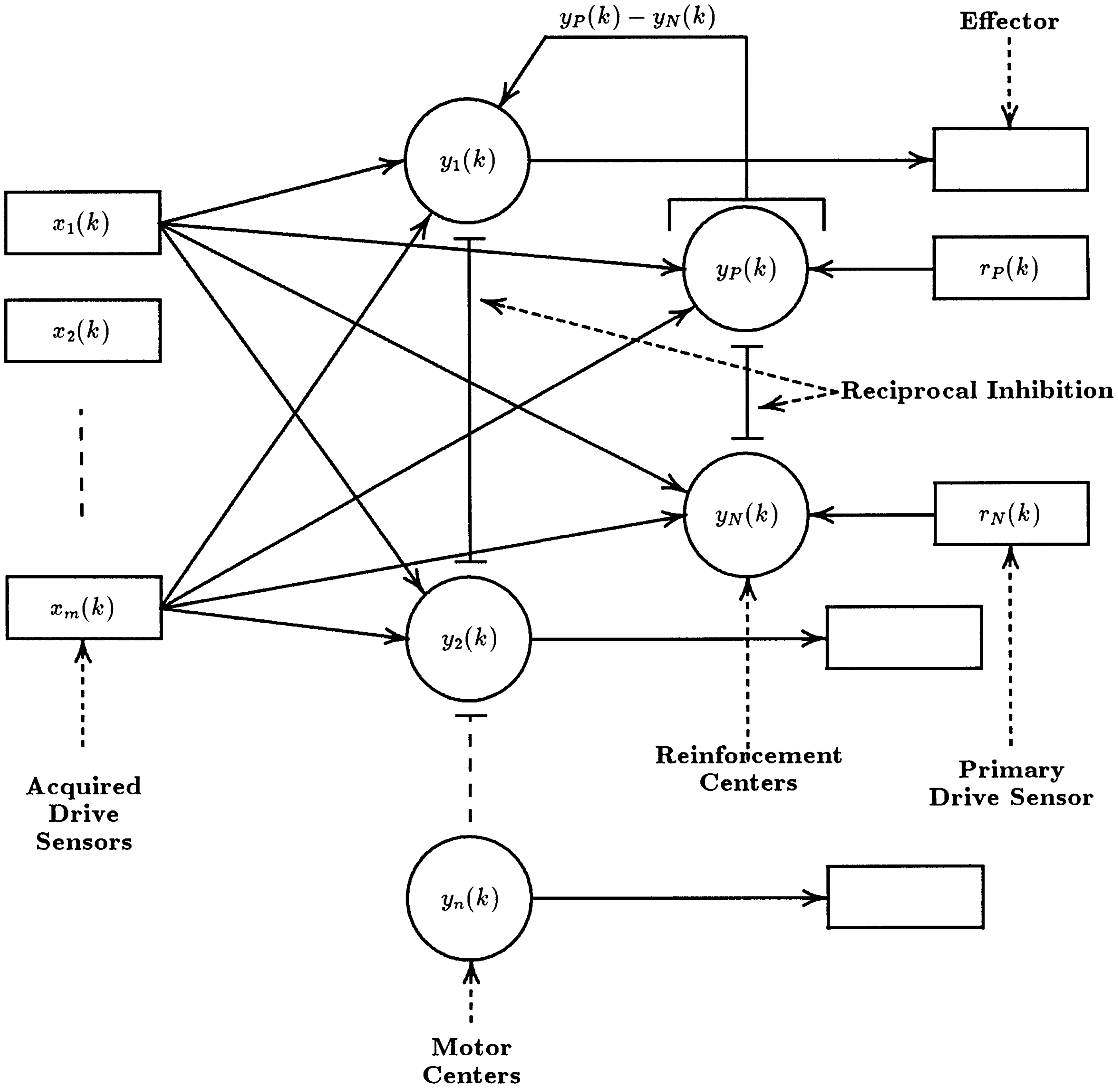

Figure 3.1. The ACP network architecture.

will lie within exactly one bin, with which a single acquired drive sensor is uniquely associated. The current output of the i^{th} acquired drive sensor, $x_i(k)$, will be either unity or zero, and exactly one acquired drive sensor will have unity output at each time step.[4] The vector of m acquired drive signals, $\underline{x}(k)$, should not be confused with the vector of state variables, the length of which equals the dimension of the state space.

A motor center and effector pair exists for each discrete network output.[5] The motor centers collectively determine the network's immediate action and, therefore, the set of n motor centers operate as a single policy center. In animal learning research, the effector encodes an action which the animal may choose to perform (e.g. to turn left). As a component of a control system, each effector represents a discrete control produced by an actuator (e.g. apply a force of 10.0 units). The output of a motor center is a real number and should not be confused with the output of the ACP network, which is an action performed by an effector.

The output of the j^{th} motor center, $y_j(k)$, equals the evaluation of a nonlinear, threshold-saturation function (Figure 3.2) applied to the weighted sum of the acquired drive sensor inputs.

$$y_j(k) = f_n\left[\sum_{i=1}^{m} \left(W_{ij}^{+}(k) + W_{ij}^{-}(k)\right) x_i(k)\right] \tag{3.1}$$

$$f_n(x) = \begin{cases} 0 & \text{if } x \leq \theta \\ 1 & \text{if } x \geq 1 \\ x & \text{otherwise} \end{cases} \tag{3.2}$$

[4] This condition is not necessary in the application of the ACP to predict animal learning results.

[5] Recall that the ACP network output must be a member of a finite set of control actions.

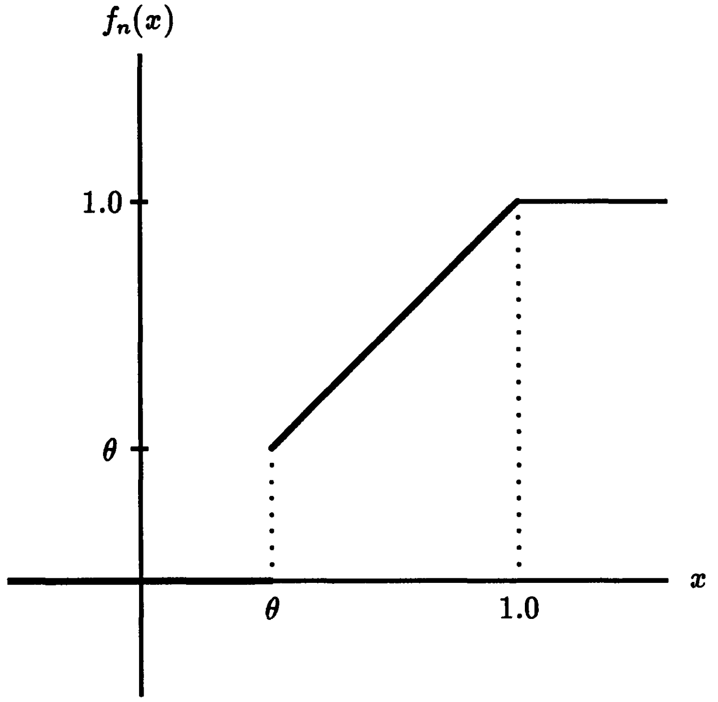

Figure 3.2. The output equation nonlinearity, (3.2).

The threshold θ is a non-negative constant less than unity. Justification for the presence of the output nonlinearity follows directly from the view that a neuronal output measures the frequency of firing of the neuron, when that frequency exceeds the neuronal threshold.[6] Negative values of $y_j(t)$, representing negative frequencies of firing, are not physically realizable.

The motor center output equation (3.1) introduces two weights from each acquired drive sensor to each motor center: a positive *excitatory* weight $W_{ij}^{+}(k)$ and a negative *inhibitory* weight $W_{ij}^{-}(k)$. Biological evidence motivates the presence of distinct excitatory and inhibitory weights that encode attraction and avoidance

[6] The term *neuronal output* refers to the output of a motor center or a reinforcement center.

behaviors, respectively, for each state-action pair. The time dependence of the weights is explicitly shown to indicate that the weights change with time through learning; the notation does not imply that functions of time are determined for each weight.

Reciprocal inhibition, the process of comparing several neuronal outputs and suppressing all except the largest to zero, prevents the motor centers that are not responsible for the current action from undergoing weight changes. Reciprocal inhibition is defined by (3.3). The motor center $j_{max}(k)$ which wins reciprocal inhibition among the m motor center outputs at time k will be referred to as the currently active motor center; $j_{max}(k-a)$, therefore, is the motor center that was active a time steps prior to the present, and $y_{j_{max}(k-a)}(k)$ is the current output of the motor center that was active a time steps prior to the present.

$$\begin{gathered} j_{max}(k) = j \\ \text{such that for all} \quad l \in \{1, 2, \ldots n\} \quad \text{and} \quad l \neq j \\ y_l(k) < y_j(k) \end{gathered} \tag{3.3}$$

The current network action corresponds to the effector associated with the single motor center which has a non-zero output after reciprocal inhibition. Potentially, multiple motor centers may have equally large outputs. In this case, reciprocal inhibition for the original ACP is defined such that no motor center will be active, no control action will be effected, and no learning will occur.

The ACP architecture contains two primary drive sensors, differentiated by the labels *positive* and *negative*. The primary drive sensors provide external evaluations of the network's performance in the form of non-negative *reinforcement* signals; the positive primary drive sensor measures reward while the negative primary drive sensor measures cost or punishment. In the language of classical conditioning, these evaluations are collectively labeled the unconditioned stimuli. In the optimal control framework, the reward equals zero and the punishment represents an evaluation of the cost functional which the control is attempting to minimize.

The ACP architecture also contains two reinforcement centers which are identified as *positive* and *negative* and which yield non-negative outputs. Each reinforcement center learns to predict the occurrence of the corresponding external reinforcement and consequently serves as a source of internal reinforcement, allowing learning to continue in the absence of frequent external reinforcement. In this way, the two reinforcement centers direct the motor centers, through learning, to select actions such that the state approaches reward and avoids cost.

Each motor center *facilitates* a pair of excitatory and inhibitory weights from each acquired drive sensor to each reinforcement center. The output of the positive reinforcement center, prior to reciprocal inhibition between the two reinforcement centers, is the sum of the positive external reinforcement $r_P(k)$ and the weighted sum of the acquired drive sensor inputs. The appropriate set of weights from the acquired drive sensors to the reinforcement center corresponds to the currently active motor center. Therefore, calculation of the outputs of the reinforcement centers requires prior determination of $j_{max}(k)$.

$$y_P(k) = f_n \left[r_P(k) + \sum_{i=1}^{m} \left(W^{+}_{Pij_{max}(k)}(k) + W^{-}_{Pij_{max}(k)}(k) \right) x_i(k) \right] \tag{3.4}$$

The output of the negative reinforcement center $y_N(k)$ is calculated similarly, using the negative external reinforcement $r_N(k)$.

$$y_N(k) = f_n \left[r_N(k) + \sum_{i=1}^{m} \left(W^{+}_{Nij_{max}(k)}(k) + W^{-}_{Nij_{max}(k)}(k) \right) x_i(k) \right] \tag{3.5}$$

The ACP learning mechanism improves the stored policy and the predictions of future reinforcements by adjusting the weights which connect the acquired drive sensors to the motor and reinforcement centers. If the j^{th} motor center is active with the i^{th} acquired drive sensor, then the reinforcement center weights $W^{\pm}_{Pij}(k)$ and $W^{\pm}_{Nij}(k)$ are eligible to change for τ subsequent time steps. The motor center weights $W^{\pm}_{ij}(k)$ are eligible to change only during the current time step.[7] Moreover, all weights for other state-action pairs will remain constant this time step.

The impetus for motor center learning is the difference, after reciprocal inhibition, between the outputs of the positive and negative reinforcement centers. The following equations define the incremental changes in the motor center weights, where the constants c_a and c_b are non-negative. The nonlinear function f_s in (3.6), defined by (3.9), requires that only positive changes in presynaptic activity, $\Delta x_i(k)$, stimulate weight changes.

$$\Delta W^{\pm}_{ij}(k) = \begin{cases} c(k) \left| W^{\pm}_{ij}(k) \right| f_s\left(\Delta x_i(k)\right) \left[y_P(k) - y_N(k) - y_j(k) \right] & \text{if } j = j_{max}(k) \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

$$c(k) = c_a + c_b \left| y_P(k) - y_N(k) \right| \tag{3.7}$$

[7] The weights of both positive and negative reinforcement centers are eligible for change even though both reinforcement centers cannot win reciprocal inhibition. In contrast, only the motor center that wins reciprocal inhibition can experience weight changes. If no motor center is currently active, however, no learning occurs in either the motor centers or the reinforcement centers.

$$\Delta x_i(k) = x_i(k) - x_i(k-1) \tag{3.8}$$

$$f_s(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

The learning process is divided into temporal intervals referred to as *trials*; the weight changes, which are calculated at each time step, are accumulated throughout the trial and implemented at the end of the trial. The symbols k_0 and k_f in (3.10) represent the times before and after a trial, respectively. A lower bound on the magnitude of every weight maintains each excitatory weight always positive and each inhibitory weight always negative (Figures 3.3 and 3.4). The constant α in (3.11) is a positive network parameter.

$$W_{ij}^{+}(k_f) = f_{W^+}\left[W_{ij}^{+}(k_0) + \sum_{k=k_0}^{k_f} \Delta W_{ij}^{+}(k)\right] \tag{3.10a}$$

$$W_{ij}^{-}(k_f) = f_{W^-}\left[W_{ij}^{-}(k_0) + \sum_{k=k_0}^{k_f} \Delta W_{ij}^{-}(k)\right] \tag{3.10b}$$

$$f_{W^+}(x) = \begin{cases} \alpha & \text{if } x < \alpha \\ x & \text{otherwise} \end{cases} \tag{3.11a}$$

$$f_{W^-}(x) = \begin{cases} -\alpha & \text{if } x > -\alpha \\ x & \text{otherwise} \end{cases} \tag{3.11b}$$

Equations (3.12) through (3.15) define the Drive-Reinforcement (DR) learning mechanism used in the positive reinforcement center; negative reinforcement center learning follows directly [12,14]. Drive-Reinforcement learning, which is a flavor of temporal difference learning [15], changes eligible connection weights as a function of the correlation between earlier changes in input signals and later changes in

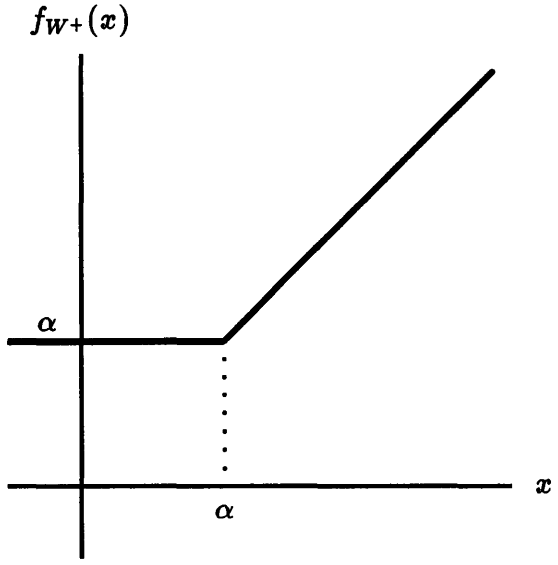

Figure 3.3. The lower bound on excitatory weights, (3.11a).

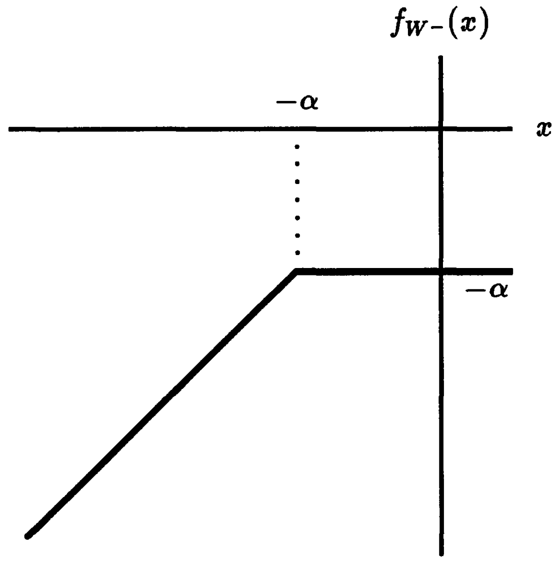

Figure 3.4. The upper bound on inhibitory weights, (3.11b).

output signals. The constants τ (which in animal learning represents the longest interstimulus interval over which delay conditioning is effective) and $c_1, c_2, \ldots c_\tau$

are non-negative. Whereas τ may be experimentally deduced for animal learning problems, selection of an appropriate value of τ in a control problem typically requires experimentation with the particular application. The incremental change in the weight associated with a reinforcement center connection depends on four terms. The correlation between the current change in postsynaptic activity, $\Delta y_P(k)$, and a previous change in presynaptic activity, $\Delta x_i(k-a)$, is scaled by a learning rate constant c_a and the absolute value of the weight of the connection at the time of the change in presynaptic activity.

$$\Delta W^{\pm}_{Pij}(k) = \Delta y_P(k) \sum_{a=1}^{\tau} c_a \left| W^{\pm}_{Pij}(k-a) \right| f_s \left(\Delta x_{ij}(k-a) \right) \tag{3.12}$$

$$\Delta y_P(k) = y_P(k) - y_P(k-1) \tag{3.13}$$

$$\Delta x_{ij}(k-a) = \begin{cases} x_i(k-a) - x_i(k-a-1) & \text{if } j = j_{max}(k-a) \\ 0 & \text{otherwise} \end{cases} \tag{3.14}$$

$$W^{+}_{Pij}(k_f) = f_{W^+} \left[W^{+}_{Pij}(k_0) + \sum_{k=k_0}^{k_f} \Delta W^{+}_{Pij}(k) \right] \tag{3.15a}$$

$$W^{-}_{Pij}(k_f) = f_{W^-} \left[W^{-}_{Pij}(k_0) + \sum_{k=k_0}^{k_f} \Delta W^{-}_{Pij}(k) \right] \tag{3.15b}$$

Note that the accumulation of weight changes until the completion of a trial eliminates the significance of the time shift in the term $\left| W^{\pm}_{Pij}(k-a) \right|$ in (3.12).

The credit assignment problem refers to the situation that some judicious choice of action at the present time may yield little or no immediate return, relative to other possible actions, but may allow maximization of future returns.[8]

[8] The term *return* denotes a single reinforcement signal that equals the reward minus the cost. In an environment that measures simultaneous non-zero reward and cost signals, a controller should maximize the return.

The assessment of responsibility among the recent actions for the current return is accomplished through the summation over the previous τ time steps in the reinforcement center learning equation (3.12). In the negative reinforcement center, for example, a correlation is achieved between $\Delta y_N(k)$ and the previous τ state transitions. This process of relating the current Δy_N to the previous Δx 's is referred to as chaining in animal learning. The learning rate coefficients discount the responsibility of previous actions for the current change in predicted return, where the reinforcement center outputs are predictions of future costs and rewards. Biological evidence suggests that no correlation exists between a simultaneous action and a change in predicted return, i.e. $c_0 = 0$, and $1 > c_j > c_\tau$ for $1 \leq j < \tau$.

3.2 Extension of the ACP to the Infinite Horizon, Optimal Control Problem

Limited modifications to the architecture and functionality of the original Associative Control Process result in a network with improved applicability to optimal control problems. Although Baird and Klopf [13] have suggested that this *modified* ACP will converge to the optimal control policy under *reasonable* assumptions, the analysis in §3.3 and the results in §3.6 suggest that the necessary conditions to obtain an optimal solution may be restrictive. This section is included to follow the development of the ACP and to motivate the single layer ACP architecture. The definition of the *modified* ACP follows from Baird and Klopf [13].

The modified ACP is applicable to a specialized class of problems; the environment with which the ACP interacts must be a non-absorbing, finite-state, finite-action, discrete-time Markov decision process. Additionally, the interface between the ACP and the environment guarantees that no acquired drive sensor will exhibit unity output for more than a single consecutive time step. This stipulation results in non-uniform time steps that are artificially defined as the intervals which elapse while the dynamic state resides within a bin.[9] The learning equations of the original ACP can be simplified by applying the fact that $x_i(k) \in \{1, 0\}$ and will not equal unity for two or more consecutive time steps. Accordingly, (3.8) and (3.9) yield,

$$f_s(\Delta x_i(k)) = \begin{cases} 1 & \text{if } x_i(k) = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{3.16}$$

Therefore, a consequence of the interface between the ACP and the environment is $f_s(\Delta x_i(k)) = x_i(k)$. A similar result follows from (3.9) and (3.14).

$$f_s(\Delta x_{ij}(k-a)) = \begin{cases} 1 & \text{if } x_i(k-a) = 1 \text{ and } j = j_{max}(k-a) \\ 0 & \text{otherwise} \end{cases} \tag{3.17}$$

The role of the reinforcement center weights becomes more well defined in the modified ACP. The sum of the inhibitory and excitatory weights in a reinforcement center estimate the expected discounted future reinforcement received if action j is performed in state i, followed by optimal actions being performed in all subsequent states. To achieve this significance, the reinforcement center output and learning equations must be recast. The external reinforcement term does not appear in the output equation of the reinforcement center; e.g. (3.4) becomes,

$$y_P(k) = f_n\left[\sum_{i=1}^{m} \left(W^+_{Pij_{max}(k)}(k) + W^-_{Pij_{max}(k)}(k)\right) x_i(k)\right]. \tag{3.18}$$

[9] Similar to §3.1, the state space is quantized into bins.

The expression for the change in the reinforcement center output is also slightly modified. Using the example of the negative reinforcement center, (3.13) becomes,

$$\Delta y_N(k) = \gamma y_N(k) - y_N(k-1) + r_N(k) \quad \text{where} \quad 0 < \gamma < 1. \tag{3.19}$$

If the negative reinforcement center accurately estimates the expected discounted future cost, $\Delta y_N(k)$ will be zero and no weight changes will occur. Therefore, the cost to complete the problem from time $k-1$ will approximately equal the cost accrued from time $k-1$ to k plus the cost to complete the problem from time k.[10]

$$y_N(k-1) = \gamma y_N(k) + r_N(k) \quad \text{when} \quad \Delta y_N(k) = 0 \tag{3.20}$$

The value of $r_N(k)$, therefore, represents the increment in the cost functional ΔJ from time $k-1$ to k. Recall that time steps are an artificially defined concept in the modified ACP; the cost increment must be an assessment of the cost functional over the real elapsed time.[11] The possibility that an action selected now does not significantly effect the cost in the far future is described by the discount factor γ, which also guarantees the convergence of the infinite horizon sum of discounted future costs.

The constants in (3.7) are defined as follows: $c_a = \frac{1}{2}$ and $c_b = 0$. Additionally, the terms which involve the absolute values of the weights are removed from both the motor center learning equation and the reinforcement center learning equation.

[10] This statement is strictly true for $\gamma = 1$.

[11] Time is discrete in this system. Time steps will coincide with an integral number of discrete time increments.

Equations (3.6) and (3.12) are written as (3.21) and (3.22), respectively. With the absence of these terms, the distinct excitatory and inhibitory weights could be combined into a single weight, which can assume positive or negative values. This change, however, is not made in [13].

$$\Delta W_{ij}^{\pm}(k) = \begin{cases} \frac{1}{2} f_s(\Delta x_i(k)) \left[y_P(k) - y_N(k) - y_j(k)\right] & \text{if } j = j_{max}(k) \\ 0 & \text{otherwise} \end{cases} \tag{3.21}$$

$$\Delta W_{Pij}^{\pm}(k) = \Delta y_P(k) \sum_{a=1}^{\tau} c_a f_s(\Delta x_{ij}(k-a)) \tag{3.22}$$

The motor center learning equation (3.21) causes the motor center weights to be adjusted so that $W_{ij}^{+}(k) + W_{ij}^{-}(k)$ will copy the corresponding sum of weights for the reinforcement center that wins reciprocal inhibition. The saturation limits on the motor center outputs are generalized; in contrast to (3.2), $f_n(x)$ is redefined as $f_{n'}(x)$.

$$f_{n'}(x) = \begin{cases} -\beta & \text{if } x \le -\beta \\ \beta & \text{if } x \ge \beta \\ x & \text{otherwise} \end{cases} \tag{3.23}$$

Additionally, the definition of reciprocal inhibition is adjusted slightly; the non-maximizing motor center outputs are suppressed to a minimum value $-\beta$ which is not necessarily zero.

Although the learning process is still divided into trials, the weight increments are incorporated into the weights at every time step, instead of after a trial has been completed. Equations (3.10) and (3.15) are now written as (3.24) and (3.25), respectively.

$$W_{ij}^{+}(k) = f_{W^+}\left[W_{ij}^{+}(k-1) + \Delta W_{ij}^{+}(k)\right] \tag{3.24a}$$

$$W_{ij}^{-}(k) = f_{W^-}\left[W_{ij}^{-}(k-1) + \Delta W_{ij}^{-}(k)\right] \tag{3.24b}$$

$$W^{+}_{Pij}(k) = f_{W^{+}}\left[W^{+}_{Pij}(k-1) + \Delta W^{+}_{Pij}(k)\right] \tag{3.25a}$$

$$W^{-}_{Pij}(k) = f_{W^{-}}\left[W^{-}_{Pij}(k-1) + \Delta W^{-}_{Pij}(k)\right] \tag{3.25b}$$

A procedural issue arises that is not encountered in the original ACP network, where the weights are only updated at the end of a trial. The dependence of the reinforcement center outputs on $j_{max}(k)$ requires that the motor center outputs be computed first. After learning, however, the motor center outputs and also $j_{max}(k)$ may have changed, resulting in the facilitation of a different set of reinforcement center weights. Therefore, if weight changes are calculated such that $j_{max}(k)$ changes, these weight changes should be implemented and the learning process repeated until $j_{max}(k)$ does not further change this time step.

In general, exploration of the state-action space is necessary to assure global convergence of the control policy to the optimal policy, and can be achieved by occasionally randomly selecting $j_{max}(k)$, instead of following reciprocal inhibition. Initiating new trials in random states also provides exploratory information.

3.3 Motivation for the Single Layer Architecture of the ACP

This section describes qualitative observations from the application of the modified two-layer ACP to the regulation of the aeroelastic oscillator; additional quantitative results appear in §3.6. In this environment, the modified ACP learning system fails to converge to a useful control policy. This section explains the failure by illustrating several characteristics of the two-layer implementation of the ACP algorithm that are incompatible with the application to optimal control problems.

The objective of a reinforcement learning controller is to construct a policy that, when followed, maximizes the expectation of the discounted future return. For the two-layer ACP network, the incremental return is presented as distinct cost and reward signals, which stimulate the two reinforcement centers to learn estimates of the expected discounted future cost and expected discounted future reward. The optimal policy for this ACP algorithm is to select, for each state, the action with the largest difference between estimates of expected discounted future reward and cost. However, the two-layer ACP network performs reciprocal inhibition between the two reinforcement centers and, therefore, selects the control action that either maximizes the estimate of the expected discounted future reward, or minimizes the estimate of the expected discounted future cost, depending on which reinforcement center wins reciprocal inhibition. Consider a particular state-action pair evaluated with both a large cost and a large reward. If the reward is slightly greater than the cost, only the large reward will be associated with this state-action pair. Although the true evaluation of this state-action pair is a small positive return, this action in this state may be incorrectly selected as optimal.

The reinforcement center learning mechanism incorporates both the current and the previous outputs of the reinforcement center. For example, the positive reinforcement center learning equation includes the term $\Delta y_P(k)$, given in (3.26), which represents the error in the estimate of the expected discounted future reward for the previous state $y_P(k-1)$.

$$\Delta y_P(k) = \gamma y_P(k) - y_P(k-1) + r_P(k) \tag{3.26}$$

A reinforcement center that loses the reciprocal inhibition process will have an output equal to zero. Consequently, the value of $\Delta y_P(k)$ will not accurately represent the error in $y_P(k-1)$ when $y_P(k)$ or $y_P(k-1)$ equals zero as a result of reciprocal inhibition. Therefore, $\Delta y_P(k)$ will be an invalid contribution to reinforcement learning if the positive and negative reinforcement centers alternate winning reciprocal inhibition. Similarly, $\Delta y_N(k)$ may be erroneous by a parallel argument. Moreover, the fact that learning occurs even for the reinforcement center which loses reciprocal inhibition assures that either $\Delta y_P(k)$ or $\Delta y_N(k)$ will be incorrect on every time step that a motor center is active. If no motor center is active, no set of weights between the acquired drive sensors and reinforcement centers are facilitated and both reinforcement centers will have zero outputs. Although no learning occurs in the reinforcement centers on this time step, both Δy_P and Δy_N will be incorrect on the next time step that a motor center is active.

The difficulties discussed above, which arise from the presence of two competing reinforcement centers, are reduced by providing a non-zero external reinforcement signal to only a single reinforcement center. However, the reinforcement center which receives zero external reinforcement will occasionally win reciprocal inhibition until it learns that zero is the correct output for every state. Using the sum of the reinforcement center output and the external reinforcement signal as the input to the reciprocal inhibition process may guarantee that a single reinforcement center will always win reciprocal inhibition.[12]

The optimal policy for each state is defined by the action which yields the largest expected discounted future return. The ACP network represents this in-

[12] The original ACP uses this technique in (3.4) and (3.5); the modified two-layer ACP eliminates the external reinforcement signal from the reinforcement center output in (3.18).

formation in the reinforcement centers and, through learning, transfers the value estimates to the motor centers, where an action is selected through reciprocal inhibition. The motor center learning mechanism copies either the estimate of expected discounted future cost or the estimate of expected discounted future reward, depending on which reinforcement center wins reciprocal inhibition, into the single currently active motor center for a given state. Potentially, each time this state is visited, a different reinforcement center will win reciprocal inhibition and a different motor center will be active. Therefore, at a future point in time, when this state is revisited, reciprocal inhibition between the motor center outputs may compare estimates of expected discounted future cost with estimates of expected discounted future reward. This situation, also generated when the two reinforcement centers alternate winning reciprocal inhibition, invalidates the result of reciprocal inhibition between motor centers. Therefore, the ACP algorithm to select a policy does not guarantee that a complete set of estimates of a consistent evaluation (i.e. reward, cost, or return) will be compared over all possible actions.

This section has introduced several fundamental limitations in the two-layer implementation of the ACP algorithm, which restrict its applicability to optimal control problems. By reducing the network to a single layer of learning centers, the resulting architecture does not interfere with the operation of the Drive-Reinforcement concept to solve infinite-horizon optimization problems.

3.4 A Single Layer Formulation of the Associative Control Process

The starting point for this research was the original Associative Control Process. However, several elements present in the original ACP network, which are consistent with the known physiology of biological neurons, are neither appropriate nor necessary in a network solely intended as an optimal controller. This section presents a single layer formulation of the modified ACP (Figure 3.5), and contains significantly fewer adjustable parameters, fewer element types, and no nonlinearity in the output equation. Although the physical structure of the single layer network is not faithful to biological evidence, the network retains the ability to predict classical and instrumental conditioning results [13].

The interface of the environment to the single layer network through m input sensors is identical to the interface to the modified ACP network through the acquired drive sensors. A single external reinforcement signal $r(k)$, which assesses the incremental return achieved by the controller's actions, replaces the distinct reward and cost external reinforcement signals present in the two-layer network.

A *node* and effector pair exists for each discrete network action.[13] The output of the j^{th} node estimates the expected discounted future return for performing action j in the current state and subsequently following an optimal policy. The sum of an excitatory and an inhibitory weight encode this estimate. Constructed from a single type of neuronal element, the single layer ACP architecture requires

[13] A *node* combines the functionality of the motor and reinforcement centers.

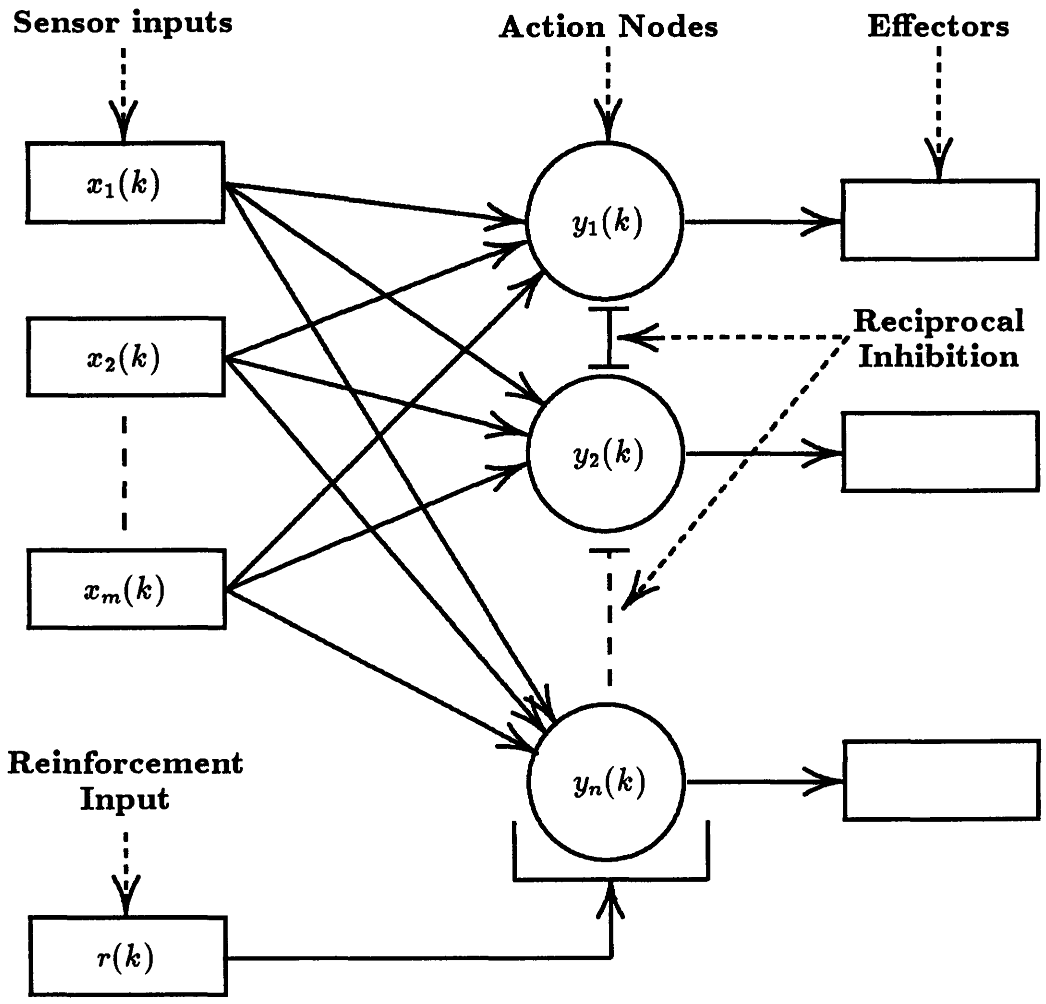

Figure 3.5. The single layer ACP architecture.

only a single linear output equation and a single learning equation.

$$y_j(k) = \sum_{i=1}^{m} \left(W_{ij}^{+}(k) + W_{ij}^{-}(k) \right) x_i(k) \tag{3.27}$$

The optimal policy, to maximize the expected discounted future return, selects for each state the action corresponding to the node with greatest output. Reciprocal inhibition between the n nodes defines a currently active node $j_{max}(k)$, similar to the process between motor centers in the two-layer ACP. However, the definition of reciprocal inhibition has been changed in the situation where multiple nodes have equally large outputs. In this case, which represents a state with multiple equally optimal actions, $j_{max}(k)$ will equal the node with the smallest index j. Therefore, the controller will perform an action and will learn on every time step.

The learning equation for a node resembles that of a reinforcement center. However, the absolute value of the connection weight at the time of the state change, which was removed in the modified ACP, has been restored into the learning equation [13]. This term, which was originally introduced for biological reasons, is not essential in the network and serves as a learning rate parameter. The discount factor γ describes how an assessment of return in the future is less significant than an assessment of return at the present. As before, only weights associated with a state-action pair being active in the previous τ time steps are eligible for change.

$$\begin{aligned} \Delta W_{ij\pm}(k) = & \left[\gamma y_{j_{max}(k)}(k) - y_{j_{max}(k-1)}(k-1) + r(k)\right] \\ & \cdot \sum_{a=1}^{\tau} c_a \left|W_{ij\pm}(k-a)\right| f_s\left(\Delta x_{ij}(k-a)\right) \end{aligned} \tag{3.28}$$

$$f_s\left(\Delta x_{ij}(k-a)\right) = \begin{cases} 1 & \text{if } j = j_{max}(k-a) \text{ and } x_i(k-a) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.29}$$

$$W_{ij}^{+}(k) = f_{W^+}\left[W_{ij}^{+}(k-1) + \Delta W_{ij}^{+}(k)\right] \tag{3.30a}$$

$$W_{ij}^{-}(k) = f_{W^-}\left[W_{ij}^{-}(k-1) + \Delta W_{ij}^{-}(k)\right] \tag{3.30b}$$

3.5 Implementation

The modified two-layer ACP algorithm and the single layer ACP algorithm were implemented in *NetSim* and evaluated as regulators of the AEO plant; fundamental limitations prevented a similar evaluation of the original ACP algorithm. The experiments discussed in this section and in §3.6 were not intended to represent an exhaustive analysis of the ACP methods. For several reasons, investigations focused more heavily on the Q learning technique, to be introduced in §4. First, the ACP algorithms can be directly related to the Q learning algorithm. Second, the relative functional simplicity of Q learning, which also possesses fewer free parameters, facilitated the analysis of general properties of direct learning techniques applied to optimal control problems.

This section details the implementation of the ACP reinforcement learning algorithms. The description of peripheral features that are common to both the ACP and Q learning environments will not be repeated in §4.5.

The requirement that the learning algorithm's input space consist of a finite set of disjoint states necessitated a BOXES [8] type algorithm to quantize the continuous dynamic state information that was generated by the simulation of the AEO equations of motion.[14] As a result, the input space was divided into 200 discrete states. The 20 angular boundaries occurred at 18° intervals, starting at 0°; the 9 boundaries in magnitude occurred at 1.15, 1.0, 0.85, 0.7, 0.55, 0.4, 0.3, 0.2, and 0.1

[14] The terms *bins* and *discrete states* are interpreted synonymously. The aeroelastic oscillator has two state variables: position and velocity. The measurement of these variables in the space of continuous real numbers will be referred to as the *dynamic state* or *continuous state*.

nondimensional units; the outer annulus of bins did not have a maximum limit on the magnitude of the state vectors that it contained.

The artificial definition of time steps as the non-uniform intervals between entering and leaving bins eliminates the significance of τ as the longest interstimulus interval over which delay conditioning is effective.

The ACP learning control system was limited to a finite number of discrete outputs: $+0.5$ and -0.5 nondimensional force units.

The learning algorithm operated through a hierarchical process of *trials* and *experiments*. Each experiment consisted of numerous trials and began with the initialization of weights and counters. Each trial began with the random initialization of the state variables and ran for a specified length of time.[15] In the two-layer architecture, the motor center and reinforcement center weights were randomly initialized using uniform distributions between $\{-1.0, -\alpha\}$ and $\{\alpha, 1.0\}$. In the single layer architecture, all excitatory weights were initialized within a small uniform random deviation of 1.0, and all inhibitory weights were initialized within a small uniform random deviation of $-\alpha$. The impetus for this scheme was to originate weights sufficiently large such that learning with non-positive reinforcement (i.e. zero reward and non-negative cost) would only decrease the weights.

The learning system operates in discrete time. At every time step, the dynamic state transitions to a new value either in the same bin or in a new bin and the system evaluates the current assessment of either cost and reward or reinforcement. For each discrete time step that the state remains in a bin, the reinforcement

[15] Initial states (position and velocity) were uniformly distributed between -1.2 and $+1.2$ nondimensional units.

Table 3.1. ACP parameters.

Name	Symbol	Value
Discount Factor	γ	0.95
Threshold	θ	0.0
Minimum Bound on $\|W\|$	α	0.1
Maximum Motor Center Output	β	1.0
Maximum Interstimulus Interval	τ	5

accumulates as the sum of the current reinforcement and the accretion of previous reinforcements discounted by γ. The arrows in Figure 3.6 with arrowheads lying in Bin 1 represent the discrete time intervals that contribute reinforcement to learning in Bin 1. Learning for Bin 1 occurs at t_5 where the total reinforcement equals the sum of r_5 and γ times the total reinforcement at t_4.

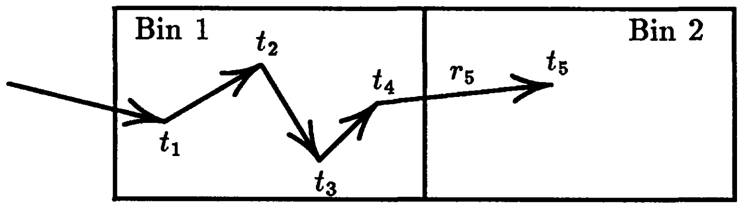

Figure 3.6. A state transition and reinforcement accumulation cartoon.

For the two-layer ACP, the reward presented to the positive reinforcement center was zero, while the cost presented to the negative reinforcement center was a quadratic evaluation of the state error. In the single layer learning architecture, the quadratic expression for the reinforcement signal r, for a single discrete time interval, was the negative of the product of the square of the magnitude of the state vector, at the final time for that interval, and the length of that time interval. The quadratic expression for cost in the two-layer ACP was $-r$. The magnitude of the control expenditure was omitted from the reinforcement function because the contribution was constant for the two-action control laws.

$$r = -(t_2 - t_3)\left[x(t_2)^2 + \dot{x}(t_2)^2\right] \tag{3.31}$$

3.6 Results

Figure 3.7 illustrates a typical segment of a trial, prior to learning, in which an ACP learning system regulated the AEO plant; the state trajectory wandered clockwise around the phase plane, suggesting the existence of two stable limit cycles.

The modified two-layer ACP system failed to learn a control law which drove the state from an arbitrary initial condition to the origin. Instead, the learned control law produced trajectories with unacceptable behavior near the origin (Figure 3.8). The terminal condition for the AEO state controlled by an optimal regulator with a finite number of discrete control levels, is a limit cycle. However, the two-layer ACP failed to converge to the optimal control policy. Although the absence of a set of learning parameters for which the algorithm would converge to an optimal solution cannot be easily demonstrated, §3.3 clearly identifies several undesirable properties of the algorithm.

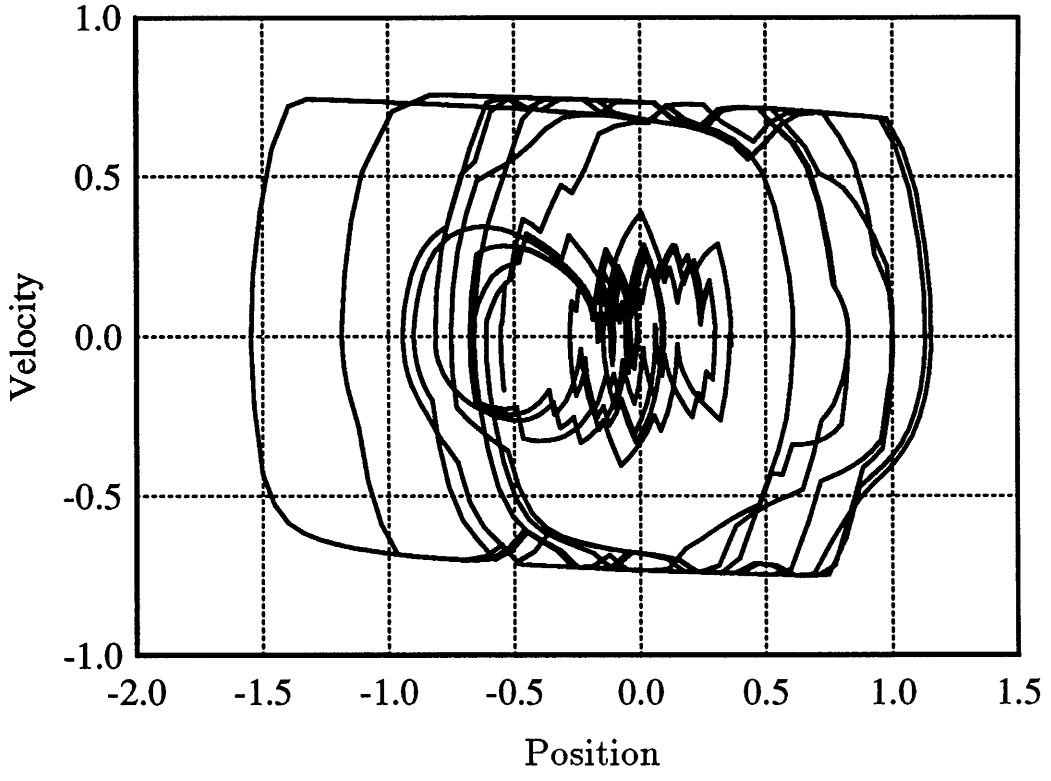

Figure 3.7. A characteristic AEO state trajectory achieved by a reinforcement learning algorithm prior to learning.

The single layer architecture of the ACP learned the optimal control law, which successfully regulated the AEO state variables near zero from any initial condition within the region of training, $\{-1.2, 1.2\}$. The performance of the control policy was limited by the coarseness of the bins and the proximity of bin boundaries to features of the nonlinear dynamics. The restricted choice of control actions also bounds the achievable performance, contributing to the rough trajectory in Figure 3.9.

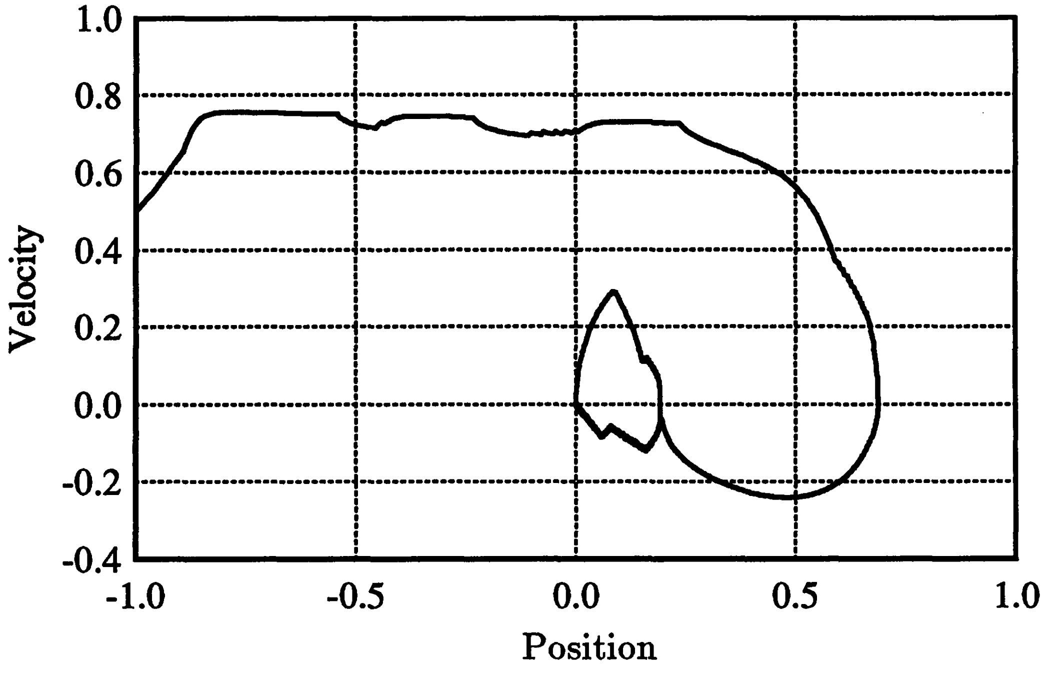

Figure 3.8. The AEO state trajectory achieved by the modified two-layer ACP after learning.

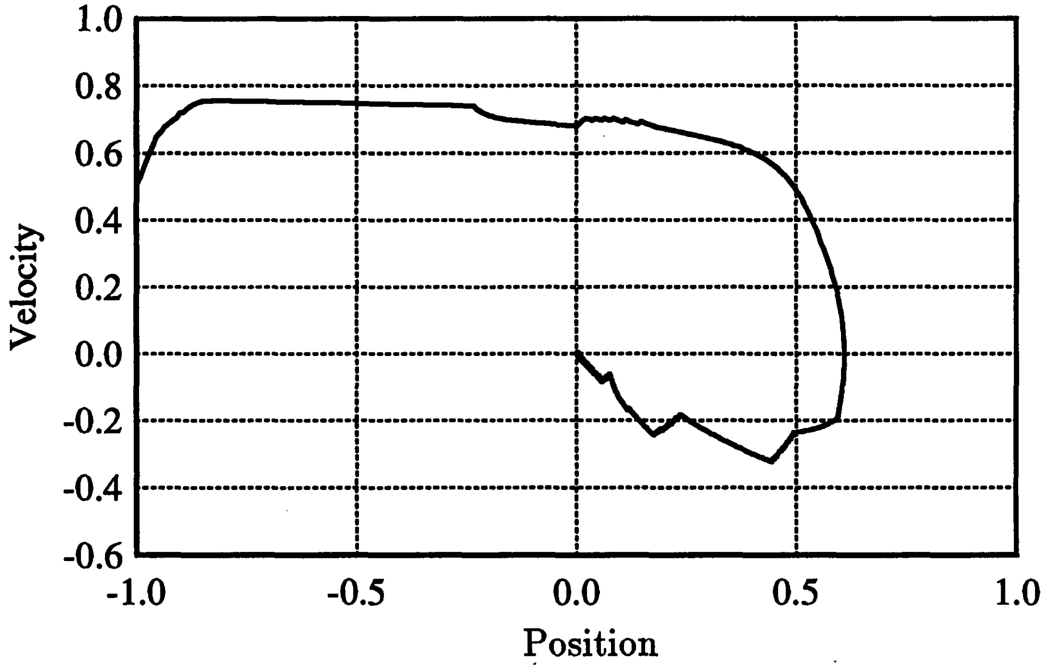

Figure 3.9. The AEO state trajectory achieved by the single layer ACP after learning.

Chapter 4

Policy and Value Iteration

The notation and concepts presented in §4.1 through §4.4 follow directly from Watkins' thesis [16] and [17]. §4.5 and §4.6 present results of applying Q learning to the AEO. §4.7 explores a continuous version of Q learning.

4.1 Terminology

4.1.1 Total Discounted Future Return

A discrete-time system that performs an action a_k in a state x_k, at time k, receives a performance evaluation r_k associated with the transition to the state x_{k+1} at time $k+1$; the evaluation r_k is referred to as the return at time k.[1] The *total future return* after time k, which equals the sum of the returns assessed between time k and the completion of the problem, may be unbounded for an infinite

[1] Watkins defines *return* as the total discounted future *reward*; this paper equates the terms *return* and *reward.*

horizon problem. However, the return received in the distant future is frequently less important, at the present time, than contemporary evaluations. Therefore, the *total discounted future return*, defined in (4.1) and guaranteed to be finite, is proposed.

$$\sum_{n=0}^{\infty} \gamma^n r_{k+n} = r_k + \gamma r_{k+1} + \gamma^2 r_{k+2} + \ldots + \gamma^n r_{k+n} + \ldots \tag{4.1}$$

The discount factor, $0 \leq \gamma < 1$, determines the present value of future returns.

4.1.2 The Markov Decision Process

A non-absorbing, finite-state, finite-action, discrete time Markov decision process is described by a bounded set of states S, a countable set of actions for each state $A(x)$ where $x \in S$, a transition function T, and an evaluation mechanism R. At time k, the state is designated by a random variable X_k and the true value x_k. The transition function defines $X_{k+1} = T(x_k, a_k)$ where $a_k \in A(x_k)$; the new state must not equal the previous state with probability equal to unity. At time k, the return is denoted by a random variable $R_k = R(x_k, a_k)$ and the actual evaluation r_k. The expectation of the return is written $\overline{R_k}$. The Markov property implies that the transition and evaluation functions depend on the current state and current action, and do not depend on previous states, actions, or evaluations.

4.1.3 Value Function

In a Markov decision process, the expectation of the total discounted future return depends only on the current state and the stationary policy. A convenient notation for the probability that performing action a in state x will leave the system in state y is $P_{xy}(a)$. The random variable representing the future state X_{k+n} achieved by starting in the state x_k at time k and following policy f for n time steps is written as $X(x_k, f, n)$.

$$X(x_k, f, 0) = x_k \tag{4.2a}$$

$$X(x_k, f, 1) = X_{k+1} = T(x_k, f(x_k)) \tag{4.2b}$$

If policy f is followed for n time steps from state x_k at time k, the return realized for applying $f(x_{k+n})$ in state x_{k+n} is expressed as $R(x_k, f, n)$.

$$R(x_k, f, 0) = R(x_k, f(x_k)) = R_k \tag{4.3a}$$

$$R(x_k, f, n) = R(X_{k+n}, f(X_{k+n})) = R_{k+n} \tag{4.3b}$$

The expected total discounted future return subsequent to the state x, applying the invariant policy f, is the *value function* $V_f(x)$.

$$V_f(x) = \overline{R(x,f,0) + \gamma R(x,f,1) + \ldots + \gamma^n R(x,f,n) + \ldots} \tag{4.4a}$$

$$= \overline{R(x,f,0)} + \gamma V_f\left(\overline{X(x,f,1)}\right) \tag{4.4b}$$

$$= \overline{R(x,f,0)} + \gamma \sum_y P_{xy}(f(x)) V_f(y) \tag{4.4c}$$

In (4.4c), y is the subset of S that is reachable from x in a single time step.

4.1.4 Action Value

The *action-value* $Q_f(x, a)$ is the expectation of the total discounted future return for starting in state x, performing action a, and subsequently following policy f. Watkins refers to action-values as Q values. A Q value represents the same information as the sum of an excitatory weight and an inhibitory weight in Drive-Reinforcement learning, which is used in the single layer ACP.

$$Q_f(x,a) = \overline{R(x,a)} + \gamma \sum_y P_{xy}(a) V_f(y) \tag{4.5}$$

The expression for an action-value (4.5) indicates that the value function for policy f must be completely known prior to computing the action-values.

Similarly, $Q_f(x,g)$ is the expected total discounted future return for starting in x, performing action $g(x)$ according to policy g, and subsequently following policy f.

4.2 Policy Iteration

The *Policy Improvement Theorem* [16] states that a policy g is uniformly better than or equivalent to a policy f if and only if,

$$Q_f(x,g) \geq V_f(x) \quad \text{for all} \quad x \in S. \tag{4.6}$$

This theorem and the definition of action-values imply that for a policy g which satisfies (4.6), $V_g(x) \geq V_f(x)$ for all $x \in S$. The *Policy Improvement Algorithm* selects an improved policy g according to the rule: $g(x) = a \in A(x)$ such that a is the argument that maximizes $Q_f(x,a)$. However, to determine the action-values $Q_f(x,a)$ for f, the entire value function $V_f(x)$ must first be calculated. In the context of a finite-state, finite-action Markov process, policy improvement will terminate after applying the algorithm a finite number of times; the policy g will converge to an optimal policy.

The *Optimality Theorem* [16] describes a policy f^* which cannot be improved using the policy improvement algorithm. The associated value function $V_{f^*}(x)$ and action-values $Q_{f^*}(x,a)$ satisfy (4.7) and (4.8) for all $x \in S$.

$$V_{f^*}(x) = \max_{a \in A(x)} Q_{f^*}(x,a) \tag{4.7}$$

$$f^*(x) = a \quad \text{such that} \quad Q_{f^*}(x,a) = V_{f^*}(x) \tag{4.8}$$

The optimal value function and action-values are unique; the optimal policy is unique except in states for which several actions yield equal and maximizing action-values.

4.3 Value Iteration

The *value iteration* [16] procedure calculates an optimal policy by choosing for each state the action which effects a transition to the new state that possesses the maximum evaluation; the optimal value function determines the evaluation of each state that succeeds the current state. The expected total discounted future return for a finite horizon process which consists of n transitions and a subsequent final return, to evaluate the terminal state, is represented as V^n. The value function, which corresponds to the infinite horizon problem, is approximated by repeatedly applying rule (4.9) to an initial estimate V^0.

$$V^n(x) = \max_{a \in A(x)} \left[\overline{R(x,a)} + \gamma \sum_y P_{xy}(a) V^{n-1}(y) \right] \tag{4.9}$$

Value iteration guarantees that the limit in (4.10) approaches zero uniformly over all states. Therefore, V^n converges to the optimal value function and the optimal policy can be derived directly from V_f^*.

$$\lim_{n \to \infty} |V^n - V_{f^*}| = 0 \tag{4.10}$$

Although this procedure is computationally simplest if all states are systematically updated so that V^n is completely determined from V^{n-1} before V^{n+1} is calculated for any state, Watkins has demonstrated that the value iteration method will still converge if the values of individual states are updated in an arbitrary order, provided that all states are updated sufficiently frequently.

4.4 Q Learning

Unfortunately, neither the optimal policy nor optimal value function can be initially known in a control problem. Therefore, the learning process involves simultaneous, incremental improvements in both the policy function and the value function. Action-values $Q_{f_k}(x_k, a_k)$ for each state-action pair at time k contain both policy and value information; the policy and value functions at time k are defined in (4.11) and (4.12) in terms of Q values.

$$f_k^Q(x) = a \quad \text{such that} \quad Q_k(x,a) = V_k^Q(x) \tag{4.11}$$

$$V_k^Q(x) = \max_a [Q_k(x,a)] \tag{4.12}$$

The superscript Q denotes the derivation of the policy and the value function from the set of action-values $Q_{f_k}(x_k, a_k)$. Single step *Q learning* adjusts the action-values according to (4.13).

$$Q_{k+1}(x_k,a_k) = (1-\alpha)Q_k(x_k,a_k) + \alpha(r_k + \gamma V_k^Q(x_{k+1})) \tag{4.13}$$

The positive learning rate constant α is less than unity. Only the action-value of the state-action pair (x_k, a_k) is altered at time k; to guarantee convergence of the value function to the optimal, each action must be repeatedly performed in each state. As a form of dynamic programming, Q learning may be described as incremental Monte-Carlo value iteration.

4.5 Implementation

This section formalizes the implementation of the Q learning algorithm as a regulator for the aeroelastic oscillator plant. The environment external to the Q learning process was similar to that used for the ACP experiments in §3.5 and §3.6.

However, the quantization of the state space was altered. The boundaries of the 260 bins that covered the state space were defined by magnitudes M and angles A; the outer annulus of bins did not have a maximum magnitude.

$$\begin{aligned} M = \{&0.0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.5, 0.6, \\ &0.7, 0.85, 1.0\} \\ A = \{&0^\circ, 18^\circ, 36^\circ, 54^\circ, 72^\circ, 90^\circ, 108^\circ, 126^\circ, 144^\circ, 162^\circ, 180^\circ, 198^\circ, \\ &216^\circ, 234^\circ, 252^\circ, 270^\circ, 288^\circ, 306^\circ, 324^\circ, 342^\circ\} \end{aligned}$$

The bins were labeled with integer numbers from 0 to 259, starting with the bins in the outer ring, within a ring increasing in index with increasing angle from 0°, and continuing to the next inner ring of bins.

For each state-action pair, the Q learning algorithm stores a real number that represents the Q value. At the start of a new *NetSim* experiment, all Q values were initialized to zero.

The two parameters which appear in (4.13) were: $\gamma = 0.95$ and $\alpha = 0.5$. In this context, α is a learning rate parameter; in the ACP description, α was the minimum bound on the absolute value of the weights. The return r_k was given in (3.31) as the negative of the product of the squared magnitude of the state vector and the length of the time interval.

4.6 Results

The results of two experiments, conducted in the *NetSim* [11] environment, characterize the performance of the Q learning algorithm. The two experiments

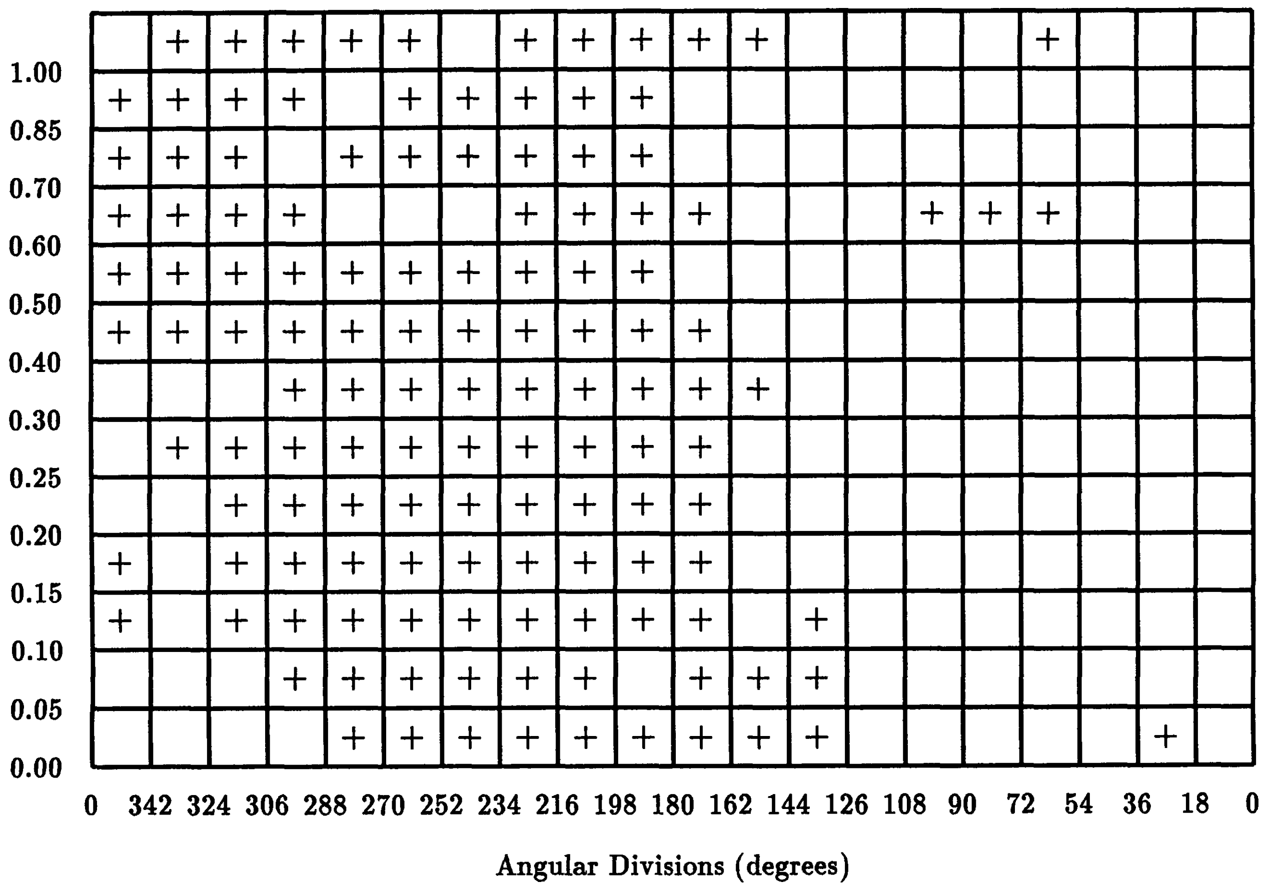

Figure 4.1a. A Cartesian representation of the two-action optimal control policy.

differ in the set of allowable control actions.

Experiment 1: $u_k \in \{0.5, -0.5\}$

Experiment 2: $u_k \in \{0.5, 0.33, 0.167, 0.0, -0.167, -0.33, -0.5\}$

The learned optimal policy for *Experiment 1* appears in Figures 4.1a and 4.1b. The control law applied a $+0.5$ force whenever the state resided in a bin containing a $+$ and applied -0.5 whenever the state was in an empty bin. The general form of this control policy resembles the non-optimal bang-bang law that was derived from

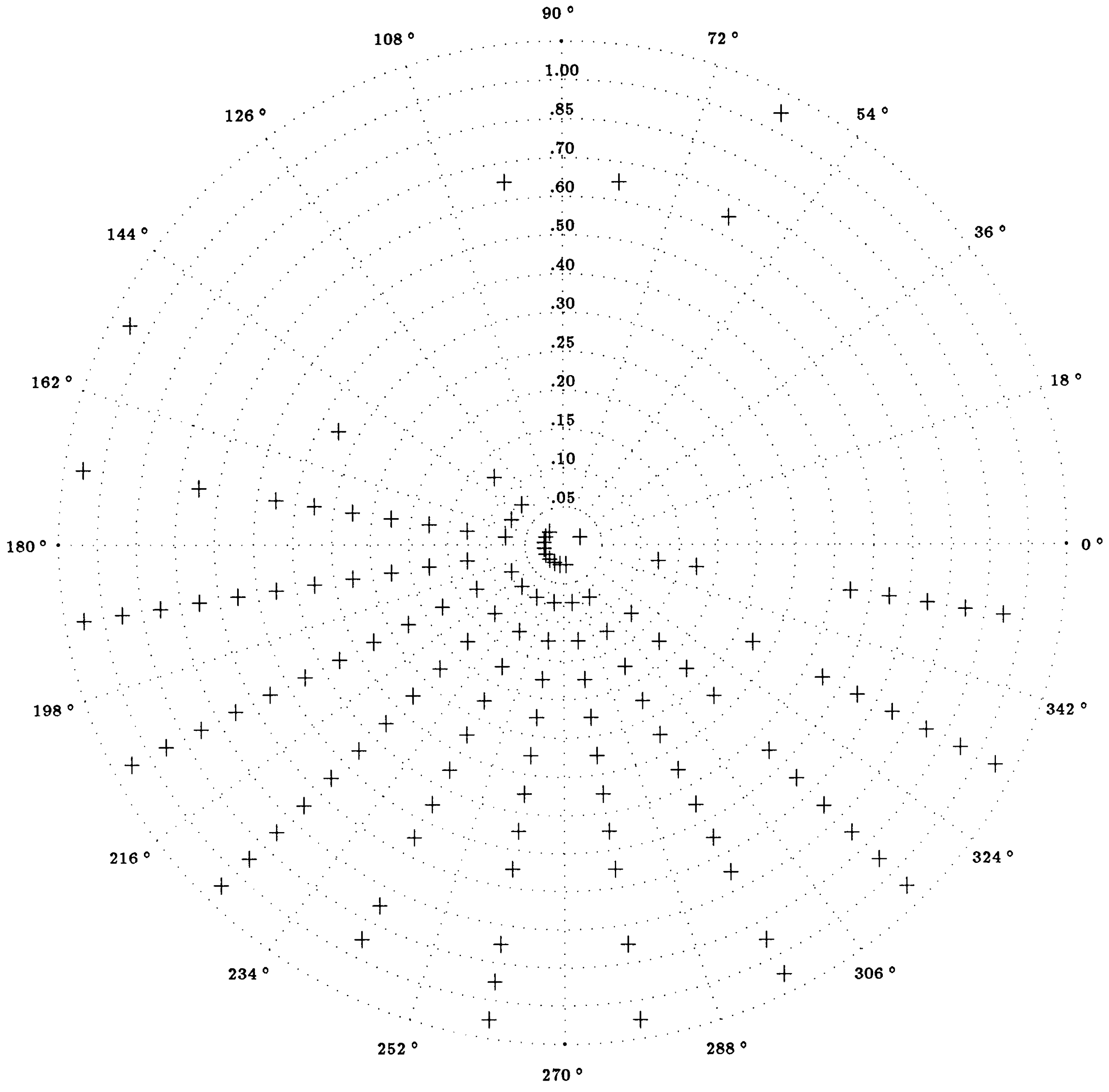

Figure 4.1b. A polar representation of the two-action optimal control policy.

a LQR solution in §2.4.3. Figure 2.9 demonstrated that for the non-optimal bang-bang control policy, the state trajectory slowly approached the origin along a linear

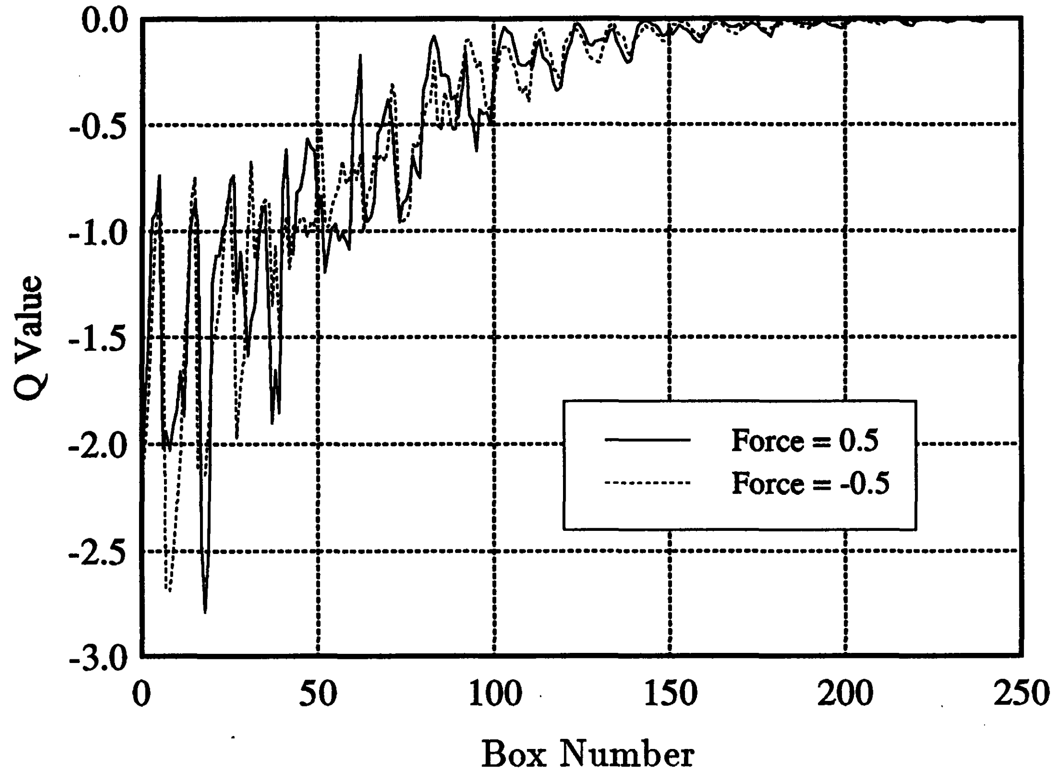

Figure 4.2. *Experiment 1*: Expected discounted future return (Q value) for each state-action pair.

switching curve. To avoid the high cost of this behavior, the optimal two-action solution will not contain a linear switching curve. A positive force must be applied in some states where the bang-bang rule (§2.4.3) dictated a negative force and a negative force must be applied in some bins below the linear switching curve. The trajectories that result from the control policies constructed in *Experiments 1* and *2* avoid the problem of slow convergence along a single switching curve. Although some regions of the control policy appear to be arbitrary, there exists a structure. For two bins bounded by the same magnitudes and separated by 180°, the optimal actions will typically be opposite. For example, the three + bins bounded by 0.6, 0.7, 54°, and 108° are reflections of the blank bins bounded by 0.6, 0.7, 234°, and

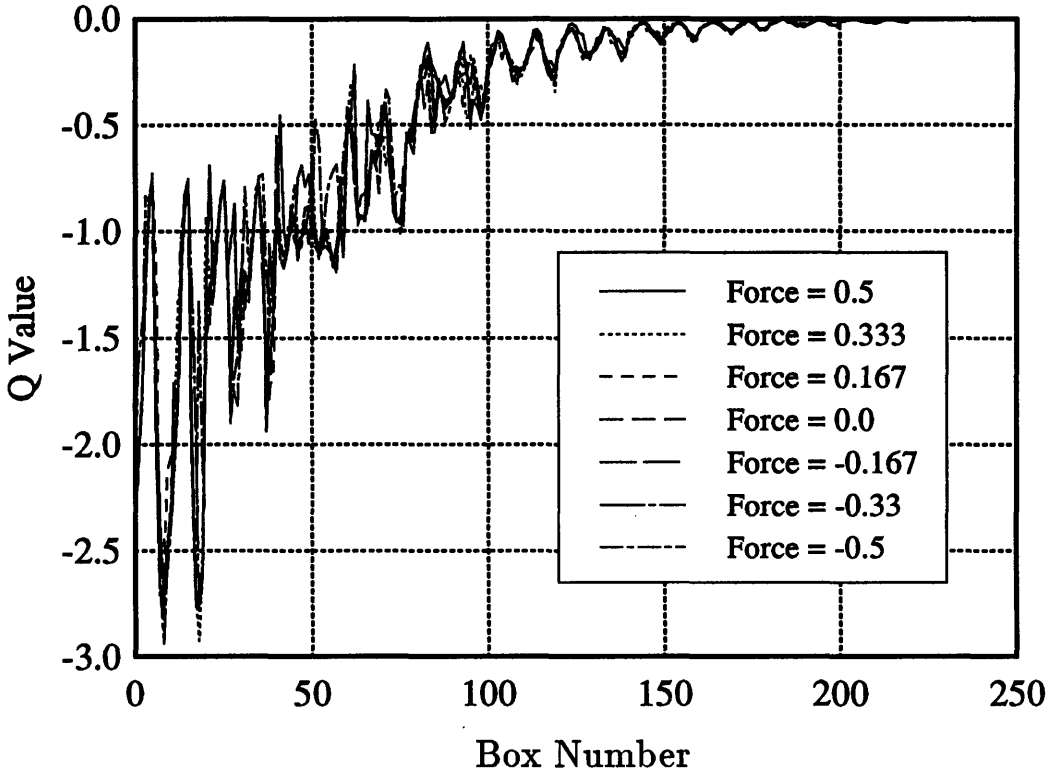

Figure 4.3. *Experiment 2*: Expected discounted future return (Q value) for each state-action pair.

288°. The 15 pairs of bins which violate this pattern lie primarily near the linear switching curve (§2.4.3).

Figures 4.2 and 4.3 compare the expected discounted future returns for all state-action pairs in *Experiments 1* and *2*, respectively. The expected discounted future return was negative for all state-action pairs because only negative return (i.e. cost) was assessed. Moreover, the Q values for bins nearer the origin were greater than the Q values for outlying bins. The fact that a non-optimal action performed in a single bin does not significantly affect the total cost for a trajectory, when optimal actions are followed in all other bins (in this problem), explains the similarity between most Q values associated with different actions and the same state. Additionally, the Q values varied almost periodically as a function of the bin number; the largest variance existed for the bins farthest from the origin. All trajectories tended to approach the origin along the same paths through the second and fourth quadrants (Figures 4.4, 4.6, 4.8, and 4.10). Therefore, if an initial condition was such that the limited control authority could not move the state onto the nearest path toward the origin, then the trajectory circled halfway around the state space to the next path toward the origin. This characteristic was a property of the AEO nonlinear dynamics, and accounted for the large differences in Q values for neighboring bins. In *Experiment 1*, there existed bins for which the choice of the initial control action determined whether this circling was necessary. For these bins, the expected discounted future returns for the two actions differed substantially.

The control law constructed in *Experiment 2* was expected to outperform the control law constructed in *Experiment 1* (i.e. for each bin, the maximum Q value from Figure 4.3 would exceed the maximum Q value from Figure 4.2). For the data presented, this expectation is true for 60% of the bins. The bins that violate this prediction are entirely located in the regions of the state space that the state enters least frequently. *Experiment 2*, with a greater number of state-action pairs, requires substantially more training than *Experiment 1*. The fact that for certain bins, the maximum Q value from *Experiment 1* exceeds that for *Experiment 2* signals insufficient learning in those bins for *Experiment 2*.

No explicit search mechanism was employed during learning. Moreover, the dynamics tended to force all trajectories onto the same paths, so that many bins were seldom entered. Therefore, to assure that a globally optimal policy was attained, sufficient trials were required so that the random selection of the initial

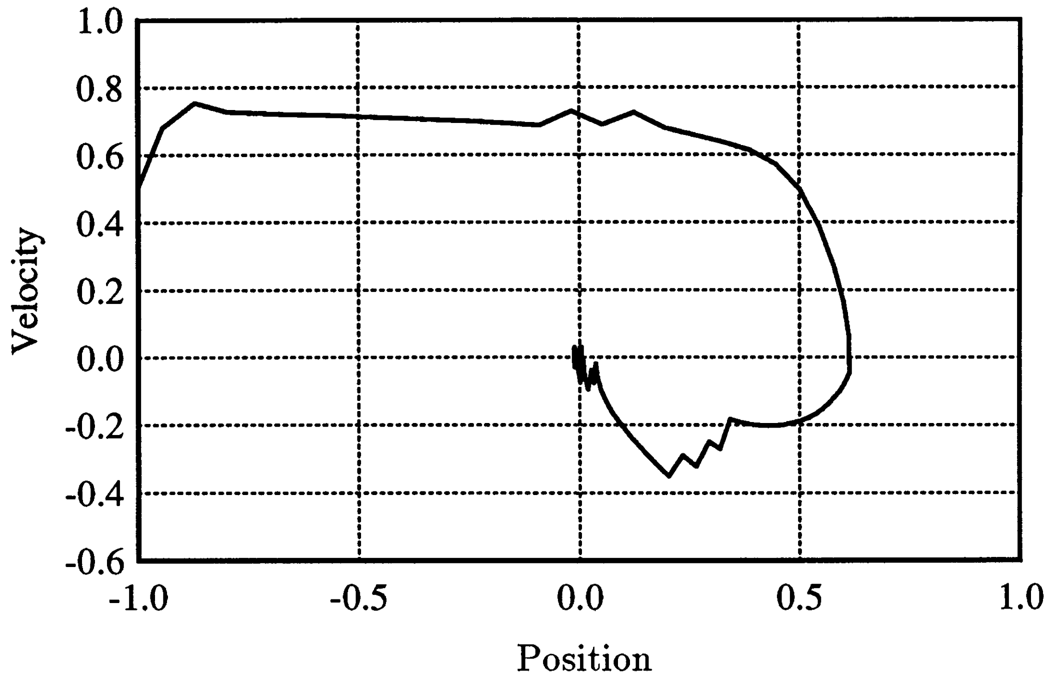

Figure 4.4. *Experiment 1*: State trajectory, $\underline{x}_0 = \{-1.0, 0.5\}$.

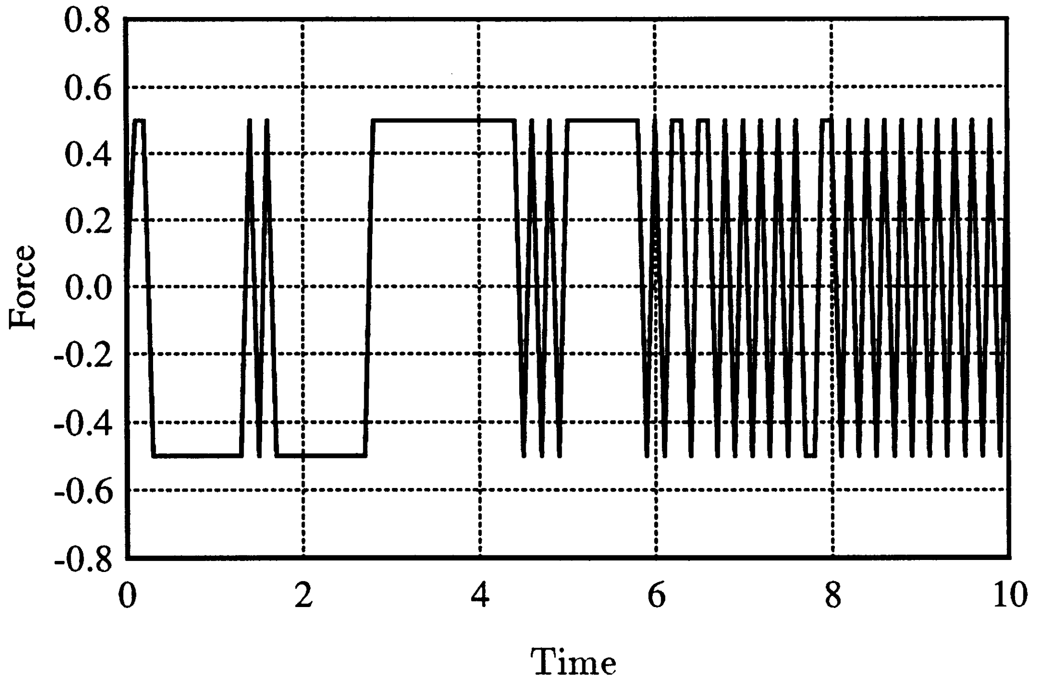

Figure 4.5. *Experiment 1*: Control history, $\underline{x}_0 = \{-1.0, 0.5\}$.

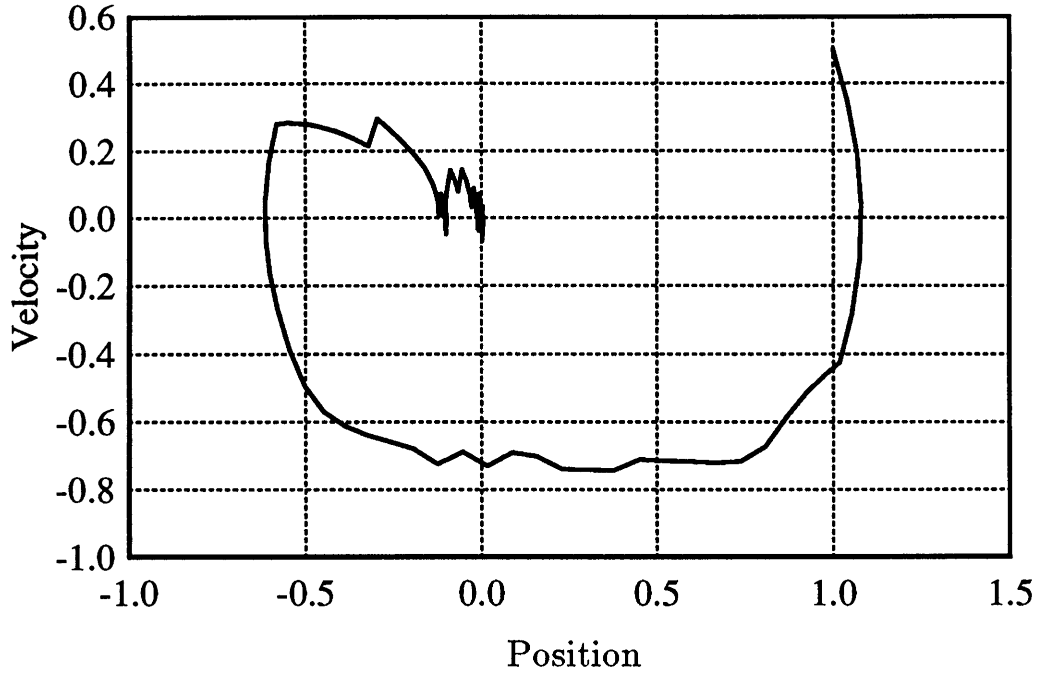

Figure 4.6. *Experiment 1*: State trajectory, $\underline{x}_0 = \{1.0, 0.5\}$.

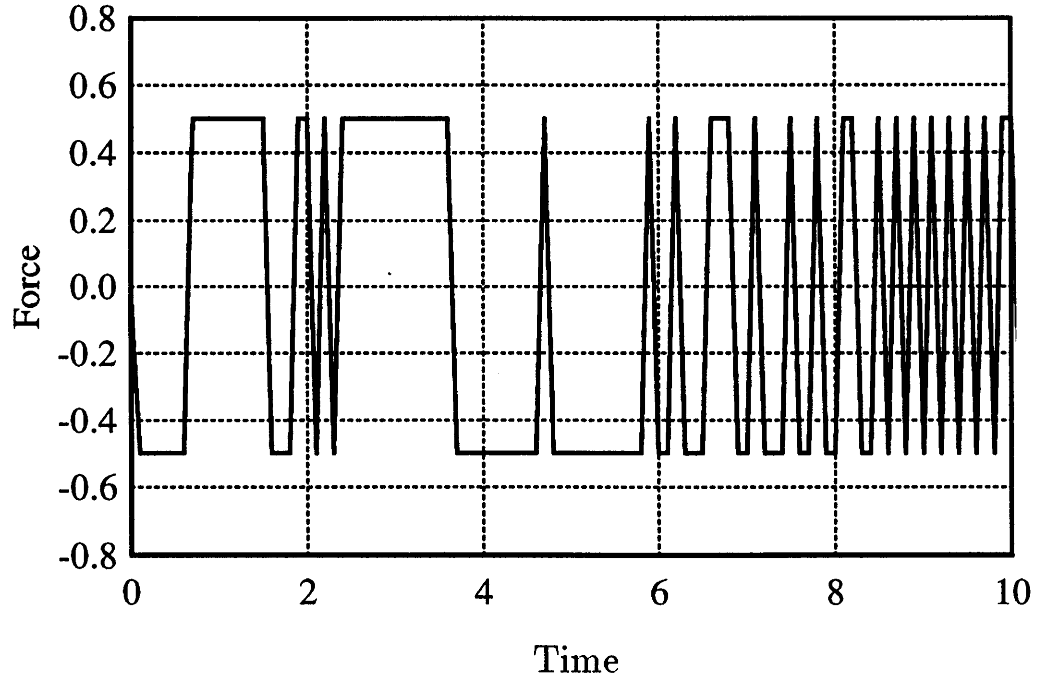

Figure 4.7. *Experiment 1*: Control history, $\underline{x}_0 = \{1.0, 0.5\}$.

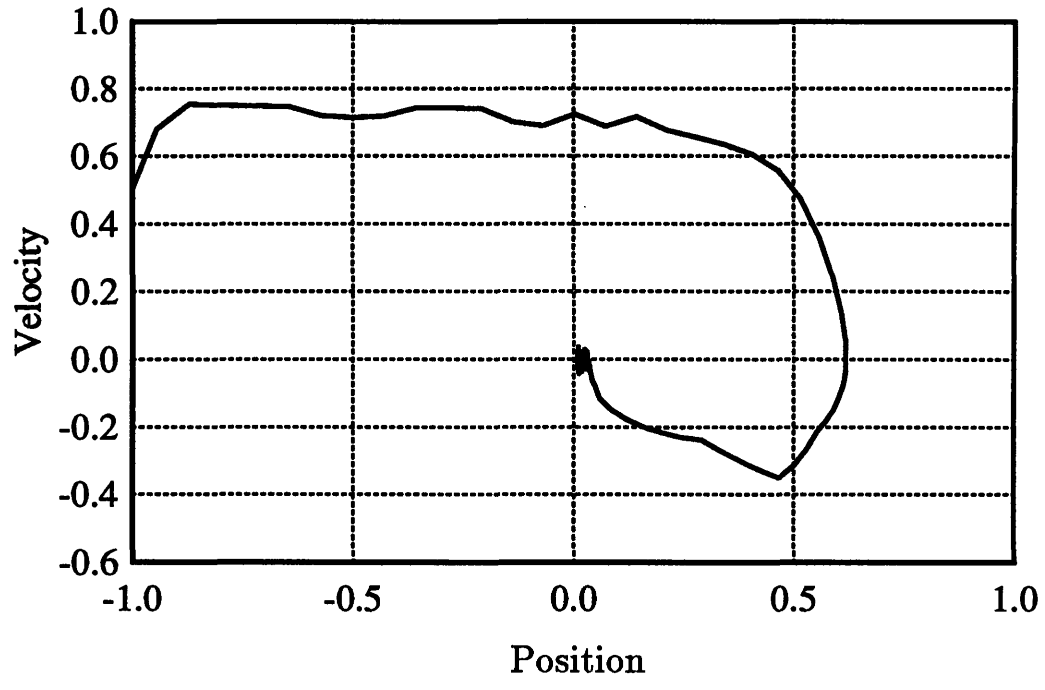

Figure 4.8. *Experiment 2*: State trajectory, $\underline{x}_0 = \{-1.0, 0.5\}$.

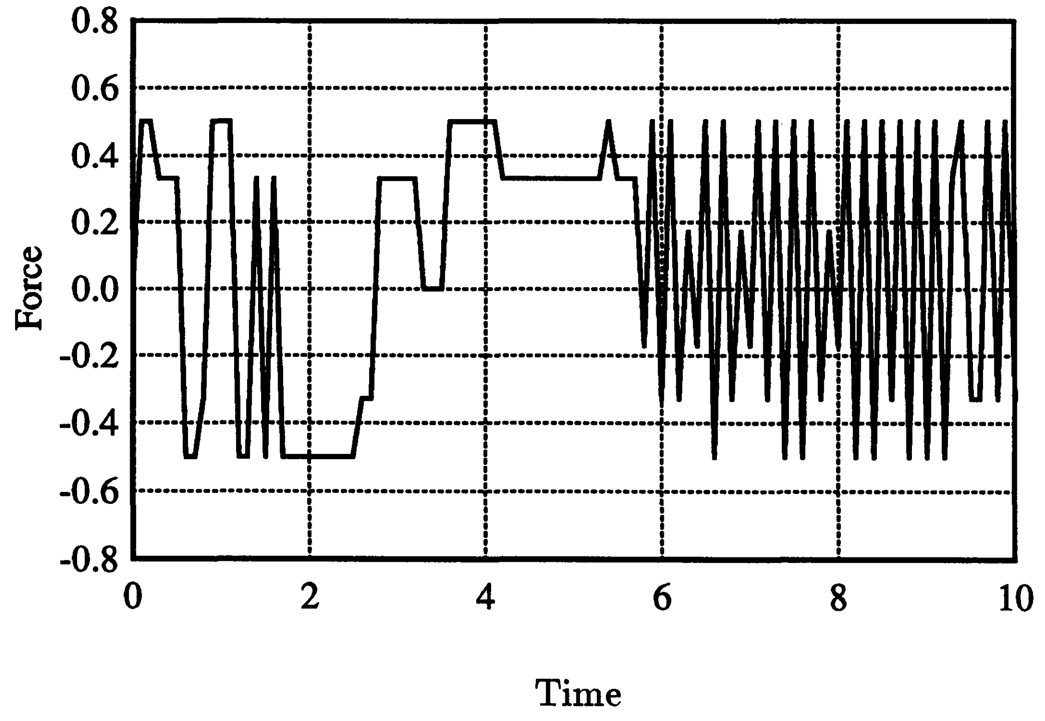

Figure 4.9. *Experiment 2*: Control history, $\underline{x}_0 = \{-1.0, 0.5\}$.

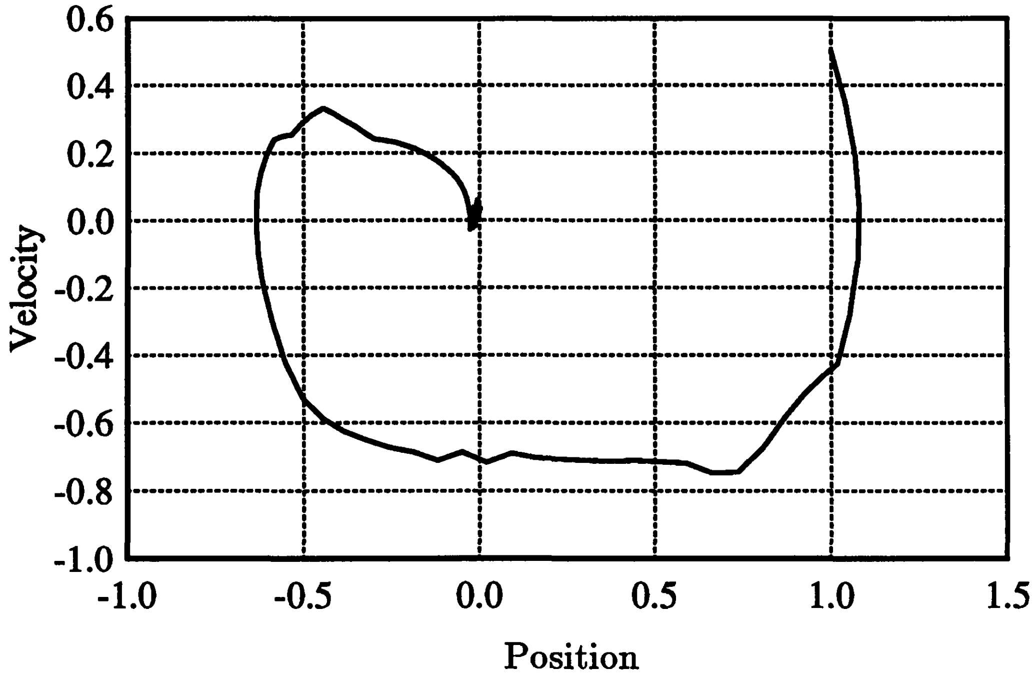

Figure 4.10. *Experiment 2*: State trajectory, $\underline{x}_0 = \{1.0, 0.5\}$.

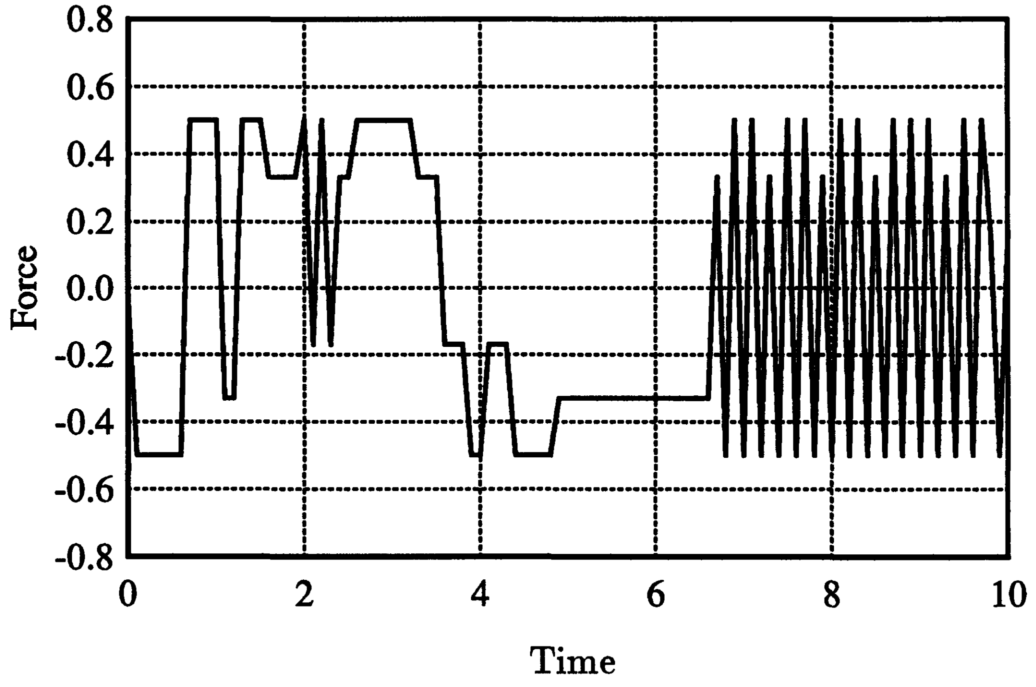

Figure 4.11. *Experiment 2*: Control history, $\underline{x}_0 = \{1.0, 0.5\}$.

state provided sufficient experience about performing every action in every bin. Over 2000 trials were performed in each experiment to train the learning system. If the learning rate had been a focus of the research, an explicit search procedure could have been employed. Additionally, in some experiments, the Q values did not converge to a steady state. Some of the bins were excessively large such that the optimal actions (in a continuous sense) associated with extreme points within the bin were quite different. Therefore, the Q values for such a bin, and subsequently the optimal policy, would vary as long as training continued.

All Q learning experiments learned a control policy that successfully regulated the aeroelastic oscillator. The state trajectories and control histories of the AEO, with initial conditions $\{-1.0, 0.5\}$ and $\{1.0, 0.5\}$, which resulted from the control laws learned in *Experiments 1* and *2*, appear in Figures 4.4 through 4.11. The limitation of the control to discrete levels, and the associated sharp control switching, resulted in rough state trajectories as well as limit cycles around the origin. The results illustrate the importance of a smooth control law; a continuous control law (LQR) was discussed in §2.4.2 and a characteristic state trajectory appeared in Figure 2.5. The absence of actuator dynamics and a penalty on the magnitude of the control allow the application of larger values of control to maximize reinforcement. Therefore, *Experiment 2* seldom selected a smaller or zero control force, even for bins near the origin. In *Experiment 1* the magnitude of the control was constant.

4.7 Continuous Q Learning

The algorithm described in §4.4 operates with both a finite set of states and discrete control actions. The optimal control a^* maximizes $Q(x, a^*)$ for the current state x. To identify the optimal control for a specific state, therefore, requires the comparison of $Q(x,a)$ for each discrete action $a \in A(x)$.[2] However, quantization of the input and output spaces is seldom practical or acceptable

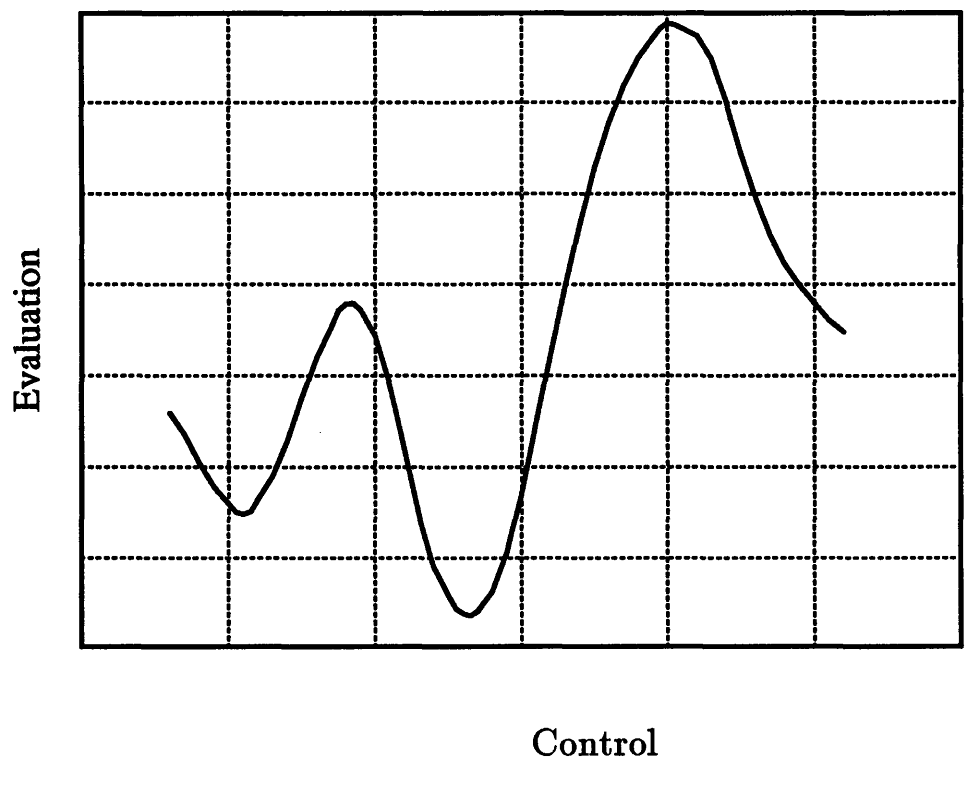

Figure 4.12. A continuous Q function for an arbitrary state x.

A potential new algorithm, related to the Q learning process of §4.4 might se-

[2] A finite number of Q values exist and, therefore, the maximum Q value is easily obtained.

lect, for each discrete state, the optimal control action from a bounded continuum and employ a continuous *Q function* that maps the control levels into evaluations of the expected discounted future return (Figure 4.12). However, to identify the optimal control for a state requires the maximization of a potentially multi-modal bounded function; this extremization procedure is problematic relative to the maximization of discrete Q values. The maximization of a multi-modal function at each stage in discrete time is itself a complicated optimization problem and, although not intractable, makes any continuous Q learning procedure impractical for real-time, on-line applications. This Q learning algorithm directly extends to incorporate several control variables; the optimal controls for a state are the arguments that maximize the multidimensional Q function.

The Q learning concept may be further generalized to employ continuous inputs and continuous outputs. The algorithm maps expectations of discounted future returns as a smooth function of the state and control variables. The current state will define a hyperplane through this Q function that resembles Figure 4.12 for a single control variable. Again, a maximization of a potentially multi-modal function is required to compute each control. Although the continuous nature of the state inputs does not operationally affect the identification of an optimal control, the mapping and learning mechanisms must incorporate the local generalization of information with respect to state, a phenomenon which does not occur for discrete state bins. A continuous Q function could be represented by any function approximation scheme, such as the spatially localized connectionist network introduced in §6.

Baird [42] addressed the difficulty of determining the global maximum of a multi-modal function. Millington [41] proposed a direct learning control method that used a spatially localized connectionist / Analog Learning Element. The learning system defined, as a distributed function of state, a continuous probability density function for control selection.

Chapter 5

Temporal Difference Methods

Temporal difference (TD) methods comprise a class of incremental learning procedures that predict future system behavior as a function of current observations. The earliest temporal difference algorithm appeared in Samuel's checker-playing program [18].[1] Manifestations of the TD algorithm also exist in Holland's bucket brigade [19], Sutton's Adaptive Heuristic Critic [5,20], and Klopf's Drive-Reinforcement learning [12]. This chapter summarizes Sutton's unification of these algorithms into a general temporal difference theory [15] and then analyzes the similarities and distinctions between the Adaptive Heuristic Critic, Drive-Reinforcement learning, and Q learning.

[1] The phrase *temporal difference* was proposed by Sutton in 1988 [15].

5.1 TD(λ) Learning Procedures

Most problems to which learning methods are applicable can be formulated as a prediction problem, where future system behavior must be estimated from transient sequences of available sensor outputs. Conventional supervised learning prediction methods associate an observation and a final outcome pair; after training, the learning system will predict the final outcome that corresponds to an input. In contrast, temporal difference methods examine temporally successive predictions of the final result to derive a similar mapping from the observations to the final outcome. Sutton demonstrates that TD methods possess two benefits relative to supervised learning prediction methods [15]. Supervised learning techniques must wait until the final outcome has been observed before performing learning calculations and, therefore, to correlate each observation with the final outcome requires storage of the sequence of observations that preceded the final result. In contrast, the TD approach avoids this storage requirement, incrementally learning as each new prediction and observation are made. This fact, and the associated temporal distribution of required calculations, make the TD algorithm amenable to running on-line with the physical plant. Through more efficient use of experience, TD algorithms converge more rapidly and to more accurate predictions. Although any learning method should converge to an equivalent evaluation with infinite experience, TD methods are guaranteed to perform better than supervised learning techniques after limited experience with a Markov decision process.

Temporal difference and conventional supervised learning are indistinguishable for single step prediction problems where the accuracy of a prediction is revealed immediately. In a multi-step prediction problem, partial information pertinent to the precision of a prediction is incrementally disclosed through temporally successive observations. This second situation is more prevalent in optimal control problems. Multi-stage problems consist of several temporally sequential observations $\{x_1, x_2, \ldots, x_m\}$ followed by a final result z. At each discrete time t, the learning system generates a prediction P_t of the final output, typically dependent on the current values of a weight set w. The learning mechanism is expressed as a rule for adjusting the weights.

Typically, supervised learning techniques employ a generalization of the Widrow-Hoff rule [21] to derive weight updates.[2]

$$\Delta w_t = \alpha(z - P_t)\Delta_w P_t \tag{5.1}$$

In contrast to (5.1), TD methods are sensitive to changes in successive predictions rather than the error between a prediction and the final outcome. Sutton has demonstrated that for a multi-step prediction problem, a TD(1) algorithm produces the same total weight changes for any observation-outcome sequence as the Widrow-Hoff procedure. The TD(1) algorithm (5.2) alters prior predictions to an equal degree.

$$\Delta w_t = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \Delta_w P_t \tag{5.2}$$

The temporal difference method generalizes from TD(1) to an algorithm that adjusts prior predictions in proportion to a factor that equals unity for the current time and

[2] The Widrow-Hoff rule, also known as the delta rule, requires that P_t be a linear function of w and x_t so that $\Delta_w P_t = x_t$.

decreases exponentially with increasing elapsed time. This algorithm is referred to as TD(λ) and (5.3) defines the learning process, where P_{m+1} is identically z.

$$\Delta w_t = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \Delta_w P_k \tag{5.3}$$

$$0 \leq \lambda \leq 1 \tag{5.4}$$

An advantage of this exponential weighting is the resulting simplicity of determining future values of the summation term in (5.3).

$$\begin{aligned} S(t+1) = \sum_{k=1}^{t+1} \lambda^{t+1-k} \Delta_w P_k &= \Delta_w P_{t+1} + \sum_{k=1}^{t} \lambda^{t+1-k} \Delta_w P_k \\ &= \Delta_w P_{t+1} + \lambda \sum_{k=1}^{t} \lambda^{t-k} \Delta_w P_k = \Delta_w P_{t+1} + \lambda S(t) \end{aligned} \tag{5.5}$$

In the limiting case where $\lambda = 0$, the learning process determines the weight increment entirely by the resulting effect on the most recent prediction. This TD(0) algorithm (5.6) resembles (5.1) if the final outcome z is replaced by the subsequent prediction.

$$\Delta w_t = \alpha(P_{t+1} - P_t)\Delta_w P_t \tag{5.6}$$

5.2 An Extension of TD(λ)

The TD family of learning procedures directly generalizes to accomplish the prediction of a discounted, cumulative result, such as the expected discounted future cost associated with an infinite horizon optimal control problem. In conformance with Sutton's notation, c_t denotes the external evaluation of the cost incurred during the time interval from $t-1$ to t. The prediction P_t, which is the output of the TD learning system, estimates the expected discounted future cost.

$$P_t \approx \sum_{n=0}^{\infty} \gamma^n c_{t+1+n} \tag{5.7}$$

The discount parameter γ specifies the time horizon with which the prediction is concerned. The recursive nature of the expression for an accurate prediction becomes apparent by rewriting (5.7).

$$P_{t-1} = c_t + \sum_{n=1}^{\infty} \gamma^n c_{t+n} = c_t + \gamma \sum_{n=0}^{\infty} \gamma^n c_{t+n+1} = c_t + \gamma P_t \tag{5.8}$$

Therefore, the error in a prediction, $(c_t + \gamma P_t) - P_{t-1}$, serves as the impetus for learning in (5.9).

$$\Delta w_t = \alpha(c_t + \gamma P_t - P_{t-1}) \sum_{k=1}^{t} \lambda^{t-k} \Delta_w P_k \tag{5.9}$$

5.3 A Comparison of Reinforcement Learning Algorithms

The modified TD(λ) rule (5.9) is referred to as the Adaptive Heuristic Critic (AHC) and learns to predict the summation of the discounted future values of the signal c_t. With slightly different learning equations, both Q learning and Drive-Reinforcement (DR) learning accomplish a similar objective. This section compares the form and function of these three direct learning algorithms.

The single step Q learning algorithm (5.10) defines a change in a Q value, which represents a prediction of expected discounted future cost, directly, rather than through adjustments to a set of weights that define the Q value.

$$Q_{t+1}(x_t, a_t) = (1-\alpha)Q_t(x_t, a_t) + \alpha(c_t + \gamma V_t^Q(x_{t+1})) \tag{5.10}$$

Although the form of the learning equation appears different than that of the AHC or DR learning, the functionality is similar. The improved Q value $Q_{t+1}(x_t, a_t)$ equals a linear combination of the initial Q value $Q_t(x_t, a_t)$ and the sum of the cost for the current stage c_t and the discounted prediction of the subsequent discounted future cost $\gamma V_t^Q(x_{t+1})$. A similar linear combination to perform incremental improvements is achieved in both the AHC and Drive-Reinforcement learning by calculating weight changes with respect to the current weights.

Both the Drive-Reinforcement (DR) and the Adaptive Heuristic Critic learning mechanisms calculate weight changes that are proportional to the prediction error ΔP_t.

$$\Delta P_t = c_t + \gamma P_t - P_{t-1} \tag{5.11}$$

The DR learning rule is rewritten in (5.12) to conform to the current notation.

$$\Delta w_t = \Delta P_t \sum_{k=1}^{\tau} c_k \; f_s\,(\Delta x_{t-k}) \tag{5.12}$$

In the DR and AHC algorithms, a non-zero prediction error causes the weights to be adjusted so that P_{t-1} would have been closer to $c_t + \gamma P_t$. The constant of proportionality between the weight change and the prediction error differs between DR learning and the AHC.

The Drive-Reinforcement weight changes are defined by (5.12). The limits on the summation over previous stages in time and the binary facilitation function f_s prescribe modifications to a finite number of previous predictions. An array of constants, c_k, encode a discount that determines the contribution of previous actions to the current prediction error. In contrast, the summation term in the AHC learning equation (5.9) allows all previous predictions to be adjusted in response to a current prediction error. The extent to which an old prediction is modified decreases exponentially with the elapsed time since that prediction. In the AHC algorithm, the sensitivity of the prior prediction to changes in the weights, $\Delta_w P_k$, scales the weight adjustment.

Similar to the AHC and DR learning, an incremental change in a Q value is proportional to the prediction error.

$$\Delta Q_t(x_t, a_t) = Q_{t+1}(x_t, a_t) - Q_t(x_t, a_t) = \alpha \left(c_t - Q_t(x_t, a_t) + \gamma V_t(x_{t+1})\right) \qquad (5.13)$$

The expression for the prediction error in (5.13) appears different from (5.11) and warrants some explanation. $V_t(x_{t+1})$, which represents $max_a\left[Q_t(x_{t+1}, a_{t+1})\right]$, denotes the optimal prediction of discounted future cost and, therefore, is functionally equivalent to P_t in (5.11). Moreover, the entire time basis for Q learning is shifted forward one stage with respect to the AHC or DR learning rules. As a result, Q_t operates similar to P_{t-1} in (5.11) and the symbol c_t performs the same function in (5.11) and (5.13), although the cost is measured over a different period of time in the Q learning rule than in the AHC or DR learning rules (5.9) and (5.12), respectively.

To summarize the comparison of direct learning algorithms, each of the three temporal difference techniques will learn a value function for the expected discounted future cost. More generally, any direct learning algorithm will maintain and incrementally improve both policy and value function information. Furthermore, although the forms of the learning equations differ slightly, each method attempts to reduce the prediction error ΔP_t.

Although the functionality of direct learning algorithms may be similar, the structure will vary. For example, Q learning distinguishes the optimal action by maximizing over the set of Q values. The Associative Control Process determines the optimal action through the biologically motivated reciprocal inhibition procedure. Furthermore, whereas Q values may equal any real number, the outputs of ACP learning centers must be non-negative, acknowledging the inability of neurons to realize negative frequencies of firing.

Chapter 6

Indirect Learning Optimal Control

Each control law derived in this chapter attempts to optimally track a reference trajectory that is generated by a linear, time-invariant reference model (6.1); optimization is performed with respect to quadratic cost functionals over finite time horizons. The notation in this chapter uses subscripts to indicate the stage in discrete time and superscripts to distinguish the plant and reference model.

$$x^r_{k+1} = \Phi^r x^r_k + \Gamma^r r_k \tag{6.1a}$$

$$y^r_k = C^r x^r_k \tag{6.1b}$$

$$y^r_{k+1} = C^r \Phi^r x^r_k + C^r \Gamma^r r_k \tag{6.1c}$$

$$y^r_{k+2} = C^r \Phi^r \Phi^r x^r_k + C^r \Phi^r \Gamma^r r_k + C^r \Gamma^r r_{k+1} \tag{6.1d}$$

Although a few subsequent values of the command input after r_k may be characterized at time k, the future input sequence will be largely unknown. To apply infinite horizon, linear quadratic (LQ) control techniques to the tracking problem requires a description of the future command inputs. Furthermore, in a multi-objective mission, such as aircraft flight involving several different flight conditions, the existence of future maneuvers should negligibly influence the optimization of performance during the current operation. Finally, optimizations over unnecessarily long time frames may require prohibitively long computations. Therefore, finite horizon LQ control directly addresses relevant control problems.

6.1 Single-Stage Quadratic Optimization

The control objective is to minimize the quadratic cost functional J_k which penalizes the current control expenditure and the output error e_{k+1}, given by the difference between the reference output and the system output at time $k+1$. The weighting matrices R and Q are symmetric and positive definite.

$$J_k = \frac{1}{2}\left[e_{k+1}^T Q e_{k+1} + u_k^T R u_k\right] \tag{6.2}$$

$$e_{k+1} = y_{k+1}^r - y_{k+1} \tag{6.3}$$

A single first-order necessary condition defines the condition for a control u_k to minimize the cost functional J_k [22,23].

$$\frac{\partial J_k}{\partial u_k} = 0 \tag{6.4a}$$

$$\left(\frac{\partial e_{k+1}}{\partial u_k}\right)^T Q e_{k+1} + R u_k = 0 \tag{6.4b}$$

6.1.1 Linear Compensation

Assuming a minimum-phase plant, the linear compensator is the solution to the problem of optimal tracking, with respect to the cost functional (6.2), of the reference system (6.1) with a linear, time-invariant system (6.5). Applied to a nonlinear system, this control law serves as a baseline with which to compare a single-stage, indirect learning control law. The fact that indirect learning control is a model based technique distinguishes the approach from direct learning control algorithms.

$$x_{k+1} = \Phi x_k + \Gamma u_k \tag{6.5a}$$

$$y_k = C x_k \tag{6.5b}$$

$$y_{k+1} = C\Phi x_k + C\Gamma u_k \tag{6.5c}$$

That the partial derivative of e_{k+1} with respect to u_k is independent of u_k implies that (6.4b) is linear in the optimal control. Therefore, (6.4b) may be written as an exact analytic expression for the optimal control [24].

$$\frac{\partial e_{k+1}}{\partial u_k} = -C\Gamma \tag{6.6}$$

$$u_k = \left[(C\Gamma)^T Q C\Gamma + R\right]^{-1} (C\Gamma)^T Q \left[C^r \Phi^r x_k^r + C^r \Gamma^r r_k - C\Phi x_k\right] \tag{6.7}$$

The sufficient condition for this control to be a minimizing solution, that $\frac{\partial^2 J_k}{\partial u_k^2}$ is non-negative, is satisfied for physical systems.

$$\frac{\partial J_k}{\partial u_k} = -(C\Gamma)^T Q e_{k+1} + R u_k = 0 \tag{6.8}$$

$$\frac{\partial^2 J_k}{\partial u_k^2} = (C\Gamma)^T Q (C\Gamma) + R \geq 0 \tag{6.9}$$

6.1.2 Learning Control

In contrast to the single-stage linear compensator, the single-stage, indirect learning controller is the solution to the problem of optimal tracking of the reference system (6.1) by a nonlinear, time-invariant system (6.10), with respect to the cost functional (6.2). Again, the zero dynamics of the plant must be stable. The expression for the discrete time state propagation (6.10a) includes the a priori linear terms from (6.5) as well as two nonlinear terms: $f_k(x_k, u_k)$ represents the initially unmodeled dynamics that have been learned by the system, and $\Psi_k(x_k)$ represents any state dependent dynamics not captured by either the a priori description or the learning augmentation. The assumption of an absence of time varying disturbances and noise from the real system implies that all dynamics are spatially dependent and will be represented in the model. The system outputs are a known linear combination of the states. The notation explicitly shows the time dependence of f_k and Ψ_k, which change as learning progresses; f_k will acquire more of the subtleties in the dynamics and, consequently, Ψ_k will approach zero.

$$x_{k+1} = \Phi x_k + \Gamma u_k + f_k(x_k, u_k) + \Psi_k(x_k) \tag{6.10a}$$

$$y_k = Cx_k \tag{6.10b}$$

$$y_{k+1} = C\Phi x_k + C\Gamma u_k + Cf_k(x_k, u_k) + C\Psi_k(x_k) \tag{6.10c}$$

In this formulation, the first-order necessary condition (6.4) for a control u_k to be optimal with respect to (6.2) cannot be directly solved for u_k because of the presence of the term $f_k(x_k, u_k)$ which may be nonlinear in u_k. The Newton-Raphson iterative technique [25] addresses this nonlinear programming problem by linearizing $f_k(x_k, u_k)$ with respect to u at u_{k-1}. $\frac{\partial f}{\partial u}$ is the Jacobian matrix of f_k with respect to u, evaluated at $\{x_k, u_{k-1}\}$. Using this approximation for $f_k(x_k, u_k)$, y_{k+1} assumes a form linear in u_k and (6.4) may be written as an analytic expression for u_k in terms of known quantities.

$$f_k(x_k, u_k) \approx f_k(x_k, u_{k-1}) + \left.\frac{\partial f}{\partial u}\right|_{x_k, u_{k-1}} (u_k - u_{k-1}) \tag{6.11}$$

$$y_{k+1} \approx C\Phi x_k + C\Gamma u_k + Cf_k(x_k, u_{k-1}) + C\left.\frac{\partial f}{\partial u}\right|_{x_k, u_{k-1}} (u_k - u_{k-1}) + C\Psi_k(x_k) \tag{6.12}$$

$$\frac{\partial e_{k+1}}{\partial u_k} = -C\Gamma - C\left.\frac{\partial f}{\partial u}\right|_{x_k, u_{k-1}} \tag{6.13}$$

The solution (6.14) is the first Newton-Raphson estimate for the optimal controls; a pseudo-inverse may be used in (6.14) if the full matrix inversion does not exist. Subsequent estimates u_k^i may be derived by linearizing (6.10) about u_k^{i-1}. However, the estimate obtained in the first iteration is often sufficiently accurate because the initial linearization is about u_{k-1} and the admissible change in control $\Delta u_k = u_k - u_{k-1}$ will be limited by actuator rate limits and a sufficiently small discrete time interval [25,26].

The form of the learning augmented control law closely resembles the linear control law (6.7). In (6.14) $C\Gamma$ is modified by $\frac{\partial f}{\partial u}$ which describes the variation in control effectiveness that was unmodeled in the linear system. The final three terms of (6.14) are not present in (6.7) and enter from the nonlinear terms in (6.10).

$$\begin{aligned} u_k^1 = &\left[\left(C\Gamma + C\frac{\partial f}{\partial u}\right)^T Q \left(C\Gamma + C\frac{\partial f}{\partial u}\right) + R\right]^{-1} \\ &\left(C\Gamma + C\frac{\partial f}{\partial u}\right)^T Q \Big[C^r\Phi^r x_k^r + C^r\Gamma^r r_k - C\Phi x_k \\ &- Cf_k(x_k, u_{k-1}) + C\frac{\partial f}{\partial u}u_{k-1} - C\Psi_k(x_k)\Big] \end{aligned} \tag{6.14}$$

A simple adaptive estimate for the unmodeled dynamics at time k is generated by solving (6.10) at the previous time index for $\Psi_{k-1}(x_{k-1})$ and assuming $\Psi_k(x_k) \approx \Psi_{k-1}(x_{k-1})$. This adaptive technique is susceptible to noise and disturbances present in the real system [27].

$$\hat{\Psi}_k(x_k) = x_k - \Phi x_{k-1} - \Gamma u_{k-1} - f_k(x_{k-1}, u_{k-1}) \tag{6.15}$$

6.1.3 Penalizing Control Rate

A parallel set of arguments may be used to derive a control law that is optimal with respect to a cost functional that also penalizes changes in control. The $\Delta u_k^T S \Delta u_k$ component in (6.16) discourages large control rates that may not be achievable, as a result of physical limitations of the actuators. The control law

(6.18) resembles (6.14) with the addition of two terms involving S, which is symmetric and positive definite.

$$J_k = \frac{1}{2}\left[e_{k+1}^T Q e_{k+1} + u_k^T R u_k + \Delta u_k^T S \Delta u_k\right] \tag{6.16}$$

$$\Delta u_k = u_k - u_{k-1} \tag{6.17}$$

$$\begin{aligned} u_k = &\left[\left(C\Gamma + C\frac{\partial f}{\partial u}\right)^T Q \left(C\Gamma + C\frac{\partial f}{\partial u}\right) + R + S\right]^{-1} \\ &\Bigg[\left(C\Gamma + C\frac{\partial f}{\partial u}\right)^T Q\Big(C^r\Phi^r x_k^r + C^r\Gamma^r r_k - C\Phi x_k \\ &- Cf_k(x_k, u_{k-1}) + C\frac{\partial f}{\partial u}u_{k-1} - C\Psi_k(x_k)\Big) + Su_{k-1}\Bigg] \end{aligned} \tag{6.18}$$

6.2 Two-Step Quadratic Optimization

This section parallels the discussion of §6.1 to derive the control laws that are optimal with respect to a two time step, quadratic cost functional (6.19); a few new issues arise. The expression for the reference output two time steps into the future (6.1d) involves a future value of the command input r_{k+1}. This derivation assumes that $r_{k+1} \approx r_k$.

$$J_k = \frac{1}{2}\left[e_{k+1}^T Q^1 e_{k+1} + e_{k+2}^T Q^2 e_{k+2} + u_k^T R^0 u_k + u_{k+1}^T R^1 u_{k+1}\right] \tag{6.19}$$

Two necessary conditions are required to define a control which is optimal with respect to (6.19). Each of the weighting matrices in (6.19) is symmetric and positive definite.

$$\frac{\partial J_k}{\partial u_k} = 0 \tag{6.20a}$$

$$\left(\frac{\partial e_{k+1}}{\partial u_k}\right)^T Q^1 e_{k+1} + \left(\frac{\partial e_{k+2}}{\partial u_k}\right)^T Q^2 e_{k+2} + R^0 u_k = 0 \tag{6.20b}$$

$$\frac{\partial J_k}{\partial u_{k+1}} = 0 \tag{6.21a}$$

$$\left(\frac{\partial e_{k+1}}{\partial u_{k+1}}\right)^T Q^1 e_{k+1} + \left(\frac{\partial e_{k+2}}{\partial u_{k+1}}\right)^T Q^2 e_{k+2} + R^1 u_{k+1} = 0 \tag{6.21b}$$

6.2.1 Linear Compensation

The output of the linear system (6.5) two time steps into the future is easily determined because the dynamics are assumed to be entirely known.

$$y_{k+2} = C\Phi\Phi x_k + C\Phi\Gamma u_k + C\Gamma u_{k+1} \tag{6.22}$$

$$e_{k+2} = y^r_{k+2} - y_{k+2} \tag{6.23}$$

The simultaneous solution of (6.20b) and (6.21b) yields a solution for u_k; an expression for u_{k+1}, calculated at time k, is also available. However, to parallel the learning control process, this control will be recalculated at the next time step. To illustrate the similarity of the linear and learning control laws, as well as to express the laws in a compact form, several substitutions are defined.

$$A = C\Gamma \tag{6.24a}$$

$$B = C\Phi\Gamma \tag{6.24b}$$

$$\Lambda = A^T Q^2 A + R^1 \tag{6.25a}$$

$$\Upsilon = B^T Q^2 A \tag{6.25b}$$

$$\Theta = A^T Q^1 \tag{6.25c}$$

$$\Xi = (B^T - \Upsilon \Lambda^{-1} A^T) Q^2 \tag{6.25d}$$

$$\begin{aligned} u_k = &\left[A^T Q^1 A + B^T Q^2 B + R^0 - \Upsilon \Lambda^{-1} \Upsilon^T\right]^{-1} \\ &\Big[(\Theta\, C^r \Phi^r + \Xi\, C^r \Phi^r \Phi^r)\, x_k^r \\ &+ (\Theta\, C^r \Gamma^r + \Xi\, (C^r \Phi^r \Gamma^r + C^r \Gamma^r))\, r_k \\ &- (\Theta\, C\Phi + \Xi\, C\Phi\Phi)\, x_k\Big] \end{aligned} \tag{6.26}$$

6.2.2 Learning Control

For the nonlinear system, the output y_{k+2} (6.27a) must be approximated by known quantities that are linear in u_k and u_{k+1}. First, the nonlinear terms in (6.27a) are evaluated at the current time k rather than at $k+1$ and an approximation $\hat{x}_{k+1}$ is derived for the next state. Additionally, the learned dynamics are estimated by linearizing $f_k(x_{k+1}, u_{k+1})$ about the point $\{x_k, u_{k-1}\}$.

$$\begin{aligned} y_{k+2} &= C\Phi x_{k+1} + C\Gamma u_{k+1} + Cf_{k+1}(x_{k+1}, u_{k+1}) + C\Psi_{k+1}(x_{k+1}) && (6.27a) \\ &\approx C\Phi \hat{x}_{k+1} + C\Gamma u_{k+1} + Cf_k(x_{k+1}, u_{k+1}) + C\Psi_k(\hat{x}_{k+1}) && (6.27b) \end{aligned}$$

$$\hat{x}_{k+1} = \Phi x_k + \Gamma u_k + f_k(x_k, u_{k-1}) + \left.\frac{\partial f}{\partial u}\right|_{x_k, u_{k-1}} (u_k - u_{k-1}) + \Psi_k(x_k) \qquad (6.28)$$

$$f_k(x_{k+1}, u_{k+1}) \approx f_k(x_k, u_{k-1}) + \left.\frac{\partial f}{\partial u}\right|_{x_k, u_{k-1}} (u_{k+1} - u_{k-1}) + \left.\frac{\partial f}{\partial x}\right|_{x_k, u_{k-1}} (\hat{x}_{k+1} - x_k) \quad (6.29)$$

$$\begin{aligned} y_{k+2} \approx{} & C\Phi \hat{x}_{k+1} + C\Gamma u_{k+1} + Cf_k(x_k, u_{k-1}) + C\left.\frac{\partial f}{\partial u}\right|_{x_k, u_{k-1}} (u_{k+1} - u_{k-1}) \\ & + C\left.\frac{\partial f}{\partial x}\right|_{x_k, u_{k-1}} (\hat{x}_{k+1} - x_k) + C\Psi_k(\hat{x}_{k+1}) && (6.30) \end{aligned}$$

Using this approximation for y_{k+2}, the simultaneous solution of (6.20b) and (6.21b) yields an expression for u_k in terms of (6.25) and (6.31). The variables A and B include both the linear components of the a priori model as well as learned state dependent corrections. $\frac{\partial f}{\partial u}$ is a correction to the constant Γ matrix and $\frac{\partial f}{\partial x}$ is a correction to the constant Φ matrix.

$$\begin{aligned} A &= C\Gamma + C\frac{\partial f}{\partial u} && (6.31a) \\ B &= C\Phi\Gamma + C\Phi\frac{\partial f}{\partial u} + C\frac{\partial f}{\partial x}\Gamma + C\frac{\partial f}{\partial u}\frac{\partial f}{\partial x} && (6.31b) \end{aligned}$$

Although the simultaneous solution of the first-order necessary conditions also yields an expression for u_{k+1} at k, u_k is calculated on every time step. This control law resembles the form of (6.26) and introduces several terms associated with the nonlinear dynamics.

$$\begin{aligned} u_k = &\left[A^T Q^1 A + B^T Q^2 B + R^0 - \Upsilon \Lambda^{-1} \Upsilon^T\right]^{-1} \\ &\Big[(\Theta\, C^r \Phi^r + \Xi\, C^r \Phi^r \Phi^r)\, x_k^r \\ &+ (\Theta\, C^r \Gamma^r + \Xi\, (C^r \Phi^r \Gamma^r + C^r \Gamma^r))\, r_k \\ &- \left(\Theta\, C\Phi + \Xi\, \left(C\Phi\Phi + C\frac{\partial f}{\partial x}\Phi - C\frac{\partial f}{\partial x}\right)\right) x_k \\ &+ \left(\Theta\, C\frac{\partial f}{\partial u} + \Xi\, \left(C\Phi\frac{\partial f}{\partial u} + C\frac{\partial f}{\partial u} + C\frac{\partial f}{\partial x}\frac{\partial f}{\partial u}\right)\right) u_{k-1} \\ &- \left(\Theta\, C + \Xi\, \left(C\Phi + C + C\frac{\partial f}{\partial x}\right)\right) f_k(x_k, u_{k-1}) \\ &- \left(\Theta\, C + \Xi\, \left(C\Phi + C\frac{\partial f}{\partial x}\right)\right) \Psi_k(x_k) \\ &- \Xi\, C\Psi_k(\hat{x}_{k+1})\Big] \end{aligned} \tag{6.32}$$

6.2.3 Multi-stage Quadratic Optimization

The arguments presented in §6.1 and thus far in §6.2 may be generalized to derive a control law which is optimal with respect to a cost functional (6.33) that looks n time steps into the future. The solution of this problem, however, will require assumptions about the propagation of the command input r for n future time steps. Additionally, the algebra required to write an explicit expression for u_k becomes involved and the necessary linearizations become less accurate.

$$J_k = \frac{1}{2}\sum_{i=1}^{n}\left[e_{k+i}^T Q^i e_{k+i} + u_{k+i-1}^T R^i u_{k+i-1} + \Delta u_{k+i-1}^T S^i \Delta u_{k+i-1}\right] \tag{6.33}$$

6.3 Implementation and Results

6.3.1 Reference Model

The reference model, which generates trajectories that the plant states attempt to follow, exhibits a substantial influence on the closed-loop system performance. While a reference model that does not satisfy performance specifications will yield an unacceptable closed-loop system, a reference model that demands unrealistic (i.e. unachievable) state trajectories may introduce instability through control saturation. Therefore, the reference model must be selected to yield satisfactory performance given the limitations of the dynamics [28].

The reference model was selected to be the closed-loop system that resulted from applying the optimal control from a linear quadratic design, to the aeroelastic oscillator dynamics linearized at the origin [29]. The discrete time representation of the reference model as well as the AEO linear dynamics are presented for $\Delta t = 0.1$.

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.34a}$$

$$R = 1.0 * 10^{-7} \tag{6.34b}$$

$$C = C^r = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix} \tag{6.35}$$

$$\Phi = \begin{bmatrix} 0.994798 & 0.106070 \\ -0.106070 & 1.122083 \end{bmatrix} \tag{6.36a}$$

$$\Gamma = \begin{bmatrix} 0.005202 \\ 0.106070 \end{bmatrix} \tag{6.36b}$$

$$\Phi^r = \begin{bmatrix} 0.905124 & 0.000286 \\ -0.905124 & -0.000286 \end{bmatrix} \tag{6.37a}$$

$$\Gamma^r = \begin{bmatrix} 0.094876 \\ 0.905124 \end{bmatrix} \tag{6.37b}$$

Design of an optimal control law might be accomplished with a learning system that incrementally increases closed-loop performance requirements, by adjusting the reference trajectory in regions of the state space where the current control law can achieve near perfect tracking. This is a topic for future research.

6.3.2 Function Approximation

The discussion of direct learning optimal control (§3 – §5) focused on learning system architectures and algorithms which, themselves, operate as controllers. The discussion of indirect learning optimal control is primarily concerned with the manner in which experiential information about unmodeled dynamics may be incorporated into optimal control laws. The method by which a supervised learning system approximates the initially unmodeled dynamics is a separate issue which has received much investigation [21,30,31,32].

After a brief summary, this thesis abstracts the technique for realizing the nonlinear mapping $f(x,u)$ into a *black box* which provides the desired information: $f_k(x_k,u_{k-1})$, $\frac{\partial f_k}{\partial u}|_{x_k,u_{k-1}}$, and $f_k(x_{k-1},u_{k-1})$.

A spatially localized connectionist network is used to represent the mapping from the space of states and control to the initially unmodeled dynamics. The linear-Gaussian network achieves spatial locality by coupling a local basis function with an influence function [4,28]. The influence function, which determines the region in the input space of applicability of the associated basis function, is a hyper-Gaussian; the basis function is a hyperplane.

The contribution of a basis function to the network output equals the product of the basis function and the influence function, evaluated at the current input, where the influence function is normalized so that the sum of all the influence functions at the current input is unity [28]. The control law provides to the network an estimate of the model errors, $x_k - \Phi x_{k-1} - \Gamma u_{k-1}$. The supervised learning procedure follows an incremental gradient descent algorithm in the space of the network errors by adjusting the parameters that describe the slopes and offsets of the basis functions.

In terms of equations and for arbitrary input and output dimensions, $Y(x)$ is the network output evaluated at the current input x. The network consists of n nodes (i.e. basis function and influence function pairs).

$$Y(x) = \sum_{i=1}^{n} L_i(x)\Gamma_i(x) \tag{6.38a}$$

$L_i(x)$ is the evaluation of the i^{th} basis function at the current input. W_i is a weight matrix that defines the slopes of the hyperplane and b_i is a bias vector. x_0 defines the center of the influence function.

$$L_i(x) = W_i(x - x_0) + b_i \tag{6.38b}$$

$\Gamma_i(x)$ is the i^{th} normalized influence function and is not related to the discrete time B matrix. $G_i(x)$ is the i^{th} influence function evaluated at x, where the diagonal matrix D_i represents the spatial decays of the Gaussians.

$$\Gamma_i(x) = \frac{G_i(x)}{\sum_{j=1}^{n} G_j(x)} \tag{6.38c}$$

$$G_i(x) = \exp\left[-\frac{1}{2}(x - x_0)^T D_i^2 (x - x_0)\right] \tag{6.38d}$$

The learning network had three inputs (position, velocity, and control) and two outputs (unmodeled position and velocity dynamics). In addition, the partial derivatives of the system outputs with respect to the inputs were available.

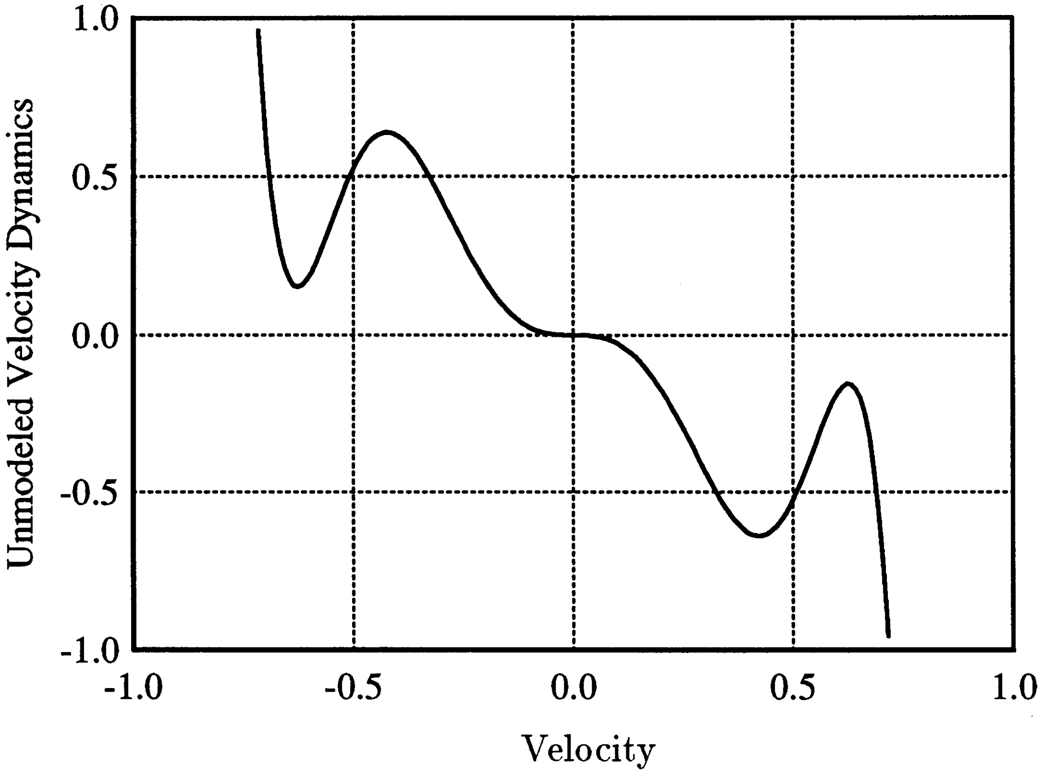

Figure 6.1. The initially unmodeled velocity dynamics $g(x_2)$ as a function of velocity x_2.

The AEO dynamics are repeated in (6.39). The learning system must synthesize on-line the initial model uncertainty, which consists of the nonlinear term $g(x_2)$ in the velocity dynamics.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & nA_1U - 2\beta \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} F + \begin{bmatrix} 0 \\ g(x_2) \end{bmatrix} \tag{6.39a}$$

$$g(x_2) = \frac{1}{1000}\left[-\frac{nA_3}{U}(1000x_2)^3 + \frac{nA_5}{U^3}(1000x_2)^5 - \frac{nA_7}{U^5}(1000x_2)^7\right] \tag{6.39b}$$

The manner in which the position and control enter the dynamics is linear and perfectly modeled. Therefore, the function f will be independent of the position and control (Figure 6.1). Additional model errors may be introduced to the a priori model by altering the coefficients that describe how the state and control enter the linear dynamics. The learning system will approximate all model uncertainty.

Although the magnitude of the control had been limited in the direct learning control results, where the reinforcement signal was only a function of the state error, limitation of the control magnitude was not necessary for indirect learning controllers because control directly entered the cost functional.

6.3.3 Single-Stage Quadratic Optimization Results

For the minimization of the weighted sum of the squares of the current control and the succeeding output error, the performance of the learning enhanced control law (6.14) was compared to the associated linear control law (6.7), in the context of the aeroelastic oscillator. Results appear in Figures 6.2 through 6.9. The control and reference model were updated at 10 Hz; the AEO simulation was integrated using a fourth-order Runge-Kutta algorithm with a step size of 0.005.

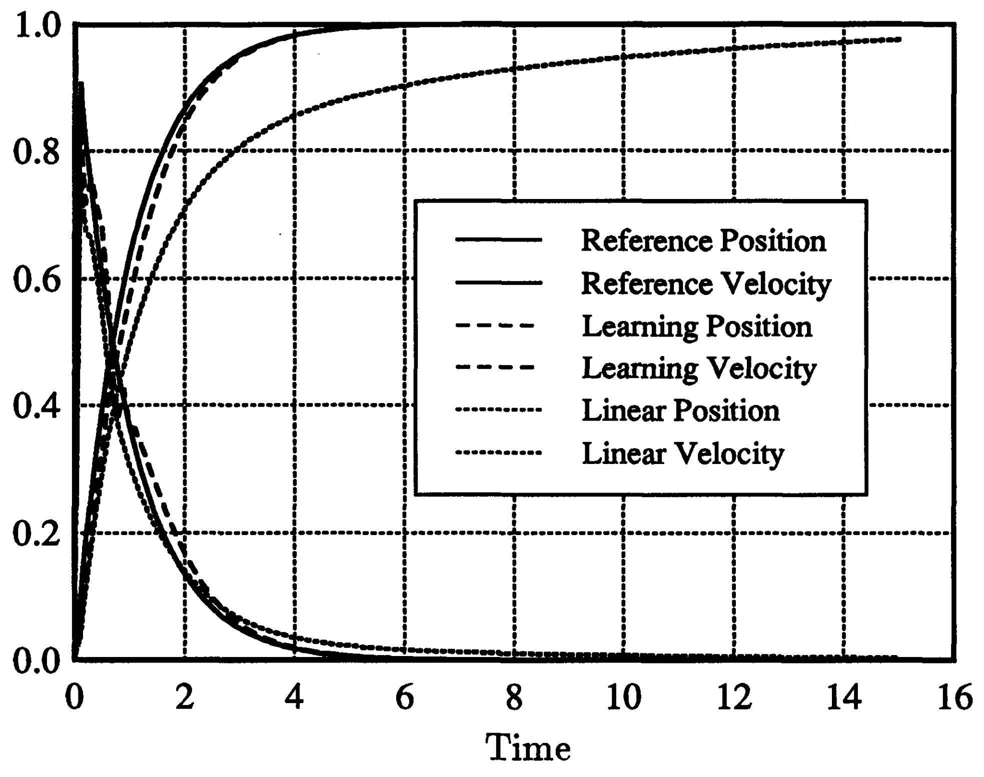

Figure 6.2. Position and velocity time histories for the reference model as well as the AEO controlled by the linear and learning control laws, for the command $r = 1$ and the initial condition $\underline{x}_0 = \{0, 0\}$.

Figure 6.2 illustrates the reference model position and velocity time histories as well as two sets of state time histories for the AEO controlled by the linear and learning control laws. The presence of unmodeled nonlinear dynamics prevented the linear control law from closely tracking the reference position. In contrast, the learning system closely followed the reference, after sufficient training. Both control laws maintained the velocity near the reference. Although the full learning control law (6.14) was implemented to produce these results, knowledge of the form of the AEO nonlinearities could have been used to eliminate the terms containing $\frac{\partial f}{\partial u}$. Figure 6.3 represents the errors between the AEO states and the reference

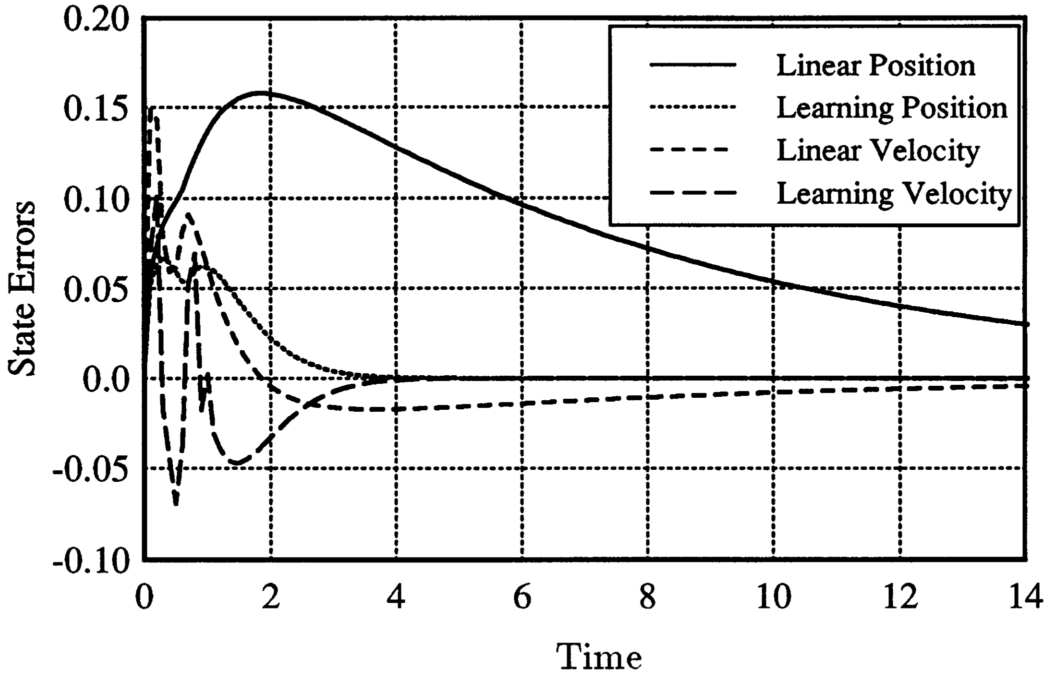

Figure 6.3. The state errors for the AEO controlled by the linear and learning control laws.

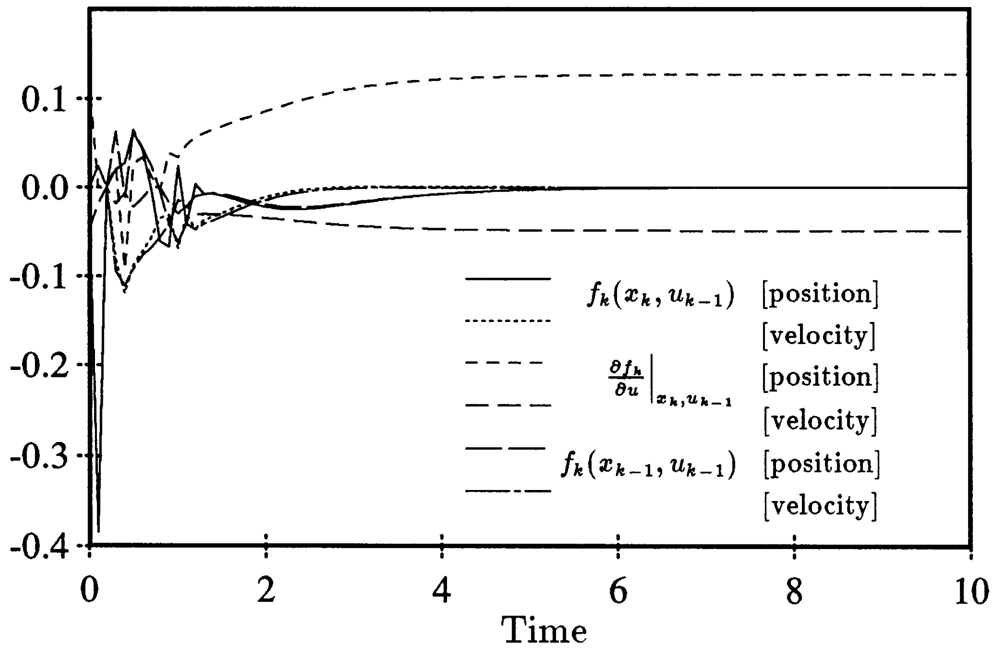

Figure 6.4. The network outputs that were used to compute u_k for the learning control law.

model, for both control law designs. In a separate experiment that introduced model uncertainty in the linear dynamics, the linear control law (6.7) failed to track a command step input with zero steady-state error. The learning control law results looked similar to Figure 6.2.

The specifics of the incremental function approximation were not a focus of this indirect learning control research. The learning process involved numerous trials (more than 1000) from random initial states within the range $\{-1.0, 1.0\}$; the commanded position was also selected randomly between the same limits. The allocation of network resources (i.e. adjustable parameters) and the selection of learning rates involved heuristics. Moreover, the learning performance depended strongly on these decisions. Automation of the network design process would have greatly facilitated this research.

The learning control law requires the values of the network outputs at the current state and the previous control, as well as the partial derivative of the network outputs with respect to the control, at the current state and the previous control. Additionally, the adaptive term $\Psi_k(x_k)$ requires the values of the network outputs at the previous state and the previous control. The network outputs, which appear in Figure 6.4, change most rapidly when the velocity is not near zero, i.e. at the beginning of a trial. Some rapid changes in the network outputs resulted from learning errors where f did not accurately approximate the nonlinear dynamics.

For the learning control law, the control as well as the terms of (6.14) that comprise the control appear in Figure 6.5. The control for the linear control law and the individual terms of (6.7) appear in Figure 6.6.

After substantial training, some errors remained in the network's approxima-

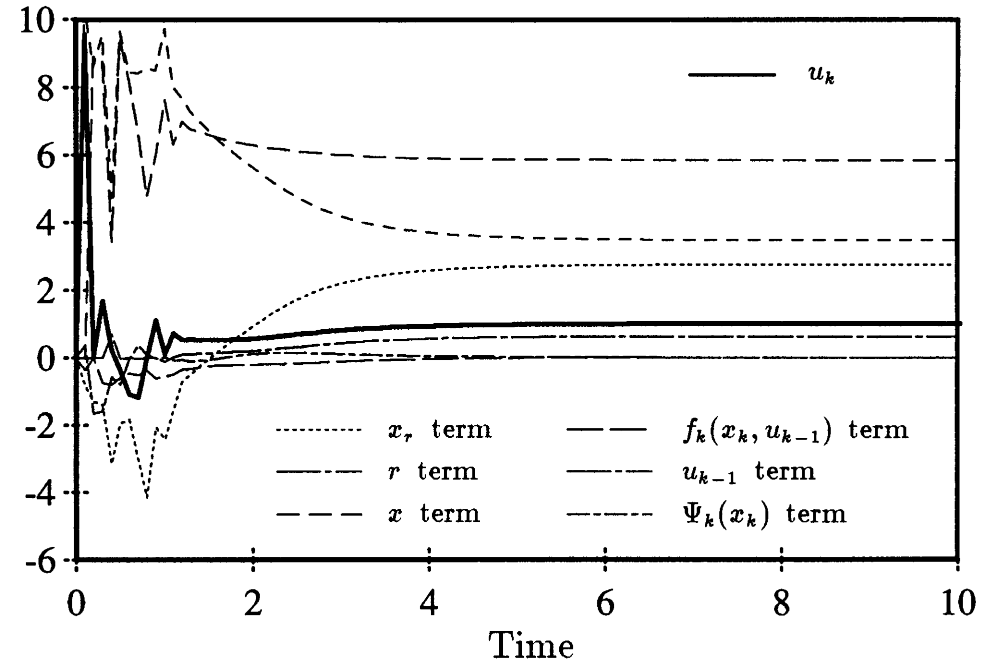

Figure 6.5. The control u_k and the constituent terms of the learning control law (6.14).

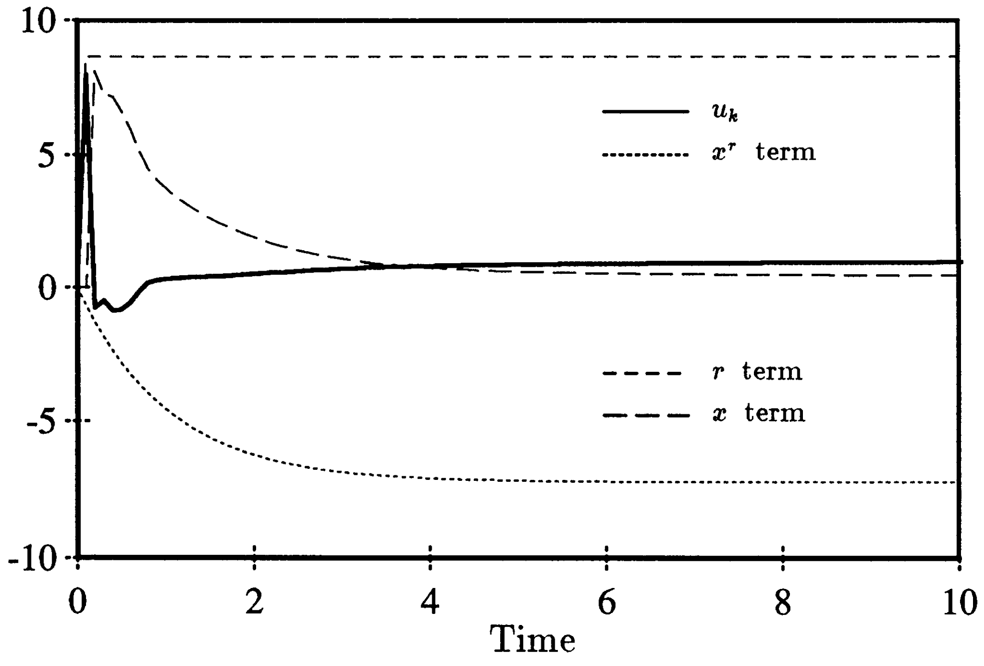

Figure 6.6. The control u_k and the constituent terms of the linear control law (6.7).

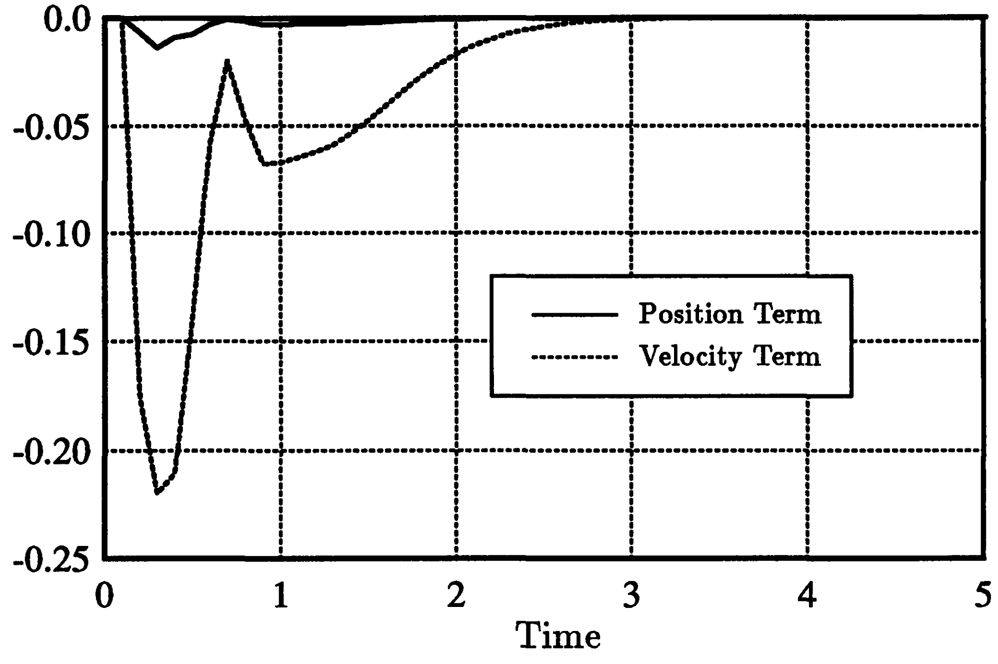

Figure 6.7. The estimated errors in the approximation of the initially unmodeled dynamics $f_k(x_k, u_{k-1})$.

tion of the nonlinear dynamics. These errors are most noticeable in the estimation of the velocity dynamics at velocities not near zero. Figure 6.8 illustrates the initial model errors not represented by the function f; the adaptive term will reduce the effect of these remaining model errors. Experiments demonstrated that the system performed nearly as well when the adaptive contribution was removed from the control. A controller that augmented the linear law with only the adaptive correction was not evaluated.

Figure 6.8 shows the results of control laws (6.14) and (6.7) regulating the AEO from $\underline{x}_0 = \{-1.0, 0.5\}$. The control magnitude was limited at 0.5 so that the results may be compared more easily to the benchmarks and the direct learning control results. Time is not explicitly shown in Figure 6.8; the state trajectory produced by

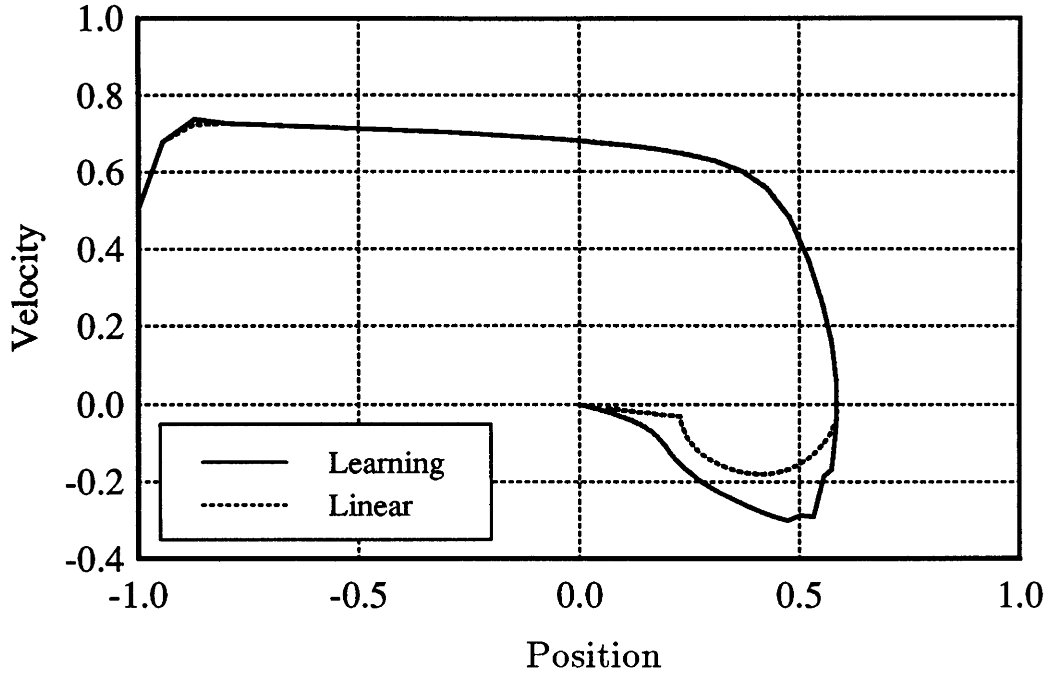

Figure 6.8. AEO Regulation from $\underline{x}_0 = \{-1.0, 0.5\}$ with control saturation at ± 0.5.

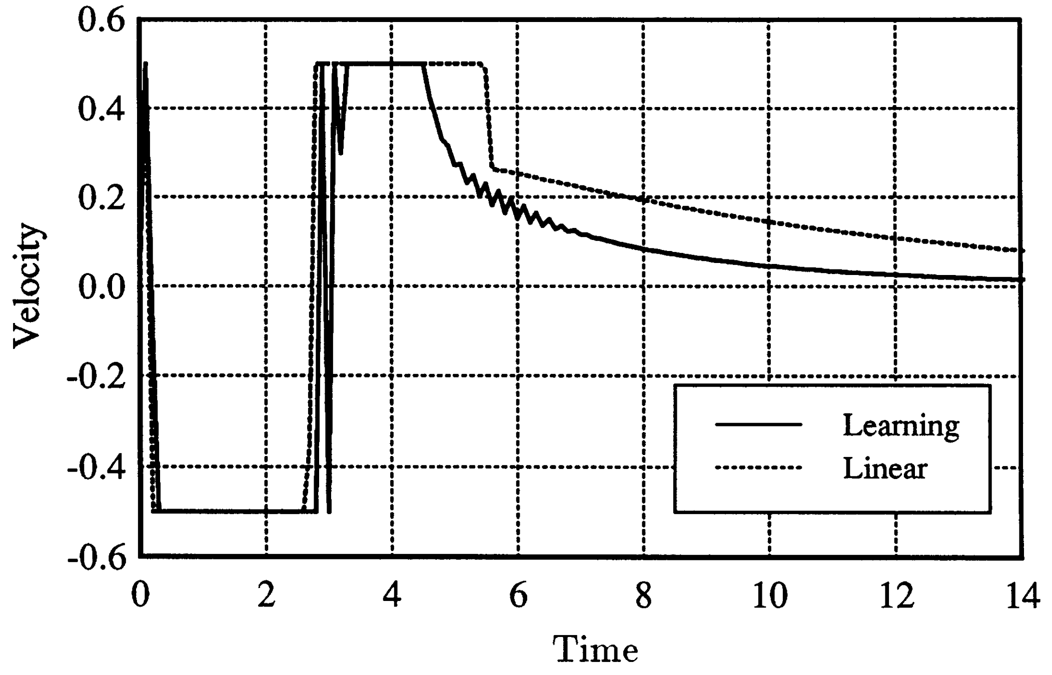

Figure 6.9. Control history associated with Figure 6.8.

the learning controller approached the origin much more quickly than the trajectory produced by the linear controller. The control objective remains to track a reference trajectory and, therefore, subtly differs from the goal of LQR (Figure 2.5). Recall that this reference model does not necessarily maximize system performance. Figure 6.9 shows the force histories which yielded the trajectories in Figure 6.8. The rapid switching in the learning control force results from learning errors and the sensitivity of the control law to the approximated Jacobian of $f_k(x_k, u_{k-1})$.

This indirect learning control technique was capable of learning, and therefore reducing the effect of, model uncertainty (linear and nonlinear). Therefore, the indirect learning controller derived from a linear model with model errors performed similar to Figure 6.8 and outperformed the LQR solution which was derived from an inaccurate linear model (Figure 2.7). The indirect learning controller with limited control authority produced state trajectories similar to the results of the direct learning control experiments.

Chapter 7

Summary

The aeroelastic oscillator demonstrated interesting nonlinear dynamics and served as an acceptable context in which to evaluate the capability of several direct and indirect learning controllers.

The ACP network was introduced to illustrate the biological origin of reinforcement learning techniques and to provide a foundation from which to develop the modified two-layer and single layer ACP architectures. The modified two-layer ACP introduced refinements that increased the architecture's applicability to the infinite horizon optimal control problem. However, results demonstrated that, for the defined plant and environment, this algorithm failed to synthesize an optimal control policy. Finally, the single layer ACP, which functionally resembled Q learning, successfully constructed an optimal control policy that regulated the aeroelastic oscillator.

Q learning approaches the direct learning paradigm from the mathematical theory of value iteration rather than from behavioral science. With sufficient training, the Q learning algorithm converged to a set of Q values that accurately described the expected discounted future return for each state-action pair. The optimal policy that was defined by these Q values successfully regulated the aeroelastic oscillator plant. The results of the direct learning algorithms (e.g. the ACP derivatives and Q learning) demonstrated the limitations of optimal control laws that are restricted to discrete controls and a quantized input space. The concept of extending Q learning to accommodate continuous inputs and controls was considered. However, the necessary maximization at each time step of a continuous, potentially multi-modal Q function may render impractical an on-line implementation of a continuous Q learning algorithm.

The optimal control laws for single-stage and two-step finite time horizon, quadratic cost functionals were derived for linear and nonlinear system models. The results of applying these control laws to cause the AEO to optimally track a linear reference model demonstrated that indirect learning control systems, which incorporate information about the unmodeled dynamics that is incrementally learned, outperform fixed parameter, linear control laws. Additionally, operating with continuous inputs and outputs, indirect learning control methods provide better performance than the direct learning methods previously mentioned. A spatially localized connectionist network was employed to construct the approximation of the initially unmodeled dynamics that is required for indirect learning control.

7.1 Conclusions

This thesis has collected several direct learning optimal control algorithms and has also introduced a class of indirect learning optimal control laws. In the process of investigating direct learning optimal controllers, the commonality between an algorithm originating in behavioral science and another founded in mathematical optimization help unify the concept of direct learning optimal control. More generally, this thesis has "drawn arrows" to illustrate how a variety of learning control concepts are related. Several learning systems were applied as controllers for the aeroelastic oscillator.

7.1.1 Direct / Indirect Framework

As a means of classifying approaches to learning optimal control laws, a *direct/indirect* framework was introduced. Both direct and indirect classes of learning controllers were shown to be capable of synthesizing optimal control laws, within the restrictions of the particular method being used. Direct learning control implies the feedback loop that motivates the learning process is closed around system performance. This approach is largely limited to discrete inputs and outputs. Indirect learning control denotes a class of incremental control law synthesis methods for which the learning loop is closed around the system model. The indirect learning control laws derived in §6 are not capable of yielding stable closed-loop systems for non-minimum phase plants.

As a consequence of closing the learning loop around system performance, direct learning control procedures acquire information about control saturation. Indirect learning control methods will learn the unmodeled dynamics as a function of the applied control, but will not "see" control saturation which occurs external to the control system.

7.1.2 Comparison of Reinforcement Learning Algorithms

The learning rules for the Adaptive Heuristic Critic (a modified TD(λ) procedure), Q learning, and Drive-Reinforcement learning (the procedure used in the ACP reinforcement centers) were compared. Each learning system was shown to predict an expected discounted future reinforcement. Moreover, each learning rule was shown to adjust the previous predictions in proportion to a prediction error that was the difference between the current reinforcement and the difference between the previous expected discounted future reinforcement and the discounted current expected discounted future reinforcement. The constants of proportionality describe the reduced importance of events that are separated by longer time intervals.

7.1.3 Limitations of Two-layer ACP Architectures

The limitations of the two-layer ACP architectures arise primarily from the simultaneous operation of two opposing reinforcement centers. The distinct positive and negative reinforcement centers, which are present in the two-layer ACP, incrementally improve estimates of the expected discounted future reward and cost, respectively. The optimal policy is to select, for each state, the action that maximizes the difference between the expected discounted future reward and cost. However, the two-layer ACP network performs reciprocal inhibition between the two reinforcement centers. Therefore, the information passed to the motor centers effects the selection of a control action that either maximizes the estimate of expected discounted future reward, or minimizes the estimate of expected discounted future cost. In general, a two-layer ACP architecture will not learn the optimal policy.

7.1.4 Discussion of Differential Dynamic Programming

For several reasons, differential dynamic programming (DDP) is an inappropriate approach for solving the problem described in §1.1. First, the DDP algorithm yields a control policy only in the vicinity of the nominally optimal trajectory. Extension of the technique to construct a control law that is valid throughout the state space is tractable only for linear systems and quadratic cost functionals. Second, the DDP algorithm explicitly requires, as does dynamic programming, an accurate model of the plant dynamics. Therefore, for plants with initially unknown dynamics, a system identification procedure must be included. The coordination of the DDP algorithm with a learning systems that incrementally improves the system model would constitute an indirect learning optimal controller. Third, since the quadratic approximations are valid only in the vicinity of the nominal state and control trajectories, the DDP algorithm may not extend to stochastic control problems for which the process noise is significant. Fourth, similar to Newton's nonlinear programming method, the original DDP algorithm will converge to a globally optimal solution only if the initial state trajectory is sufficiently close to the optimal state trajectory.

7.2 Recommendations for Future Research

Several aspects of this research warrant additional thought. The extension of direct learning methods to continuous inputs and continuous outputs might be an ambitious endeavor. Millington [41] addressed this issue by using a spatially localized connectionist / Analog Learning Element that defined, as a distributed function of state, a continuous probability density function for control selection. The learning procedure increased the probability of selecting a control that yielded, with a high probability, a large positive reinforcement. The difficulty of generalizing the Q learning algorithm to continuous inputs and outputs has previously been discussed.

The focus of indirect learning control research should be towards methods of incremental function approximation. The accuracy of the learned Jacobian of the unmodeled dynamics critically impacts the performance of indirect learning optimal control laws. The selection of network parameters (e.g. learning rates, the number of nodes, and the influence function centers and spatial decay rates) determines how successfully the network will map the initially unmodeled dynamics. The procedure that was used for the selection of parameters was primarily heuristic. Automation of this procedure could improve the learning performance and facilitate the control law design process. Additionally, indirect learning optimal control methods should be applied to problems with a higher dimension. The closed-loop system performance as well as the difficulty of the control law design process should be compared with a gain-scheduled linear approach to control law design.

Appendix A

Differential Dynamic Programming

A.1 Classical Dynamic Programming

Differential dynamic programming (DDP) shares many features with the classical dynamic programming (DP). For this reason, and because dynamic programming is a more recognized algorithm, this chapter begins with a summary of the dynamic programming algorithm. R. E. Bellman introduced the classical dynamic programming technique, in 1957, as a method to determine the control function that minimizes a performance criterion [33]. Dynamic programming, therefore, serves as an alternative to the calculus of variations, and the associated two-point boundary value problems, for determining optimal controls.

Starting from the set of state and time pairs which satisfy the terminal conditions, the dynamic programming algorithm progresses backward in discrete time. To accomplish the necessary minimizations, dynamic programming requires a quantization of both the state and control spaces. At each discrete state, for every stage in time, the optimal action is the action which yields the minimum cost to complete the problem. Employing the principle of optimality, this completion cost from a given discrete state, for a particular choice of action, equals the sum of the cost associated with performing that action and the minimum cost to complete the problem from the resulting state [23]. $J_t^*(\underline{x})$ equals the minimum cost to complete a problem from state $\underline{x}$ and discrete time t, $g(\underline{x}, \underline{u}, t)$ is the incremental cost function, where $\underline{u}$ is the control vector, and $\underline{T}(\underline{x}, \underline{u}, t)$ is the state transition function. Further, define a mapping from the state to the optimal controls, $S(\underline{x}; t) = \underline{u}(t)$ where $\underline{u}(t)$ is the argument that minimizes the right side of (A.1).

$$J_t^*(\underline{x}(t)) = \min_{\underline{u}(t)} \left[g(\underline{x}(t), \underline{u}(t), t) + J_{t+1}^*(\underline{T}(\underline{x}(t), \underline{u}(t), t)) \right] \qquad (A.1)$$

The principle of optimality substantially increases the efficiency of the dynamic programming algorithm to construct $S(\underline{x}; t)$ with respect to an exhaustive search, and is described by Bellman and S. E. Dreyfus.

> An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision [34].

The backward recursion process ends with the complete description of $S(\underline{x}; t)$ for all states and for $t = N-1$, $N-2$, ... 1, where N is the final time. Given the initial state $\underline{x}^*(1) = \underline{x}(1)$, (A.2) defines the forward DP recursion step.

$$\underline{u}^*(t) = S(\underline{x}^*; t) \qquad (A.2a)$$

$$\underline{x}^*(t+1) = \underline{T}(\underline{x}^*, \underline{u}^*, t) \qquad (A.2b)$$

Although dynamic programming provides a general approach to optimal control problems, including the optimal control of nonlinear systems with state and control constraints, the DP algorithm requires substantial data storage and a large number of minimizations. The substantial data storage that dynamic programming requires results from the inefficient lookup table representation of J_t^* and $\underline{u}^*$ at each quantized state and time; each item of data is represented exactly by a unique adjustable parameter. This *curse of dimensionality* also existed in the direct learning algorithms. Many practical problems, having fine levels of state and control quantization, require a continuous functional mapping, for which a single adjustable parameter encodes information over some region of the input space. Additionally, a continuous mapping eliminates the necessity to interpolate between discrete grid points in the input space to determine the appropriate control action for an arbitrary input. A learning system could be employed to perform this function approximation. A second disadvantage of the DP algorithm is the necessity of an accurate dynamic model. If the equations of motion are not accurately known a priori, explicit system identification is necessary to apply any dynamic programming procedure. The coordination of the DP algorithm with a learning system that incrementally improves the system model would constitute an indirect learning optimal controller.

A.2 Differential Dynamic Programming

Discrete time differential dynamic programming, introduced by D. Q. Mayne [35] and refined by D. H. Jacobson and Mayne [36], is a numeric approximation to the classical dynamic programming algorithm and is, therefore, also applicable to nonlinear discrete time optimal control problems.[1] Starting with a nominal state trajectory and a nominal control sequence, the DDP algorithm selects neighboring trajectories and sequences that yield the optimal decrease in the second-order approximation to the cost functional $J(\underline{u}) = \sum_{t=1}^{N} g(\underline{x}, \underline{u}, t)$.

The differential dynamic programming class of algorithms incorporates features of both dynamic programming and the calculus of variations. Before presenting an overview of the basic DDP algorithm, several of these properties will be reviewed. DDP does not involve the discretization of state and control spaces, which dynamic programming requires. Additionally, whereas dynamic programming constructs the value function of expected future cost to achieve the terminal conditions for each discrete state and each stage in discrete time, DDP constructs a continuous quadratic approximation to the value function for all states near the nominal trajectory. Finally, DDP solves for a control sequence iteratively, as do many solution techniques for the two-point boundary-value problems which arise from the calculus of variations. Bellman's algorithm (DP), in contrast, generates a control policy in a single computationally intensive procedure.

Each iteration of the differential dynamic programming algorithm consists of two phases: a backward run to determine $\underline{\delta u}(\underline{x}; t)$, the linear policy which defines the change from the nominal control as a function of state, for states near the nominal, and a forward run to update the nominal state trajectory and nominal control sequence [37,38]. The DDP algorithm requires accurate models of the incremental

[1] Jacobson and Mayne also applied the differential dynamic programming method to continuous time systems [36].

cost function $g(\underline{x}, \underline{u}, t)$ and the state transition function $\underline{T}(\underline{x}, \underline{u}, t)$. Furthermore, the original DDP algorithm requires that both of these functions are twice differentiable with respect to states and controls; this condition is relaxed to a necessary single differentiability in several modified DDP algorithms.

The following development of the original DDP algorithm follows primarily from Yakowitz [39]. The nominal control sequence $\underline{\mathbf{u}}_n$ along with the initial state $\underline{x}(1)$ defines a nominal state trajectory $\underline{\mathbf{x}}_n$.

$$\underline{\mathbf{u}}_n = \{\underline{u}_n(1),\, \underline{u}_n(2),\, \ldots\, \underline{u}_n(N)\} \tag{A.3a}$$

$$\underline{\mathbf{x}}_n = \{\underline{x}_n(1),\, \underline{x}_n(2),\, \ldots\, \underline{x}_n(N)\} \tag{A.3b}$$

The backward recursion commences at the final decision time, N, by constructing a quadratic approximation to the nominal cost.

$$L(\underline{x}, \underline{u}, N) = QP\left[g(\underline{x}, \underline{u}, N)\right] \tag{A.4}$$

The $QP[\cdot]$ operator selects the quadratic and linear, but not the constant, terms of the Taylor's series expansion of the argument about the nominal state and control sequences. A first order necessary condition for a control $\underline{u}^*$ to minimize $L(\underline{x}, \underline{u}, N)$ appears in (A.5), which can be solved for the optimal input.

$$\Delta_{\underline{u}} L(\underline{x}, \underline{u}, N) = 0 \tag{A.5}$$

$$\underline{\delta u}(\underline{x}, N) = \underline{u}^*(N) - \underline{u}_n(N) = \underline{\alpha}_N + \beta_N \underline{\delta x}(N) \tag{A.6a}$$

$$\underline{\delta x}(N) = \underline{x}^*(N) - \underline{x}_n(N) \qquad (A.6b)$$

The optimal value function, $f(\underline{x}, N) = \min_{\underline{u}}[g(\underline{x}, \underline{u}, N)]$, is also approximated by a quadratic.

$$V(\underline{x}; N) = L(\underline{x}, \underline{u}(\underline{x}, N), N) \qquad (A.7)$$

The DDP backward calculations proceed for $t = N{-}1,\ N{-}2, \ldots 1$. Assuming that $V(\underline{x}; t{+}1)$ has been determined, the cost attributed to the current stage together with the optimal subsequent cost to achieve the terminal conditions is represented by $L(\underline{x}, \underline{u}, t)$.

$$L(\underline{x}, \underline{u}, t) = QP\,[g(\underline{x}, \underline{u}, t) + V(\underline{T}(\underline{x}, \underline{u}, t); t + 1)] \qquad (A.8)$$

The necessary condition $\Delta_{\underline{u}} L(\underline{x}, \underline{u}, t) = 0$ yields the policy for the incremental control.

$$\underline{\delta u}(\underline{x}; t) = \underline{\alpha_t} + \beta_t(\underline{x}(t) - \underline{x}_n(t)) \qquad (A.9)$$

$$\underline{u}(\underline{x}, t) = \underline{x}_n(t) + \underline{\delta u}(\underline{x}; t) \qquad (A.10)$$

The expression for the variation in control (A.9) is valid for any state $\underline{x}(t)$ sufficiently close to the nominal state $\underline{x}_n(t)$. The vector $\underline{\alpha_t}$ and matrix β_t, $1 \le t \le N$, must be maintained for use in the forward stage of DDP. The optimal value function appears in (A.11).

$$V(\underline{x}; t) = L(\underline{x}, \underline{u}(\underline{x}, t), t) \qquad (A.11)$$

The forward run calculates a successor control sequence and the corresponding state trajectory. Given $\underline{x}(1)$, $\underline{u}^*(1) = \underline{u}_n(1) + \underline{\alpha_1}$ by (A.9) and (A.10). Therefore,

$\underline{x}^*(2) = \underline{T}(\underline{x}(1), \underline{u}^*(1), 1)$. For $t = 2, 3, \ldots N$, (A.12) defines the new control and state sequences which become the nominal values for the next iteration.

$$\underline{u}^*(\underline{x}, t) = \underline{\delta u}(\underline{x}(t), t) + \underline{u}_n(t) \tag{A.12a}$$

$$\underline{x}^*(t+1) = \underline{T}(\underline{x}^*(t), \underline{u}^*(t), t) \tag{A.12b}$$

The reduction of required computations, which differential dynamic programming demonstrates with respect to conventional mathematical programming algorithms, is most noticeable for problems with many state and control variables and many stages in discrete time. Whereas each iteration of the DDP algorithm involves solving a low dimensional problem for each stage in time, mathematical programming schemes for the numerical determination of an optimal trajectory typically require the solution of a single high dimensional problem for each iteration. To quantify this relationship, consider the problem where the state vector is a member of R^n, the control vector lies in R^m, and N represents the number of stages in discrete time. The DDP algorithm inverts N matrices of order m, for each iteration; the computational effort, therefore, grows linearly with N.[2] The method of variation of extremals provides a numeric solution to two-point boundary-value problems [23]. A single iteration of Newton's method for determining the roots of nonlinear equations, a technique for implementing the variation in extremals algorithm, in contrast, requires an $N \cdot m$ matrix to be inverted; the cost of an iteration, therefore, grows in proportion to N^3 [40]. Furthermore, both algorithms are quadratically convergent. In the case where $N = 1$, however, the DDP algorithm and Newton's method define identical incremental improvements in state and

[2] The control sequence will be in $R^{N \cdot m}$

control sequences. [39] Similar computational differences exist between the DDP algorithm and other iterative numerical techniques such as the method of steepest descent and quasilinearization [23].

Appendix B

An Analysis of the AEO Open-loop Dynamics

This analysis follows directly from Parkinson and Smith [9]. Equation (2.12) may be written in the form of an ordinary differential equation with small nonlinear damping.

$$\frac{d^2X}{d\tau^2} + X = \mu f\left(\frac{dX}{d\tau}\right) \quad \text{where} \quad \mu = nA_1 \ll 1 \qquad (B.1)$$

If $\mu = 0$, the solution is a harmonic oscillator with a constant maximum vibration amplitude $\overline{X}$ and phase ϕ.[1]

$$X = \overline{X}cos(\tau + \phi) \qquad (B.2a)$$

$$\frac{dX}{d\tau} = -\overline{X}sin(\tau + \phi) \qquad (B.2b)$$

If μ is non-zero but much less than one ($0 < \mu \ll 1$) the solution may be expressed by a series expansion in powers of μ.

$$X = \overline{X}cos(\tau + \phi) + \mu g_1(\overline{X}, \tau, \phi) + \mu^2 g_2(\overline{X}, \tau, \phi) + \ldots \qquad (B.3)$$

[1] All quantities in this analysis are nondimensional.

In the expansion, $\overline{X}$ and ϕ are slowly varying functions of τ. To first-order, this series may be approximated by (B.2), where $\overline{X}$ and ϕ are now functions of τ. For slowly varying $\overline{X}$ and ϕ, these equations predict nearly circular trajectories in the phase plane. The parameters presented in §2.2 and used for all AEO experiments do not strictly satisfy these assumptions. However, the analysis provides insights to the AEO dynamics.

Following the outline presented in [9], each side of (B.1) is multiplied by $\dot{X}$ and the algebra is manipulated.

$$(\ddot{X} + X)\dot{X} = \mu \dot{X} f(\dot{X}) \tag{B.4}$$

$$(\ddot{X} + X)\dot{X} = \frac{1}{2}\frac{d}{d\tau}(X^2 + \dot{X}^2) \tag{B.5}$$

$$X^2 + \dot{X}^2 = \overline{X}^2 cos^2(\tau + \phi) + \overline{X}^2 sin^2(\tau + \phi) = \overline{X}^2 \tag{B.6}$$

$$\mu \dot{X} f(\dot{X}) = -\mu \overline{X} sin(\tau + \phi) f\left(-\overline{X} sin(\tau + \phi)\right) \tag{B.7}$$

$$\frac{1}{2}\frac{d\overline{X}^2}{d\tau} = -\mu \overline{X} sin(\tau + \phi) f\left(-\overline{X} sin(\tau + \phi)\right) \tag{B.8}$$

That $\overline{X}$ varies slowly with τ implies that the cycle period is small compared with the time intervals during which appreciable changes in the amplitude occur. Therefore, an average of the behavior over a single period eliminates the harmonics and is still sufficient for the purpose of examining the time evolution of the amplitude.

$$\frac{1}{2}\frac{d\overline{X}^2}{d\tau} = -\mu \frac{1}{2\pi}\int_0^{2\pi} \overline{X} sin(\tau + \phi)\, f\left(-\overline{X}(\tau + \phi)\right) d\tau \tag{B.9}$$

Recall from (2.12) and (B.1) that $f(\frac{dX}{d\tau})$ is given by

$$f(\frac{dX}{d\tau}) = \frac{1}{1000}\left[1000\left(U - \frac{2\beta}{nA_1}\right)\frac{dX}{d\tau} - \left(\frac{A_3}{A_1U}\right)(1000\frac{dX}{d\tau})^3 + \left(\frac{A_5}{A_1U^3}\right)(1000\frac{dX}{d\tau})^5 - \left(\frac{A_7}{A_1U^5}\right)(1000\frac{dX}{d\tau})^7\right]. \qquad (B.10)$$

Therefore, (B.9) reduces to

$$\frac{d\overline{X}^2}{d\tau} = \frac{nA_1}{1000}\left[1000\left(U - \frac{2\beta}{nA_1}\right)\overline{X}^2 - \frac{3}{4}\left(\frac{A_3}{A_1U}\right)1000^3\overline{X}^4 + \frac{5}{8}\left(\frac{A_5}{A_1U^3}\right)1000^5\overline{X}^6 - \frac{35}{64}\left(\frac{A_7}{A_1U^5}\right)1000^7\overline{X}^8\right]. \qquad (B.11)$$

In the following analysis, let R represent the square of the amplitude of the reduced vibration, i.e. $R = \overline{X}^2$. Equation (B.11) may immediately be rewritten in terms of R.

$$\frac{dR}{d\tau} = aR - bR^2 + cR^3 - dR^4 \qquad (B.12)$$

Recalling that $\mu \ll 1$, stationary oscillations are nearly circular and correspond to constant values of $\overline{X}^2$; constant values of $\overline{X}^2$ are achieved when

$$\frac{dR}{d\tau} = 0. \qquad (B.13)$$

This condition is satisfied by $R = 0$ and also by the real, positive roots of the cubic $a - bR + cR^2 - dR^3 = 0$. Negative or complex values for the squared amplitude of vibration would not represent physical phenomena.

Stability is determined by the tendency of the oscillator to converge or diverge in response to a small displacement δR. The sign of $\frac{d}{dR}\left(\frac{dR}{d\tau}\right)\Big|_{R=R_i}$ determines the stability of the focus and the limit cycles and will be positive, negative, or zero for unstable, stable, and neutrally stable trajectories, respectively.

$$\frac{d}{dR}\left(\frac{dR}{d\tau}\right) = a - 2bR + 3cR^2 - 4dR^3 \tag{B.14}$$

The stability of the focus is easily analyzed.

$$\frac{d}{dR}\left(\frac{dR}{d\tau}\right)\Bigg|_{R=0} = a = nA_1\left(U - \frac{2\beta}{nA_1}\right) \tag{B.15}$$

$$A_1 = \frac{dC_l}{d\alpha}\Bigg|_{\alpha=0} > 0 \tag{B.16}$$

Given that n, U, A_1, A_3, A_5, and A_7 are positive, the coefficients b, c, and d will also be positive. If $\beta = 0$, the system has no mechanical damping and a will be positive for all values of windspeed. However, if $\beta > 0$, then $a > 0$ only if $U > U_c = \frac{2\beta}{nA_1}$. Therefore, if $\beta = 0$ the focus is unstable for all windspeeds greater than zero, and if $\beta > 0$ the focus is unstable for $U > U_c$. This minimum airspeed for oscillation is the definition of the reduced critical windspeed; oscillation can be eliminated for windspeeds below a specific value by sufficiently increasing the mechanical damping.

Three distinct solutions exist when $a > 0$; the focus is unstable for each. The choice among these possibilities, which are characterized by the real positive roots of the cubic $a - bR + cR^2 - dR^3 = 0$, depends upon the windspeed. (1) If R_1 is the

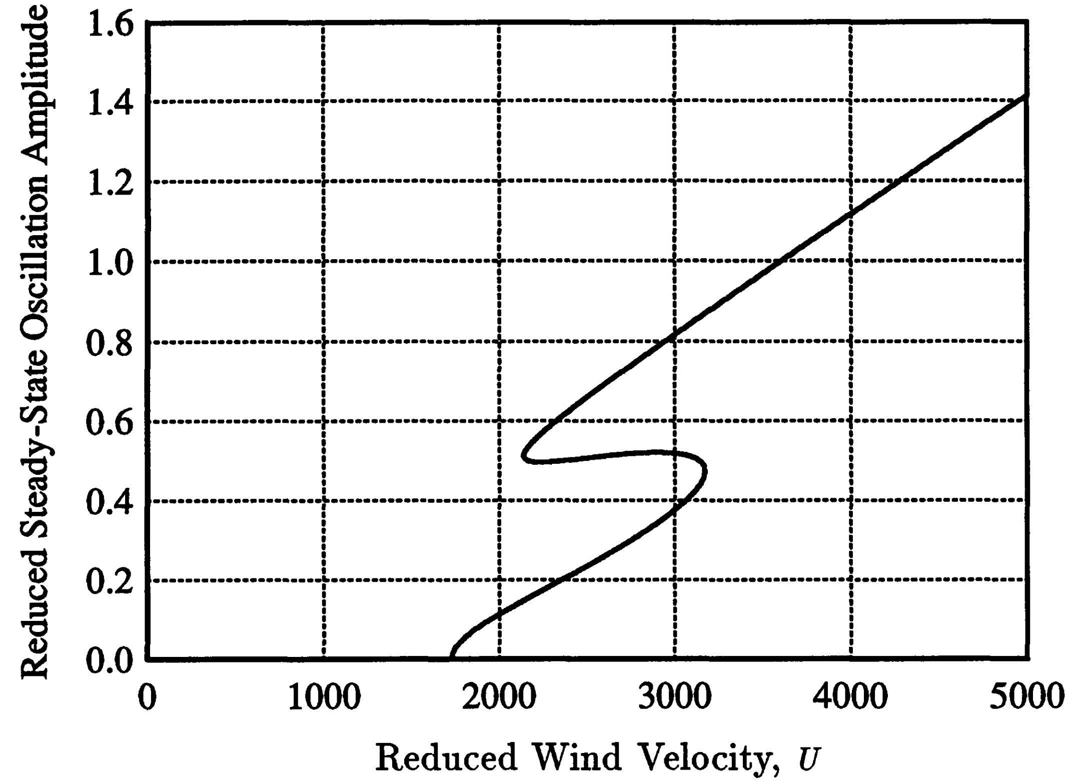

Figure B.1. The steady state amplitudes of oscillation $\overline{X}_{ss}$ versus the incident windspeed U.

single real, positive root, there is a single stable limit cycle of radius $\sqrt{R_1}$ around the unstable focus. This condition exists for two ranges of the reduced windspeed. (2) Three real, distinct, positive roots, $R_3 > R_2 > R_1$, correspond to two stable limit cycles at $\sqrt{R_1}$ and $\sqrt{R_3}$ separated by an unstable limit cycle at $\sqrt{R_2}$. (3) R_1 and R_2 coalesce to a double, real, positive root while R_3 is a distinct, real, positive root. The magnitude of the radius of the single stable limit cycle depends on prior state information; this hysteresis is discussed below. This condition occurs at two values of the reduced incident windspeed.

The most interesting dynamics occur when the second of these situations exists. Figure B.1 plots the steady state amplitude of oscillation $\overline{X}_{ss}$, for circular

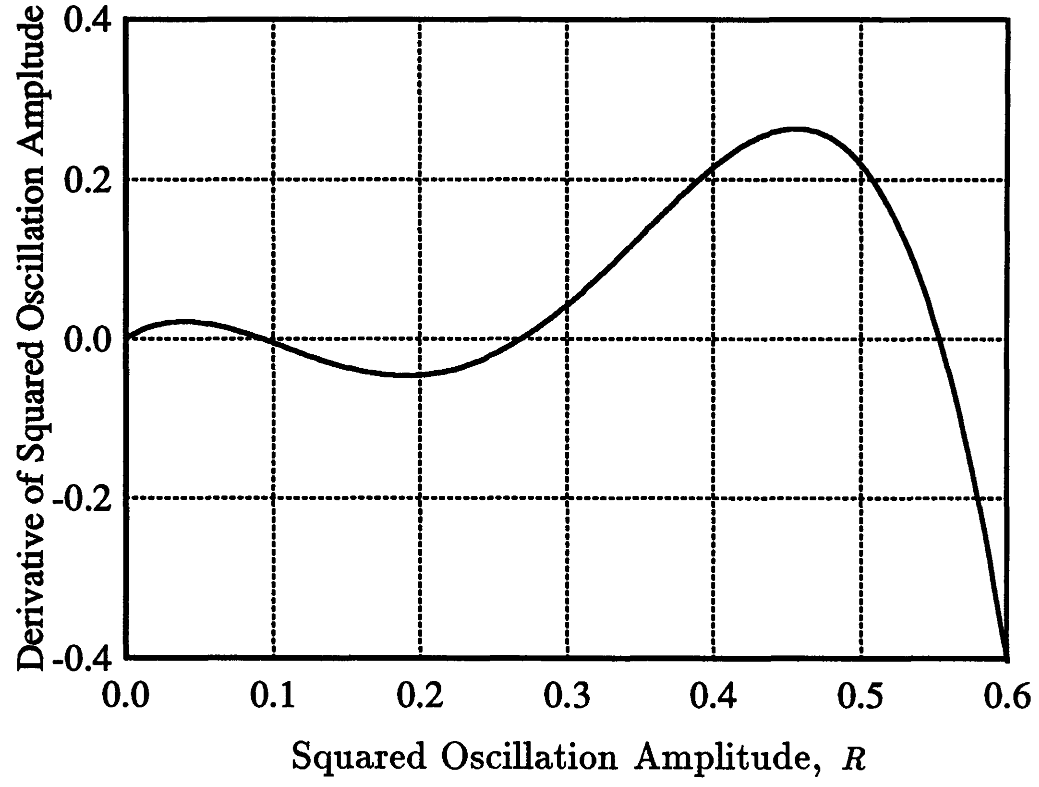

Figure B.2. $\frac{dR}{d\tau}$ versus R for $U = 2766.5$.

limit cycles, as a function of incident windspeed.

A hysteresis in $\overline{X}_{ss}$ can be demonstrated by increasing the reduced airspeed from $U < U_c$, where $\overline{X}_{ss} = 0$. For $U_c < U < U_2$, the amplitude of the steady state oscillation will correspond to the inner stable limit cycle; for $U > U_2$, $\overline{X}_{ss}$ jumps to the larger stable limit cycle. As the dimensionless windspeed is decreased from $U > U_2$, the amplitude of the steady state oscillation will remain on the outer stable limit cycle while $U > U_1$.[2] When the windspeed is decreased below $U = U_1$, the steady state amplitude of oscillation decreases to the inner stable limit cycle. Therefore, for a constant windspeed $U_1 < U < U_2$, $\overline{X}_{ss}$ resides on the inner

[2] $U_c < U_1 < U_2$.

stable limit cycle when the initial displacement is less than the magnitude of the unstable limit cycle, and $\overline{X}_{ss}$ lies on the outer stable limit cycle when the initial displacement is greater than the magnitude of the unstable limit cycle.

For a specific value of the reduced wind velocity, the rate of change of the square of the oscillation amplitude, $\frac{dR}{d\tau}$, can be plotted against the square of the amplitude of oscillation R (Figure B.2). If $\frac{dR}{d\tau}$ is positive, the oscillation amplitude will increase with time, and if $\frac{dR}{d\tau}$ is negative the oscillation amplitude will decrease with time. Therefore, an oscillation amplitude where the value of $\frac{dR}{d\tau}$ crosses from positive to negative with increasing R is a stable amplitude. The focus will be stable when the time rate of change of oscillation amplitude is negative for R slightly greater than zero.

References

[1] Sutton, Richard S., Andrew G. Barto, and Ronald J. Williams, "Reinforcement Learning is Direct Adaptive Optimal Control," *IEEE Control Systems Magazine*, vol. 12, no. 2, pp. 2143–2146, (1992).

[2] Farrell, Jay and Walter Baker, "Learning Augmented Control for Advanced Autonomous Underwater Vehicles," Proceedings of *The 18'th Annual Symposium & Exhibit of the Association for Unmanned Vehicle Systems*, Washington D.C., (1991).

[3] Baker, Walter L. and Jay A. Farrell, "Learning Augmented Flight Control for High Performance Aircraft," Proceedings of *AIAA GN&C Conference*, New Orleans, LA, (1991).

[4] Baker, Walter and Jay Farrell, "An Introduction to Connectionist Learning Control Systems," *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive*, D. White and D. Sofge (eds.), Van Nostrand Reinhold, New York, pp. 35–64, (1992).

[5] Barto, Andrew G., Richard S. Sutton, and Charles W. Anderson, "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 5, (1983).

[6] Barto, Andrew G. and P. Anandan, "Pattern-Recognizing Stochastic Learning Automata," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 3, pp. 360–375, (1985).

[7] Millington, Peter J. and Walter L. Baker, "Associative Reinforcement Learning for Optimal Control," Proceedings of *The AIAA Guidance, Navigation, and Control Conference*, Portland, OR, (1990).

[8] Michie, D. and R. A. Chambers, "BOXES: An Experiment in Adaptive Control," *Machine Intelligence 2*, Ella Dale and Donald Michie (eds.), American Elsevier Publishing Company, New York, pp. 137–152, (1968).

[9] Parkinson, G. V. and J. D. Smith, "The Square Prism as an Aeroelastic Nonlinear Oscillator," *Quarterly Journal of Mechanics and Applied Mathematics*, vol. 17, pp. 225–239, (1964).

[10] Thompson, J. M. T. and H. B. Stewart, *Nonlinear Dynamics and Chaos*, pp. 60–64, John Wiley and Sons, New York, (1986).

[11] Alexander, Jeff, L. Baird, W. Baker, and J. Farrell, "A Design & Simulation Tool for Connectionist Learning Control Systems: Application to Autonomous Underwater Vehicles," Proceedings of *The Conference of the Society for Computer Simulation*, Baltimore, MD, (1991).

[12] Klopf, A. Harry, "A Neuronal Model of Classical Conditioning," *Psychobiology*, vol. 16, no. 2, pp. 85–125, (1988).

[13] Baird III, Leemon C. and A. Harry Klopf, "Extensions of the Associative Control Process (ACP) Network: Hierarchies and Provable Optimality," Proceedings of the conference *Simulation of Adaptive Behavior*, (1992).

[14] Klopf, A. Harry, James S. Morgan, and Scott E. Weaver, "A Hierarchical Network of Control Systems that Learn: Modeling Nervous System Function During Classical and Instrumental Conditioning," Submitted to *Adaptive Behavior*, (1992).

[15] Sutton, Richard S., "Learning to Predict by Methods of Temporal Differences," *Machine Intelligence 3*, Kluwer Academic Publishers, Boston, pp. 9–44, (1988).

[16] Watkins, C., "Learning from Delayed Rewards," Doctoral thesis, Cambridge University, Cambridge, England, (1989).

[17] Watkins, Christopher J. C. H. and Peter Dayan, "Q-Learning," *Machine Learning*, (1992).

[18] Samuel, A. L., "Some Studies in Machine Learning Using the Game of Checkers," *Computers and Thought*, E. A. Freigenbaum and J. Feldman (eds.), McGraw Hill, New York, (1959).

[19] Holland, J. H., "Escaping Brittleness: The Possibility of General-purpose Learning Algorithms Applied to Parallel Rule-based Systems," *Machine Learning: An Artificial Intelligence Approach*, vol. 2, R. S. Michalski, J. G. Carbonell, and J. M. Mitchell (eds.) Morgan Kaufmann, Los Altros, CA, (1986).

[20] Sutton, R. S., "Temporal Credit Assignment in Reinforcement Learning," Doctoral thesis, Department of Computer and Information Science, University of Massachusetts, Amherst, MA, (1984).

[21] Widrow, B. and M. E. Hoff, "Adaptive Switching Circuits," *1960 WESCON Convention Record*, Part 4, pp. 96–104, (1960).

[22] Bryson, Jr., Arthur E. and Yu-Chi Ho, *Applied Optimal Control*, Hemisphere Publishing Corp., New York, (1975).

[23] Kirk, Donald E., *Optimal Control Theory*, Prentice-Hall Inc., Englewood Cliffs, NJ, (1970).

[24] Anderson, Mark R. and David K. Schmidt, "Error Dynamics and Perfect Model Following with Application to Flight Control," *AIAA Journal of Guidance, Control, and Dynamics*, vol. 14, no. 5, pp. 912–919, (1991).

[25] Baker, Walter L. and Peter J. Millington, "Design and Evaluation of a Learning Augmented Longitudinal Fight Control System," Proceedings of *The 32'nd IEEE Conference on Decision and Control*, San Antonio, TX, (1993).

[26] Press, W., B. Flannery, S. Teukolsky, and W. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, (1988).

[27] Youcef-Toumi, K. and Ito Osamu, "A Time Delay Controller for Systems with Unknown Dynamics," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 112, (1990).

[28] Nistler, Noel F., "A Learning Enhanced Flight Control System for High Performance Aircraft," Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, (1992).

[29] Franklin, G. F., J. D. Powell, and M. L. Workman, *Digital Control of Dynamic Systems*, Addison-Wesley, Reading, MA, (1980).

[30] Funahashi, K. "On the Approximate Realization of Continuous Mappings by Neural Networks," *Neural Networks*, vol. 2, pp. 183–192, (1988).

[31] Minsk, L. and S. Papert, *Perceptrons*, MIT Press, Cambridge, MA, (1969).

[32] Poggio, T. and F. Girosi, "Networks for Approximation and Learning," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1481–1497, (1990).

[33] Bellman, R. E., *Dynamic Programming*, Princeton University Press, Princeton, NJ, (1957).

[34] Bellman, R. E., and S. E. Dreyfus, *Applied Dynamic Programming*, Princeton University Press, Princeton, NJ, (1962).

[35] Mayne, D., "A Second-Order Gradient Method for Determining Optimal Trajectories of Nonlinear Discrete-Time Systems," *International Journal on Control*, vol. 3, pp. 85–95, (1966).

[36] Jacobson, David H. and David Q. Mayne, *Differential Dynamic Programming*, American Elsevier Publishing Company, Inc., New York, (1970).

[37] Lopez-Coronado, J. and L. Le Letty, "Differential Dynamic Programming — Implementation of Algorithms and Applications," *Simulation in Engineering Sciences*, J. Burger and Y. Jarny (eds.), Elsevier Science Publishers, B. V. North-Holland, pp. 93–102, (1983).

[38] Sen, S. and S. J. Yakowitz, "A Quasi-Newton Differential Dynamic Programming Algorithm for Discrete-time Optimal Control," *Automatica*, vol. 23, no. 6, pp. 749–752, (1987).

[39] Yakowitz, Sydney, "Algorithms and Computational Techniques in Differential Dynamic Programming," *Control and Dynamic Systems*, vol. 31, (1989).

[40] Pantoja, J. F. A. DE O., "Differential Dynamic Programming and Newton's Method," *International Journal of Control*, vol. 47, no. 5, pp. 1539–1553, (1988).

[41] Millington, Peter J., "Associative Reinforcement Learning for Optimal Control," Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, (1991).

[42] Baird, Leemon C., "Function Minimization for Dynamic Programming Using Connectionist Networks," Proceedings of *The IEEE Conference on Systems, Man, and Cybernetics*, Chicago, IL, (1992).