NONLINEAR H_2/H_∞ CONSTRAINED FEEDBACK CONTROL: A PRACTICAL DESIGN APPROACH USING NEURAL NETWORKS

by

MURAD MUHAMMAD SAMIR MUHAMMAD ALI ABU-KHALAF

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2005

Copyright © by Murad Muhammad Samir Muhammad Ali Abu-Khalaf 2005

All Rights Reserved

In the name of Allah,
Most Gracious, Most Merciful

This dissertation is dedicated to my parents
Suzan & Samir

ACKNOWLEDGEMENTS

I would like to express my gratitude to Professor Frank L. Lewis whose close supervision, deep knowledge, patience, and strong financial support were the main drive behind the success of my doctoral studies.

My thanks also go to the doctoral supervising committee Professors: Michael T. Manry, Dan O. Popa, Panayiotis S. Shiakolas, Harry E. Stephanou, and Kai-Shing Yeung. I thank them for their time and the efforts they put to improve this work.

I am indebted to my mother Suzan and father Samir for their prayers, worries, and support during my long years abroad. I thank my sister Ayah and brother Samer. Not to mention my brother Amir and cousin Ashraf with whom I never felt away from home.

During my stay at Arlington, I enjoyed the warm friendship of many people. In particular; Asma Altamimi, Abdulrahman Alkelani, Ahmad Almardini, Mahmoud Almasri, Iyad Alfaluji, Hussam Alshammari, Kamal Atwat, Amer Hamdan, Mahmoud Smadi, Nabil Mandahawi and Muhammad Mayyas.

I would like to thank my colleagues at the Automation & Robotics Research Institute in particular those I worked with at the Advanced Controls and Sensors group.

This work was funded by the National Science Foundation ECS-0140490 grant, and by the Army Research Office DAAD 19-02-1-0366 grant.

July 15th, 2005

ABSTRACT

NONLINEAR H_2/H_∞ CONSTRAINED FEEDBACK CONTROL: A PRACTICAL DESIGN APPROACH USING NEURAL NETWORKS

Publication No. ______

Murad Muhammad Samir Muhammad Ali Abu-Khalaf, PhD.

The University of Texas at Arlington, 2005

Supervising Professor: Frank L. Lewis

In this research, practical methods for the design of H_2 and H_∞ optimal state feedback controllers for constrained input systems are proposed. The dynamic programming principle is used along with special quasi-norms to derive the structure of both the saturated H_2 and H_∞ optimal controllers in feedback strategy form. The resulting Hamilton-Jacobi-Bellman (HJB) and Hamilton-Jacobi-Isaacs (HJI) equations are derived respectively. It is shown that introducing quasi-norms on the constrained input in the performance functional allows unconstrained minimization of the Hamiltonian of the corresponding optimal control problem.

Moreover, it is shown how to obtain nearly optimal minimum-time and

constrained state controllers by modifying the performance functional of the optimization problem.

Policy iterations on the constrained input for both the H_2 and H_∞ cases are studied. It is shown that the resulting sequence of Lyapunov functions in the H_2 case, cost functions in the H_∞ case, converge uniformly to the value function of the associated optimal control problem that solves the corresponding Hamilton-Jacobi equation. The relation between policy iterations for the zero-sum game appearing in the H_∞ optimal control and the theory of dissipative systems is studied. It is shown that policy iterations on the disturbance player solve the nonlinear bounded real lemma problem of the associated closed loop system. Moreover, the relation between the domain of validity of the game value function and the corresponding L_2-gain is addressed through policy iterations.

Neural networks are used along with the least-squares method to solve for the linear in the unknown differential equations resulting from policy iterations on the saturated control in the H_2 case, and the saturated control and the disturbance in the H_∞ case. The result is a neural network constrained feedback controller that has been tuned a priori offline with the training set selected using Monte Carlo methods from a prescribed region of the state space which falls within the region of asymptotic stability of an initial stabilizing control used to start the policy iterations.

Finally, the obtained algorithms are applied to different examples including the Nonlinear Benchmark Problem to reveal the power of the proposed method.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

NOMENCLATURE

x	state vector of the dynamical system
$\|x\|$	the 2-norm of vector x
x'	transpose of the vector x
$V(x)$	value or cost of x
V_x	Jacobian of V with respect to x
H_2	2-norm on the Hardy space
H_∞	∞-norm on the Hardy space
$\mathbb{R}^n$	n-dimensional Euclidean space
Ω	compact set of the state space
$C^m(\Omega)$	continuous and differentiable up to the m^{th} degree on Ω
w	neural network weight
$\mathbf{w}$	neural network weight vector
σ	neural network activation function
$\boldsymbol{\sigma}$	neural network activation functions vector
$\nabla\boldsymbol{\sigma}$	gradient of $\boldsymbol{\sigma}$ with respect to x
HJB	Hamilton-Jacobi-Bellman
HJI........................	Hamilton-Jacobi-Isaacs
DOV.....................	Domain of Validity

$\exists$ .... there exists

$\sup\limits_{x \in \Omega}$ .... supremum of a function with respect to x on Ω

$\min\limits_{u}$ .... minimum with respect to u

$\max\limits_{d}$ .... maximum with respect to d

$\langle a(x), b(x) \rangle$ .... integral $\int a(x)b(x)dx$ for scalar $a(x)$ and $b(x)$

CHAPTER 1

INTRODUCTION

1.1 Significance and Contribution of the Research

The design of control systems requires one to consider various types of constraints and performance measures. Constraints encountered in control systems design are due to physical limitations imposed on the controller and the plant. This includes actuator saturation and constraints on the states. Performance measures on the other hand are related to optimality issues. This includes objectives like, minimum fuel, minimum energy, minimum-time, and robustness. Combining constraints with performance measures requires, in general, solving complicated optimal control problems. Only in limited cases one may obtain a closed-form solution, *i.e.* feedback solution, of the controller. In most cases, solutions are obtained using numerical open loop methods [43]. For example, there are many ways to find the open loop controller for a linear quadratic regulator (LQR) with input constraints. However, it is unclear how to directly obtain the closed form solution.

In this research, a practical design method to design H_2 and H_∞ optimal state feedback controllers for constrained input systems is proposed. The value function of the associated optimization problem is solved for in a least-squares sense resulting in nearly optimal neural network state feedback controllers that are valid over a prescribed region of the state space. These feedback controllers are more appropriate for engineering applications. Hence, this work tries to bridge the gap between theoretical optimal control and practical implementations of optimal controllers for systems mainly experiencing actuator saturation. A unified framework for constructing neural network controllers that are nearly H_2 and H_∞ optimal for constrained input systems is provided.

The control of systems with saturating actuators has been the focus of many researchers for many years. Several methods for deriving control laws considering the saturation phenomena are found in Saberi, et al. [69], Sussmann, et al., [74]. Other methods that deal with constraints on the states of the system as well as the control inputs are found in Bitsoris, et al., [21]; Hu, et al.,[36]; Henrion, et al., 2001; Gilbert and Tan, 1991. Most of these methods are based on mathematical programming and the set invariance theory resulting in controllers that satisfy the required constraints. However, the controllers developed are not necessarily in closed-loop form. Moreover, optimality issues are not the main concern in this theme of work. Most of these methods do not consider finding optimal control laws for general nonlinear systems.

The optimal control of constrained input systems is theoretically well established. The controller can be found by applying the Pontryagin's minimum principle. This usually requires solving a split boundary differential equation and the result is an open loop optimal control [50].

There has been several studies to derived and solve for closed loop optimal control laws for constrained input systems. Bernstein [18] studied the performance optimization of saturated actuators control. Lyshevski [58], [57], presented a general framework for the design of optimal state feedback control laws based on dynamic programming. He proposes the use of nonquadratic performance functionals to encode various kinds of constraints on the control system. These performance functionals are used along with the famous Hamilton-Jacobi-Bellman (HJB) equation that appears in optimal control theory [50]. The resulting control law structure is in state feedback form. This is since the HJB gives a control that is a function of the value function of the optimization problem which is in turn a function of the states of the system. However, it remains unclear how to solve for the value function of the HJB equation formulated using nonquadratic performance functionals.

Optimal L_2-gain disturbance attenuation controllers are also treated in this work. This comes under the framework of H_∞ optimal control. The H_∞ norm has played an important role in the study and analysis of robust optimal control theory since its original formulation in an input-output setting by Zames, [81]. Earlier solution techniques involved operator-theoretic methods. State space solutions were rigorously derived in [26] for the linear system case that required solving several associated Riccati equations. Later, more insight into the problem was given after the H_∞ linear control problem was posed as a zero-sum two-person differential game by Başar [13]. The nonlinear counterpart of the H_∞ control theory was developed by Van der Schaft [76]. He utilized the notion of dissipativity, introduced by Willems [80], [79], Hill and Moylan for nonlinear systems [34], to formulate the H_∞ control theory into a nonlinear L_2-gain optimal control problem. He made use of the fact that the H_∞ norm in the frequency domain is nothing but the L_2-induced norm from the input time-function to the output-time function for initial zero state. The L_2-gain optimal control problem requires solving a Hamilton-Jacobi equation, namely the Hamilton-Jacobi-Isaacs (HJI) equation. Conditions for the existence of smooth solutions of the Hamilton-Jacobi equation were studied through invariant manifolds of Hamiltonian vector fields and the relation with the Hamiltonian matrices of the corresponding Riccati equation for the linearized problem, [76]. Later some of these conditions were relaxed by Isidori and Astolfi [39], into critical and noncritical cases. Viscosity solutions of the HJI equation were considered in [9], [11].

Although the formulation of the nonlinear theory of H_∞ control has been well developed, solving the HJI equation remains a challenge. Several methods have been proposed to solve the HJI equation. In the work by Huang [38], the smooth solution is found by solving for the Taylor series expansion coefficients in a very efficient and organized manner. Another interesting method is by Beard and coworkers [17]. Beard proposed to iterate in policy space to solve the HJI successively by breaking the, nonlinear in value function, differential equation to a sequence of, linear in the cost function, differential equations. He then proposed a numerically efficient algorithm that solves the sequence of linear differential equations using Galerkin techniques which requires computing numerous integrals over a well valid region of the state space.

Therefore, in this research, special nonquadratic performance functionals are used to encode the various constraints on the optimal control problem. Using the dynamic programming principle, the structure of the feedback strategy for the optimal control law is derived. Then, offline least-squares neural network policy iterations are applied to obtain a closed-form solution of the feedback strategy for both the optimal control, H_2, and zero-sum game, H_∞, problems.

1.2 Approach

In this dissertation, a special quasi-norm to encode the input constraints is used. This allows the definition of new nonquadratic performance functionals. With this quasi-norm, minimizing the Hamiltonian of the optimal control problem with respect to the constrained control input, the minimax controller in the game case, becomes an unconstrained problem. Following that, the resulting Hamilton-Jacobi equations are iteratively solved over a compact set of the asymptotic stability region of an initial stabilizing control using a neural network least squares approach.

Neural networks have been used to control nonlinear systems. In [60], Werbos first proposed using neural networks to find optimal control laws using the HJB equation in what later came to be known as the adaptive critic approach. Parisini used neural networks in [66] to derive optimal control laws for discrete-time stochastic nonlinear system. Successful neural network controllers have been reported in [24], [49], [67], [68], [70], [71]. It has been shown that neural networks can effectively extend adaptive control techniques to nonlinearly parameterized systems. The status of neural network control as of 2001 appears in [64].

1.2.1 H_2 Optimal Control: Hamilton-Jacobi-Bellman equation

The approach here is based on policy iterations for the control input along with neural networks. In this case, the value function of the associated HJB equation is solved for by solving for a sequence of cost functions satisfying a sequence of

Lyapunov equations (LE) resulting from the policy iterations. A neural network is used to approximate the cost function associated with each LE using the method of least squares on a well-defined region of attraction of an initial stabilizing controller. As the order of the neural network is increased, the least-square solution of the HJB equation converges uniformly to the exact solution of the inherently nonlinear HJB equation associated with the saturating control input. The result is a nearly optimal constrained state feedback controller that has been tuned a priori off-line.

1.2.2 H_∞ Optimal Control: Hamilton-Jacobi-Isaacs equation

The approach here is based on policy iterations on the constrained input and the disturbance. Here using a quasi norm to encode the input constraints enables applying quasi L_2-gain analysis of the corresponding closed-loop nonlinear system. The policy iterations on the disturbance solves for the available storage of the dissipative system with respect to a special nonquadratic supply rate. In other words, it solves the corresponding nonlinear bounded real lemma. When followed by policy iterations on the controller, an H_∞ optimal control is obtained for the constrained input systems and the resulting available storage solves for the value function of the associated Hamilton-Jacobi-Isaacs (HJI) equation of the associated zero-sum game. The saddle point strategy corresponding to the related zero-sum differential game is derived, and shown to be the unique feedback saddle point. This iterative game theoretic approach allows a deeper insight on the relation between the attenuation gain and the domain of validity of the H_∞ controller for constrained input systems.

CHAPTER 2

POLICY ITERATIONS AND THE HAMILTON-JACOBI-BELLMAN EQUATION FOR H_2 STATE FEEDBACK CONTROL WITH INPUT SATURATION

2.1 Introduction

In this chapter, the constrained optimal control problem through the framework of the HJB equation is studied. It is shown how to break the HJB equation originally formulated to constrained input systems in [58] into a sequence of Lyapunov equations that are easier to handle. The solution of the HJB equation is a challenging problem due to its inherently nonlinear nature. For linear systems with no constraints, the HJB equation results in the well-known Riccati equation used to derive a linear state feedback control. However, even when the system is linear, the saturated control requirement makes the value function and hence the required control law nonlinear.

In the general nonlinear case, the HJB equation generally cannot be solved for explicitly. There has been a great deal of effort to confront this issue. Approximate HJB solutions have been found using many techniques such as those developed by Saridis [72], Beard [15], [16], [14], Lendaris [63], Lee [48], Bertsekas and Tsitsiklis [19], Munos [62], Lewis and Kim [42], Balakrishnan [32], [53], [52], Lyshevski [56], [58], [57], [54], [55], Huang [38].

In this presentation, the focus is on solving the HJB solution using the so-called generalized HJB equation (GHJB) [14], [72], which is referred to in this dissertation as a Lyapunov Equation (LE) since it is the nonlinear counterpart of the matrix Lyapunov equation [50]. In [72], Saridis et al. developed a policy iteration method that improves a given initial stabilizing control. This method reduces to the well-known Kleinman iterative method for solving the Riccati equation for linear systems [44]. However, for nonlinear systems, it is unclear how to solve the LE equation. Therefore, successful application of the LE was limited until the novel work of Beard [15], [16], [14]. He uses a Galerkin spectral approximation method to find approximate solutions to the LE at each iteration on a given compact set. The framework in which the algorithm is presented in Beard's work requires the computation of a large number of integrals and it is also not able to handle explicit constraints on the controls, which is the main interest of this dissertation.

In this chapter, the policy iterations method is applied to performance functionals that are nonquadratic. And in the next chapter, neural networks are used to solve for the value function of the HJB equation, and to construct a nearly optimal constrained state feedback controller.

In summary, the objective of this chapter is study the application of the policy iteration method to the HJB equation formulated using nonquadratic performance functionals to confront the saturation issue. For constrained input systems, two optimal control problems are presented. The first is a regular optimal saturated regulator, while the second is a minimum time optimal control problem. Therefore, in section 2.2, the HJB equation for constrained input systems is introduced using nonquadratic performance functions. In section 2.3, the LE is introduced that will be useful in implementing the policy iteration method and study the convergence properties of this method. It will be shown that instead of solving for the value function using the HJB directly. One can solve for a sequence of cost functions through the LE equation that converge uniformly to the value function that solves the HJB equation. In section 2.4, it is shown how to construct nonquadratic performance functional to address minimum-time and constrained state problems.

2.2 Optimal Regulation of Systems with Actuator Saturation

Consider an affine in the control nonlinear dynamical system of the form

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

where $x \in \mathbb{R}^n$, $f(x) \in \mathbb{R}^n$, $g(x) \in \mathbb{R}^{n \times m}$. And the input $u \in U, U = \{u = (u_1, \ldots, u_m) \in \mathbb{R}^m : \alpha_i \le u_i \le \beta_i, i = 1, \ldots, m\}$, where α_i, β_i are constants. Assume that $f + gu$ is Lipschitz continuous on a set $\Omega \subseteq \mathbb{R}^n$ containing the origin, and that the system (2.1) is stabilizable in the sense that there exists a continuous control on Ω that asymptotically stabilizes the system. It is desired to find u, which minimizes a generalized nonquadratic functional

$$V(x_0) = \int_0^\infty [Q(x) + W(u)]\, dt \tag{2.2}$$

where $Q(x)$ and $W(u)$ are positive definite functions on Ω, $\forall x \ne 0$ $Q(x) > 0$ and $x = 0 \Rightarrow Q(x) = 0$. For unbounded control inputs, a common choice for $W(u)$ is $W(u) = u'Ru$, where $R \in \mathbb{R}^{m \times m}$. Note that the control u must not only stabilize the system on Ω, but also make the integral finite. Such controls are defined to be admissible [16].

Definition 2.1 *(Admissible Controls) A control* u *is defined to be admissible with respect to (2.2) on* Ω*, denoted by* $u \in \Psi(\Omega)$*, if* u *is continuous on* Ω*;* $u(0)=0$*;* u *stabilizes (2.1) on* Ω*;* $\forall x_0 \in \Omega$*,* $V(x_0)$ *is finite.*

Equation (2.2) can be expanded as follows

$$\begin{aligned} V(x_0) &= \int_0^T [Q(x)+W(u)]\,dt + \int_T^\infty [Q(x)+W(u)]\,dt \\ &= \int_0^T [Q(x)+W(u)]\,dt + V(x(T)). \end{aligned} \tag{2.3}$$

If the cost function V is differentiable at x_0, then rewriting equation (2.3)

$$\begin{aligned} &\lim_{T\to 0} \frac{V(x_0)-V(x(T))}{T} = \lim_{T\to 0} \frac{1}{T}\int_0^T [Q(x)+W(u)]\,dt, \\ &\dot{V} = V_x'\,(f+gu) = -Q(x)-W(u). \end{aligned} \tag{2.4}$$

Equation (2.4) is the infinitesimal version of equation (2.2) and is a non linear Lyapunov equation,

$$LE(V,u) \triangleq V_x'\,(f+gu)+Q+W(u)=0, V(0)=0. \tag{2.5}$$

The LE equation becomes the well-known HJB equation, [50], on substitution of the optimal control

$$u^*(x) = -\frac{1}{2}R^{-1}g^T V_x^{*\prime} \tag{2.6}$$

where $V^*(x)$ is the value function of the optimal control problem which solves the HJB equation

$$HJB(V^*) \triangleq V_x^{*\prime} f + Q - \frac{1}{4} V_x^{*\prime} g R^{-1} g' V_x^{*\prime} = 0, \quad V^*(0) = 0. \tag{2.7}$$

It is shown in [58] that the value function obtained from (2.7) serves as a Lyapunov function on Ω.

To confront bounded controls, Lyshevski [58], [57] introduced a generalized nonquadratic functional

$$W(u) = 2\int_0^u \phi^{-1}(v) R dv, \quad v \in \mathbb{R}^m, \phi \in \mathbb{R}^m, \phi(v) = \begin{bmatrix} \phi(v_1) \\ \vdots \\ \phi(v_m) \end{bmatrix}, \phi^{-1}(u) = \begin{bmatrix} \phi^{-1}(u_1) \\ \vdots \\ \phi^{-1}(u_m) \end{bmatrix} \tag{2.8}$$

where $\phi(\cdot)$ satisfies is a bounded one to one function that belongs to $C^p \ (p \geq 1)$, and $L_2(\Omega)$. Moreover It is a monotonic odd function with its first derivative bounded by the constant M. An example of such a function is the hyperbolic tangent $\phi(\cdot) = \tanh(\cdot)$. R is positive definite and assumed to be symmetric for simplicity of analysis. Note that $W(u)$ is positive definite because $\phi^{-1}(u)$ is monotonic odd and R is positive definite.

The LE equation when (2.8) is used becomes

$$V_x'(f + gu) + Q + 2\int_0^u \phi^{-1}(v) R dv = 0, V(0) = 0. \tag{2.9}$$

Note that the LE equation becomes the HJB equation upon substituting the constrained optimal feedback control

$$u^*(x) = -\phi(\tfrac{1}{2}R^{-1}g'V_x^{*\prime}), \tag{2.10}$$

where $V^*(x)$ solves the following HJB equation

$$\begin{aligned} &V_x^{*\prime}\left(f - g\phi(\tfrac{1}{2}R^{-1}g'V_x^{*\prime})\right) + Q + 2\int_0^{-\phi(\frac{1}{2}R^{-1}g'V_x^{*\prime})} \phi^{-T}(v)R\,dv = 0, \\ &V^*(0) = 0. \end{aligned} \tag{2.11}$$

This is a nonlinear differential equation for which there may be many solutions. Existence and uniqueness of the value function has been shown in [55]. This HJB equation cannot generally be solved. There is no current method for rigorously confronting this type of equation to find the value function for the system. Moreover, current solutions are not well defined over a specific region in the state space.

Remark 2.1. *Optimal control problems do not necessarily have smooth or even continuous value functions, [37][11]. In [51], using the theory of viscosity solutions, it is shown that for infinite horizon optimal control problems with unbounded cost functionals and under certain continuity assumptions of the dynamics, the value function is continuous,* $V^*(x) \in C(\Omega)$. *Moreover, if the Hamiltonian is strictly convex and if the continuous viscosity is semiconcave, then* $V^*(x) \in C^1(\Omega)$, *[11] satisfying the HJB equation everywhere. Note that for affine in input systems, (2.1), the Hamiltonian is strictly convex if the system dynamics are not bilinear, and if the integrand of the performance functional (2.2) does not have cross terms of the states and the input. In* *this chapter, all derivations are performed under the assumption of smooth solutions to (2.9) and (2.11) with all what this requires of necessary conditions. See [76][72] for similar framework of solutions. If this smoothness assumption is released, then one needs to use the theory of viscosity solutions, [11], to show that the continuous cost solutions of (2.9) do converge to the continuous value function of (2.11).*

2.3 Policy Iterations for Constrained Input Systems

It is important to note that the LE is linear in the cost function derivative, while the HJB is nonlinear in the value function derivative. Solving the LE for the cost function requires solving a linear partial differential equation, while the HJB equation solution involves a nonlinear partial differential equation, which may be impossible to solve. This is the reason for introducing the policy iteration technique for the solution of the HJB equation, which is based on a sound proof in [72].

Policy iterations using the LE has not yet been rigorously applied for bounded controls. In this section, it is shown that the policy iterations technique can be used for constrained controls when certain restrictions on the control input are met.

The policy iteration technique is now applied to the new set of equations (2.9), (2.10). The following lemma shows how equation (2.10) can be used to improve the control law. It will be required that the bounding function $\phi(\cdot)$ is nondecreasing.

Lemma 2.1. *If $u_j \in \Psi(\Omega)$, and $V_j \in C^1(\Omega)$ satisfies the equation $LE(V_j, u_j) = 0$ with the boundary condition $V_j(0) = 0$, then the new control derived as*

$$u_{j+1}(x) = -\phi(\tfrac{1}{2} R^{-1} g' V_{x\,j}{}') \tag{2.12}$$

is an admissible control for the system on Ω. Moreover, if the bounding function $\phi(\cdot)$ is monotone odd function, and V_{j+1} is the unique positive definite function satisfying equation $LE(V_{j+1},u_{j+1})=0$, with the boundary condition $V_{j+1}(0)=0$, then $V^*(x)\le V_{j+1}(x)\le V_j(x)\ \ \forall x\in\Omega$.

Proof. To show the admissibility part, since $V_j\in C^1(\Omega)$, the continuity assumption on g implies that u_{j+1} is continuous. Since V_{j+1} is positive definite it attains a minimum at the origin, and thus, $V_{x\,j}=dV_j\big/dx$ must vanish there. This implies that $u_{j+1}(0)=0$. Taking the derivative of V_j along the system $f+gu_{j+1}$ trajectory one has,

$$\dot{V}_j(x,u_{j+1})=V_{x\,j}{}'f+V_{x\,j}{}'gu_{j+1}, \tag{2.13}$$

$$V_{x\,j}{}'f=-V_{x\,j}{}'gu_j-Q-2\int_0^{u_j}\phi^{-1}(v)Rdv\,. \tag{2.14}$$

Therefore equation (2.13) becomes

$$\begin{aligned}&\dot{V}_j(x,u_{j+1})=\\&-V_{x\,j}{}'gu_j+V_{x\,j}{}'gu_{j+1}-Q-2\int_0^{u_j}\phi^{-1}(v)Rdv.\end{aligned} \tag{2.15}$$

Since $V_{x\,j}{}'g(x)=-2\phi^{-1\prime}(u)R$, one has

$$\begin{aligned}&\dot{V}_j(x,u_{j+1})=\\&-Q+2\left[\phi^{-1}(u_{j+1})R(u_j-u_{j+1})-\int_0^{u_j}\phi^{-1}(v)Rdv\right].\end{aligned} \tag{2.16}$$

The second term in the previous equation is negative when ϕ^{-1}, and hence ϕ, is nondecreasing. To see this, note that the design matrix R is symmetric positive definite, this means that one can rewrite it as $R = \Lambda\Sigma\Lambda$ where Σ is a triangular matrix with its values being the singular values of R and Λ is an orthogonal symmetric matrix. Substituting for R in (2.16), one has

$$\begin{aligned}
&\dot{V}_j(x,u_{j+1}) = -Q + \\
&2\left[\phi^{-1}(u_{j+1})\Lambda\Sigma\Lambda(u_j - u_{j+1}) - \int_0^{u_j} \phi^{-1}(v)\Lambda\Sigma\Lambda dv\right].
\end{aligned} \tag{2.17}$$

Applying the coordinate change $u = \Lambda^{-1}z$ to (2.17)

$$\begin{aligned}
&\dot{V}_j(x,u_{j+1}) = -Q + 2\phi^{-1}(\Lambda^{-1}z_{j+1})\Lambda\Sigma\Lambda(\Lambda^{-1}z_j - \Lambda^{-1}z_{j+1}) - \\
&2\int_0^{z_j} \phi^{-1}(\Lambda^{-1}\zeta)\Lambda\Sigma\Lambda\Lambda^{-1}d\zeta \\
&= -Q + 2\phi^{-1}(\Lambda^{-1}z_{j+1})\Lambda\Sigma(z_j - z_{j+1}) \\
&-2\int_0^{z_j} \phi^{-1}(\Lambda^{-1}\zeta)\Lambda\Sigma d\zeta \\
&= -Q + 2\boldsymbol{\pi}'(z_{j+1})\Sigma(z_j - z_{j+1}) - 2\int_0^{z_j} \boldsymbol{\pi}'(\zeta)\Sigma d\zeta.
\end{aligned} \tag{2.18}$$

where $\boldsymbol{\pi}'(z_j) = \phi^{-1\prime}(\Lambda^{-1}z_j)\Lambda$.

Since Σ is a triangular matrix, one can now decouple the transformed input vector such that

$$\dot{V}_j(x,u_{j+1}) =$$

$$-Q + 2\boldsymbol{\pi}'(z_{j+1})\Sigma(z_j - z_{j+1}) - 2\int_0^{z_{k\,j}} \boldsymbol{\pi}'(\zeta)\Sigma d\zeta = \tag{2.19}$$

$$-Q + 2\sum_{k=1}^{m} \Sigma_{kk}\left[\pi(z_{k\,j+1})(z_{k\,j} - z_{k\,j+1}) - \int_0^{z_{k\,j}} \pi(\zeta_k)d\zeta_k\right].$$

Since the matrix R is positive definite, then one has the singular values Σ_{kk} being all positive. Also, from the geometrical meaning of

$$\pi(z_{k\,j+1})(z_{k\,j} - z_{k\,j+1}) - \int_0^{z_{k\,j}} \pi(\zeta_k)d\zeta_k \, ,$$

this term is always negative if $\pi(\cdot)$ is monotone and odd. Because $\phi(\cdot)$ is monotone and odd, and because it is a one to one function, it follows that $\phi^{-1}(\cdot)$ is odd and monotone. Hence, since $\boldsymbol{\pi}'(z_j) = \phi^{-1\prime}(\Lambda^{-1}z_j)\Lambda$, it follows that $\pi(\cdot)$ is monotone and odd. This implies that $V_j(x,u_{j+1}) \leq 0$ and that $V_j(x)$ is a Lyapunov function for u_{j+1} on Ω. Following Definition 2.1, u_{j+1} is admissible on Ω.

For the second part of the lemma, along the trajectories of $f + gu_{j+1}$, and $\forall x_0$ one has

$$V_{j+1} - V_j = \int_0^{\infty}\left\{Q(x(\tau,x_0,u_{j+1})) + 2\int_0^{u_{j+1}(x(\tau,x_0,u_{j+1}))} \phi^{-1}(v)Rdv\right\}d\tau -$$

$$\int_0^{\infty}\left\{Q(x(\tau,x_0,u_{j+1})) + 2\int_0^{u_j(x(\tau,x_0,u_{j+1}))} \phi^{-1}(v)Rdv\right\}d\tau = \tag{2.20}$$

$$-\int_0^{\infty}\left(V_{x\,j+1}{}' - V_{x\,j}{}'\right)\left[f + gu_{j+1}\right]d\tau.$$

Because $LE(V_{j+1}, u_{j+1}) = 0$, $LE(V_j, u_j) = 0$

$$V_{x\,j}{}' f = -V_{x\,j}{}' g u_j - Q - 2\int_0^{u_j} \phi^{-1}(v) R dv, \tag{2.21}$$

$$V_{x\,j+1}{}' f = -V_{x\,j+1}{}' g u_{j+1} - Q - 2\int_0^{u_{j+1}} \phi^{-1}(v) R dv. \tag{2.22}$$

Substituting (2.21) and (2.22) in (2.20), one obtains

$$\begin{aligned} &V_{j+1}(x_0) - V_j(x_0) = \\ &-2\int_0^{\infty} \left\{ \phi^{-1}(u_{j+1}) R (u_{j+1} - u_j) - \int_{u_j}^{u_{j+1}} \phi^{-1}(v) R dv \right\} d\tau. \end{aligned} \tag{2.23}$$

By decoupling the equation (2.23) using $R = \Lambda \Sigma \Lambda$, it can be shown that $V_{j+1}(x_0) - V_j(x_0) \le 0$ when $\phi(\cdot)$ is nondecreasing. Moreover, it can be shown by contradiction that $V^*(x_0) \le V_{j+1}(x_0)$. ■

The next theorem is a key result on which the rest of the chapter is justified. It shows that policy iterations on the saturated control law converges to the optimal saturated control law for the given actuator saturation model $\phi(\cdot)$. But first the following definition is required.

Definition 2.2. *Uniform Convergence: A sequence of functions* $\{f_n\}$ *converges uniformly to* f *on a set* Ω *if* $\forall \varepsilon > 0, \exists N(\varepsilon) : n > N \Rightarrow |f_n(x) - f(x)| < \varepsilon \; \forall x \in \Omega$, *or equivalently* $\sup_{x \in \Omega} |f_n(x) - f(x)| < \varepsilon$, *where* $|\;|$ *is the absolute value.*

Theorem 2.1. *If* $u_0 \in \Psi(\Omega)$, *then* $u_j \in \Psi(\Omega), \forall j \ge 0$. *Moreover,* $V_j \to V^*, u_j \to u^*$

uniformly on Ω.

Proof. From Lemma 2.1, it can be shown by induction that $u_j \in \Psi(\Omega), \forall j \geq 0$. Furthermore, Lemma 2.1 shows that V_j is a monotonically decreasing sequence and bounded below by $V^*(x)$. Hence V_j converges pointwise to V_∞. Because Ω is compact, then uniform convergence follows immediately from Dini's theorem, [6]. Due to the uniqueness of the value function [50][55], it follows that $V_\infty = V^*$. Controllers u_j are admissible, therefore they are continuous having unique trajectories due to the locally Lipschitz continuity assumptions on the dynamics. Since (2.2) converges uniformly to V^*, this implies that system's trajectories converges $\forall x_0 \in \Omega$. Therefore $u_j \to u_\infty$ uniformly on Ω. If dV_j/dx converges uniformly to dV^*/dx, one concludes that $u_\infty = u^*$. To prove that $dV_j/dx \to dV^*/dx$ uniformly on Ω, note that dV_j/dx converges uniformly to some continuous function J. Since $V_j \to V^*$ uniformly and dV_j/dx exists $\forall j$, hence it follows that the sequence dV_j/dx is term-by-term differentiable, [6], and $J = dV^*/dx$. ■

The following is a result from [14] which is tailored here to the case of saturated control inputs. It basically guarantees that improving the control law does not reduce the region of asymptotic stability of the initial saturated control law.

Corollary 2.1. *If* Ω^* *denotes the region of asymptotic stability (RAS) of the constrained optimal control* u^**, then* Ω^* *is the largest region of asymptotic stability of any other admissible control law.*

Proof. The proof is by contradiction. Lemma 1 showed that the saturated control u^* is asymptotically stable on Ω_0, where Ω_0 is the stability region of the saturated control u_0. Assume that u_{Largest} is an admissible controller with the largest region of asymptotic stability Ω_{Largest}. Then, there is $x_0 \in \Omega_{\text{Largest}}, x_0 \notin \Omega^*$. From Theorem 2.1, $x_0 \in \Omega^*$ which completes the proof. ■

Note that there may be stabilizing saturated controls that have larger stability regions than u^*, but are not admissible with respect to $Q(x)$ and the system (f, g).

2.4 Nonquadratic Performance Functionals for Minimum-Time and Constrained States Control

2.4.1 Minimum-Time Problems

For a system with saturated actuators, one maybe interested in finding the control signal required to drive the system to the origin in minimum time. This requirement can be addressed by the following nonquadratic performance functional

$$V = \int_0^\infty \left[\tanh(x'Qx) + 2\int_0^u \phi^{-1}(v) R dv \right] dt. \tag{2.24}$$

By choosing the coefficients of the weighting matrix R very small, and for $x^T Q x \gg 0$, the performance functional becomes,

$$V = \int_0^{t_s} 1 \, dt, \tag{2.25}$$

and for $x^T Q x \approx 0$, the performance functional becomes,

$$V=\int_{t_s}^{\infty}\left[x^TQx+2\int_0^u\phi^{-1}(v)Rdv\right]dt. \tag{2.26}$$

Equation (2.25) represents usually performance functionals used in minimum-time optimization because the only way to minimize (2.25) is by minimizing t_s.

Around the time t_s, one has the performance functional slowly switching to a nonquadratic regulator that takes into account the actuator saturation. Note that this method allows an easy formulation of a minimum-time problem, and that the solution will follow using the policy iteration technique. The solution is a nearly minimum-time controller that is easier to find compared with techniques aimed at finding the exact minimum-time controller. Finding an exact minimum-time controller requires finding a bang-bang controller based on a switching surface that is hard to determine [50], [43].

2.4.2 Constrained States

In literature, there exists several techniques that finds a domain of initial states such that starting within this domain guarantees a specific control policy will not violate the constraints, [31]. However, one is interested in improving given control laws so that they do not violate specific state space constraints. For this the following nonquadratic performance functional can be chosen

$$Q(x,k)=x'Qx+\sum_{l=1}^{n_c}\left(\frac{x_l}{B_l-\alpha_l}\right)^{2k} \tag{2.27}$$

where $n_c, B_l,$ are the number of constrained states, the upper bound on x_l respectively. The integer k is positive, and α_l is a small positive number. As k increases, and

$\alpha_l \to 0$, the nonquadratic term will dominate the quadratic term when the state space constraints are violated. However, the nonquadratic term will be dominated by the quadratic term when the state space constraints are not violated. Note that in this approach, the constraints are considered soft constraints that can be hardened by using higher values for k and smaller values for α_l.

2.5 Conclusion

In this chapter, policy iterations for optimal control of constrained input systems is discussed. Having the policy iteration established for constrained input systems, in the next chapter a neural network approximation of the value function is introduced, and the policy iterations method is employed in a least-squares sense over a mesh with certain size on Ω. This is far simpler than the Galerkin approximation appearing in [15], [16].

CHAPTER 3

NEARLY H_2 OPTIMAL NEURAL NETWORK CONTROL FOR CONTRAINED INPUT SYSTEMS

Although equation (2.9) is a linear differential equation, when substituting (2.10) into (2.9), it is still difficult to solve for the cost function $V_j(x)$. Therefore, Neural Nets are now used to approximate the solution for the cost function $V_j(x)$ at each policy iteration j. Moreover, for the approximate integration, a mesh is introduced in $\mathbb{R}^n$. This yields an efficient, practical, and computationally tractable solution algorithm for general nonlinear systems with saturated controls. This chapter provides a theoretically rigorous justification of this algorithm.

The solution technique of this chapter combines the policy iteration method with the method of weighted residuals to get a least squares solution of the HJB that is formulated using a nonquadratic functional to encode constraints on the input. In section 3.5 are some numerical examples to demonstrate the techniques presented in this chapter and that serve as a tutorial for other dynamical systems.

3.1 A Neural Network Solution to the *LE*(*V*,*u*)

It is well known that neural networks can be used to approximate smooth functions on prescribed compact sets [49]. Since our analysis is restricted to a set within the stability region, neural networks are natural for our application. Therefore, to successively solve (2.9), (2.10) for bounded controls, one can approximate V_j with

$$\hat{V}_j(x) = \sum_{k=1}^{L} w_{k\,j}\sigma_k(x) = \mathbf{w}_j{}'\boldsymbol{\sigma}_L(x) \tag{3.1}$$

which is a neural network with the activation functions $\sigma_k(x) \in C^1(\Omega)$, $\sigma_j(0) = 0$. The neural network weights are $w_{k\,j}$ and L is the number of hidden-layer neurons. Vectors $\boldsymbol{\sigma}_L(x) \equiv [\sigma_1(x)\, \sigma_2(x) \cdots \sigma_L(x)]'$, $\mathbf{w}_j \equiv [w_{1\,j}\ w_{2\,j} \cdots w_{L\,j}]'$ are the vector activation function and the vector weight respectively. The neural network weights will be tuned to minimize the residual error in a least-squares sense over a set of points within the stability region Ω of the initial stabilizing control. The least squares solution attains the lowest possible residual error with respect to the Neural Network weights.

For the $LE(V,u) = 0$, the solution V is replaced with V_L having a residual error

$$LE\left(\hat{V}(x) = \sum_{k=1}^{L} w_k\sigma_k(x), u\right) = e_L(x)\,. \tag{3.2}$$

To find the least squares solution, the method of weighted residuals is used [28]. The weights $\mathbf{w}_L$ are determined by projecting the residual error onto $de_L(x)/d\mathbf{w}_L$ and setting the result to zero $\forall x \in \Omega$ using the inner product, i.e.

$$\left\langle \frac{de_L(x)}{d\mathbf{w}}, e_L(x) \right\rangle = 0\,, \tag{3.3}$$

where $\langle \mathrm{f,g} \rangle = \int_{\Omega} fg dx$ is a Lebesgue integral. Equation (3.3) becomes,

$$\langle \nabla \boldsymbol{\sigma}_L (f+gu), \nabla \boldsymbol{\sigma}_L (f+gu) \rangle \mathbf{w} + \left\langle Q + 2\int_0^u \phi^{-1}(v) R dv, \nabla \boldsymbol{\sigma}_L (f+gu) \right\rangle = 0 . \quad (3.4)$$

The following technical results are needed.

Lemma 3.1. *If the set* $\{\sigma_k\}_1^L$ *is linearly independent and* $u \in \Psi(\Omega)$*, then the set*

$$\left\{ \nabla \sigma_k{}'(f+gu) \right\}_1^L \quad (3.5)$$

is also linearly independent.

Proof. See [16]. ■

Because of Lemma 3.1, $\langle \nabla \boldsymbol{\sigma}_L (f+gu), \nabla \boldsymbol{\sigma}_L (f+gu) \rangle$ is of full rank, and thus is invertible. Therefore a unique solution for $\mathbf{w}$ exists and computed as

$$\mathbf{w} = -\langle \nabla \boldsymbol{\sigma}_L (f+gu), \nabla \boldsymbol{\sigma}_L (f+gu) \rangle^{-1} \cdot \left\langle Q + 2\int_0^u \phi^{-1}(v) R dv, \nabla \boldsymbol{\sigma}_L (f+gu) \right\rangle . \quad (3.6)$$

Having solved for the neural net weights, the improved control is given by

$$\hat{u} = -\phi\left(\frac{1}{2} R^{-1} g'(x) \nabla \boldsymbol{\sigma}_L{}' \mathbf{w} \right). \quad (3.7)$$

Equations (3.6) and (3.7) are successively solved at each policy iteration i until convergence.

3.2 Convergence of the Method of Least-Squares to the Solution of the *LE*(*V*,*u*)

In what follows, convergence results associated with the method of least squares approach to solve for the cost function the LE equation using the Fourier series expansion (3.1) are shown. But before this, the following notations and definitions associated with convergence issues are considered.

Definition 3.1. *Convergence in the Mean: A sequence of functions* $\{f_n\}$ *that is Lebesgue-integrable on a set* Ω, $L_2(\Omega)$, *is said to converge in the mean to* f *on* Ω *if* $\forall \varepsilon > 0, \exists N(\varepsilon) : n > N \Rightarrow \|f_n(x) - f(x)\|_{L_2(\Omega)} < \varepsilon$, *where* $\|f\|_{L_2(\Omega)}^2 = \langle f, f \rangle$.

The convergence proofs for the least squares method is done in the Sobolev function space setting. This space allows defining functions that are $L_2(\Omega)$ with their partial derivatives.

Definition 3.2. *Sobolev Space* $H^{m,p}(\Omega)$: *Let* Ω *be an open set in* $\mathbb{R}^n$ *and let* $u \in C^m(\Omega)$. *Define a norm on* u *by*

$$\|u\|_{m,p} = \sum_{0 \le |\alpha| \le m} \left(\int_{\Omega} |D^{\alpha} u(x)|^p \, dx \right)^{1/p}, \quad 1 \le p < \infty .$$

This is the Sobolev norm in which the integration is the Lebesgue integration. The completion of $\left\{ u \in C^m(\Omega) : \|u\|_{m,p} < \infty \right\}$ *with respect to* $\|\ \|_{m,p}$ *is the Sobolev space* $H^{m,p}(\Omega)$. *For* $p = 2$, *the Sobolev space is a Hilbert space, [5].*

The LE equation can be written using the linear operator A defined on the Hilbert space $H^{1,2}(\Omega)$

$$\overbrace{V_x'\left(f + gu\right)}^{AV} = \overbrace{-Q - W(u)}^{P} .$$

In [59], it is shown that if the set $\left\{\sigma_j\right\}_1^L$ is complete, and the operator A and its inverse are bounded, then $\left\|A\hat{V}-AV\right\|_{L_2(\Omega)}\to 0$ and $\left\|\hat{V}-V\right\|_{L_2(\Omega)}\to 0$. However, for the LE equation, it can be shown that these sufficiency conditions are violated.

Neural networks based on power series have an important property that they are differentiable. This means that they can approximate uniformly a continuous function with all its partial derivatives of order m using the same polynomial, by differentiating the series termwise. This type of series is m-uniformly dense. This is known as the High Order Weierstrass Approximation theorem. Other types of neural networks not necessarily based on power series that are m-uniformly dense are studied in [35].

Lemma 3.2. *High Order Weierstrass Approximation Theorem: Let* $f(x)\in C^m(\Omega)$ *in the compact set* Ω*, then there exists a polynomial,* $f_N(x)$*, such that it converges uniformly to* $f(x)\in C^m(\Omega)$*, and such that all its partial derivatives up to order* m *converges uniformly, [28], [35].*

Lemma 3.3. *Given* N *linearly independent set of functions* $\{f_n\}$*. Then*

$$\left\|\alpha_N f_N\right\|_{L_2(\Omega)}^2 \to 0 \Leftrightarrow \left\|\alpha_N\right\|_{l_2}^2 \to 0 .$$

Proof. To show the sufficiency part, note that the Gram matrix, $G=\langle f_N, f_N\rangle$, is positive definite. Therefore, $\alpha_N^T G_N \alpha_N \geq \underline{\lambda}(G_N)\left\|\alpha_N\right\|_{l_2}^2$, $\underline{\lambda}(G_N)>0\ \forall N$. If $\alpha_N{}' G_N \alpha_N \to 0$, then $\left\|\alpha_N\right\|_{l_2}^2 = \alpha_N{}' G_N \alpha_N \big/ \underline{\lambda}(G_N) \to 0$ because $\underline{\lambda}(G_N)>0\ \forall N$.

To show the necessity part, note that

$$\|\alpha_N\|^2_{L_2(\Omega)} - 2\|\alpha_N f_N\|^2_{L_2(\Omega)} + \|f_N\|^2_{L_2(\Omega)} = \|\alpha_N - f_N\|^2_{L_2(\Omega)},$$
$$2\|\alpha_N f_N\|^2_{L_2(\Omega)} = \|\alpha_N\|^2_{L_2(\Omega)} + \|f_N\|^2_{L_2(\Omega)} - \|\alpha_N - f_N\|^2_{L_2(\Omega)},$$

Using the Parallelogram Law

$$\|\alpha_N - f_N\|^2_{L_2(\Omega)} + \|\alpha_N + f_N\|^2_{L_2(\Omega)} = 2\|\alpha_N\|^2_{L_2(\Omega)} + 2\|f_N\|^2_{L_2(\Omega)},$$

As $N \to \infty$

$$\begin{aligned} \|\alpha_N - f_N\|^2_{L_2(\Omega)} + \|\alpha_N + f_N\|^2_{L_2(\Omega)} &= \overbrace{2\|\alpha_N\|^2_{L_2(\Omega)}}^{\to 0} + 2\|f_N\|^2_{L_2(\Omega)}, \\ &\Rightarrow \|\alpha_N - f_N\|^2_{L_2(\Omega)} \to \|f_N\|^2_{L_2(\Omega)}, \\ &\Rightarrow \|\alpha_N + f_N\|^2_{L_2(\Omega)} \to \|f_N\|^2_{L_2(\Omega)}. \end{aligned}$$

As $N \to \infty$

$$2\|\alpha_N f_N\|^2_{L_2(\Omega)} = \overbrace{\|\alpha_N\|^2_{L_2(\Omega)}}^{\to 0} + \|f_N\|^2_{L_2(\Omega)} - \overbrace{\|\alpha_N - f_N\|^2_{L_2(\Omega)}}^{\to \|f_N\|^2_{L_2(\Omega)}} \to 0, .$$

Therefore, $\|\alpha_N\|^2_{l_2} \to 0 \Rightarrow \|\alpha_N f_N\|^2_{L_2(\Omega)} \to 0$. ■

Before discussing the convergence results for the method of least squares, the following four assumptions are needed.

Assumption 3.1. *The LE solution is positive definite. This is guaranteed for stabilizable dynamics and when the performance functional satisfies zero-state observability.*

Assumption 3.2. *The system's dynamics and the performance integrands* $Q(x) + W(u(x))$ *are such that are such that the solution of the LE is continuous and differentiable, therefore, belonging to the Sobolev space* $V \in H^{1,2}(\Omega)$.

Assumption 3.3. *One can choose a complete coordinate elements* $\{\sigma_j\}_1^\infty \in H^{1,2}(\Omega)$ *such that the solution* $V \in H^{1,2}(\Omega)$ *and its partial derivatives* $\{\partial V/\partial x_1, \ldots, \partial V/\partial x_n\}$ *can be approximated uniformly by the infinite series built from* $\{\sigma_j\}_1^\infty$.

Assumption 3.4. *The sequence* $\{\psi_j = A\sigma_j\}$ *is linearly independent and complete.*

In general the infinite series, constructed from the complete coordinate elements $\{\sigma_j\}_1^\infty$, need not be differentiable. However, from Lemma 3.1 and [35], it is known that several types of neural networks can approximate a function and all its partial derivatives uniformly.

Linear independence of $\{\psi_j\}$ follows from Lemma 3.1. While *completeness* follows from Lemma 3.2 and [35],

$$\forall V, \varepsilon \quad \exists L: \quad \left|\hat{V} - V\right| < \varepsilon \text{ and } \forall k \left|\partial\hat{V}/\partial x_k - \partial V/\partial x_k\right| < \varepsilon .$$

This implies that $L \to 0$

$$\sup_{x \in \Omega}\left|A\hat{V} - AV\right| \to 0 \Rightarrow \left\|A\hat{V} - AV\right\|_{L_2(\Omega)} \to 0 ,$$

and therefore *completeness* of the set $\{\psi_j\}$ is established.

The next theorem uses these assumptions to conclude convergence results of the least squares method which is placed in the Sobolev space $H^{1,2}(\Omega)$.

Theorem 3.1. *If assumptions 3.1-3.4 hold, then approximate solutions exist for the LE equation using the method of least squares and are unique for each* L. *In addition, the* *following results are achieved:*

R1) $\left\| LE(\hat{V}(x)) - LE(V(x)) \right\|_{L_2(\Omega)} \to 0$,

R2) $\left\| \hat{V}_x - V_x \right\|_{L_2(\Omega)} \to 0$,

R3) $V_x{}' f + \frac{1}{4\gamma^2} V_x{}' kk' V_x + h'h \le 0, V(0) = 0$.

Proof. *Existence* of a least squares solution for the LE equation can be easily shown. The least squares solution V_L is nothing but the solution of the minimization problem

$$\left\| A\hat{V} - P \right\|^2 = \min_{\Pi \in S_L} \left\| A\Pi - P \right\|^2 = \min_{\mathbf{w}} \left\| \mathbf{w}' \boldsymbol{\psi}_L - P \right\|^2,$$

where S_L is the span of $\{\sigma_1, \ldots, \sigma_L\}$.

Uniqueness follows from the linear independence of $\{\psi_1, \ldots, \psi_L\}$.

The first results, R1, follows from the *completeness* of $\{\psi_j\}$.

To show the second result, R2, write the LE equation in terms of its series expansion on Ω with coefficients c_j

$$LE\left(\hat{V} = \sum_{i=1}^{L} w_i \sigma_i \right) - \overbrace{LE(V = \sum_{i=1}^{\infty} c_i \sigma_i)}^{=0} = \varepsilon_L(x),$$

$$(\mathbf{w} - \mathbf{c}_L)' \nabla \boldsymbol{\sigma}_L \left(f + gu \right) = \varepsilon_L(x) + \overbrace{\sum_{i=L+1}^{\infty} c_i \frac{d\sigma_i}{dx} (f + gu)}^{e_L(x)}.$$

Note that $e_L(x)$ converges uniformly to zero due to Lemma 3.2, and hence converges in the mean. On the other hand $\varepsilon_L(x)$ is shown to converge in the mean to zero using the least squares method as seen in R1. Therefore,

$$\begin{aligned}&\|(\mathbf{w}-\mathbf{c}_L)'\nabla\boldsymbol{\sigma}_L(f+gu)\|_{L_2(\Omega)}^2 = \|\varepsilon_L(x)+e_L(x)\|_{L_2(\Omega)}^2 \le \\ &2\|\varepsilon_L(x)\|_{L_2(\Omega)}^2 + 2\|e_L(x)\|_{L_2(\Omega)}^2 \to 0\end{aligned}.$$

Because $\nabla\boldsymbol{\sigma}_L(f+gu)$ is linearly independent, using Lemma 3.3, one concludes that $\|\mathbf{w}-\mathbf{c}_L\|_{l2}^2 \to 0$. Therefore, because the set $\{d\sigma_i/dx\}$ is linearly independent, one concludes from Lemma 3.3 that $\|(\mathbf{w}-\mathbf{c}_L)'\nabla\boldsymbol{\sigma}_L\|_{L_2(\Omega)}^2 \to 0$. Because the infinite series with c_j converges uniformly it follows that $\left\|\hat{V}_x - V_x\right\|_{L_2(\Omega)} \to 0$.

Finally, the third result, R3, follows by noting that $g(x)$ is continuous and therefore bounded on Ω, this implies using R2 that

$$\left\|-\frac{1}{2}R^{-1}g'(\hat{V}_x - V_x)\right\|_{L_2(\Omega)}^2 \le \left\|-\frac{1}{2}R^{-1}g'\right\|_{L_2(\Omega)}^2 \left\|(\hat{V}_x - V_x)\right\|_{L_2(\Omega)}^2 \to 0.$$

Denote $\hat{\alpha}_k(x) = -\frac{1}{2}g_k{}'\hat{V}_x$, $\alpha_k(x) = -\frac{1}{2}g_k{}'V_x$

$$\begin{aligned}u_L - u &= -\phi(\frac{1}{2}g'\hat{V}_x) + \phi(\frac{1}{2}g'V_x), \\ &= \begin{bmatrix}\phi(\hat{\alpha}_1(x)) - \phi(\alpha_1(x)) \\ \vdots \\ \phi(\hat{\alpha}_m(x)) - \phi(\alpha_m(x))\end{bmatrix}.\end{aligned}$$

Because $\phi(\cdot)$ is smooth, and under the assumption that its first derivative is bounded by a constant M, then one has $\phi(\hat{\alpha}_j) - \phi(\alpha_j) \le M(\hat{\alpha}_j(x) - \alpha_j(x))$, therefore

$$\left\|\hat{\alpha}_j(x) - \alpha_j(x)\right\|_{L_2(\Omega)} \to 0 \Rightarrow \left\|\phi(\hat{\alpha}_j) - \phi(\alpha_j)\right\|_{L_2(\Omega)} \to 0,$$

hence R3 follows. ■

Corollary 3.1. *If the results of Theorem 3.1 hold, then*

$$\sup_{x\in\Omega}\left|\hat{V}_x - V_x\right| \to 0, \quad \sup_{x\in\Omega}\left|\hat{V} - V\right| \to 0, \quad \sup_{x\in\Omega}\left|\hat{u} - u\right| \to 0.$$

Proof. As the coefficients of the neural network, w_j, series converge to the coefficient of the uniformly convergent series, c_j, that is $\|\mathbf{w} - \mathbf{c}_L\|_{l2}^2 \to 0$. And since the mean error goes to zero in R2 and R3, hence uniform convergence follows. ■

The next theorem is required to show the admissibility of the controller derived using the technique presented in this chapter.

Corollary 3.2. *Admissibility of* $\hat{u}(x)$ *:*

$$\exists M : L \geq M, \hat{u} \in \Psi(\Omega).$$

Proof. Consider the following LE equation

$$\dot{V}_j(x,\hat{u}_{j+1}) = -Q - 2\phi^{-1\prime}(u_{j+1})R(\hat{u}_{j+1} - u_{j+1}) - 2\int_0^{u_{j+1}} \phi^{-1}(v)Rdv$$
$$\overbrace{-2\int_{u_{j+1}}^{u_j} \phi^{-1}(v)Rdv + 2\phi^{-1\prime}(u_{j+1})R(u_j - u_{j+1})}^{\leq 0}.$$

Since $\hat{u}_{j+1}$ is guaranteed to be within a tube around u_{j+1} because $\hat{u}_{j+1} \to u_{j+1}$ uniformly. Therefore one can easily see that

$$\phi^{-1\prime}(u_{j+1})R\hat{u}_{j+1} \geq 1/2 \cdot \phi^{-1\prime}(u_{j+1})Ru_{j+1} + \alpha\int_0^{u_{j+1}} \phi^{-1}Rdv.$$

with $\alpha > 0$ is satisfied $\forall x \in \Omega \cap \Omega_1(\varepsilon_L)$ where $\Omega_1(\varepsilon_L) \subseteq \Omega$ containing the origin. Hence $\dot{V}_j(x, \hat{u}_{j+1}) < 0 \;\; \forall x \in \Omega \cap \Omega_1(\varepsilon_L)$. Given that $\hat{u}_{j+1}(0) = 0$, and from the continuity of $\hat{u}_{j+1}$, there exists $\Omega_2(\varepsilon_L) \subseteq \Omega_1(\varepsilon_L)$ containing the origin for which $\dot{V}_j(x, \hat{u}_{j+1}) < 0$. As L increases, $\Omega_1(\varepsilon_L)$ gets smaller while $\Omega_2(\varepsilon_L)$ gets larger and the inequality is satisfied $\forall x \in \Omega$. Therefore, $\exists L_0 : L \geq L_0, \dot{V}_j(x, \hat{u}_{j+1}) < 0 \quad \forall x \in \Omega$ and hence $\hat{u} \in \Psi(\Omega)$.

■

Corollary 3.3: *Positive definiteness of* $\hat{V}(x)$: $\hat{V}(x) = 0 \Leftrightarrow x = 0$, elsewhere $\hat{V}(x) > 0$.

Proof: The proof is going to be by contradiction. Assuming that $u \in \Psi(\Omega)$, then Lemma 3.1 is satisfied. Therefore

$$\mathbf{w} = -\langle \nabla \boldsymbol{\sigma}_L (f + gu), \nabla \boldsymbol{\sigma}_L (f + gu) \rangle^{-1} \cdot \left\langle Q + 2\int_0^u \phi^{-1}(v) R dv, \nabla \boldsymbol{\sigma}_L (f + gu) \right\rangle.$$

Assume also that

$$\exists\, x_a \neq 0, \text{ s.t. } \sum_{j=1}^{L} w_j \sigma_j(x_a) = \mathbf{w}' \boldsymbol{\sigma}_L(x_a) = 0 .$$

Then,

$$-\left\langle Q + 2\int_0^u \phi^{-1}(v) R dv, \nabla \boldsymbol{\sigma}_L (f + gu) \right\rangle' \cdot \langle \nabla \boldsymbol{\sigma}_L (f + gu), \nabla \boldsymbol{\sigma}_L (f + gu) \rangle^{-1'} \boldsymbol{\sigma}_L(x_a) = 0 .$$

Note that because Lemma 3.1 is satisfied then $\langle \nabla \boldsymbol{\sigma}_L (f + gu), \nabla \boldsymbol{\sigma}_L (f + gu) \rangle^{-1}$ is a positive definite constant matrix. This implies that

$$\left\langle Q+2\int_0^u \phi^{-1}(v)R\,dv, \nabla\boldsymbol{\sigma}_L(f+gu)\right\rangle^T \boldsymbol{\sigma}_L(x_a)=0$$

One can expand this matrix representation into a series form,

$$\left\langle Q+2\int_0^u \phi^{-1}(v)R\,dv, \nabla\boldsymbol{\sigma}_L(f+gu)\right\rangle' \boldsymbol{\sigma}_L(x_a)=\sum_{j=1}^{L}\left\langle Q+2\int_0^u \phi^{-1}(v)R\,dv, \frac{d\sigma_j}{dx}(f+gu)\right\rangle \sigma_j(x_a)$$
$$=0.$$

Note that,

$$\left\langle Q+2\int_0^u \phi^{-1}(v)R\,dv, \frac{d\sigma_j}{dx}(f+gu)\right\rangle = \int_\Omega \left\{\left(Q+2\int_0^u \phi^{-1}(v)R\,dv\right)\left(\frac{d\sigma_j}{dx}(f+gu)\right)\right\}dx.$$

Thus,

$$\sum_{j=1}^{L}\int_\Omega \left\{\left(Q+2\int_0^u \phi^{-1}(v)R\,dv\right)\left(\frac{d\sigma_j}{dx}(f+gu)\right)\right\}dx\cdot\sigma_j(x_a)=0\,.$$

Using the mean value theorem, $\exists\xi\in\Omega$ such that,

$$\int_\Omega \left\{\left[Q+2\int_0^u \phi^{-1}(v)R\,dv\right]\times\left[\boldsymbol{\sigma}_L^T(x_a)\nabla\boldsymbol{\sigma}_L(f+gu)\right]\right\}dx =$$
$$\mu(\Omega)\left\{\left[Q+2\int_0^u \phi^{-1}(v)R\,dv\right]\times\left[\boldsymbol{\sigma}_L^T(x_a)\nabla\boldsymbol{\sigma}_L(f+gu)\right]\right\}(\xi).$$

where $\mu(\Omega)$ is the Lebesgue measure of Ω.

This implies that,

$$
\begin{aligned}
0 &= \sum_{j=1}^{L} \mu(\Omega)\left[\left(Q + 2\int_0^u \phi^{-1}(v)Rdv\right)\cdot \frac{d\sigma_j}{dx}(f+gu)\right](\xi)\times\sigma_j(x_a) \\
&= \mu(\Omega)\left[Q + 2\int_0^u \phi^{-1}(v)Rdv\right](\xi)\cdot\sum_{j=1}^{L}\left[\frac{d\sigma_j}{dx}(f+gu)\right](\xi)\times\sigma_j(x_a) \\
&\Rightarrow \sum_{j=1}^{L}\left[\frac{d\sigma_j}{dx}(f+gu)\right](\xi)\times\sigma_j(x_a) = 0.
\end{aligned}
$$

Now, one can select a constant $\sigma_j(x_a)$ to be equal to a constant c_j. Thus one can rewrite the above formula as follows:

$$
\sum_{j=1}^{L} c_j\left[\frac{d\sigma_j}{dx}(f+gu)\right](\xi) = 0.
$$

Since ξ depends on Ω, which is arbitrarily, this means that, $\nabla\sigma_j(f+gu)$ is not linearly independent, which contradicts our assumption. ■

Corollary 3.4. *It can be shown that* $\sup_{x\in\Omega}|\hat{u}(x)-u(x)|\to 0$ *implies that* $\sup_{x\in\Omega}|J(x)-V(x)|\to 0$*, where* $LE(J,\hat{u})=0$*,* $LE(V,u)=0$.

3.3 Convergence of the Method of Least Squares to the Solution of the HJB

In this section, a theorem analogous to Theorem 3.1 which guarantees that least-squares policy iterations converge to the value function of the HJB equation (2.11) is presented.

Theorem 3.2. *Under the assumptions of Theorem 3.1, the following is satisfied* $\forall j\geq 0$*:*

i. $\sup_{x\in\Omega}\left|\hat{V}_j - V_j\right|\to 0,$ *ii.* $\sup_{x\in\Omega}\left|\hat{u}_{j+1}-u_{j+1}\right|\to 0$*,*

iii. $\exists N: L\geq N, \hat{u}_{j+1}\in\Psi(\Omega)$.

Proof. The proof is by induction.

Basis Step:

Using Corollary 3.1 and 3.2, it follows that for any $u_0 \in \Psi(\Omega)$, one has

I. $\sup_{x\in\Omega}\left|\hat{V}_0 - V_0\right| \to 0,$ II. $\sup_{x\in\Omega}\left|\hat{u}_1 - u_1\right| \to 0$

III. $\exists N : L \geq N, \hat{u}_1 \in \Psi(\Omega)$.

Inductive Step:

Assume that

i. $\sup_{x\in\Omega}\left|\hat{V}_{j-1} - V_{j-1}\right| \to 0,$ b. $\sup_{x\in\Omega}\left|\hat{u}_j - u_j\right| \to 0$

c. $\exists N : L \geq N, \hat{u}_j \in \Psi(\Omega)$.

If J_j is such that $LE(J_j, \hat{u}_j) = 0$. Then from Corollary 3.1, J_j can be uniformly approximated by $\hat{V}_j$. Moreover from assumption b and Corollary 3.4. It follows that as $\hat{u}_j \to u_j$ uniformly then $J_j \to V_j$ uniformly. Therefore $\hat{V}_j \to V_j$ uniformly.

Because $\hat{V}_j \to V_j$ uniformly, then $\hat{u}_{j+1} \to u_{j+1}$ uniformly by Corollary 3.1. From Corollary 3.2, $\exists M : L \geq M \Rightarrow \hat{u}_{j+1} \in \Psi(\Omega)$.

Hence the proof by induction is complete. ■

The next theorem is an important result upon which the algorithm proposed in Figure 3.1.

Theorem 3.3. $\forall \varepsilon > 0, \exists M, N : j \geq M, L \geq N$ *the following is satisfied*

A. $\sup_{x\in\Omega}\left|\hat{V}_j - V^*\right| < \varepsilon,$

B. $\sup_{x\in\Omega}\left|\hat{u}_j - u^*\right| < \varepsilon$,

C. $\hat{u}_j \in \Psi(\Omega)$.

Proof. The proof follows directly from Theorem 2.1 and Theorem 3.2. ■

3.4 Algorithm for Nearly Optimal Neurocontrol Design with Saturated Controls: Introducing a Mesh in $\mathbb{R}^n$

Solving the integration in (3.6) is expensive computationally. However, an integral can be fairly approximated by replacing the integral with a summation series over a mesh of points on the integration region. This results in a nearly optimal, computationally tractable solution procedure.

By introducing a mesh on Ω, with mesh size equal to Δx, one can rewrite some terms of (3.6) as follows:

$$X = \left\lfloor \nabla\boldsymbol{\sigma}_L(f+gu)\big|_{x_1} \quad \cdots \quad \nabla\boldsymbol{\sigma}_L(f+gu)\big|_{x_p} \right\rfloor' \tag{3.8}$$

$$Y = \left\lfloor Q + 2\int_0^u \phi^{-1}(v)R\,dv\Big|_{x_1} \quad \cdots \quad Q + 2\int_0^u \phi^{-1}(v)R\,dv\Big|_{x_p} \right\rfloor' \tag{3.9}$$

where p in x_p represents the number of points of the mesh. This number increases as the mesh size is reduced. Note that

$$\begin{aligned} \left\langle \nabla\boldsymbol{\sigma}_L(f+gu), \nabla\boldsymbol{\sigma}_L(f+gu)\right\rangle &= \lim_{\|\Delta x\|\to 0}\left(X'X\right)\cdot\Delta x \\ \left\langle Q + 2\int_0^u \phi^{-T}(v)R\,dv, \nabla\boldsymbol{\sigma}_L(f+gu)\right\rangle &= \lim_{\|\Delta x\|\to 0}\left(X'Y\right)\cdot\Delta x \end{aligned} \tag{3.10}$$

This implies that one can calculate $\mathbf{w}_L$ as

$$\mathbf{w} = -(X'X)^{-1}(X'Y). \tag{3.11}$$

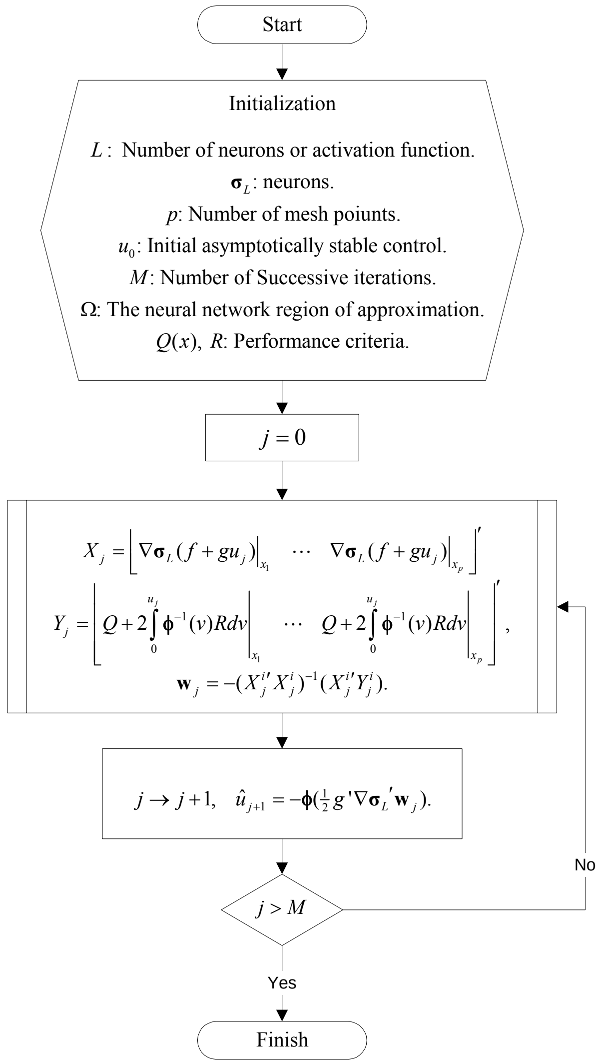


Figure 3.1 Policy iterations algorithm for nearly optimal saturated neurocontrol

One can also use Monte Carlo integration techniques in which the mesh points are sampled stochastically instead of being selected in a deterministic fashion, [27]. This allows more efficient numerical integration technique. In any case however, the numerical algorithm at the end requires solving (3.11) which is a least squares computation of the neural network weights.

Numerically stable routines that compute equations like (3.11) do exists in several software packages like MATLAB which is used to perform the simulations in this chapter.

A flowchart of the computational algorithm presented in this chapter is shown in Figure 3.1. This is an offline algorithm run a priori to obtain a neural network feedback controller that is a nearly optimal solution to the HJB equation for the constrained control input case. The neurocontrol law structure is shown in Figure 3.2. It is a neural network with activation functions given by $\boldsymbol{\sigma}$, multiplied by a function of the system's state variables.

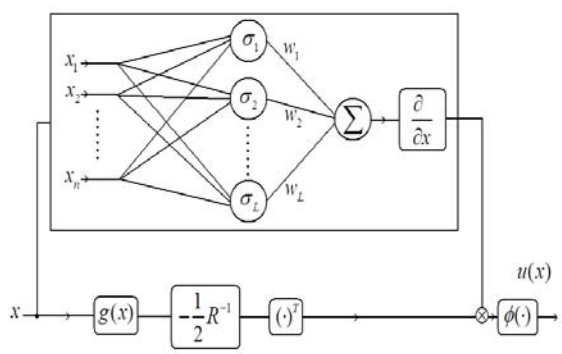


Figure 3.2 Neural-network-based nearly optimal saturated control law.

3.5 Numerical Examples

The power of the neural network control technique of finding nearly optimal nonlinear saturated controls for general systems is demonstrated. Four examples are presented.

3.5.1 Multi Input Canonical Form Linear System with Constrained Inputs

The algorithm obtained is applied to the following linear system

$$\begin{aligned}\dot{x}_1 &= 2x_1 + x_2 + x_3, \\ \dot{x}_2 &= x_1 - x_2 + u_2, \\ \dot{x}_3 &= x_3 + u_1.\end{aligned}$$

It is desired to control the system with input constraints $|u_1| \le 3, |u_2| \le 20$. This system when uncontrolled has eigenvalues with positive real parts. This systems is not asymptotically null controllable, therefore global asymptotic stabilization cannot be achieved, [74].

The algorithm developed in this chapter is used to derive a nearly optimal neurocontrol law for a specified region of stability around the origin. The following smooth function is used to approximate the value function of the system,

$$\begin{aligned}&V_{21}(x_1, x_2, x_3) = w_1 x_1^2 + w_2 x_2^2 + w_3 x_3^2 + w_4 x_1 x_2 + w_5 x_1 x_3 + \\ &w_6 x_2 x_3 + w_7 x_1^4 + w_8 x_2^4 + w_9 x_3^4 + w_{10} x_1^2 x_2^2 + w_{11} x_1^2 x_3^2 + \\ &w_{12} x_2^2 x_3^2 + w_{13} x_1^2 x_2 x_3 + w_{14} x_1 x_2^2 x_3 + w_{15} x_1 x_2 x_3^2 + \\ &w_{16} x_1^3 x_2 + w_{17} x_1^3 x_3 + w_{18} x_1 x_2^3 + w_{19} x_1 x_3^3 + w_{20} x_2 x_3^3 + \\ &w_{21} x_2^3 x_3\end{aligned}$$

Selecting the approximation for $V(x)$ is usually a natural choice guided by engineering experience and intuition. With this selection, one guarantees that $V(0) = 0$. This is a neural net with polynomial activation functions, Volterra neural network. It has 21 activation functions containing powers of the state variable of the system up to the 4th power. Neurons with 4th order power of the states variables were selected because for neurons with 2^{nd} order power of the states, the algorithm did not converge.

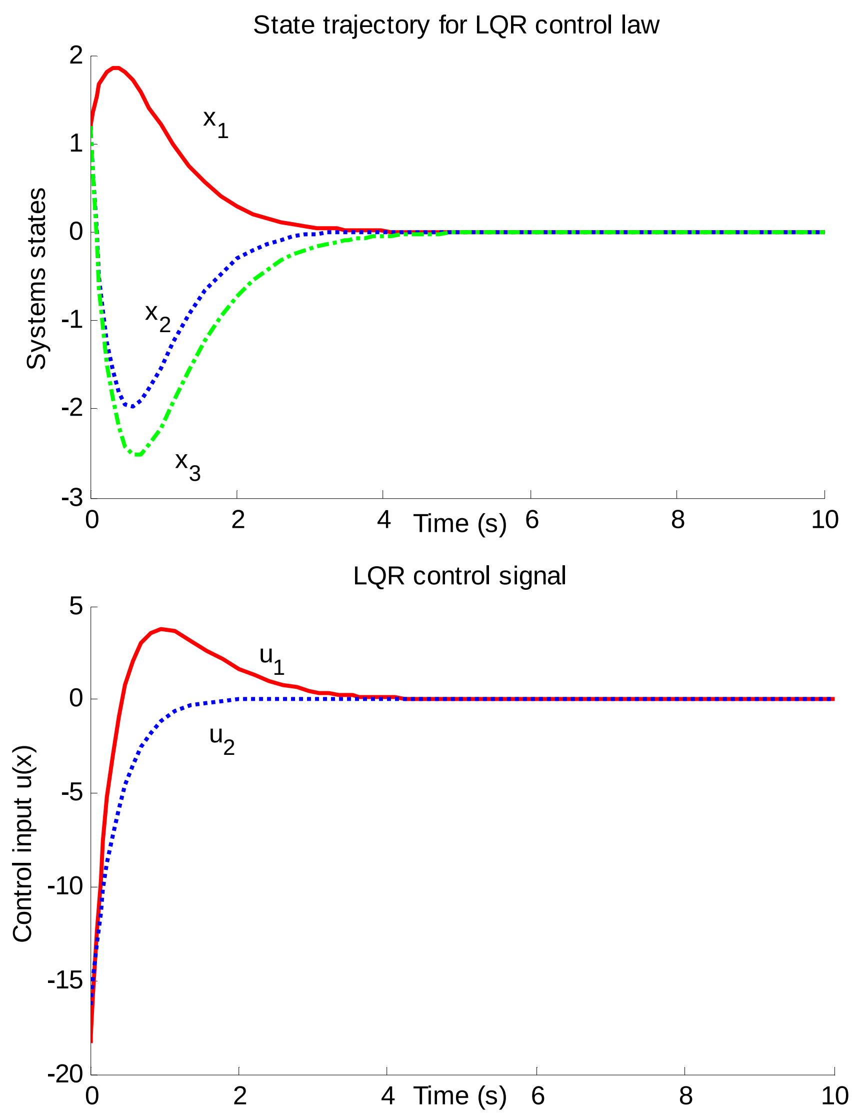


Figure 3.3 LQR optimal unconstrained control

Moreover, it is found that 6^{th} power polynomials did not improve the performance over 4^{th} power ones. The number of neurons required is chosen to guarantee the uniform convergence of the algorithm. If fewer neurons are used, then the algorithm might not properly approximate the cost function associated with the initial stabilizing control, and thus the improved control using this approximated cost might not be admissible. The activation functions for the neural network neurons selected in this example satisfy the properties of activation functions discussed in Section 3.1 and [49].

To initialize the algorithm, a stabilizing control is needed. It is very easy to find this using Linear Quadratic Regulator (LQR) for unconstrained controls. In this case, the performance functional is

$$\int_0^\infty \left(x_1^2 + x_2^2 + x_3^2 + u_1^2 + u_2^2 \right) dt \,.$$

Solving the corresponding Riccati equation, the following stabilizing unconstrained state feedback control is obtained

$$\begin{aligned} u_1 &= -8.31x_1 - 2.28x_2 - 4.66x_3, \\ u_2 &= -8.57x_1 - 2.27x_2 - 2.28x_3. \end{aligned}$$

However, when the LQR controller works through saturated actuators, the stability region shrinks. Further, this optimal control law derived for the linear case will not be optimal anymore working under saturated actuators. Fig. 3.3 shows the performance of this controller assuming working with unsaturated actuators for the initial conditions $x_i(0) = 1.2, i = 1,2,3$. Fig. 3.4 shows the performance when this control signal is bounded by $|u_1| \le 3, |u_2| \le 20$. Note how the bounds destroy the performance.

In order to model the saturation of the actuators, a nonquadratic cost performance term (2.8) is used as explained before. To show how to do this for the general case of $|u| \le A$, it is assumed that the function $\phi(cs)$ is given as $A^* \tanh(1/A \cdot cs)$, where cs is assumed to be the command signal to the actuator. Fig. 3.5 shows this for the case $|u| \le 3$.

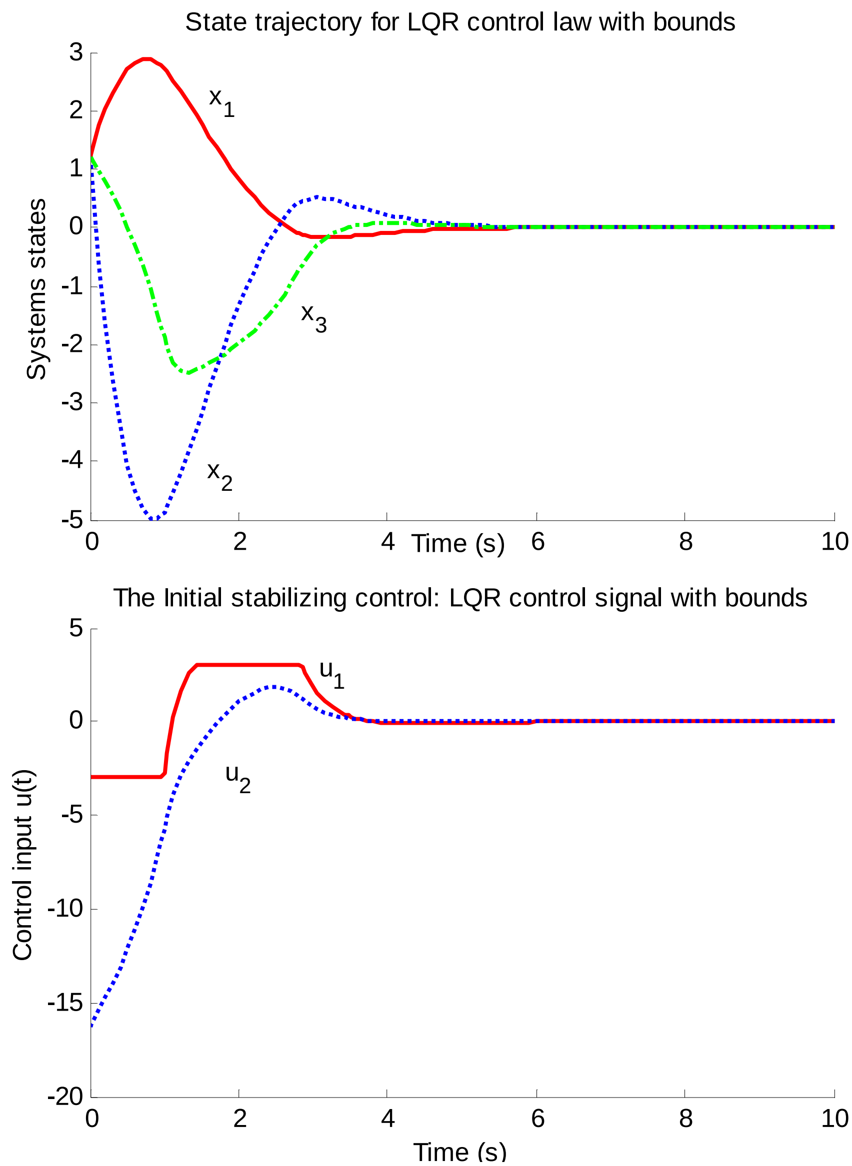


Figure 3.4 LQR control with actuator saturation

Following that, the nonquadratic cost performance is calculated to be

$$
\begin{aligned}
W(u) &= 2\int_0^u \phi^{-1}(v) R dv \\
&= 2\int_0^u \left(A \tanh^{-1}(v/A) \right)' R dv \\
&= 2A \times R \times u \times \tanh^{-1}(u/A) + A^2 \times R \times \ln\left(1 - u^2/A^2\right)
\end{aligned}
$$

This nonquadratic cost performance is then used in the algorithm to calculate the optimal bounded control. The improved bounded control law is found using the technique presented in the previous section. The algorithm is run over the region $-1.2 \le x_1 \le 1.2$, $-1.2 \le x_2 \le 1.2$, $-1.2 \le x_3 \le 1.2$ with the design parameters $R = I_{2x2}$, $Q = I_{3x3}$. This region falls within the region of asymptotic stability of the initial stabilizing control. Methods to estimate the region of asymptotic stability are discussed in [41].

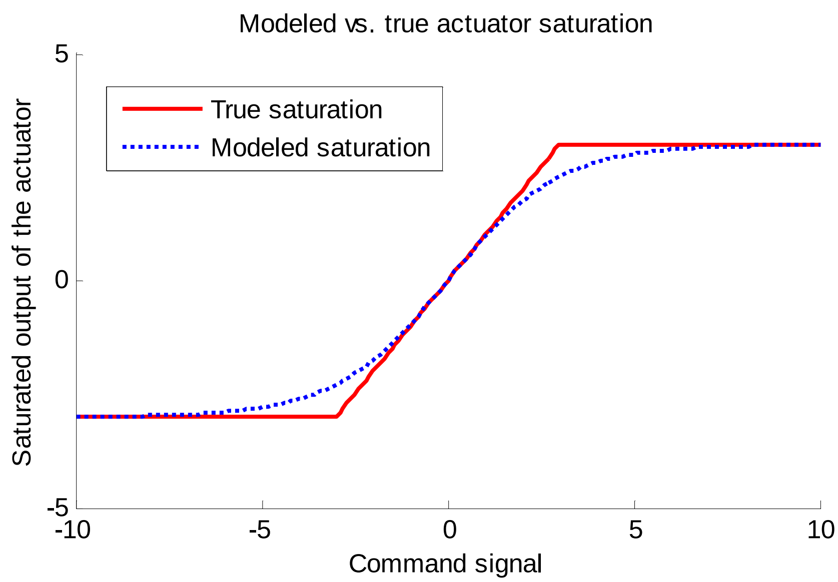


Figure 3.5 Model of saturation

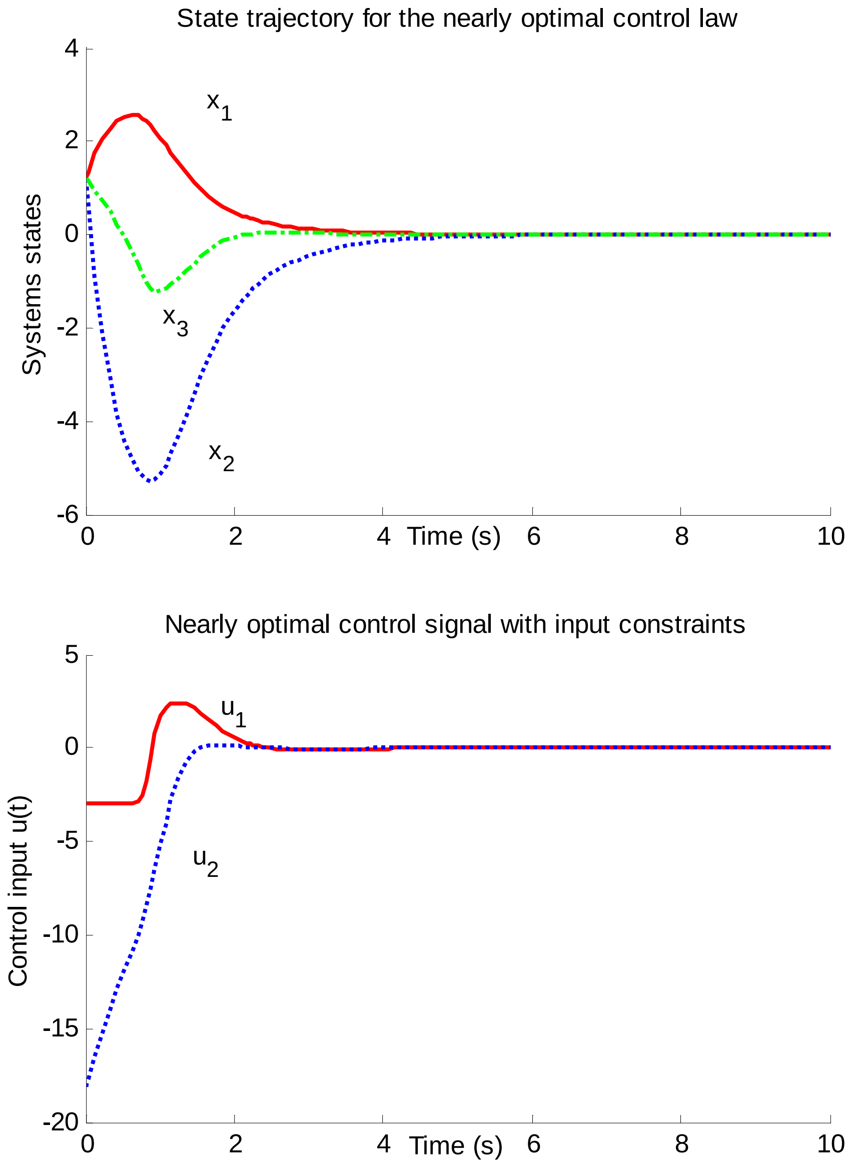


Figure 3.6 Nearly optimal nonlinear neural control law
for the linear system considering actuator saturation

After 20 policy iterations, the algorithm converges to

$$u_1 = -3\tanh\left(\frac{1}{3}\left\{\begin{array}{l} 7.7x_1 + 2.44x_2 + 4.8x_3 + 2.45x_1^3 + 2.27x_1^2 x_2 + \\ 3.7x_1 x_2 x_3 + 0.71x_1 x_2^2 + 5.8x_1^2 x_3 + 4.8x_1 x_3^2 + \\ 0.08x_2^3 + 0.6x_2^2 x_3 + 1.6x_2 x_3^2 + 1.4x_3^3 \end{array}\right\}\right)$$

$$u_2 = -20\tanh\left(\frac{1}{20}\left\{\begin{matrix} 9.8x_1 + 2.94x_2 + 2.44x_3 - 0.2x_1^3 - 0.02x_1^2x_2 + \\ 1.42x_1x_2x_3 + 0.12x_1x_2^2 + 2.3x_1^2x_3 + 1.9x_1x_3^2 + \\ 0.02x_2^3 + 0.23x_2^2x_3 + 0.57x_2x_3^2 + 0.52x_3^3 \end{matrix}\right\}\right).$$

This is a nearly optimal saturated control law in feedback strategy form. It is given in terms of the state variables and a neural net following the structure shown in Figure 3.2. The suitable performance of this saturated control law is revealed in Figure 3.6.

3.5.2 Nonlinear Oscillator with Constrained Input

Consider the nonlinear oscillator having the dynamics

$$\begin{aligned} \dot{x}_1 &= x_1 + x_2 - x_1(x_1^2 + x_2^2), \\ \dot{x}_2 &= -x_1 + x_2 - x_2(x_1^2 + x_2^2) + u. \end{aligned}$$

It is desired to control the system with control limits of $|u| \le 1$. The following smooth function is used to approximate the value function of the system,

$$\begin{aligned} V_{24}(x_1, x_2) = {} & w_1x_1^2 + w_2x_2^2 + w_3x_1x_2 + w_4x_1^4 + w_5x_2^4 + w_6x_1^3x_2 + w_7x_1^2x_2^2 + w_8x_1x_2^3 + \\ & w_9x_1^6 + w_{10}x_2^6 w_{11}x_1^5x_2 + w_{12}x_1^4x_2^2 + w_{13}x_1^3x_2^3 + w_{14}x_1^2x_2^4 + w_{15}x_1x_2^5 + w_{16}x_1^8 + \\ & w_{17}x_2^8 + w_{18}x_1^7x_2 + w_{19}x_1^6x_2^2 + w_{20}x_1^5x_2^3 + w_{21}x_1^4x_2^4 + w_{22}x_1^3x_2^5 + w_{23}x_1^2x_2^6 + w_{24}x_1x_2^7. \end{aligned}$$

This neural net has 24 activation functions containing powers of the state variable of the system up to the 8th power. In this example, the order of the neurons is higher than in the previous example to guarantee uniform convergence. The complexity of the neural network is selected to guarantee convergence of the algorithm to an admissible control law. When only up to the 6th order powers are used, convergence of the iteration to admissible controls was not observed.

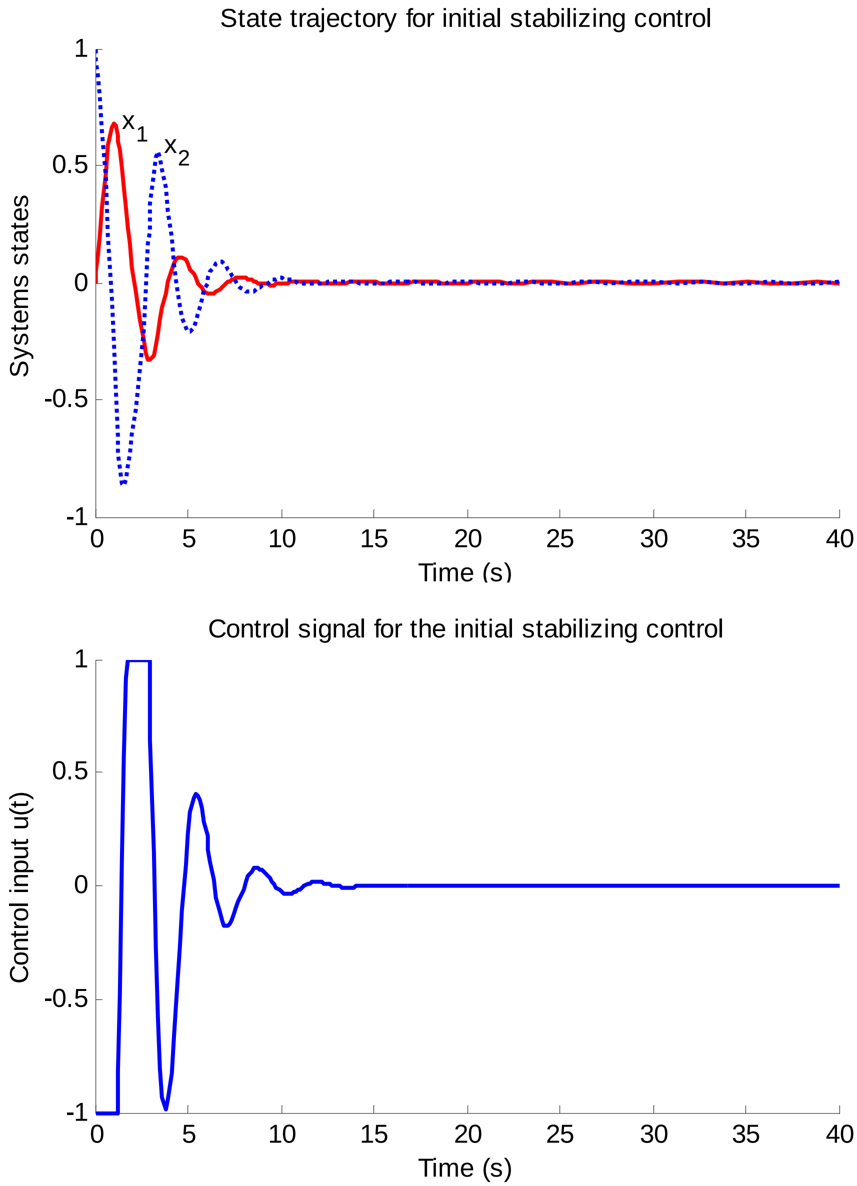


Figure 3.7 Performance of the initial stabilizing control when saturated

The unconstrained state feedback control $u = -5x_1 - 3x_2$, is used as an initial stabilizing control for the iteration. This is found after linearizing the nonlinear system around the origin, and building an unconstrained state feedback control which makes the eigenvalues of the linear system all negative. Fig. 3.7 shows the performance of the bounded controller $u = sat_{-1}^{+1}(-5x_1 - 3x_2,)$ when running it through a saturated actuator for $x_1(0) = 0, x_2(0) = 1$. Note that it is not good.

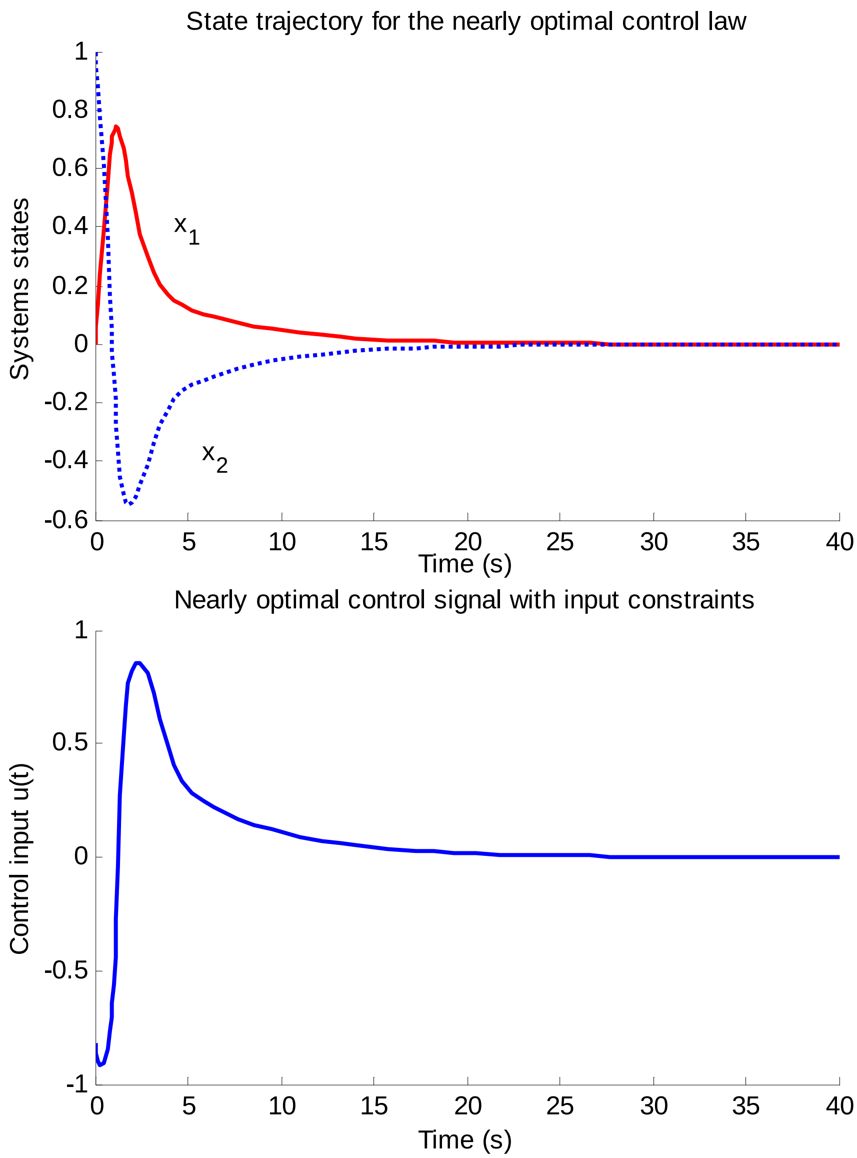


Figure 3.8 Nearly optimal nonlinear control law for the nonlinear oscillator considering actuator saturation

The nearly optimal saturated control law is now found through the technique presented in Figure 3.1. The algorithm is run over the region $-1 \le x_1 \le 1, \quad -1 \le x_2 \le 1,$

$R=1, \quad Q=I_{2x2}$. After 20 policy iterations, the nearly optimal saturated control law is found to be,

$$u=-\tanh\begin{pmatrix} 2.6x_1 + 4.2x_2 + 0.4x_2^3 - 4.0x_1^3 - 8.7x_1^2x_2 - 8.9x_1x_2^2 - 5.5x_2^5 + \\ 2.26x_1^5 + 5.8x_1^4x_2 + 11x_1^3x_2^2 + 2.6x_1^2x_2^3 + 2.00x_1x_2^4 + 2.1x_2^7 - 0.5x_1^7 - \\ 1.7x_1^6x_2 - 2.71x_1^5x_2^2 - 2.19x_1^4x_2^3 - 0.8x_1^3x_2^4 + 1.8x_1^2x_2^5 + 0.9x_1x_2^6 \end{pmatrix}$$

This is the control law in terms of a neural network following the structure shown in Figure 3.2. The suitable performance of this saturated control law is revealed in Figure 3.8. Note that the states and the saturated input in Figure 3.8 have fewer oscillations when compared to those of Figure 3.7.

3.5.3 Constrained State Linear System

Consider the following system

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= x_1 + x_2 + u \\ |x_1| &\le 3. \end{aligned}$$

For this, select the following performance functional

$$\begin{aligned} Q(x,14) &= x_1^2 + x_2^2 + \left(\frac{x_1}{3-1}\right)^{10}, \\ W(u) &= u^2. \end{aligned}$$

Note that the coefficient k is chosen to be 10, and $B_1 = 3$, and $\alpha_1 = 1$. A reason why k is selected to be 10 is that a larger value for k requires using many activation functions in which a large number of them will have to have powers higher than the value k. However, since this simulation was carried on a double precision computer, then power terms higher than 14 do not add up nicely and round-off errors seriously affect determining the weights of the neural network by causing a rank deficiency.

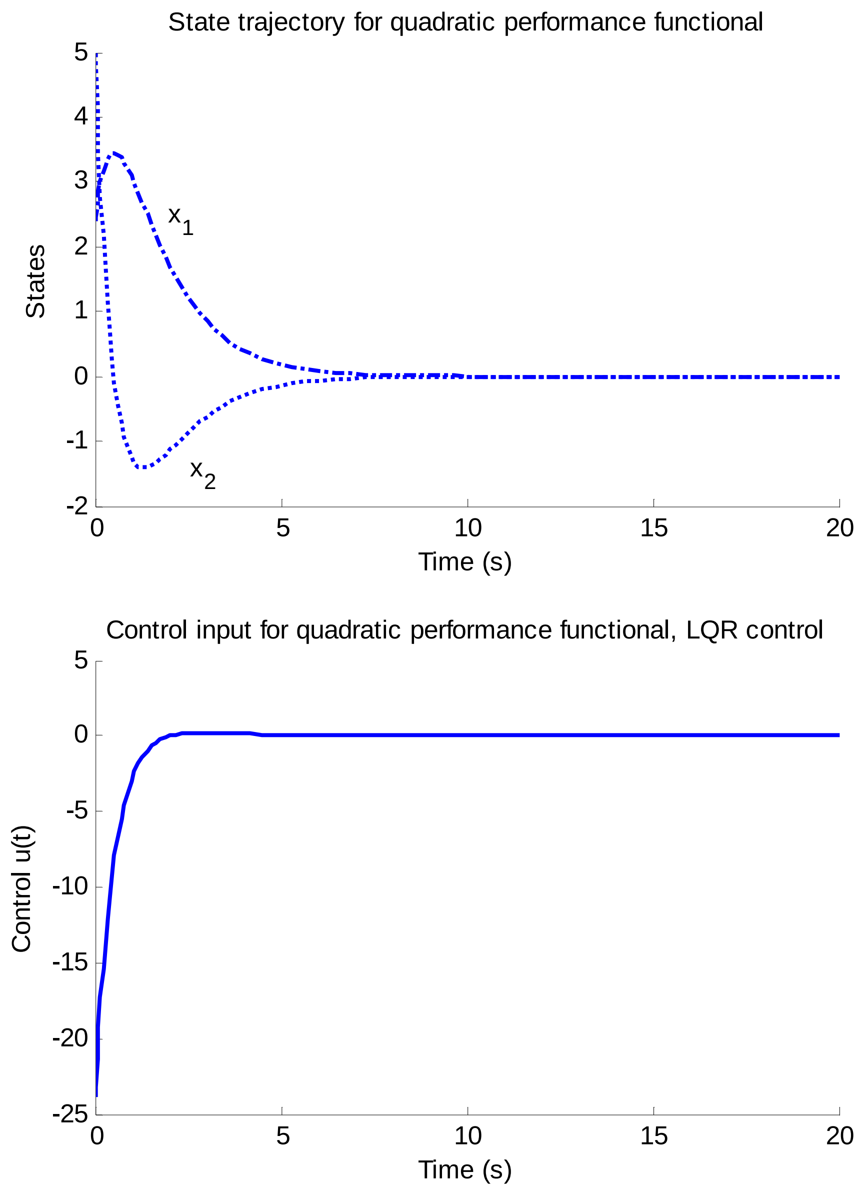


Figure 3.9 LQR control without considering the state constraint.

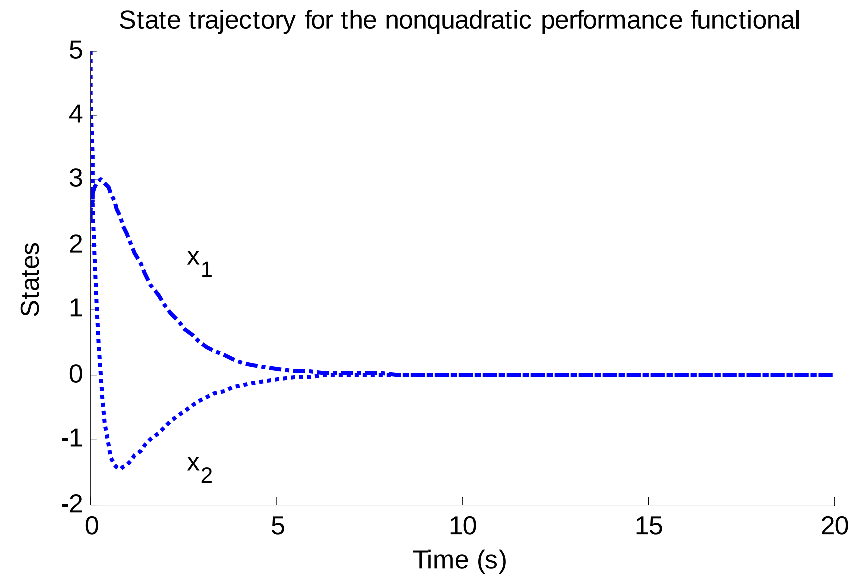
An initial stabilizing controller, the LQR $-2.4x_1 - 3.6x_2$, that violates the state constraints is shown in Figure 3.9. The performance of this controller is improved by stochastically sampling from the region $-3.5 \le x_1 \le 3.5, -5 \le x_2 \le 5$, where $p = 3000$, and running the policy iterations algorithm for 20 times.

It can be seen that the nearly optimal control law that considers the state constraint tends not to violate the state constraint as the LQR controller does. It is important to realize, that as the order k in the performance functional is increased, then one gets larger and larger control signals at the starting time of the control process to avoid violating the state constraints.

A smooth function of the order 45 that resembles the one used for the nonlinear oscillator in the previous example is used to approximate the value function of the system. The weights $\mathbf{w}_0$ are found by the policy iteration method. Since $R = 1$, the final control law becomes,

$$u(x) = -\frac{1}{2}\mathbf{w}_0{}' \frac{\partial V}{\partial x_2}.$$

It was noted that the nonquadratic performance functional returns an over all cost of 212.33 when the initial conditions are $x_1 = 2.4, x_2 = 5.0$ for the optimal controller, while this cost increases to 316.07 when the linear controller is used. It is this increase in cost detected by the nonquadratic performance functional that causes the system to avoid violating the state constraints. If this difference in costs is made bigger, then one actually increases the set of initial conditions that do not violate the constraint. This however, requires a larger neural network, and high precision computing machines.



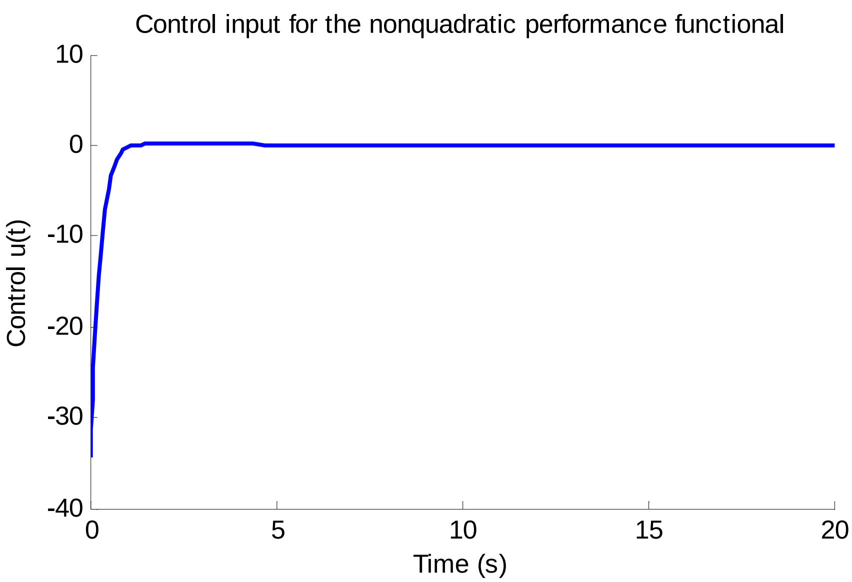


Figure 3.10 Nearly optimal nonlinear control law considering the state constraint

3.5.4 Minimum-Time Control

Consider the following system

$$\dot{x}_1 = x_2,$$
$$\dot{x}_2 = -x_2 + u.$$

It is desired to control the system with control limits of $|u| \le 1$ to drive it to origin in minimum time. Typically, from classical optimal control theory [43], one finds out that the control law required is a bang-bang controller that switches back and forth based on a switching surface that is calculated using Pontryagin's minimum principle. It follows that the minimum time control law for this system is given by

$$s(x) = x_1 - \frac{x_2}{|x_2|} \ln\left(|x_2| + 1\right) + x_2,$$
$$u^*(x) = \begin{cases} -1, & \text{for } x \text{ such that } s(x) > 0, \\ +1, & \text{for } x \text{ such that } s(x) < 0, \\ -1, & \text{for } x \text{ such that } s(x) = 0 \text{ and } x_2 < 0, \\ 0, & \text{for } x = 0. \end{cases}$$

The response to this controller is shown in Figure 3.11. It can be seen that this is a highly nonlinear control law that requires the calculation of a switching surface. This is however a formidable task even for linear systems with state dimension larger than 3. However, when using the method presented in this chapter, finding a nearly minimum-time controller becomes a less complicated matter.

The following nonquadratic performance functional is used

$$Q(x) = \tanh\left(x_1^2 / 0.1^2 + x_2^2 / 0.1^2\right), \quad W(u) = 0.001 \times 2 \int_0^u \tanh^{-1}(\mu) d\mu.$$

A smooth function of the order 35 is used to approximate the value function of the system. This neural network is solved for by stochastic sampling, Monte Carlo methods, [27]. Let $p = 5000$ for $-0.5 \le x_1 \le 0.5, -0.5 \le x_2 \le 0.5$.

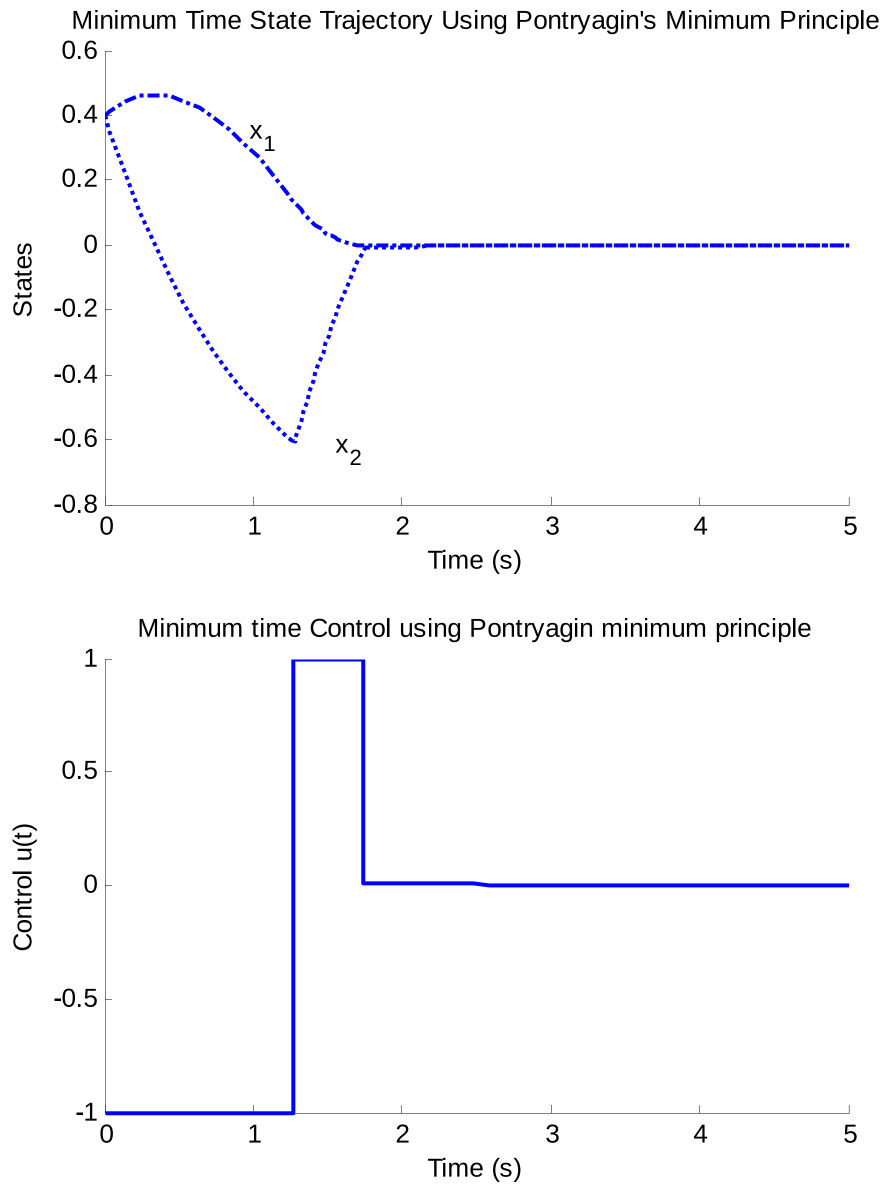


Figure 3.11 Performance of the exact minimum-time controller.

The weights $\mathbf{w}_o$ are found after iterating for 20 times. Since $R=1$, the final control law becomes

$$u(x) = -\tanh\left(\frac{1}{2}\mathbf{w}_0{}'\frac{\partial V}{\partial x_2}\right).$$

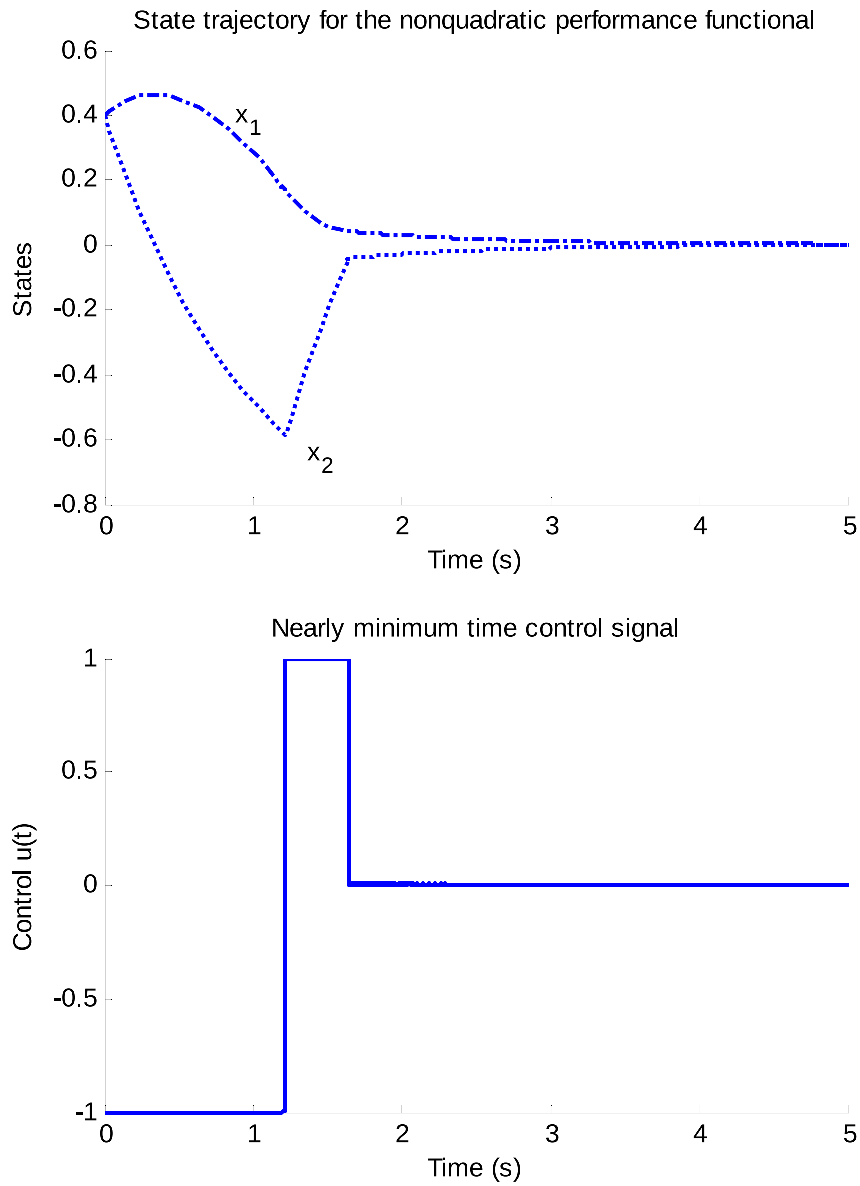


Figure 3.12 Performance of the nearly minimum-time controller

Figure 3.12 shows the performance of the controller obtained using the algorithm presented in this chapter and compares it with that of the exact minimum-time controller. Figure 3.13 plots the state trajectory of both controllers. Note that the nearly minimum-time controller behaves as a bang-bang controller until the states come close to the origin when it starts behaving as a regulator.

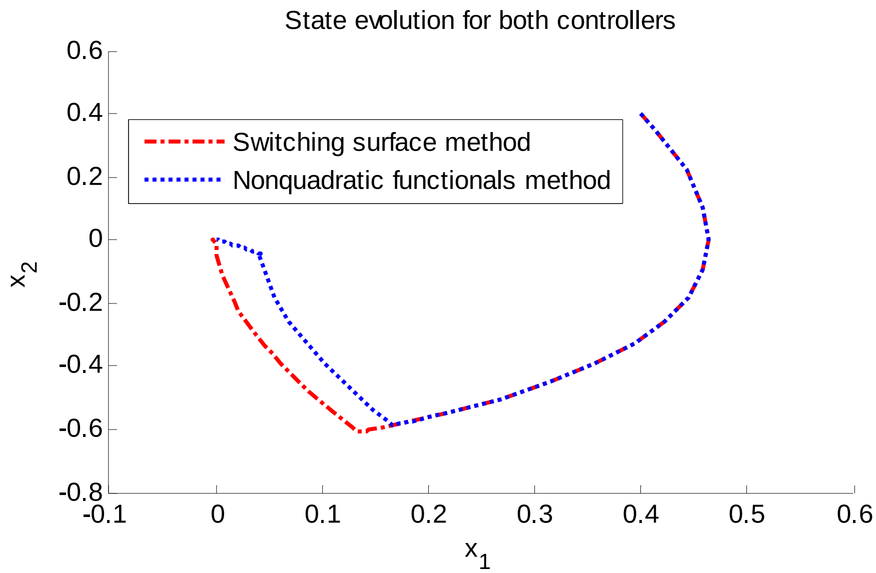


Figure 3.13 State evolution for both minimum-time controllers

3.6 Conclusions

A rigorous computationally effective algorithm to find nearly optimal controllers in state feedback form for general nonlinear systems with constraints is presented that is approaches the problem of constrained optimization from a practical engineering tractable point. The control is given as the output of a neural network. This is an extension of the novel work in [14], [56]. Conditions under which the theory of policy iterations, [72], applies were shown. Several numerical examples were discussed and simulated.

This algorithm requires further research into the problem of increasing the region of asymptotic stability. Moreover, adaptive control techniques can be blended to formulate and adaptive optimal controllers for general nonlinear systems with constraints and unknown system dynamics f, g.

CHAPTER 4

POLICY ITERATIONS AND THE HAMILTON-JACOBI-ISAACS EQUATION FOR H_∞ STATE FEEDBACK CONTROL WITH INPUT SATURATION

4.1 Introduction

In this chapter, the HJI equation for systems with input constraints is derived and then an algorithmic solution to solve the obtained HJI equation using policy iterations on the corresponding zero-sum game is developed. Although the formulation of the nonlinear theory of H_∞ control has been well developed, [76], [13], [79], [76], [39], [9], and [11], solving the corresponding HJI equation remains a challenge.

The H_∞ norm has played an important role in the study and analysis of robust optimal control theory since its original formulation in an input-output setting by Zames, [81]. Earlier solution techniques involved operator-theoretic methods. State space solutions were rigorously derived in [26] for the linear system case that required solving several associated Riccati equations. Later, more insight into the problem was given after the H_∞ linear control problem was posed as a zero-sum two-person differential game by Başar [13]. The nonlinear counterpart of the H_∞ control theory was developed by Van der Schaft [76]. He utilized the notion of dissipativity introduced by Willems [80], [79] and formulated the H_∞ control theory into a nonlinear L_2-gain optimal control problem. The L_2-gain optimal control problem requires solving a Hamilton-Jacobi equation, namely the Hamilton-Jacobi-Isaacs (HJI) equation.

Conditions for the existence of smooth solutions of the Hamilton-Jacobi equation were studied through invariant manifolds of Hamiltonian vector fields and the relation with the Hamiltonian matrices of the corresponding Riccati equation for the linearized problem, [76]. Later some of these conditions were relaxed by Isidori and Astolfi [39], into critical and noncritical cases.

The HJI equation is hard to solve directly. Several method based on policy iterations were proposed. In [76], it was proven that there exist a sequence of iterative policies to pursue the smooth solution of the HJI equation. Later Beard and McLain, [17], proposed, for the first time, to use policy iterations on the disturbance, if they exists, as well as policy iterations on the controller. However, the existence of such policies for the disturbance was not proven.

This chapter has three objectives. *First*, prove the existence of policy iterations on the disturbance input and converging to the available storage of the associated dissipative closed loop dynamics. Hence, this is a way to solve the HJB equation of the nonlinear bounded real lemma. *Second,* a formal solution is given to the suboptimal H_∞ control problem of dynamical systems with constraints on the input using a special quasi-norm to perform the L_2-gain analysis and derive the corresponding HJI equation. *Third,* policy iterations on both players are used to break the HJI of constrained controls into a sequence of linear partial differential equations. This is analogous to the work in chapter two and [1] where the second and third objectives have been applied to the HJB equation appearing in optimal control theory.

Remark 4.1: *Necessary conditions for the existence of smooth solutions of the HJI* *equation in the case of systems with no input constraints have been studied earlier by [39], [76]. Other lines of research study the nonsmooth solutions of the HJI equation using the theory of viscosity solutions, [11]. This notion of solutions was studied for the* H_∞ *control problem [9]. In this note, the proposed results are valid under regularity assumptions as done in [39], [76] and is justified by assumptions on the quasi-norm described later in the note. See [1] for the HJB case.*

4.2 Policy Iterations and the Nonlinear Bounded Real Lemma

Consider the system described by

$$\begin{aligned} \dot{x} &= f(x) + k(x)d \\ z &= h(x) \end{aligned} \tag{4.1}$$

where $f(0)=0$, $d(t)$ is considered a disturbance, and $z(t)$ is a fictitious output. $x=0$ is assumed to be an equilibrium point of the system. It is known that the system (4.1) has an L_2-gain $\le \gamma$, $\gamma \ge 0$, if

$$\int_0^T \|z(t)\|^2 \, dt \le \gamma^2 \int_0^T \|d(t)\|^2 \, dt \tag{4.2}$$

for all $T \ge 0$ and all $d \in L_2(0,T)$, with $x(0)=0$. Dynamical systems that are finite L_2-gain stable are said to be dissipative, [79].

Definition 4.1: *System (4.1) with supply rate* $w(t)$ *is said to be dissipative if there exists* $V \ge 0$*, called the storage function, such that*

$$V(x_0) + \int_{t_0}^{t_1} w(t)dt \ge V(x_1), \tag{4.3}$$

where $x_1 = \varphi(t_1, t_0, x_0, d)$.

If $x_0 = 0$ and $V \geq 0$ satisfying (4.3) exists such that $V(x_0) = 0$ and $w(t) = \gamma^2 \|d(t)\|^2 - \|z(t)\|^2$, then

$$\int_{t_0}^{t_1} w(t)dt \geq V(x_1) \geq 0 \Rightarrow \int_{t_0}^{t_1} \|z(t)\|^2 \, dt \leq \gamma^2 \int_{t_0}^{t_1} \|d(t)\|^2 \, dt \, .$$

It has been shown that a lower bound on the storage function is given by the so-called *available storage*. The existence of the available storage is essential in determining whether or not a system is dissipative.

Definition 4.2: *The available storage* V_a *of (4.1) is given by the following optimal control problem*

$$V_a(x) = \sup_{d(\cdot),\, T \geq 0} \int_0^T -w(d, z)dt \tag{4.4}$$

It was shown in [80][79] that for a system to be dissipative, the so-called *available storage* V_a needs to be finite. The available storage, $V_a \geq 0$, provides a lower bound on the storage function of the dynamical system, $0 \leq V_a \leq V$.

To find the available storage, one needs to solve an optimization problem which can be approached by solving a variational problem as in optimal control theory, [43][50]. The Hamiltonian of the optimization problem is given by,

$$H(x, p, d) = p'\left(f + kd\right) + h'h - \gamma^2 d'd \, . \tag{4.5}$$

The Hamiltonian is a polynomial of degree two in d, and has a unique maximum at

$$d^* = \tfrac{1}{2\gamma^2} k'(x) p$$

given by

$$H^*(x,p) = p' f(x) + \frac{1}{4\gamma^2} p' k'(x) k(x) p + h'(x) h(x) . \qquad (4.6)$$

Therefore, the value function of the optimization problem (4.4), the available storage, when smooth $V_a \geq 0 \in C^1$, is the stabilizing solution of the following Hamilton-Jacobi-Bellman equation

$$V_{a_x} f + \tfrac{1}{4\gamma^2} V_{a_x}{}' k k' V_{a_x} + h'h = 0, V_a(0) = 0 . \qquad (4.7)$$

The optimal policy is given by

$$d^* = \tfrac{1}{2\gamma^2} k'(x) V_{a_x}(x) \qquad (4.8)$$

which can be thought of as the policy for extracting the maximum energy from the system for a supply rate given by $w(t) = \gamma^2 \|d(t)\|^2 - \|z(t)\|^2$. It can be interpreted as the worst possible L_2 disturbance that can affect the system (4.1).

Definition 4.3: *Zero*-State *Observability: The nonlinear system is zero-state observable if* $y(t) = 0$ *and* $u(t) = 0$ *for all* $t \geq 0$ *implies that* $x(t) = 0$ *for all* $t \geq 0$.

It is assumed that system (4.1) is zero-state observable and hence $V_a > 0$ with a certain domain of validity as defined next, [20].

Definition 4.4: *The set* Ω *of all* x *satisfying (4.7) is said to be the domain of validity*

(DOV) of $V_a(x)$.

Lemma 4.1: $V(x)$*, the solution to (4.7) is positive definite whenever the system is zero-state observable. Moreover the free system* $\dot{x} = f(x)$ *is at least locally asymptotically stable. Global asymptotic stability follows if* $V(x)$ *is also a proper function, or radially unbounded.*

Proof: From (4.7), it follows that

$$\frac{dV}{dx} f(x) \leq -h(x)' h(x) .$$

Hence positive definiteness follows from zero-state observability as shown in Lemma 1 [34]. Since $V > 0$, asymptotic stability follows from LaSalle's invariance principle, and zero-state observability. ■

Lemma 4.2: *If the system dynamics*

$$\dot{x} = f + \frac{1}{2\gamma^2} kk^T \frac{dV}{dx}, \tag{4.9}$$

is asymptotically stable, where V *solves (4.7), then* L_2-gain<γ.

Proof: See [76], [45]. ■

Lemma 4.3: *If system (4.1) has* L_2-gain<γ*, then one has* $P(x)$ *such that*

$$P_x' f + h'h + \tfrac{1}{4\gamma^2} P_x' kk' P_x = Q(x) < 0 . \tag{4.10}$$

Proof: See [77]. ■

Lemma 4.4: *It can be also been shown that any $V(x) \geq 0$ that solves the following Hamilton-Jacobi inequality*

$$V_x{}' f + \tfrac{1}{4\gamma^2} V_x{}' k k' V_x + h'h \leq 0, V(0) = 0\,, \tag{4.11}$$

is a possible storage function.

Proof: See [77]. ■

Equation (4.7) is nonlinear in $V_a(x)$, therefore it is hard if not impossible to solve. In Theorem 4.1, policy iterations on d is used to break (4.7) into a sequence of equations that are linear in $V(x)$. This type of policy iterations, also known as Newton's method, has been used to solve

$$A'P + PA + \tfrac{1}{\gamma^2} PBB'P + C'C = 0 \tag{4.12}$$

appearing in the Bounded Real Lemma problem for linear systems. Existence of iterative policies to solve (4.12) appears in [46]. Theorem 4.1 generalizes this to (4.1).

Theorem 4.1: *Let $V^* > 0 \in C^1$ be the stabilizing of (4.7). Then one can solve for V^* by policy iterations starting with $d^0 = 0$, and solving for V^i*

$$V_x^{i\prime}(f + kd^i) + h'h - \gamma^2 \left\| d^i \right\|^2 = 0, \tag{4.13}$$

and updating the disturbance at each iteration according to

$$d^{i+1} = \tfrac{1}{2\gamma^2} k' V_x^i\,. \tag{4.14}$$

with $\dot{x} = f + kd^{i+1}$ asymptotically stable $\forall i$. Moreover,

$$i \to \infty \Rightarrow \sup_{x \in \Omega^*} \left| V^i - V^* \right| \to 0$$

with $0 < V^i(\Omega^i) \le V^{i+1}(\Omega^{i+1})$ *and* $\Omega^{i+1} \subseteq \Omega^i$.

Proof: *Existence:* Assume that there is d^i such that $\dot{x} = f + kd^i$ is asymptotically stable. Then since

$$\begin{aligned} V_x^{i\prime}(f + \tfrac{1}{2\gamma^2}kk'V_x^{i-1}) &= -h'h + \tfrac{1}{4\gamma^2}V_x^{i-1\prime}kk'V_x^{i-1}, \\ P_x'(f + \tfrac{1}{2\gamma^2}kk'V_x^{i-1}) &= -h'h + Q(x) + \tfrac{1}{4\gamma^2}V_x^{i-1\prime}kk'V_x^{i-1} - \tfrac{1}{4\gamma^2}(P_x - V_x^{i-1})'kk'(P_x - V_x^{i-1}), \end{aligned}$$

therefore $\Omega^i \subseteq \Omega^{i-1}$ and

$$(P_x - V_x^i)'(f + \tfrac{1}{2\gamma^2}kk'V_x^{i-1}) = Q(x) - \tfrac{1}{4\gamma^2}(P_x - V_x^{i-1})'kk'(P_x - V_x^{i-1}) < 0.$$

Since the vector field $\dot{x} = f + kd^i$ is asymptotically stable, this implies that. And one then has the following equations

$$\begin{aligned} V_x^{i\prime}(f + \tfrac{1}{2\gamma^2}kk'V_x^i) &= -h'h + \tfrac{1}{4\gamma^2}V_x^{i\prime}kk'V_x^i + \tfrac{1}{4\gamma^2}(V_x^i - V_x^{i-1})'kk'(V_x^i - V_x^{i-1}) \\ P_x'(f + \tfrac{1}{2\gamma^2}kk'V_x^i) &= -h'h + Q(x) + \tfrac{1}{4\gamma^2}V_x^{i\prime}kk'V_x^i - \tfrac{1}{4\gamma^2}(P_x - V_x^i)'kk'(P_x - V_x^i), \end{aligned}$$

then asymptotic stability of $\dot{x} = f + kd^{i+1}$ follows from

$$\begin{aligned} (P_x - V_x^i)'(f + \tfrac{1}{2\gamma^2}kk'V_x^i) &= Q(x) - \tfrac{1}{4\gamma^2}(P_x - V_x^i)'kk'(P_x - V_x^i) - \tfrac{1}{4\gamma^2}(V_x^i - V_x^{i-1})'kk'(V_x^i - V_x^{i-1}) \\ &< 0. \end{aligned}$$

Starting with $d^0 \equiv 0$, and by asymptotic stability of $\dot{x} = f$, the proof follows by induction.

Convergence: Since (d^i, V^i) exists and is asymptotically stable. Then, $\forall i, V^{i+1} \ge V^i$.

This is shown by integrating V^i and V^{i+1} over the state trajectory of $\dot{x} = f + kd^{i+1}$ for $x_0 \in \Omega^i \wedge \Omega^{i+1}$. Since

$$V_x^{i+1\prime}(f + kd^{i+1}) = -h'h + \gamma^2 \left\| d^{i+1} \right\|^2, \quad V_x^{i\prime} f = -V_x^{i\prime} kd^i - h'h + \gamma^2 \left\| d^i \right\|^2, \quad V_x^{i\prime} k = 2\gamma^2 d^{i+1}{}'.$$

Then it follows that

$$\begin{aligned} V^{i+1}(x_0) - V^i(x_0) &= -\int_0^\infty \left\{ \dot{V}^{i+1}(x_0) - \dot{V}^i(x_0) \right\} dt \\ &= \int_0^\infty \left\{ V_x^{i\prime}(f + kd^{i+1}) - V_x^{i+1\prime}(f + kd^{i+1}) \right\} dt. \\ &= \int_0^\infty \left\{ \gamma^2 \left\| d^i \right\|^2 + 2\gamma^2 d^{i+1\prime}(d^{i+1} - d^i) - \gamma^2 \left\| d^{i+1} \right\|^2 \right\} dt \\ &= \gamma^2 \int_0^\infty \left\{ \left\| d^{i+1} - d^i \right\|^2 \right\} dt \ge 0, \end{aligned}$$

and hence pointwise convergence to the solution of (4.7) follows. Since Ω^* is compact, uniform convergence of V^i to V^* on Ω^* follows from Dini's theorem, [6]. ■

Theorem 4.2: *If (4.1) satisfies (4.2) for $\gamma_2 \le \gamma_1$ and if*

$$\dot{x} = f + \tfrac{1}{2\gamma^2} kk'V_{x\gamma_1}^* \text{ and } \dot{x} = f + \tfrac{1}{2\gamma^2} kk'V_{x\gamma_2}^*$$

are asymptotically stable on Ω_{γ_1} and Ω_{γ_2}. Then $\Omega_{\gamma_2} \subseteq \Omega_{\gamma_1}$ and $V_{\gamma_2}^* \ge V_{\gamma_1}^*$.

Proof: Since for γ_2, the available storage $V_{\gamma_2}^*$ satisfies

$$V_{x\gamma_2}^*{}' f + \tfrac{1}{4\gamma_2^2} V_{x\gamma_2}^*{}' kk'V_{x\gamma_2}^* + h'h = 0 \Rightarrow V_{x\gamma_2}^*{}' f + \tfrac{1}{4\gamma_1^2} V_{x\gamma_2}^*{}' kk'V_{x\gamma_2}^* + h'h \le 0 .$$

$V_{\gamma_2}^*$ is a possible storage function with gain γ_1. Therefore, $V_{\gamma_1}^*$ is valid on Ω_{γ_2} and

$\Omega_{\gamma_2} \subseteq \Omega_{\gamma_1}$. Integrating over the trajectory of the system $\dot{x} = f + kd^*_{\gamma_1}$ it follows that

$$V^*_{\gamma_2}(x_0) - V^*_{\gamma_1}(x_0) = \int_0^\infty \left\{ \dot{V}^*_{\gamma_1}(x_0, d^*_{\gamma_1}) - \dot{V}^*_{\gamma_2}(x_0, d^*_{\gamma_1}) \right\} dt \geq \int_0^\infty \left\{ \gamma_2^2 \left\| d^*_{\gamma_2} - d^*_{\gamma_1} \right\|^2 \right\} dt \geq 0$$

and this completes the proof. ■

4.3 L_2-gain of Nonlinear Control Systems with Input Saturation

Consider the following nonlinear system

$$\Sigma : \begin{Bmatrix} \dot{x} = f(x) + g(x)u + k(x)d, \\ \|z\|^2 = \|h\|^2 + \|u\|^2, \end{Bmatrix} \tag{4.15}$$

where $x \in \mathbb{R}^n, u \in \mathbb{R}^m, d \in \mathbb{R}^q$, $f(0) = 0$, $x = 0$ is an equilibrium point of the system, $z(t)$ a fictitious output, $d(t) \in L_2[0,\infty)$ is the disturbance, and $u(t) \in U$ is the control with U defined as

$$U = \{ u(t) \in L_2[0,\infty) \mid -\alpha_i \leq u_i \leq \alpha_i, i = 1,\ldots,m \}.$$

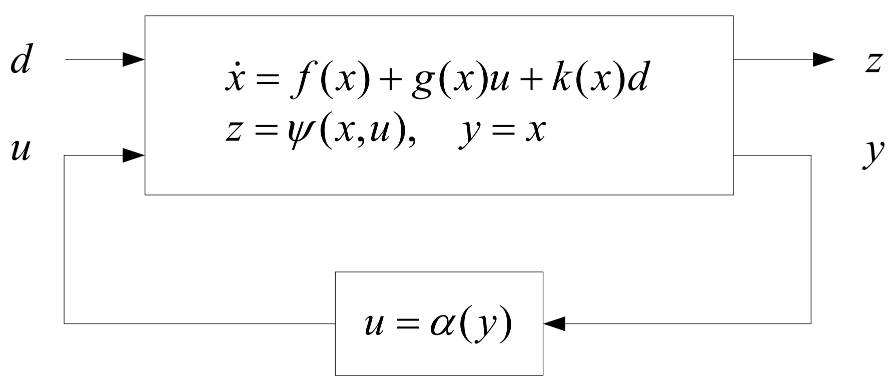


Figure 4.1 State feedback nonlinear H_∞ controller.

In the L_2-gain problem, one is interested in u which for some prescribed γ

renders

$$V(x_0) = \int_0^\infty \left(\overbrace{h'h + \|u\|^2}^{\|z(t)\|^2} - \gamma^2 \|d\|^2 \right) dt , \tag{4.16}$$

nonpositive for all $d(t) \in L_2(0,\infty)$ and $x(0) = 0$. In other words

$$\int_0^\infty \|z(t)\|^2 \, dt \le \gamma^2 \int_0^\infty \|d(t)\|^2 \, dt . \tag{4.17}$$

It is well known, [13], that L_2-gain problem is equivalent to the solvability of the zero-sum game

$$V^*(x_0) = \min_{u \in U} \max_d \int_0^\infty \left(h'h + \|u(t)\|^2 - \gamma^2 \|d\|^2 \right) dt . \tag{4.18}$$

The Hamiltonian of the previous zero-sum game is

$$H(x,p,u,d) = p'(f + gu + kd) + h'h + \|u\|^2 - \gamma^2 \|d\|^2 . \tag{4.19}$$

Finding the stationarity conditions of this Hamiltonian requires solving for

$$\min_{u \in U} \max_d H(x,p,u,d) \text{ and } \max_d \min_{u \in U} H(x,p,u,d) \tag{4.20}$$

which is a constrained optimization with respect to the control policy, $u \in U$.

To confront this constrained optimization problem difficulty of the Hamiltonian, a quasi-norm is used to transform the constrained optimization problem (4.18) into

$$V^*(x_0) = \min_u \max_d \int_0^\infty \left(h'h + \|u\|_q^2 - \gamma^2 \|d\|^2 \right) dt . \tag{4.21}$$

Definition 4.5: *A quasi-norm, $\|\cdot\|_q$, on a vector space X, has the following properties*

$$\|x\|_q = 0 \Leftrightarrow x = 0,\ \|x+y\|_q \le \|x\|_q + \|y\|_q,\ \|x\|_q = \|-x\|_q .$$

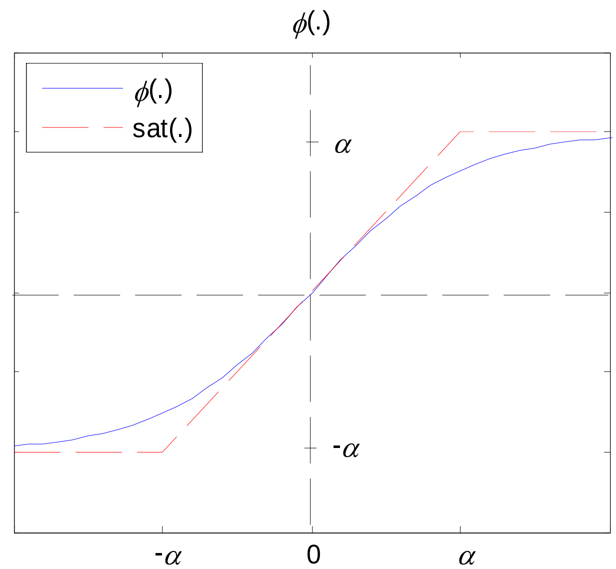


Figure 4.2 Approximation of control saturation.

This definition is weaker than the definition of a norm, in which the third property is replaced by homogeneity, $\|\alpha x\|_q = |\alpha|\|x\|_q\ \forall \alpha \in \Re$, [6]. A suitable quasi-norm to confront control saturation is

$$\|u\|_q^2 = 2\int_0^u \phi^{-1}(v)dv = \sum_{k=1}^m 2\int_0^{u_k} \phi^{-1}(v)dv, \tag{4.22}$$

where $\|u\|_q \in C^1$ one to one, and ϕ^{-1} is assumed to be monotonically increasing, i.e. $\phi(\cdot) = \tanh(\cdot)$ for $|u| \le 1$. Hence $\|u(t)\|_q^2 \simeq \|u(t)\|^2$ and is locally quadratic in u.

The Hamiltonian of this modified zero-sum game, (4.21), is

$$H(x,p,u,d) = p'(f + gu + kd) + h'h + 2\int_0^u \phi^{-1}(v)dv - \gamma^2 \|d\|^2 . \qquad (4.23)$$

In this case finding the stationarity conditions of this Hamiltonian requires solving for

$$\min_u \max_d H(x,p,u,d) \text{ and } \max_d \min_u H(x,p,u,d) \qquad (4.24)$$

where the minimization of the Hamiltonian with respect to u is unconstrained. See [1], [54], and Chapter two for a similar work done in the framework of HJB equations.

The next Lemma shows a property that is satisfied by the quasi-normn this work.

Lemma 4.5: *If ϕ^{-1} is monotonically increasing, then*

$$\int_b^a \phi^{-1}(v)dv - \phi^{-1}(b)'(a-b) > 0, \quad \forall\ a \neq b . \qquad \blacksquare$$

4.4 The HJI Equation and the Saddle Point

To study the HJI equation corresponding to (4.21), the finite-gorizon game is first studied. Under feedback strategy information structure for both players, [13]. It is shown that Isaacs condition is satisfied and there is a unique saddle point solving the finite-horizon zero-sum game

$$V^*(x_0,T) = \min_u \max_d \int_0^T \left(h'h + 2\int_0^u \phi^{-1}(v)dv - \gamma^2 \|d\|^2 \right) dt . \qquad (4.25)$$

The Hamiltonian of the game (4.25) is

$$H(x,p,u,d) = p'(f + gu + kd) + h'h + 2\int_0^u \phi^{-1}(v)dv - \gamma^2 \|d\|^2. \qquad (4.26)$$

Lemma 4.6: Isaacs condition: $\min_u \max_d H = \max_d \min_u H$.

Proof: Applying the stationarity conditions $\partial H/\partial u = 0$ and $\partial H/\partial d = 0$ to (4.26) gives

$$u^*(x) = -\phi(\tfrac{1}{2} g(x)' p),\ d^*(x) = \tfrac{1}{2\gamma^2} k(x)' p\ . \qquad (4.27)$$

$$H(x,p,u^*,d^*) = p'f - 2\phi^{-1}(u^*)'u^* + h'h + 2\int_0^{u^*} \phi^{-1}(v)dv + \tfrac{1}{4\gamma^2} p'kk'p. \qquad (4.28)$$

Rewriting (4.26) in terms of (4.28) gives

$$H(x,p,u,d) = H(x,p,u^*,d^*) - \gamma^2 \|d - d^*\|^2 + 2\left\{ \int_{u^*}^{u} \phi^{-1}(v)dv - \phi^{-1}(u^*)'(u - u^*) \right\}^2 .$$

From Lemma 4.5, one has

$$H(x_0,u^*,d) \le H(x_0,u^*,d^*) \le H(x_0,u,d^*) \qquad (4.29)$$

and Isaacs condition follows. ■

The Hamilton-Jacobi-Isaacs equation, HJI, corresponding to (4.25) is

$$\begin{aligned}
-\frac{\partial V(t;x)}{\partial t} &= \min_u \max_d H(x, \frac{\partial V(t;x)}{\partial x}, u, d) \\
&= \max_d \min_u H(x, \frac{\partial V(t;x)}{\partial x}, u, d) \\
&= \frac{\partial V(t;x)}{\partial x}\left(f(x) + g(x)u^* + k(x)d^* \right) \\
V(T;x) &= 0.
\end{aligned} \qquad (4.30)$$

Under regularity assumptions, from Theorem 2.6 [13], if there exists $V^*(x_0) \in C^1$ solving the HJI (4.30), then

$$V(x_0, u^*, d) \le V(x_0, u^*, d^*) \le V(x_0, u, d^*) \tag{4.31}$$

and the zero-sum game has a value and the pair of policies (4.27) are in saddle point equilibrium.

The zero-sum game (4.21) is an infinite-horizon zero-sum game. Therefore, it is important to see the behavior of the finite-horizon game (4.25) as $T \to \infty$. It is seens that as $T \to \infty$ in (4.25), one obtains the following Isaacs equation

$$H^*(x, p, u^*, d^*) = V_x'(f + gu^* + kd^*) + h'h + 2\int_0^{u^*} \phi^{-1}(v)dv - \gamma^2 \left\|d^*\right\|^2 = 0 \,. \tag{4.32}$$

On substitution of (4.27) in (4.32), the HJI equation is obtained

$$V_x' f - V_x' g\phi\left(\tfrac{1}{2} g'V_x\right) + h'h + 2\int_0^{-\phi\left(\frac{1}{2}g'V_x\right)} \phi^{-1}(v)dv + \tfrac{1}{4\gamma^2} V_x' kk'V_x = 0, \quad V(0) = 0 \tag{4.33}$$

and hence the game has a value.

Next, it is shown that (4.27) remains in saddle point equilibrium as $T \to \infty$ if they are sought among finite energy strategies. See [12] for unconstrained policies.

Theorem 4.3: *Suppose that there exists a $V(x) \in C^1$ satisfying the HJI equation (4.33) and that*

$$\dot{x} = f - g\phi\left(\tfrac{1}{2} g'V_x\right) + \tfrac{1}{2\gamma^2} kk'V_x \tag{4.34}$$

is asymptotically stable, then

$$u^*(x) = -\phi(\tfrac{1}{2} g' V_x), \quad d^*(x) = \tfrac{1}{2\gamma^2} k' V_x \tag{4.35}$$

are in saddle point equilibrium for the infinite horizon game among strategies $u \in U, d \in L_2[0,\infty)$.

Proof: The proof is made by completing the squares,

$$J_T(u,d;x_0) = \int_0^T \left(h'h + \|u(t)\|_q^2 - \gamma^2 \|d\|^2 \right) dt + V^*(x_0) - V^*(x_T) + \int_0^T \dot{V}^* dt, \tag{4.36}$$

where V^* solves (4.33). This becomes

$$\begin{aligned}
J_T(u,d;x_0) &= \int_0^T \left(h'h + \|u(t)\|_q^2 - \gamma^2 \|d\|^2 \right) dt \\
&= \int_0^T \left(h'h + \|u(t)\|_q^2 - \gamma^2 \|d\|^2 \right) dt + V^*(x_0) - V^*(x_T) + \int_0^T V_x^{*\prime}(f + gu + kd) dt \\
&= \int_0^T \left(h'h + \|u(t)\|_q^2 - \gamma^2 \|d\|^2 + V_x^{*\prime}(f + gu + kd) \right) dt + \\
&\quad V^*(x_0) - V^*(x_T) \\
&= \int_0^T \left(2\int_{u^*}^{u} \phi^{-1}(v) dv - 2\phi^{-1}(u^*)'(u - u^*) - \gamma^2 \left\| d - d^* \right\|^2 \right) dt + \\
&\quad V^*(x_0) - V^*(x_T).
\end{aligned}$$

Since $u(t), d(t) \in L_2[0,\infty)$, and since the game has a finite value as $T \to \infty$, this implies that $x(t) \in L_2[0,\infty)$, therefore $x(t) \to 0$, $V^*(x(\infty)) = 0$ and

$$J_\infty(u,d;x_0) = V^*(x_0) + \int_0^\infty \left(2\int_{u^*}^{u} \phi^{-1}(v) dv - 2\phi^{-1}(u^*)'(u - u^*) - \gamma^2 \left\| d - d^* \right\|^2 \right) dt. \tag{4.37}$$

Hence u^*, d^* are in saddle point equilibrium in the class of finite energy strategies. ■

Since (4.35) satisfies the Isaacs equation, it can be shown that the feedback saddle point is unique in the sense that it is strongly time consistent and noise insensitive [12].

It is important to see how the solution of the infinite-horizon zero-sum game with the quasi-norm relates to the original constrained input L_2-gain control problem. To see this, note that substituting u^* in (4.37), one has

$$
\begin{aligned}
&V_x'(f+gu^*+kd)+h'h+2\int_0^{u^*}\phi^{-1}(v)dv-\gamma^2\|d\|^2=-\gamma^2\|d-d^*\|^2\\
&\dot{V}(u^*,d)+h'h+2\int_0^{u^*}\phi^{-1}(v)dv-\gamma^2\|d\|^2\le 0\\
&\int_0^T\left[\dot{V}(u^*,d)+h'h+2\int_0^{u^*}\phi^{-1}(v)dv-\gamma^2\|d\|^2\right]dt\le 0.
\end{aligned}
\tag{4.38}
$$

Integrating both sides, one has

$$
\begin{aligned}
&\int_0^T\left[\dot{V}(u^*,d)+h'h+2\int_0^{u^*}\phi^{-1}(v)dv-\gamma^2\|d\|^2\right]dt\le 0\\
&V(x(T))-V(x(0))+\int_0^T\left[h'h+2\int_0^{u^*}\phi^{-1}(v)dv\right]dt\le\gamma^2\int_0^T\|d\|^2dt
\end{aligned}
\tag{4.39}
$$

If the closed loop system is asymptotically stable and $d(\cdot)\in L_2[0,\infty)$, then

$$
h^Th+2\int_0^{u^*}\phi^{-T}(v)\in L_2[0,\infty).
$$

Thus (4.40) follows from $x(0)=0$ and $\lim_{T\to\infty}x(T)=0$

$$\int_0^\infty \left(h'h + 2\int_0^{u^*} \phi^{-1}(v)dv \right) dt \le \gamma^2 \int_0^\infty \|d\|^2 dt . \tag{4.40}$$

4.5 Solving the HJI Using Policy Iterations

To solve (4.33) by policy iterations, one starts by showing the existence and convergence of policy iterations on the constrained input as in [76] for systems with no input constraints. Then policy iterations on both players as proposed in [17], are performed on the constrained controller and d .

Theorem 4.4*: Assume that the closed-loop dynamics for the constrained stabilizing controller* u_j*,*

$$\dot{x} = f(x) + g(x)u_j + k(x)d \equiv f_j(x) + k(x)d .$$

satisfy all assumptions of Theorem 2.2. If the constrained controller is updated according to,

$$u_{j+1} = -\phi\left(\tfrac{1}{2} g' V_{x\,j}\right), \tag{4.41}$$

where V_j *is the available storage that solves*

$$V_{x\,j}{}' f_j + h'h + 2\int_0^{u_j} \phi^{-1}(v)dv + \tfrac{1}{4\gamma^2} V_{x\,j}{}' kk' V_{x\,j} = 0 . \tag{4.42}$$

Then $\dot{x} = f_{j+1} + kd$ *remains dissipative with respect to* $d(t)$ *for the same* γ*. Moreover,*

$$j \to \infty \Rightarrow \sup_{x\in\Omega_0} \left| V_j - V^* \right| \to 0$$

with $V_{j+1} \le V_j$ *with* V_{j+1} *valid on* Ω_0*, and* V^* *is the stabilizing solution of (4.33).*

Proof: To show the first part,

$$
\begin{aligned}
V_{xj}{}' f_{j+1} &= V_{xj}{}' f + V_{xj}{}' g u_{j+1} \\
&= V_{xj}{}' f + V_{xj}{}' g u_j + V_{xj}{}' g(u_{j+1} - u_j) \\
&= -h'h - \tfrac{1}{4\gamma^2} V_{xj}{}' kk' V_{xj} - 2\int_0^{u_j} \phi^{-1}(v)dv - 2\phi^{-1}(u_{j+1})'(u_{j+1} - u_j) \\
&= -h'h - 2\int_0^{u_{j+1}} \phi^{-1}(v)dv - \tfrac{1}{4\gamma^2} V_{xj}{}' kk' V_{xj} + 2\int_{u_j}^{u_{j+1}} \phi^{-1}(v)dv - 2\phi^{-1}(u_{j+1})'(u_{j+1} - u_j).
\end{aligned}
$$

From Lemma 4.5, one has the following HJ inequality,

$$
V_{xj}{}' f_{j+1} + h'h + 2\int_0^{u_{j+1}} \phi^{-1}(v)dv + \tfrac{1}{4\gamma^2} V_{xj}{}' kk' V_{xj} \le 0 .
$$

From Lemma 4.4, this means that V_j is a possible storage for $\dot{x} = f_{j+1}$. Hence one has

$$
V_{x\,j+1}{}' f_{j+1} + h'h + 2\int_0^{u_{j+1}} \phi^{-1}(v)dv + \tfrac{1}{4\gamma^2} V_{x\,j+1}{}' kk' V_{x\,j+1} = 0
$$

where $V_{j+1} \le V_j$ and V_{j+1} valid on Ω_j and hence valid on Ω_0. V_j converges pointwise to V^* follows, and since Ω_* is compact, uniform convergence of V_j to V^* on Ω_* follows by Dini's theorem, [6]. ■

Corollary 4.1: *The available storage V^* of u^*, (4.35), has the largest DOV of any other constrained controller guaranteeing (4.17) a prescribed γ.*

Proof: The proof follows immediately from Theorem 4.4 since V^* is valid for any Ω_0, the DOV of the available storage of any u guaranteeing (4.17). ■

This implies that u^* has the largest DOV within which L_2-performance for a given γ is guaranteed.

Policy iterations in Theorem 4.4 and Theorem 4.1 can be combined together to provide a two loop policy iterations solution method for the HJI equation. Specifically, select u_j, and find V_j that solves (4.42) by inner loop policy iterations on

$$V_{x\,j}^{i}{}'(f_j + kd^i) + h'h + 2\int_0^{u_j} \phi^{-1}(v)dv - \gamma^2 \left\|d^i\right\|^2 = 0\,. \qquad (4.43)$$

and the disturbance as in Theorem 4.1 until $V_j^\infty \to V_j$. Then by Theorem 4.4, use (4.41) in an outer loop policy iteration on the constrained control.

Equation (4.43) is denoted as $PI(V_j^i, u_j, d^i) = 0$, where PI stands for Policy Iteration. It becomes equivalent to the *LE* equation in Chapter 2 when $\gamma = \infty$.

Controllers derived using (4.33) for a fixed γ are *suboptimal* H_∞ controllers. *Optimal* H_∞ are achieved for the lowest possible γ^* for which the HJI is solvable. The next theorem demonstrates what happens to the DOV of the value of the game as γ decreases.

Theorem 4.5: *If $\gamma_1 \ge \gamma_2 > \gamma^*$, then $\Omega_{\gamma_1}^* \supseteq \Omega_{\gamma_2}^*$ where $\Omega_{\gamma_1}^*$ and $\Omega_{\gamma_2}^*$ denotes the DOV of the available storage functions $V_{\gamma_1}^*$ and $V_{\gamma_2}^*$ solving (4.33) for γ_1 and γ_2 respectively with γ^* being the smallest gain for which a stabilizing solution of the HJI exists.*

Proof: Follows from Theorem 4.4, and Corollary 4.1. ■

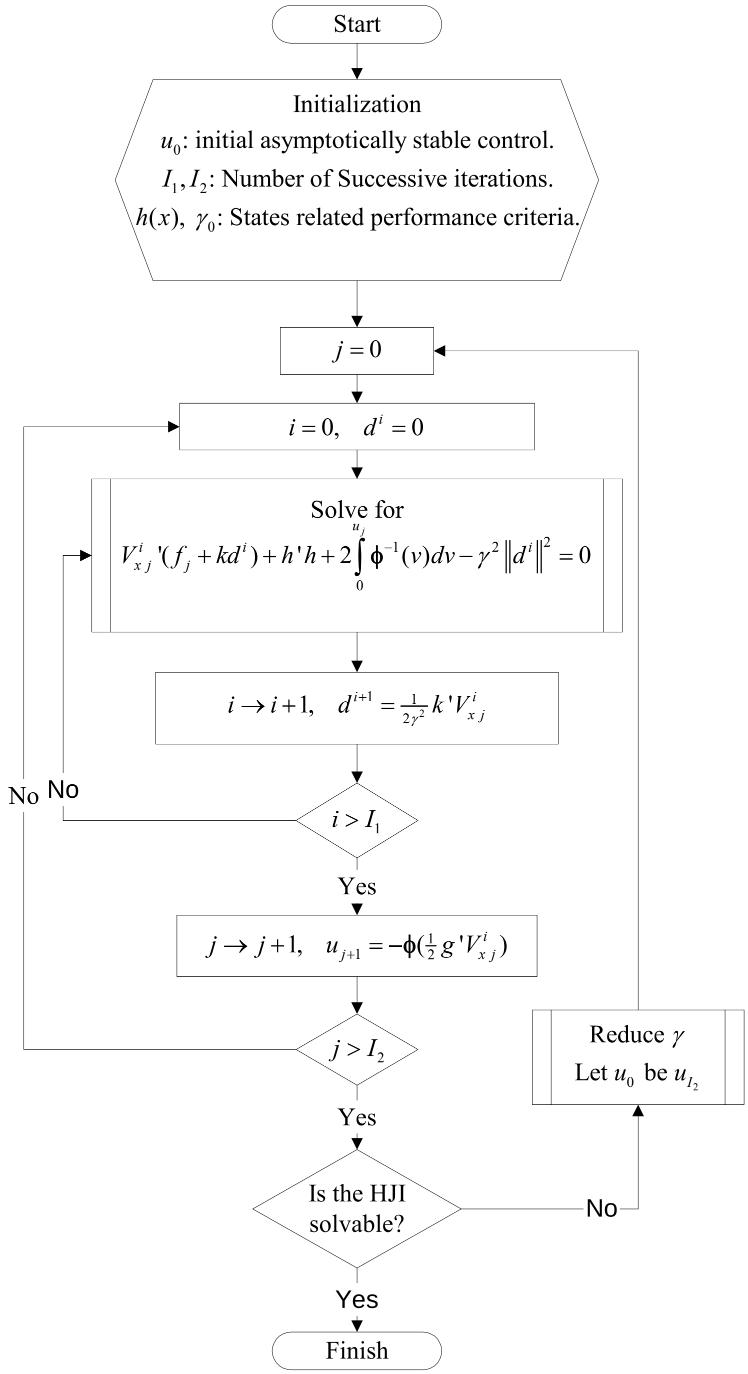


Figure 4.3 Policy iterations to solve the constrained input HJI

This implies that once the HJI is solved for a particular attenuation, γ_1, one can use the converged control policy as an initial stabilizing solution to try and solve for the HJI with a smaller attenuation γ_2. This is summarized in Figure 4.3.

Remark 4.2: *It maybe possible that the DOV of the HJI shrinks to null as one approches γ^*. See [77] for unconstrained control cases.*

4.6 Conclusions

The constrained input HJI equation along with two players policy iterations provide a sequence of differential equations for which approximate closed-form solutions are easier to obtain. This is an extension to the novel work of Beard and McLain [17], Lyshevski [54], and to our earlier work on HJB equations [1].

In the next Chapter, it is shown how to use neural networks to obtain least squares solution of the HJI equation. It is demonstrated how to approximately solve for V_j^i in $PI(V_j^i, u_j, d^i) = 0$ at each iteration on i and j. Therefore, one obtains a practical method to derive L_2-gain optimal, or suboptimal H_∞, controllers of nonlinear systems affine in input and experiencing actuator saturation.

CHAPTER 5

NEARLY H_∞ OPTIMAL NEURAL NETWORK CONTROL FOR CONTRAINED INPUT SYSTEMS

In our earlier work presented in the fourth chapter of this dissertation and appearing in [2], the zero-sum game for L_2-gain optimal control, suboptimal H_∞ control, of affine in input nonlinear systems with control constraints was treated. Moreover, the Hamilton-Jacobi-Isaacs (HJI) equation using performance functionals with quasi-norms to encode input constraints was derived. As for unconstrained inputs [76], once the game value function of the HJI equation is smooth and computed, a feedback controller can be synthesized that results in closed-loop asymptotic stability and provides L_2-gain disturbance attenuation. However, computing the value of the game is a formidable task when solutions of the HJI are approached directly.

For unconstrained affine in input nonlinear systems, a direct approach to solve the HJI equation is given by the third coauthor, [38], where the assumed smooth solution is found by solving for the Taylor series expansion coefficients in a very efficient and organized manner. In [17], an indirect method to solve the HJI equation for unconstrained systems based on policy iterations is proposed where the solution of a sequence of differential equations, linear in the associated cost, converges to the solution of the related HJI equation which is nonlinear in the available storage. Galerkin techniques are used to solve the sequence of linear differential equations, resulting in a numerically efficient algorithm that, however, requires computing numerous integrals over a well-defined region of the state space.

In [2], policy iterations were proposed to solve the constrained-input HJI equation. In this chapter, one builds on the results in [2] by using neural networks to solve for the sequence of linear differential equations in a least-squares sense on a prescribed compact set of the state-space. This is an extension to our earlier neural network policy iteration approach to solve the constrained-input HJB equation [1].

The importance of this chapter stems from the fact that a practical solution method based on neural networks to solve for suboptimal H_∞ control of constrained input systems is provided. The remainder of this chapter is organized as follows. In Section 5.1 appear the novel results of this chapter where a neural network least-squares based algorithm is described to practically solve for the constrained-input HJI equation. Section 5.2 demonstrates the stability and convergence of the proposed neural network algorithm. Section 5.3 illustrates a successful application of the proposed algorithm to the Rotational/Translational Actuator (RTAC) nonlinear benchmark problem under actuator saturation originally proposed in [22]. Conclusions are given in section 5.4.

In the next section, it is shown how to approximate V_j^i in $PI(V_j^i, u_j, d^i) = 0$ at each iteration on i and j using neural networks.

5.1 Neural Network Representation of Policies

Although equation

$$V_{x\,j}^{i}\,'(f_j + kd^i) + h'h + 2\int_0^{u_j} \phi^{-1}(v)dv - \gamma^2 \left\| d^i \right\|^2 = 0 \tag{5.1}$$

is in principle easier to solve for V_j^i than solving the HJI (4.33) directly, it remains difficult to get an exact closed-form solution for V_j^i at each iteration. Therefore, one seeks to approximately solve for V_j^i at each iteration. In this section, a computationally practical neural network based algorithm is presented that solves for V_j^i on a compact set domain of the state space in a least-squares sense. Proofs of convergence and stability of the neural network policies are discusses in Section IV.

It is well known that neural networks can be used to approximate smooth functions on prescribed compact sets [49]. Therefore, V_j^i is approximated at each inner loop iteration i over a prescribed region of the state-space with a neural net,

$$\hat{V}_j^i(x) = \sum_{k=1}^{L} w_{j,k}^i \sigma_k(x) = \mathbf{w}_j^{i\prime} \boldsymbol{\sigma}_L(x), \tag{5.2}$$

where the activation functions $\sigma_j(x) : \Omega \to \Re$, are continuous, $\sigma_j(0) = 0$, span $\{\sigma_j\}_1^\infty \subseteq L_2(\Omega)$. The neural network weights are w_k and L is the number of hidden-layer neurons. Vectors $\boldsymbol{\sigma}_L(x) \equiv [\sigma_1(x)\, \sigma_2(x) \cdots \sigma_L(x)]'$, $\mathbf{w} \equiv [w_1\, w_2 \cdots w_L]'$ are the vector activation function and the vector weight respectively. The neural network weights are tuned to minimize the residual error in a least-squares sense over a set of points within the stability region Ω of the initial stabilizing control. The least-squares solution attains the lowest possible residual error with respect to the neural network weights.

Replacing V_j^i in $PI(V_j^i, u_j, d^i) = 0$ with $\hat{V}_j^i$, one has

$$PI\left(\hat{V}_j^i(x)=\sum_{k=1}^{L} w_k\sigma_k(x), u_j, d^i\right)=e_L(x), \tag{5.3}$$

where $e_L(x)$ is the residual error.

To find the least-squares solution, the method of weighted residuals is used [28]. The weights, $\mathbf{w}_j^i$, are determined by projecting the residual error onto $de_L(x)\big/d\mathbf{w}_j^i$ and setting the result to zero $\forall x\in\Omega$ using the inner product, i.e.

$$\left\langle \frac{de_L(x)}{d\mathbf{w}_j^i}, e_L(x)\right\rangle = 0, \tag{5.4}$$

where $\langle \mathrm{f,g}\rangle = \int_\Omega fgdx$ is a Lebesgue integral. Rearranging the resulting terms, one has

$$\begin{gathered}\mathbf{w}_j^i = -\left\langle \nabla\boldsymbol{\sigma}_L F_j^i, \nabla\boldsymbol{\sigma}_L F_j^i\right\rangle^{-1} \bullet \left\langle H_j^i, \nabla\boldsymbol{\sigma}_L F_j^i\right\rangle, \\ F_j^i = f + gu_j + kd^i, \quad H_j^i = h'h + 2\int_0^{u_j} \phi^{-1}(v)dv - \gamma^2 \left\| d^i \right\|^2 .\end{gathered} \tag{5.5}$$

Equation (5.5) involves a matrix inversion. The following lemma discusses the invertibility of this matrix.

Lemma 5.1: *If the set* $\{\sigma_j\}_1^L$ *is linearly independent, then*

$$\left\{\nabla\sigma_j{}' F_j^i\right\}_1^L$$

is also linearly independent.

Proof: This follows from the asymptotic stability of the vector field $\dot{x} = F_j^i$ shown in [2], and from [1]. ■

Because of Lemma 1, the term $\left\langle \nabla\boldsymbol{\sigma}_L F_j^i, \nabla\boldsymbol{\sigma}_L F_j^i \right\rangle$ is guaranteed to have a full rank, and thus is invertible, as long as $\dot{x} = F_j^i$ is asymptotically stable. This in turn guarantees a unique, $\mathbf{w}_j^i$, of (5.5).

Having solved for the neural net weights, the disturbance policy is updated as

$$\hat{d}^{i+1} = \tfrac{1}{2\gamma^2} k' \nabla\boldsymbol{\sigma}_L{}' \mathbf{w}_j^i . \tag{5.6}$$

It is important that the new dynamics $\dot{x} = f + gu_j + k\hat{d}^{i+1}$ to be asymptotically stable in order to be able to solve for $\mathbf{w}_j^{i+1}$ in (5.5). Theorem 1 in the next section discusses the asymptotic stability of $\dot{x} = f + gu_j + k\hat{d}^{i+1}$.

Policy iterations on the disturbance requires solving iteratively between equations (5.5) and (3.7) at each inner loop iterations on i until the sequence of neural network weights, $\mathbf{w}_j^i$, converges to some value denoted by $\mathbf{w}_j^*$. Then the control is updated using $\mathbf{w}_j^*$ as

$$\hat{u}_{j+1} = -\phi(\tfrac{1}{2} g' \nabla\boldsymbol{\sigma}_L{}' \mathbf{w}_j^*) \tag{5.7}$$

in the outer-loop iteration on j.

Finally, one can approximate the integrals needed to solve (5.5) by introducing a mesh on Ω with mesh size equal to Δx. Equation (5.5) becomes

$$X_j^i = \left\lfloor \nabla\boldsymbol{\sigma}_L F_j^i\Big|_{x_1} \cdots\cdots \nabla\boldsymbol{\sigma}_L F_j^i\Big|_{x_p} \right\rfloor', \quad Y_j^i = \left\lfloor H_j^i\Big|_{x_1} \cdots\cdots H_j^i\Big|_{x_p} \right\rfloor' \tag{5.8}$$

where p in x_p represents the number of points of the mesh and H and F are as shown in (5.5). The number p increases as the mesh size is reduced. Therefore

$$\begin{aligned}\left\langle \nabla\boldsymbol{\sigma}_L F_j^i, \nabla\boldsymbol{\sigma}_L F_j^i \right\rangle &= \lim_{\|\Delta x\|\to 0} (X_j^{i\prime} X_j^i)\cdot\Delta x \\ \left\langle H_j^i, \nabla\boldsymbol{\sigma}_L F_j^i \right\rangle &= \lim_{\|\Delta x\|\to 0} (X_j^{i\prime} Y_j^i)\cdot\Delta x\end{aligned} \tag{5.9}$$

This implies that one can calculate $\mathbf{w}_j^i$ as

$$\mathbf{w}_j^i = -(X_j^{i\prime} X_j^i)^{-1}(X_j^{i\prime} Y_j^i). \tag{5.10}$$

An interesting observation is that equation (5.10) is the standard least-squares method of estimation for a mesh on Ω. Note that the mesh size Δ should be such that the number of points p is greater than or equal to the order of approximation L. This guarantees a full rank for $(X_j^{i\prime} X_j^i)$.

There do exist various ways to efficiently approximate integrals as those appearing in (5.5). Monte Carlo integration techniques can be used. Here the mesh points are sampled stochastically instead of being selected in a deterministic fashion, [27]. In any case however, the numerical algorithm at the end requires solving (5.10) which is a least-squares computation of the neural network weights. Numerically stable routines to compute equations like (5.10) do exists in several software packages like MATLAB which is used the next section.

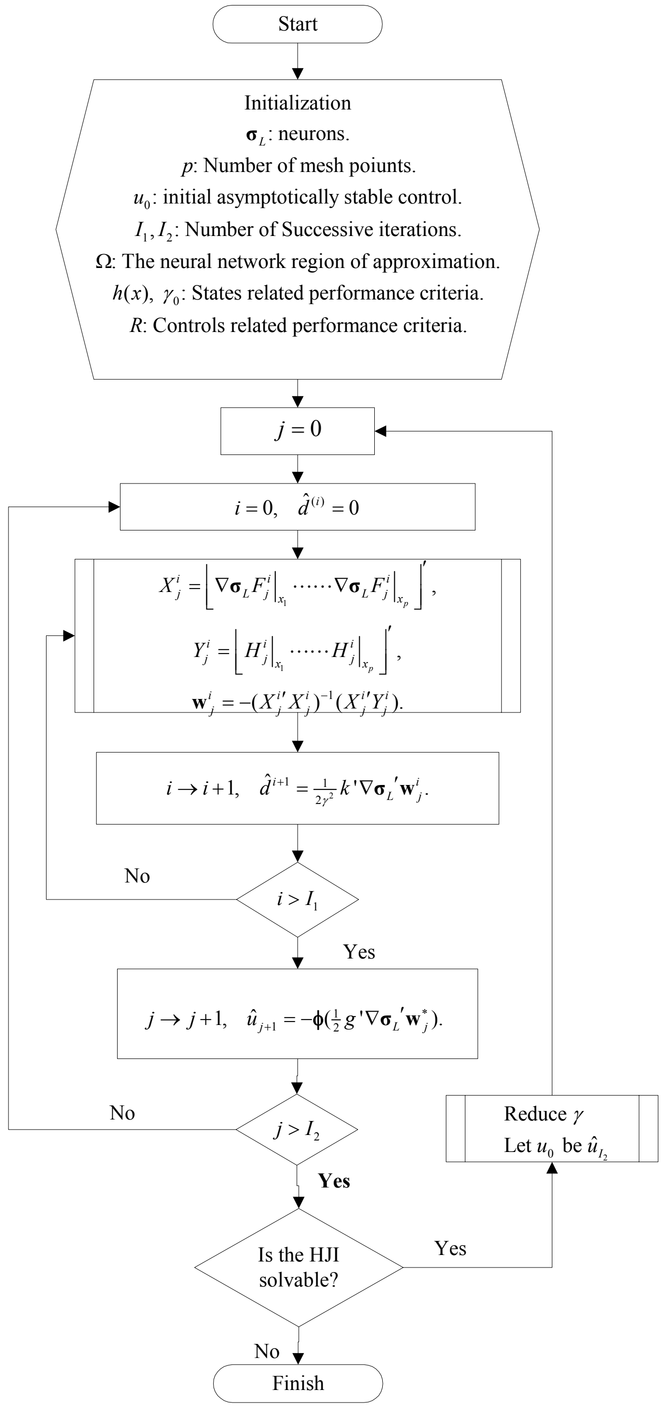


Figure 5.1 Flowchart of the algorithm.

A flowchart of the computational algorithm presented in this chapter is shown in

Figure 5.1. This is an offline algorithm run a priori to obtain a neural network constrained state feedback controller that is nearly L_2-gain optimal. In this algorithm, once the policies converge for some γ_1, one may use the control policy as an initial policy for new inner outer loop policy iterations with $\gamma_2 < \gamma_1$. The attenuation γ is reduced until the HJI equation is no longer solvable on the desired compact set.

5.2 Stability and Convergence of Least-Squares Neural Network Policy Iterations

In this section, the stability and convergence of policy iterations between (5.5), (3.7) and (5.7) is studied. Mainly, it is shown that the closed-loop dynamics resulting from the in the inner loop iterations on the disturbance (3.7) is asymptotically stable as $\hat{d}^{i+1}$ uniformly converges to d^{i+1}. Then later, it is shown that the updated $\hat{u}_{j+1}$ is also stabilizing. Hence, this section starts by showing convergence results of the method of least squares when neural networks are used to solve for V_j^i in. Note that (5.2) is a Fourier series expansion.

In this chapter, a linear in parameter Volterra neural network is used. This gives a power series neural network that has the important property of being differentiable. This means that they can approximate uniformly a continuous function with all its partial derivatives up to order m using the same polynomial, by differentiating the series termwise. This type of series is m-uniformly dense as shown in [1]. Other m-uniformly dense neural networks, not necessarily based on power series, are studied in [35]. To study the convergence properties of the developed neural network algorithm, the following assumptions are required.

Assumption 1: It is assumed that the available storage exists and is positive definite. This is guaranteed for stabilizable dynamics and when the performance functional satisfies zero-state observability.

Assumption 2: The system dynamics and the performance integrands are such that the solution of the $PI(V_j^i, u_j, d^i) = 0$ is continuous and differentiable for all i and j, therefore, belonging to the Sobolev space $V \in H^{1,2}(\Omega)$, [5].

Assumption 3: One can choose complete coordinate elements $\{\sigma_j\}_1^\infty \in H^{1,2}(\Omega)$ such that the solution $V \in H^{1,2}(\Omega)$ and $\{\partial V/\partial x_1, \ldots, \partial V/\partial x_n\}$ can be uniformly approximated by the infinite series built from $\{\sigma_j\}_1^\infty$.

Assumption 4: The sequence $\{\psi_j = A\sigma_j\}$ is linearly independent and complete, and given by

$$A\sigma_j = \frac{d\sigma_j}{dx}' (f + gu + kd).$$

Assumptions 1-3 are standard in H_∞ control theory and neural network control literature. Lemma 1 assures the linear independence required in the fourth assumption while the High-order Weierstrass approximation theorem, [1] [35], shows that

$$\forall V, \varepsilon\, \exists L, \mathbf{w}_L \;\because \left|\hat{V} - V\right| < \varepsilon, \forall k \left|d\hat{V}/dx_k - dV/dx_k\right| < \varepsilon .$$

which implies that as $L \to \infty$

$$\sup_{x\in\Omega}\left|A\hat{V} - AV\right| \to 0 \Rightarrow \left\|A\hat{V} - AV\right\|_{L_2(\Omega)} \to 0 ,$$

and therefore completeness of $\{\psi_j\}$ is established, and the fourth assumption is satisfied.

Similar to the HJB equation [1], one can use the previous assumptions to conclude the uniform convergence of the least-squares method which is placed in the Sobolev space $H^{1,2}(\Omega)$, [5].

Theorem 5.1: *The neural network least squares approach converges uniformly for*

$$\begin{gathered}\sup_{x\in\Omega}\left|d\hat{V}_j^i/dx - dV_j^i/dx\right| \to 0, \quad \sup_{x\in\Omega}\left|\hat{V}_j^i - V_j^i\right| \to 0, \quad \sup_{x\in\Omega}\left|\hat{d}^{i+1} - d^{i+1}\right| \to 0 \\ \sup_{x\in\Omega}\left|\hat{u}_{j+1} - u_{j+1}\right| \to 0\end{gathered}.$$

■

Next, it is shown that the system $\dot{x} = f_j + k\hat{d}^{i+1}$ is asymptotically stable, and hence equation (5.5) can be used to find $\hat{V}^{i+1}$.

Theorem 5.2: $\exists L_0 : L \geq L_0$ *such that* $\dot{x} = f_j + k\hat{d}^{i+1}$ *is asymptotically stable.*

Proof: Since the system $\dot{x} = f_j + kd$ is dissipative with respect to γ, this implies, ,[76] that there exists $P(x) > 0$ such that

$$P_x' f_j + h'h + 2\int_0^{u_j} \phi^{-1}(v)dv + \tfrac{1}{4\gamma^2} P_x' kk' P_x = Q(x) < 0 \tag{5.11}$$

where $\forall i, P(x) \geq V^i(x)$. Since

$$V_x^{i+1\prime}(f_j + \tfrac{1}{2\gamma^2} kk' V_x^i) = -h'h - 2\int_0^{u_j} \phi^{-1}(v)dv + \tfrac{1}{4\gamma^2} V_x^{i\prime} kk' V_x^i, \tag{5.12}$$

one can write the following using equations (5.12) and (5.11)

$$
\begin{aligned}
(P_x - V_x^{i+1})'(f_j + kd^{i+1}) &= P_x' f_j + h'h + 2\int_0^{u_j} \phi^{-1}(v)dv + \tfrac{1}{4\gamma^2} P_x' kk' P_x \\
&\quad - \tfrac{1}{4\gamma^2}(P_x - V_x^i)' kk'(P_x - V_x^i) \qquad (5.13) \\
&= Q(x) - \tfrac{1}{4\gamma^2}(P_x - V_x^i)' kk'(P_x - V_x^i) < 0.
\end{aligned}
$$

Since $\dot{x} = f_j + kd^{i+1}$ and the right hand side of (5.13) is negative definite, it follows that $P(x) - V^{i+1}(x) > 0$. Using $P(x) - V^{i+1}(x) > 0$ as a Lyapunov function candidate for the dynamics $\dot{x} = f_j + k\hat{d}^{i+1}$, one has

$$
\begin{aligned}
(P_x - V_x^{i+1})'(f_j + \tfrac{1}{2\gamma^2} kk'\hat{V}_x^i) &= P_x' f_j + h'h + 2\int_0^{u_j} \phi^{-1}(v)dv + \tfrac{1}{4\gamma^2} P_x' kk' P_x \\
&\quad - \tfrac{1}{4\gamma^2}(P_x - V_x^i)' kk'(P_x - V_x^i) + \tfrac{1}{2\gamma^2}(P_x - V_x^{i+1})' kk'(\hat{V}_x^i - V_x^i) \\
&\le Q(x) + \tfrac{1}{2\gamma^2}(P_x - V_x^{i+1})' kk'(\hat{V}_x^i - V_x^i).
\end{aligned}
$$

From uniform convergence of $\hat{V}^i$ to V^i, $\exists L_0 : L \ge L_0$ such that

$$\forall x \in \Omega, \quad \tfrac{1}{2\gamma^2}(P_x - V_x^{i+1})' kk'(\hat{V}_x^i - V_x^i) > Q(x).$$

This implies that

$$\forall x \in \Omega, \quad (P_x - V_x^{i+1})'(f_j + \tfrac{1}{2\gamma^2} kk'\hat{V}_x^i) < 0.$$

■

Next, it is shown that neural network policy iterations on the control as given by (5.7) is asymptotically stabilizing and L_2-gain stable for the same attenuation γ on Ω.

Theorem 5.3: $\exists L_0 : L \geq L_0$ *such that* $\dot{x} = f + \hat{u}_{j+1}$ *is asymptotically stable.*

Proof: This proof is in essence contained in Corollary 3 in [1] where the positive definiteness of $h(x)$ is utilized by show that uniform convergence of $\hat{V}_j$ to V_j, implies that $\exists L_0 : L \geq L_0$ such that

$$\forall x \in \Omega, \quad (V_{x\,j})'(f + \hat{u}_{j+1}) < 0 .$$

■

Theorem 5.4: *If* $\dot{x} = f + gu_{j+1} + kd$ *has L₂-gain less than* γ*, then it can be shown that* $\exists L_0 : L \geq L_0$ *such that* $\dot{x} = f + g\hat{u}_{j+1} + kd$ *has L₂-gain less than* γ*.*

Proof: Since $\dot{x} = f + gu_{j+1} + kd$ has L_2-gain less than γ, then this implies that there exists a $P(x) > 0$ such that

$$P_x'(f + gu_{j+1}) + h'h + 2\int_0^{u_{j+1}} \phi^{-1}(v)dv + \tfrac{1}{4\gamma^2} P_x' kk' P_x = Q(x) < 0 .$$

Hence, one can show that

$$P_x'(f + g\hat{u}_{j+1}) + h'h + 2\int_0^{\hat{u}_{j+1}} \phi^{-1}(v)dv + \tfrac{1}{4\gamma^2} P_x' kk' P_x = Q(x) + P_x' g(\hat{u}_{j+1} - u_{j+1}) + 2\int_{u_{j+1}}^{\hat{u}_{j+1}} \phi^{-1}(v)dv .$$

From uniform convergence of $\hat{u}_{j+1}$ to u_{j+1}, $\exists L_0 : L \geq L_0$ such that

$$\forall x \in \Omega, \quad P_x' g(\hat{u}_{j+1} - u_{j+1}) + 2\int_{u_{j+1}}^{\hat{u}_{j+1}} \phi^{-1}(v)dv > Q(x) .$$

This implies that

$$\forall x \in \Omega, \quad P_x' g(f + \hat{u}_{j+1}) + h'h + 2\int_0^{\hat{u}_{j+1}} \phi^{-1}(v)dv + \tfrac{1}{4\gamma^2} P_x' kk' P_x < 0 .$$

■

The importance of Theorem 4 is that it justifies solving for the available storage for the new updated dynamics $\dot{x} = f + g\hat{u}_{j+1} + kd$. Hence, all of the preceding theorems can be used to show by induction the following main convergence results.

The next theorem is an important result upon which the algorithm proposed in section 4.4 of this chapter is justified.

Theorem 5.5. $\exists L_0 : L \geq L_0$ *such that*

A. For all j *,* $\dot{x} = f + g\hat{u}_{j+1} + kd$ *is dissipative with* L_2*-gain less than* γ *on* Ω.

B. For all j *and* i *,* $\dot{x} = f + g\hat{u}_{j+1} + kd^i$ *is asymptotically stable on* Ω.

C. $\forall \varepsilon, \exists L_1 > L_0$ *such that* $\sup_{x\in\Omega} \left|\hat{u}_j - u^*\right| < \varepsilon$ *and* $\sup_{x\in\Omega} \left|\hat{V}_j^i - V^*\right| < \varepsilon$.

Proof: The proof follows directly from Theorem 1-4 by induction.

■

5.3 RTAC: The Nonlinear Benchmark Problem

The RTAC benchmark problem was originally proposed in [22] which has received much attention since then. The dynamics of this nonlinear plant pose a challenge as both the rotational and translation motions are coupled as shown. In [75] and [61], unconstrained controls were obtained to solve the L_2 disturbance problem of the RTAC system based on Taylor series solutions of the HJI equation. In [61], unconstrained controllers based on the state-dependent Riccati equation (SDRE) were obtained. The SDRE is easier to solve than the HJI equation and results in a time varying controller that was shown to be suboptimal.

In this section, a neural network constrained input H_∞ state feedback controller is computed for the RTAC shown in Figure 5.2. To our knowledge, this is the first treatment in which inputs constraints are explicitly considered during the design of the optimal H_∞ controller that guarantees optimal disturbance attenuation.

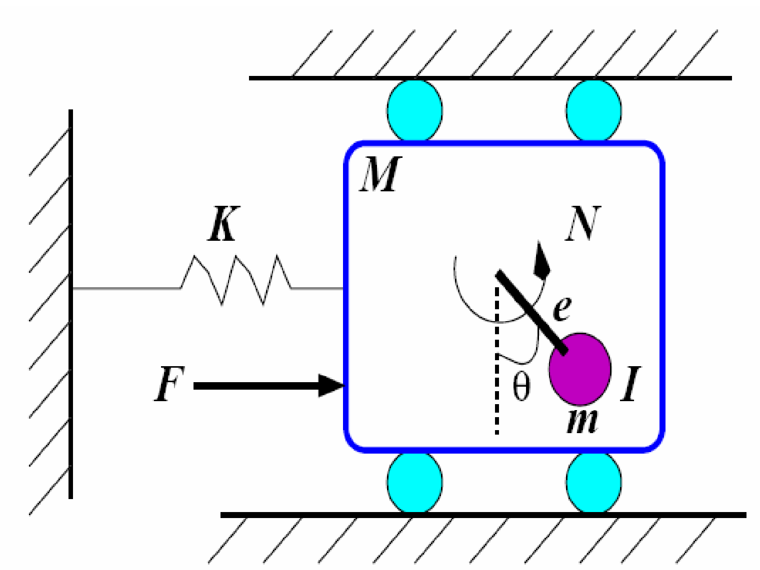


Figure 5.2 Rotational actuator to control a translational oscillator.

The dynamics of the nonlinear plant are given as

$$\dot{x} = f(x) + g(x)u + k(x)d, \quad |u| \le 2$$

$$z'z = x_1^2 + 0.1x_2^2 + 0.1x_3^2 + 0.1x_4^2 + \|u\|_q^2,$$

$$\varepsilon \triangleq me \Big/ \sqrt{(I + me^2)(M + m)} = 0.2, \quad \gamma = 10,$$

$$f = \begin{bmatrix} x_2 \\ \dfrac{-x_1 + \varepsilon x_4^2 \sin x_3}{1 - \varepsilon^2 \cos^2 x_3} \\ x_4 \\ \dfrac{\varepsilon \cos x_3 (x_1 - \varepsilon x_4^2 \sin x_3)}{1 - \varepsilon^2 \cos^2 x_3} \end{bmatrix},$$

$$g = \begin{bmatrix} 0 \\ \dfrac{-\varepsilon \cos x_3}{1 - \varepsilon^2 \cos^2 x_3} \\ 0 \\ \dfrac{1}{1 - \varepsilon^2 \cos^2 x_3} \end{bmatrix}, \quad k = \begin{bmatrix} 0 \\ \dfrac{1}{1 - \varepsilon^2 \cos^2 x_3} \\ 0 \\ \dfrac{-\varepsilon \cos x_3}{1 - \varepsilon^2 \cos^2 x_3} \end{bmatrix}. \tag{5.14}$$

with the state $x_1 = r, x_2 = \dot{r}\ x_3 = \theta, x_4 = \dot{\theta}$, [22].

The design steps procedure goes as follows:

- Initial control selection:

The following H_∞ controller of the linear system resulting from Jacobian linearization of (5.14) is chosen

$$u_0 = 2\tanh(2.4182x_1 + 1.1650x_2 - 0.3416x_3 - 1.0867x_4),$$

and forced to obey the $|u| \le 2$ constraint. This is a stabilizing controller that guarantees that L_2-gain<6 for the Jacobian linearized system, [75]. The neural network is going to be trained on the following region of the state space $|x_i| \le 2 \quad i = 1,2,3,4$ which is a subset of the region of asymptotic stability of u_0 that can be estimated using techniques in [30].

- Policy iterations:

The iterative algorithm starts by approximately solving for the HJI with $\gamma = 30$. The approximate solution is done by inner loop iterations between (3.7) and (5.10) followed by outer-loop policy iterations (5.7).

In the simulation performed, the neurons of the neural network were chosen from the 6th order series expansion of the value function. Only polynomial terms of even order were considered, therefore having the total number of neural networks is $L = 129$ and is shown in Figure 5.3. A sixth order series approximation of the value function was satisfactory for our purposes, and it results in a 5th order controller as done for the unconstrained case in [38].

Once the neural network algorithm converge, and an approximate solution for (4.33) with $\gamma = 30$, the resulting controller can be used as an initial controller for a new inner outer loop iterations to solve (4.33) with a smaller γ.

The computational routine was successful in obtaining approximate solutions to (4.33) with $\gamma = 10$ with the final weights are given Figure 5.4.

The controller is finally given as

$$u = -\frac{1}{2} g'(x) \nabla \boldsymbol{\sigma}_L{}' \mathbf{w}.$$

The neural network activation functions are shown in Figure 5.3. Note that this is a Volterra type neural network.

$$
\begin{aligned}
\sigma_L = [\, & x1^2,\ x1\,x2,\ x1\,x3,\ x1\,x4,\ x2^2, \\
& x2\,x3,\ x2\,x4,\ x3^2,\ x3\,x4,\ x4^2,\ x1^4, \\
& x1^3\,x2,\ x1^3\,x3,\ x1^3\,x4,\ x1^2\,x2^2,\ x1^2\,x2\,x3, \\
& x1^2\,x2\,x4,\ x1^2\,x3^2,\ x1^2\,x3\,x4,\ x1^2\,x4^2, \\
& x1\,x2^3,\ x1\,x2^2\,x3,\ x1\,x2^2\,x4,\ x1\,x2\,x3^2, \\
& x1\,x2\,x3\,x4,\ x1\,x2\,x4^2,\ x1\,x3^3,\ x1\,x3^2\,x4, \\
& x1\,x3\,x4^2,\ x1\,x4^3,\ x2^4,\ x2^3\,x3,\ x2^3\,x4, \\
& x2^2\,x3^2,\ x2^2\,x3\,x4,\ x2^2\,x4^2,\ x2\,x3^3, \\
& x2\,x3^2\,x4,\ x2\,x3\,x4^2,\ x2\,x4^3,\ x3^4,\ x3^3\,x4, \\
& x3^2\,x4^2,\ x3\,x4^3,\ x4^4,\ x1^6,\ x1^5\,x2,\ x1^5\,x3, \\
& x1^5\,x4,\ x1^4\,x2^2,\ x1^4\,x2\,x3,\ x1^4\,x2\,x4, \\
& x1^4\,x3^2,\ x1^4\,x3\,x4,\ x1^4\,x4^2,\ x1^3\,x2^3, \\
& x1^3\,x2^2\,x3,\ x1^3\,x2^2\,x4,\ x1^3\,x2\,x3^2, \\
& x1^3\,x2\,x3\,x4,\ x1^3\,x2\,x4^2,\ x1^3\,x3^3, \\
& x1^3\,x3^2\,x4,\ x1^3\,x3\,x4^2,\ x1^3\,x4^3,\ x1^2\,x2^4, \\
& x1^2\,x2^3\,x3,\ x1^2\,x2^3\,x4,\ x1^2\,x2^2\,x3^2, \\
& x1^2\,x2^2\,x3\,x4,\ x1^2\,x2^2\,x4^2,\ x1^2\,x2\,x3^3, \\
& x1^2\,x2\,x3^2\,x4,\ x1^2\,x2\,x3\,x4^2,\ x1^2\,x2\,x4^3, \\
& x1^2\,x3^4,\ x1^2\,x3^3\,x4,\ x1^2\,x3^2\,x4^2,\ x1^2\,x3\,x4^3, \\
& x1^2\,x4^4,\ x1\,x2^5,\ x1\,x2^4\,x3,\ x1\,x2^4\,x4, \\
& x1\,x2^3\,x3^2,\ x1\,x2^3\,x3\,x4,\ x1\,x2^3\,x4^2, \\
& x1\,x2^2\,x3^3,\ x1\,x2^2\,x3^2\,x4,\ x1\,x2^2\,x3\,x4^2, \\
& x1\,x2^2\,x4^3,\ x1\,x2\,x3^4,\ x1\,x2\,x3^3\,x4, \\
& x1\,x2\,x3^2\,x4^2,\ x1\,x2\,x3\,x4^3,\ x1\,x2\,x4^4, \\
& x1\,x3^5,\ x1\,x3^4\,x4,\ x1\,x3^3\,x4^2,\ x1\,x3^2\,x4^3, \\
& x1\,x3\,x4^4,\ x1\,x4^5,\ x2^6,\ x2^5\,x3,\ x2^5\,x4, \\
& x2^4\,x3^2,\ x2^4\,x3\,x4,\ x2^4\,x4^2,\ x2^3\,x3^3, \\
& x2^3\,x3^2\,x4,\ x2^3\,x3\,x4^2,\ x2^3\,x4^3,\ x2^2\,x3^4, \\
& x2^2\,x3^3\,x4,\ x2^2\,x3^2\,x4^2,\ x2^2\,x3\,x4^3, \\
& x2^2\,x4^4,\ x2\,x3^5,\ x2\,x3^4\,x4,\ x2\,x3^3\,x4^2, \\
& x2\,x3^2\,x4^3,\ x2\,x3\,x4^4,\ x2\,x4^5,\ x3^6,\ x3^5\,x4, \\
& x3^4\,x4^2,\ x3^3\,x4^3,\ x3^2\,x4^4,\ x3\,x4^5,\ x4^6\,]
\end{aligned}
$$

Figure 5.3 Volterra neural network used in the RTAC example.

$$
\mathbf{w} = [
$$

7.5591 -0.5592 -0.0398 -2.0616 7.5212 1.7514 $\cdots$
3.0072 0.3526 1.2436 1.3561 0.0910 0.0082 $\cdots$
-0.1817 -0.1380 0.1958 0.1807 0.1441 0.3113 $\cdots$
0.4315 0.2912 0.0057 -0.1288 -0.0817 0.2979 $\cdots$
0.3864 0.1383 -0.2192 0.4320 0.1636 0.0131 $\cdots$
0.1107 0.1727 0.2055 0.0897 0.3292 0.3234 $\cdots$
-0.4341 -1.9855 -0.1703 -0.0064 0.1540 -0.1364 $\cdots$
-0.2915 0.0053 0.0407 0.0029 -0.0125 0.0142 $\cdots$
0.0071 0.0061 -0.0099 -0.0072 -0.0060 -0.0123 $\cdots$
-0.0082 -0.0110 0.0289 0.0193 0.0033 -0.0147 $\cdots$
0.0052 0.0074 0.0098 0.0001 0.0016 0.0047 $\cdots$
-0.0138 -0.0084 -0.0047 -0.0192 -0.0258 -0.0177 $\cdots$
-0.0408 -0.0187 -0.0053 -0.0012 -0.0144 -0.0260 $\cdots$
-0.0080 0.0062 -0.0011 0.0140 0.0109 -0.0031 $\cdots$
-0.0127 -0.0051 -0.0041 -0.0134 -0.0131 -0.0141 $\cdots$
-0.0292 -0.0178 -0.0089 -0.0243 -0.0125 0.0022 $\cdots$
-0.0482 -0.0388 0.0184 0.0366 0.0064 0.0011 $\cdots$
-0.0063 -0.0042 -0.0004 -0.0102 -0.0150 -0.0141 $\cdots$
-0.0515 -0.0319 -0.0144 0.0157 0.0003 0.0200 $\cdots$
0.0398 0.0091 0.0346 0.1461 -0.0217 -0.0407 $\cdots$
-0.0048 -0.0008 -0.0273 0.0100 0.0493 0.0037 $\cdots$
-0.0105 -0.0167 -0.0058]$'$.

Figure 5.4 Weight of the Volterra neural network used in the RTAC example.

- Simulation:

Figures 5.5 and 5.6 show the states trajectories when the system is at rest and experiencing a disturbance $d(t) = 5\sin(t)e^{-t}$. Figure 5.7 shows the control signal, while Figure 5.8 shows the attenuation

$$\int_0^\infty \|z(t)\|^2 \, dt \Big/ \int_0^\infty \|d(t)\|^2 \, dt \, .$$

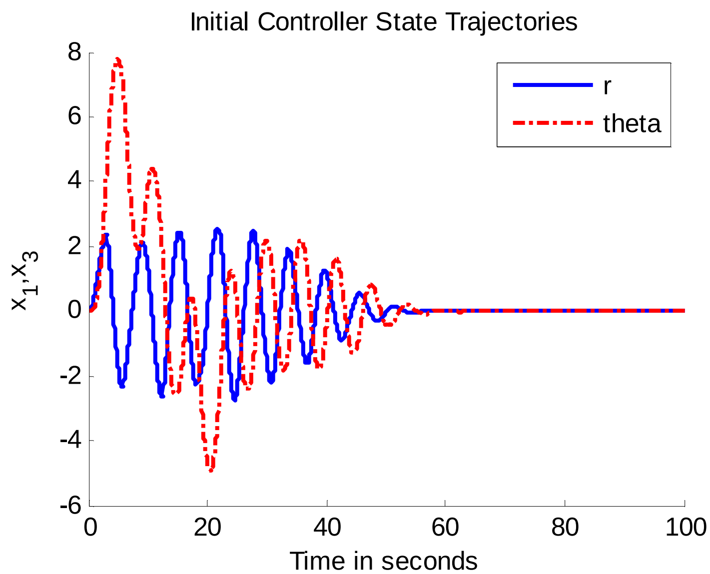


Figure 5.5 r , θ state trajectories.

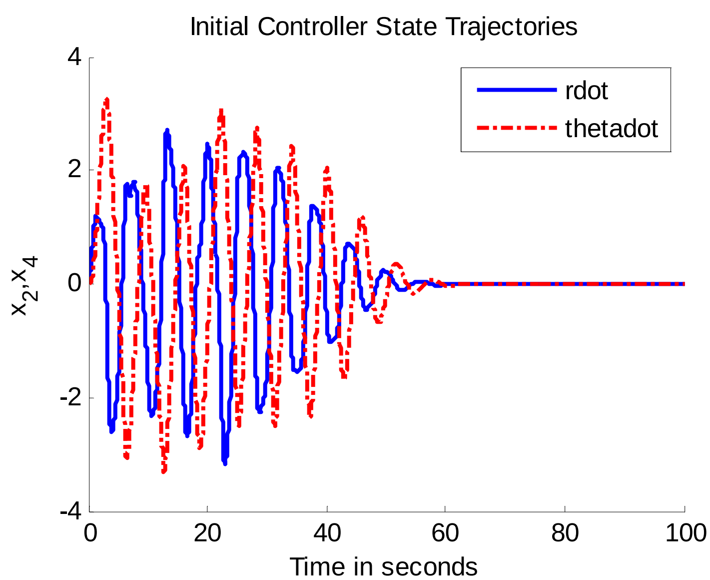


Figure 5.6 $\dot{r}$, $\dot{\theta}$ state trajectories.

Figures 5.9 and 5.10 shows the states trajectories when the system is at rest and experiencing a disturbance $d(t) = 5\sin(t)e^{-t}$. Figures 5.11 and 5.12 shows the control signal and attenuation respectively.

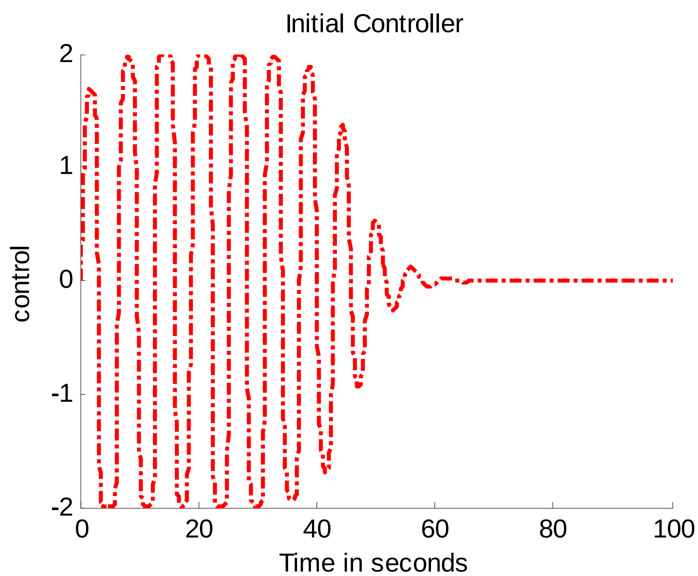


Figure 5.7 $u(t)$ control input.

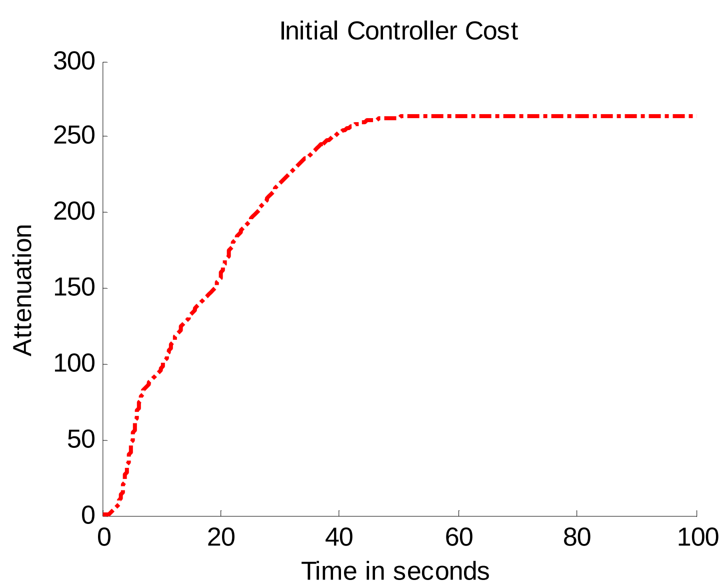


Figure 5.8 Disturbance attenuation.

The nearly optimal nonlinear constrained input H_∞ controller is shown to perform much better than the initial controller the algorithm started with. It is novel utilization of neural networks approximation property to obtain a closed-form solution to the constrained input H_∞ control policy that is very hard to find otherwise.

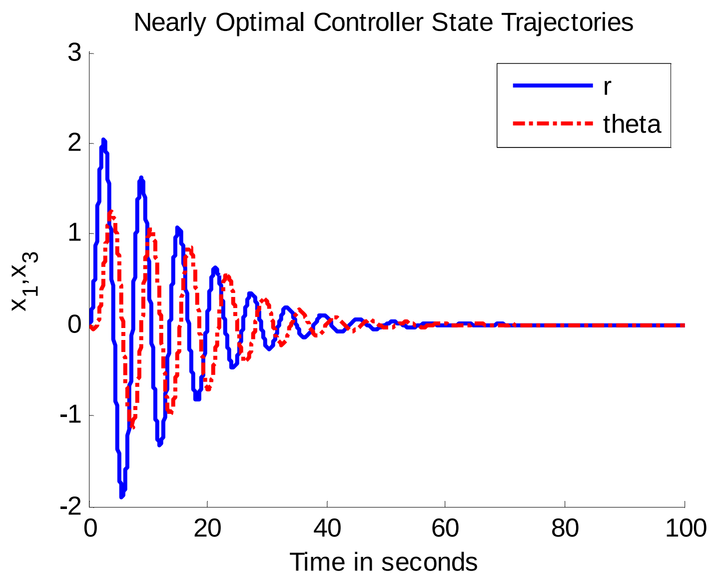


Figure 5.9 Nearly optimal r , θ state trajectories.

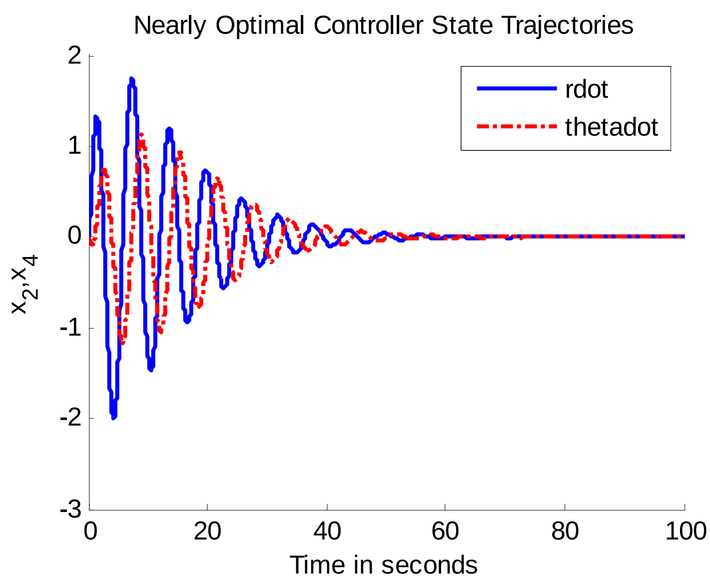


Figure 5.10 Nearly optimal $\dot{r}$, $\dot{\theta}$ state trajectories.

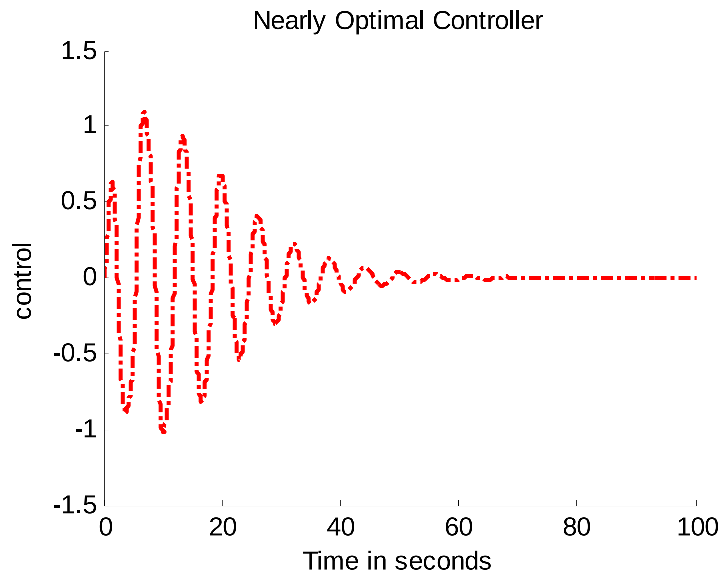


Figure 5.11 Nearly optimal $u(t)$ control input.

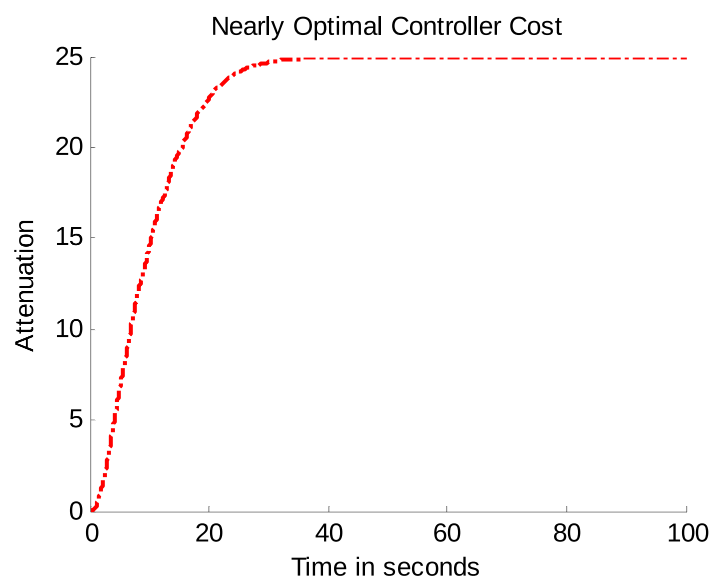


Figure 5.12 Nearly optimal disturbance attenuation.

5.4 Conclusions

This chapter presents an application of neural networks to find closed form representation of feedback strategies for a zero-sum game that appears in the H_∞ control. The systems considered are affine in input with control saturation. The algorithm relies on policy iterations that has been proposed for unconstrained, [17], and constrained, [2], control case. The presented algorithms is an extension to the optimal quadratic regulations for constrained inputs using the HJB equation appearing in [1]. The results of this chapter and [1] can be further researched to provide an adaptive optimal control schemes, approximate dynamic programming, in which the presented algorithm is required to be implemented online.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

In this dissertation, neural networks are used to obtain closed-form representation of feedback policies for optimal control and zero-sum games with actuator saturation. The main theme of this research is applying policy iterations and neural network function approximation property to solve the corresponding Hamilton-Jacobi equations. The stability and convergence results of these techniques were demonstrated throughout the dissertation.

6.1 Contributions

The contributions of this research can be summarized in the following points:

1. In Chapter two, it is shown that the HJB equation previously derived for constrained input systems using quasi-norms in [58] can be broken into a sequence of Lyapunov equations using the method of policy iterations, which has some history and applied earlier unconstrained input systems [72], [14]. The uniform convergence of the policy iteration method is demonstrated, and it is shown that the constrained input optimal controller has the largest region of attraction.
2. In Chapter three, the sequence of Lyapunov equations derived in Chapter two are solved for using neural networks in the least-squares sense. Convergence results are shown. Several examples are given to illustrate the approach. Constrained state, and

minimum-time control problems are discussed.

3. In Chapter 4, the HJI equation for constrained input zero-sum games is derived using quasi-norms, and it is shown that the resulting policies are in saddle point equilibrium.
4. Another contribution of Chapter 4 is that it proves convergence of policy iterations to the HJB equation obtained in the nonlinear Bounded Real Lemma in L_2-gain problems.
5. Another contribution in Chapter 4 is it is shown how to use two-player policy iterations for continuous-time zero-sum games to solve the constrained input HJI equation. This sort of policy iterations is known for systems with no constraints. The contribution, besides introducing them to systems with constraints, is that in Chapter 4 it is shown that two-player policy iterations have a connection with the convergence of the policy iteration method for the nonlinear Bounded Real Lemma. Two-player policy iterations to solve continuous-time zero-sum games appears for the first time in [17], however convergence of the method, in particular, the inner loop iteration is not clearly understood. In Chapter 4, this issue was resolved in Theorem 4.1.
6. In Chapter 5, it is shown how to use neural networks to solve for the policy iteration equations appearing in Chapter 4.
7. In Chapter 5, the constrained input H_∞ controller for the nonlinear benchmark problem, [22], is solved. Earlier work on this problem did not consider the constraints on the input.

6.2 Future Work

In this dissertation, it is assumed that one has access to the full state information. In future work, it is important to consider output feedback problems. Currently work is on the way for the static output feedback problem, [40].

Further more, one can considered the case of online training of the neural network. So far, the algorithms considered in this dissertation were offline techniques.

It would be interesting to see how the policy iteration technique can be employed to solve optimal control problems of discrete-time nonlinear systems.

Another major thrust would be to implement adaptive version of the optimal control laws derived by tuning them in real time without requiring the explicit knowledge of the system dynamics. It has been noticed that policy iterations with Q-learning known in the artificial intelligence converges to the optimal controller of a linear discrete-time system without the explicit knowledge of the system model [47].

APPENDIX A

MATLAB M-FILES OF NONLINEAR BENCHMARK PROBLEM

*****Policy iteration main file*****

```
% RTAC Example for Hinfinity
% Prepared by: MAK

close all;clc;clear all;
EPS=0.2;
N=30000;
neurons=129;
gamma=10.0;
A=2;
tic
for Control_Iteration=1:7
    for Disturbance_Iteration=1:7
        P=1e10*eye(neurons,neurons);
        W=zeros(neurons,1);
        for RLS_Iteration=1:N
            x1=-2+2*2*rand;
            x2=-2+2*2*rand;
            x3=-2+2*2*rand;
            x4=-2+2*2*rand;
            dNN=[   2*x1            0               0               0
                x2              x1              0               0
                x3              0               x1              0
                x4              0               0               x1
                0               2*x2            0               0
                0               x3              x2              0
                0               x4              0               x2
                0               0               2*x3            0
                0               0               x4              x3
                0               0               0               2*x4

                4*x1^3          0               0               0
                3*x1^2*x2       x1^3            0               0
                3*x1^2*x3       0               x1^3            0
                3*x1^2*x4       0               0               x1^3
                2*x1*x2^2       2*x1^2*x2       0               0
                2*x1*x2*x3      x1^2*x3         x1^2*x2         0
                2*x1*x2*x4      x1^2*x4         0               x1^2*x2
                2*x1*x3^2       0               2*x1^2*x3       0
                2*x1*x3*x4      0               x1^2*x4         x1^2*x3
                2*x1*x4^2       0               0               2*x1^2*x4
                x2^3            3*x1*x2^2       0               0
                x2^2*x3         2*x1*x2*x3      x1*x2^2         0
                x2^2*x4         2*x1*x2*x4      0               x1*x2^2
                x2*x3^2         x1*x3^2         2*x1*x2*x3      0
                x2*x3*x4        x1*x3*x4        x1*x2*x4        x1*x2*x3
                x2*x4^2         x1*x4^2         0               2*x1*x2*x4
                x3^3            0               3*x1*x3^2       0
                x3^2*x4         0               2*x1*x3*x4      x1*x3^2
                x3*x4^2         0               x1*x4^2         2*x1*x3*x4
                x4^3            0               0               3*x1*x4^2
                0               4*x2^3          0               0
                0               3*x2^2*x3       x2^3            0
                0               3*x2^2*x4       0               x2^3
                0               2*x2*x3^2       2*x2^2*x3       0
                0               2*x2*x3*x4      x2^2*x4         x2^2*x3
                0               2*x2*x4^2       0               2*x2^2*x4
                0               x3^3            3*x2*x3^2       0
                0               x3^2*x4         2*x2*x3*x4      x2*x3^2
                0               x3*x4^2         x2*x4^2         2*x2*x3*x4
                0               x4^3            0               3*x2*x4^2
                0               0               4*x3^3          0
                0               0               3*x3^2*x4       x3^3
                0               0               2*x3*x4^2       2*x3^2*x4
                0               0               x4^3            3*x3*x4^2
```

0	0	0	4*x4^3

6*x1^5	0	0	0
5*x1^4*x2	x1^5	0	0
5*x1^4*x3	0	x1^5	0
5*x1^4*x4	0	0	x1^5
4*x1^3*x2^2	2*x1^4*x2	0	0
4*x1^3*x2*x3	x1^4*x3	x1^4*x2	0
4*x1^3*x2*x4	x1^4*x4	0	x1^4*x2
4*x1^3*x3^2	0	2*x1^4*x3	0
4*x1^3*x3*x4	0	x1^4*x4	x1^4*x3
4*x1^3*x4^2	0	0	2*x1^4*x4
3*x1^2*x2^3	3*x1^3*x2^2	0	0
3*x1^2*x2^2*x3	2*x1^3*x2*x3	x1^3*x2^2	0
3*x1^2*x2^2*x4	2*x1^3*x2*x4	0	x1^3*x2^2
3*x1^2*x2*x3^2	x1^3*x3^2	2*x1^3*x2*x3	0
3*x1^2*x2*x3*x4	x1^3*x3*x4	x1^3*x2*x4	x1^3*x2*x3
3*x1^2*x2*x4^2	x1^3*x4^2	0	2*x1^3*x2*x4
3*x1^2*x3^3	0	3*x1^3*x3^2	0
3*x1^2*x3^2*x4	0	2*x1^3*x3*x4	x1^3*x3^2
3*x1^2*x3*x4^2	0	x1^3*x4^2	2*x1^3*x3*x4
3*x1^2*x4^3	0	0	3*x1^3*x4^2
2*x1*x2^4	4*x1^2*x2^3	0	0
2*x1*x2^3*x3	3*x1^2*x2^2*x3	x1^2*x2^3	0
2*x1*x2^3*x4	3*x1^2*x2^2*x4	0	x1^2*x2^3
2*x1*x2^2*x3^2	2*x1^2*x2*x3^2	2*x1^2*x2^2*x3	0
2*x1*x2^2*x3*x4	2*x1^2*x2*x3*x4	x1^2*x2^2*x4	x1^2*x2^2*x3
2*x1*x2^2*x4^2	2*x1^2*x2*x4^2	0	2*x1^2*x2^2*x4
2*x1*x2*x3^3	x1^2*x3^3	3*x1^2*x2*x3^2	0
2*x1*x2*x3^2*x4	x1^2*x3^2*x4	2*x1^2*x2*x3*x4	x1^2*x2*x3^2
2*x1*x2*x3*x4^2	x1^2*x3*x4^2	x1^2*x2*x4^2	2*x1^2*x2*x3*x4
2*x1*x2*x4^3	x1^2*x4^3	0	3*x1^2*x2*x4^2
2*x1*x3^4	0	4*x1^2*x3^3	0
2*x1*x3^3*x4	0	3*x1^2*x3^2*x4	x1^2*x3^3
2*x1*x3^2*x4^2	0	2*x1^2*x3*x4^2	2*x1^2*x3^2*x4
2*x1*x3*x4^3	0	x1^2*x4^3	3*x1^2*x3*x4^2
2*x1*x4^4	0	0	4*x1^2*x4^3
x2^5	5*x1*x2^4	0	0
x2^4*x3	4*x1*x2^3*x3	x1*x2^4	0
x2^4*x4	4*x1*x2^3*x4	0	x1*x2^4
x2^3*x3^2	3*x1*x2^2*x3^2	2*x1*x2^3*x3	0
x2^3*x3*x4	3*x1*x2^2*x3*x4	x1*x2^3*x4	x1*x2^3*x3
x2^3*x4^2	3*x1*x2^2*x4^2	0	2*x1*x2^3*x4
x2^2*x3^3	2*x1*x2*x3^3	3*x1*x2^2*x3^2	0
x2^2*x3^2*x4	2*x1*x2*x3^2*x4	2*x1*x2^2*x3*x4	x1*x2^2*x3^2
x2^2*x3*x4^2	2*x1*x2*x3*x4^2	x1*x2^2*x4^2	2*x1*x2^2*x3*x4
x2^2*x4^3	2*x1*x2*x4^3	0	3*x1*x2^2*x4^2
x2*x3^4	x1*x3^4	4*x1*x2*x3^3	0
x2*x3^3*x4	x1*x3^3*x4	3*x1*x2*x3^2*x4	x1*x2*x3^3
x2*x3^2*x4^2	x1*x3^2*x4^2	2*x1*x2*x3*x4^2	2*x1*x2*x3^2*x4
x2*x3*x4^3	x1*x3*x4^3	x1*x2*x4^3	3*x1*x2*x3*x4^2
x2*x4^4	x1*x4^4	0	4*x1*x2*x4^3
x3^5	0	5*x1*x3^4	0
x3^4*x4	0	4*x1*x3^3*x4	x1*x3^4
x3^3*x4^2	0	3*x1*x3^2*x4^2	2*x1*x3^3*x4
x3^2*x4^3	0	2*x1*x3*x4^3	3*x1*x3^2*x4^2
x3*x4^4	0	x1*x4^4	4*x1*x3*x4^3
x4^5	0	0	5*x1*x4^4
0	6*x2^5	0	0
0	5*x2^4*x3	x2^5	0
0	5*x2^4*x4	0	x2^5
0	4*x2^3*x3^2	2*x2^4*x3	0
0	4*x2^3*x3*x4	x2^4*x4	x2^4*x3
0	4*x2^3*x4^2	0	2*x2^4*x4
0	3*x2^2*x3^3	3*x2^3*x3^2	0
0	3*x2^2*x3^2*x4	2*x2^3*x3*x4	x2^3*x3^2
0	3*x2^2*x3*x4^2	x2^3*x4^2	2*x2^3*x3*x4

```
   0               3*x2^2*x4^3     0               3*x2^3*x4^2
   0               2*x2*x3^4       4*x2^2*x3^3     0
   0               2*x2*x3^3*x4    3*x2^2*x3^2*x4  x2^2*x3^3
   0               2*x2*x3^2*x4^2  2*x2^2*x3*x4^2  2*x2^2*x3^2*x4
   0               2*x2*x3*x4^3    x2^2*x4^3       3*x2^2*x3*x4^2
   0               2*x2*x4^4       0               4*x2^2*x4^3
   0               x3^5            5*x2*x3^4       0
   0               x3^4*x4         4*x2*x3^3*x4    x2*x3^4
   0               x3^3*x4^2       3*x2*x3^2*x4^2  2*x2*x3^3*x4
   0               x3^2*x4^3       2*x2*x3*x4^3    3*x2*x3^2*x4^2
   0               x3*x4^4         x2*x4^4         4*x2*x3*x4^3
   0               x4^5            0               5*x2*x4^4
   0               0               6*x3^5          0
   0               0               5*x3^4*x4       x3^5
   0               0               4*x3^3*x4^2     2*x3^4*x4
   0               0               3*x3^2*x4^3     3*x3^3*x4^2
   0               0               2*x3*x4^4       4*x3^2*x4^3
   0               0               x4^5            5*x3*x4^4
   0               0               0               6*x4^5];
beta_x=-x1+EPS*x4^2*sin(x3);
gamma_x=1-EPS^2*cos(x3)^2;
f = [x2
     beta_x/gamma_x
     x4
    -EPS*beta_x*cos(x3)/gamma_x];
g=[  0;
    -EPS*cos(x3)/gamma_x
     0
     1/gamma_x];
k=[  0;
     1/gamma_x
     0
    -EPS*cos(x3)/gamma_x];

if Control_Iteration==1
   K=[2.41817 1.16494 -.34158 -1.08667];
   K=-[-1.3862   -0.0271    1.0000    1.8634];
   U=K*([x1;x2;x3;x4]-[0 0 0 0]');
   u=A*tanh(1/A*U);
   u = A*tanh(-0.5*g'*dNN'*…
 [7.5591  -0.5592 -0.0398 -2.0616 7.5212  1.7514  3.0072  0.3526  1.2436...
 1.3561  0.091   0.0082  -0.1817 -0.138  0.1958  0.1807  0.1441  0.3113...
 0.4315  0.2912  0.0057  -0.1288 -0.0817 0.2979  0.3864  0.1383  -0.2192...
 0.432   0.1636  0.0131  0.1107  0.1727  0.2055  0.0897  0.3292  0.3234...
 -0.4341 -1.9855 -0.1703 -0.0064 0.154   -0.1364 -0.2915 0.0053  0.0407...
 0.0029  -0.0125 0.0142  0.0071  0.0061  -0.0099 -0.0072 -0.006  -0.0123...
 -0.0082 -0.011  0.0289  0.0193  0.0033  -0.0147 0.0052  0.0074  0.0098...
 0.0001  0.0016  0.0047  -0.0138 -0.0084 -0.0047 -0.0192 -0.0258 -0.0177...
 -0.0408 -0.0187 -0.0053 -0.0012 -0.0144 -0.026  -0.008  0.0062  -0.0011...
 0.014   0.0109  -0.0031 -0.0127 -0.0051 -0.0041 -0.0134 -0.0131 -0.0141...
 -0.0292 -0.0178 -0.0089 -0.0243 -0.0125 0.0022  -0.0482 -0.0388 0.0184...
 0.0366  0.0064  0.0011  -0.0063 -0.0042 -0.0004 -0.0102 -0.015  -0.0141...
 -0.0515 -0.0319 -0.0144 0.0157  0.0003  0.02    0.0398  0.0091  0.0346...
 0.1461  -0.0217 -0.0407 -0.0048 -0.0008 -0.0273 0.01    0.0493  0.0037...
 -0.0105 -0.0167 -0.0058]'/A);
   if abs(u)>0.9999999999*A
       u=0.9999999999*A*sign(u);
   end
   else
    u = A*tanh(-0.5*g'*dNN'*Woo/A);
    if abs(u)>0.9999999999*A
        u=0.9999999999*A*sign(u);
    end
end
if Disturbance_Iteration==1
    d = 0;
else
```

```
                d = 0.5*k'*dNN'*Wo/gamma^2;
            end
            % Implement RLS
            phi=dNN*(f+g*u+k*d);
            y    = -x1^2-0.1*x2^2-0.1*x3^2-0.1*x4^2-2*A*(u*atanh(u/A)+0.5*A*log(1.0-
(u/A)^2))+gamma^2*d*d;
            yhat = W'*phi;
            P=P-P*phi/(1+phi'*P*phi)*phi'*P;
            K=P*phi;
            W=W+K*(y-yhat);
        end
%         clc;
        Wo=W;
        gamma
        Control_Iteration
        Disturbance_Iteration
        Wo(1:10)
%         signal(:,Disturbance_Iteration)=Wo;
%         figure(1); hold on;
%         plot(signal'); plot(signal','.');
        toc
    end
    close all;
    Woo=Wo
end
save W.txt W -ASCII
```

*****Simulation ODESTART file*****

```
close all;clear all;clc;

global W;
load W.txt;
global A;
A=2;
ti=0;
tf=100;
tspan=[ti tf];
%for ii=1:100
x0=[-1.0 1.7 1.5 1.0 0 0]*0;
%x0=[1 0 1 0 0 0];
%x0=[-1.0 1 -1 1.0 0 0];



options=odeset('RelTol',1e-8);
[t,x]= ode45('RTACfile',tspan,[x0],options);
figure(1);hold on;
ylabel('x_1,x_3');xlabel('Time in seconds');%title('No title yet');
plot(t,x(:,1),'b-','LineWidth',2);
plot(t,x(:,3),'r-.','LineWidth',2);
legend('r','theta');
title('Nearly Optimal Controller State Trajectories');
title('Initial Controller State Trajectories');

figure(2);hold on;
ylabel('x_2,x_4');xlabel('Time in seconds');%title('No title yet');
plot(t,x(:,2),'b-','LineWidth',2);
plot(t,x(:,4),'r-.','LineWidth',2);
legend('rdot','thetadot');
title('Nearly Optimal Controller State Trajectories');
title('Initial Controller State Trajectories');

figure(3);hold on;
for i=1:length(x)
    x1=x(i,1);x2=x(i,2);x3=x(i,3);x4=x(i,4);
    dPHI=[   2*x1            0               0               0
                x2              x1              0               0
                x3              0               x1              0
                x4              0               0               x1
                0               2*x2            0               0
                0               x3              x2              0
                0               x4              0               x2
                0               0               2*x3            0
                0               0               x4              x3
                0               0               0               2*x4


                4*x1^3          0               0               0
                3*x1^2*x2       x1^3            0               0
                3*x1^2*x3       0               x1^3            0
                3*x1^2*x4       0               0               x1^3
                2*x1*x2^2       2*x1^2*x2       0               0
                2*x1*x2*x3      x1^2*x3         x1^2*x2         0
                2*x1*x2*x4      x1^2*x4         0               x1^2*x2
                2*x1*x3^2       0               2*x1^2*x3       0
                2*x1*x3*x4      0               x1^2*x4         x1^2*x3
                2*x1*x4^2       0               0               2*x1^2*x4
                x2^3            3*x1*x2^2       0               0
                x2^2*x3         2*x1*x2*x3      x1*x2^2         0
                x2^2*x4         2*x1*x2*x4      0               x1*x2^2
                x2*x3^2         x1*x3^2         2*x1*x2*x3      0
```

x2*x3*x4	x1*x3*x4	x1*x2*x4	x1*x2*x3
x2*x4^2	x1*x4^2	0	2*x1*x2*x4
x3^3	0	3*x1*x3^2	0
x3^2*x4	0	2*x1*x3*x4	x1*x3^2
x3*x4^2	0	x1*x4^2	2*x1*x3*x4
x4^3	0	0	3*x1*x4^2
0	4*x2^3	0	0
0	3*x2^2*x3	x2^3	0
0	3*x2^2*x4	0	x2^3
0	2*x2*x3^2	2*x2^2*x3	0
0	2*x2*x3*x4	x2^2*x4	x2^2*x3
0	2*x2*x4^2	0	2*x2^2*x4
0	x3^3	3*x2*x3^2	0
0	x3^2*x4	2*x2*x3*x4	x2*x3^2
0	x3*x4^2	x2*x4^2	2*x2*x3*x4
0	x4^3	0	3*x2*x4^2
0	0	4*x3^3	0
0	0	3*x3^2*x4	x3^3
0	0	2*x3*x4^2	2*x3^2*x4
0	0	x4^3	3*x3*x4^2
0	0	0	4*x4^3

6*x1^5	0	0	0
5*x1^4*x2	x1^5	0	0
5*x1^4*x3	0	x1^5	0
5*x1^4*x4	0	0	x1^5
4*x1^3*x2^2	2*x1^4*x2	0	0
4*x1^3*x2*x3	x1^4*x3	x1^4*x2	0
4*x1^3*x2*x4	x1^4*x4	0	x1^4*x2
4*x1^3*x3^2	0	2*x1^4*x3	0
4*x1^3*x3*x4	0	x1^4*x4	x1^4*x3
4*x1^3*x4^2	0	0	2*x1^4*x4
3*x1^2*x2^3	3*x1^3*x2^2	0	0
3*x1^2*x2^2*x3	2*x1^3*x2*x3	x1^3*x2^2	0
3*x1^2*x2^2*x4	2*x1^3*x2*x4	0	x1^3*x2^2
3*x1^2*x2*x3^2	x1^3*x3^2	2*x1^3*x2*x3	0
3*x1^2*x2*x3*x4	x1^3*x3*x4	x1^3*x2*x4	x1^3*x2*x3
3*x1^2*x2*x4^2	x1^3*x4^2	0	2*x1^3*x2*x4
3*x1^2*x3^3	0	3*x1^3*x3^2	0
3*x1^2*x3^2*x4	0	2*x1^3*x3*x4	x1^3*x3^2
3*x1^2*x3*x4^2	0	x1^3*x4^2	2*x1^3*x3*x4
3*x1^2*x4^3	0	0	3*x1^3*x4^2
2*x1*x2^4	4*x1^2*x2^3	0	0
2*x1*x2^3*x3	3*x1^2*x2^2*x3	x1^2*x2^3	0
2*x1*x2^3*x4	3*x1^2*x2^2*x4	0	x1^2*x2^3
2*x1*x2^2*x3^2	2*x1^2*x2*x3^2	2*x1^2*x2^2*x3	0
2*x1*x2^2*x3*x4	2*x1^2*x2*x3*x4	x1^2*x2^2*x4	x1^2*x2^2*x3
2*x1*x2^2*x4^2	2*x1^2*x2*x4^2	0	2*x1^2*x2^2*x4
2*x1*x2*x3^3	x1^2*x3^3	3*x1^2*x2*x3^2	0
2*x1*x2*x3^2*x4	x1^2*x3^2*x4	2*x1^2*x2*x3*x4	x1^2*x2*x3^2
2*x1*x2*x3*x4^2	x1^2*x3*x4^2	x1^2*x2*x4^2	2*x1^2*x2*x3*x4
2*x1*x2*x4^3	x1^2*x4^3	0	3*x1^2*x2*x4^2
2*x1*x3^4	0	4*x1^2*x3^3	0
2*x1*x3^3*x4	0	3*x1^2*x3^2*x4	x1^2*x3^3
2*x1*x3^2*x4^2	0	2*x1^2*x3*x4^2	2*x1^2*x3^2*x4
2*x1*x3*x4^3	0	x1^2*x4^3	3*x1^2*x3*x4^2
2*x1*x4^4	0	0	4*x1^2*x4^3
x2^5	5*x1*x2^4	0	0
x2^4*x3	4*x1*x2^3*x3	x1*x2^4	0
x2^4*x4	4*x1*x2^3*x4	0	x1*x2^4
x2^3*x3^2	3*x1*x2^2*x3^2	2*x1*x2^3*x3	0
x2^3*x3*x4	3*x1*x2^2*x3*x4	x1*x2^3*x4	x1*x2^3*x3
x2^3*x4^2	3*x1*x2^2*x4^2	0	2*x1*x2^3*x4
x2^2*x3^3	2*x1*x2*x3^3	3*x1*x2^2*x3^2	0
x2^2*x3^2*x4	2*x1*x2*x3^2*x4	2*x1*x2^2*x3*x4	x1*x2^2*x3^2
x2^2*x3*x4^2	2*x1*x2*x3*x4^2	x1*x2^2*x4^2	2*x1*x2^2*x3*x4

```
              x2^2*x4^3       2*x1*x2*x4^3    0               3*x1*x2^2*x4^2
              x2*x3^4         x1*x3^4         4*x1*x2*x3^3    0
              x2*x3^3*x4      x1*x3^3*x4      3*x1*x2*x3^2*x4 x1*x2*x3^3
              x2*x3^2*x4^2    x1*x3^2*x4^2    2*x1*x2*x3*x4^2 2*x1*x2*x3^2*x4
              x2*x3*x4^3      x1*x3*x4^3      x1*x2*x4^3      3*x1*x2*x3*x4^2
              x2*x4^4         x1*x4^4         0               4*x1*x2*x4^3
              x3^5            0               5*x1*x3^4       0
              x3^4*x4         0               4*x1*x3^3*x4    x1*x3^4
              x3^3*x4^2       0               3*x1*x3^2*x4^2  2*x1*x3^3*x4
              x3^2*x4^3       0               2*x1*x3*x4^3    3*x1*x3^2*x4^2
              x3*x4^4         0               x1*x4^4         4*x1*x3*x4^3
              x4^5            0               0               5*x1*x4^4
              0               6*x2^5          0               0
              0               5*x2^4*x3       x2^5            0
              0               5*x2^4*x4       0               x2^5
              0               4*x2^3*x3^2     2*x2^4*x3       0
              0               4*x2^3*x3*x4    x2^4*x4         x2^4*x3
              0               4*x2^3*x4^2     0               2*x2^4*x4
              0               3*x2^2*x3^3     3*x2^3*x3^2     0
              0               3*x2^2*x3^2*x4  2*x2^3*x3*x4    x2^3*x3^2
              0               3*x2^2*x3*x4^2  x2^3*x4^2       2*x2^3*x3*x4
              0               3*x2^2*x4^3     0               3*x2^3*x4^2
              0               2*x2*x3^4       4*x2^2*x3^3     0
              0               2*x2*x3^3*x4    3*x2^2*x3^2*x4  x2^2*x3^3
              0               2*x2*x3^2*x4^2  2*x2^2*x3*x4^2  2*x2^2*x3^2*x4
              0               2*x2*x3*x4^3    x2^2*x4^3       3*x2^2*x3*x4^2
              0               2*x2*x4^4       0               4*x2^2*x4^3
              0               x3^5            5*x2*x3^4       0
              0               x3^4*x4         4*x2*x3^3*x4    x2*x3^4
              0               x3^3*x4^2       3*x2*x3^2*x4^2  2*x2*x3^3*x4
              0               x3^2*x4^3       2*x2*x3*x4^3    3*x2*x3^2*x4^2
              0               x3*x4^4         x2*x4^4         4*x2*x3*x4^3
              0               x4^5            0               5*x2*x4^4
              0               0               6*x3^5          0
              0               0               5*x3^4*x4       x3^5
              0               0               4*x3^3*x4^2     2*x3^4*x4
              0               0               3*x3^2*x4^3     3*x3^3*x4^2
              0               0               2*x3*x4^4       4*x3^2*x4^3
              0               0               x4^5            5*x3*x4^4
              0               0               0               6*x4^5];

              EPS=0.2;
              beta_x=-x1+EPS*x4^2*sin(x3);
              gamma_x=1-EPS^2*cos(x3)^2;


 g=[  0;
      -EPS*cos(x3)/gamma_x
      0
      1/gamma_x];
    u(i)=A*tanh(1/A*-0.5*g'*dPHI'*W);
end
K=[2.41817 1.16494 -.34158 -1.08667];
%K=-[-1.3862   -0.0271    1.0000    1.8634];


u=A*tanh(K*x(:,1:4)'/A);
ylabel('control');xlabel('Time in seconds');title('No title yet');
plot(t,u,'r-.','LineWidth',2);
title('Nearly Optimal Controller');
title('Initial Controller');



figure(4);hold on;
ylabel('Attenuation');xlabel('Time in seconds');%title('No title yet');
```

```
plot(t(10:length(t)),x(10:length(t),5)./x(10:length(t),6),'r-.','LineWidth',2);
title('Nearly Optimal Controller Cost');
title('Initial Controller Cost');


%end
```

*****Simulation ODEfile*****

```
function [xdot,u]=BB(t,x);
x1=x(1);
x2=x(2);
x3=x(3);
x4=x(4);


global W;
global A;


Q=[1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
R=1;


% COMPUTE THE CONTROL INPUT U
  dPHI=[   2*x1            0               0               0
                x2              x1              0               0
                x3              0               x1              0
                x4              0               0               x1
                0               2*x2            0               0
                0               x3              x2              0
                0               x4              0               x2
                0               0               2*x3            0
                0               0               x4              x3
                0               0               0               2*x4


                4*x1^3          0               0               0
                3*x1^2*x2       x1^3            0               0
                3*x1^2*x3       0               x1^3            0
                3*x1^2*x4       0               0               x1^3
                2*x1*x2^2       2*x1^2*x2       0               0
                2*x1*x2*x3      x1^2*x3         x1^2*x2         0
                2*x1*x2*x4      x1^2*x4         0               x1^2*x2
                2*x1*x3^2       0               2*x1^2*x3       0
                2*x1*x3*x4      0               x1^2*x4         x1^2*x3
                2*x1*x4^2       0               0               2*x1^2*x4
                x2^3            3*x1*x2^2       0               0
                x2^2*x3         2*x1*x2*x3      x1*x2^2         0
                x2^2*x4         2*x1*x2*x4      0               x1*x2^2
                x2*x3^2         x1*x3^2         2*x1*x2*x3      0
                x2*x3*x4        x1*x3*x4        x1*x2*x4        x1*x2*x3
                x2*x4^2         x1*x4^2         0               2*x1*x2*x4
                x3^3            0               3*x1*x3^2       0
                x3^2*x4         0               2*x1*x3*x4      x1*x3^2
                x3*x4^2         0               x1*x4^2         2*x1*x3*x4
                x4^3            0               0               3*x1*x4^2
                0               4*x2^3          0               0
                0               3*x2^2*x3       x2^3            0
                0               3*x2^2*x4       0               x2^3
                0               2*x2*x3^2       2*x2^2*x3       0
                0               2*x2*x3*x4      x2^2*x4         x2^2*x3
                0               2*x2*x4^2       0               2*x2^2*x4
                0               x3^3            3*x2*x3^2       0
                0               x3^2*x4         2*x2*x3*x4      x2*x3^2
                0               x3*x4^2         x2*x4^2         2*x2*x3*x4
                0               x4^3            0               3*x2*x4^2
                0               0               4*x3^3          0
                0               0               3*x3^2*x4       x3^3
                0               0               2*x3*x4^2       2*x3^2*x4
                0               0               x4^3            3*x3*x4^2
                0               0               0               4*x4^3


                6*x1^5          0               0               0
```

5*x1^4*x2	x1^5	0	0
5*x1^4*x3	0	x1^5	0
5*x1^4*x4	0	0	x1^5
4*x1^3*x2^2	2*x1^4*x2	0	0
4*x1^3*x2*x3	x1^4*x3	x1^4*x2	0
4*x1^3*x2*x4	x1^4*x4	0	x1^4*x2
4*x1^3*x3^2	0	2*x1^4*x3	0
4*x1^3*x3*x4	0	x1^4*x4	x1^4*x3
4*x1^3*x4^2	0	0	2*x1^4*x4
3*x1^2*x2^3	3*x1^3*x2^2	0	0
3*x1^2*x2^2*x3	2*x1^3*x2*x3	x1^3*x2^2	0
3*x1^2*x2^2*x4	2*x1^3*x2*x4	0	x1^3*x2^2
3*x1^2*x2*x3^2	x1^3*x3^2	2*x1^3*x2*x3	0
3*x1^2*x2*x3*x4	x1^3*x3*x4	x1^3*x2*x4	x1^3*x2*x3
3*x1^2*x2*x4^2	x1^3*x4^2	0	2*x1^3*x2*x4
3*x1^2*x3^3	0	3*x1^3*x3^2	0
3*x1^2*x3^2*x4	0	2*x1^3*x3*x4	x1^3*x3^2
3*x1^2*x3*x4^2	0	x1^3*x4^2	2*x1^3*x3*x4
3*x1^2*x4^3	0	0	3*x1^3*x4^2
2*x1*x2^4	4*x1^2*x2^3	0	0
2*x1*x2^3*x3	3*x1^2*x2^2*x3	x1^2*x2^3	0
2*x1*x2^3*x4	3*x1^2*x2^2*x4	0	x1^2*x2^3
2*x1*x2^2*x3^2	2*x1^2*x2*x3^2	2*x1^2*x2^2*x3	0
2*x1*x2^2*x3*x4	2*x1^2*x2*x3*x4	x1^2*x2^2*x4	x1^2*x2^2*x3
2*x1*x2^2*x4^2	2*x1^2*x2*x4^2	0	2*x1^2*x2^2*x4
2*x1*x2*x3^3	x1^2*x3^3	3*x1^2*x2*x3^2	0
2*x1*x2*x3^2*x4	x1^2*x3^2*x4	2*x1^2*x2*x3*x4	x1^2*x2*x3^2
2*x1*x2*x3*x4^2	x1^2*x3*x4^2	x1^2*x2*x4^2	2*x1^2*x2*x3*x4
2*x1*x2*x4^3	x1^2*x4^3	0	3*x1^2*x2*x4^2
2*x1*x3^4	0	4*x1^2*x3^3	0
2*x1*x3^3*x4	0	3*x1^2*x3^2*x4	x1^2*x3^3
2*x1*x3^2*x4^2	0	2*x1^2*x3*x4^2	2*x1^2*x3^2*x4
2*x1*x3*x4^3	0	x1^2*x4^3	3*x1^2*x3*x4^2
2*x1*x4^4	0	0	4*x1^2*x4^3
x2^5	5*x1*x2^4	0	0
x2^4*x3	4*x1*x2^3*x3	x1*x2^4	0
x2^4*x4	4*x1*x2^3*x4	0	x1*x2^4
x2^3*x3^2	3*x1*x2^2*x3^2	2*x1*x2^3*x3	0
x2^3*x3*x4	3*x1*x2^2*x3*x4	x1*x2^3*x4	x1*x2^3*x3
x2^3*x4^2	3*x1*x2^2*x4^2	0	2*x1*x2^3*x4
x2^2*x3^3	2*x1*x2*x3^3	3*x1*x2^2*x3^2	0
x2^2*x3^2*x4	2*x1*x2*x3^2*x4	2*x1*x2^2*x3*x4	x1*x2^2*x3^2
x2^2*x3*x4^2	2*x1*x2*x3*x4^2	x1*x2^2*x4^2	2*x1*x2^2*x3*x4
x2^2*x4^3	2*x1*x2*x4^3	0	3*x1*x2^2*x4^2
x2*x3^4	x1*x3^4	4*x1*x2*x3^3	0
x2*x3^3*x4	x1*x3^3*x4	3*x1*x2*x3^2*x4	x1*x2*x3^3
x2*x3^2*x4^2	x1*x3^2*x4^2	2*x1*x2*x3*x4^2	2*x1*x2*x3^2*x4
x2*x3*x4^3	x1*x3*x4^3	x1*x2*x4^3	3*x1*x2*x3*x4^2
x2*x4^4	x1*x4^4	0	4*x1*x2*x4^3
x3^5	0	5*x1*x3^4	0
x3^4*x4	0	4*x1*x3^3*x4	x1*x3^4
x3^3*x4^2	0	3*x1*x3^2*x4^2	2*x1*x3^3*x4
x3^2*x4^3	0	2*x1*x3*x4^3	3*x1*x3^2*x4^2
x3*x4^4	0	x1*x4^4	4*x1*x3*x4^3
x4^5	0	0	5*x1*x4^4
0	6*x2^5	0	0
0	5*x2^4*x3	x2^5	0
0	5*x2^4*x4	0	x2^5
0	4*x2^3*x3^2	2*x2^4*x3	0
0	4*x2^3*x3*x4	x2^4*x4	x2^4*x3
0	4*x2^3*x4^2	0	2*x2^4*x4
0	3*x2^2*x3^3	3*x2^3*x3^2	0
0	3*x2^2*x3^2*x4	2*x2^3*x3*x4	x2^3*x3^2
0	3*x2^2*x3*x4^2	x2^3*x4^2	2*x2^3*x3*x4
0	3*x2^2*x4^3	0	3*x2^3*x4^2
0	2*x2*x3^4	4*x2^2*x3^3	0
0	2*x2*x3^3*x4	3*x2^2*x3^2*x4	x2^2*x3^3

```
                0               2*x2*x3^2*x4^2  2*x2^2*x3*x4^2  2*x2^2*x3^2*x4
                0               2*x2*x3*x4^3    x2^2*x4^3       3*x2^2*x3*x4^2
                0               2*x2*x4^4       0               4*x2^2*x4^3
                0               x3^5            5*x2*x3^4       0
                0               x3^4*x4         4*x2*x3^3*x4    x2*x3^4
                0               x3^3*x4^2       3*x2*x3^2*x4^2  2*x2*x3^3*x4
                0               x3^2*x4^3       2*x2*x3*x4^3    3*x2*x3^2*x4^2
                0               x3*x4^4         x2*x4^4         4*x2*x3*x4^3
                0               x4^5            0               5*x2*x4^4
                0               0               6*x3^5          0
                0               0               5*x3^4*x4       x3^5
                0               0               4*x3^3*x4^2     2*x3^4*x4
                0               0               3*x3^2*x4^3     3*x3^3*x4^2
                0               0               2*x3*x4^4       4*x3^2*x4^3
                0               0               x4^5            5*x3*x4^4
                0               0               0               6*x4^5];

% DYNAMICS
EPS=.2;
beta_x=-x1+EPS*x4^2*sin(x3);
gamma_x=1-EPS^2*cos(x3)^2;

 f = [x2
     beta_x/gamma_x
     x4
     -EPS*beta_x*cos(x3)/gamma_x];
 g=[  0;
      -EPS*cos(x3)/gamma_x
      0
      1/gamma_x];
 k=[  0;
      1/gamma_x
      0
     -EPS*cos(x3)/gamma_x];
K=[2.41817 1.16494 -.34158 -1.08667];          % Linear Hinfinity Controller
%K=-[-1.3862   -0.0271    1.0000    1.8634];

%u=A*tanh(K*x(1:4)/A);
u=A*tanh(-0.5*g'*dPHI'*W/A);
d=5*sin(t)*exp(-1*t)*1;
xdot=[f+g*u+k*d;
      x(1:4)'*Q*x(1:4)+u*R*u
      d*d]; %cost
```

REFERENCES

[1] Abu-Khalaf, M., F. L. Lewis, "Nearly optimal controls laws for nonlinear systems with saturating actuators using a neural network HJB approach," *Automatica,* Issue 5, pp. 779-791, 2005.

[2] Abu-Khalaf, M., F. L. Lewis, J. Huang, "Hamilton-Jacobi-Isaacs formulation for constrained input nonlinear systems," Proceedings of the 43rd IEEE CDC, pp. 5034-5040, Atlantis, Paradise Island, Bahamas, 2004.

[3] Abu-Khalaf, M., F. L. Lewis, J. Huang, "HJI", submitted to IEEE TAC.

[4] Abu-Khalaf, M., F. L. Lewis, J. Huang, "HJI NN", submitted to IEEE TNN.

[5] Adams, R., Fournier, J., "Sobolev Spaces," 2nd edition, Academic Press, 2003.

[6] Apostol, T., "Mathematical Analysis," Addison-Wesley, USA 1974.

[7] Astolfi, A., P. Colaneri, "A Hamilton-Jacobi Setup for the Static Output Feedback Stabilization of Nonlinear Systems," *IEEE Transactions on Automatic Control*, Vol. 47, No. 12, December 2002.

[8] Astolfi, A., P. Colaneri, "Static output feedback stabilization: from linear to nonlinear and back," *Nonlinear and Adaptive Control in 2000*, Vol 1, Springer-Verlag, New York, 2000.

[9] Ball, J., W. Helton, "Viscosity Solutions of Hamilton-Jacobi Equations Arising in Nonlinear H_∞-Control", *Journal of Mathematical Systems, Estimation, and Control*, vol. 6, no. 1, pp. 1-22, 1996.

[10] Ball, J., W. Helton, M. Walker, "H_∞ Control for Nonlinear Systems with Output Feedback", *IEEE Trans. Automat. Control*, vol. 38, no. 4, pp. 546-559, 1993.

[11] Bardi, M., I. Capuzzo-Dolcetta, "*Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations*," Birkhauser, Boston, MA, 1997.

[12] Başar, T., G. J. Olsder, *Dynamic Noncooperative Game Theory*, 2nd edition, SIAM's Classic in Applied Mathematics 23, SIAM, Philadelphia, 1999.

[13] Başar, T., P. Bernard, H_∞ *Optimal Control and Related Minimax Design Problems,* Birkhäuser, 1995.

[14] Beard, R., "Improving the Closed-Loop Performance of Nonlinear Systems," PhD thesis, Rensselaer Polytechnic Institute, Troy, NY 12180, 1995.

[15] Beard, R., G. Saridis, J. Wen, "Approximate Solutions to the Time-Invariant Hamilton-Jacobi-Bellman Equation," *Journal of Optimization Theory and Application*, Vol 96, No. 3, March 1998, pp. 589-626.

[16] Beard, R., G. Saridis, J. Wen, "Galerkin Approximations of the Generalized Hamilton-Jacobi-Bellman Equation," *Automatica* 33:12, December, pp. 2159-2177, 1997.

[17] Beard, R., T. McLain, (1998). Successive Galerkin approximation algorithms for nonlinear optimal and robust control. *International Journal of Control*, vol 71. no. 5, pp 717-743.

[18] Bernstein, D. S., "Optimal nonlinear, but continuous, feedback control of systems with saturating actuators," *International Journal of Control,* vol 62, NO. 5, pp. 1209-1216, 1995.

[19] Bertsekas, D. P., J. N. Tsitsiklis, "*Neuro-Dynamic Programming,*" Athena Scientific, Belmont, MA, 1996.

[20] Bianchini, G., R. Genesio, A. Parentri, A. Tesi, "Global H_∞ Controllers for a Class of Nonlinear Systems", *IEEE Trans. Automat. Control*, vol. 49 no. 2, pp. 244-249, 2004.

[21] Bitsoris, G., E. Gravalou. Design Techniques for the Control of Discrete-Time Systems Subject to State and Control Constraints. *IEEE Trans. Automat. Control*, vol. 44, no. 5, pp. 1057-1061, 1999.

[22] Bupp R., D. Bernstein, V. Coppola, "A benchmark problem for nonlinear control design," International Journal of Robust and Nonlinear Control, vol 8, 307-310, 1998.

[23] Burk, F., "Lebesgue Measure and Integration," John Wiley & Sons, New York, NY, 1998.

[24] Chen, F.-C., C.-C. Liu, ``Adaptively controlling nonlinear continuous-time systems using multilayer neural networks," *IEEE Trans. Automat. Control*, vol. 39, no. 6, pp. 1306-1310, June 1994.

[25] Deng, F., J. Huang, "Computer-aided design of nonlinear H_∞ control law: The benchmark problem," Proceeding of 2001 Chinese Control Conference, Dalin, China, 840-845, 2001.

[26] Doyle, J. H., K. Glover, P. Khargonekar, B. Francis, "State-Space Solutions to Standard H_2 and H_∞ Control Problems," *IEEE Trans. Automat. Control*, vol. 34, no. 8, pp. 831-847, 1989.

[27] Evans, M., Swartz, T., “Approximating Integrals Via Monte Carlo and Deterministic Methods,” Oxford University Press, 2000.

[28] Finlayson, B. A., "The Method of Weighted Residuals and Variational Principles,” Academic Press, New York, NY, 1972.

[29] Ge, S. S., C. C Hang, T. H. Lee, T. Zhang, *Stable Adaptive Neural Network Control*, Asian Studies in Computer and Information Science, Kluwer Academic Publishers, MA, 2002.

[30] Genesio, R., M. Tartaglia (1985). On the Estimation of Asymptotic Stability Regions: State of the Art and New Proposals. *IEEE Trans. Automat. Control*, vol. 30, no. 8, pp. 747-755.

[31] Gilbert, E., K. T. Tan. Linear Systems with State and Control Constraints: The Theory and Application of Maximal Output Admissible Sets. *IEEE Trans. Automat. Control*, vol. 36, no. 9, pp. 1008-1020, 1991.

[32] Han, D., S. N. Balakrishnan, "State-Constrained Agile Missile Control with Adaptive-Critic Based Neural Networks,” *Proc. American Control Conference*, June. 2000, pp.1929 – 1933.

[33] Henrion, D., S. Tarbouriech, V. Kucera. Control of Linear Systems Subject to Input Constraints: A polynomial Approach. *Automatica*, vol. 37, no.4, pp.597-604, 2001.

[34] Hill, D., P. Moylan, “The Stability of Nonlinear Dissipative Systems,” *IEEE Trans. Automatic Control*, vol. 21, pp. 708-711, 1976.

[35] Hornik, K., M. Stinchcombe, H. White, "Universal Approximation of an Unknown Mapping and Its Derivatives Using Multilayer Feedforward Networks," *Neural Networks,* vol. 3, pp. 551-560, 1990.

[36] Hu, T., Z. Lin, B. M. Chen, "An analysis and design method for linear systems subject to actuator saturation and disturbance," *Automatica*, vol. 38, no. 2, pp. 351-359, 2002.

[37] Huang, C.-S., S. Wang, K. L. Teo, "Solving Hamilton-Jacobi-Bellman equations by a Modified Method of Characteristics," *Nonlinear Analysis,* vol. 40, pp. 279-293, 2000.

[38] Huang, J., C. F. Lin, "Numerical Approach to Computing Nonlinear H_∞ Control Laws," *Journal of Guidance, Control, and Dynamics,* vol. 18, no. 5, pp. 989-994, September-October 1995.

[39] Isidori, A., A. Astolfi, "Disturbance Attenuation and H_∞-Control via Measurement Feedback in Nonlinear Systems," *IEEE Trans. Automat. Control*, vol. 37, no. 9, pp. 1283-1293, 1992.

[40] J. Gadewadikar, F. Lewis, M. Abu-Khalaf, "Necessary and sufficient conditions for H-infinity static output-feedback control," *Journal of Guidance, Control, and Dynamics,* (Accepted).

[41] Khalil, H., "*Nonlinear Systems*", 3rd Edition, Prentice Hall, Upper Saddle River, NJ, 2003.

[42] Kim, Y. H., F. L. Lewis, D. Dawson, "Intelligent optimal control of robotic manipulators using neural networks," *Automatic 36*, 2000, pp. 1355 – 1364.

[43] Kirk, D. *Optimal Control Theory: An Introduction.* Prentice Hall, New Jersey, 1970.

[44] Kleinman, D., "*On an iterative Technique for Riccati Equation Computations,*" *IEEE Trans. Automatic Control*, pp. 114-115, February 1968.

[45] Knobloch, H., Isidori, A., Flockerzi, D., *Topics in Control Theory,* Springer Verlag, Boston, 1993.

[46] Lancaster, P., L. Rodman, *Algebraic Riccati Equations*, Oxford University Press Inc., New York, 1995.

[47] Landelius, T., Reinforcement Learning and Distributed Local Model Synthesis. PhD thesis, LinkSping University, 1997.

[48] Lee, H. W. J., K. L. Teo, W. R. Lee, S. Wang, "Construction of Suboptimal Feedback Control for Chaotic Systems Using B-Splines with Optimally Chosen Knot Points," *International Journal of Bifurcation and Chaos,* Vol. 11, No. 9, pp. 2375-2387, 2001.

[49] Lewis, F. L., S. Jagannathan, A. Yesildirek, "Neural Network Control of Robot Manipulators and Nonlinear Systems," Taylor & Francis, London, 1999.

[50] Lewis, F. L., V. L. Syrmos, "*Optimal Control,*" John Wiley & Sons, Inc. New York, NY, 1995.

[51] Lio, F. D., "On the Bellman Equation for Infinite Horizon Problems with Unbounded Cost Functional," *Applied Mathematics and Optimization*, vol 41, pp.171-197, 2000.

[52] Liu, X., S. N. Balakrishnan, "Adaptive Critic Based Neuro-Observer," *Proc. American Control Conference*, June. 2001, pp.1616 – 1621.

[53] Liu, X., S. N. Balakrishnan, "Convergence Analysis of Adaptive Critic Based Optimal Control," *Proc. American Control Conference*, June. 2000, pp.1929 – 1933.

[54] Lyshevski, S. E., "Role of Performance Functionals in Control Laws Design." *Proc. American Control Conference,* pp. 2400 – 2405, June. 2001.

[55] Lyshevski, S. E., "Constrained Optimization and Control of Nonlinear Systems: New Results in Optimal Control," *Proc. IEEE Conference on Decision and Control*, December. 1996, pp. 541 - 546.

[56] Lyshevski, S. E., "*Control Systems Theory with Engineering Applications,*" Birkhauser, Boston, MA,2001.

[57] Lyshevski, S. E., "Optimal Control of Nonlinear Continuous-Time Systems: Design of Bounded Controllers Via Generalized Nonquadratic Functionals," *Proc. American Control Conference*, June. 1998, pp.205 – 209.

[58] Lyshevski, S. E., A. U. Meyer, "Control System Analysis and Design Upon the Lyapunov Method," *Proc. American Control Conference*, June. 1995, pp. 3219 - 3223.

[59] Mikhlin, S. G., "Variational Methods in Mathematical Physics," Pergamon, Oxford, 1964.

[60] Miller, W. T., R. Sutton, P. Werbos, "Neural Networks for Control," The MIT Press, Cambridge, Massachusetts, 1990.

[61] Mracek, C. J. Cloutier, “A preliminary control design for the nonlinear benchmark problem,” Proceedings of the 1996 International Conference on Control Applications, Dearborn, MI, 265-272, 1996.

[62] Munos, R., L. C. Baird, A. Moore, "Gradient Descent Approaches to Neural-Net-Based Solutions of the Hamilton-Jacobi-Bellman Equation,” *International Joint Conference on Neural Networks IJCNN*, 1999, Vol 3, pp. 2152 -- 2157.

[63] Murray, J., C. Cox, R. Saeks and G. Lendaris, “Globally Convergent Approximate Dynamic Programming Applied to an Autolander," *Proc. American Control Conference*, June. 2001, pp.2901 – 2906.

[64] Narendra, K. S., F. L. Lewis, ed. “Special issue on neural network feedback control,” *Automatica* vol.37, no. 8, August 2001, pp. 1147-1148.

[65] Naylor, A. W., Sell, G. R., *Linear Operator Theory in Engineering and Science*, Holt, Rinehart and Winston, INC, 1971.

[66] Parisini, T., R. Zoppoli, “Neural Approximations for Infinite-Horizon Optimaol Control of Nonlinear Stochastic Systems,” *IEEE Trans Neural Net.* vol. 9, no. 6, November 1998, pp. 1388-1408.

[67] Polycarpou, M., ``Stable adaptive neural control scheme for nonlinear systems,” IEEE *Trans. Automat. Control*, vol. 41, no. 3, pp. 447-451, Mar. 1996.

[68] Rovithakis, G.A., and M.A. Christodoulou, ``Adaptive control of unknown plants using dynamical neural networks," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 24, no. 3, pp. 400-412, 1994.

[69] Saberi, A., Z. Lin, A. Teel, "Control of Linear Systems with Saturating Actuators," *IEEE Transactions on Automatic Control,* Vol 41, NO. 3, March 1996, pp. 368 -378.

[70] Sadegh, N., ``A perceptron network for functional identification and control of nonlinear systems," *IEEE Trans. Neural Networks*, vol. 4, no. 6, pp. 982-988, Nov. 1993.

[71] Sanner, R.M., and J.-J.E. Slotine, ``Stable adaptive control and recursive identification using radial gaussian networks," *Proc. IEEE Conf. Decision and Control*, pp. 2116-2123, Brighton, 1991.

[72] Saridis, G., C. S. Lee, "An Approximation Theory of optimal Control for Trainable Manipulators," *IEEE Trans. Systems, Man, Cybernetics*, Vol. 9, No. 3, pp. 152-159, March 1979.

[73] Seshagiri S., H K. Khalil, "Output feedback control of nonlinear systems using RBF neural networks," *IEEE Trans. Neural Networks*, vol. 11, pp. 69-79, Jan. 2000.

[74] Sussmann, H. , E. D. Sontag, Y. Yang, "A General Result on the Stabilization of Linear Systems Using Bounded Controls," *IEEE Trans. Automatic Control*, Vol. 39, No. 12, pp. 2411-2425, December 1994.

[75] Tsiotras P., M. Corless, M. Rotea, "An L_2 disturbance attenuations solution to the nonlinear benchmark problem," International Journal of Robust and Nonlinear Control, vol 8, 311-330, 1998.

[76] Van Der Schaft, A. J., "L_2-gain Analysis of Nonlinear Systems and Nonlinear State Feedback H_∞ Control," *IEEE Trans. Automat. Control*, vol. 37 no. 6, pp. 770-784, 1992.

[77] Van Der Schaft, A. J., L_2-Gain and Passivity Techniques in Nonlinear Control. London, U.K.: Springer-Verlag, 1999.

[78] Vinter, R., J. Clark, M. James, "The Interpretation of Discontinuous State Feedback Control Laws as Nonanticipative Control Strategies in Differential Games," *IEEE Trans. Automat. Control*, vol. 49, no. 8, pp. 1360-1365, 2004.

[79] Willems, J. C. ,"Dissipative Dynamical Systems Part II: Linear Systems with Quadratic Supplies," Archive for Rational Mechanics and Analysis, vol 45, no.1, pp. 352-393, 1972.

[80] Willems, J. C., "Dissipative Dynamical Systems Part I: General Theory," Archive for Rational Mechanics and Analysis, vol 45, no.1, pp. 321-351, 1972.

[81] Zames, G., "Feedback and Optimal Sensitivity: Model Reference Transformations, Multiplicative Seminorms, and Approximate Inverses," *IEEE Trans. Automat. Control*, vol. 26 no. 2, pp. 301-320, 1981.

BIOGRAPHICAL INFORMATION

Murad Abu-Khalaf was born in Jerusalem, Palestine in 1977. He did his high school studies at Ibrahimiah College in Jerusalem. He received his Bachelor's Degree in Electronics and Electrical Engineering from Boğaziçi University in Istanbul, Turkey in 1998. He then joined The University of Texas at Arlington from which he received the M.Sc and Ph.D. in Electrical Engineering in 2000 and 2005 respectively and served as a research assistant at the Automation and Robotics Research Institute of The University of Texas at Arlington in the same period. He is an IEEE member, and a member of Etta Kappa Nu honor society. His interest is in the areas of nonlinear control, optimal control, neural network control.