CHAPTER 19

NEURAL NETWORKS IN FEEDBACK CONTROL SYSTEMS

F. L. Lewis
Automation and Robotics Research Institute
University of Texas at Arlington
Fort Worth, Texas

Shuzhi Sam Ge
Department of Electrical and Computer Engineering
National University of Singapore
Singapore

1 INTRODUCTION **792**

2 BACKGROUND **793**
2.1 Neural Networks 793
2.2 NN Control Topologies 794

3 FEEDBACK LINEARIZATION DESIGN OF NN TRACKING CONTROLLERS **795**
3.1 Multilayer NN Controller 796
3.2 Single-Layer NN Controller 798
3.3 Feedback Linearization of Nonlinear Systems Using NNs 798
3.4 Partitioned NNs and Input Preprocessing 799

4 NN CONTROL FOR DISCRETE-TIME SYSTEMS **800**

5 MULTILOOP NN FEEDBACK CONTROL STRUCTURES **800**
5.1 Backstepping Neurocontroller for Electrically Driven Robot 801
5.2 Compensation of Flexible Modes and High-Frequency Dynamics Using NNs 802
5.3 Force Control with Neural Nets 803

6 FEEDFORWARD CONTROL STRUCTURES FOR ACTUATOR COMPENSATION **804**
6.1 Feedforward Neurocontroller for Systems with Unknown Deadzone 804
6.2 Dynamic Inversion Neurocontroller for Systems with Backlash 805

7 NN OBSERVERS FOR OUTPUT FEEDBACK CONTROL **806**

8 REINFORCEMENT LEARNING CONTROL USING NNs **807**
8.1 NN Reinforcement Learning Controller 808
8.2 Adaptive Reinforcement Learning Using Fuzzy Logic Critic 809

9 OPTIMAL CONTROL USING NNs **810**
9.1 NN H_2 Control Using the Hamilton–Jacobi–Bellman Equation 811
9.2 NN H_∞ Control Using the Hamilton–Jacobi–Isaacs Equation 813

10 APPROXIMATE DYNAMIC PROGRAMMING AND ADAPTIVE CRITICS **815**

11 HISTORICAL DEVELOPMENT, REFERENCED WORK, AND FURTHER STUDY **817**
11.1 NN for Feedback Control 817
11.2 Approximate Dynamic Programming 819

REFERENCES **821**

BIBLIOGRAPHY **825**

Copyrighted Material

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

1 INTRODUCTION

Dynamical systems are ubiquitous in nature and include naturally occurring systems such as the cell and more complex biological organisms, the interactions of populations, and so on, as well as man-made systems such as aircraft, satellites, and interacting global economies. Von Bertalanffy[1] were among the first to provide a modern theory of systems at the beginning of the century. Systems are characterized as having outputs that can be measured, inputs that can be manipulated, and internal dynamics. *Feedback control* involves computing suitable control inputs, based on the difference between observed and desired behavior, for a dynamical system such that the observed behavior coincides with a desired behavior prescribed by the user. All biological systems are based on feedback for survival, with even the simplest of cells using chemical diffusion based on feedback to create a potential difference across the membrane to maintain its *homeostasis,* or required equilibrium condition for survival. Volterra was the first to show that feedback is responsible for the balance of two populations of fish in a pond, and Darwin showed that feedback over extended time periods provides the subtle pressures that cause the evolution of species.

There is a large and well-established body of design and analysis techniques for feedback control systems which has been responsible for successes in the industrial revolution, ship and aircraft design, and the space age. Design approaches include classical design methods for linear systems, multivariable control, nonlinear control, optimal control, robust control, H_∞ control, adaptive control, and others. Many systems one desires to control have unknown dynamics, modeling errors, and various sorts of disturbances, uncertainties, and noise. This, coupled with the increasing complexity of today's dynamical systems, creates a need for advanced control design techniques that overcome limitations on traditional feedback control techniques.

In recent years, there has been a great deal of effort to design feedback control systems that mimic the functions of living biological systems. There has been great interest recently in ''universal model-free controllers'' that do not need a mathematical model of the controlled plant but mimic the functions of biological processes to learn about the systems they are controlling online, so that performance improves automatically. Techniques include fuzzy logic control, which mimics linguistic and reasoning functions, and artificial neural networks (NNs), which are based on biological neuronal structures of interconnected nodes, as shown in Fig. 1. By now, the theory and applications of these nonlinear network structures in feedback control have been well documented. It is generally understood that NNs provide an elegant extension of adaptive control techniques to nonlinearly parameterized learning systems.

This chapter shows how NNs fulfill the promise of providing *model-free learning controllers* for a class of nonlinear systems, in the sense that a structural or parameterized model of the system dynamics is not needed. The control structures discussed are *multiloop controllers* with NNs in some of the loops and an outer tracking unity-gain feedback loop. Throughout, there are repeatable design algorithms and guarantees of system performance, including both small tracking errors and bounded NN weights. It is shown that as uncertainty about the controlled system increases or as one desires to consider human user inputs at higher levels of abstraction, the NN controllers acquire more and more structure, eventually acquiring a hierarchical structure that resembles some of the elegant architectures proposed by computer science engineers using high-level design approaches based on cognitive linguistics, reinforcement learning, psychological theories, adaptive critics, or optimal dynamic programming techniques.

Many researchers have contributed to the development of a firm foundation for analysis and design of NNs in control system applications. See Section 11 on historical development and further study.

Copyrighted Material

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

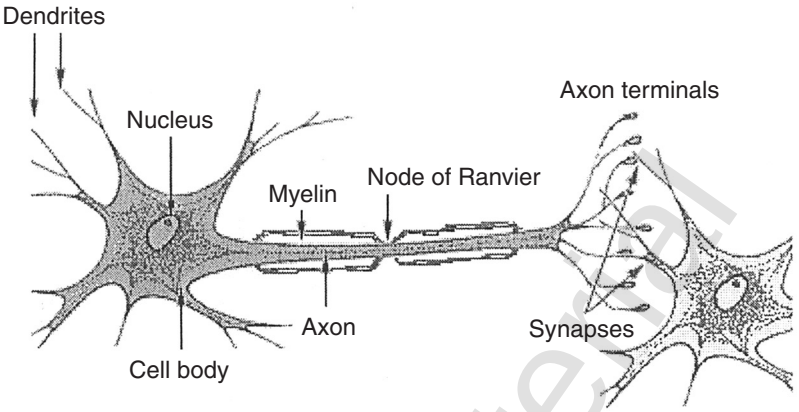


Figure 1 Nervous system cell. (With permission from http://www.sirinet.net/~jgjohnso/index.html.)

2 BACKGROUND

2.1 Neural Networks

The multilayer NN is modeled based on the structure of biological nervous systems (see Fig. 1) and provides a nonlinear mapping from an input space $\mathcal{R}^n$ into an output space $\mathcal{R}^m$. Its properties include function approximation, learning, generalization, classification, and so on. It is known that the two-layer NN has sufficient generality for closed-loop control purposes. The two-layer NN shown in Fig. 2 consists of two layers of weights and thresholds and has a hidden layer and an output layer. The input function $x(t)$ has n components, the hidden layer has L neurons, and the output layer has m neurons.

One may describe the NN mathematically as

$$y = W^{\mathrm{T}}\sigma(V^{\mathrm{T}}x)$$

where V is a matrix of first-layer weights and W is a matrix of second-layer weights. The second-layer thresholds are included as the first column of the matrix W^{T} by augmenting the vector activation function $\sigma(\cdot)$ by 1 in the first position. Similarly, the first-layer thresholds are included as the first column of the matrix V^{T} by augmenting vector x by 1 in the first position.

The main property of NNs we are concerned with for control and estimation purposes is the *function approximation property.*[2,3] Let $f(x)$ be a smooth function from $\mathcal{R}^n \to \mathcal{R}^m$. Then, it can be shown that if the activation functions are suitably selected and is restricted to a compact set $S \in \mathcal{R}^n$, then for some sufficiently large number L of hidden-layer neurons, there exist weights and thresholds such that one has

$$f(x) = W^{\mathrm{T}}\sigma(V^{\mathrm{T}}x) + \varepsilon(x)$$

with $\varepsilon(x)$ suitably small. Here, $\varepsilon(x)$ is called *the neural network functional approximation error*. In fact, for any choice of a positive number ε_N, one can find a NN of large enough size L such that $\varepsilon(x) \leq \varepsilon_N$ for all $x \in S$.

Finding a suitable NN for approximation involves adjusting the parameters V and W to obtain a good fit to $f(x)$. Note that tuning of the weights includes tuning of the thresholds as well. The neural net is *nonlinear in the parameters V,* which makes adjustment of these parameters difficult and was initially one of the major hurdles to be overcome in closed-

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

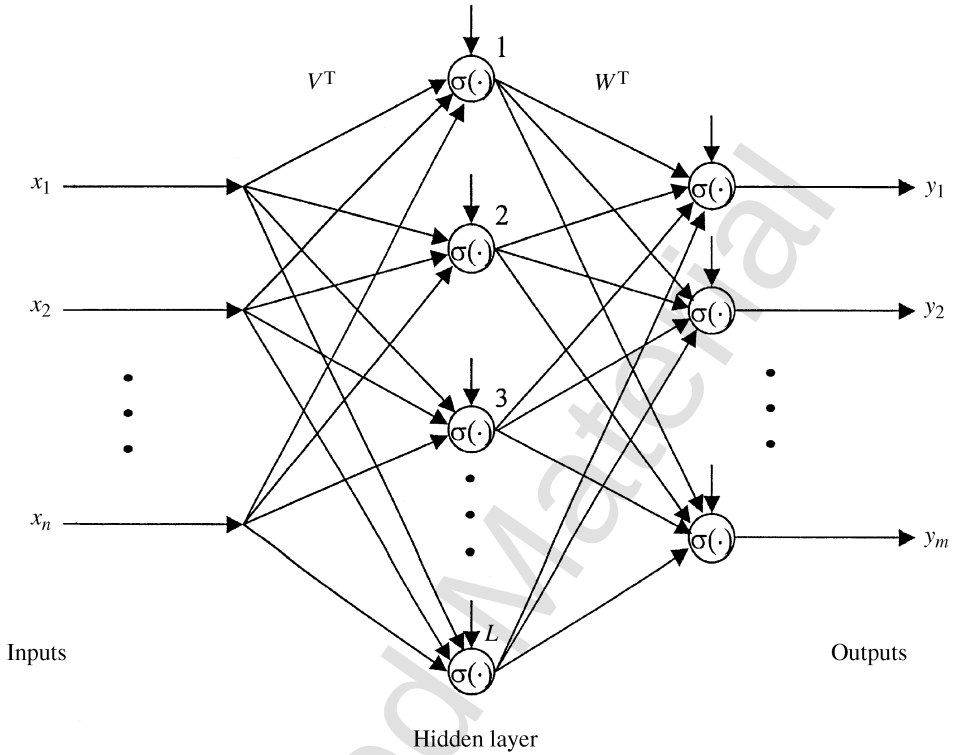


Figure 2 Two-layer NN.

loop feedback control applications. If the first-layer weights V are fixed, then the NN is linear in the adjustable parameters W (LIP). It has been shown that, if the first-layer weights V are suitably fixed, then the approximation property can be satisfied by selecting only the output weights W for good approximation. For this to occur, $\sigma(V^T x)$ must provide a *basis*. It is not always straightforward to pick a basis $\sigma(V^T x)$. It has been shown that the cerebellar model articulation controller (CMAC),[4] radial basis function (RBF),[5] fuzzy logic,[6] and other structured NN approaches allow one to choose a basis by suitably partitioning the compact set S. However, this can be tedious. If one selects the activation functions suitably (e.g., as sigmoids), then it was shown in Ref. 7 that $\sigma(V^T x)$ is almost always a basis if is selected randomly.

2.2 NN Control Topologies

Feedback control involves the measurement of output signals from a dynamical system or *plant* and the use of the *difference* between the measured values and certain prescribed *desired values* to compute system inputs that cause the measured values to follow, or *track,* the desired values. In feedback control design it is crucial to guarantee by rigorous means both the tracking performance and the internal stability or boundedness of all variables. Failure to do so can cause serious problems in the closed-loop system, including instability and unboundedness of signals that can result in system failure or destruction.

The use of NNs in control systems was first proposed by Werbos[8] and Narendra and Parthasarathy.[9] NN control has had two major thrusts: approximate dynamic programming,

Copyrighted Material

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

which uses NNs to approximately solve the optimal control problem, and NNs in closed-loop feedback control. Many researchers have contributed to the development of these fields. See Section 11 and the References and Bibliography.

Several NN feedback control topologies are illustrated in Fig. 3,[10] some of which are derived from standard topologies in adaptive control.[11] Solid lines denote control signal flow loops while dashed lines denote tuning loops. There are basically two sorts of feedback control topologies: indirect and direct techniques. In *indirect* NN control there are two functions; in an identifier block, the NN is tuned to learn the dynamics of the unknown plant, and the controller block then uses this information to control the plant. *Direct* control is more efficient and involves directly tuning the parameters of an adjustable NN controller.

The challenge in using NNs for feedback control purposes is to select a suitable control system structure and then to demonstrate using mathematically acceptable techniques how the NN weights can be tuned so that closed-loop stability and performance are guaranteed. In this chapter, we shall show different methods of NN controller design that yield guaranteed performance for systems of different structure and complexity. Many researchers have participated in the development of the theoretical foundation for NNs in control applications. See Section 11.

3 FEEDBACK LINEARIZATION DESIGN OF NN TRACKING CONTROLLERS

In this section, the objective is to design an NN feedback controller that causes a robotic system to follow, or track, a prescribed trajectory or path. The dynamics of the robot are unknown, and there are unknown disturbances. The dynamics of an n-link robot manipulator may be expressed as[12]

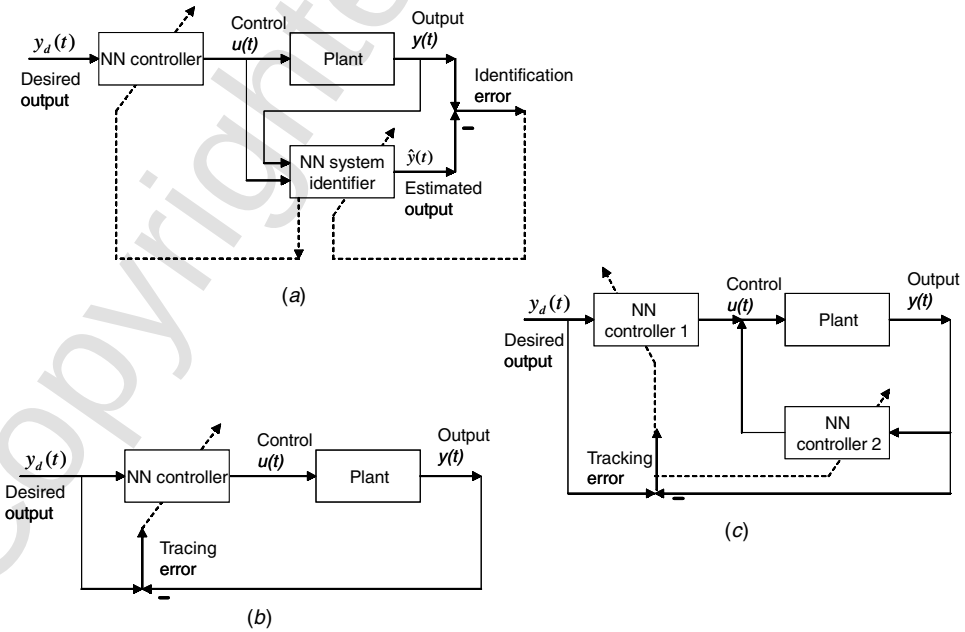


Figure 3 NN control topologies: (*a*) indirect scheme; (*b*) direct scheme; (*c*) feedback/feedforward scheme.

Copyrighted Material

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

$$M(q)\ddot{q} + V_m(q,\dot{q})(\dot{q} + G(q) + F(\dot{q}) + \tau_d = \tau \tag{1}$$

with $q(t) \in R^n$ the joint variable vector, $M(q)$ an inertia matrix, V_m a centripetal/Coriolis matrix, $G(q)$ a gravity vector, and $F(\cdot)$ representing friction terms. Bounded unknown disturbances and modeling errors are denoted by τ_d and the control input torque is $\tau(t)$.

The sliding-mode control approach of Slotine[13,14] can be generalized to NN control systems. Given a desired arm trajectory $q_d(t) \in R^n$, define the tracking error $e(t) = q_d(t) - q(t)$ and the sliding variable error $r = \dot{e} + \Lambda e$, where $\Lambda = \Lambda^T > 0$. A *sliding-mode manifold* is defined by $r(t) = 0$. The NN tracking controller is designed using a feedback linearization approach to guarantee that $r(t)$ is forced into a neighborhood of this manifold. Define the nonlinear robot function

$$f(x) = M(q)(\ddot{q}_d + \Lambda\dot{e}) + V_m(q,\dot{q})(\dot{q}_d + \Lambda e) + G(q) + F(\dot{q}) \tag{2}$$

with the known vector $x(t)$ of measured signals suitably defined in terms of $e(t)$, $q_d(t)$. The NN input vector x can be selected, for instance, as

$$x = [e^T \quad \dot{e}^T \quad q_d^T \quad \dot{q}_d^T \quad \ddot{q}_d^T]^T \tag{3}$$

3.1 Multilayer NN Controller

A NN controller may be designed based on the *functional approximation properties* of NNs, as shown in Ref. 15. Thus, assume that $f(x)$ is unknown and given approximately as the output of a NN with unknown "ideal" weights W, V so that $f(x) = W^T\sigma(V^Tx) + \varepsilon$ with ε an approximation error. The key is now to approximate $f(x)$ by the NN functional estimate $\hat{f}(x) = \hat{W}^T\sigma(\hat{V}^Tx)$, with $\hat{V}$, $\hat{W}$ the current (estimated) NN weights as provided by the tuning algorithms. This is *nonlinear in the tunable parameters* $\hat{V}$. Standard adaptive control approaches only allow LIP controllers.

Now select the control input

$$\tau = \hat{W}^T\sigma(\hat{V}^Tx) + K_vr - v \tag{4}$$

with K_v a symmetric positive-definite (PD) gain and $v(t)$ a certain robustifying function detailed in Ref. 15. This NN control structure is shown in Fig. 4. The outer PD tracking loop guarantees robust behavior. The inner loop containing the NN is known as a *feedback linearization loop,*[16] and the NN effectively learns the unknown dynamics online to cancel the nonlinearities of the system.

Let the estimated sigmoid Jacobian be $\hat{\sigma}' \equiv d\sigma(z)/dz|_{z=\hat{V}^Tx}$. Note that this jacobian is *easily computed in terms of the current NN weights*. Then, the next result is representative of the sort of theorems that occur in NN feedback control design. It shows how to tune or train the NN weights to obtain guaranteed closed-loop stability.

Theorem (*NN Weight Tuning for Stability*) Let the desired trajectory $q_d(t)$ and its derivatives be bounded. Take the control input for (1) as (4). Let NN weight tuning be provided by

$$\dot{\hat{W}} = F\hat{\sigma}r^T - F\hat{\sigma}'\hat{V}^Txr^T - \kappa F\|r\|\hat{W} \qquad \dot{\hat{V}} = Gx(\hat{\sigma}'^T\hat{W}r)^T - \kappa G\|r\|\hat{V} \tag{5}$$

with any constant matrices $F = F^T > 0$, $G = G^T > 0$, and scalar tuning parameter $\kappa > 0$. Initialize the weight estimates as $\hat{W} = 0$, $\hat{V}$ = random. Then the sliding error $r(t)$ and NN weight estimates $\hat{W}$, $\hat{V}$ are uniformly ultimately bounded.

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

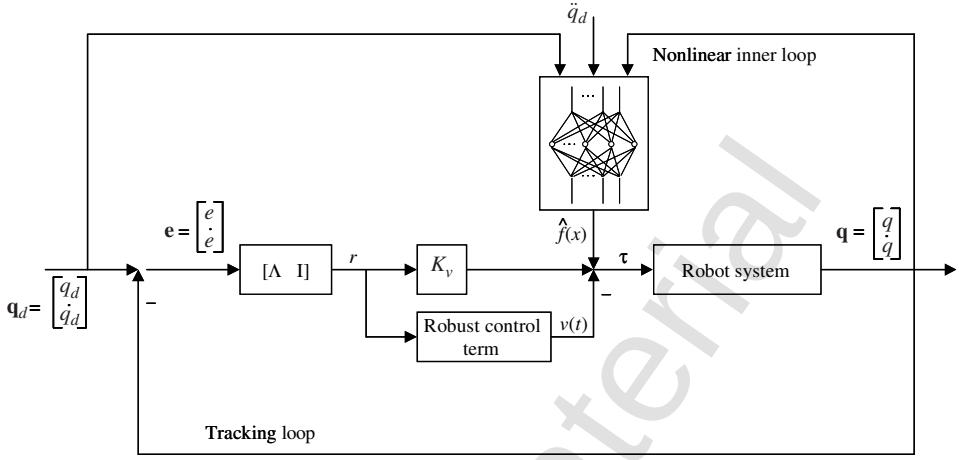


Figure 4 NN robot controller.

A proof of stability is always needed in control systems design to guarantee performance. Here, the stability is proven using nonlinear stability theory (e.g., an extension of Lyapunov's theorem). A Lyapunov energy function is defined as

$$L = \tfrac{1}{2} r^{\mathrm{T}} M(q) r + \tfrac{1}{2} \operatorname{tr}\{\tilde{W}^{\mathrm{T}} F^{-1} \tilde{W}\} + \tfrac{1}{2} \operatorname{tr}\{\tilde{V}^{\mathrm{T}} F^{-1} \tilde{V}\}$$

where the weight estimation errors are $\tilde{V} = V - \hat{V}$, $\tilde{W} = W - \hat{W}$, with $\operatorname{tr}\{\cdot\}$ the trace operator so that the Frobenius norm of the weight errors is used. In the proof, it is shown that the Lyapunov function derivative is negative outside a compact set. This guarantees the boundedness of the sliding variable error $r(t)$ as well as the NN weights. Specific bounds on $r(t)$ and the NN weights are given in Ref. 15. The first terms of (4) are very close to the (continuous-time) backpropagation algorithm.[17] The last terms correspond to Narendra's e-modification[18] extended to nonlinear-in-the-parameters adaptive control.

Robust adaptive tuning methods for nonlinear-in-the-parameters NN controllers have been derived based on the adaptive control approaches of e-modification, Ioannou's σ-modification, or projection methods. These techniques are compared by Ioannou and Sun[19] for standard adaptive control systems.

Robustness and Passivity of the NN When Tuned Online

Though the NN in Fig. 4 is static, since it is tuned online, it becomes a dynamic system with its own internal states (e.g., the weights). It can be shown that the tuning algorithms given in the theorem make the NN *strictly passive* in a certain novel strong sense known as "state-strict passivity," so that the energy in the internal states is bounded above by the power delivered to the system. This makes the closed-loop system *robust* to bounded unknown disturbances. This strict passivity accounts for the fact that no persistence of excitation condition is needed.

Standard adaptive control approaches assume that the unknown function $f(x)$ is linear in the unknown parameters and a certain regression matrix must be computed. By contrast, the NN design approach allows for nonlinearity in the parameters, and in effect the NN learns its own basis set online to approximate the unknown function $f(x)$. It is not required

Copyrighted Material

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

to find a regression matrix. This is a consequence of the NN universal approximation property.

3.2 Single-Layer NN Controller

If the first-layer weights V are fixed so that $\hat{f}(x) = \hat{W}^{\mathrm{T}}\sigma(V^{\mathrm{T}}x) \equiv \hat{W}^{\mathrm{T}}\phi(x)$, with $\phi(x)$ selected as a basis, then one has the simplified tuning algorithm for the output layer weights given by

$$\dot{\hat{W}} = F\phi(x)r^{\mathrm{T}} - \kappa F\|r\|\hat{W}$$

Then, the NN is LIP and the tuning algorithm resembles those used in adaptive control. However, NN design still offers an advantage in that the NN provides a universal basis for a class of systems, while adaptive control requires one to find a regression matrix, which serves as a basis for each particular system.

3.3 Feedback Linearization of Nonlinear Systems Using NNs

Many systems of interest in industrial, aerospace, and U.S. Department of Defense (DoD) applications are in the affine form $\dot{x} = f(x) + g(x)u + d$, with $d(t)$ a bounded unknown disturbance, nonlinear functions $f(x)$ unknown, and $g(x)$ unknown but bounded below by a known positive value g_b. Using nonlinear stability proof techniques such as those above, one can design a control input of the form

$$u = \frac{-\hat{f}(x) + v}{\hat{g}(x)} + u_r \equiv u_c + u_r$$

that has two parts, a *feedback linearization* part $u_c(t)$ and an *extra robustifying part* $u_r(t)$. Now, *two NNs* are required to manufacture the two estimates $\hat{f}(x)$, $\hat{g}(x)$ of the unknown functions. This controller is shown in Fig. 5. The weight updates for the $\hat{f}(\mathbf{x})$ NN are given exactly as in (5). To tune the $\hat{g}$ NN, a formula similar to (5) is needed, but it must be modified to ensure that the output $\hat{g}(x)$ of the second NN is bounded away from zero, to

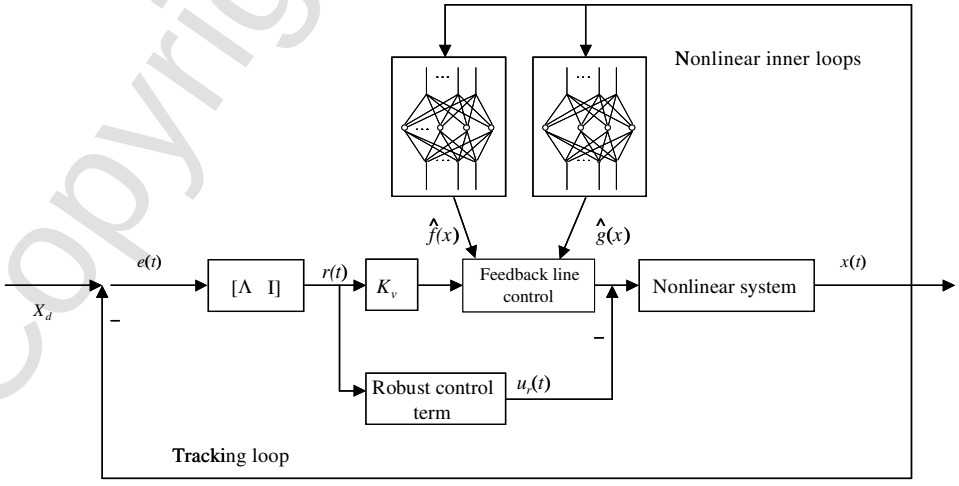


Figure 5 Feedback linearization NN controller.

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

keep the control $u(t)$ finite. It is called a controller singularity problem if $u(t)$ becomes infinity. More advanced control is possible using novel techniques. One good example is the use of integral Lyapunov functions in Refs. 20 and 21.

3.4 Partitioned NNs and Input Preprocessing

In this section we show how NN controller implementation may be streamlined by partitioning the NN into several smaller subnets to obtain more efficient computation. Also discussed in this section is preprocessing of input signals for the NN to improve the efficiency and accuracy of the approximation.

Partitioned NNs

A major advantage of the NN approach is that it allows one to partition the controller in terms of partitioned NN or neural subnets. This (i) simplifies the design, (ii) gives added controller structure, and (iii) makes for faster weight-tuning algorithms.

The unknown nonlinear robot function (2) can be written as

$$f(x) = M(q)\zeta_1(x) + V_m(q,\dot{q})\zeta_2(x) + G(q) + F(\dot{q})$$

with $\zeta_1(x) = \ddot{q}_d + \Lambda\dot{e}$, $\zeta_2(x) = \dot{q}_d + \Lambda e$. Taking the four terms one at a time,[22] one can use a small NN to approximate each term, as depicted in Fig. 6. This procedure results in four neural subnets, which we term a *structured or partitioned NN*. This approach can also utilize the properties of the physical systems conveniently for control system design and implementation. It can be directly shown that the individual partitioned NNs can be separately tuned exactly as in (5), making for a faster weight update procedure.

An advantage of this structured NN is that if some terms in the robot dynamics are well known [e.g., inertia matrix $M(q)$ and gravity $G(q)$], then their NNs can be replaced by equations that explicitly compute these terms. NNs can be used to reconstruct only the unknown terms or those too complicated to compute, which will probably include the friction $F(\dot{q})$ and the Coriolis/centripetal terms $V_m(q,\dot{q})$.

Preprocessing of Neural Net Inputs

The selection of a suitable NN input vector $x(t)$ for computation should be addressed. Some preprocessing of signals yields a more advantageous choice than (3) since it can explicitly

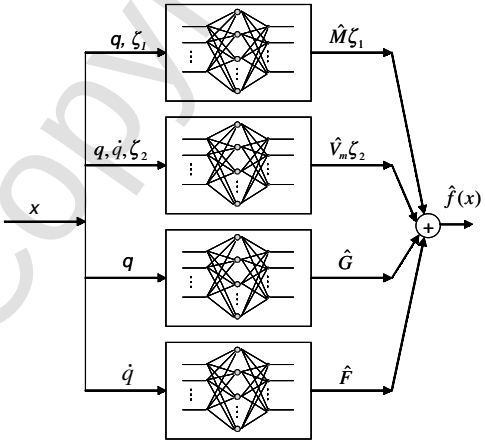


Figure 6 Partitioned NN.

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

introduce some of the nonlinearities inherent to robot arm dynamics. This reduces the burden of expectation on the NN and, in fact, also reduces the functional reconstruction error.

Consider an n-link robot having all revolute joints with joint variable vector $q(t)$. In revolute joint dynamics, the only occurrences of the joint variables are as sines and cosines,[23] so that the vector x can be taken as

$$x = [\zeta_1^T \quad \zeta_2^T \quad (\cos q)^T \quad (\sin q)^T \quad \dot{q}^T \quad \text{sgn}(q)^T]^T$$

where the signum function is needed in the friction terms.

4 NN CONTROL FOR DISCRETE-TIME SYSTEMS

Most feedback controllers today are implemented on digital computers. This requires the specification of control algorithms in *discrete-time* or digital form.[22] To design such controllers, one may consider the discrete-time dynamics $x(k + 1) = f(x(k)) + g(x(k))u(k)$, with functions $f(\cdot)$ and $g(\cdot)$ unknown. The digital NN controller derived in this situation still has the form of the feedback linearization controller shown in Fig. 4.

One can derive tuning algorithms for a discrete-time NN controller with N layers that guarantee system stability and robustness.[15] For the ith layer the weight updates are of the form

$$\hat{W}_i(k + 1) = \hat{W}_i(k) - \alpha_i \hat{\phi}_i(k) \hat{y}_i^T(k) - \Gamma \| I - \alpha_i \hat{\varphi}_i(k) \hat{\varphi}_i^T(k) \| \hat{W}_i(k)$$

where $\hat{\varphi}_i(k)$ are the output functions of layer i, $0 < \Gamma < 1$ is a design parameter, and

$$\hat{y}_i(k) \equiv \hat{W}_i^T(k) \hat{\varphi}_i(k) + K_v r(k) \quad \text{for } i = 1, \ldots, N - 1$$

and

$$\hat{y}_N(k) \equiv r(k + 1) \quad \text{for last layer}$$

with $r(k)$ a filtered error. This tuning algorithm has two parts: The first two terms correspond to a gradient algorithm often used in the NN literature. The last term is a discrete-time robustifying term that guarantees that the NN weights remain bounded. The latter has been called a "forgetting term" in NN terminology and has been used to avoid the problem of "NN weight overtraining."

Recently, NN control has been successfully extended to systems in strict-feedback form with a modified tuning law.[24]

5 MULTILOOP NN FEEDBACK CONTROL STRUCTURES

Actual industrial or military mechanical systems may have *additional dynamical complications* such as vibratory modes, high-frequency electrical actuator dynamics, compliant couplings or gears, and so on. Practical systems may also have *additional performance requirements* such as requirements to exert specific forces or torques as well as perform position trajectory following (e.g., robotic grinding or milling). In such cases, the NN in Fig. 4 still works if it is modified to include *additional inner feedback loops* to deal with the additional plant or performance complexities. Using Lyapunov energy-based techniques, it can be shown that, if each loop is state-strict passive, then the overall multiloop NN controller provides stability, performance, and bounded NN weights. Details appear in Ref. 15.

Copyrighted Material

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

5.1 Backstepping Neurocontroller for Electrically Driven Robot

Many industrial systems have high-frequency dynamics in addition to the basic system dynamics being controlled. An example of such systems is the n-link rigid robot arm with motor electrical dynamics given by

$$M(q)(\ddot{q}) + V_m(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + \tau_d = K_T i$$

$$L\dot{i} + R(i,\dot{q}) + \tau_e = u_e$$

with $q(t) \in R^n$ the joint variable, $i(t) \in R^n$ the motor armature currents, $\tau_d(t)$ and $\tau_e(t)$ the mechanical and electrical disturbances, and motor terminal voltage vector $u_e(t) \in R^n$ the control input. This plant has unknown dynamics in both the robot subsystem and the motor subsystem.

The problem with designing a feedback controller for this system is that one desires to control the behavior of the robot joint vector $q(t)$; however, the available control inputs are the motor voltages $u_e(t)$, which only affect the motor torques. As a second-order effect, the torques affect the joint angles.

Backstepping NN Design

The NN tracking controller in Fig. 7 may be designed using the *backstepping* technique.[25] This controller has *two neural networks,* one (NN 1) to estimate the unknown robot dynamics and an additional NN in an inner feedback loop (NN 2) to estimate the unknown motor dynamics. This multiloop controller is typical of control systems designed using rigorous system-theoretic techniques. It can be shown that by selecting suitable weight-tuning algorithms for both NNs, one can guarantee closed-loop stability as well as tracking performance in spite of the additional high-frequency motor dynamics. Both NN loops are state-strict passive. Proofs are given in terms of a modified Lyapunov approach. The NN tuning algorithms are similar to the ones presented above, but with some extra terms.

In standard backstepping, one must find several regression matrices, which can be complicated. By contrast, NN backstepping design does not require regression matrices since the NNs provide a universal basis for the unknown functions encountered.

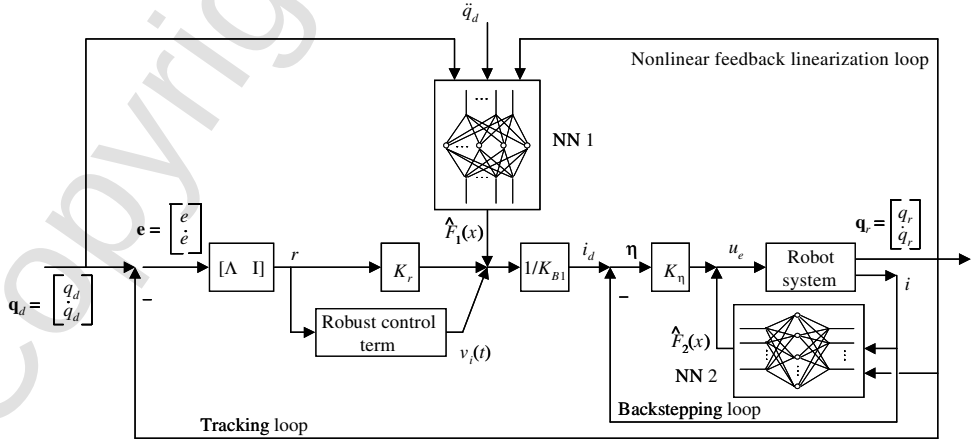


Figure 7 Backstepping NN controller for robot with motor dynamics.

Copyrighted Material

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

5.2 Compensation of Flexible Modes and High-Frequency Dynamics Using NNs

Actual industrial or military mechanical systems may have additional dynamical complications such as vibratory modes, compliant couplings or gears, and so on. Such systems are characterized by having more degrees of freedom than control inputs, which compounds the difficulty of designing feedback controllers with good performance. In such cases, the NN controller in Fig. 4 still works if it is modified to include additional inner feedback loops to deal with the additional plant complexities.

Using the Bernoulli–Euler equation, infinite series expansion, and the assumed mode shapes method, the dynamics of flexible-link robotic systems can be expressed in the form

$$\begin{bmatrix} M_{rr} & M_{rf} \\ M_{fr} & M_{ff} \end{bmatrix} \begin{bmatrix} \ddot{q}_r \\ \ddot{q}_f \end{bmatrix} + \begin{bmatrix} V_{rr} & V_{rf} \\ V_{fr} & V_{ff} \end{bmatrix} \begin{bmatrix} \dot{q}_r \\ \dot{q}_f \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & K_{ff} \end{bmatrix} \begin{bmatrix} q_r \\ q_f \end{bmatrix} + \begin{bmatrix} F_r \\ 0 \end{bmatrix} + \begin{bmatrix} G_r \\ 0 \end{bmatrix} = \begin{bmatrix} B_r \\ B_f \end{bmatrix} \tau$$

where $q_r(t)$ is the vector of rigid variables (e.g., joint angles), $q_f(t)$ the vector of flexible mode amplitudes, M an inertia matrix, V a Coriolis/centripetal matrix, and matrix partitioning is represented according to subscript r for the rigid modes and subscript f for the flexible modes. Friction F and gravity G apply only for the rigid modes. Stiffness matrix K_{ff} describes the vibratory frequencies of the flexible modes.

The problem in controlling such systems is that the input matrix $B = [B_r^{\mathrm{T}} \quad B_f^{\mathrm{T}}]^{\mathrm{T}}$ is not square but has more rows than columns. This means that while one is attempting to control the rigid-mode variable $q_r(t)$, one is also affecting $q_f(t)$. This causes undesirable vibrations. Moreover, the zero dynamics of such systems is non–minimum phase, which results in unstable flexible modes if care is not taken in choosing a suitable controller.

Singular Perturbations NN Design

To overcome this problem, an additional *inner feedback loop* based on singular perturbation theory[26] may be designed. The resulting multiloop controller is shown in Fig. 8, where a NN compensates for friction, unknown nonlinearities, and gravity and the inner loop manages the flexible modes. The internal dynamics controller in the inner loop may be designed using a variety of techniques, including H_∞ robust control and linear quadratic Gaussian/loop transfer recovery (LQG/LTR). Such controllers are capable of compensating for the effects

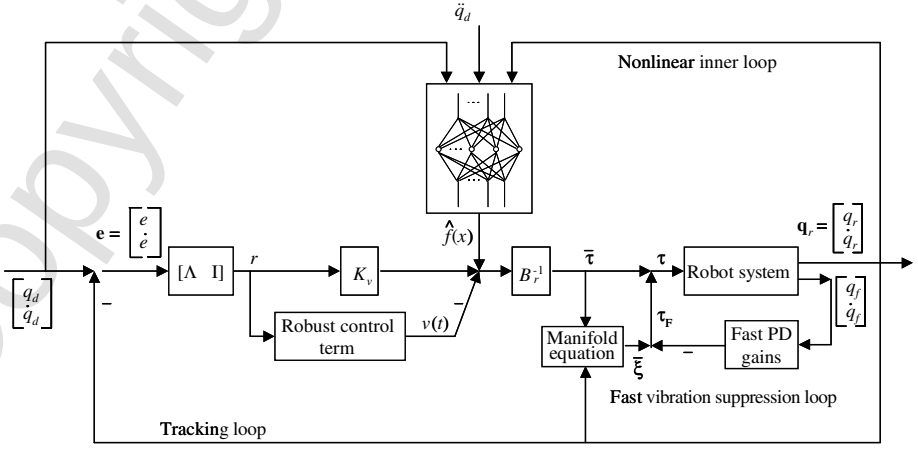


Figure 8 NN controller for flexible-link robotic system.

Copyrighted Material

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

of inexactly known or changing flexible mode frequencies. An observer can be used to avoid strain rate measurements.

In many industrial or aerospace designs, flexibility effects are limited by restricting the speed of motion of the system. This limits performance. By contrast, using the singular perturbation NN controller, *a flexible system can far outperform a rigid system in terms of speed of response*. The key is to use the flexibility effects to speed up the response in much the same manner as the cracking of a whip. That is, the flexibility effects of advanced structures are not merely a debility that must be overcome, but they offer the possibility of *improved performance* over rigid structures if they are suitably controlled. By exploiting recent advances in materials, such as piezoelectric materials, further improved performance is attainable for the so-called smart material flexible robots.[20b]

5.3 Force Control with Neural Nets

Many practical robot applications require the control of the force exerted by the manipulator normal to a surface along with position control in the plane of the surface. This is the case in milling and grinding, surface finishing, and so on. In applications such as MEMS assembly, where highly nonlinear forces, including van der Waals, surface tension, and electrostatics dominate gravity, advanced control schemes such as NNs are especially required.

In such cases, the NN force/position controller in Fig. 9 can be derived using rigorous Lyapunov-based techniques. It has guaranteed performance in that both the position-tracking error $r(t)$ and the force error $\tilde{\lambda}(t)$ are kept small while all the NN weights are kept bounded. The figure has an *additional inner force control loop*. The control input is now given by

$$\tau(t) = \hat{W}^{\mathrm{T}}\sigma(\hat{V}^{\mathrm{T}}x) + K_v(Lr) - J^{\mathrm{T}}(\lambda_d - K_f\tilde{\lambda}) - v$$

where the selection matrix L and Jacobian J are computed based on the decomposition of the joint variable $q(t)$ into two components—the component $q_1(t)$ (e.g., tangential to the given surface) in which position tracking is desired and the component $q_2(t)$ (e.g., normal to the surface) in which force exertion is desired. This is achieved using holonomic constraint techniques based on the prescribed surface that are standard in robotics (e.g., work by

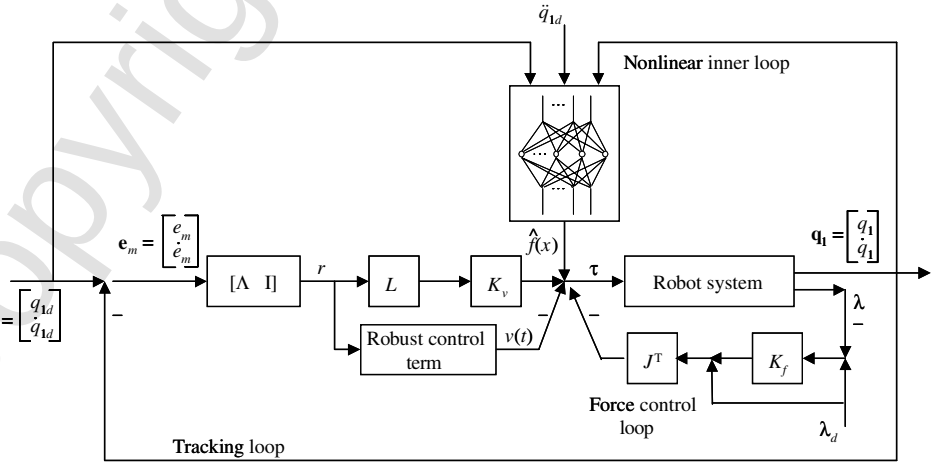


Figure 9 NN force/position controller.

Copyrighted Material

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

McClamroch[27] and others). The filtered position-tracking error in $q_1(t)$ is $r(t)$, that is, $r(t) = q_{1d} - q_1$ with $q_{1d}(t)$ the desired trajectory in the plane of the surface. The desired force is described by $\lambda_d(t)$ and the force exertion error is captured in $\tilde{\lambda}(t) = \lambda(t) - \lambda_d(t)$ with $\lambda(t)$ describing the actual measured force exerted by the manipulator. The position-tracking gain is K_v and the force-tracking gain is K_f.

6 FEEDFORWARD CONTROL STRUCTURES FOR ACTUATOR COMPENSATION

Industrial, aerospace, DoD, and MEMS assembly systems have actuators that generally contain deadzone, backlash, and hysteresis. Since these actuator nonlinearities appear in the *feedforward* loop, the NN compensator must also appear in the feedforward loop. The design problem for neurocontrollers where the NN appears in the feedforward loop is significantly more complex than for feedback NN controllers. Details are given in Ref. 28.

6.1 Feedforward Neurocontroller for Systems with Unknown Deadzone

Most industrial, vehicle, and aircraft actuators have deadzones. The deadzone characteristic appears in Fig. 10 and causes motion control problems when the control signal takes on small values or passes through zero, since only values greater than a certain threshold can influence the system.

Feedforward controllers can offset the effects of deadzones if properly designed. It can be shown that a NN deadzone compensator has the structure shown in Fig. 11. The NN compensator consists of *two* NNs: NN II is in the direct feedforward control loop, and NN I is not directly in the control loop but serves as an observer to estimate the (unmeasured) applied torque $\tau(t)$. The feedback stability and performance of the NN deadzone compensator have been rigorously proven using nonlinear stability proof techniques.

The two NNs were each selected as having one tunable layer, namely the output weights. The activation functions were set as a basis by selecting fixed random values for the first-layer weights.[7] To guarantee stability, the output weights of the inversion NN II and the estimator NN I should be tuned respectively as

$$\dot{\hat{W}}_i = T\sigma_i(U_i^{\mathrm{T}} w)r^{\mathrm{T}}\hat{W}^{\mathrm{T}}\sigma'(U^{\mathrm{T}}u)U^{\mathrm{T}} - k_1 T\|r\|\hat{W}_i - k_2 T\|r\|\|\hat{W}_i\|\hat{W}_i$$

$$\dot{\hat{W}} = -S\sigma'(U^{\mathrm{T}}u)U^{\mathrm{T}}\hat{W}_i\sigma_i(U_i^{\mathrm{T}}w)r^{\mathrm{T}} - k_i S\|r\|\hat{W}$$

where subscript i denotes weights and sigmoids of the inversion NN II and variables without subscripts correspond to NN I. Note that σ' denotes the Jacobian. Design parameters are the positive-definite matrices T and S and tuning gains k_1, k_2. The form of these tuning laws is

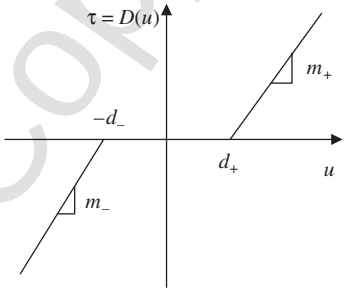


Figure 10 Deadzone response characteristic.

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

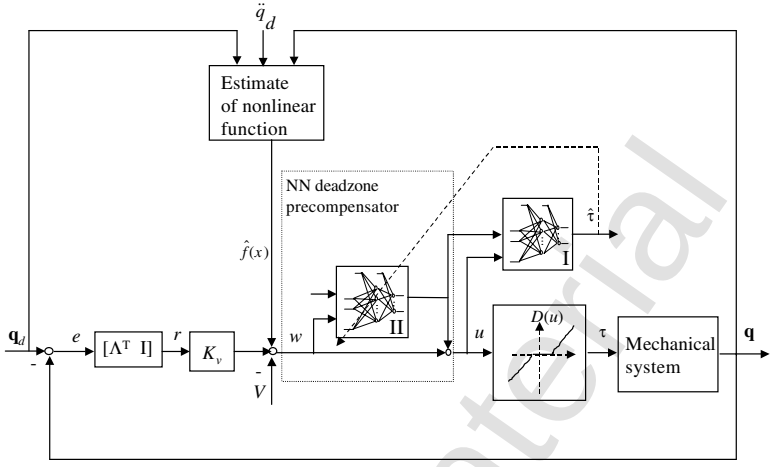


Figure 11 Feedforward NN for deadzone compensation.

intriguing. They form a coupled nonlinear system with each NN helping to tune itself and the other NN. Moreover, signals *are backpropagated through NN I* to tune NN II. That is, the two NNs function as *a single NN with two layers,* first NN II, then NN I, but with the second layer not in the direct control path. Note the additional terms, which are a combination of Narendra's *e*-modification and Ioannou's σ-modification.

Reinforcement Learning Structure

Neural network I is not in the control path but serves as a *higher level critic* for tuning NN II, the action-generating net. The critic NN I actually functions to provide an estimate of the torque supplied to the system in the absence of deadlock, which is a *target torque.* It is intriguing that this use of NN in the feedforward loop (as opposed to the feedback loop) requires such a *reinforcement learning* structure. Reinforcement learning techniques generally have the critic NN outside the main feedback loop, on a higher level of the control hierarchy.

6.2 Dynamic Inversion Neurocontroller for Systems with Backlash

Backlash is a common problem in actuators with gearing. The backlash characteristic is shown in Fig. 12 and causes motion control problems when the control signal reverses in direction, often due to dead space between gear teeth.

Dynamic inversion is a popular controller design technique in aircraft control and elsewhere.[29] Dynamic inversion by NNs has been used by Calise and co-workers[30] in aircraft control using NNs. Using dynamic inversion, a NN controller for systems with backlash is designed in Ref. 28. The neurocontroller appears in the feedforward loop, as in Fig. 13, and is a *dynamic or recurrent NN.* In this neurocontroller, a desired torque $\tau_{\text{des}}(t)$ to be applied is determined; then, using a backstepping type of approach,[25] the neurocontroller structure shown in Fig. 13 is derived. A NN is used to approximate certain nonlinear functions appearing in the derivation. Unlike backstepping, dynamic inversion lets the required derivative appear explicitly in the controller. In the design, a filtered derivative $\xi(t)$ is used to allow implementation in actual systems.

Copyrighted Material

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

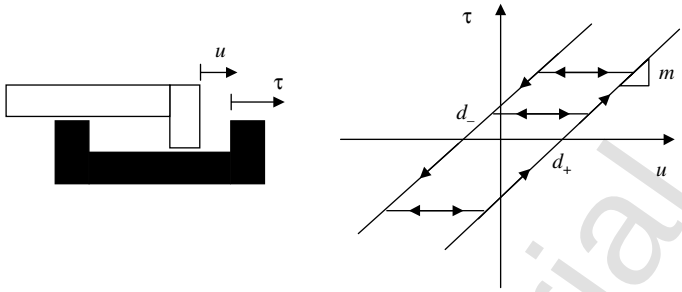


Figure 12 Backlash response characteristic.

The NN precompensator shown in Fig. 13 effectively adds control energy to invert the dynamical backlash function. The control input into the backlash element is given by

$$\hat{u}(t) = K_b\tilde{\tau} + \xi - y_{\text{nn}} + v_2$$

where $\tilde{\tau}(t) = \tau_{\text{des}}(t) - \tau(t)$ is the torque error, $y_{\text{nn}}(t)$ is the NN output, and $v_2(t)$ is a certain robust control term detailed in Ref. 28. Weight-tuning algorithms given there guarantee closed-loop stability and effective backlash compensation.

7 NN OBSERVERS FOR OUTPUT FEEDBACK CONTROL

Thus far, we have described NN controllers in the case of full state feedback, where all internal system information is available for feedback. However, in actual industrial and com-

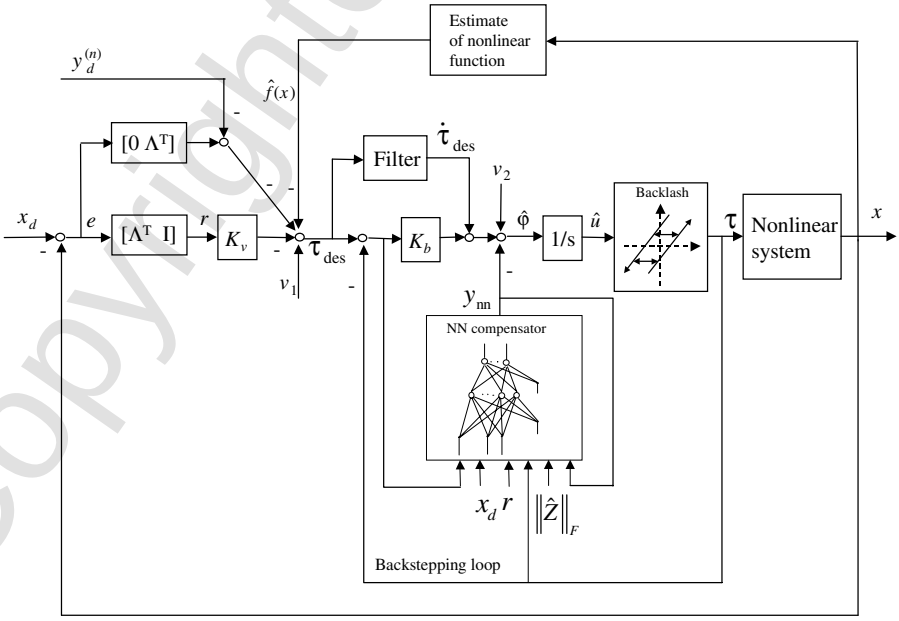


Figure 13 Dynamic inversion NN compensator for system with backlash.

Copyrighted Material

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

mercial systems, there are usually available only certain restricted measurements of the plant. In this output feedback case one may use an *additional dynamic NN* with its own internal dynamics in the controller. The function of this additional NN is effectively to provide estimates of the unmeasurable plant states, so that the dynamic NN functions as what is known as an *observer* in control system theory.

The issues of observer design using NNs can be appreciated with rigid robotic systems.[12] For these systems, the dynamics can be written in state-variable form as

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = M^{-1}(x_1)[-N(x_1, x_2) + \tau]$$

where $x \equiv q$, $x_2 \equiv \dot{q}$ and the nonlinear function $N(x_1, x_2) = V_m(x_1, x_2)x_2 + G(x_1) + F(x_2)$ is assumed to be unknown. It can be shown[31] that the following dynamic NN observer can provide estimates of the entire state $x = [x_1^T \quad x_2^T]^T \equiv [q^T \quad \dot{q}^T]^T$ given measurements of only $x_1(t) = q(t)$:

$$\dot{\hat{x}}_1 = \hat{x}_2 + k_D x_1$$

$$\dot{\hat{z}}_2 = M^{-1}(x_1)[-\hat{W}_o^T \sigma_o(\hat{x}) + k_P \tilde{x}_1 + \tau].$$

$$\hat{x}_2 = \hat{z}_2 + k_{P2} \tilde{x}_1$$

In this system, the hat denotes estimates and the tilde denotes estimation errors. It is assumed that the inertia matrix $M(q)$ is known, but all other nonlinearities are estimated by the observer NN $\hat{W}_o^T \sigma_o(\hat{x})$, which has output layer weights $\hat{W}_o$ and activation functions $\sigma_o(\cdot)$. Signal $v_o(t)$ is a certain observer robustifying term, and the observer gains k_P, k_D, k_{P2} are positive design constants detailed in Ref. 31.

The NN output feedback tracking controller shown in Fig. 14 uses the dynamic NN observer to reconstruct the missing measurements $x_2(t) = \dot{q}(t)$ and then employs a second static NN for tracking control, exactly as in Fig. 4. Note that the outer tracking PD loop structure has been retained but an additional dynamic NN loop is needed. In Ref. 31, weight-tuning algorithms that guarantee stability are given for both the dynamic estimator NN and the static control NN.

8 REINFORCEMENT LEARNING CONTROL USING NNs

Reinforcement learning techniques are based on psychological precepts of reward and punishment as used by I. P. Pavlov in the training of dogs at the turn of the century. The key tenet here is that the performance indicators of the controlled system should be simple, for instance, $+1$ for a successful trial and -1 for a failure, and that these simple signals should tune or adapt a NN controller so that its performance improves over time. This gives a learning feature driven by the basic success or failure record of the controlled system. Reinforcement learning has been studied by many researchers, including Refs. 32 and 33.

It is difficult to provide rigorous designs and analysis for reinforcement learning in the framework of standard control system theory since the reinforcement signal has reduced information, which makes study, including Lyapunov techniques, very complicated. Reinforcement learning is related to the so-called sign error tuning in adaptive control[34] which has not been proven to yield stability.

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

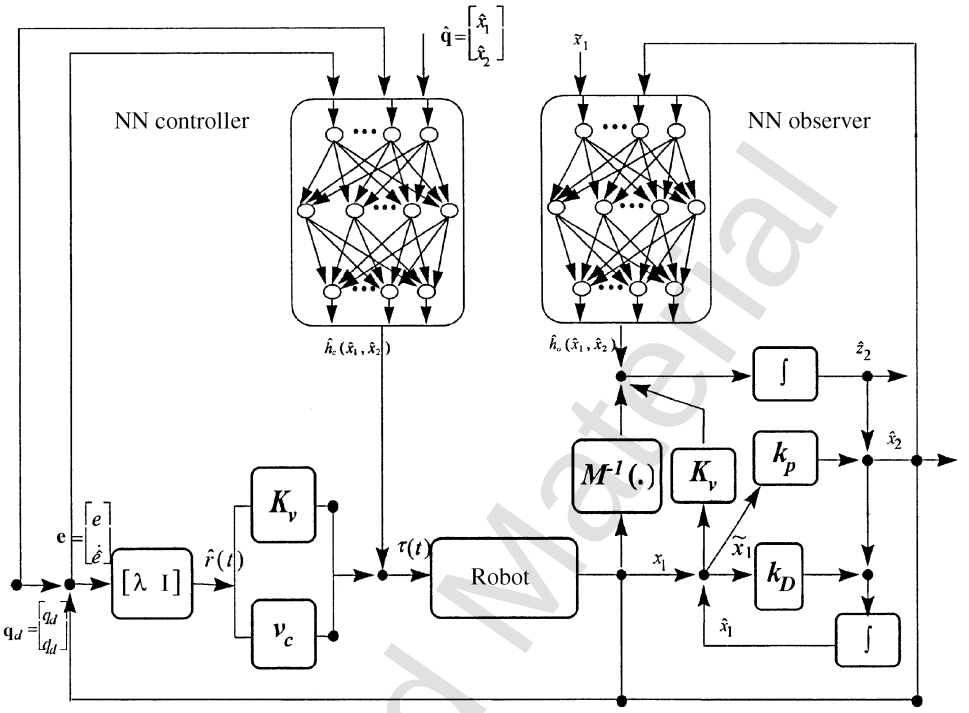


Figure 14 NN observer for output feedback control.

8.1 NN Reinforcement Learning Controller

A simple signal related to the performance of a robotic system is the signum of the sliding variable error $R(t) = \text{sgn}(r(t))$, with the sliding variable error given by $r = \dot{e} + \Lambda e$, where $e = q_d - q$ is the tracking error and matrix Λ is positive definite. Signal $R(t)$ satisfies the criteria required in reinforcement learning control: (i) It is simple, having values of only 0, ± 1 and (ii) the value of zero corresponds to a reward for good performance, while nonzero values correspond to a punishment signal. Therefore, $R(t)$ may be taken as a suitable reinforcement learning signal.

Rigorous proofs of closed-loop stability and performance for reinforcement learning may be provided[31] by (i) using *nonstandard Lyapunov functions,* (ii) deriving novel modified NN tuning algorithms, and (iii) selection of a suitable multiloop control structure. The architecture of the reinforcement adaptive learning NN controller derived is shown in Fig. 15. A performance evaluation loop has the desired trajectory $q_d(t)$ as the user input; this loop manufactures $r(t)$, which may be considered as the *instantaneous utility*. The critic element evaluates the signum function and so provides the reinforcement signal $r(t)$ which critiques the performance of the system.

It is not easy to show how to tune the action-generating NN using only the reinforcement signal $r(t)$, which contains significantly less information than the full error signal $r(t)$. A successful proof can be based on the *Lyapunov energy function*

$$L(t) = \sum_{i=1}^{n} |r_i| + \frac{1}{2} \text{tr}(\tilde{W}^T F^{-1} \tilde{W})$$

Copyrighted Material

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

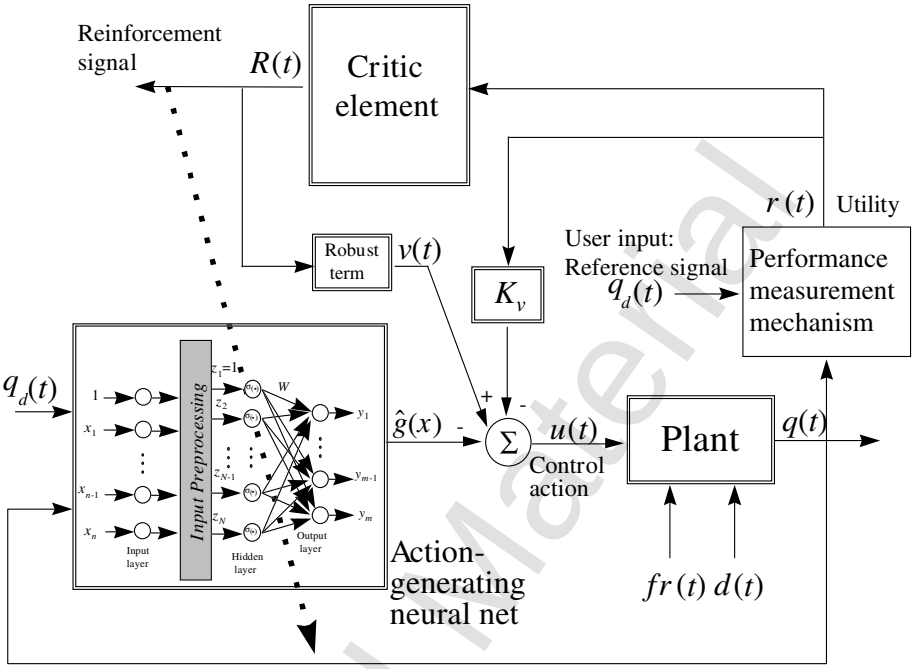


Figure 15 Reinforcement learning NN controller.

where $r(t) \in R^n$. This is not a standard Lyapunov function in feedback system theory but is similar to energy functions used in some NN convergence proofs (e.g., by Hopfield). Using this Lyapunov function, one can derive NN tuning algorithms that guarantee closed-loop stability and tracking. The NN weights are tuned using *only the reinforcement signal* $R(t)$ according to

$$\dot{\hat{W}} = F\sigma(x)R^{\mathrm{T}} - \kappa F\hat{W}$$

This is similar to what has been called *sign error tuning* in adaptive control, which has usually been proposed without giving any proof of stability or performance.

8.2 Adaptive Reinforcement Learning Using Fuzzy Logic Critic

Fuzzy logic systems are based on the higher level linguistic and reasoning abilities of humans and offer intriguing possibilities for use in feedback control systems. The idea of using backpropagation tuning to tune fuzzy logic systems was proposed by Werbos.[33] Through the work of Wang,[6] K. Passino, and S. Yurkovich,[35] and others, it is now known how to tune fuzzy logic systems so that they learn online to yield very good performance in closed-loop control applications.

A fuzzy logic (FL) system with product inferencing, centroid defuzzification, and singleton output membership functions has output vector $y(t)$ whose components are given in terms of the input vector $x(t) \in R^n$ by

$$y_k = \sum_{j=1}^{L} w_{kj}\sigma_j(x,U) \qquad \text{or} \qquad y = W^{\mathrm{T}}\sigma(x,U)$$

Copyrighted Material

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

where $W^T = [w_{kj}]$ is a matrix of output representative values and the FL basis functions $\sigma_j(\cdot)$ play the role of NN activation functions. Using product inferencing, the basis functions are given in terms of the one-dimensional membership functions (MFs) $\mu_{ij}(x,U_{ij})$ by

$$\sigma_j(x,U) = \frac{\prod_{i=1}^{n} \mu_{ij}(x_i,U_{ij})}{\sum_{j=1}^{L} \prod_{i=1}^{n} \mu_{ij}(x_i,U_{ij})}$$

where U_{ij} is a vector of parameters of the MFs including the centroids and spreads. The number of rules is L. The standard choice for the MFs is triangle functions. However, other choices have been used, including splines (c.f. Ref. 4, CMAC NN), second- or third-degree polynomials, or the RBF functions.[36]

FL systems have the connotation of higher level supervisors since they are rule based. The fuzzy-neural reinforcement learning scheme shown in Fig. 16 has been developed, where a FL system serves as a critic and a NN serves as an action-generating network that controls the system. The reinforcement controller is adaptive in the sense that the FL critic is tuned as well as the NN action-generating network to improve system performance through online learning. Stability and convergence proofs have been provided and depend on using certain specialized tuning schemes for the FL critic membership functions and the NN weights. Tuning the membership functions has the effect of modifying them so they converge onto the region in with highest state trajectory activity, a form of *dynamic focusing of awareness.*

The advantage of the FL/NN adaptive reinforcement learning structure is that the critic can be initialized using linguistic/heuristic notions by the human user. Finally, for FL systems one can look at the final MFs and interpret what information has been stored in the system through learning.

9 OPTIMAL CONTROL USING NNs

Heretofore we have discussed the design of NN controllers for tracking and stabilization based on control theory techniques including feedback linearization, backstepping, singular perturbations, force control, dynamic inversion, and observer design. The point was made

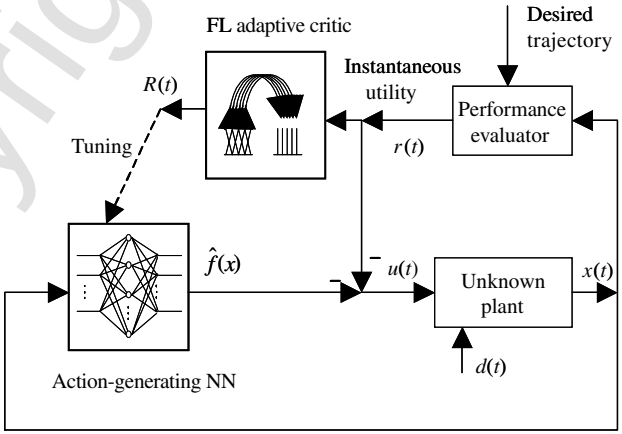


Figure 16 Fuzzy logic adaptive reinforcement learning NN controller.

Copyrighted Material

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

that as the system dynamical structure becomes more complex or the performance requirements become more stringent, it is necessary to add more feedback loops. Rigorous neurocontroller design algorithms may be given in terms of Lyapunov energy-based techniques, passivity, and so on.

Nonlinear optimal control design provides a very powerful theory that is applicable for systems in any form. Solution of the so-called Hamilton–Jacobi (HJ) equations will directly yield a controller with guaranteed properties in terms of stability and performance for any sort of nonlinear system. Unfortunately, the HJ equations are difficult to solve and may not even have analytic solutions for general nonlinear systems. In the special case of *linear* optimal control,[37] solution techniques are available based on Riccati equation techniques, and that theory provides a cornerstone of control design for aerospace systems, vehicles, and industrial plants. It would be very valuable to have tractable controller design techniques for general nonlinear systems. In fact, it has been shown that NNs afford computationally effective techniques for solving general HJ equations, and so for designing closed-loop controllers for general nonlinear systems.

9.1 NN H_2 Control Using the Hamilton–Jacobi–Bellman Equation

In work by Abu-Khalaf and Lewis[38] it has been shown how to solve the Hamilton–Jacobi–Bellman (HJB) equation that appears in optimal control for general nonlinear systems by a successive approximation (SA) technique based on NNs. Rigorous results have been proven, and a computationally effective scheme for nearly optimal controller design was provided based on NNs.

This technique allows one to consider general affine nonlinear systems of the form

$$\dot{x} = f(x) + g(x)u(x) \tag{6}$$

To give internal stability and good closed-loop performance, one may select the L_2 norm performance index

$$V(x(0)) = \int_0^\infty [Q(x) + u^{\mathrm{T}}Ru]\, dt \tag{7}$$

with matrix R positive definite and $Q(x)$ generally selected as a norm. It is desired to select the control input $u(t)$ to minimize the cost $V(x)$. Under suitable assumptions of detectability, this guarantees that the states and controls are bounded and hence that the closed-loop systems is stable.

An infinitesimal equivalent to the cost is given by

$$0 = \frac{\partial V^{\mathrm{T}}}{\partial x}(f + gu) + Q + u^{\mathrm{T}}Ru \equiv H\left(x, \frac{\partial V}{\partial x}, u\right) \tag{8}$$

which defines the Hamiltonian function $H(\cdot)$ and the costate as the cost gradient $\partial V/\partial x$. This is a nonlinear Lyapunov equation. It has been called a *generalized HJB equation* by Saridis and Lee.[39]

Differentiating with respect to the control input $u(t)$ to find a minimum yields the control in the form

$$u(x) = -\frac{1}{2} R^{-1} g^{\mathrm{T}}(x) \frac{\partial V(x)}{\partial x} \tag{9}$$

Substituting this into the previous equation yields the HJB equation of optimal control

Copyrighted Material

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

$$0 = \frac{\partial V^{\mathrm{T}}}{\partial x} f + Q - \frac{1}{4} \frac{\partial V^{\mathrm{T}}}{\partial x} g(x) R^{-1} g(x) \frac{\partial V}{\partial x} \tag{10}$$

The boundary condition for this equation is $V(0) = 0$. Solving this equation yields the optimal value function $V(x)$, whence the optimal control may be computed from the cost gradient using (4).

This procedure will give the optimal control in feedback form for any nonlinear system. Unfortunately, the HJB equation cannot be solved for most nonlinear systems. In the linear system case, the HJB equation yields the Riccati equation, for which efficient solution techniques are available. However, most systems of interest today in aerospace, vehicles, and industry are nonlinear.

Therefore, one may use a SA approach wherein (8) and (9) are iterated to determine sequences $V^{(i)}$, $u^{(i)}$. The initial stabilizing control $u^{(0)}$ used in (8) to find $V^{(0)}$ is easily determined using, for example, the linear quadratic regulator (LQR) for the linearization of (6). It has been shown by Saridis and Lee[39] that the SA converges to the optimal solution V^*, u^* of the HJB equation. Let the region of asymptotic stability of the optimal solution be Ω^* and the region with asymptotic stability (RAS) at iteration i be $\Omega^{(i)}$. Then, in fact, it has been shown that

$u^{(i)}$ is stabilizing for all i;

$V^{(i)} \rightarrow V^*$, $u^{(i)} \rightarrow u^*$, $\Omega^{(i)} \rightarrow \Omega^*$ uniformly;

$V^{(i)}(x) \geq V^{(i+1)}(x)$, that is, the value function decreases; and

$\Omega^{(i)} \leq \Omega^{(i+1)}$, that is, the RAS increases.

In fact, Ω^* is the largest RAS of any other admissible control law.

NNs for Computation of Successive Approximation Solution

It is difficult to solve Eqs. (8) and (9) as required for the SA method just given. Beard et al.[40] showed how to implement the SA algorithm using the Galerkin approximation to solve the nonlinear Lyapunov equation. This method is computationally intensive, since it requires the evaluation of numerous integrals. It was shown in Ref. 38 how to use NNs to compute the SA solution at each iteration. This yields a computationally effective method for determining nearly optimal controls for a general class of nonlinear constrained input systems. The value function at each iteration is approximated using a NN by

$$V(x) \approx V(x, w_j) = w^{(i)\mathrm{T}} \sigma(x)$$

with w_j the NN weights and $\sigma(x)$ a basis set of activation functions. To satisfy the initial condition $V^{(i)}(0) = 0$ and the symmetry requirements on $V(x)$, the activation functions were selected as a basis of even polynomials in x. Then the parameterized nonlinear Lyapunov equation becomes

$$0 = w^{(i)\mathrm{T}} \nabla\sigma(x)(f(x) + g(x)u^{(i)}) + Q + u^{(i)\mathrm{T}} R u^{(i)}$$

with $u^{(i)}$ the current control value. Evaluating this equation at enough sample values of x, it can easily be solved for the weights using, for example, least squares. The sample values of x must satisfy a condition known as *persistence of excitation* in order to obtain a unique least-squares solution for the weights. The number of samples selected must be greater than the number of NN weights. Then, the next iteration value of the control is given by

$$u^{(i+1)}(x) = -\tfrac{1}{2} R^{-1} g^{\mathrm{T}}(x) \nabla\sigma^{\mathrm{T}}(x) w^{(i)}$$

Using a Sobolev space setting, it was shown that under certain mild assumptions the NN solution converges in the mean to a suitably close approximation of the optimal solution.

Copyrighted Material

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

Moreover, if the initial NN weights are selected to yield an admissible control, then the control is admissible (which implies stability) at each iteration.

The control given by this approach is shown in Fig. 17. It is a *feedback control* in terms of a nonlinear NN. This approach has also been given for constrained input systems, such as industrial and aircraft actuator systems.

9.2 NN H_∞ Control Using the Hamilton–Jacobi–Isaacs Equation

Many systems contain unknown disturbances, and the optimal control approach just given may not be effective. In this case, one may use the H_∞ design procedure.

Consider the dynamical system in Fig. 18, where $u(t)$ is an action or control input, $d(t)$ is a disturbance or opponent, $y(t)$ is the measured output, and $z(t)$ is a performance output with $\|z\|^2 = h^{\mathrm{T}}h + \|u\|^2$. Here we take full state feedback $y=x$ and desire to determine the action or control $u(t) = u(x(t))$ such that, under the worst disturbance, one has the L_2 gain bounded by a prescribed γ so that

$$\frac{\int_0^\infty \|z(t)\|^2\, dt}{\int_0^\infty \|d(t)\|^2\, dt} = \frac{\int_0^\infty (h^{\mathrm{T}}h + \|u\|^2)\, dt}{\int_0^\infty \|d(t)\|^2\, dt} \le \gamma^2$$

This is a differential game with two players[41,42] and can be confronted by defining the utility

$$r(x,u,d) = h^{\mathrm{T}}(x)h(x) + \|u(t)\|^2 - \gamma^2\|d(t)\|^2$$

and the long-term value (cost-to-go)

$$V(x(t)) = \int_t^\infty r(x,u,d)\, dt = \int_t^\infty (h^{\mathrm{T}}(x)h(x) + \|u(t)\|^2 - \gamma^2\|d(t)\|^2)\, dt \tag{11}$$

The optimal value is given by

$$V^*(x(t)) = \min_{u(t)} \max_{d(t)} \int_t^\infty r(x,u,d)\, dt$$

The optimal control and worst-case disturbance are given by the stationarity conditions as

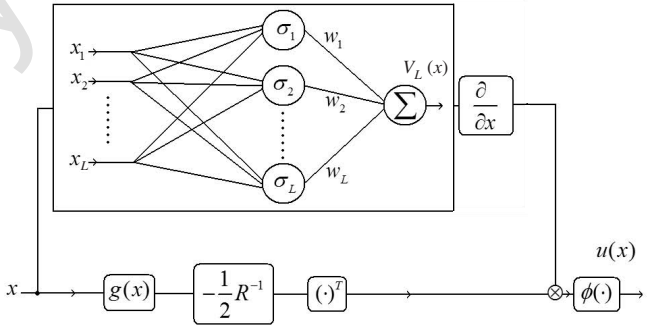


Figure 17 Nearly optimal NN feedback control for constrained input nonlinear systems.

Copyrighted Material

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

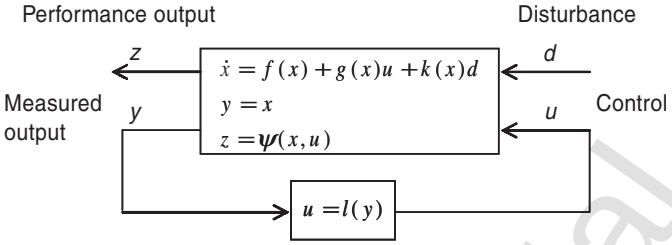


Figure 18 Bounded L_2 gain problem.

$$u^*(x(t)) = -\frac{1}{2} g^{\mathrm{T}}(x) \frac{\partial V^*}{\partial x} \tag{12}$$

$$d^*(x(t)) = \frac{1}{2\gamma^2} k^{\mathrm{T}}(x) \frac{\partial V^*}{\partial x} \tag{13}$$

If the *min–max* and *max–min* solutions are the same, then a saddle point exists and the game has a unique solution. Otherwise, we consider the min–max solution, which confers a slight advantage to the action input $u(t)$.

The infinitesimal equivalent to (11) is found using Leibniz's formula to be

$$0 = \dot{V} + r(x,u,d) = \left(\frac{\partial V}{\partial x}\right)^{\mathrm{T}} \dot{x} + r(x,u,d) = \left(\frac{\partial V}{\partial x}\right)^{\mathrm{T}} F(x,u,d) + r(x,u,d) \equiv H\left(x, \frac{\partial V}{\partial x}, u, d\right) \tag{14}$$

with $V(0)=0$, where $H(x,\lambda,u,d)$ is the Hamiltonian with $\lambda(t)$ the costate and $\dot{x} = F(x,u,d) \equiv f(x) + g(x)u + k(x)d$. This is a nonlinear Lyapunov equation.

Substituting u^* and d^* into (14) yields the nonlinear HJI equation

$$0 = \left(\frac{dV^*}{dx}\right)^{\mathrm{T}} f + h^{\mathrm{T}}h - \frac{1}{4}\left(\frac{dV^*}{dx}\right)^{\mathrm{T}} gg^{\mathrm{T}} \frac{dV^*}{dx} + \frac{1}{4\gamma^2}\left(\frac{dV^*}{dx}\right)^{\mathrm{T}} kk^{\mathrm{T}} \frac{dV^*}{dx} \tag{15}$$

whose solution provides the optimal value V^* and hence the solution to the min–max differential game. Unfortunately, this equation cannot generally be solved.

In Ref. 43 it has been shown that the following two-loop successive approximation policy iteration algorithm has very desirable properties like those delineated above for the H_2 case. First one finds a stabilizing control for zero disturbance. Then one iterates Eqs. (13) and (14) until there is convergence with respect to the disturbance. Now one selects an improved control using (12). The procedure repeats until there is convergence of both loops. Note that it is easy to select the initial stabilizing control u_0 by setting $d(t)=0$ and using LQR design[37] on the linearized system dynamics.

NN Solution of HJI Equation for H_∞ Control

To implement this algorithm practically one may approximate the value at each step using a one-tunable-layer NN as

$$V(x) \approx V(x,w_j^i) = w_j^{i^{\mathrm{T}}} \sigma(x)$$

with $\sigma(x)$ a basis set of activation functions. The disturbance iteration is in index i and the control iteration is in index j. Then the parameterized nonlinear Lyapunov equation (14) becomes

Copyrighted Material

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

$$0 = w_j^{i\mathrm{T}} \nabla\sigma(x)\dot{x} + r(x,u_j,d^i) = w_j^i \nabla\sigma(x)F(x,u_j,d^i) + h^{\mathrm{T}}h + \|u_j\|^2 - \gamma^2\|d^i\|^2$$

which can easily be solved for the weights using, for example, least squares. Then, on disturbance iterations the next disturbance is given by

$$d^{i+1}(x) = -\tfrac{1}{2}R^{-1}k^{\mathrm{T}}(x)\,\nabla\sigma^{\mathrm{T}}(x)w_j^i$$

and on control iterations the improved control is given by

$$u_{j+1}(x) = -\tfrac{1}{2}R^{-1}g^{\mathrm{T}}(x)\,\nabla\sigma^{\mathrm{T}}(x)w_j^i$$

This algorithm is shown to converge to the approximately optimal H_∞ solution. This yields a NN feedback controller as shown in Fig. 17 for the H_2 case.

10 APPROXIMATE DYNAMIC PROGRAMMING AND ADAPTIVE CRITICS

Approximate dynamic programming (ADP) is based on the optimal formulation of the feedback control problem. For discrete-time systems, the optimal control problem may be solved using dynamic programming,[37] which is a backward-in-time procedure and so unsuitable for online implementation. ADP is based on using nonlinear approximators to solve the HJ equations forward in time and was first suggested by Werbos.[44] See the Section 11 for cited works of major researchers in the area of ADP. The current status of work in ADP is given in Ref. 45.

The previous section presented the continuous-time formulation of the optimal control problem. For discrete-time systems of the form

$$x_{k+1} = f(x_k,u_k)$$

with k the time index, one may select the cost or performance measure

$$V(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} r(x_i,u_i)$$

with γ a discount factor and $r(x_k,u_k)$ known as the instantaneous utility. A first-difference equivalent to this yields a recursion for the value function given by

$$V(x_k) = r(x_k,u_k) + \gamma V(x_{k+1})$$

One may invoke Bellman's principle to find the optimal cost as

$$V^*(x_k) = \min_{u_k} (r(x_k,u_k) + \gamma V^*(x_{k+1}))$$

and the optimal control as

$$u^*(x_k) = \arg\min_{u_k} (r(x_k,u_k) + \gamma V^*(x_{k+1}))$$

Determining the optimal controller using these equations requires an iterative procedure known as dynamic programming that progresses backward in time. This is unsuitable for real-time implementation and is computationally complex.

The goal of ADP is to provide approximate techniques for evaluating the optimal value and optimal control using techniques that progress forward in time, so that they can be implemented in actual control systems. Howard[46] showed that the following successive iteration scheme, known as *policy iteration,* converges to the optimal solution:

Copyrighted material

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

1. Find the value for the prescribed policy $u_j(x_k)$:

$$V_j(x_k) = r(x_k,u_j(x_k)) + \gamma V_j(x_{k+1})$$

2. Policy improvement:

$$u_{j+1}(x_k) = \arg\min_{u_k} (r(x_k,u_k) + \gamma V_j(x_{k+1}))$$

Werbos[33] and others (see Section 11) showed how to implement ADP controllers by four basic techniques—HDP, DHP, and their action-dependent forms ADHDP, and ADDHP—to be described next.

Heuristic Dynamic Programming (*HDP*). In HDP, one approximates the value by a *critic* NN with tunable parameters w_j and the control by an *action-generating* NN with tunable parameters v_j so that

$$\text{Critic NN:} \qquad V_j(x_k) \approx V(x_k,w_j)$$

$$\text{Action NN:} \qquad u_j(x_k) \approx u(x_k,v_j)$$

HDP then proceeds as follows:

Critic Update

Find the desired target value using

$$V^D{}_{k,j+1} = r(x_k,u_j(x_k)) + \gamma V(x_{k+1},w_j)$$

Update critic weights using recursive least squares (RLS), backprop, and so on:

$$w_{j+1} = w_j + \alpha_j \frac{\partial V}{\partial w_j} [V^D{}_{k,j+1} - V(x_k,w_j)]$$

Action Update

Find the desired target action using

$$u^D{}_{k,j+1} = \arg\min_{u_k} (r(x_k,u_k) + \gamma V(x_{k+1},w_{j+1}))$$

Update critic weights using RLS, backprop, and so on:

$$v_{j+1} = v_j + \alpha_k \frac{\partial u}{\partial v_j} [u^D{}_{k,j+1} - u(x_k,v_j)]$$

This procedure is straightforward to implement given today's software (e.g., MATLAB). The value required for the next state x_{k+1} may be found either using the dynamics equation (1) or the next state can be observed from the actual system.

Dual Heuristic Programming (*DHP*). Noting that the control only depends on the value function gradient [e.g., see (9)], it is advantageous to approximate not the value but its gradient using a NN. This yields a more complex algorithm, but DHP converges faster than HDP. Details are in Ref. 33.

Q-Learning or Action-Dependent HDP. A function that is more advantageous than the value function for ADP is the Q function, defined by Watkins[47] and Werbos[8] as

$$Q(x_k,u_k) = r(x_k,u_k) + \gamma V(x_{k+1})$$

Note that Q is a function of both x_k and the control action u_k and that

Copyrighted Material

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

$$Q_h(x_k,h(x_k)) = V_h(x_k)$$

where subscript h denotes a prescribed control or policy sequence $u_k = h(x_k)$. A recursion for Q is given by

$$Q_h(x_k,u_k) = r(x_k,u_k) + \gamma Q_h(x_{k+1},h(x_{k+1}))$$

In terms of Q, Bellman's principle is particularly easy to write; in fact, defining the optimal Q value as

$$Q^*(x_k,u_k) = r(x_k,u_k) + \gamma V^*(x_{k+1}))$$

one has the optimal value as

$$V^*(x_k) = \min_{u_k}(Q^*(x_k,u_k))$$

The optimal control policy is given by

$$h^*(x_k) = \arg\min_{u_k}(Q^*(x_k,u_k))$$

Watkins showed that the following successive iteration scheme, known as *Q learning,* converges to the optimal solution:

1. Find the Q value for the prescribed policy $h_j(x_k)$:

$$Q_j(x_k,u_k) = r(x_k,u_k) + \gamma Q_j(x_{k+1},h_j(x_{k+1}))$$

2. Policy improvement:

$$h_{j+1}(x_k) = \arg\min_{u_k}(Q_j(x_k,u_k))$$

Using NN to approximate the Q function and the policy, one can write the ADHDP algorithm in a very straightforward manner. Since the control input action u_k is now explicitly an input to the critic NN, this is known as action-dependent HDP. Q learning converges faster than HDP and can be used in the case of unknown system dynamics.[48]

An action-dependent version of DHP is also available wherein the gradients of the Q function are approximated using NNs. Note that two NNs are needed, since there are two gradients, as Q is a function of both x_k and u_k.

11 HISTORICAL DEVELOPMENT, REFERENCED WORK, AND FURTHER STUDY

A firm foundation for the use of NNs in feedback control systems has been developed over the years by many researchers. Included here is a historical development and references to the body of work in neurocontrol.

11.1 NN for Feedback Control

The use of NNs in feedback control systems was first proposed by Werbos.[8] Since then, NNs control has been studied by many researchers. Recently, NNs have entered the mainstream of control theory as a natural extension of adaptive control to systems that are nonlinear in the tunable parameters. The state of NN control is well illustrated by papers in the Automatica Special issue on NN control.[49] Overviews of the initial work in NN control are provided by Miller et al.[50] and the *Handbook of Intelligent Control,*[51] which highlighted a host of difficulties to be addressed for closed-loop control applications. Neural network

Copyrighted Material

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

applications in closed-loop control are fundamentally different from open-loop applications such as classification and image processing. The basic multilayer NN tuning strategy is backpropagation.[17] Basic problems that had to be addressed for closed-loop NN control[33,44] included weight initialization for feedback stability, determining the gradients needed for backpropagation tuning, determining what to backpropagate, obviating the need for preliminary off-line tuning, modifying backprop so that it tunes the weights forward through time, and providing efficient computer code for implementation. These issues have since been addressed by many approaches.

Initial work in NN was for system identification and identification-based indirect control. In closed-loop control applications, it is necessary to show the stability of the tracking error as well as boundedness of the NN weight estimation errors. Proofs for internal stability, bounded NN weights (e.g., bounded control signals), guaranteed tracking performance, and robustness were absent in early works. Uncertainty as to how to initialize the NN weights led to the necessity for ''preliminary off-line tuning.'' Work on off-line learning was formalized by Kawato.[52] Off-line learning can yield important structural information.

Subsequent work in NNs for control addressed closed-loop system structure and stability issues. Work by Sussmann[53] and Albertini and Sontag[54] was important in determining system properties of NNs (e.g., minimality and uniqueness of the ideal NN weights, observability of dynamic NNs). The seminal work of Narendra and Parthasarathy[9,10] had an emphasis on finding the gradients needed for backprop tuning in feedback systems, which, when the plant dynamics are included, become recurrent nets. In recurrent nets, these gradients themselves satisfy difference or differential equations, so they are difficult to find. Sadegh[55] showed that knowing an approximate plant Jacobian is often good enough to guarantee suitable closed-loop performance.

The approximation properties of NN[2,3] are basic to their feedback controls applications. Based on this and analysis of the error dynamics, various modifications to backprop were presented that guaranteed closed-loop stability as well as weight error boundedness. These are akin to terms added in adaptive control to make algorithms robust to high-frequency unmodeled dynamics. Sanner and Slotine[5] used radial basis functions in control and showed how to select the NN basis functions, Polycarpou and Ioannou[56,57] used a projection method for weight updates, and Lewis and Syrmos[37] used backprop with an *e*-modification term.[18] All this work used NNs that are *linear* in the unknown parameter. In linear NNs, the problem is relegated to determining activation functions that form a *basis set* (e.g., RBF[5] and functional link programmable network (FLPN)[55]). It was shown by Sanner and Slotine[5] how to systematically derive stable NN controllers using approximation theory and basis functions. Barron[58] has shown that using NNs that are linear in the tunable parameters gives a fundamental limitation of the approximation accuracy to the order of $1/L^{2/n}$, where L is the number of hidden layer neurons and n is the number of inputs. Nonlinear-in-the-parameters NNs overcome this difficulty and were first used by Chen and Khalil,[59] who used backprop with deadzone weight tuning, and Lewis et al.,[60] who used Narendra's *e*-modification term in the backprop. In nonlinear-in-the-parameters NNs, the basis is automatically selected online by tuning the first-layer weights and thresholds. Multilayer NNs were rigorously used for discrete-time control by Jagannathan and Lewis.[61] Polycarpou[62] derived NN controllers that do not assume known bounds on the ideal weights. Dynamic/recurrent NNs were used for control by Rovithakis and Christodoulou,[63] Poznyak,[64] Rovithakis,[65] who considered multiplicative disturbances; Zhang and Wang,[66] and others.

Most stability results on NN control have been local in nature, and global stability has been treated by Kwan et al.,[67] and others. Recently, NN control has been used in conjunction with other control approaches to extend the class of systems that yields to nonparametric control. Calise and coworkers[30,68,69] used NNs in conjunction with dynamic inversion to

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

control aircraft and missiles. Feedback linearization using NNs has been addressed by Chen and Khalil,[59] Yesildirek and Lewis,[70] Ge et al.,[22] and others. NNs were used with backstepping[25] by Lewis, et al.,[15] Arslan and Basar,[71] Wang and Huang,[72] Ge et al.,[20,21] and others.

NNs have been used in conjunction with the Isidori–Byrnes regulator equations for output-tracking control by Wang and Huang.[72] A multimodel NN control approach has been given by Narendra and Balakrishnan.[73] Applications of NN control have been extended to partial differential equation systems by Padhi et al.[74] NNs have been used for control of stochastic systems by Poznyak and Ljung.[75] Parisini and co-workers have developed receding horizon controllers based on NNs[76] and hybrid discrete-event NN controllers.[77]

In practical implementations of NN controllers there remain problems to overcome. Weight initialization still remains an issue, and one may also find that the NN weights become unbounded despite proofs to the contrary. Practical implementation issues were addressed by Chen and Chang,[78] Gutierrez and Lewis,[79] and others. Random initialization of the first-layer NN weights often works in practice, and work by Igelnik and Pao[7] shows that it is theoretically defensible. Computational complexity makes NNs with many hidden layer neurons difficult to implement. Recently, work has intensified in wavelets, NNs that have localized basis functions, and NNs that are self-organizing in the sense of adding or deleting neurons automatically.[36,80,81]

By now it is understood that NNs offer an elegant extension of adaptive control and other techniques to systems that are nonlinear in the unknown parameters. The universal approximation properties of NNs[2,3] avoid the use of specialized basis sets, including regression matrices. Formalized improved proofs avoid the use of assumptions such as certainty equivalence. Robustifying terms avoid the need for persistency of excitation. Recent books on NN feedback control include Refs. 15, 20, 22, 28, 31, and 82.

11.2 Approximate Dynamic Programming

Adaptive critics are reinforcement learning designs that attempt to approximate dynamic programming.[83,84] They approach the optimal solution through forward approximate dynamic programming. Initially, they were proposed by Werbos.[44] Overviews of the initial work in NN control are provided by Miller et al.[50] and the *Handbook of Intelligent Control*.[52] Howard[46] showed the convergence of an algorithm relying on the successive policy iteration solution of a nonlinear Lyapunov equation for the cost (value) and an optimizing equation for the control (action). This algorithm relied on perfect knowledge of the system dynamics and is an off-line technique. Later, various online dynamic-programming-based reinforcement learning algorithms emerged and were mainly based on Werbos's HDP,[33] Sutton's temporal differences (TDs) learning methods,[85] and Q-learning, which was introduced by Watkins[47] and Werbos[8] (called *action-dependent* critic schemes there). Critic and action network tuning was provided by RLS, gradient techniques, or the backpropagation algorithm.[17] Early work on dynamic-programming-based reinforcement learning focused on discrete finite-state and action spaces. These depended on lookup tables or linear function approximators. Convergence results were shown in this case, such as Dayan.[86]

For continuous-state and action spaces, convergence results are more challenging as adaptive critics require the use of nonlinear function approximators. Four schemes for approximate dynamic programming were given in Ref. 33, the HDP and DHP algorithms and their action-dependent versions (ADHDP and ADDHP). The linear quadratic regulation (LQR) problem[37] served as a testbed for much of these studies. Solid convergence results were obtained for various adaptive critic designs for the LQR problem. We mention the work of Bradtke et al.,[48] where Q learning was shown to converge when using nonlinear function

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

approximators. An important persistence of excitation notion was included. Further work was done by Landelius,[87] who studied the four adaptive critic architectures. He demonstrated convergence results for all four cases in the LQR case and discussed when the design is model free. Hagen and Krose[88] discussed the effect of model noise and exploration noise when the adaptive critic is viewed as a stochastic approximation technique. Prokhorov and Feldkamp[89] looked at Lyapunov stability analysis. Other convergence results are due to Balakrishnan and co-workers,[74,90] who have also studied the optimal control of aircraft and distributed-parameter systems governed by partial differential equations. Anderson et al.[91] showed convergence and stability for a reinforcement learning scheme. All of these results were done for the discrete-time case.

A thorough treatment of neurodynamic programming is given in the seminal book by Bertsekas and Tsitsiklis.[92] Various successful practical implementations have been reported, including aircraft control examples by Ferrari and Stengel,[93] an Auto Lander by Murray et al.,[94] state estimation using dynamic NNs by Feldkamp and Prokhorov,[95] and Neuro-Observers by Liu and Balakrishnan.[96] Si et al. have provided analysis[97] and applied ADP to aircraft control.[46] An account of adaptive critic designs is found in Prokhorov and Wunsch.

Applications of adaptive critics in the continuous-time domain were mainly done through discretization and the application of well-established discrete-time results (e.g., Ref. 99). Various continuous-time nondynamic reinforcement learning strategies were discussed by Campos and Lewis[100] and Rovithakis,[101] who approximated a Lyapunov function derivative. In Kim and Lewis[31] the HJB equation of dynamic programming is approximated by a Riccati equation, and a suboptimal controller based on NN feedback linearization is implemented with full stability and convergence proofs. Murray et al.[102] prove convergence of an algorithm that uses system state measurements to find the cost to go. An array of initial conditions is needed. Unknown plant dynamics in the linear case is confronted by estimating a matrix of state derivatives. The cost functional is shown to be a Lyapunov function and is approximated using either quadratic functions or an RBF neural network. Saridis and Lee[39] showed the convergence of an off-line algorithm relying on the successive iteration solution of a nonlinear Lyapunov equation for the cost (value) and an optimizing equation for the control (action). This is the continuous-time equivalent of Howard's work. Beard et al.,[40] showed how to actually solve these equations using Galerkin integral approximations, which require much computational effort.

Q-learning is not well posed when sampling times become small and so is not useful for extension to continuous-time systems. Continuous-time dynamic-programming-based reinforcement learning is reformulated using the so-called advantage learning by Baird,[103] who defines a differential increment from the optimal solution and explicitly takes into account the sampling interval Δt. Doya[104] derives results for online updating of the critic using techniques from continuous-time nonlinear optimal control. The advantage function follows naturally from this approach and in fact coincides with the continuous-time Hamiltonian function. Doya gives relations with the TD(0) and TD(λ) techniques of Sutton.[85] Lyshevski[105] has focused on a general parametrized form for the value function and obtained a set of algebraic equations that can be solved for an approximate value function.

Acknowledgments

The referenced work of Lewis and co-workers was sponsored by National Science Foundation grant ECS-01-40490 and Army Research Office grant DAAD19-02-1-0366.

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

REFERENCES

1. von Bertalanffy, L., *General System Theory,* Braziller, New York, 1968.
2. G. Cybenko, ''Approximation by Superpositions of a Sigmoidal Function,'' *Mathematics of Control, Signals and Systems* **2**(4), 303–314 (1989).
3. J. Park and I. W. Sandberg, ''Universal Approximation Using %Radial-Basis-Function Networks,'' *Neural Computation* **3,** 246–257 (1991).
4. J. S. Albus, ''A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller Equations (CMAC),'' *Transactions ASME Journal of Dynamics, Systems, Measurement, and Control,* **97,** 220–227 (1975).
5. R. M. Sanner and J.-J. E. Slotine, ''Gaussian Networks for Direct Adaptive Control,'' *IEEE Transactions on Neural Networks* **3**(6), 837–863 (1992).
6. L.-X. Wang, *Adaptive Fuzzy Systems and Control: Design and Stability Analysis,* Prentice-Hall, Englewood Cliffs, NJ, 1994.
7. B. Igelnik and Y.-H. Pao, ''Stochastic Choice of Basis Functions in Adaptive Function Approximation and Functional-Link Net,'' *IEEE Transaction on Neural Networks* **6**(6), 1320–1329 (1995).
8. P. J. Werbos, ''Neural Networks for Control and System Identification,'' in *Proceedings of the IEEE Conference on Decision and Control,* FL, 1989.
9. K. S. Narendra and K. Parthasarathy, ''Identification and Control of Dynamical Systems Using Neural Networks,'' *IEEE Transactions on Neural Networks,* **1,** 4–27 (1990).
10. K. S. Narendra and K. Parthasarathy, ''Gradient Methods for the Optimization of Dynamical Systems Containing Neural Networks,'' *IEEE Transactions on Neural Networks* **2**(2), 252–262 (1991).
11. Y. D. Landau, *Adaptive Control,* Marcel Dekker, New York, 1979.
12. F. L. Lewis, D. M. Dawson, and C. T. Abdallah, *Robot Manipulator Control,* Marcel Dekker, New York, 2004.
13. J. J. E. Slotine and J. A. Coetsee, ''Adaptive Sliding Controller Synthesis for Nonlinear Systems,'' *International Journal of Control* **43**(4), 1631–1651 (1986).
14. J. J. E. Slotine and W. Li, ''On the Adaptive Control of Robot Manipulators,'' *International Journal of Robotics Research* **6**(3), 49–59 (1987).
15. F. L. Lewis, S. Jagannathan, and A. Yesildirek, *Neural Network Control of Robot Manipulators and Nonlinear Systems,* Taylor and Francis, London, 1999.
16. L. R. Hunt, R. Su, and G. Meyer, ''Global Transformations of Nonlinear Systems,'' *IEEE Transactions on Automatic Control* **28,** 24–31 (1983).
17. P. J. Werbos, ''Beyond Regression: New Tools for Prediction and Analysis in the Behavior Sciences,'' Ph.D. Thesis, Committee on Applied Mathemathics, Harvard University, 1974.
18. K. S. Narendra and A. M. Annaswamy, ''A New Adaptive Law for Robust Adaptation without Persistent Excitation,'' *IEEE Transactions on Automatic Control,* **AC-32**(2), 134–145 (1987).
19. P. Ioannou and J. Sun, *Robust Adaptive Control,* Prentice-Hall, Englewood Cliffs, NJ, 1996; electronic copy available at http://www-rcf.usc.edu/~ioannou/Robust_Adaptive_Control.htm
20. S. S. Ge, C. C. Hang, T. H. Lee, and T. Zhang, *Stable Adaptive Neural Network Control, Kluwer, Boston, MA, 2001.*
21. S. S. Ge, T. H. Lee, and Z. P. Wang, ''Adaptive Neural Network Control for Smart Materials Robots Using Singular Perturbation Technique,'' *Asian Journal of Control* **3**(2), 143–155 (2001).
22. S. S. Ge, T. H. Lee, and C. J. Harris, *Adaptive Neural Network Control of Robotic Manipulators,* World Scientific, Singapore, 1998.
23. F. L. Lewis, *Applied Optimal Control and Estimation: Digital Design and Implementation,* TI Series, Prentice-Hall, Englewood Cliffs, NJ, 1992.
24. S. S. Ge, G. Y. Li, and T. H. Lee, ''Adaptive NN Control for a Class of Strict Feedback Discrete-Time Nonlinear Systems,'' *Automatica* **39,** 807–819 (2003).
25. M. Krstic, I. Kanellakopoulos, and P. Kokotovic, *Nonlinear and Adaptive Control Design,* Wiley, New York, 1995.
26. P. V. Kokotovic, ''Applications of Singular Perturbation Techniques to Control Problems,'' *SIAM Review,* **26**(4), 501–550 (1984).

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

27. N. H. McClamroch and D. Wang, "Feedback Stabilization and Tracing of Constrained Robots," *IEEE Transactions on Automatic Control,* **33,** 419–426 (1988).

28. F. L. Lewis, J. Campos, and R. Selmic, *Neuro-Fuzzy Control of Industrial Systems with Actuator Nonlinearities,* Society of Industrial and Applied Mathematics Press, Philadelphia, PA, 2002.

29. B. L. Stevens and F. L. Lewis, *Aircraft Control and Simulation,* 2nd ed., Wiley, New York, 2003.

30. A. J. Calise, N. Hovakimyan, and H. Lee, "Adaptive Output Feedback Control of Nonlinear Systems Using Neural Networks," *Automatica* **37**(8), 1201–1211 (2001).

31. Y. H. Kim and F. L. Lewis, *High-Level Feedback Control with Neural Networks,* World Scientific, Singapore, 1998.

32. A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuron-like Elements That Can Solve Difficult Learning," *IEEE Transactions on Systems, Man, and Cybernetics,* **13**(5), 634–646 (1983).

33. P. J. Werbos, "Approximate Dynamic Programming for Real-Time Control and Neural Modeling," in *Handbook of Intelligent Control,* D. A. White and D. A. Sofge (eds.), Van Nostrand Reinhold, New York, 1992.

34. C. R. Johnson, Jr., *Lectures on Adaptive Parameter Estimation,* Prentice-Hall, Englewood Cliffs, NJ, 1988.

35. K. M. Passino and S. Yurkovich, *Fuzzy Control,* Addison-Wesley, Menlo Park, NJ, 1998.

36. R. M. Sanner and J.-J. E. Slotine, "Structurally Dynamic Wavelet Networks for Adaptive Control of Robotic Systems," *International Journal of Control* **70**(3), 405–421 (1998).

37. F. L. Lewis and V. Syrmos, *Optimal Control,* 2nd ed., Wiley, New York, 1995.

38. M. Abu-Khalaf and F. L. Lewis, "Nearly Optimal State Feedback Control of Constrained Nonlinear Systems Using a Neural Networks HJB Approach," *IFAC Annual Reviews in Control* **28,** 239–251 (2004).

39. G. Saridis and C. S. Lee, "An Approximation Theory of Optimal Control for Trainable Manipulators," *IEEE Transactions on Systems, Man, and Cybernetics,* **9**(3), 152–159 (1979).

40. R. Beard, G. Saridis, and J. Wen, "Galerkin Approximations of the Generalized Hamilton-Jacobi-Bellman Equation," *Automatics,* **33**(12), 2159–2177 (1997).

41. H. Knobloch, A. Isidori, and D. Flockerzi, *Topics in Control Theory,* Springer Verlag, Boston, 1993.

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

43. M. Abu-Khalaf, F. L. Lewis, and J. Huang, "Computational Techniques for Constrained Nonlinear State Feedback *H*-Infinity Optimal Control Using Neural Networks," paper 1141, presented at the Mediterranean Conference on Control and Automation, Kusadasi, Turkey, June 2004.

44. P. J. Werbos., "A Menu of Designs for Reinforcement Learning Over Time," in *Neural Networks for Control,* W. T. Miller, R. S. Sutton, and P. J. Werbos (eds.), MIT Press, Cambridge, MA, 1991, pp. 67–95.

45. J. Si, A. Barto, W. Powell, and D. Wunsch, *Handbook of Learning and Approximate Dynamic Programming,* IEEE Press, West Conshohocken, PA, 2004.

46. R. Howard, *Dynamic Programming and Markov Processes,* MIT Press, Cambridge, MA, 1960.

47. C. Watkins, "Learning from Delayed Rewards," Ph.D. Thesis, Cambridge University, Cambridge, England, 1989.

48. S. Bradtke, B. Ydstie, and A. Barto, "Adaptive Linear Quadratic Control Using Policy Iteration," CMPSCI-94-49, University of Massachusetts, Amherst, MA, June 1994.

49. K. S. Narendra and F. L. Lewis, Special Issue on Neural Network Feedback Control, *Automatica* **37**(8) (2001).

50. W. T. Miller, R. S. Sutton, and P. J. Werbos (eds.), *Neural Networks for Control,* MIT Press, Cambridge, MA, 1991.

51. D. A. White and D. A. Sofge (eds.), *Handbook of Intelligent Control,* Van Nostrand Reinhold, New York, 1992.

52. M. Kawato, "Computational Schemes and Neural Network Models for Formation and Control of Multijoint Arm Trajectory," in *Neural Networks for Control,* W. T. Miller, R. S. Sutton, and P. J. Werbos (eds.), MIT Press, Cambridge, MA, 1991, pp. 197–228.

53. H. J. Sussmann, "Uniqueness of the Weights for Minimal Feedforward Nets with a Given Input-Output Map," *Neural Networks* **5,** 589–593 (1992).

Copyrighted Material

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

54. F. Albertini and E. D. Sontag, "For Neural Nets, Function Determines Form," *Proceedings of the IEEE Conference Decision and Control,* December 1992, pp. 26–31.

55. N. Sadegh, "A Perceptron Network for Functional Identification and Control of Nonlinear Systems," *IEEE Transactions on Neural Networks* **4**(6), 982–988 (1993).

56. M. M. Polycarpou and P. A. Ioannou, "Identification and Control Using Neural Network Models: Design and Stability Analysis," Technical Report 91-09-01, Dept. Elect. Eng. Sys., University of Southern California, Los Angeles, CA, September 1991.

57. M. M. Polycarpou and P. A. Ioannou, "Neural Networks as On-Line Approximators of Nonlinear Systems," in *Proceedings of the IEEE Conference on Decision and Control,* Tucson, December 1992, pp. 7–12.

58. A. R. Barron, "Universal Approximation Bounds for Superpositions of a Sigmoidal Function," *IEEE Transactions on Information Theory* **39**(3), 930–945 (1993).

59. F.-C. Chen and H. K. Khalil, "Adaptive Control of Nonlinear Systems Using Neural Networks," *International Journal of Control,* **55**(6), 1299–1317 (1992).

60. F. L. Lewis, A. Yesildirek, and K. Liu, "Multilayer Neural Net Robot Controller with Guaranteed Tracking Performance," *IEEE Transactions on Neural Networks,* **7**(2), 388–399 (1996).

61. S. Jagannathan and F. L. Lewis, "Multilayer Discrete-Time Neural Net Controller with Guaranteed Performance," *IEEE Transactions on Neural Networks* **7**(1), 107–130 (1996).

62. M. M. Polycarpou, "Stable Adaptive Neural Control Scheme for Nonlinear Systems," *IEEE Transactions on Automatic Control* **41**(3), 447–451 (1996).

63. G. A. Rovithakis and M. A. Christodoulou, "Adaptive Control of Unknown Plants Using Dynamical Neural Networks," *IEEE Transactions on Systems, Man, and Cybernetics* **24**(3), 400–412 (1994).

64. A. S. Poznyak, E. N. Sanchez, and W. Yu, *Differential Neural Networks for Robust Nonlinear Control,* World. Scientific, Singapore, 2001.

65. G. A. Rovithakis, "Performance of a Neural Adaptive Tracking Controller for Multi-Input Nonlinear Dynamical Systems," *IEEE Transactions on Systems, Man, and Cybernetic, Part A* **30**(6), 720–730, (2000).

66. Y. Zhang and J. Wang, "Recurrent Neural Networks for Nonlinear Output Regulation," *Automatica* **37**(8), 1161–1173 (2001).

67. C. Kwan, D. M. Dawson, and F. L. Lewis, "Robust Adaptive Control of Robots Using Neural Network: Global Stability," *Asian Journal of Control* **3**(2), 111–121 (2001).

68. J. Leitner, A. J. Calise, and J. V. R. Prasad, "Analysis of Adaptive Neural Networks for Helicopter Flight Control," *Journal of Guidance, Control, and Dynamics* **20**(5), 972–979 (1997).

69. M. B. McFarland and A. J. Calise, "Adaptive Nonlinear Control of Agile Anti-Air Missiles Using Neural Networks," *IEEE Transactions on Control Systems Technology* **8**(5), 749–756 (2000).

70. A. Yesildirek and F. L. Lewis, "Feedback Linearization Using Neural Networks," *Automatica* **31**(11), 1659–1664 (1995).

71. G. Arslan and T. Basar, "Disturbance Attenuating Controller Design for Strict-Feedback Systems with Structurally Unknown Dynamics," *Automatica* **37**(8), 1175–1188 (2001).

72. D. Wang and J. Huang, "Neural Network Based Adaptive Tracking of Uncertain Nonlinear Systems in Triangular Form," *Automatica* **38,** 1365–1372 (2002).

73. K. S. Narendra and J. Balakrishnan, "Adaptive Control Using Multiple Models," *IEEE Transactions on Automatic Control* **42**(2) 171–187 (1997).

74. R. Padhi, S. N. Balakrishnan, and T. Randolph, "Adaptive-Critic Based Optimal Neuro Control Synthesis for Distributed Parameter Systems," *Automatica* **37**(8), 1223–1234 (2001).

75. A. S. Poznyak and L. Ljung, "On-Line Identification and Adaptive Trajectory Tracking for Nonlinear Stochastic Continuous Time Systems Using Differential Neural Networks," *Automatica* **37**(8), 1257–1268 (2001).

76. T. Parisini, M. Sanguineti, and R. Zoppoli, "Nonlinear Stabilization by Receding-Horizon Neural Regulators," *International Journal of Control* **70,** 341–362 (1998).

77. T. Parisini and S. Sacone, "Stable Hybrid Control Based on Discrete-Event Automata and Receding-Horizon Neural Regulators," *Automatica* **37**(8), 1279–1292 (2001).

78. F.-C. Chen and C.-H. Chang, "Practical Stability Issues in CMAC Neural Network Control Systems," *IEEE Transactions on Control Systems Technology,* **4**(1), 86–91 (1996).

79. L. B. Gutierrez and F. L. Lewis, "Implementation of a Neural Net Tracking Controller for a Single Flexible Link: Comparison with PD and PID Controllers," *IEEE Transactions on Industrial Electronics* **45**(2), 307–318 (1998).

Copyrighted Material

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

80. J. A. Farrell, ''Stability and Approximator Convergence in Nonparametric Nonlinear Adaptive Control,'' *IEEE Transactions on Neural Networks* **9**(5), 1008–1020 (1998).

81. J. Y. Choi and J. A. Farrell, ''Nonlinear Adaptive Control Using Networks of Piecewise Linear Approximators,'' *IEEE Transactions on Neural Networks* **11**(2), 390–401 (2000).

82. R. Zbikowski and K. J. Hunt, *Neural Adaptive Control Technology,* World Scientific, Singapore, 1996.

83. A. G. Barto, ''Connectionist Learning for Control,'' in *Neural Networks for Control,* W. T. Miller, R. S. Sutton, P. J. Werbos (eds.), MIT Press, Cambridge, MA, 1991.

84. A. G. Barto and T. G. Dietterich, ''Reinforcement Learning and Its Relationship to Supervised Learning,'' in *Handbook of Learning and Approximate Dynamic Programming,* J. Si, A. Barto, W. Powell, and D. Wunsch (eds.), IEEE Press, West Conshohocken, PA, 2004.

85. R. Sutton, ''Learning to Predict by the Method of Temporal Differences,'' *Machine Learning* **3,** 9–44 (1988).

86. P. Dayan, ''The Convergence of TD(λ) for General λ,'' *Machine Learning* **8**(3–4), 341–362 (1992).

87. T. Landelius, ''Reinforcement Learning and Distributed Local Model Synthesis,'' Ph.D. Dissertation, Linköping University, 1997.

88. S. T. Hagen and B. Krose, ''Linear Quadratic Regulation Using Reinforcement Learning,'' in *Proceedings of the Eighth Belgian-Dutch Conference on Machine Learning, BENELEARN'98,* F. Verdenius and W. van den Broek (eds.), Wageningen, October 1998, pp. 39–46.

89. D. V. Prokhorov amd L. A. Feldkamp, ''Analyzing for Lyapunov Stability with Adaptive Critics,'' in *Proceedings of the International Conference on Systems, Man, Cybernetics,* Dearborn, MI, 1998, pp. 1658–1661.

90. X. Liu and S. N. Balakrishnan, ''Convergence Analysis of Adaptive Critic Based Optimal Control,'' in *Proceedings of the American Control Conference,* Chicago, IL, 2000, pp. 1929–1933.

91. C. Anderson, R. M. Kretchner, P. M. Young, and D. C. Hittle, ''Robust Reinforcement Learning Control with Static and Dynamic Stability,'' *International Journal of Robust and Nonlinear Control,* **11** (2001).

92. D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming,* Athena Scientific, MA, 1996.

93. S. Ferrari and R. Stengel, ''An Adaptive Critic Global Controller,'' in *Proceedings of the American Control Conference,* Anchorage, AK, 2002, pp. 2665–2670.

94. J. Murray, C. Cox, R. Saeks, and G. Lendaris, ''Globally Convergent Approximate Dynamic Programming Applied to an Autolander,'' in *Proc. ACC,* Arlington, VA, 2001, pp. 2901–2906.

95. L. Feldkamp and D. Prokhorov, ''Recurrent Neural Networks for State Estimation,'' paper presented at the Twelfth Yale Workshop on Adaptive and Learning Systems, New Haven, CT, 2003, pp. 17–22.

96. X. Liu and S. N. Balakrishnan, ''Adaptive Critic Based Neuro-Observer,'' in *Proceedings of the American Control Conference,* Arlington, VA, 2001, pp. 1616–1621.

97. J. Si and Y.-T. Wang, ''On-Line Control by Association and Reinforcement,'' *IEEE Transactions on Neural Networks* **12**(2), 264–276 (2001).

98. D. Prokhorov and D. Wunsch, ''Adaptive Critic Designs,'' *IEEE Transactions on Neural Networks* **8**(5) (1997).

99. J. N. Tsitsiklis, ''Efficient Algorithms for Globally Optimal Trajectories,'' *IEEE Transactions on Automatic Control* **40**(9), 1528–1538 (1995).

100. J. Campos and F. L. Lewis, ''Adaptive Critic Neural Network for Feedforward Compensation,'' in *Proceedings of the American Control Conference,* San Diego, CA, June 1999.

101. G. A. Rovithakis, ''Stable Adaptive Neuro-Control Via Lyapunov Function Derivative Estimation,'' *Automatica* **37**(8), 1213–1221 (2001).

102. J. Murray, C. Cox, G. Lendaris, and R. Saeks, ''Adaptive Dynamic Programming,'' *IEEE Transactions on Systems, Man, and Cybernetics* **32**(2) (2002).

103. L. Baird, ''Reinforcement Learning in Continuous Time: Advantage Updating,'' in *Proceedings of the International Conference on Neural Networks,* Orlando, FL, June 1994.

104. K. Doya, ''Reinforcement Learning in Continuous Time and Space,'' *Neural Computation* **12,** 219–245 (2000).

105. S. E. Lyshevski, *Control Systems Theory with Engineering Applications,* Birkhauser, Berlin, 2001.

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com

BIBLIOGRAPHY

Beard, R., G. Saridis, and J. Wen, ''Approximate Solutions to the Time-Invariant Hamilton-Jacobi-Bellman Equation,'' *Automatica* **33**(12), 2159–2177 (1997).

Gong, J. Q., and B. Yao, ''Neural Network Adaptive Robust Control of Nonlinear Systems in Semi-Strict Feedback Form,'' *Automatica* **37**(8), 1149–1160 (2001).

Lewis, F. L., ''Nonlinear Network Structures for Feedback Control,'' *Asian Journal of Control* **1**(4), 205–228 (1999).

Lewis, F. L., K. Liu, and A. Yesildirek, ''Neural Net Robot Controller with Guaranteed Tracking Performance,'' *IEEE Transactions on Neural Networks* **6**(3), 703–715 (1995).

Li, Y., N. Sundararajan, and P. Saratchandran, ''Neuro-Controller Design for Nonlinear Fighter Aircraft Maneuver Using Fully Tuned Neural Networks,'' *Automatica* **37**(8), 1293–1301 (2001).

Mendel, J. M., and R. W. MacLaren, ''Reinforcement Learning Control and Pattern Recognition Systems,'' in *Adaptive, Learning, and Pattern Recognition Systems: Theory and Applications,* J. M. Mendel and K. S. Fu (eds.), Academic, New York, 1970, pp. 287–318.

Miyamoto, H., M. Kawato, T. Setoyama, and R. Suzuki, ''Feedback-Error-Learning Neural Network for Trajectory Control of a Robotic Manipulator,'' *Neural Networks,* **1,** 251–265 (1988).

Narendra, K. S., ''Adaptive Control of Dynamical Systems Using Neural Networks,'' in *Handbook of Intelligent Control,* D. A. White and D. A. Sofge (eds.), Van Nostrand Reinhold, New York, 1992, pp. 141–183.

Poznyak, A. S., W. Yu, E. N. Sanchez, and J. P. Perez, ''Nonlinear Adaptive Trajectory Tracking Using Dynamic Neural Networks,'' *IEEE Transactions on Neural Networks* **10**(6), 1402–1411 (1999).

Selmic, R., F. L. Lewis, A. J. Calise, and M. B. McFarland, ''Backlash Compensation Using Neural Network,'' U.S. Patent 6,611,823, August 26, 2003.

Wang, J., and J. Huang, ''Neural Network Enhanced Output Regulation in Nonlinear Systems,'' *Automatica* **37**(8) 1189–1200 (2001).

Zhang, T., S. S. Ge, and C. C. Hang, ''Adaptive Neural Network Control for Strict Feedback Nonlinear Systems Using Backstepping Design,'' *Automatica* **36**(12), 1835–1846 (2000).

Copyrighted Material

Copyright © 2006 John Wiley & Sons Retrieved from: www.knovel.com