To appear in Mechanical Engineer's Handbook,
John Wiley, New York, 2005.

Neural Networks in Feedback Control Systems

F.L. Lewis
Automation and Robotics Research Institute
The University of Texas at Arlington
7300 Jack Newell Blvd. S, Ft. Worth, Texas 76118
Tel. 817-272-5972, lewis@uta.edu, http://arri.uta.edu/acs

and

Shuzhi Sam Ge
Department of Electrical Engineering
National University of Singapore
Singapore 117576, Tel. 6874 6821
elegesz@nus.edu.sg, http://vlab.ee.nus.edu.sg/~sge

Table of Contents

Introduction 2
Background 3
Neural Networks 3
Neural Network Control Topologies 4
Feedback Linearization Design of NN Tracking Controllers 4
Multi-Layer Neural Network Controller 5
Single-layer Neural Network Controller 6
Feedback Linearization of Nonlinear Systems Using NN 6
Partitioned Neural Networks and Input Preprocessing 7
NN Control for Discrete-Time Systems 8
Multi-loop Neural Network Feedback Control Structures 8
Backstepping Neurocontroller for Electrically Driven Robot 8
Compensation of Flexible Modes and High-Frequency Dynamics Using NN 9
Force Control with Neural Nets 10
Feedforward Control Structures for Actuator Compensation 10
Feedforward Neurocontroller for Systems with Unknown Deadzone 10
Dynamic Inversion Neurocontroller for Systems with Backlash 11
Neural Network Observers for Output-Feedback Control 12
Reinforcement Learning Control Using NN 13
Neural Network Reinforcement Learning Controller 13
Adaptive Reinforcement Learning Using Fuzzy Logic Critic 14
Optimal Control Using NN 15
Neural Network *H-2* Control Using the Hamilton-Jacobi-Bellman Equation 15
Neural Network *H-Infinity* Control Using the Hamilton-Jacobi-Isaacs Equation 17
Approximate Dynamic Programming and Adaptive Critics 18
Historical Development, Referenced Work, and Further Study 21
Neural Network for Feedback Control 21
Approximate Dynamic Programming 22
References 24

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. A.N. Whitehead and L. von Bertalanffy [1968] 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 so 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-infinity 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 on-line, so that performance improves automatically. Techniques include fuzzy logic control, which mimics linguistic and reasoning functions, and artificial neural networks, 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 NN provide an elegant extension of adaptive control techniques to nonlinearly parameterized learning systems.

This article shows how NN 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 in this article are *multiloop controllers* with NN 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 neural networks in control system applications. See the section on historical development and further study.

Background

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 $\Re^n$ into an output space $\Re^m$. Its properties include function approximation, learning, generalization, classification, etc. It is known that the 2-layer NN has sufficient generality for closed-loop control purposes. The 2-layer neural network 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

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

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 $\sigma(.)$ by '1' in the first position. Similarly, the first-layer thresholds are included as the first column of matrix V^T by augmenting vector x by '1' in the first position.

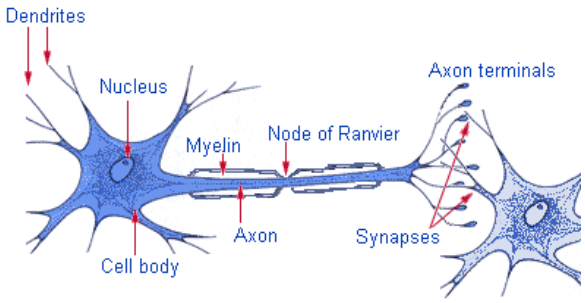

Fig. 1 Nervous System Cell. Cited with permission from http://www.sirinet.net/~jgjohnso/index.html

The main property of NN we are concerned with for control and estimation purposes is the *function approximation property* [Cybenko 1989]. Let $f(x)$ be a smooth function from $\Re^n \to \Re^m$. Then it can be shown that if the activation functions are suitably selected and x is restricted to a compact set $S \in \Re^n$, then for some sufficiently large number L of hidden-layer neurons, there exist weights and thresholds such one has

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

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

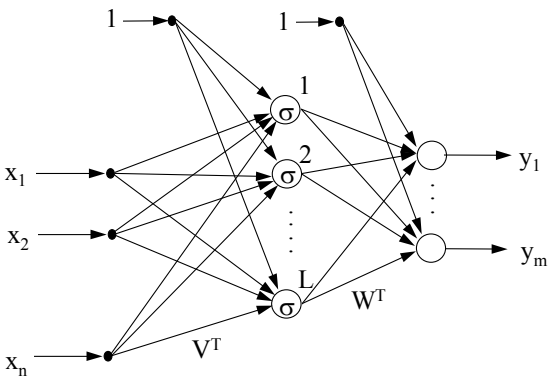

Fig. 2 Two-Layer neural network (NN)

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-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 cerebellar model articulation controller (CMAC) [Albus 1975],

radial basis function (RBF) [Sanner and Slotine 1992], fuzzy logic [L.X. Wang 1994], 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 [Igelnik and Pao 1995] that $\sigma(V^T x)$ is almost always a basis if V is selected randomly.

Neural Network 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 NN in control systems was first proposed by Werbos [1989] and Narendra [1990]. NN control has had two major thrusts: Approximate Dynamic Programming, which uses NN to approximately solve the optimal control problem, and NN in closed-loop feedback control. Many researchers have contributed to the development of these fields. See the Historical Development and References Section at the end of this article.

Several NN feedback control topologies are illustrated in Fig. 3 [Narendra and Parthasarathy 1991], some of which are derived from standard topologies in adaptive control [Landau 1979]. Solid lines denote control signal flow loops while dashed lines denote tuning loops. There are basically two sorts of feedback control topologies- indirect techniques 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.

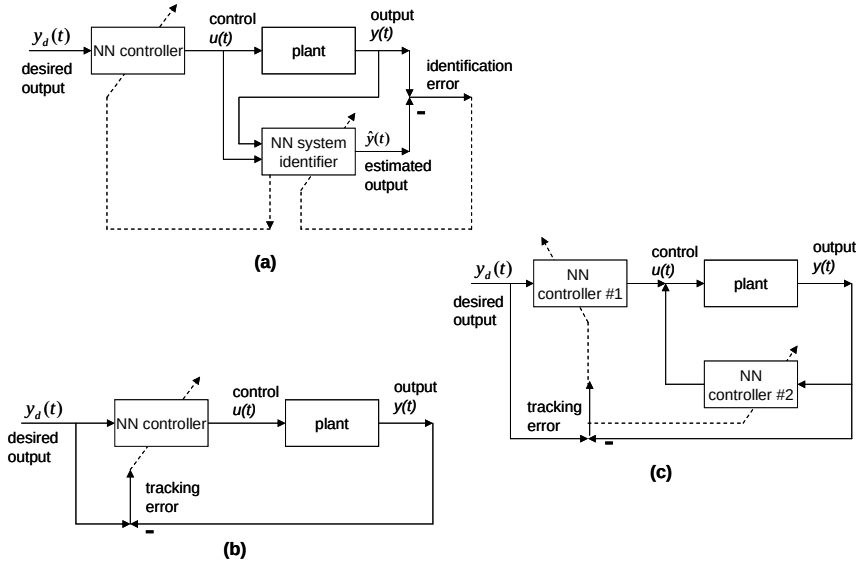

Fig. 3 NN Control Topologies. (a) Indirect scheme. (b) Direct scheme. (c) Feedback/feedforward scheme.

The challenge in using NN 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 [Lewis 1999]. In this article, 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 NN in control applications. See the section on historical development.

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 [Lewis, Dawson, Abdallah 2004]

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

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*(.) representing friction terms. Bounded unknown disturbances and modeling errors are denoted by τ_d and the control input torque is $\tau(t)$.

Given a desired arm trajectory $q_d(t) \in R^n$ define the tracking error $e(t) = q_d(t) - q(t)$ and the filtered tracking 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

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

Multi-Layer Neural Network Controller

A NN controller may be designed based on the *functional approximation properties* of NN as shown in [Lewis, Jagannathan, Yesildirek 1999]. Thus, assume that $f(x)$ is unknown and given approximately as the output of a NN with unknown "ideal" weights *W, V* so that $f(x) = W^T\sigma(V^T x) + \varepsilon$ with ε an approximation error. The key is now to approximate $f(x)$ by the NN functional estimate $\hat{f}(x) = \hat{W}^T\sigma(\hat{V}^T x)$, 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 linear-in-the-parameters (LIP) controllers.

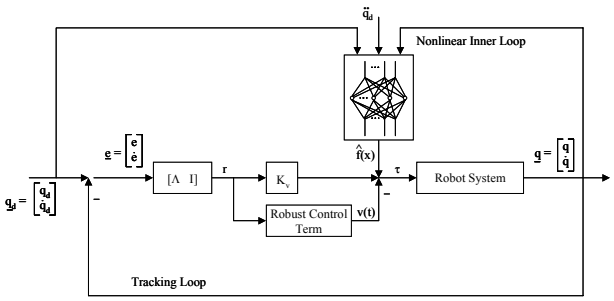

Fig. 4 Neural network robot controller

Now select the control input

$$\tau = \hat{W}^T\sigma(\hat{V}^T x) + K_v r - v \tag{4}$$

with K_v a symmetric positive definite gain and $v(t)$ a certain robustifying function detailed in the cited reference. 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* [Hunt, Su, and Meyer 1983], and the NN effectively learns the unknown dynamics on-line to cancel the nonlinearities of the system.

Let the estimated sigmoid jacobian be $\hat{\sigma}' \equiv d\sigma(z)/dz\,|_{z=\hat{V}^T x}$. 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

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

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

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

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

where the weight estimation errors are $\tilde{V} = V - \hat{V}, \quad \tilde{W} = W - \hat{W}$, with *tr{.}* 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 filtered tracking error $r(t)$ as well as the NN weights. Specific bounds on $r(t)$ and the NN weights are given in [Lewis, Jagannathan, and Yesildirek 1999]. The first terms of (4) are very close to the (continuous-time) backpropagation algorithm [Werbos 1974]. The last terms correspond to Narendra's e-modification [Narendra and Annaswamy 1987] extended to nonlinear-in-the-parameters adaptive control.

Robustness and Passivity of the NN When Tuned On-Line. Though the NN in Fig. 4 is static, since it is tuned on line 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 on-line to approximate the unknown function *f(x)*. It is not required to find a regression matrix. This is a consequence of the NN universal approximation property.

Single-layer Neural Network Controller

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

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

Then, the NN is linear-in-the-parameters and the tuning algorithms resembles those used in adaptive control. However, NN design still offer 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.

Feedback Linearization of Nonlinear Systems Using NN

Many systems of interest in industrial, aerospace, and DoD applications are in the affine form $\dot{x} = f(x) + g(x)u + d$, with $d(t)$ a bounded unknown disturbance, and 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 $u_c(t)$, plus an *extra robustifying part* $u_r(t)$. Now, *two NN* 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 $\hat{f}(x)$ 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 keep the control $u(t)$ finite.

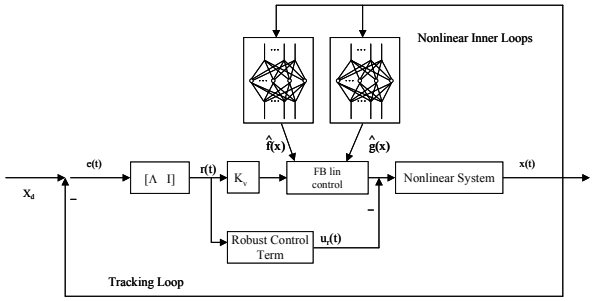

Fig. 5 Feedback linearization neural network controller

Partitioned Neural Networks 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 Neural Networks. 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

$$f(x) = M(q)\varsigma_1(x) + V_m(q,\dot{q})\varsigma_2(x) + G(q) + F(\dot{q})$$

with $\varsigma_1(x) = \ddot{q}_d + \Lambda\dot{e}, \quad \varsigma_2(x) = \dot{q}_d + \Lambda e$. Taking the four terms one at a time [Ge, Lee, and Harris 1998], 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*. 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 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 [Lewis, Dawson, Abdallah 2004], so that the vector *x* can be taken as

$$x = [\varsigma_1^T \ \varsigma_2^T \ (\cos q)^T \ (\sin q)^T \ \dot{q}^T \ \mathrm{sgn}(q)^T]^T$$

where the signum function is needed in the friction terms.

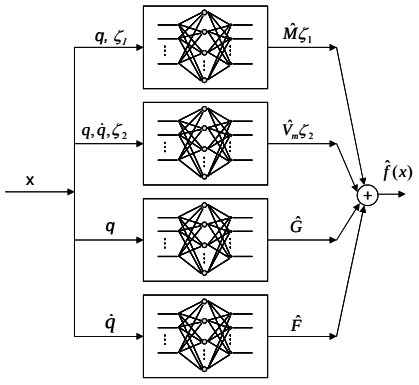

Fig. 6 Partitioned neural network

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 [Lewis 1992]. 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(.)* and *g(.)* 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 neural network controller with N layers, that guarantee system stability and robustness [Lewis, Jagannathan, and Yesildirek 1999]. For the i-th layer the weight updates are of the form

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

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

$$\hat{y}_i(k) \equiv \hat{W}_i^T(k)\hat{\varphi}_i(k) + K_v r(k), \quad \textit{for } i = 1, \cdots, N-1 \quad \textit{and} \quad \hat{y}_N(k) \equiv r(k+1), \quad \textit{for last layer}$$

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 successful extended to systems in strict-feedback form with a modified tuning law [Ge, Li, and Lee 2003].

Multi-loop Neural Network 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, etc. 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 [Lewis, Jagannathan, and Yesildirek 1999].

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 effect the motor torques. As a second-order effect, the torques effect the joint angles.

Backstepping NN Design. The NN tracking controller in Fig. 7 may be designed using the *backstepping* technique [Kanellakopoulos 1991]. 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 NN, 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.

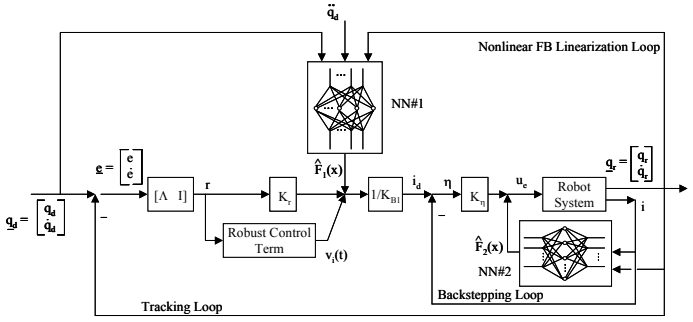

Fig. 7 Backstepping NN controller for robot with motor dynamics

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 NN provide a universal basis for the unknown functions encountered.

Compensation of Flexible Modes and High-Frequency Dynamics Using NN

Actual industrial or military mechanical systems may have additional dynamical complications such as vibratory modes, compliant couplings or gears, etc. 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

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

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 $B=[B_r^{\ T}\ B_f^{\ T}]^T$ is not square, but has more rows than columns. This means that while one is attempting to control the rigid modes 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 [Kokotovic 1994] 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

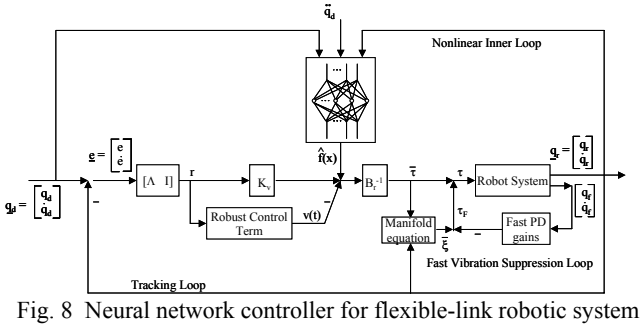

Fig. 8 Neural network controller for flexible-link robotic system

designed using a variety of techniques including H-infinity robust control and LQG/LTR. Such controllers are capable of compensating for the effects 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 perturbations 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.

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, etc. 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 NN 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

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

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 N.H. McClamroch [1988] 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.

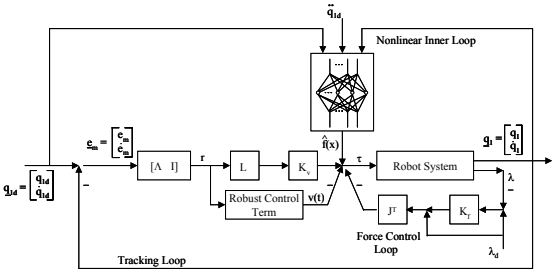

Fig. 9 NN/force/position controller

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 [Lewis, Campos, and Selmic 2002].

Feedforward Neurocontroller for Systems with Unknown Deadzone

Most industrial and 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 deadzone 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* NN. NN II is in the actual feedforward control loop, and NN I is not 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.

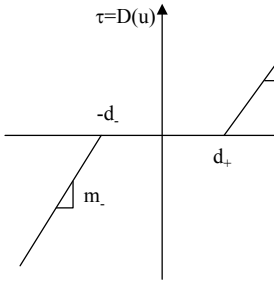

Fig. 10 Deadzone response characteristic

The two NN 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 [Igelnik and Pao 1995]. To guarantee stability the output weights of the inversion NN II and the estimator NN I should be tuned respectively as

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

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

where subscript '*i*' denotes weights and sigmoids of the inversion NN II and nonsubscripted variables 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 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 NN function as *a single NN with two layers*, first NN II then NN I, but with the second layer not in the control path. Note the additional terms, which are a combination of Narendra's e-mod and Ioannou's sigma mod.

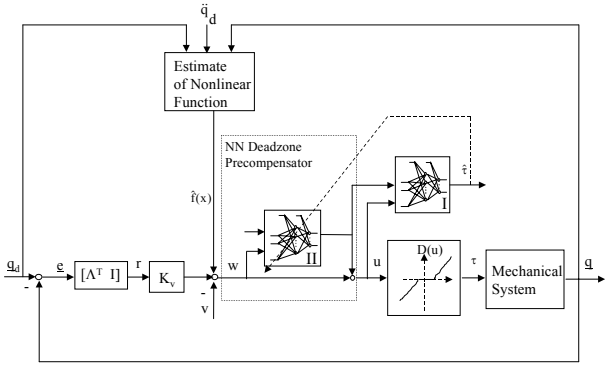

Fig. 11 Feedforward NN for deadzone compensation

Reinforcement Learning Structure. NN 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.

Dynamic Inversion Neurocontroller for Systems with Backlash

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

Dynamic inversion is a popular controller design technique in aircraft control and elsewhere [Stevens and Lewis 2003]. Dynamic inversion by NN has been used by Calise and coworkers in aircraft control using NN [2001]. Using dynamic inversion, a NN controller for systems with backlash is designed in [Selmic, Lewis, Calise 2000]. The neurocontroller appears in the feedforward loop as in Fig. 13, and is a *dynamic or recurrent NN.* In this neurocontroller, a desired

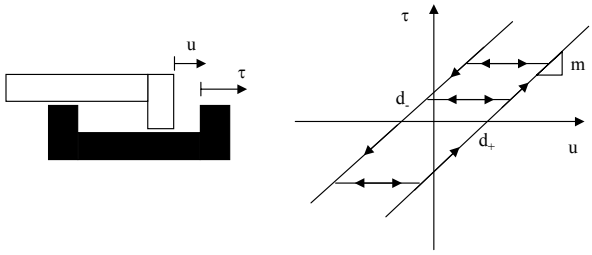

Fig. 12 Backlash response characteristic

torque $\tau_{des}(t)$ to be applied is determined, then, using a backstepping-type of approach [Kanellakopoulos 1991], the neurocontroller structure shown in the figure 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.

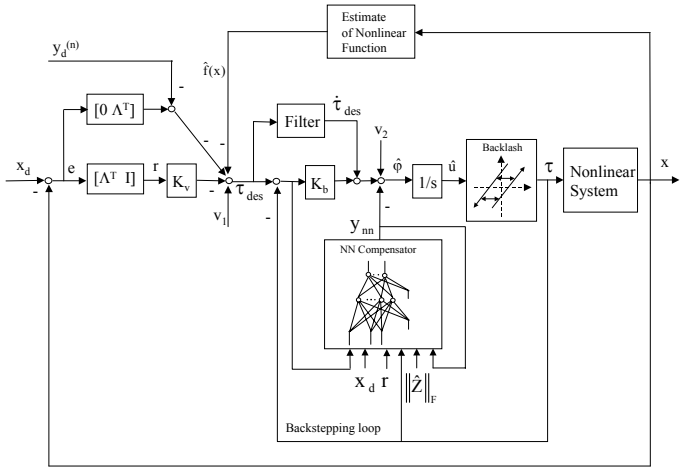

Fig. 13 Dynamic Inversion NN compensator for system with backlash

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

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

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

Neural Network 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 commercial 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 NN can be appreciated using the case of rigid robotic systems [Lewis, Dawson, Abdallah 2004]. 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 $x_1 \equiv q, x_2 \equiv \dot{q}$ 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 [Kim and Lewis 1998] that the following dynamic NN observer can provide estimates of the entire state $x = [x_1^T \; x_2^T]^T \equiv [q^T \; \dot{q}^T]^T$ given measurements of only $x_1(t) = q(t)$:

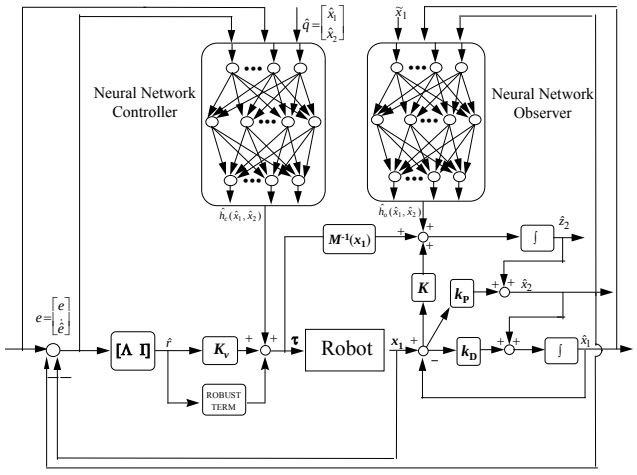

Fig. 14 NN Observer for Output-Feedback Control

$$\dot{\hat{x}}_1 = \hat{x}_2 + k_D x_1$$
$$\dot{\hat{z}}_2 = M^{-1}(x_1)[-\hat{W}_o^T \sigma_o(\hat{x}) + \tau + k_P \tilde{x}_1 + v_o].$$
$$\hat{x}_2 = \hat{z}_2 + k_{P2} \tilde{x}_1$$

In this system, hat denotes estimates and 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 $\sigma_o(.)$. 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 the references.

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 the references, weight tuning algorithms that guarantee stability are given for both the dynamic estimator NN and the static control NN.

Reinforcement Learning Control Using NN

Reinforcement learning techniques [Mendel 1970] 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, 'plus one' for a successful trial and 'negative one' 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, Munro, Williams, Barto [1991], Werbos [1992], etc.

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 [Johnson 1988] which has not been proven to yield stability.

Neural Network Reinforcement Learning Controller

A simple signal related to the performance of a robotic system is the signum of the filtered tracking error $R(t) = \text{sgn}(r(t))$, with the filtered tracking error given by $r = \dot{e} + \Lambda e$, where $e = q_d - q$ the tracking error and matrix Λ positive definite. Signal $R(t)$ satisfies the criteria required in reinforcement learning control: (i) It is simple, having values of only $0, \pm 1$, 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 [Kim and Lewis 1998] 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

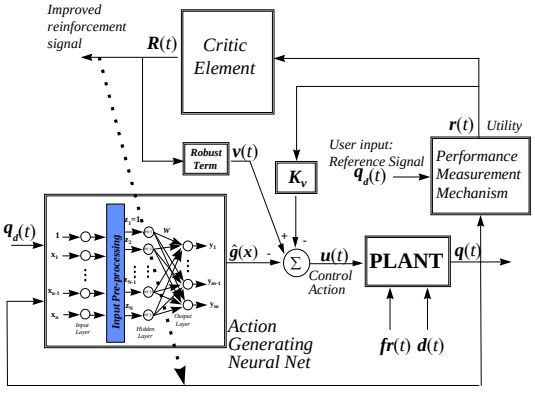

Fig. 15 Reinforcement Learning NN Controller

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*

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

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

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

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.

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 [1992]. Through the work of (see references) L.-X. Wang [1994], F.L. Lewis, K. Passino, S. Yurkovich, and others, it is now known how to tune fuzzy logic systems so that they learn on-line 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

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

where $W^T = [w_{kj}]$ is a matrix of output representative values and the fuzzy logic basis functions $\sigma_j(.)$ play the role of NN activation functions. Using product inferencing, the basis functions are given in terms of the 1-D membership functions (MF) $\mu_{ij}(x,U_{ij})$ 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. Albus [1975] CMAC NN), 2nd or 3rd degree polynomials, or the RBF functions [Sanner and Slotine]1998.

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 fuzzy logic system serves as a critic

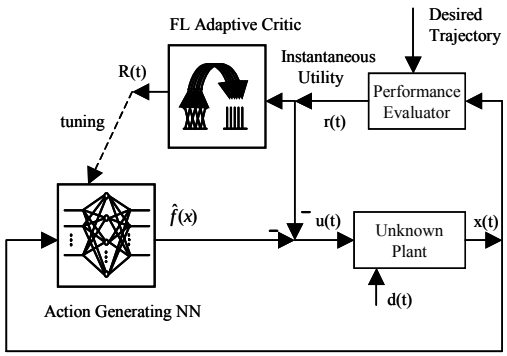

Fig. 16 Fuzzy logic adaptive reinforcement learning NN controller

and a neural network 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 on-line 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 R^n 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.

Optimal Control Using NN

Heretofore we have discussed the design of neural network 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 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 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 [Lewis and Syrmos 1995], solution techniques are available based on Riccati equation techniques, etc., 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 neural networks afford computationally effective techniques for solving general HJ equations, and so for designing closed-loop controllers for general nonlinear systems.

Neural Network *H-2* Control Using the Hamilton-Jacobi-Bellman Equation

In work by Abu-Khalaf and Lewis [2004] it has been shown how to solve the Hamilton-Jacobi-Bellman (HJB) that appears in optimal control for general nonlinear systems by a successive approximation (SA) technique based on neural networks. Rigorous results have been proven, and a computationally effective scheme for nearly optimal controller design was provided based on NN.

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

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

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

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

which defines the Hamiltonian function *H(.)* 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 et al. [1979].

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

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

Substituting this into the previous equation yields the Hamilton-Jacobi-Bellman equation of optimal control

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

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 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 successive approximation 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, e.g. the LQR for the linearization of (6). It has been shown by Saridis [1979] that the SA converges to the optimal solution V^*, u^* of the HJB. Let the region of asymptotic stability of the optimal solution be Ω^*, and the RAS at iteration *i* be $\Omega^{(i)}$. Then, in fact, it has been shown that:

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

$V^{(i)} \to V^*, \quad u^{(i)} \to u^*, \quad \Omega^{(i)} \to \Omega^*$ uniformly

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

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

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

Neural Networks for Computation of Successive Approximation Solution. It is difficult to solve equations (8) and (9) as required for the SA method just given. Beard and Saridis [1997] showed how to implement the SA algorithm using 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 [Abu-Khalaf and Lewis 2004] how to use NN 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

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

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

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

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, e.g. 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

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

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

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

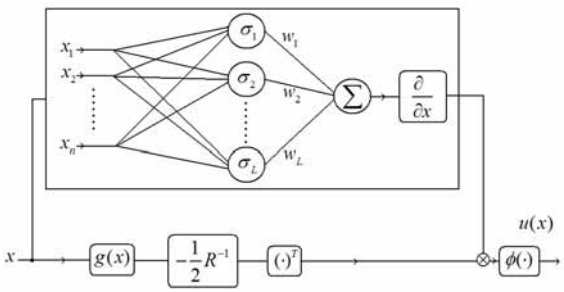

Fig. 17 Nearly Optimal NN Feedback control for constrained input nonlinear systems

Neural Network *H-Infinity* 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-infinity* 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 $\|z\|^2 = h^T h + \|u\|^2$. 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

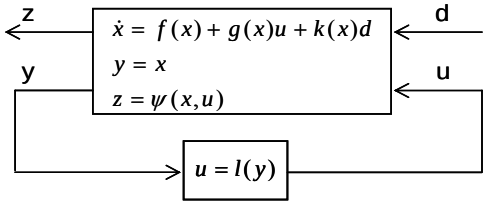

Fig. 18 Bounded L_2 gain problem

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

This is a differential game with two players [Knobloch and Isidori 1993, Basar], and can be confronted by defining the utility

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

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

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

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

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

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

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

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

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 Hamilton-Jacobi-Isaacs (HJI) equation

$$0 = \left(\frac{dV^*}{dx}\right)^T f + h^T h - \frac{1}{4}\left(\frac{dV^*}{dx}\right)^T g g^T \frac{dV^*}{dx} + \frac{1}{4\gamma^2}\left(\frac{dV^*}{dx}\right)^T k k^T \frac{dV^*}{dx} \qquad (15)$$

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 [Abu-Khalaf, Lewis, Huang 2004] 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 the equations (13) and (14) until convergence with respect to the disturbance. Now one selects an improved control using (12). The procedure repeats until 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 [Lewis and Syrmos 1995] on the linearized system dynamics.

Neural Network Solution of HJI Equation for H-Infinity Control. To implement this algorithm practically one may approximate the value at each step using a one-tunable-layer neural network as

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

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

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

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

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

and on control iterations the improved control is given by

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

This algorithm is shown to converge to the approximately optimal H-infinity solution. This yields a neural network feedback controller as shown in the figure above for the H_2 case.

Approximate Dynamic Programming and Adaptive Critics

Approximate Dynamic Programming 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 [Lewis and Syrmos 1995], which is a backwards-in-time procedure and so unsuitable for on-line implementation. ADP is based on using nonlinear approximators to solve the HJ equations forward in time, and was first suggested by P. Werbos [1991]. See the Historical Development Section for ADP below for cited work of major researchers. The current status of work in ADP is given in [Si, Barto, Powell, Wunsch 2004].

In the previous section was 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 backwards in time. This is unsuitable for real-time implementation and 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 [1960] showed that the following successive iteration scheme, known as *policy iteration*, converges to the optimal solution:

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 [1992] and others (see Historical Development and References Section) showed how to implement ADP controllers by four basic techniques, HDP, DHP, ADHDP, ADDHP, to be described next.

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

Critic NN $\quad V_j(x_k) \approx V(x_k, w_j)$

Action NN $\quad u_j(x_k) \approx u(x_k, v_j)$

HDP then proceeds as follows:

Critic Update

Find 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 RLS, backprop, etc., e.g.

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

Action Update

Find 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, etc., e.g.

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

This procedure is direct 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 the Werbos references.

Q-Learning or Action Dependent HDP. A function more advantageous than the value function for ADP is the Q function, defined by Watkins [1989] and Werbos [1989] 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

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

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 down the ADHDP algorithm in very straightforward manner. Since the control input action u_k is now explicitly an input ro 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 [Bradtke, Ydstie, and Barto 1994].

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

Historical Development, Referenced Work, and Further Study

A firm foundation for the use of neural networks in feedback control systems has been developed over the years by many researchers. Here is included a historical development, and references to the body of work in neurocontrol.

Neural Network for Feedback Control

The use of neural networks (NN) in feedback control systems was first proposed by Werbos [1989]. Since then, NN control has been studied by many researchers. Recently, NN 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 [Narendra and Lewis 2001]. Overviews of the initial work in NN control are provided by [Miller 1991] and the *Handbook of Intelligent Control* [White 1992], which highlighted a host of difficulties to be addressed for closed-loop control applications. Neural network 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 [Werbos 1974]. Basic problems that had to be addressed for closed-loop NN control [Werbos 1991, 1992] 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, providing efficient computer code for implementation. These issues have since been addressed by many approaches.

Initial work was in NN 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 [1991]. Off-line learning can yield important structural information.

Subsequent work in NN for control addressed closed-loop system structure and stability issues. Work by Sussmann [1992] and Sontag [Albertini 1992] was important in determining system properties of NN (e.g. minimality and uniqueness of the ideal NN weights, observability of dynamic NN). The seminal work of Narendra et al. [1990, 1991] 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 [1993] showed that knowing an approximate plant Jacobian is often good enough to guarantee suitable closed-loop performance.

The approximation properties of NN [Cybenko 1989, Park and Sandberg 1991] 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 [1992] used radial basis functions in control and showed how to select the NN basis functions, Polycarpou and Ioannou [1991, 1992] used a projection method for weight updates, Lewis [1995] used backprop with an e-mod term [Narendra and Annaswamy 1987]. This work used NN that are *linear* in the unknown parameter. In linear NN, the problem is relegated to determining activation functions that form a *basis set* (e.g. RBF [Sanner 1992], FLPN [Sadegh 1993]). Barron [1993] has shown that using NN 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 NN overcome this difficulty and were first used by F.-C. Chen [1992] who used backprop with deadzone weight tuning, and Lewis [1996], who used a Narendra's e-mod term in backprop. In nonlinear-in-the-parameters NN, the basis is automatically selected on-line by tuning the first-layer weights and thresholds. Multilayer NN were rigorously used for discrete-time control by Jagannathan and Lewis [1996]. Polycarpou [1996] derived NN controllers that do not assume

known bounds on the ideal weights. Dynamic/recurrent NN were used for control by Rovithakis and Christodoulou [1994], Poznyak [1999], Rovithakis [2000] who considered multiplicative disturbances, [Zhang and Wang 2001], and others.

Most stability results on NN control have been local in nature, and global stability has been treated by Annaswamy, by [Kwan, Dawson, Lewis 2001], and by 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 [Leitner 1997, McFarland 2000, Calise 2001] has used NN in conjunction with dynamic inversion to control aircraft and missiles. Feedback linearization using NN has been addressed by F.C. Chen [1992], Yesildirek and Lewis [1995], Zhang, Hang, and Ge [2000], and others. NN were used with backstepping [Kanellakopoulos 1991] by [Lewis, Jagannathan, and Yesildirek 1999], [Arslan and Basar 2001], [Gong 2001], [Wang and Huang 2001], and others.

NN have been used in conjunction with the Isidori-Byrnes regulator equations for output tracking control by [Wang and Huang 2001]. A multimodel NN control approach has been given by Narendra ([Narendra and Balakrishnan 1997]). Applications of NN control have been extended to partial differential equation systems by [Padhi, Balakrishnan, Randolph 2001]. NN have been used for control of stochastic systems by [Poznyak and Ljung 2001]. Parisini has developed receding horizon controllers based on NN [1998] and hybrid discrete event NN controllers [2001].

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 F.C. Chen [1996], Gutierrez and Lewis [1998], and others. Random initialization of the first-layer NN weights often works in practice, and work by Igelnik [1995] shows that it is theoretically defensible. Computational complexity makes NN with many hidden-layer neurons difficult to implement. Recently, work has intensified in wavelets, NN that have localized basis functions, and NN that are self-organizing in the sense of adding or deleting neurons automatically [Farrell 1998, Choi and Farrell 2000, Sanner and Slotine 1998, Y. Li 2001].

By now it is understood that NN offer an elegant extension of adaptive control and other techniques to systems that are nonlinear in the unknown parameters. The universal approximation properties of NN (Cybenko [1989]) 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 [Zbikowski and Hunt 1996], [Lewis, Jagannathan, Yesildirek 1999], [Kim and Lewis 1998], [Lewis, Campos, Selmic 2002], Ge et al. [1998, 2001].

Approximate Dynamic Programming

Adaptive critics are reinforcement learning designs that attempt to approximate dynamic programming [Barto 1991, 2004]. They approach the optimal solution through forward approximate dynamic programming. Initially, they were proposed by Werbos [1991]. Overviews of the initial work in NN control are provided by [Miller 1991] and the *Handbook of Intelligent Control* [White 1992]. Howard [1960] showed the convergence of an algorithm relying on 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 on-line dynamic programming based reinforcement learning algorithms emerged and were mainly based on Werbos' Heuristic Dynamic Programming (HDP) [1992], Sutton's Temporal Differences (TD) learning methods [1988], and Q-Learning, which was introduced by Watkins [1989] and Werbos [1989] (called 'action dependent' critic schemes there). Critic and action network tuning was provided by RLS, gradient techniques, or the backpropagation algorithm [Werbos 1974]. 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 [1992].

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 Werbos [1992], the HDP and dual heuristic programming (DHP) algorithms, and their action dependent versions- ADHDP and ADDHP. The Linear Quadratic Regulation (LQR) problem [Lewis and Syrmos 1995] 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, Ydstie, Barto [1994] where Q-Learning was shown to converge when using non-linear function approximators. An important persistence of excitation notion was included. Further work was done by Landelius [1997] 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 [1998] discussed the effect of model noise and exploration noise when adaptive critic is viewed as a stochastic approximation technique. Prokhorov and Feldkamp [1998] look at Lyapunov stability analysis. Other convergence results are due to Balakrishnan and coworkers [Liu and Balakrishnan 2000, Padhi et al. 2001], who have also studied optimal control of aircraft and distributed parameter systems governed by PDEs. Anderson [2001] showed convergence and stability for a reinforcement learning scheme. All of these results were done for the discrete-time case.

A thorough treatment of neuro dynamic programming is given in the seminal book by Bertsekas and Tsitsiklis [1996]. Various successful practical implementations have been reported, including aircraft control examples by Ferrari and Stengel [2002], an Auto Lander by Murray, Cox, Saeks, Lendaris [2001], state estimation using dynamic NN by Feldkamp and Prokhorov [2003], and Neuro-Observers by Balakrishnan and coworkers [Liu and Balakrishnan 2001]. J. Si et al. have provided analysis [2001] and applied ADP to aircraft control [2004]. An account of the Adaptive Critic Designs is found in Prokhorov and Wunsch [1997].

Applications of adaptive critics in the continuous-time domain were mainly done through discretization, and the application of the well-established discrete-time results (e.g. [Tsitsiklis 1995]). Various continuous-time non-dynamic reinforcement learning were discussed by Campos and Lewis [1999], and Rovithakis [2001], who approximated a Lyapunov function derivative. In Kim and Lewis [1998] the HJB equation of dynamic programming is approximated by a Riccati equation and a suboptimal controller based on neural network feedback linearization is implemented with full stability and convergence proofs. Murray, Cox, Lendaris, Saeks [2002] 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 approximated using either quadratic functions or an RBF neural network. Saridis [1979] showed the convergence of an off-line algorithm relying on 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 and Saridis [1997] 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 [1994], who defines a differential increment from the optimal solution and explicitly take into account the sampling interval Δt. Doya [2000] derives results for on-line 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 [1988]. P. Tsiotras has used Wavelets to find approximate solutions to HJB. Lyshevski [2001] 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.

Acknowledgements. The referenced work of Lewis and coworkers was sponsored by NSF grant ECS-01-40490 and ARO Grant DAAD19-02-1-0366.

References

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, vol. 28, pp. 239-251, 2004.

M. Abu-Khalaf, F.L. Lewis, and J. Huang, "Computational techniques for constrained nonlinear state feedback H-infinity optimal control using neural networks," Proc. Mediterranean Conf. Control and Automation, paper 1141, Kusadasi, Turkey, June 2004.

F. Albertini and E.D. Sontag, "For neural nets, function determines form," Proc. IEEE Conf. Decision and Control, pp. 26-31, Dec. 1992.

J.S. Albus, "A new approach to manipulator control: the cerebellar model articulation controller equations (CMAC)," Trans. ASME J. Dynam. Syst. Meas. Control, vol. 97, pp. 220-227, 1975.

C. Anderson, R.M. Kretchner, P.M. Young, and D.C. Hittle "Robust reinforcement learning control with static and dynamic stability," Int. J. Robust and Nonlinear Control, vol. 11, 2001.

G. Arslan and T. Basar, "Disturbance attenuating controller design for strict-feedback systems with structurally unknown dynamics," Automatica, vol. 37, no. 8, pp. 1175-1188, Aug. 2001.

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

R. Beard, G. Saridis, and J. Wen, "Approximate solutions to the time-invariant Hamilton-Jacobi-Bellman equation," Automatica, vol. 33, no. 12, pp. 2159-2177, Dec. 1997.

A.R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," IEEE Trans. Info. Theory, vol. 39, no. 3, pp. 930-945, May 1993.

A.G. Barto, "Connectionist learning for control," in *Neural Networks for Control*, Cambridge: MIT Press, 1991.

A.G. Barto and T.G. Dietterich, "Reinforcement learning and its relationship to supervised learning," in ed. Si, J., A. Barto, W. Powell, D. Wunsch, *Handbook of Learning and Approximate Dynamic Programming*, IEEE Press, USA, 2004.

L. von Bertalanffy, General System Theory, Braziller, New York, 1968.

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

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

A.J. Calise, N. Hovakimyan, and H. Lee, "Adaptive output feedback control of nonlinear systems using neural networks," Automatica, vol. 37, no. 8, pp. 1201-1211, Aug. 2001.

Campos, J., F. L. Lewis, "Adaptive Critic Neural Network For Feedforward Compensation", *Proceedings of the American Control Conference*, San Diego, CA, June 1999.

F.-C. Chen and H.K. Khalil, "Adaptive control of nonlinear systems using neural networks," Int. J. Control, vol. 55, no. 6, pp. 1299-1317, 1992.

F.-C. Chen and C.-H. Chang, "Practical stability issues in CMAC neural network control systems," IEEE Trans. Control Systems Technol., vol. 4, no. 1, pp. 86-91, Jan. 1996.

J.Y. Choi and J.A. Farrell, "Nonlinear adaptive control using networks of piecewise linear approximators," IEEE Trans. Neural Networks, vol. 11, no. 2, pp. 390-401, Mar. 2000.

G. Cybenko, "Approximation by superpositions of a sigmoidal function," Mathematics of Control. Signals and Systems, vol. 2, no. 4, pp. 303-314, 1989.

Dayan, P., The Convergence of TD(λ) for General λ, Machine Learning, v.8 n.3-4, p.341-362, May 1992.

Doya, K., "Reinforcement Learning in Continuous Time and Space," *Neural Computation*, vol. 12, pp. 219-245, MIT Press, 2000.

J.A. Farrell, "Stability and approximator convergence in nonparametric nonlinear adaptive control," IEEE Trans. Neural Networks, vol. 9, no. 5, pp. 1008-1020, Sept. 1998.

Feldkamp, Lee., D. Prokhorov, "Recurrent Neural Networks for State Estimation," *12th Yale Workshop on Adaptive and Learning Systems*, pp. 17-22, New Haven, CT, 2003.

Ferrari, S., R. Stengel, "An Adaptive Critic Global Controller," *Proceedings of the American Control Conference*, pp. 2665-2670, Anchorage, AK, 2002.

S.S. Ge, T.H. Lee, and C.J. Harris, Adaptive Neural Network Control of Robotic Manipulators, World Scientific, Singapore, 1998.

S.S. Ge, C.C. Hang, T.H. Lee, and T. Zhang, Stable Adaptive Neural Network Control, Kluwer, Boston, MA, 2001.

S.S. Ge, G.Y. Li and T.H. Lee, "Adaptive NN control for a class of strict feedback discrete-time nonlinear systems" Automatica, Vol. 39, pp. 807-819, 2003.

J.Q. Gong and B. Yao, "Neural network adaptive robust control of nonlinear systems in semi-strict feedback form," Automatica, vol. 37, no. 8, pp. 1149-1160, Aug. 2001.

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 Trans. Industrial Electronics, vol. 45, no. 2, pp. 307-318, April 1998.

Hagen, S., B. Kröse, "Linear quadratic regulation using reinforcement learning." In F. Verdenius and W. van den Broek, editors, *Proc. of the 8th Belgian-Dutch Conf. on Machine Learning, BENELEARN'98*, page 39-46, Wageningen, October 1998.

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

L.R. Hunt, R. Su., and G. Meyer, "Global transformations of nonlinear systems," IEEE Trans. Autom. Control, vol. 28, pp. 24-31, 1983.

B. Igelnik and Y.-H. Pao, "Stochastic choice of basis functions in adaptive function approximation and functional-link net," IEEE Trans. Neural Networks, vol. 6, no. 6, pp. 1320-1329, Nov. 1995.

S. Jagannathan and F.L. Lewis, "Multilayer discrete-time neural net controller with guaranteed performance," IEEE Trans. Neural Networks, vol. 7, no. 1, pp. 107-130, Jan. 1996.

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

I. Kanellakopoulos, P.V. Kokotovic, and A.S. Morse, "Systematic design of adaptive controllers for feedback linearizable systems," IEEE Trans. Automat. Control, vol. 36, pp. 1241-1253, 1991.

M. Kawato, "Computational schemes and neural network models for formation and control of multijoint arm trajectory, Neural Networks for Control, pp. 197-228, ed. W.T. Miller, R.S. Sutton, P.J. Werbos, Cambridge: MIT Press, 1991.

P.V. Kokotovic, "Applications of singular perturbation techniques to control theory," SIAM Review, Vol. 26, no. 4, pp. 501-550, 1984.

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

C. Kwan, D.M. Dawson, and F.L. Lewis, "Robust adaptive control of robots using neural network: global stability," Asian J. Control, vol. 3, no. 2, pp. 111-121, June 2001.

Y.D. Landau, Adaptive Control, Marcel Dekker, New York, 1979.

T. Landelius, "Reinforcement Learning and Distributed Local Model Synthesis," PhD Dissertation, Linköping University, 1997.

J. Leitner, A.J. Calise, and J.V.R. Prasad, "Analysis of adaptive neural networks for helicopter flight control," J. Guid., Control., and Dynamics, vol. 20, no. 5, pp. 972-979, Sept.-Oct. 1997.

F.L. Lewis, *Applied Optimal Control and Estimation: Digital Design and Implementation*, Prentice-Hall, New Jersey, TI Series, Feb. 1992.

F.L. Lewis, K. Liu, and A. Yesildirek, "Neural net robot controller with guaranteed tracking performance," IEEE Trans. Neural Networks, vol. 6, no. 3, pp. 703-715, 1995.

F.L. Lewis, A. Yesildirek, and K. Liu, "Multilayer neural net robot controller with guaranteed tracking performance, IEEE Trans. Neural Networks, vol. 7, no. 2, pp. 388-399, Mar. 1996.

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

F.L. Lewis, "Nonlinear network structures for feedback control," Asian J. Control, vol. 1, no. 4, pp. 205-228, Dec. 1999.

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

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, 2002.

F.L. Lewis, D.M. Dawson, and C.T. Abdallah, Robot Manipulator Control, Marcel Dekker, New York, 2004.

Y. Li, N. Sundararajan, and P. Saratchandran, "Neuro-controller design for nonlinear fighter aircraft maneuver using fully tuned neural networks," Automatica, vol. 37, no. 8, pp. 1293-1301, Aug. 2001.

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

X. Liu, S. N. Balakrishnan, "Convergence Analysis of Adaptive Critic Based Optimal Control", *Proceedings of the American Control Conference*, pp. 1929-1933, Chicago, IL, 2000.

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

N.H. McClamroch and D. Wang, "Feedback stabilization and tracking of constrained robots," IEEE Trans. Automat. Control, vol. 33, pp. 419-426, 1988.

M.B. McFarland and A.J. Calise, "Adaptive nonlinear control of agile anti-air missiles using neural networks," IEEE Trans. Control Systems Technol., vol. 8, no. 5, pp. 749-756, Sept. 2000.

J.M. Mendel and R.W. MacLaren, "Reinforcement learning control and pattern recognition systems," in Adaptive, Learning, and Pattern Recognition Systems: Theory and Applications, ed. J.M. Mendel and K.S. Fu, pp. 287-318, Academic Press, New York, 1070.

W.T. Miller, R.S. Sutton, P.J. Werbos, ed., Neural Networks for Control, Cambridge: MIT Press, 1991.

H. Miyamoto, M. Kawato, T. Setoyama, and R. Suzuki, "Feedback-error-learning neural network for trajectory control of a robotic manipulator," Neural Networks, vol. 1, pp. 251-265, 1988.

Murray, J., C. Cox, G. Lendaris, R. Saeks, "Adaptive Dynamic Programming," *IEEE Transaction on Systems, Man, and Cybernetics*, vol. 32, no. 2, may 2002.

J. Murray, C. Cox, R. Saeks, and G. Lendaris, "Globally convergent approximate dynamic programming applied to an autolander," Proc. ACC, pp. 2901-2906, Arlington, VA, 2001.

K.S. Narendra, "Adaptive Control of dynamical systems using neural networks," Handbook of Intelligent Control, pp. 141-183, ed. D.A. White and D.A. Sofge, New York: Van Nostrand Reinhold, 1992.

K.S. Narendra and A.M. Annaswamy, "A new adaptive law for robust adaptation without persistent excitation," IEEE Trans. Automat. Control, vol. AC-32, no. 2, pp. 134-145, Feb. 1987.

K.S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," IEEE Trans. Neural Networks, vol. 1, pp. 4-27, Mar. 1990.

K.S. Narendra and K. Parthasarathy, "Gradient methods for the optimization of dynamical systems containing neural networks," IEEE Trans. Neural Networks, vol. 2, no. 2, pp. 252-262, Mar. 1991.

K.S. Narendra and J. Balakrishnan, "Adaptive control using multiple models," IEEE Trans. Automatic Control, vol. 42, no. 2, pp. 171-187, Feb. 1997.

K.S. Narendra and F.L. Lewis, Special Issue on Neural Network feedback Control, Automatica, vol. 37, no. 8, Aug. 2001.

R. Padhi, S.N. Balakrishnan, and T. Randolph, “Adaptive-critic based optimal neuro control synthesis for distributed parameter systems,” Automatica, vol. 37, no. 8, pp. 1223-1234, Aug. 2001.

J. Park and I.W. Sandberg, ``Universal approximation using %radial-basis-function networks," Neural Comp., vol. 3, pp. 246-257, 1991.

T. Parisini, M. Sanguineti, and R. Zoppoli, “Nonlinear stabilization by receding-horizon neural regulators,” Int. J. Control, vol. 70. pp. 341-362, 1998.

T. Parisini and S. Sacone, “Stable hybrid control based on discrete-event automata and receding-horizon neural regulators,” Automatica, vol. 37, no. 8, pp. 1279-1292, Aug. 2001.

M.M. Polycarpou and P.A. Ioannou, “Identification and control using neural network models: design and stability analysis,” Tech. Report 91-09-01, Dept. Elect. Eng. Sys., Univ. S. Cal., Sept. 1991.

M.M. Polycarpou and P.A. Ioannou, “Neural networks as on-line approximators of nonlinear systems,” Proc. IEEE Conf. Decision and Control, pp. 7-12, Tucson, Dec. 1992.

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

A.S. Poznyak, W. Yu, E.N. Sanchez, and J.P. Perez, “Nonlinear adaptive trajectory tracking using dynamic neural networks,” IEEE Trans. Neural Networks, vol. 10, no. 6, pp. 1402-1411, Nov. 1999.

A.S. Poznyak and L. Ljung, “On-line identification and adaptive trajectory tracking for nonlinear stochastic continuous time systems using differential neural networks,” Automatica, vol. 37, no. 8, pp. 1257-1268, Aug. 2001.

Prokhorov, D., and D. Wunsch, “Adaptive Critic Designs,” *IEEE Transactions on Neural Networks,* vol. 8, no. 5, September 1997.

D.V. Prokhorov amd L.A. Feldkamp, “Analyzing for Lyapunov stability with adaptive critics,” Proc. Int. Conf. Systems, Man, Cybernetics, pp. 1658-1661, Dearborn, 1998.

G.A. Rovithakis and M.A. Christodoulou, “Adaptive control of unknown plants using dynamical neural networks,” IEEE Trans. Systems, Man, and Cybernetics, vol. 24, no. 3, pp. 400-412, 1994.

G.A. Rovithakis, “Performance of a neural adaptive tracking controller for multi-input nonlinear dynamical systems,” IEEE Trans. Systems, Man, Cybern. Part A, vol. 30, no. 6, pp. 720-730, Nov. 2000.

G.A. Rovithakis, “Stable adaptive neuro-control via Lyapunov function derivative estimation,” Automatica, vol. 37, no. 8, pp. 1213-1221, Aug. 2001.

N. Sadegh, “A perceptron network for functional identification and control of nonlinear systems,” IEEE Trans. Neural Networks, vol. 4, no. 6, pp. 982-988, Nov. 1993.

R.M. Sanner and J.-J.E. Slotine, “Gaussian networks for direct adaptive control,” IEEE Trans. Neural Networks, vol. 3, no. 6, pp. 837-863, Nov. 1992.

R.M. Sanner and J.-J. E. Slotine, “Structurally dynamic wavelet networks for adaptive control of robotic systems,” Int. J. Control, vol. 70, no. 3, pp. 405-421, 1998.

G. Saridis and C.S. Lee, “An approximation theory of optimal control for trainable manipulators,” IEEE Trans. Systems, Man, Cybernetics, vol. 9, no. 3, pp. 152-159, March 1979.

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

J. Si and Y.-T. Wang, “On-line control by association and reinforcement,” IEEE Trans. Neural Networks, vol. 12, no. 2, pp. 264-276, Mar. 2001.

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

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

H.J. Sussmann, “Uniqueness of the weights for minimal feedforward nets with a given input-output map,” Neural Networks, vol. 5, pp. 589-593, 1992.

Sutton, R., "Learning to predict by the method of temporal differences," *Machine Learning*, 3:9-44, 1988.

J.N. Tsitsiklis, "Efficient algorithms for globally optimal trajectories," IEEE Trans. Automatic Control, vol. 40, no. 9, pp. 1528-1538, Sept. 1995

D. Wang and J. Huang, "Neural network based adaptive tracking of uncertain nonlinear systems in triangular form," preprint, 2001.

J. Wang and J, Huang, "Neural network enhanced output regulation in nonlinear systems," Automatica, vol. 37, no. 8, pp. 1189-1200, Aug. 2001.

L.-X. Wang, Adaptive Fuzzy Systems and Control: Design and Stability Analysis, Prentice-Hall, New Jersey, 1994.

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

P.J. Werbos, Beyond Regression: New Tools for Prediction and Analysis in the Behavior Sciences, Ph.D. Thesis, Committee on Appl. Math. Harvard Univ., 1974.

P.J. Werbos, "Neural networks for control and system identification," Proc. IEEE Conf. Decision and Control, Fla., 1989.

P.J. Werbos., "A menu of designs for reinforcement learning over time," , Neural Networks for Control, pp. 67-95, ed. W.T. Miller, R.S. Sutton, P.J. Werbos, Cambridge: MIT Press, 1991.

P.J. Werbos, "Approximate dynamic programming for real-time control and neural modeling," Handbook of Intelligent Control, ed. D.A. White and D.A. Sofge, New York: Van Nostrand Reinhold, 1992.

D.A. White and D.A. Sofge, ed. Handbook of Intelligent Control, New York: Van Nostrand Reinhold, 1992.

A. Yesildirek and F.L. Lewis, "Feedback linearization using neural networks," Automatica, pp. 1659-1664, vol. 31., no. 11, Nov. 1995.

T. Zhang, C.C. Hang, and S.S. Ge, "Robust adaptive controller for general nonlinear systems using multilayer neural networks," preprint, 2001.

Y. Zhang and J. Wang, "Recurrent neural networks for nonlinear output regulation," Automatica, vol. 37, no. 8, pp. 1161-1173, Aug. 2001.

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