Control of Nonlinear Dynamical Systems
Using Neural Networks: Controllability
and Stabilization
Asriel U. Levin and Kumpati S. Narendra, Fellow, IEEE

Abstract—The main objective of this paper is to indicate how practically viable controllers can be designed using neural networks, based on results in nonlinear control theory. The paper considers the problem of stabilization of a dynamical system around an equilibrium point, when the state of the system is accessible. Simulation results are included to complement the theoretical discussions.

I. INTRODUCTION

Since the early 1980’s, there has been an explosive growth in pure and applied research related to neural networks. During this period, the multilayer feedforward neural network (MNN) was introduced [18] and immediately found wide application in many fields. These applications were primarily in the area of pattern recognition and hence pertain to static systems. Since dynamics constitute an essential part of all physical systems, it was suggested by one of the authors [15] that neural networks should be used as components in dynamical systems. Since that time the study of neural networks in a dynamical context has been in progress at Yale University within a unified framework of systems theory. During the past three years, numerous models for the practical identification of nonlinear dynamical systems were proposed and later used for the design of controllers [15]. Extensive simulation studies carried out have shown that the models proposed are particularly effective for the identification and control of nonlinear systems.

In spite of the advances, much of the above work is of a heuristic nature. The success achieved in the simulation studies using such models has now generated new interest in relating the theory of nonlinear control to the synthesis techniques proposed at Yale University. This paper presents a first attempt to relate the experimental studies to theoretical developments and one of our aims is to propose a general methodology by which control methods based on neural networks can be made more rigorous.

From a mathematical point of view, even the control of known nonlinear dynamical systems is a formidable problem. This becomes substantially more complex when the representation of the system is not completely known. In such cases, stringent assumptions have to be made if neural networks are to be used for their identification and control. Further, while several theoretically elegant results have been reported in the control literature, many of them are not directly applicable to the practical identification and control of nonlinear systems. Hence our objective in this paper is threefold: The first is to provide the reader with some background concerning concepts in control theory on which many of the results proposed are based. The second objective is to present theoretical results along with the assumptions they entail, which can be used in the practical synthesis of identifiers and controllers. It is hoped that these assumptions can be relaxed as our experience with such problems increases. A third and final objective is to demonstrate, through simulation studies, the effectiveness of the proposed methods.

A. Adaptive Control of Nonlinear Systems Using Neural Networks—Prior Information

Consider the discrete time dynamical system described by the equations:

\[
\begin{align*}
x(k+1) &= f[x(k), u(k)] \\
y(k) &= h[x(k)],
\end{align*}
\]

(1)

where \(x(k) \in \mathcal{X} \subseteq \mathbb{R}^n\), \(u(k) \in \mathcal{U} \subseteq \mathbb{R}^r\), and \(y(k) \in \mathcal{Y} \subseteq \mathbb{R}^m \cdot u(k), x(k), \) and \(y(k)\) represent, respectively the input, state and the output of the system at time \(k\).

In the control problem the objective is to determine the input \(u(k)\) so that the system behaves in a desired fashion. As mentioned in the introduction, even when the nonlinear functions \(f\) and \(h\) are known, this problem is a difficult one. It is well known in the field of nonlinear control that, for solutions of (1) to exist, \(f\) and \(h\) must satisfy various conditions which may be quite involved and hard to verify [4], [17]. In the adaptive control problem, \(f, h\) are assumed to be unknown and hence the problem is substantially more complex. To make the problem tractable, we need to make certain assumptions concerning the controllability and observability properties of the system (and hence concerning \(f\) and \(h\)). Even in the simpler case of adaptive control of linear time-invariant systems, prior information concerning the system was assumed to obtain a solution (for example, order of the system, relative degree, high frequency gain, etc., are assumed to be known). Hence, our objective is to make such assumptions concerning the plant which are sufficient for the methods proposed to
be applied and gradually relax them as was done in linear adaptive control theory.

**B. Scope of Paper**

The problem of controlling a plant, can be conveniently divided into the regulation and tracking problems. In the former, the main objective is to stabilize the plant around a fixed operating point. In the latter, the aim is to make the output of the plant follow a specified signal asymptotically. While our ultimate goal is to determine the control input \( u \) based only on output measurements for both regulation and tracking [10], [11], we will confine our attention in this paper to the problem of regulation when the state of the system is accessible. This implies that our interest is only in the system described by the first part of (1), i.e.,

\[
\sum : x(k + 1) = f[x(k), u(k)].
\]

This paper is organized as follows: In Section I-A the general philosophy of the approach used to deal with the adaptive control problem of nonlinear systems is briefly discussed. Section II presents mathematical preliminaries and is devoted to concepts and definitions as well as mathematical theorems which will be used throughout the paper. In Section III the problem is formally stated, and in Section IV the control problem is treated in detail. In all cases the architectures chosen for the controller is based on results in nonlinear control theory. Simulation results are presented to complement the theoretical discussions.

**II. MATHEMATICAL PRELIMINARIES AND DEFINITIONS**

In this section, many of the basic concepts and definitions in control theory, as well as some theorems from mathematical analysis, which will be used throughout this paper are introduced. A brief introduction to the class of neural networks used to model nonlinear dynamical systems is also given towards the end of the section.

**A. System Theory Framework**

**Definition 1:** A point \( \bar{x} \) is an equilibrium of (2) if there exists an input \( \bar{u} \) s.t. \( \bar{x} = f(\bar{x}, \bar{u}) \).

We assume that every dynamical system has at least one equilibrium state and that, with no loss of generality, both \( \bar{x} \) and \( \bar{u} \) can be chosen to be zero. Throughout the rest of the paper we shall consider the origin as an equilibrium state.

A basic concept in system theory is controllability, which concerns the ability to influence the state of a dynamical system through the application of inputs.

**Definition 2:** A dynamical system is controllable if, for any two states \( x_1 \) and \( x_2 \), there exists an input sequence of finite length that will transfer the system from \( x_1 \) to \( x_2 \).

For nonlinear systems, conditions for global controllability are very hard to establish and verify, and thus we confine our attention to local notions.

**Definition 3:** A system is locally controllable around an equilibrium state \( x = 0 \) if, for every neighborhood \( V \) of the origin, there is some neighborhood \( W \) of the origin such that for any two states \( x_1, x_2 \in W \), there exists an input sequence of finite length that will transfer the system from \( x_1 \) to \( x_2 \) without leaving \( V \).

The notion of controllability merely assures that a control input \( u \) exists which can transfer the system from one state to another in a finite number of steps. Such a control input can be either a function of \( k \) or a function of the state of the system at time \( k \). The former is referred to as open loop control. If \( x(k_0) = x_1 \) and \( x(k_T) = x_2 \) the open loop control input \( u(k) \) is computed only from a knowledge of \( x_1, x_2, k_0, \) and \( k_T \). Since such a control input at any time \( k \) for \( k_0 < k < k_T \) is not explicitly determined by the actual state of the system at that instant, it follows that a system using an open loop controller can be sensitive to noise and external disturbances. In contrast to this, feedback control, in which \( u \) is a function of the state, is robust with respect to such disturbances.

When feedback control is used, i.e., \( u = g(x) \), system (2) becomes autonomous and is described by

\[
x(k + 1) = f[x(k), g(x(k))] = f[x(k)].
\]

The choice of the feedback law \( u = g(x) \) consequently depends upon the behavior expected of the controlled system (3).

**Definition 4:** Let \( x = 0 \) be an equilibrium point of (3). Then the origin is a stable equilibrium if for every neighborhood \( V \) of the origin there is a neighborhood \( W \subset V \) of zero such that every solution \( x(k) \) with \( x(0) \in W \) is in \( V \) for all \( k > 0 \).

If \( W \) can be chosen so that, in addition to the properties described above, \( \lim_{k \to \infty} x(k) = 0 \), then the origin is asymptotically stable. If this can be achieved in a finite number of steps then \( W \) is finitely (n-step) stable with respect to the origin. When \( W \) consists of the whole space, then the origin is globally asymptotically stable.

Using the above notion of stability, a second fundamental property of a system can be defined as follows.

**Definition 5:** If there exists a feedback law that makes an equilibrium point \( x = 0 \) asymptotically stable, then the system is stabilizable around that point.

It is well known for linear time invariant systems that controllability and stabilizability are closely related. In particular, controllability implies stabilizability. We will show that under suitable assumptions, similar results can be derived locally for nonlinear systems.

**B. Results from Analysis**

The ensuing results rely on some theorems from analysis which we will merely state here for easy reference [6]. In the following \( E, F, G \) will denote normed vector spaces and \( x \in E, y \in F, \) and \( z \in G \) will denote generic points in them.

**Theorem 1:** [Inverse Function Theorem] Let \( U \) be open in \( E \), let \( a \in U \), and let \( f : U \to F \) be \( C^p \) (i.e., continuous \( p \) derivative) map. Assume that the Jacobian \( Df(a) : E \to F \) is invertible. Then \( f \) is locally \( C^p \)-invertible at \( a \).

**Theorem 2:** [Implicit Function Theorem] Let \( U \) be open in the product \( E \times F \) and let \( \eta : U \to G \) be \( C^p \) map. Let \( (a, b) \) be a point of \( U \) with \( a \in E \) and \( b \in F \). Let \( f(a, b) = 0 \). Assume that the Jacobian of \( f \) with respect to \( y \) denoted by
$D_2 f(a, b) : F \rightarrow G$ is invertible. Then there is an open ball $V$ centered at $a$ in $E$ and a continuous map $g : V \rightarrow F$ such that $g(a) = b$ and $f[x, g(x)] = 0$ for all $x \in V$. If $V$ is a sufficiently small ball, then $g$ is uniquely determined and is of class $C^0$.

**Example 1:** Let $f(x, y) = x^2 + y^2 - 1$. The pairs $x, y$ for which $f(x, y) = 0$ form a circle in $\mathbb{R}^2$. In this example $(\partial f)/\partial x = 2y$ and from Theorem 2 if we choose a point $(a, b)$ such that $a^2 + b^2 = 1$ and $b \neq 0$ locally around $(a, b)$ there exists a function $g(x)$ such that $f[x, g(x)] = 0$. For example choose $a = 0$, $b = 1$ (or any other point on the upper semicircle with $b \neq 0$), then $g(x) = \sqrt{1 - x^2}$.

The inverse and implicit function theorems, play a central role in the results derived in this paper. As will be seen in Section IV, these theorems are used to determine the control input explicitly as a function of the state.

**Definition 6:** Let $E$ and $F$ be normed vector spaces and let $D \subseteq E$. Let $T$ be an operator $T : D \rightarrow F$. If there is a constant $c$ such that

$$
|T(x_1) - T(x_2)| \leq c|x_1 - x_2| \quad \text{for all } x_1, x_2 \in D
$$

then $T$ is said to satisfy a Lipschitz condition. $T$ is a contraction mapping if $c < 1$.

**Theorem 3 (Contraction Mapping Theorem):** Let $D$ be a closed subset of the normed vector space $E$. Let $T : E \rightarrow E$ be a contraction mapping on $D$. Then there exists a unique fixed point $\hat{x} \in D$ such that $T(\hat{x}) = \hat{x}$. Furthermore for all $x \in D \lim_{n \rightarrow \infty} T^n(x) = \hat{x}$.

**C. Liapunov Theory**

A very important tool in the stability analysis of dynamical systems is given by Liapunov's theory [7], [8].

**Definition 7:** A function $V(x)$ is said to be positive definite in a region $W$ containing the origin if (i) $V(0) = 0$ and (ii) $V(x) > 0$ for all $x \in W$, $x \neq 0$.

**Definition 8:** Let $W$ be any set in $\mathbb{R}^n$ containing the origin and $V : \mathbb{R}^n \rightarrow \mathbb{R}$. We say that $V$ is a Liapunov function of the equation

$$
x(k + 1) = f[x(k)]
$$

on $W$ if (i) $V$ is continuous on $\mathbb{R}^n$. (ii) $V$ is positive definite with respect to the origin in $W$. (iii) $\Delta V(k) \equiv V[x(k + 1)] - V[x(k)] \leq 0$ along the trajectory of (4) for all $x \in W$.

The existence of a Liapunov function assures stability as given by the following theorem and its corollary.

**Theorem 4 (Liapunov's Stability Theorem):** If $V$ is a Liapunov function of (4) in some neighborhood of an equilibrium state $x = \hat{x}$, then $x = \hat{x}$ is a stable equilibrium.

**Corollary 1:** If in addition $-\Delta V$ is positive definite with respect to $x = 0$, then the origin is asymptotically stable.

So far, the definitions of stability and asymptotic stability are in terms of perturbations of initial conditions. When we model a real world process, using neural networks or any other means, we cannot expect to have a perfect model of the real system and thus if the real system is described by (4) our model will be given by

$$
x(k + 1) = f[x(k)] + e[x(k)]
$$

where $e[x(k)] = f[x(k)] - f[x(k)]$ is a model error. For the system (2) $e = e[k, x(k)]$ and depends explicitly on $k$.

If this model error $e$ is small, one hopes that at least qualitatively, the behavior of the original system and that of the perturbed one will be similar. For the exact relation, stability under perturbations needs to be defined [20].

**Definition 9:** Let $x(x_0, k)$ denote the solution of (4) with the initial condition $x_0 = x(x_0, 0)$. The origin $x = 0$ is said to be stable under perturbations if for all $\epsilon > 0$ there exists $\delta_1(\epsilon)$ and $\delta_2(\epsilon)$ such that $\|x_0\| < \delta_1$ and $\|e(k, x)\| < \delta_2$ for all $k > 0$ imply $\|x(x_0, k)\| < \epsilon$ for all $k \geq 0$.

If, in addition, for all $\epsilon > 0$ there is an $r$ and a $K(\epsilon)$ such that $\|x_0\| < r$ and $\|e(k, x)\| < K(\epsilon)$ for all $k \geq 0$ imply $\|x(x_0, k)\| < \epsilon$ for all $k \geq K(\epsilon)$, the origin is said to be strongly stable under perturbations.

With this in mind the following theorem [8], [20] can be stated.

**Theorem 5:** If $f$ is Lipschitz continuous in a neighborhood of the equilibrium, then the system (4) is strongly stable under perturbations iff it is asymptotically stable.

This last result will play a central role in the use of neural networks for the control of nonlinear dynamical systems and is frequently referred to in the following sections.

**D. Artificial Neural Networks**

In this paper the term neuron will refer to an operator which maps $\mathbb{R}^n \rightarrow \mathbb{R}$ and is explicitly described by the equation:

$$
y = \Gamma \left( \sum_{j=1}^{n} w_{j} y_j + w_0 \right)
$$

where $U^T = [y_1, y_2, \ldots, y_n]$ is the input vector, $W^T = [w_{1}, w_{2}, \ldots, w_{n}]$ is referred to as the weight vector of the neuron and $w_0$ is termed its bias. $\Gamma(\cdot)$ is a monotone continuous function $\Gamma : \mathbb{R} \rightarrow (-1, 1)$ (commonly referred to as a "sigmoidal function", e.g., $\tanh(\cdot)$).

A set of interconnected neurons is a neural network. If the neurons are organized in layers $l = 0, 1, \ldots, L$ and a neuron at layer $l$ receives its inputs only from neurons in the $l - 1$ layer, the network will be called a feedforward neural network.

The output of the $i$th element in layer $l$ is given by

$$
y^l_i = \Gamma \left( \sum_{j=1}^{d} w_{i,j}^l y_j^{l-1} + w_{i,0}^l \right)
$$

where $W^{lT} = [w_{i,0}^l, w_{i,1}^l, \ldots, w_{i,n_{l-1}}^l]$ is the weight vector associated with the $i$th neuron at the $l$th layer. It is common practice to refer to the $l = 0$ layer as the input layer, to the $l = L$ layer as the output layer and to all others as hidden layers. The biases $w_{i,0}^l$ can be treated as additional weights from a neuron with output always at one.

A neural network, as defined above, represents a specific family of parametrized maps. If there are $n_0$ input elements and $n_L$ output elements, the network defines a continuous mapping $NN : \mathbb{R}^{n_0} \rightarrow \mathbb{R}^{n_L}$. To enable this map to be surjective (onto), we will choose the output layer to be linear.
Following the notation introduced in [15], such a family of networks with \( n_l \) neurons at the \( l \)-th layer will be denoted by

\[
N_{n_0, n_1, \ldots, n_L}^L.
\]

For example, a network with 2 inputs, 3 neurons at the first hidden layer, 5 in the second and 1 output unit will be described by \( N_{2,3,5,1}^3 \).

Two facts make the networks defined above powerful tools for approximating functions.

1) **Multilayer feedforward neural networks are universal approximators**: It was proved by Cybenko [1] and Hornik et al. [3] that any continuous mapping over a compact domain can be approximated as accurately as necessary by a feedforward neural network with one hidden layer. This implies that given any \( \epsilon > 0 \) a neural network with sufficiently large number of nodes can be determined such that

\[
||f(x) - NN(x)|| < \epsilon \quad \text{for all} \quad x \in D
\]

where \( f \) is the function to be approximated and \( D \) is a compact domain of a finite dimensional normed vector space.

The **backpropagation algorithm**: This algorithm, [18], which performs gradient descent, provides an efficient method to train a feedforward neural network to approximate a given continuous function over a compact domain \( D \).

Let \( u \in D \) be a given input. The network approximation error for this input is given by

\[
e(u) = f(u) - NN(u).
\]

Training \( NN(\cdot) \) to closely approximate \( f \) over \( D \) is equivalent to minimizing

\[
I = \int_D ||e(u)|| \, du.
\]

The training procedure for the network is carried out as follows: The network is presented with a sequence of training data (input-output pairs). Let \( \theta \) denote a generic parameter (or weight) of the network. Following each training example, the weights of the network are adjusted according to

\[
\theta(k+1) = \theta(k) - \eta(k) \frac{\partial I}{\partial \theta} \bigg|_{\theta = \theta(k)}.
\]

Stochastic approximation theory [12] guarantees that, if the step size \( \eta(k) \) satisfies certain conditions, \( f \) will converge to a local minimum w.p.1. If the performance surface is unimodal, this implies that the global minimum is achieved.

The backpropagation algorithm provides a recursive method to calculate these gradients in layered networks. That is, the partial derivatives with respect to the weights at the \( l \)-th layer can be calculated recursively from the ones of the \( l-1 \) layer (for details see [18], [16]). Weight adjustments can be performed at each time step or in a batch mode. In the latter case, the error function depends upon the errors due to a finite set of input vectors.

Once feedback connections are introduced into the network, thus making it recurrent, its behavior can no longer be described in terms of a static mapping from the input to the output space. Rather, its output will exhibit complex temporal behavior that depends on the current states of the neurons as well as the inputs. To make such a feedforward connection physically meaningful, a delay (denoted by \( z^{-1} \)) must be attached to it, thereby avoiding algebraic loops. Thus from the system theoretic point of view, the states of the system, as defined earlier, consist only of those neurons which possess a delay at their output.

A natural performance criterion for the recurrent network would be the summation of the square of the error between the sequence we want the network to follow, denoted by the vector process \( y(k) \), and the outputs of the network denoted by \( \hat{y}(k) \):

\[
I(k) = \sum_k ||y(k) - \hat{y}(k)||^2 \equiv \sum_k ||e(k)||^2.
\]

In the same manner that a feedforward layered network can be trained to emulate a static mapping, a training algorithm called **dynamic back propagation** [16], [24], [25] has been proposed to train a recurrent network to follow a temporal sequence. The Dynamic Back Propagation Algorithm is based on the fact that the dependence of the output of a dynamical system on a parameter is itself described by a recursive equation. The latter in turn contains terms which depend both explicitly and implicitly on the parameter. The application of dynamic back propagation in a specific context is treated in Section IV-B. For a detailed review of the algorithm the reader is referred to [16].

### III. Statement of the Problem

Consider the nonlinear discrete time dynamical systems described by (2):

\[
x(k+1) = f(x(k), u(k))
\]

where \( x(k) \in \mathcal{X} \) and \( u(k) \in \mathcal{U} \). \( f(0,0) = 0 \) so that \( x = 0 \) is an equilibrium state.

Our ultimate objective in this paper is to determine a feedback control law which stabilizes the system around the origin and demonstrate how it can be implemented using neural networks. As will be shown, the existence of such a stabilizing controller is guaranteed if the system is locally controllable around the origin.

We shall consider both the case when the function \( f \) is known and when it is not known. In both cases a neural network is used to approximate \( f \) and is in turn used to determine the control input.

Some prior information concerning the System (2) is required to determine the solutions to the above problems. These include the order \( n \) (or an upper bound on \( n \)) of the system, the nature of the function \( f \), and the region \( S \subseteq \mathcal{X} \) where controllability and stability of the system are of interest.

We shall assume that \( f : \mathcal{R}^{n+1} \rightarrow \mathcal{R}^n \) is continuously differentiable and Lipschitz. It is assumed to be bounded for all \( x, u \), though it may be unstable around the equilibrium point (alternately, if it is possible to reset the initial condition of the system whenever it goes beyond a prescribed value, the boundedness assumption can be dropped). The controllability of the system in \( S \) is checked directly when \( f \) is known and deduced when \( f \) is approximated using a neural network. The
relevance of these assumptions and precisely where they are needed will become clear in the following sections.

Note: For clarity of exposition we will assume $u$ to be a scalar. Extensions of results to the multiple input case are quite straightforward.

A. Estimation of $f$

The architectures described herein assume that the adaptation of the neural networks is carried out using error back propagation. Thus an essential first step in the control problem treated in Section IV is to set up a model of the system, which in turn will be used to determine the controller.

If the function $f$ in (3) is known, it follows from [1, 3] that a neural network $NN_f$ which approximates $f$ over a region of interest, can be designed off-line. When $f$ in (2) is assumed to be unknown, such a network has to be trained on-line using the observed input–output pairs $((x(k), u(k)) \rightarrow (x(k+1)))$. Denoting the network by $NN_f[x(k), u(k), \theta]$, where $\theta$ represents the weight vector of the neural network (described in Section II-D), the problem of estimation of $f$ is reduced to a parameter estimation problem. The estimation error $e(k+1)$ is defined by

$$e(k+1) \equiv x(k+1) - \hat{x}(k+1)$$

$$= f[x(k), u(k)] - NN_f[x(k), u(k), \theta]$$

and the objective is to determine a parameter vector $\theta^*$ such that

$$\|f[x, u] - NN_f[x, u, \theta^*]\| = \|e\| < \epsilon \quad \forall (x, u) \in D$$

where $\epsilon$ is a suitably chosen constant and $D$ is the region of operation.

As described in Section II-D, to converge to such a $\theta^*$ the parameter $\theta$ is adjusted using static back propagation, i.e., the weights are adjusted along the negative gradient using the rule

$$\theta(k+1) = \theta(k) + \eta e(k+1) \frac{\partial f(k+1)}{\partial \theta} \bigg|_{\theta=\theta(k)}$$

where $\eta > 0$ is the step size. The architecture used to estimate $f$ is shown in Fig. 1. After training is completed, the plant is assumed to be sufficiently accurately described by the model

$$\hat{x}(k+1) = NN_f[\hat{x}(k), u(k), \theta^*] = NN_f[\hat{x}(k), u(k)]$$

with $NN_f[0, 0] = 0$.

1) Practical Considerations: The determination of an approximate map using a neural network raises both theoretical and practical questions. There may be a tradeoff between the size of the network chosen, which directly influences learning time, and the accuracy that can be achieved. From a practical point of view it is of considerable interest to know how large an error can be tolerated and how it will affect the control of the system. This issue will be addressed in Section IV.

If the initial state can be chosen at the discretion of the designer, the task is relatively easy. By choosing initial values with uniform distribution in $X$, $f(\cdot)$ can be approximated to any degree of accuracy off-line. In many practical problems this is not the case and information concerning $f$ can be obtained only by the choice of suitable inputs $u(k)$ and observation of the corresponding state $x(k+1)$. This implies that the state space can be explored only through the effect of the input $u(k)$ on the system (which is the independent variable) and thus the distributions of the input samples to the network (a concatenation of $x(k)$ and $u(k)$), may be quite nonuniform, making the training task much more difficult, if not impossible. This issue of uniformity of samples will not be dealt with in this paper, but the reader is referred to [14] where a training procedure using multiple neural networks was proposed to solve the problem.

IV. CONTROL

In this section, we discuss the stabilization of the nonlinear system (2) around an equilibrium state. After presenting the simplest method, i.e., stabilization using a linear controller, we first attempt, in Section IV-B, feedback linearization by determining suitable transformations based on the state of the system and the reference input so that the overall system is both linear and stable. This has been studied extensively in the nonlinear control literature. The desired transformations are achieved in the present approach with neural networks which are trained using input–output data. Simulation results included in Section IV-B-1) provide the details of the procedure for implementing the above results.

Since the class of systems that are feedback linearizable is small, the problem of directly stabilizing the given nonlinear system is described in Section IV-C. The simulation results presented in Section IV-C-3) clearly reveal that the domain of attraction achieved by a nonlinear feedback controller can be significantly greater than that realized by a linear controller.

A. Stabilization using Linear Controllers

The simplest scheme for stabilization is by the use of a linear controller. Let $\Sigma_L$ be a linearization of (2) around an equilibrium point $x = 0$. If $\Sigma_L$ is controllable, then linear theory tells us that there exists a linear feedback law $u = K^T x$ that stabilizes $\Sigma_L$ around the origin. Since $\Sigma_L$ is the first order approximation of the original nonlinear system, one might expect that the same linear feedback law will locally make the origin an asymptotically stable point of (2), as shown in Theorem 6 [21].

**Theorem 6:** Let $\Sigma$ be a time-invariant discrete time system as described in (2), and let $x = 0$, $u = 0$ be an equilibrium pair. If the linearization $\Sigma$ of $\Sigma$ around $(0, 0)$ is controllable, then
\[ \sum \text{ is locally controllable and there exists a linear feedback law } u = K^T x \text{ s.t. the closed loop system} \]

\[ x(k+1) = f(x(k), K^T x(k)) \]

is locally asymptotically stable.

This theorem was extensively used in designing controllers to stabilize nonlinear systems around unstable equilibria (e.g., the classical problem of balancing an inverted pendulum).

1) Implementation: The design of a linear controller does not require a neural network. Given the model of the plant, the \( A, b \) matrices of the linearized plant are just the Jacobians of \( NN_{f}(\cdot) \) with respect to the states and the inputs.

Once these are calculated, with a linear feedback law \( u = K^T x \), the linearized system is given by

\[ \delta x(k+1) = A \delta x(k) + b K^T \delta x(k) = (A + b K^T) \delta x(k) \]

and it will be asymptotically stable if the eigenvalues of \( A + b K^T \) lie inside the unit circle of the complex plane [7]. With a feedback law chosen in this manner, the above theorem assures that the nonlinear system will also be locally asymptotically stable.

Though the linear controller will stabilize the nonlinear system around the origin, the range over which the system will be stable depends upon the system and maybe small for nonlinear systems (see simulations 1 and 2). Thus one hopes that by employing an appropriate nonlinear controller, the range over which the system is stabilized can be increased. The following sections address the issue of nonlinear controllers, and the linear controller is used as a benchmark for the evaluation of the performance of more sophisticated controllers.

B. Stabilization Through Linearization

Given the nonlinear system (2), the question arises whether the system is locally equivalent to a linear system via the following two transformations:

1) a change of coordinates in the state space \( z = \Phi(x) \) with \( \Phi(\cdot) \) invertible and continuously differentiable;

2) a feedback law \( u(k) = \Psi(x(k), v(k)) \)

If the latter can be achieved, the tools of linear theory can be used to control and stabilize the system around any desired equilibrium point.

Applying the above transformations, we have

\[ z(k+1) = \Phi(x(k+1)) = \Phi(f(\Phi^{-1}(z(k)), \Psi(\Phi^{-1}(z(k)), v(k)))) \]

(9)

with \( z(k) \) its states and \( v(k) \) its new inputs. If such transformations that make (9) linear exist, the system is feedback linearizable. If the transformations exist only in a neighborhood of \((0,0)\), the system is locally feedback linearizable at \((0,0)\).

Necessary and sufficient conditions for a system to be locally feedback linearizable are given in [5], [6]. Reference [9] also gives necessary and sufficient conditions for systems to be linearizable only through a coordinate transformation. Since the latter form a very restrictive class they will not be treated here.

In the following, we will describe the version given in [5]. For that we need the notion of a distribution.

**Definition 10:** Let \( \mathcal{V} \subset \mathbb{R}^n \) be a set on which \( d \) smooth functions \( s_1, s_2, \ldots, s_d : \mathcal{V} \to \mathbb{R}^d \) are defined. At any given point \( x \in \mathcal{V} \), the vectors \( s_1(x), s_2(x), \ldots, s_d(x) \) span a vector space (a subspace of \( \mathbb{R}^n \)). Let this vector space, which depends on \( x \) be denoted by \( \Delta(x) \), i.e.,

\[ \Delta(x) = \text{span}[s_1(x), s_2(x), \ldots, s_d(x)] \]

and by this we have assigned to each \( x \) a vector space. This assignment is called a distribution.

Returning to the original system (2), let

\[ f_x(x, u) = \frac{\partial}{\partial x} f(x, u), f_u(x, u) = \frac{\partial}{\partial u} f(x, u). \]

Define the following distributions on \( \mathbb{R}^n \) depending on \( u \):

\[ \Delta_0(x, u) = 0 \]

\[ \Delta_1(x, u) = f_x^{-1}(x, u) \text{ Im } f_u(x, u) \]

\[ \Delta_{i+1}(x, u) = f_x^{-1}(x, u)[\Delta_i(f_x(x, u), u) + \text{ Im } f_u(x, u)] \]

where \( \text{ Im } f_u(x, u) \) denotes the image of \( f_u \) and \( f_x^{-1} \text{ Im } \) denotes the counter image of the subspace \( \mathcal{V} \) under the linear map \( f_x \). \( \Delta_i(\cdot, u) \) are distributions depending on \( u \) and it can be shown that

\[ \Delta_0(x, u) \subset \Delta_1(x, u) \subset \Delta_2(x, u) \cdots \]

where \( \Delta_i \) attains maximum rank after at most \( n \) steps. Finally, denoting the Jacobian of \( f \) with respect to \( x \) and \( u \) by \( df = (f_x, f_u) \) we have the following theorem [5].

**Theorem 7:** Let \((x = 0, u = 0)\) be an equilibrium point of (1). Assume that \( \text{rank} \text{ Im } df(0,0) = n \). Then system (1) is locally feedback linearizable at \((0,0)\) iff the distributions \( \Delta_0(x, u), \Delta_1(x, u), \Delta_2(x, u) \cdots \) are of constant dimension and independent of \( u \) in a neighborhood of \((0,0,0)\) and \( \dim \Delta_n(0,0) = n \).

For a linear system \( x(k+1) = Ax(k) + bu(k) \), the above distributions are given by \( \Delta_0(x, u) = 0, \Delta_1(x, u) = A^{-1} \text{ Im } b \) and \( \Delta_{i+1} \) is recursively given by \( \Delta_{i+1}(x, u) = A^{-1}(\Delta_i + \text{ Im } b) \).

We see that for linear systems each \( \Delta_i \) describes the subspace that can be controlled to the origin in \( i \) steps. Clearly for linear systems these subspaces are of constant dimension and independent of \( u \). Thus qualitatively, the theorem may be interpreted to imply that these properties are invariant under feedback and bijective coordinate transformations and only those systems that initially posses them can be made linear.

**Example 2:** Given the second order system

\[ x_1(k+1) = x_2(k) \]

\[ x_2(k+1) = [1 + x_1(k)]u(k) \]

For this system we have

\[ f_x(x, u) = \begin{bmatrix} 0 & 1 \\ u & 0 \end{bmatrix}, f_u(x, u) = \begin{bmatrix} 0 \\ 1 + x_1 \end{bmatrix} \]

The rank condition is satisfied at the origin.

\[ \Delta_0(x, u) = 0, \Delta_1(x, u) = \text{span} \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \Delta_2(x, u) = \mathbb{R}^2. \]
Thus the system is locally feedback linearizable. In view of the simplicity of the example, we can see that any input of the form

$$u(k) = \frac{v(k)}{1 + x_1(k)}$$

will locally linearize the system.

**Example 3:** A planar nonlinear system where $u$ influences only one state directly is given by

$$x_1(k+1) = f_1(x_1(k), x_2(k))$$
$$x_2(k+1) = f_2[x_1(k), x_2(k), u(k)].$$

For this system we have

$$f_x(x, u) = \begin{bmatrix} f_{11}(x) & f_{12}(x) \\ f_{21}(x, u) & f_{22}(x, u) \end{bmatrix},$$
$$f_u(x, u) = \begin{bmatrix} 0 \\ f_{2u}(x, u) \end{bmatrix}$$

with $f_{ij} \equiv (\partial f_i)/\partial x_j$ and $x = (x_1, x_2)$.

The rank condition is satisfied at the origin if $f_{2u}(x, u) \neq 0$ and $f_{12}(x) \neq 0$. The distributions are given by

$$\Delta_0 = 0, \Delta_1 = \text{span} \left[ -f_{12}(x) \right].$$

$\Delta_1$ depends only on $x$ and $\Delta_2$ together with $f_u$ spans the whole space.

$$\Rightarrow \Delta_2 = \mathbb{R}^2.$$

Thus the system is locally feedback linearizable.

1) **Implementation:** If the equations of the plant are available and they satisfy the conditions for feedback linearization, a solution is known to exist and the task would be to find two mappings $\Phi: \mathbb{R}^n \rightarrow \mathbb{R}^n$ and $\Psi: \mathbb{R}^{n+1} \rightarrow \mathbb{R}$ such that

$$z = \Phi(x), u = \Psi(x, v)$$

and

$$z(k+1) = \Phi[x(k+1)] = \Phi[f(x(k), \Psi(x(k), v(k)))] = Az(k) + bu(k)$$

where $A$, $b$ is a controllable pair. In particular we can choose $A$ and $b$ to be in the controllable canonical form - or

$$A = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ \vdots \\ \vdots \\ 1 \end{bmatrix}. \quad (10)$$

If, on the other hand, we have only a model of the real plant given by (8), the question arises whether feedback linearizability of (2) implies that (8) is also feedback linearizable.

Following Section IV, we assume that the model error over the region of operations $D$ is bounded by $\epsilon < 1$ i.e.,

$$\|NN_f(x, u) - f(x, u)\| < \epsilon \quad \text{for all } x, u \in D.$$

Applying $\Phi$ and $\Psi$ on $NN_f$ we obtain

$$\Phi[NN_f(x(k), \Psi(x(k), v(k)))] = \Phi[f(x(k), \Psi(x(k), v(k)))] + \epsilon(x(k), \Psi(x(k), v(k))).$$

(11)

Since $\Phi(\cdot)$ is a smooth function, if $\|\epsilon(\cdot, \cdot)\| < \epsilon$ and $\epsilon$ is assumed to be small, (11) can be written as

$$\Phi[f(x(k), \Psi(x(k), v(k)))] + \epsilon_{11}[x, u, \Psi(\cdot), \Phi(\cdot)] = Az + bu + \epsilon_{11}[x, u, \Psi(\cdot), \Phi(\cdot)]. \quad (12)$$

The bound on $\epsilon_{11}$ is a function of $\epsilon$ and sup $\|\Phi/\partial x\|$. Hence, if the model (8) is sufficiently accurate, it can be transformed into a system of form (12) which approximates a linear system.

Now, the objective is to simultaneously train two networks $NN_f$ and $NN_\theta$ such that, when the input to (8) is given by $u = NN_\theta(x, u)$, $\tilde{z} = NN_f(x)$ will track the output $z(k)$ of the linear model (Fig. 2) given by the equation

$$z(k+1) = Az(k) + bu(k) \quad (13)$$

where $A$, $b$ are given in (10).

Without loss of generality we can assume $\theta(0) = 0$ (mapping the origin of $x$ to the origin of $z$). Thus if both systems are initiated at the origin, the instantaneous error is given by $e(k) = z(k) - \tilde{z}(k)$ and the performance over an interval can be characterized by $I$ where

$$I = \sum_k ||z(k) - \tilde{z}(k)||^2 \equiv \sum_k ||e(k)||^2.$$

Since $NN_f$ is directly connected to the output, its weights can be adjusted using static backpropagation. The model however, contains a feedback loop. Thus to calculate the gradients of the performance criterion with respect to the weights of $NN_\theta$, dynamic backpropagation needs to be used.

Assuming that $\theta \in \Theta(NN_f)$ (where $\Theta$ is the set of parameters of $NN_\theta$), the gradient of $I$ with respect to $\theta$ is derived as follows:

$$\frac{dI}{d\theta} = -2 \sum_k [z(k) - \tilde{z}(k)]^T \frac{d\tilde{z}(k)}{d\theta} \quad (14)$$

$$\frac{d\tilde{z}(k)}{d\theta} = \sum_{j} \frac{\partial \tilde{z}(k)}{\partial x_j(k)} \frac{dx_j(k)}{d\theta} \quad (15)$$

Fig. 2. Architecture for feedback linearization.
\[
\frac{dx_j(k)}{db} = \sum \frac{\partial x_j(k)}{\partial x_i(k+1)} \frac{dx_i(k+1)}{db} + \frac{\partial x_j(k)}{\partial \theta}.
\] (16)

Thus the gradient of the output with respect to \( \theta \) is given by the output of the linear system:

\[
\frac{dx(k+1)}{db} = A \frac{dx(k)}{db} + b \frac{\partial x(k)}{\partial \theta}\\
\frac{d\tilde{x}(k)}{db} = c^T \frac{dx(k)}{db}
\] (17)

where \((dx(k))/db) is the state vector, \((\partial x(k))/\partial \theta) is the input and \(A, b, c\) are defined by \(a_{ij} = (\partial x_i(k+1))/\partial x_j(k))\), \(b_i = 1\) and \(c_i = (\partial \tilde{x}(k))/\partial x_i(k))\). Initial conditions for the states are set to zero.

Once \(NN_\Phi\) and \(NN_\Psi\) are trained, the behavior of \(\tilde{x}(k)\) is given by

\[
\tilde{x}(k+1) = NN_\Phi[NN_\Psi[x(k), NN\Phi(x(k), v(k))]\\
= A\tilde{x}(k) + b v(k) + e_{12}(x(k), v(k))
\] (18)

where \(e_{12}\) is a small error, representing the deviation of the transformed system from a perfect linear model.

 Earlier in this section, it was shown that feedback linearization of the system (2) will guarantee approximate feedback linearization of the model (8). The same is true in the reverse direction and thus, from (18) we have

\[
\tilde{x}(k+1) = NN_\Phi[NN_\Psi[x(k), NN\Phi(x(k), v(k)))]\\
= A\tilde{x}(k) + b v(k) + e_1(x(k), v(k))
\] (19)

where \(e_1 = e_{12} + e_1\). The first term results from inaccuracies in the identification and the accuracy of the imperfect linearization of (8).

With zero input, the linear system (13) is asymptotically stable. Thus using Theorem 5, it is strongly stable under perturbation, i.e., for every \(\epsilon_0\) there exists \(\epsilon_0(\epsilon_0)\) and \(r(\epsilon_0)\) s.t. if

\[
||e_1(x, 0)|| < \epsilon_1 \quad \text{for all } ||x|| < r
\]

then

\[
\tilde{x}(k+1) = A\tilde{x}(k) + e_1(x(k), 0)
\] (20)

will converge to the \(\epsilon_0\) ball \((B_{\epsilon_0})\) around the origin.

Further, to understand the effect of the perturbation \(e_1\) on (20) let \(\tilde{z}(k, z_i)\) denote the solution of (20) with \(\tilde{z}(0, z_i) = z_i\). Similarly let \(z(k, z_i)\) denote the solution of the linear system \(z(k+1) = Az(k)\) and let \(e_1^j(z_i) \equiv \tilde{z}(n, z_i) - z(n, z_i)\).

**Proposition 1:** If there exists a set \(S\) s.t. \(||e_1^j(z)|| < \epsilon_0^j\) for all \(z \in S\), then for any initial condition inside \(S\) the system (20) will converge to the \(\epsilon_0^j\) ball around the origin in at most \(n\) steps.

**Proof:** The above result follows readily from the fact that \(A^ny = 0\) for all \(k \geq n\).

Finally, since \(NN_\Phi\) was trained to map the origin of \(z\) to the origin of \(x\),

\[
z(k+1) = NN_\Phi[x(k), NN\Phi(x(k), 0)]
\]

will also converge to the \(\epsilon'\) ball around the origin, where \(\epsilon'\) is determined by \(NN_\Phi^{-1}(B_{\epsilon_1})\).

**Simulation 1 (Feedback Linearization):** We consider here the feedback linearization of a planar system described by the equations

\[
x_1(k+1) = f_1[x_1(k), x_2(k)] = x_2(k)x_1(k) - 1\\
x_2(k+1) = f_2[x_1(k), x_2(k), u(k)]\\
= \tanh[2x_1(k) + (1 + x_2(k))u(k)]
\]

We first consider the case where \(f_1\) and \(f_2\) are known. Following that, we discuss the adaptive control problem where the two functions are unknown. In the first case, as shown in Example 3, since only \(f_2\) depends upon \(u(k)\), and \(f_{2u}(0, 0) = 1\) and \(f_{12}(0, 0) = -1 \neq 0\), the system is feedback linearizable. In the second case where \(f_1\) and \(f_2\) are unknown, the order of the system is assumed to be known and using the method described in Section IV, a dynamical model

\[
\tilde{x}(k+1) = NN_{\tilde{x}}[\tilde{x}(k), u(k)]
\]

which approximates the input–output behavior of the plant is determined. Here a network \(NN_{\tilde{x}} \subset N_{3, 10, 5, 2}\) was trained to emulate the system. In this example, after the system is identified, it is seen that the first component of \(NN_{\tilde{x}}\) is almost independent of \(u\) (input weights are very small). Thus following Section VI-B, since the model closely approximates a feedback linearizabile system, we can assume that the system can be transformed to a system which approximates a linear system in the region of interest (i.e., is “approximately feedback linearizable”).

**Note:** In general, for more complicated systems, conditions for feedback linearization as required by Theorem 7 may be very hard to verify. In such cases one may have no other recourse in practical situations, except to assume that they are valid and apply the linearization procedure.

At this stage (whether the first or second assumption is made) the objective is to determine two networks \(NN_\Phi\) and \(NN_\Psi\) to approximate the desired transformations. For the problem under consideration \(NN_\Phi \subset N_{3, 10, 5, 2}\) and \(NN_\Psi \subset N_{3, 10, 5, 1}\) were used and trained using input–output data. The linear time invariant system, which the overall system is required to track, has the form

\[
z(k+1) = Az(k) + bv(k)
\]

where

\[
A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}
\]

and \(z^T(k) \equiv [z_1(k), z_2(k)]\).

Since we require \(\Phi(0) = 0\), training was initiated at the origin. A random input uniformly distributed in \([-1, 1]\) was used to drive the system. The error summed over \(k\) steps was used as a performance index and adjustment of the weights was performed at the end of each sequence to decrease the error. After each adjustment the system was reset to the origin. The length of the training sequences were gradually increased, starting with \(k = 10\). After successful learning was achieved on consecutive runs, the length of the sequence was increased by ten till \(k = 100\) was reached. To improve convergence,
variable step size given by $\eta = 0.01\times$ (recursive average error) was used to adjust the weights.

To test the efficiency of the proposed method, the response of the linearized system was compared with the response of the corresponding linear system for different inputs. For the problem under consideration, the latter is merely a delay of two units of time. In Fig. 3, the response of the linearized system for a square wave input of amplitude 0.5 and period of 20 time units is shown. The error between the two responses, shown in the same figure, reveals the accuracy of the linearization procedure.

If the objective is to stabilize the system, the proposed method can be compared with standard linearization procedures. In Fig. 4, the response of the nonlinear system with $x(0) = [2, 2]^T$ is shown, when $v(k) = 0$. Since the output tends to zero in two steps, it can be concluded that the feedback linearization is valid for this initial condition.

The linearized equations around the origin have the form

$$\delta x(k + 1) = \begin{bmatrix} 0 & -1 \\ 2 & 0 \end{bmatrix} \delta x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \delta u(k).$$

It is clear that the linearized system can be made nilpotent by choosing $u(k) = -2x_1(k)$. The response of the nonlinear system with this linear controller, shown also in Fig. 4, clearly reveals that the linearization is not valid for the initial condition $x(0) = [2, 2]^T$.

The results presented were achieved after 60,000 steps (with time between consecutive weight adjustments varying between ten and a hundred time steps).

C. Direct Stabilization

Though elegant, feedback linearization can be applied only for a limited class of systems. Further, even when it is possible, the actual implementation described in Section IV-B-1) involves dynamic back propagation, which is computationally not efficient. Hence, in this section, nonlinear controllers that directly stabilize the nonlinear system will be derived and in turn implemented using neural networks.

1) Open Loop Control: As a first step, we shall consider the use of nonlinear controllers in an open loop mode. A basic result in linear control theory states that an n-th order linear time invariant system defined by

$$x(k + 1) = Ax(k) + bu(k)$$

where $x(k) \in \mathcal{R}^n$, $u(k) \in \mathcal{R}^r$, $A$ and $b$ are respectively $n \times n$ and $n \times r$ matrices, will be controllable iff the matrix

$$M_c = [b, Ab, \ldots, A^{n-1}b]$$

is of rank n. If $b$ is an $(n \times 1)$ vector, this implies that $M_c$ is nonsingular. $M_c$ is called the controllability matrix.

For linear systems controllability is a global property. If the system is controllable, any point can be reached within n time steps, and if the rank condition is not satisfied only the subspace spanned by the columns of $M_c$ can be reached. For the nonlinear system (2), the conditions for global controllability are very hard to establish and one has to resort to local concepts. The local controllability of the nonlinear system (2) around an equilibrium state can be derived by examining its linearization around the equilibrium point.

Linearizing (2) around the equilibrium one gets the linear system:

$$\sum_{L} \delta x(k + 1) = f_x(0, 0) \delta x(k) + f_u(0, 0) \delta u(k) = A \delta x(k) + b \delta u(k)$$

where $A = f_x(0, 0)$ $b = f_u(0, 0)$.

The state at time $k + n$ is given by:

$$x(k + n) = f[x(k + n - 1), u(k + n - 1)]$$

$$= f[f[x(k + n - 2), u(k + n - 2)], u(k + n - 1)]$$

$$\vdots$$

$$= f[\cdots f(x(k), u(k)), \cdots, u(k + n - 1)]$$

$$= F[x(k), u(k), \cdots, u(k + n - 1)].$$

(22)
Taking the derivative of $F$ with respect to $u(i) = k, \cdots, k + n - 1$ at $(0,0)$ will give

$$
\frac{\partial x(k+n)}{\partial u(k+n-1)} = b, \quad \frac{\partial x(k+n)}{\partial u(k+n-2)} = Ab, \cdots, \frac{\partial x(k+n)}{\partial u(k)} = A^{n-1}b.
$$

(23)

Defining $U_u(k) = [u(k), u(k+1), \cdots, u(k+n-1)]$ we get

$$
D_{U_u}(k)x(k+n) \equiv \frac{\partial x(k+n)}{\partial U_u(k)} = [A^{n-1}b | A^{n-2}b | \cdots | b]
$$

which is just $M_e$ with the columns rearranged. This, in turn, leads to the following theorem which has been discussed in the control literature:

**Theorem 8:** Let $\Sigma$ be the nonlinear dynamical system (2) and its linearization around the equilibrium $x = 0$ as given in (21). If the linearized system is controllable, then $\Sigma$ is locally controllable around the origin.

**Proof:** The proof follows from a direct application of the inverse mapping theorem. Consider the mapping $G(x(k), U_u(k)) = (x(k), x(k+n))$ where $G : \mathbb{R}^{2n} \to \mathbb{R}^{2n}$.

The Jacobian matrix of this mapping at $(0,0)$ is given by

$$
D_{(0,0)} G = \begin{bmatrix}
I & 0 \\
D_{x(k)}x(k+n) & D_{U_u(k)}x(k+n)
\end{bmatrix}
$$

where $I$ is an $n$-dimensional identity matrix. Using the inverse function theorem, if $\text{rank}[D_{U_u}(k)x(k+n)] = n$, then locally there exists an inverse $\psi = G^{-1}$ such that $(x(k), x(k+n)) = \psi(x(k), x(k+n))$. Alternately, given any two points $x_i$ and $x_f$ an input sequence $U_u(k) = \psi_u(x_i, x_f)$ will drive the system from $x_i$ to $x_f$ in $n$ steps. For every neighborhood $V \subset X$, by the continuity of $f$ and $\psi$, if $x_i$ and $x_f$ are chosen sufficiently close to zero, the system will be transferred from $x_i$ to $x_f$ without leaving $V$.

**Example 4 (An Ideal Juggler):** In this example we consider the task of juggling a ball using a flat board. We make the simplifying assumption that the ball moves in a two-dimensional plane. We also assume that there are no losses in the system, that the ball follows a perfect ballistic curve while in the air, and that the collision with the board is elastic and always at the same height (Fig. 5). The states of the system are given by the angle of the ball and its horizontal location just before impact. The direction of the ball is controlled by setting the angle of the board, which can consequently be considered as the control input.

The equations of the system are:

$$
\begin{align*}
\rho(k+1) &= \rho(k) + \frac{2E}{mg} \sin[4\alpha(k) - 2\varphi(k)] \\
\varphi(k+1) &= \varphi(k) - 2\alpha(k)
\end{align*}
$$

(24)

$k$ \hspace{0.5cm} count of impacts;  
$\rho$ \hspace{0.5cm} horizontal location of impact;  
$\varphi$ \hspace{0.5cm} angle at time of impact (with respect to vertical);  
$\alpha$ \hspace{0.5cm} angle of board (with respect to horizontal), the control input;  
$E, m$ energy and mass of the ball.

The origin $(\rho = 0, \varphi = 0)$ — (ball bounces vertically at center of board) is an equilibrium state. The linearized equation at the origin are

$$
\begin{bmatrix}
\delta\rho(k+1) \\
\delta\varphi(k+1)
\end{bmatrix} = 
\begin{bmatrix}
-\frac{4E}{mg} & 0 \\
0 & -2
\end{bmatrix}
\begin{bmatrix}
\delta\rho(k) \\
\delta\varphi(k)
\end{bmatrix} + 
\begin{bmatrix}
\frac{8E}{mg} \\
\frac{16E}{mg}
\end{bmatrix} \delta\alpha(k).
$$

The controllability matrix $M_e$ is given by

$$
M_e = \begin{bmatrix}
[mg] & [16E] \\
[-2] & [-2]
\end{bmatrix}.
$$

Since $M_e$ is full of rank, the system can be locally controlled from any initial point $(\rho_i, \varphi_i)$ to any other point in at most two steps. In particular, if the desired state of the system is the origin, then the control

$$
\begin{align*}
\alpha_1(\rho_i, \varphi_i, 0, 0) &= \frac{1}{4} \left[ 2\varphi_i - \sin^{-1} \left( \frac{mg}{2E} \rho_i \right) \right] \\
\alpha_2(\rho_i, \varphi_i, 0, 0) &= \frac{1}{4} \sin^{-1} \left( \frac{mg}{2E} \rho_i \right)
\end{align*}
$$

will achieve the transfer from any initial state to the origin (note that $\sin^{-1}$ is locally invertible around the origin).

For this scheme to be useful in real world problems, the controller needs to be robust with respect to noise or outside disturbance. Since open loop control assumes full knowledge of the system and does not take into account such possibilities, it is of little practical use and our goal is to establish the existence of a feedback controller.

2) **Nonlinear Feedback Stabilization:** Though not practical in itself, the method used to derive the open loop controller will provide the basis for a feedback controller.

**Theorem 9:** Let $\Sigma$ be the nonlinear dynamical system (2) and its linearization around the origin (as given in (21)). If the linearized system is controllable then there exists a neighborhood $\mathcal{V}_e \subset \mathcal{X}$ around the origin and a continuous feedback law $u(k) = g[x(k)]$ that will make $\mathcal{V}$ $n$-step stable with respect to the origin. (i.e., any point $x_0 \in \mathcal{V}$ can be brought to the origin in at most $n$ steps).

**Proof:** If the linearized system around the origin is controllable, then, from Theorem 8, there exist neighborhoods $\mathcal{W}_e \subset \mathcal{X}$ and $\mathcal{W}_u \subset \mathcal{U}$ such that $x(0, u) \in \mathcal{W}_u \times \mathcal{W}_e$ and for any $x_0, x_f \in \mathcal{W}_e$ there is a unique sequence of inputs $u(k) = g(x_0, x_f) \in \mathcal{W}_u, k = 0, \cdots, n - 1$ that will transfer the system from $x_0$ to $x_f$ in $n$ steps.

If we choose $x_f$ to be zero we get $u(k) = g_k(x_0)$ (a restatement of the Implicit Function Theorem). Define

![Fig. 5. The ideal juggler.](image-url)
\[ x_1 = f[x_0, g_0(x_0)] \] and recursively \( x_{k+1} = f[x_k, g_k(x_k)] \).

By construction \( x_0 = 0 \) and since \( g_k(\cdot) \) are continuous, there exists a neighborhood \( \mathcal{V}_0 \subset \mathcal{W}_2 \) s.t. \( x_0 \in \mathcal{V}_0 \), then for all \( k < n, x_k \in \mathcal{W}_2 \).

Now, assume the system was started at \( x(0) = x_1 \). Since \( x_1 \in \mathcal{W}_2 \), the sequence of inputs \( u(k) = g_k(x_1) \) will drive it to the origin in \( n \) steps. On the other hand, the original input sequence \( u(k) = g_{k+1}(x_0) \) will drive it to the origin in \( n - 1 \) steps. The origin, however, is an equilibrium state (with zero input the system will remain at the origin). Thus the input sequence \( (g_1(x_0), g_2(x_0), \ldots, g_{n-1}(x_0), 0) \) will also drive \( x_1 \) to the origin in \( n \) steps. But for any \( x \in \mathcal{W}_2 \), the input sequence that drives it to the origin in \( n \) steps is unique and thus we get that \( g_0(x_1) \) must be equal to \( g_1(x_0) \).

The same reasoning, applied to each of the \( x_i \), will lead to \( g_0(x_i) = g_i(x_0) \). Hence, for any \( x \in \mathcal{V}_0 \) the system \( x(k + 1) = f[x, g_0(x)] \) will converge to the origin in at most \( n \) steps. \( \square \)

The equivalent result for a linear systems \( x(k + 1) = Ax(k) + Bu(k) \) is that state feedback using \( u(k) = k^T x \) makes the combined matrix \( \tilde{A} = A + bk^T \) singular. For a two-dimensional canonical system

\[
\begin{bmatrix} 0 & 1 \\ a_1 & a_2 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k)
\]

choosing \( k^T = [-a_1, -a_2] \) the state feedback becomes \( u(k) = -a_1 x_1(k) - a_2 x_2(k) \) and that will bring the system to the origin in at most two steps. A controller that stabilizes a system around a point in finite time is called a **dead beat controller**.

**Example 5 (The juggling revisited):** Again the juggling’s equations are given by

\[
\begin{align*}
\rho(k+1) &= \rho(k) + \frac{2E}{mg} \sin[4\alpha(k) - 2\varphi(k)] \\
\varphi(k+1) &= \varphi(k) - 2\alpha(k)
\end{align*}
\]

and in Example 4 an open loop control that transfers the system to the origin in two steps was given.

From Theorem 9 it follows that the system can be stabilized around the origin using a feedback control. In fact, this will be accomplished by the control law

\[
\alpha(k) = g(\rho(k), \varphi(k)) = \frac{1}{4} \left[ 2\varphi(k) - \sin^{-1} \left( \frac{mg}{2E} \rho(k) \right) \right].
\]

**Example 6:** Let a second order system be

\[
\begin{align*}
x_1(k+1) &= x_2(k) + u(k) \\
x_2(k+1) &= -x_1 - x_2(u(k)).
\end{align*}
\]

The rank condition is satisfied at the origin and hence the system is locally stabilizable. On the other hand the system does not satisfy the conditions for feedback linearization and thus, cannot be transformed into a linear system.

To extend the above result, let us define \( \bar{f}_g(\cdot) \equiv f[\cdot, g(\cdot)] \) and \( \bar{F}_g(\cdot) \equiv F_g(\cdot) \). From Theorem 9 we know that there exists an open set \( \mathcal{V} \subset \mathcal{X} \) containing the origin such that for all \( x \in \mathcal{V} \), \( \bar{F}_g(x) = 0 \). Since \( \bar{F}_g(\cdot) \) is a continuous function, it follows that there exists a larger open set \( \mathcal{W} \supset \mathcal{V} \) such that for all \( x \in \mathcal{W} \),

\[
\|\bar{F}_g(x) - \bar{F}_g(0)\| = \|\bar{F}_g(x) - 0\| < \|x\|.
\]

Hence, using the contraction mapping theorem, for any \( x \in \mathcal{W} \), \( \lim_{k \to \infty} \bar{F}_g(x) = 0 \), \( \lim_{k \to \infty} \bar{F}_g(x) = 0 \), and further, for any given \( x, k \) finite is it this fact that is exploited in the design of a nonlinear controller for the system (2) to make it stable in a finite number of steps. In particular, the design goal is to determine a feedback law \( g(\cdot) \) which will make \( \mathcal{W} \) as large as possible. For any given \( g \), one may define the autonomous dynamical system

\[
\ddot{x}(k+1) = \bar{F}_g(x(k)).
\]

From the above discussion and the Liapunov function of the system for all \( \dot{x} \in \mathcal{W} \). Thus to make \( \mathcal{W} \) as large as possible, \( g \) needs to be designed so that \( \|\dot{x}(k+1)\| < \|\dot{x}(k)\| \). This, as will be shown, is achieved naturally using neural networks. In the method described thus far, the region \( \mathcal{W} \) depends upon the system and is at least as large as that obtained using a linear controller. Example 7 as well as Simulation 2 help to clarify this issue.

**Example 7:** Consider the first order system

\[
x(k+1) = x(k) + u(k) + au^2(k).
\]

The linearized system around the equilibrium \( (x = 0, u = 0) \) is given by

\[
\delta x(k+1) = \delta x(k) + \delta u(k)
\]

and a linear control law that will stabilize the system is given by

\[
u(k) = -x(k).
\]

Applying this linear feedback law to the nonlinear system will result in the autonomous system

\[
x(k+1) = x(k) - ax^2(k) = -ax^2
\]

which is stable in the interval \((1/4, 1/4a)) \). Since the linearized system is controllable, Theorem 9 guarantees the existence of a local nonlinear controller \( u(k) = \hat{g}(x(k)) \) that will bring the system to the origin in one step, i.e., \( x + g_1(x) + ag_2^2(x) = 0 \). Solving for \( g_1 \), we get

\[
\hat{g}(x) = \frac{-1 + \sqrt{1 - 4ax}}{2a}.
\]

This control law is defined on the interval \((-\infty, 1/4a))\) and will converge to the linear controller as \( a \to 0 \).

To extend the range of the nonlinear controller, instead of controlling the system to zero in one step, we choose a contraction coefficient \( \rho \) and design the controller to move the system from \( x \) to \( \rho x \) in one step when \( x > (1/4a) \). To achieve this, we need to determine \( g_2 \) so that \( x + g_2(x) + ag_2^2(x) = \rho x \) and again solving for \( g_2 \) we obtain

\[
g_2(x) = \frac{-1 + \sqrt{1 - 4ax(1-\rho x)}}{2a}.
\]
For a given $\rho$, this control law will stabilize the system on the interval $(-\infty,(1/4\rho(1-\rho))]$, approaching $(-\infty,\infty)$ as $\rho \to 1$. Combining the two control laws we get a global law
\[
g(x(k)) = \begin{cases} 
g_1(x), & \text{if } x \in \left[\frac{1}{4\rho}, \frac{1}{2\rho}\right] 
g_2(x), & \text{otherwise.} \end{cases}
\]

Finally, the existence of a local deadbeat controller that stabilizes the system around the origin, enables us the following result:

**Theorem 10:** Let $Z$ denote the set controllable to the origin.

Then there exists a feedback law $g : \mathcal{X} \to \mathcal{U}$ that makes $Z$ finitely stable with respect to the origin.

**Proof:** Let $g_0(\cdot)$ be the deadbeat stabilizing controller and let $V_0$ be the $n$-step stable set associated with it. Let $V_1$ denote the set of all points that can be driven to $V_0$ in one step. For each such $x \in V_1$ let $u_0(x)$ be a control input (which need not be unique) that drives $x$ into $V_0$. Clearly $V_0 \subset V_1$. For $x \in V_1$ a feedback law $g_1 : V_1 \to \mathcal{U}$ will be given by
\[
g_1(x) = \begin{cases} 
g_0(x), & \text{if } x \in V_0 
u(x), & \text{otherwise.} \end{cases}
\]

By this construction, $V_1$ is an $n+1$ stable set with respect to the origin.

Similarly we define $V_{k+1}$ the set of points that can be driven to $V_k$ in one step and the control law on $V_{k+1}$ will be defined by
\[
g_{k+1}(x) = \begin{cases} 
g_k(x), & \text{if } x \in V_k \nu_k(x), & \text{otherwise.} \end{cases}
\]

We claim that
\[
x \in V_k \Rightarrow x \in Z: \text{By definition of } Z \text{ for all } k, V_k \subset Z. 
x \in Z \Rightarrow x \in V_k \text{ for some } k: \text{Assume there exists an } x \in Z 
\text{such that for any } k, x \notin V_k. \text{ Thus by the construction of } V_k, 
x \text{cannot be driven to } V_0 \text{ in } k-1 \text{ steps and thus will not approach zero even asymptotically, which is a contradiction.}

Note that the feedback law constructed this way is also the minimal time controller.

The above feedback law, though global, need not be continuous. Hence it is not clear as to how it can be implemented using continuous neural networks. For piecewise linear systems, a result similar to Theorem 10 was proved by Somtág [22]. In a recent paper [23] he has shown that discontinuous controllers for such piecewise linear systems can be realized by two hidden layer feedforward neural networks consisting of linear threshold units. However, the backpropagation algorithm is based on the differentiability of the neuron’s output and thus cannot be used to train such networks.

In this paper we will limit ourselves to control laws that are continuous and as suggested in this section, will present learning methods that optimize the range over which a contraction stabilizing controller is valid.

3) **Implementation:** For the above derivation to be valid, the rank condition needs to be checked. This is done by determining the Jacobian of $NN_f$ with respect to the inputs at the equilibrium point. Let $\hat{A}$ and $\hat{b}$ be defined as
\[
\hat{A} = \frac{\partial NN_f(x,u)}{\partial x} \bigg|_{0,0}, \quad \hat{b} = \frac{\partial NN_f(x,u)}{\partial u} \bigg|_{0,0}.
\]

Using the matrices $\hat{A}$ and $\hat{b}$, the rank of the model’s controllability matrix $M_c$, is checked.

Let $M_c$ be of full rank. Let $S$ denote the region of interest in which we wish to stabilize the system. Our goal is to train a neural network $NN_g$ as a controller of (2) that will make $S$ finitely stable with respect to the origin. The results developed earlier establish that there exists a control law $u = g(x)$ for which the following is true. (i) There exists an open set $V$ containing the origin such that for all $x \in V$, $F_g(x) = 0$. (ii) There exists a larger open set $W \supset V$ such that for all $x \in W$, $F_g(x)$ is a contraction mapping.

Based on these results, the performance of a controller can be evaluated only in intervals of $n$ steps. We assume that a control law can be determined so that $W$ covers $S$. Though our ultimate goal is to stabilize the actual system, the training of the controller is done using the model, and thus we can assume arbitrary initial conditions. The latter are selected using a random uniform distribution over $S$. Let
\[
NN_{f,g}(x) = NN_f(x,N_g(x)).
\]

Once an initial point $x_0$ is chosen, $x_n = NN_{f,g}(x_0)$ is calculated by running the controlled model $n$ steps.

Since it is only for $x \in V$ (which is unknown) that the system can be brought to zero in $n$ steps, the training error for the controller must be chosen as follows:
\[
e(x) = \begin{cases} 
x_n, & \text{if } ||x|| < \rho \text{ or } ||x|| > \lambda ||x_0|| 
0, & \text{otherwise.} \end{cases}
\]

With $\rho > 0$ initially chosen small and the parameter $\lambda (0 < \lambda < 1)$ determining the contraction over $W$, initially chosen close to 1.

To calculate the gradient of the output error with respect to a typical parameter $\theta \in \Theta(NN_g)$, one needs to add together the contribution to the error due to $\theta$ at each step up to time $n$. This is done by constructing a static mapping that is an $n$ step unfolding of the original system (as shown in Fig. 6). The error is back propagated through the layers. The contribution of a given weight to the error is merely the sum of its contributions at the different layers. Hence, once this is computed, the weight is adjusted along the error gradient.

If the process does not converge within a predetermined number of steps, the value of $\rho$ is decreased. This is equivalent to assuming a smaller $V$. If it does converge, an increase in $\rho$ may shorten the time it takes to control the system, resulting in a more robust stabilizer. Similarly, $\lambda$ is initially chosen close to 1, to maximize $W$. If learning can be achieved with a smaller $\lambda$, the resulting controller will have faster response. From the above discussion it is clear that the ability to converge locally to the equilibrium state within $n$ steps, and thus establishing the existence of a contraction mapping, is crucial in choosing the particular architecture (or number of unfoldings). Once this is known, the learning procedure that
is required involves only static backpropagation and is fairly easy to implement.

Once $NN_p$ is trained to stabilize

$$x(k+1) = NN_F[x(k), NN_p(x(k))]$$

over $S$, one would hope that it can stabilize the actual system. Applying the control law $u(k) = NN_p(x(k))$ to (2) we have

$$x(k+1) = f[x(k), NN_p(x(k))]$$
$$= NN_F[x(k), NN_p(x(k))] + c[x(k)].$$  \hspace{1cm} (30)

Since the model is asymptotically stable, from Theorem 5, it is strongly stable under perturbation, i.e., for every $c_0$ there exists $e(c_0)$ s.t. if $\|e(x)\| < c_0$ for all $x \in S$ then (30) will converge to the $c_0$ ball around the origin. $e(\cdot)$ depends upon the accuracy of the identification model. Thus the more accurate the model, the smaller the control error.

This qualitative statement can be strengthened by considering the special form of the stabilizer. Recall that the controller was trained so that (i) $NN_F(x_0) = 0$ for $x_0 \in B_\rho$, (ii) $\|NN_F(x_0)\| < \lambda \|x_0\|$ for $x_0 \in S$, $\lambda$ being a predetermined contraction coefficient.

With this in mind, the following proposition can be stated.

**Proposition 2:** Let $\lambda$ denote the model’s contraction coefficient. Let $e^n(x)$ denote the $n$-step error between the controlled model and the controlled system, both initialized at $x$, and assume $\|e^n(x)\| < e^0$ for all $x \in S$. If $B(e^n)/(1 - \lambda) \subset S$ then (30) will converge to $B(e^n)/(1 - \lambda)$.

Furthermore, if $e^n/(1 - \lambda) < \rho$ (30) will converge to $B_\rho$.

**Proof:** Assume $x(k) = x_0$ and $\|x_0\| > (e^n)/(1 - \lambda)$.

Then

$$\|x(k + n)\| = \|\hat{F}_{NN}(x_0)\| = \|NN_{F,p}(x_0) + e^n(x_0)\|$$
$$\leq \|NN_{F,p}(x_0)\| + \|e^n(x_0)\| < \lambda \|x_0\| + e^n$$
$$= [1 + (\lambda - 1)] \|x_0\| + e^n$$
$$= \|x_0\| - (1 - \lambda) \|x_0\| + e^n < \|x_0\|.$$  

The sequence is monotonically decreasing and bounded from below ($\|x(k)\| > 0$). Hence, it must converge to some constant $|x| = c$. But if $c > (e^n)/(1 - \lambda)$, as just shown, $\|x(k + n)\| < c$. Thus the system will converge to $B(e^n)/(1 - \lambda)$.

Now, if $(e^n)/(1 - \lambda) < \rho$, system will converge to $B_\rho$. If $x(0) = x_0 \in B_\rho$ then

$$\|x(n)\| = \|\hat{F}_{NN}(x_0)\|$$
$$= \|NN_{F,p}(x_0) + e^n(x_0)\|$$
$$\leq \|NN_{F,p}(x_0)\| + \|e^n(x_0)\|$$
$$< e^n.$$  

**Simulation 2 (Direct Stabilization):** Let us consider a third order system described by the following system of equations:

$$x_1(k+1) = f_1[x_2(k), x_3(k), u(k)]$$
$$= x_2(k) - x_3(k) + u(k)$$
$$x_2(k+1) = f_2[x_1(k), x_2(k), u(k)]$$
$$= 2x_1(k) - [1 + .5x_2(k)]u(k)$$
$$x_3(k+1) = f_3[x_1(k), x_2(k), x_3(k), u(k)]$$
$$= x_1(k)[x_2(k) - x_3(k)] + u(k)$$

which has to be regulated around the origin. As in the previous example we first consider the case when $f_1$, $f_2$, and $f_3$ are known and later the case when they are unknown. In the latter problem, the designer is assumed to have some prior information concerning the plant, as described as follows.

When $f_i (i = 1, 2, 3)$ are known, we conclude the following.

(i) The system is locally controllable around the origin. Local Controllability is easily verified by examining the $A$, $b$ matrices of the linearized system. These are given by

$$A = \begin{bmatrix} 0 & 1 & -1 \\ 2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, b = \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}.$$  

The controllability matrix is

$$M_c = [b|Ab|A^2b] = \begin{bmatrix} 0 & -2 & 0 \\ -1 & 0 & -4 \\ 1 & 0 & 0 \end{bmatrix},$$

and hence the system is controllable. Further $A$ is unstable (the eigenvalues being $(\sqrt{2}, -\sqrt{2}, 0)$).

(ii) The system is not feedback linearizable since it does not satisfy the conditions required for Theorem 7 to hold (i.e., distributions will depend upon the value of $u$).

Since the system is not feedback linearizable, the procedure used in the previous simulation cannot be employed (if attempted, training of $NN_p$ and $NN_q$ will not converge). However, since the controllability rank condition around the origin is satisfied, we can attempt to stabilize the system directly using nonlinear state feedback.

In the adaptive control case where $f_1$, $f_2$, and $f_3$ are unknown, three neural networks $NN_{f_i} \subset \mathcal{N}^3_{2,10.5,1}$ were trained to emulate the system (one for each component of $f$). Then, the Jacobian matrix of the model around the origin can be determined to assure its controllability. For the problem under consideration, this is given by

$$A = \begin{bmatrix} 0.01 & 1.01 & -0.97 \\ 2.10 & 0.08 & 0.06 \\ -0.04 & 0.00 & 0.00 \end{bmatrix}, b = \begin{bmatrix} -0.01 \\ -0.98 \\ 1.09 \end{bmatrix}.$$
and controllability is easily verified.

After checking the controllability conditions, an additional network $NN_p \subset \mathbb{R}^{2,10,5,1}$ was trained to learn the controller. The objective was to train a controller that would stabilize the system inside $\mathcal{S}$, $\mathcal{S} = \{x||x|| \leq 2\}$. The initial conditions of the model were chosen randomly in $\mathcal{S}$ and the output of $NN_p$ was used as the input to the plant. After 3 steps (3 being the order of the system), the error between the final state of the system and the origin, i.e., $||x(3)||$, was used to adjust the weights of $NN_p$. The training of $NN_p$ was achieved after 50,000 steps.

After learning was complete, the feedback control system was tested by starting it at different initial conditions. Fig. 7 shows a typical trajectory with initial value $x_0 = (1, -1, 1)$. The same qualitative response was observed for other initial conditions within $\mathcal{S}$. For comparison purposes, a linear controller was designed based on the linearization of the system around the origin. With $u(k) = x_1(k)$, the linearized system is seen to be $\dot{x}(k+1) = A_1 x(k)$ where

$$A_1 = \begin{bmatrix} 0 & 1 & -1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

is a nilpotent matrix. While such a controller performed as expected for values of initial conditions $||x_0|| < 0.4$, it becomes unstable for larger initial conditions. The response of the system for the initial condition $x_0 = (0.15, -0.25, 0.35)$, shown in Fig. 8, indicates that the trajectory becomes very large after 10 units of time. Simulation 2 reveals how the method proposed using a nonlinear controller may increase the region of attraction substantially.

V. CONCLUSION

A general approach to the design of neural-network based controllers for nonlinear dynamical systems is discussed in the paper. The state vector of the nonlinear plant to be controlled is assumed to be accessible and the main objective is to stabilize the plant around an equilibrium state. The theoretical and practical questions that arise are examined and the prior information needed to design the controllers are specified. The most important aspect of the methods proposed is that they are practically viable. This is demonstrated through simulation studies on the stabilization of nonlinear systems for which other methods are not currently available.

The problem of control becomes substantially more complex (as is well known in linear theory) when the state of the system is not accessible and control has to be achieved using input-output data. The use of observers (or identifiers) and the problem of output regulation will be treated in two following reports [10], [11].

REFERENCES


Asriel E. Levin was born in Haifa, Israel on June 10, 1960. He received the B.Sc. degree in electrical engineering and the B.A. degree in physics from the Technion-Israel Institute of Technology in 1982, both with honors. From 1982 to 1987 he served with the Israeli Defense Force, Ordnance Corps and as a project officer was engaged in the development of electro-optical imaging systems. In 1987, he joined the Department of Electrical Engineering, Yale University, New Haven, CT, as a Ph.D. candidate and received the M.S., M.Phil., and Ph.D. degrees in 1989, 1990, and 1992, respectively. Currently he is a Senior Research Scientist with the Department of Computer Science and Engineering at the Oregon Graduate Institute of Science and Technology. His research interests include neural networks, learning systems, decision dynamics in distributed and hierarchical environments, dynamical systems, and Markov decision processes.

Kumpati S. Narendra (S’55–M’56–SM’63–F’79) is currently Professor of Electrical Engineering and Director of the Center for Systems Science at Yale University. He is the author of numerous technical publications in the area of systems theory. He is the author of three books: Frequency Domain Criteria for Absolute Stability (co-author J. H. Taylor), Stable Adaptive Systems (co-author A. M. Annaswamy) and Learning Automata — An Introduction (co-author M. A. L. Thathachar), and editor of four books (Learning Automata (Journal of Cybernetics and Information Science, 1977), Applications of Adaptive Control (with R. V. Monopoli, Academic Press, 1980), Adaptive and Learning Systems (Plenum Press, 1985), and Advances in Adaptive Control (with R. Ortega and P. Dorato, IEEE Press, 1991). His research interests are in the areas of stability theory, adaptive control, learning automata and the control of complex systems using neural networks. He has served on several national and international technical committees as well as the editorial boards of several technical journals.

Dr. Narendra is a member of Sigma Xi and a fellow of IEE (United Kingdom), and the American Association for the Advancement of Science. He was the recipient of the 1972 Franklin V. Taylor Award of the IEEE SMC Society, the 1988 George S. Axelby Best Paper Award of the Control Systems Society, the 1990 American Automatic Control Council Education Award and the 1991 Outstanding Paper Award of the Neural Network Council.