NONLINEAR NETWORK STRUCTURES FOR FEEDBACK CONTROL

F.L. Lewis

ABSTRACT

A framework is given for controller design using Nonlinear Network Structures, which include both neural networks and fuzzy logic systems. These structures possess a universal approximation property that allows them to be used in feedback control of unknown systems without requirements for linearity in the system parameters or finding a regression matrix. Nonlinear nets can be linear or nonlinear in the tunable weight parameters. In the latter case weight tuning algorithms are not straightforward to obtain. Feedback control topologies and weight tuning algorithms are given here that guarantee closed-loop stability and bounded weights. Extensions are discussed to force control, backstepping control, and output feedback control, where dynamic nonlinear nets are required.


I. INTRODUCTION

In recent years, there has been a great deal of effort to design feedback control systems that mimic the functions of living biological systems [57,39]. 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. Neural networks (NNs) have achieved great success in classification, pattern recognition, and other open-loop applications in digital signal processing and elsewhere. Rigorous analyses have shown how to select NN topologies and weights, for instance, to discriminate between specified exemplar patterns. By now, the theory and applications of NN in open-loop applications are well understood, so that NN have become an important tool in the repertoire of the signal processor and computer scientist.

With regards to the use of NN for control, one must distinguish between two classes of control applications—open-loop identification and closed-loop feedback control. Identification applications are close in spirit to signal processing/classification, so that the same open-loop algorithms (e.g. backpropagation weight tuning) may often be used. On the other hand, in closed-loop feedback applications the NN is inside the control loop so that special steps must be taken to ensure that the tracking error and the NN weights remain bounded in the closed-loop system.

There is a large literature on NN for feedback control of unknown plants. Until the 1990’s, design and analysis techniques were ad hoc, with no repeatable design algorithms or proofs of stability and guaranteed performance. In spite of this, simulation results appearing in the literature showed good performance. Most of the early approaches used standard backpropagation weight tuning [56] since rigorous derivations of tuning algorithms suitable for closed-loop control purposes were not available. Many NN design techniques mimicked adaptive control approaches, where rigorous analysis results were available [1,29,12], proposing NN feedback control topologies such as indirect identification-based control, inverse dynamics control, series-parallel techniques, etc.

In these early NN control techniques there were serious unanswered questions. Since it was not known how to initialize the NN weights to provide closed-loop stability, most approaches required an off-line learning phase, where the NN weights were tuned using measurements of system inputs and outputs in a preliminary phase before the controller was allowed to provide closed-loop system inputs. Such an open-loop phase has serious detrimental repercussions for industrial and mechanical
systems where control is usually required immediately. Moreover, in early applications of direct closed-loop control the gradients (jacobians) needed for backpropagation weight tuning depended on the unknown system and/or satisfied their own differential equations; this made them impossible or very difficult to compute. The most serious problem was that rigorous stability proofs and guarantees of closed-loop performance were not available, so that the performance of these controllers on actual industrial or mechanical systems was open to serious question.

The basic problem issues in NN closed-loop feedback control are to provide repeatable design algorithms, to provide on-line learning algorithms that do not require preliminary off-line tuning, to show how to initialize the NN weights to guarantee stability, to rigorously prove closed-loop trajectory following, to show how to compute various gradients needed for weight tuning, to show that the NN weights remain bounded despite unmodelled dynamics (since bounded weights guarantee bounded control signals).

In keeping with the philosophy of those working in control system theory since Maxwell, Lyapunov, A.N. Whitehead, von Bertalanffy, and others, to address such issues it is necessary to begin with the knowledge available about the system being controlled. Narendra [40, 42] and others [39, 57, 56] pioneered rigorous NN controls applications by studying the dynamical behavior of NN in closed-loop systems, including computation of the gradients needed for backprop tuning. Other groups providing rigorous analysis and design techniques for closed-loop NN controllers included [47] who showed how to use radial basis function NN in feedback control, [4, 5] who provided NN tuning algorithms based on deadzone methods, [44, 43] who used projection methods, [34, 35] who used an e-mod approach, [46] who provided NN controllers for discrete-time systems, and [45] who used dynamic NN for feedback control. By now many researchers are following this lead and are offering rigorous design techniques for various sorts of NN controllers. Included are [38] and related papers by that group where NN are used for dynamic inversion aircraft control, and the work in [52] and related papers where NN controllers are designed for general nonlinear systems. By now the first textbooks in NN control have appeared [31, 11, 21].

In this paper is given a comprehensive approach to the design and analysis of nonlinear network controllers based on several Ph.D. dissertations and a body of published work, including [31, 21]. Both neural network and fuzzy logic systems are included in this class of nonlinear networks. More details are provided in [31] and the papers by the author and co-workers. All of the basic problem issues mentioned are solved for a large class of mechanical motion systems with Lagrangian dynamics, including robotic manipulators. The control structures discussed here are multiloop controllers with NN in some of the loops and an outer tracking unity-gain PD 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 performance requirements increase, the NN controllers require more and more structure.

NN controllers have advantages over standard adaptive robot control approaches in that no linearity-in-the-parameters assumption is needed and no ‘regression matrix’ must be determined. This is primarily due to the NN universal function approximation property. NN controllers may be called ‘nonparametric controllers’ in that they are not parametrized in terms of system parameters (see also [7] which provides a nonparametric adaptive robot control approach). If designed correctly, the NN controller does not need persistence of excitation or certainty equivalence assumptions, even in discrete-time.

In this paper, some discussion is first provided on multilayer nonlinear-in-the-parameters (NLIP) neural nets, then on linear-in-the-parameters (LIP) networks including FLNN, RBF, and CMAC. Fuzzy logic nets are addressed and related to neural nets; the nonlinear Network design techniques provided herein work for both. Tracking controllers are designed for LIP and then NLIP NN, with tuning algorithms provided that guarantee closed-loop stability and bounded weights; the backprop algorithm is not generally suitable but must be modified for closed-loop control. Passivity and robustness properties are defined and studied for NN. Some extensions are discussed, including NN force control, NN backstepping control, and output-feedback control where dynamic NN are needed.

II. BACKGROUND IN NEURAL AND FUZZY NONLINEAR NETWORKS

In this section is provided the background in nonlinear network structures required for feedback control. For more details see [15, 31, 32]. Both neural network (NN) and fuzzy logic (FL) structures [53] are discussed. Their similarities are stressed. Two key features that make NN and FL systems useful for feedback control are their universal approximation property and their learning capability, which arises due to the fact that their weights are tunable parameters that can be updated to improve controller performance. The universal approximation property is the main feature that makes nonlinear network structures more suitable for robot control than adaptive robot controllers, which have generally relied upon the determination of a regression matrix, which in turn requires linearity in the tunable parameters (LIP). Nonlinear networks can have one layer of tunable parameters or several. In the former case the net is linear in the parameters and standard adaptive control proofs may be used to derive weight tuning algorithms that guarantee closed-loop stability. In the latter case the net is nonlinear.
in the parameters (NLIP), which affords many advantages yet causes additional difficulties in deriving stabilizing tuning algorithms.

2.1 Neural network structures and properties

First multilayer NLIP nets are covered then 1-layer LIP nets, including functional-link NN, RBF, and CMAC.

2.1.1 Multilayer neural networks

A multilayer neural network is shown in Fig. 1. This NN has two layers of adjustable weights, and is called here a ‘two-layer’ net. This NN has no internal feedback connections and so is termed feedforward, and no internal dynamics and so is said to be static. The NN output $y$ is a vector with $m$ components that are determined in terms of the $n$ components of the input vector $x$ by the recall equation

$$y_i = \sum_{j=1}^{L} w_{ij} \sigma(\sum_{k=1}^{N} v_{jk} x_k + \theta_{w_i}) + \theta_{v_i}; \quad i = 1 \ldots, m$$

(1)

where $\sigma(\cdot)$ are the activation functions and $L$ is the number of hidden-layer neurons. The first-layer interconnections weights are denoted by $v_{jk}$ and the second-layer interconnection weights by $w_{ij}$. The threshold offsets are denoted by $\theta_{v_k}, \theta_{w_i}$.

Many different activation functions $\sigma(\cdot)$ are in common use. For feedback control using multilayer NN it is required that $\sigma(\cdot)$ be smooth enough so that at least its first derivative exists. Suitable choices include the sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}},$$

(2)

the hyperbolic tangent

$$\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}},$$

(3)

and other logistic-curve-type functions, as well as the gaussian and an assortment of other functions.

By collecting all the NN weights $v_{jk}, w_{ij}$ into matrices of weights $V^T, W^T$, the NN recall equation may be written in terms of vectors as

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

(4)

The thresholds are included as the first columns of the weight matrices $W^T, V^T$; to accommodate this, the vectors $x$ and $\sigma(\cdot)$ need to be augmented by placing a ’1’ as their first element (e.g. $x \equiv [1, x_1, x_2, \ldots x_N]^T$). In this equation, to represent (1) one has sufficient generality if $\sigma(\cdot)$ is taken as a diagonal function from $\mathbb{R}^L$ to $\mathbb{R}^L$, that is $\sigma(z) \equiv \text{diag}\{\sigma(z_j)\}$ for a vector $z = [z_1, z_2, \ldots, z_L]^T \in \mathbb{R}^L$.

Note that the recall equation is nonlinear in the first-layer weights and thresholds $V$.

Universal Function Approximation Property. NN satisfy many important properties. A main one of concern for feedback control purposes is the universal function approximation property [16]. Let $f(x)$ be a general smooth function from $\mathbb{R}^n$ to $\mathbb{R}^n$. Then, it can be shown that, as long as $x$ is restricted to a compact set $S$ of $\mathbb{R}^n$, there exist weights and thresholds such that one has

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

(5)

for some number of hidden-layer neurons $L$. This holds for a large class of activation functions, including those just mentioned. The value $\epsilon$ is called the NN functional approximation error, and it generally decreases as the net size $L$ increases. In fact, for any choice of a positive number $\epsilon_N$, one can find a feedforward NN such that

$$|\epsilon| < \epsilon_N$$

(6)

for all $x \in S$. The selection of the net size $L$ for a prescribed accuracy $\epsilon_N$ is an open question for general unstructured fully-connected NN. This problem can be solved for CMAC, FL nets, and other structured nonlinear nets as subsequently described.

The ideal NN weights in matrices $W, V$ that are needed to best approximate a given nonlinear function $f(x)$ are difficult to determine. In fact, they may not even be unique. However, all one needs to know for controls purposes is that, for a specified value of $\epsilon_N$, some ideal approximating NN weights exist. Then, an estimate of $f(x)$ can be given by

$$\hat{f}(x) = W^T \sigma(V^T x)$$

(7)
where $\hat{W}$, $\hat{V}$ are estimates of the ideal NN weights that are provided by some on-line weight tuning algorithms subsequently to be detailed. Note that all ‘hat’ quantities are known, since they may be computed using measurable signals and current NN weight estimates.

The assumption that there exist ideal weights such that the approximation property holds is very much like various similar assumptions in adaptive control [1,29], including Erzberger’s assumptions and linearity in the parameters. The very important difference is that in the NN case, the universal approximation property always holds, while in adaptive control such assumptions often do not hold in practice, and so they imply restrictions on the form of the systems that can be controlled.

**Overcoming the NLIP Problem.** Multilayer NN are nonlinear in the weights $V$ and so weight tuning algorithms that yield guaranteed stability and bounded weights in closed-loop feedback systems have been difficult to discover until a few years ago. The NLIP problem is easily overcome if a correct and rigorous approach is taken. One approach is the following appropriate use of the Taylor series [37].

Define the functional estimation error
\begin{equation}
\hat{f} = f - \hat{f},
\end{equation}
the weight estimation errors
\begin{equation}
\hat{W} = W - \hat{W}, \quad \hat{V} = V - \hat{V},
\end{equation}
and the hidden-layer output error
\begin{equation}
\hat{\sigma} = \sigma - \hat{\sigma} \equiv \sigma(V^T x) - \sigma(\hat{V}^T x).
\end{equation}
For any $z$ one may write the Taylor series
\begin{equation}
\sigma(z) = \sigma(z) + \sigma'(z)z + O(z)^2,
\end{equation}
where $\sigma'$ is the jacobian and the last term means terms of order $z^2$. Therefore,
\begin{equation}
\hat{\sigma} = \sigma'(V^T x) \hat{V}^T x + O(\hat{V}^T x)^2 \equiv \hat{\sigma} \hat{V}^T x + O(\hat{V}^T x)^2. \tag{12}
\end{equation}
This key equation allows one to write the functional estimation error as
\begin{align}
\hat{f} &= f - \hat{f} = W^T \sigma(V^T x) - \hat{W}^T \sigma(\hat{V}^T x) + \epsilon \\
&= W^T \sigma(V^T x) + W^T [\sigma(V^T x) - \sigma(\hat{V}^T x)] + \epsilon \\
&= W^T \sigma + W^T [\sigma \sigma' V x + O(\sigma(V^T x)^2)] + \epsilon \\
&= W^T [\sigma - \hat{\sigma} V x] + W^T \sigma' \hat{V}^T x + \hat{W}^T \sigma' \hat{V}^T x \\
&\quad + \hat{W}^T O(\hat{V}^T x)^2 + \epsilon. \tag{13}
\end{align}
The first term has $\hat{W}$ multiplied by a known quantity (in square braces), and the second term has $\hat{V}$ multiplied by a known quantity. When used subsequently in the closed-loop error dynamics, this form allows one to derive tuning laws for $\hat{V}$ and $\hat{W}$ that guarantee closed-loop stability.

**Weight Tuning Algorithms.** Multilayer NN are nonlinear in the weights, which presents a challenge in deriving weight tuning algorithms in closed-loop feedback control systems that yield stability as well as bounded weights. Tuning algorithms that are suitable for closed-loop control are given later in this paper. The problem of suitably tuning the weights in open-loop applications (e.g. system identification [40]) has been solved for years. Many types of NN weight tuning algorithms are used, usually based on some sort of gradient algorithm. Tuning algorithms may either be given in continuous-time or in discrete-time, where the weights are updated only at discrete time points (e.g. the delta rule [15]). Discrete-time tuning is potentially useful in digital control applications of neural networks.

A common weight tuning algorithm is the gradient algorithm based on the backpropagated error [55], where the NN is trained to match specified exemplar pairs $(x_o, y_o)$, with $x_o$, the ideal NN input that yields the desired NN output $y_o$. The discrete-time version of the backpropagation algorithm for the two-layer NN is given by
\begin{equation}
\hat{W}_{k+1} = \hat{W}_k + F \sigma(\hat{V}_{k, x}^T E_k^T) \tag{14}
\end{equation}
\begin{equation}
\hat{V}_{k+1} = \hat{V}_k + G x \left( \sigma'(\hat{W}_k^T E_k) \right)^T,
\end{equation}
where $k$ is the discrete time index and $F$, $G$ are positive definite design parameter matrices governing the speed of convergence of the algorithm. The error $E_k$ that is backpropagated is selected as the desired NN output minus the actual NN output $\hat{y}_k = y_k - y_d$.

Note that the hidden-layer output gradient or jacobian $\hat{\sigma}'$ is the same as the jacobian in (12). It may be explicitly computed; for the sigmoid activation functions, for instance, it is
\begin{equation}
\hat{\sigma}' = \sigma'(\hat{V}^T x) = \text{diag} \{ \sigma(\hat{V}^T x) \} \left[ I - \text{diag} \{ \sigma(\hat{V}^T x) \} \right]. \tag{15}
\end{equation}
where $\text{diag} \{ z \}$ means a diagonal matrix whose diagonal elements are the components of the vector $z$. Backprop tuning is accomplished off-line and requires specified training data pairs $(x_o, y_o)$, so it is a supervised training scheme.

The continuous-time version of the backpropagation algorithm for the two-layer NN is given by
\begin{equation}
\dot{W} = F \sigma(\hat{V}^T x) E^T \tag{16}
\end{equation}
\begin{equation}
\dot{V} = G x \left( \sigma'(\hat{W} E) \right)^T.
\end{equation}
A simplified NN weight tuning scheme is the Hebbian algorithm, a continuous-time version of which is

\[
\dot{W} = F(\sigma(V^T x)) E^T
\]

\[
\dot{V} = G(\sigma(V^T x))^T .
\]  

(17)

In Hebbian tuning, no Jacobian need be computed; instead, the weights in each layer are updated based on the outer product of the input and output signals of that layer. The Hebbian algorithm is based on classical conditioning experiments in psychology and associative memory paradigms.

It has been difficult to apply standard backprop tuning to closed-loop feedback control since the Jacobian as computed in feedback loops depends on the unknown plant, it is not known how to obtain exemplar pairs \((x, y)\), it is unclear how to initialize the weight estimates, and it is not obvious what signal should be backpropagated (i.e. how does \(E\) relate to the tracking error?). Some obvious solutions to some of these problems are to use preliminary off-line training, where a NN model of the plant is first estimated off-line and then used for control purposes, or indirect control schemes, where a model of the plant must be estimated independently of the controller. Neither of these approaches provides the most efficient controllers. Narendra and coworkers [40,42] have worked to overcome these approaches provides the most efficient controllers. Narendra and coworkers [40,42] have worked to overcome these problems, showing that, in closed-loop systems, the Jacobians required for backprop tuning themselves satisfy differential equations.

2.1.2 Linear-in-the-parameter neural nets

If the first-layer weights and thresholds \(V\) in (4) are fixed and only the second-layer weights and thresholds \(W\) are tuned, then the NN has only one layer of tunable weights. One may then define the fixed function \(\phi(x) = \sigma(V^T x)\) so that such a 1-layer NN has the recall equation

\[
y = W^\phi(x),
\]  

(18)

where \(x \in \mathbb{R}^n, y \in \mathbb{R}^m, \phi(\cdot) : \mathbb{R}^n \to \mathbb{R}^m\), and \(L\) is the number of hidden-layer neurons. This NN is linear in the tunable parameters \(W\), so that it is far easier to tune. In fact, standard adaptive control proofs can be used to derive suitable tuning algorithms.

Though LIP, these NN still offer an enormous advantage over standard adaptive control approaches that require the determination of a regression matrix since they satisfy a universal approximation property if the functions \(\phi(\cdot)\) are correctly chosen. Note that adaptive controllers are linear in the system parameters, while 1-layer NN are linear in the NN weights. An advantage of 1-layer NN over 2-layer NN is that firm results exist for the former on the selection of the number of hidden neurons \(L\) for a specified approximation accuracy. A disadvantage of LIP networks is that Barron [3] has shown a lower bound on the approximation accuracy.

**Functional-Link Basis Neural Networks.** More generality is gained if \(\sigma(\cdot)\) is not diagonal, but \(\phi(\cdot)\) is allowed to be a general function from \(\mathbb{R}^n\) to \(\mathbb{R}^L\). This is called a functional-link neural net (FLNN) [46]. For LIP NN, the functional approximation property does not generally hold. However, a 1-layer NN can still approximate functions as long as the activation functions \(\phi(\cdot)\) are selected as a basis, which must satisfy the following two requirements on a compact simply-connected set \(\mathcal{S}\) of \(\mathbb{R}^n\):

1. A constant function on \(\mathcal{S}\) can be expressed as (18) for a finite number \(L\) of hidden-layer neurons.
2. The functional range of (18) is dense in the space of continuous functions from \(\mathcal{S}\) to \(\mathbb{R}^m\) for countable \(L\).

If \(\phi(\cdot)\) provides a basis, then a smooth function \(f(x)\) from \(\mathbb{R}^n\) to \(\mathbb{R}^m\) can be approximated on a compact set \(\mathcal{S}\) of \(\mathbb{R}^n\), by

\[
f(x) = W^\phi(x) + \epsilon .
\]  

(19)

for some ideal weights and thresholds \(W\) and some number of hidden layer neurons \(L\). In fact, for any choice of a positive number \(\epsilon_0\), one can find a feedforward NN such that \(|\epsilon| < \epsilon_0\) for all \(x\) in \(\mathcal{S}\).

The approximation for \(f(x)\) is given by

\[
\tilde{f}(x) = W^T \phi(x) ,
\]  

(20)

where \(\tilde{W}\) are the NN weight and threshold estimates given by the tuning algorithm. For LIP NN, the functional estimation error is given by

\[
\tilde{f} = f - \tilde{f} = W^T \phi(x) - W^T \phi(x) + \epsilon
\]

\[= W^T \phi(x) + \epsilon ,
\]  

(21)

which, when used subsequently in the error dynamics, directly yields tuning algorithms for \(\tilde{W}\) that guarantee closed-loop stability.

Barron [3] has shown that for all LIP approximators there is a fundamental lower bound, so that \(\epsilon\) is bounded below by terms on the order of \(1/L^2\). Thus, as the number of NN inputs \(n\) increases, increasing \(L\) to improve the approximation accuracy becomes less effective. As shown subsequently, this is not a major limitation on adaptive controllers designed using 1-layer NN. This lower bound problem does not occur in the NLIP multi-layer nonlinear nets.

Some special FLNN are now discussed. We often use \(\sigma(\cdot)\) in place of \(\phi(\cdot)\), with the understanding that, for LIP nets, this activation function vector is not diagonal, but is a general function from \(\mathbb{R}^n\) to \(\mathbb{R}^L\).
Random Vector Functional Link (RVFL) Nets. It is often difficult to select the activation functions $\sigma(\cdot)$ in unstructured LIP NN so that they provide a basis. This problem may be addressed as shown below (RBF and CMAC nets) by introducing additional structure. Another approach is to select the matrix $V$ in (4) randomly. It is shown in [17] that, for these random vector functional link (RVFL) nets, the resulting function $\phi(x) = \sigma(Vx)$ is a basis so that the RVFL NN has the universal approximation property.

When using sigmoid activation functions (2), this approach amounts to randomly selecting the sigmoid scaling parameters $\upsilon_{jk}$ and shift parameters $\theta_{jk}$ in $\sigma(\Sigma_{j}\upsilon_{jk}x_{k} + \theta_{jk})$. This produces a family of $L$ activation functions with different scaling and shifts [21].

Gaussian or Radial Basis Function (RBF) Networks. The selection of a basis set of activation functions is considerably simplified in various sorts of structured nonlinear networks, including radial basis function, CMAC, and fuzzy logic nets. It will be shown here that the key to the design of such structured nonlinear nets lies in a more general set of NN thresholds than allowed in the standard equation (1), and in their appropriate selection.

A NN activation function often used is the gaussian or radial basis function (RBF) [47] given when $x$ is a scalar as

$$\sigma(x) = e^{-(x-x_{j})^{2} / 2p_{j}},$$

where $x$ is the mean and $p$ the variance. RBF NN can be written as (4), but have an advantage over the usual sigmoid NN in that the $n$-dimensional gaussian function is well understood from probability theory, Kalman filtering, and elsewhere, so that $n$-dimensional RBF are easy to conceptualize.

The $j$-th activation function can be written as

$$\sigma_{j}(x) = e^{-(x-x_{j})^{T}p_{j}^{-1}(x-x_{j})}$$

with $x, x_{j} \in \mathbb{R}^{n}$. Define the vector of activation functions as $\sigma(x) = [\sigma_{1}(x) \sigma_{2}(x) \ldots \sigma_{L}(x)]^{T}$. If the covariance matrix is diagonal so that $P_{j} = \text{diag}(p_{jk})$, then (23) becomes separable and may be decomposed into components as

$$\sigma_{j}(x) = e^{-\frac{1}{2}x^{T}x_{j} - \frac{1}{2}(x - x_{j})^{T}p_{j}(x - x_{j})} = \prod_{k=1}^{n} e^{-\frac{1}{2}(x_{j}^{k} - x_{k})^{2} / p_{jk}},$$

where $x_{j}, x_{k}$ are the $k$-th components of $x, x_{j}$. Thus, the $n$-dimensional activation functions are the product of $n$ scalar functions. Note that this equation is of the form of the activation functions in (1), but with more general thresholds, as a threshold is required for each different component of $x$ at each hidden layer neuron $j$; that is, the threshold at each hidden-layer neuron in Fig. 1 is a vector. The RBF variances $p_{jk}$ and offsets $x_{j}$ are usually selected in designing the RBF NN and left fixed; only the output-layer weights $W$ are generally tuned. Therefore, the RBF NN is a special sort of FLNN (18) (where $\phi(x) = \sigma(x)$).

Figure 2 shows separable gaussians for the case $x \in \mathbb{R}^{2}$. In this figure, all the variances $p_{jk}$ are identical, and the mean values $x_{j}$ are chosen in a special way that spaces the activation functions at the node points of a 2-D grid. To form an RBF NN that approximates functions over the region $\{-1 < x_{1} \leq 1, -1 < x_{2} \leq 1\}$ one has here selected $L = 5 \times 5 = 25$ hidden-layer neurons, corresponding to 5 cells along $x_{1}$ and 5 along $x_{2}$. Nine of these neurons have 2-D gaussian activation functions, while those along the boundary require the illustrated ‘one-sided’ activation functions.

An alternative to selecting the gaussian means and variances is to use random choice as in the RVFL approach. In 2-D, for instance (c.f. Fig. 2), this produces a set of $L$ gaussians scattered at random over the $(x_{1}, x_{2})$ plane with different variances.

The importance of RBF NN is that they show how to select the activation functions and the number of hidden-layer neurons for specific NN applications (e.g. function approximation— see below).

Cerebellar Model Articulation Controller (CMAC) Nets. A CMAC NN [2] has separable activation functions generally composed of splines. The activation functions of a 2-D CMAC composed of second-order splines (e.g. triangle functions) are shown in Fig. 3, where $L = 5 \times 5 = 25$. The activation functions of a CMAC NN are called receptive field functions in analogy with the optical receptor fields of the eye. An advantage of CMAC NN is that the receptive field functions based on splines have finite support so that they may be efficiently evaluated. An additional computational advantage is provided by the
fact that higher-order splines may be computed recursively from lower-order splines.

On the Required Number of Hidden-Layer Neurons. The problem of determining the number of hidden-layer neurons for general fully-connected NN (4) has not been solved. However, for NN such as RBF or CMAC there is sufficient structure to allow a solution to this problem. For CMAC, for instance, one solution is as follows.

Let \( x \in \mathbb{R}^n \) and define uniform partitions in each component \( x_k \). Let \( \delta_k \) be the partition interval for \( x_k \) and \( \delta \equiv \sqrt[n]{\sum_{k=1}^{n} \delta_k^2} \). (e.g. In Fig. 3 where \( n = 2 \), one has \( \delta_1 = \delta_2 = 0.5 \).) The next result shows the maximum partition size \( \delta \) allowed for approximation with a desired accuracy \( \epsilon \) [8].

**Theorem 1.** (Partition Interval for CMAC Approximation) Let a function \( f(x) : \mathbb{R}^n \rightarrow \mathbb{R}^m \) be continuous with Lipschitz constant \( \lambda \) so that

\[
|f(x) - f(z)| \leq \lambda |x - z|
\]

for all \( x, z \) in some compact set \( S \) of \( \mathbb{R}^n \). Construct a CMAC with triangular receptive field functions \( \phi(\cdot) \) in the recall equation (20). Then there exist weights \( W \) such that

\[
|f(x) - \hat{f}(x)| \leq \epsilon
\]

for all \( x \in S \) if the CMAC is designed so that

\[
\delta \leq \frac{\epsilon}{m \lambda}.
\]

In fact, CMAC designed with this partition interval can approximate on \( S \) any continuous function smooth enough to satisfy the Lipschitz condition for the given \( \lambda \). Now, given limits on the dimensions of \( S \) one can translate this upper bound on \( \delta \) to a lower bound on the number \( L \) of hidden-layer neurons. Note that as the functions \( f(x) \) become less smooth, so that \( \lambda \) increases, the grid nodes become more finely spaced so that the required number \( L \) of hidden-layer neurons increases.

In [47] is given a similar result for designing RBF which selects the fineness of the grid partition based on a frequency-domain smoothness measure for \( f(x) \) instead of a Lipschitz constant smoothness measure.

### 2.2 Fuzzy logic systems

There are many ways to bring together NN and fuzzy logic (FL) systems [25,53] including architectures having both NN and FL components, e.g., using FL systems to initialize NN weights or NN to adapt FL membership functions. However, one point of view is to consider FL systems as a special class of structured nonlinear networks. RBF and CMAC NN introduce structure to the NN by using separability of the \( n \)-D activation functions into products of \( 1 \)-D functions. FL networks introduce additional structure by formalizing the selection of the NN weights and thresholds by relating the \( 1 \)-D functions to concepts in human linguistics. It can be shown that fuzzy logic systems using product inferencing and weighted defuzzification are equivalent to special sorts of nonlinear nets with suitably chosen separable activation functions, variances, and offsets.

**Membership Functions and Defuzzification.** Refer at this point to (1). In FL systems [33], a scalar function \( \sigma(\cdot) \) is selected, often the triangle function. Then, for \( x \in \mathbb{R}^n \) with components \( x_i \), the scalar function

\[
X^j_i(x_i) \equiv \sigma(v_{j-k} x_k + \theta_{j-k})
\]

is the \( j \)-th membership function (MF) along component \( x_i \), shifted by \( \theta_{j-k} \) and scaled by \( v_{j-k} \). See Fig. 4. Using product inferencing, the \( n \)-D MFs are composed using multiplication of the scalar MFs so that

\[
X_{j_1,j_2,\ldots,j_n}(x) = X_{j_1}^1(x_1)X_{j_2}^2(x_2)\cdots X_{j_n}^n(x_n)
\]

for a specified index set \( \{j_1, j_2, \ldots, j_n\} \). Thus, in FL nets, the \( n \)-D MFs are by construction separable in terms of products of \( 1 \)-D functions, as in (24).

FL systems are very closely related to the RBF and CMAC NN. In fact, the RBF NN in Fig. 2 is equivalent to a 2-D fuzzy system with gaussian membership functions along \( x_1 \) and \( x_2 \), while the CMAC NN in Fig. 3 is equivalent to a 2-D FL system with triangle membership functions. Note that FL systems have more general thresholds than (1), since, as for RBF and CMAC NN, the thresholds at each hidden-layer neuron must be \( n \)-vectors.

With \( y \in \mathbb{R}^n \) having components \( y_i \), using centroid defuzzification the \( i \)-th output component is given as
If both $W$ and $V$ are adapted, the FL system is NLIP and possesses the universal approximation property (5). Then, for tuning $V$ one requires MF that are differentiable. The jacobians required for backprop tuning in the NLIP case using e.g., the Gaussian MFs are easy to compute.

For closed-loop feedback control one must select parameter tuning algorithms that guarantee stability and bounded parameters [53]. This may be accomplished as detailed in upcoming sections of this paper.

### III. BACKGROUND IN DYNAMICAL SYSTEMS

Given a matrix $A = [a_{i,j}]$ the Frobenius norm is defined by

$$
\left| A \right|_F = \text{tr}(A) = \sum_{i,j} a_{ij}^2
$$

with $\text{tr}(\cdot)$ the trace. The associated inner product is $\langle A, B \rangle_F = A^T B$ for compatibly dimensioned $B$. Though the Frobenius norm cannot be defined as the matrix induced norm for any vector norm, it is compatible with the $2$-norm so that for any vector $x$, $\left| A x \right|_F \leq \left| A \right|_F \cdot \left| x \right|_2$. All matrix norms used here shall be understood to be the Frobenius norm.

A vector $w(t)$ is said to be persistently exciting (PE) if there exist positive constants $\delta, \alpha_1, \alpha_2$ such that

$$
\alpha_1 I \leq \int_{t_0}^{t_0 + \delta} w(\tau)w^T(\tau) d\tau \leq \alpha_2 I
$$

for all $t_0 \geq 0$.

Given the dynamical system $\dot{x} = f(x)$, the solution is said to be uniformly ultimately bounded (UUB) if there exists a compact set $S \subset \mathbb{R}^n$ so that for all $x_0 \in S$ there exists a bound $B$ and a time $T(B, x_0)$ such that $\|x(t)\| \leq B$ for all $t \geq t_0 + T$.

The system $\dot{x} = f(x, u), y = h(x, u)$ is said to be passive if it verifies an equality of the so-called power form [50]

$$
\dot{L}(t) = y^T u - g(t)
$$

for some lower-bounded $L(t)$ and some $g(t) \geq 0$. That is,

$$
\int_{t_0}^{T} y^T(\tau) u(\tau) d\tau \geq \int_{t_0}^{T} g(\tau) d\tau - \gamma^2
$$

for all $T \geq 0$ and some $\gamma \geq 0$. The system is dissipative if it is passive and in addition

$$
\int_{t_0}^{\infty} y^T(\tau) u(\tau) d\tau \neq 0 \implies \int_{t_0}^{\infty} g(\tau) d\tau > 0.
$$
A special sort of dissipativity occurs if $g(t)$ is a quadratic function of $x$ with bounded coefficients. We call this state strict passivity (SSP); then, the norm of the internal states is bounded in terms of the power delivered to the system. Somewhat surprisingly, the concept of SSP has not been extensively used in the literature [34,48], though see [12] where input and output strict passivity are defined. SSP turns out to be pivotal in studying the passivity properties of NN controllers, and as shown in Subsection 4.6 allows one to conclude some internal boundedness properties without any assumptions of persistence of excitation.

IV. TRACKING CONTROL USING STATIC NONLINEAR NETWORKS

In this section is discussed feedback tracking control design using nonlinear nets and assuming full state-variable feedback. For more details see [31] and other listed references. If full state feedback is available, then a static feedforward NN suffices for control. A control topology and net tuning algorithms are provided here that guarantee closed-loop stability and bounded weights. The techniques to be discussed apply for general nonlinear nets including both neural networks and fuzzy logic systems [36,53], so that the abbreviation NN might henceforth be construed as meaning ‘nonlinear network’. The resulting multiloop control topology has an outer PD tracking loop and an inner NN feedback linearization or action generating loop. It is found that backprop tuning does not generally suffice, but modified tuning algorithms are needed for guaranteed closed-loop performance.

Many industrial mechanical systems, as well as automobiles, aircraft, and spacecraft, have dynamics in the Lagrangian form, which are exemplified by the class of rigid robot systems. Therefore, the Lagrangian robot dynamics will be considered [30,37]. The NN control techniques presented may also be applied to other unknown systems including certain important classes of nonlinear systems [31].

4.1 Robot arm dynamics and error system

The dynamics of rigid Lagrangian systems, including robot arms, have some important physical, structural, and passivity properties [30,9,49,51] that make it very natural to use NN in their control. These properties should be taken into account in the design of any controller — in fact, they provide the foundation for rigorous design algorithms for NN controllers.

The dynamics of an $n$-link rigid (i.e. no flexible links or high-frequency joint/motor dynamics) robot manipulator may be expressed in the Lagrange form

$$M(q)\ddot{q} + V_{\alpha}(q, \dot{q})\dot{q} + G(q) + F(q) + \tau_d = \tau$$  \hspace{1cm} (35)$$

with $q(t) \in \mathbb{R}^n$ the joint variable vector, whose entries are the robot arm joint angles or link extensions. $M(q)$ is the inertia matrix, $V_{\alpha}(q, \dot{q})$ the coriolis/centripetal matrix, $G(q)$ the gravity vector, and $F(q)$ the friction. Bounded unknown disturbances (including e.g. unstructured unmodelled dynamics) are denoted by $\tau_d$, and the control input torque is $\tau(t)$. The robot dynamics have the following standard properties:

Property 1. $M(q)$ is a positive definite symmetric matrix bounded by $m_1M < M(q) < m_2M$, with $m_1, m_2$ positive constants.

Property 2. The norm of the matrix $V_{\alpha}(q, \dot{q})$ is bounded by $\|u_\dot{\theta}(q)\| \|\dot{q}\|$, for some function $u_\dot{\theta}(q)$.

Property 3. The matrix $M - 2V_{\alpha}$ is skew-symmetric. This is equivalent to the fact that the internal forces do no work.

Property 4. The unknown disturbance satisfies $\|\tau_d\| < d_\tau$, with $d_\tau$ a positive constant.

Given a desired arm trajectory $q_d(t) \in \mathbb{R}^n$, the tracking error is

$$e(t) = q_d(t) - q(t)$$  \hspace{1cm} (36)$$

and the filtered tracking error is

$$r = \ddot{e} + \Lambda e$$  \hspace{1cm} (37)$$

where $\Lambda$ is a symmetric positive definite design parameter matrix, usually selected diagonal. If a controller is found such that $r(t)$ is bounded, then $e(t)$ is also bounded; in fact $\|e\| \leq \|r\|/\lambda_{\min}(\Lambda)$ and $\|e\| \leq \|r\|$, with $\lambda_{\min}(\Lambda)$ the minimum singular value of $\Lambda$.

Differentiating $r(t)$ and using (5), the arm dynamics may be written in terms of the filtered tracking error as

$$Mr = -V_{\alpha}r - \tau + f + \tau_d$$  \hspace{1cm} (38)$$

where the nonlinear robot function is

$$f(x) = M(q)(\dot{q}_d + \Lambda \dot{e}) + V_{\alpha}(q, \dot{q})(\dot{q}_d + \Lambda \dot{e}) + G(q) + F(q) \right.$$  \hspace{1cm} (39)$$

The vector $x$ required to compute $f(x)$ can be defined, for instance, as

$$x = [\dot{e}^T \dot{e}^T \dot{q}_d^T \dot{q}_d^T \dot{q}_d^T]^T,$$  \hspace{1cm} (40)$$

which can be measured. Function $f(x)$ contains potentially unknown robot parameters including payload mass and complex forms of friction.

A suitable control input for trajectory following is given by the computed-torque-like control
\[ \tau = \dot{f} + K_x \dot{r} - \nu \]  \hspace{1cm} (41)

with \( K_x = K_x^T > 0 \) a gain matrix, generally chosen diagonal, and \( \dot{f}(x) \) an estimate of the robot function \( f(x) \) that is provided by some means. The robustifying signal \( \nu(t) \) is needed to compensate for unmodelled unstructured disturbances. Using this control, the closed-loop error dynamics is

\[ M \dot{r} = - (K_x + V_o) r + \dot{f} + \tau_d + \nu . \]  \hspace{1cm} (42)

In computing the control signal, the estimate \( \dot{f} \) can be provided by several techniques, including adaptive control or neural or fuzzy networks. The auxiliary control signal \( \nu(t) \) can be selected by several techniques, including sliding-mode methods and others under the general aegis of robust control methods.

The desired trajectory is assumed bounded so that

\[
\begin{bmatrix}
q_d(t) \\
\dot{q}_d(t) \\
\ddot{q}_d(t)
\end{bmatrix}
\leq q_B ,
\]  \hspace{1cm} (43)

with \( q_B \) a known scalar bound. It is easy to show that for each time \( t, x(t) \) is bounded by

\[
|v| \leq c_1 + c_2 |r| \leq q_B + c_3 \sigma(0) + c_4 |r|
\]  \hspace{1cm} (44)

for computable positive constants \( c_0, c_1, c_2 \).

### 4.2 Adaptive control

Standard adaptive control techniques in robotics [9,30,49,51] use the assumption that the nonlinear robot function is linear in the tunable parameters so that

\[ \dot{f}(x) = \Theta(x) p + \epsilon \]  \hspace{1cm} (45)

where \( p \) is a vector of unknown parameters and \( \Theta(x) \) is a known regression matrix. The control is selected as

\[ \tau = K_x \dot{r} + \Theta(x) \dot{p} \]  \hspace{1cm} (46)

and tuning algorithms are determined for the parameter estimates \( \dot{p} \). This is facilitated by the LIP assumption. If the approximation error \( \epsilon \) is nonzero, then the tuning algorithm must be modified in order to guarantee bounded parameter estimates. Various robustifying techniques may be used for this including the \( \sigma \)-mod [18], \( e \)-modification [41], or dead-zone techniques [26].

There are adaptive control techniques by now that do not require LIP or the determination of a regression matrix [7]. It is interesting to compare the complexity of these to the NN controllers to be discussed herein.

### 4.3 Neural net feedback tracking controller

A NN will be used in the control \( \tau(t) \) in (41) to provide the estimate \( \dot{f} \) for the unknown robot function \( f(x) \). The NN approximation property assures us that there always exists a NN that can accomplish this within a given accuracy \( \epsilon_\nu \). For 2-layer NLIP NN the approximation is

\[ \dot{f}(x) = W^T \sigma(V^T x) , \]  \hspace{1cm} (47)

while for 1-layer LIP NN it is

\[ \dot{f}(x) = W^T \phi(x) , \]  \hspace{1cm} (48)

with \( \phi(.) \) selected as a basis.

The structure of the NN controller appears in Fig. 5, where \( \sigma \equiv [\sigma^T \ \sigma^T] \) and \( q \equiv [q^T \ q^T]^T \). The neural network that provides the estimate for \( f(x) \) appears in an inner control loop, and there is an outer tracking loop provided by the PD term \( K_x \dot{r} \). In control theory terminology, the inner loop is a feedback linearization controller [50], while in computer science terms it is the action generating loop [39,56,57]. This multiloop intelligent control structure is derived naturally from robot control notions, and is not ad hoc. As such, it is immune to philosophical deliberations concerning suitable NN control topologies including the common discussions on feedforward vs. feedback, direct vs. indirect, and so on. It is to be noted that the static feedforward NN in this diagram is turned into a dynamic NN by closing a feedback loop around it (c.f. [40]).

**Advantage of NN over LIP Adaptive Control.** Each robotic system has its own regression matrix, so that a different \( \Theta(x) \) must be determined for each system. This regression matrix is often complicated and determining it can be time consuming. One notes that the regression matrix effectively provides a basis for function approximation for a specific given system. On the other hand, the universal approximation property of nonlinear networks shows that NN provide a basis for all sufficiently smooth systems. Therefore, NN can be used to approximate smooth \( f(x) \) for all rigid robotic systems, effectively allow-
ing the design of a single tunable controller for a class of systems. No regression matrix need be determined, for the same NN activation functions suffice for the whole class of plants.

Even the 1-layer NN, which is LIP, has this advantage. Note that the 1-layer NN is linear not in the system parameters, but in the NN weights. Even the LIP NN has a universal approximation property so that no regression matrix is needed.

**Initial Tracking Errors and Initial NN Weights.** It is now required to determine how to tune the NN weights to yield guaranteed closed-loop stability. Several cases are considered for NN controller design in this section. All of them need the following construction.

Since the NN approximation property holds on a compact set, one must define an allowable initial condition set as follows. Let the NN approximation property hold for the function $f(x)$ given in (39) with a given accuracy $\epsilon_N$ in (6) for all $x$ inside the ball of radius $b_x > q_{B_x}$. Define the set of allowable initial tracking errors as

$$ S_r = \{r \ | \ |r| < (b_x - q_{B_x})/(c_0 + c_z) \}. $$

(49)

Note that the approximation accuracy of the NN determines size of $S_r$. For a larger NN (i.e. more hidden-layer units), $\epsilon_N$ is small for a larger radius $b_x$. Thus, the allowed initial condition set $S_r$ is larger. On the other hand, a more active desired trajectory (e.g. containing higher frequency components) results in a larger acceleration $\dot{q}_d(t)$, which yields a larger bound $q_{B_x}$ thereby decreasing $S_r$. It is important to note the dependence of $S_r$ on the PD design ratio $\Lambda$—both $c_0$ and $c_z$ depend on $\Lambda$.

A key feature of our the Initial Condition Requirement is its independence of the NN initial weights. This is in stark contrast to other techniques in the literature where the proofs of stability depend on selecting some initial stabilizing NN weights, which is very difficult to do.

### 4.4 Tuning algorithms for linear-in-the-parameters NN

Suppose now that a LIP FLNN is used to approximate the nonlinear robot function (39) according to (19) with $|e| < \epsilon_N$ on a compact set, the ideal approximating weights $\hat{W}$ constant, and $\phi(x)$ selected as a basis. An estimate of $f(x)$ is given by (48). Then the control law (41) becomes

$$ \tau = \hat{W}^T \phi(x) + K_r r - v \tag{50} $$

and the closed-loop filtered error dynamics (42) are

$$ M \ddot{r} = -(K_r + V_m) r + \hat{W}^T \phi(x) + (\epsilon + \tau_d) + v. \tag{51} $$

#### 4.4.1 Unsupervised backpropagation through time tuning

The next result shows how to tune the NN weights $\hat{W}$ on-line so as to guarantee stable tracking [34,35]

**Theorem 2.** (FLNN Controller with PE Requirement). Let the desired trajectory $q_d(t)$ be bounded by $q_{B_x}$ and the initial tracking error $r(0)$ be in $S_r$. Let the NN reconstruction error bound $\epsilon_N$ and the disturbance bound $d_B$ be constants. Let the control input for (35) be given by (50) with $u(t) = 0$ and gain satisfying the condition

$$ K_{\phi \min} \geq \frac{(\epsilon_N + d_B)(c_0 + c_z)}{b_x - q_{B_x}}. \tag{52} $$

on the minimum singular value of $K_\phi$. Let NN weight tuning be provided by

$$ \hat{W} = F \phi(x) r^T. \tag{53} $$

with $F = F^T > 0$ a constant design parameter matrix. Suppose the hidden-layer output $\phi(x)$ is persistently exciting. Then the filtered tracking error $r(t)$ is UUB, with a practical bound given by the right-hand side of (59), and the NN weight estimates $\hat{W}$ are bounded. Moreover, $r(t)$ may be kept as small as desired by increasing the gain $K_\phi$.

**Proof.** Let the NN approximation property (19) hold for the function $f(x)$ given in (39) with a given accuracy $\epsilon_N$ for all $x$ in the compact set $S_r \equiv \{x \ | \ |x| < b_x\}$ with $b_x > q_{B_x}$. Let $r(0) \in S_r$. Then the approximation property holds at time 0.

Define the Lyapunov function candidate

$$ L = \frac{1}{2} r^T M r + \frac{1}{2} tr(\hat{W}^T F^{-1} \hat{W}) \tag{54} $$

Differentiating yields

$$ \dot{L} = r^T M r + \frac{1}{2} r^T M r + tr(\hat{W}^T F^{-1} \hat{W}) \tag{55} $$

whence substitution from (51) yields

$$ \dot{L} = -r^T K_r r + \frac{1}{2} r^T (M - 2V_m) r + tr(\hat{W}^T (F^{-1} \hat{W} + \phi r^T)) $$

$$ + r^T (\epsilon + \tau_d). \tag{56} $$

The skew symmetry property makes the second term zero and the third term is zero if we select

$$ \hat{W} = -F \phi r^T \tag{57} $$

Since $\hat{W} = W - \hat{W}$ and $W$ is constant, this yields the weight tuning law.
Now,
\[
\dot{L} = - r^T K_o r + r^T (\epsilon + \tau_d) \leq - K_{v_{\text{min}}} |r| + (\epsilon_n + d_B) \|r\| \tag{58}
\]
with \(K_{\text{min}}\) the minimum singular value of \(K_v\). Since \(\epsilon_n + d_B\) is constant, \(\dot{L} \leq 0\) as long as
\[
|r| > (\epsilon_n + d_B) / K_{v_{\text{min}}} \equiv b_r. \tag{59}
\]

Selecting the gain according to (52) ensures that the compact set defined by \(|r| \leq b_r\) is contained in \(S_r\), so that the approximation property holds throughout. Therefore, the tracking error \(r(t)\) is bounded and continuity of all functions shows as well the boundedness of \(r(t)\).

It remains to show that \(\dot{W}\), or equivalently \(\dot{\bar{W}}\), is bounded. Boundedness of \(r(t)\) guarantees the boundedness of \(e(t)\) and \(\dot{e}(t)\), whence boundedness of the desired trajectory shows \(q\) and \(\dot{q}\) are bounded. Property 2 then shows boundedness of \(V_n(q, \dot{q})\). These facts guarantee boundedness of the function
\[
y \equiv Mr + (K_o + V_m)r - (\epsilon + \tau_d) \tag{60}
\]
since \(M(q)\) is bounded. Therefore, the dynamics relative to \(\bar{W}\) are given by
\[
\dot{\bar{W}} = - F \phi \dot{r}^T
\]
\[
y^T = \phi^T \bar{W} \tag{61}
\]
with \(y(t)\) and \(r(t)\) bounded. (The second equation is (51).)

Note that \(\bar{W}\) is a matrix. Using the Kronecker product \(\otimes\) allows one to write the vector dynamics
\[
\frac{d}{dt} \text{vec}(\bar{W}) = -(I \otimes F \phi) r
\]
\[
y = (I \otimes \phi^T) \text{vec}(\bar{W})
\]
where the \(\text{vec}(A)\) operator stacks the columns of a matrix \(A\) to form a vector, and one notes that \(\text{vec}(z^T) = z\) for a vector \(z\). Now, the PE condition on \(\phi\) is equivalent to PE of \((I \otimes \phi)\), and so to the uniform complete observability of this system, so that boundedness of \(y(t)\) and \(r(t)\) assures the boundedness of \(\bar{W}\), and hence of \(W\). (Note that boundedness of \(x(t)\) verifies boundedness of \(F\phi(x(t))\).)

This proof is very similar to adaptive control proofs since the FLNN is LIP. The major difference is the need to ensure that the NN approximation property holds on the region of attraction. This requires the use of the initial tracking error set \(S\), and leads to the PD gain condition (52). According to this condition, the required PD gains increase with \(\epsilon_n\) and the disturbances, and also as the desired trajectory becomes more active. They decrease as the NN size \(L\) increases since this decreases \(\epsilon_n\).

This theorem defines the notion of PE for a 1-layer NN: it amounts to a PE requirement on the hidden-layer outputs \(\phi(x)\).

The case where the NN estimation error bound \(\epsilon_n\) depends on \(x\) is covered in [35]. In this case the robustifying term \(\theta(t)\) in the control (50) is required.

The next result shows that if the NN can approximate perfectly, no PE condition is needed.

**Theorem 3.** (FLNN Controller in an Ideal Case). Let the hypotheses of Theorem 2 hold, with however the NN functional reconstruction error \(\epsilon_n\) and unmodeled disturbances \(\tau(t)\) equal to zero. Use the same controller as in Theorem 2. Then the tracking error \(r(t)\) goes to zero with \(t\) and the weight estimates \(\dot{W}\) are bounded. Neither PE of \(\phi(x)\) nor the gain condition (52) are required for this result.

The proof of this result relies on a standard adaptive control approach, using the same Lyapunov function as in Theorem 2 and then invoking Barbalat’s Lemma [3].

Since the 1-layer NN used here is LIP, the results and the proofs are familiar from adaptive control. However, the NN controller is far more powerful than standard robot adaptive controllers since the LIP is not in the system parameters, but in the NN weights. The NN has a universal approximation property so that the controller works for any rigid robot arm. No regression matrix is needed.

In practice it is not necessary to compute the constants \(c_0, c_1, c_2\) nor determine \(S_r\). The size of the NN \(L\) and the PD gains \(K_o\) are simply selected large and a simulation is performed. The simulation is repeated with a different \(L\) and \(K_o\). Based on the difference in behavior between the two simulations, \(L\) and \(K_o\) can be modified.

Some practical considerations of implementing FLNN NN controllers are given in [6,14]. The following additional properties of the NN controller are important.

**Unsupervised Backprop through Time Tuning.** Note that algorithm (53) is nothing but the continuous-time backpropagation algorithm (16) for the 1-layer case. However, it is an unsupervised version of backprop in that the ideal plant output is not needed; instead the filtered error \(\dot{r}(t)\), which is easily measurable in the closed-loop system, is used in tuning the weights. It should also be realized that this is a version of the backprop through time algorithm, as the weights are continuously tuned as a function of time \(t\).

**Weight Initialization and On-Line Tuning.** In the NN control schemes derived in this paper there is no preliminary off-line learning phase. The weights are simply initialized at zero, for then Fig. 5 shows that the controller is just a PD controller. Standard results in the robotics literature [10] show that a PD controller gives bounded
errors if \( K_p \) is large enough. Therefore, the closed-loop system remains stable until the NN begins to learn. The weights are tuned on-line in real-time as the system tracks the desired trajectory. As the NN learns \( f(x) \), the tracking performance improves. This is a significant improvement over other NN control techniques where one must find some initial stabilizing weights, generally an impossible feat for complex nonlinear systems.

Note that the fixed parameters in the first layer of the NN, that is the parameters of the activation functions, must be selected so that \( \phi(x) \) is a basis. This may be accomplished using either results such as Theorem 1 or by random choice as in RVFL.

**Bounds on the Tracking Error and NN Weight Estimation Errors.** In the ideal case of no NN approximation error \( \epsilon \) or unmodeled disturbances \( \tau(t) \), Theorem 3 shows that the tracking error \( r(t) \) vanishes with time. In the nonideal case of Theorem 2, a PE condition is needed, and then the tracking error does not vanish but is UUB. The right-hand side of (59) can be taken as a practical bound on the tracking error in the sense that \( r(t) \) will never stray far above it. It is important to note from this equation that the tracking error in the sense that the right-hand side of (59) can be taken as a practical bound on the tracking error may be achieved by selecting large gains \( K_v \). This is in spite of Barron’s lower bound on \( \epsilon \).

Note that the NN weights \( W \) are not guaranteed in Theorem 2 to approach the ideal unknown weights \( W \) that give good approximation of \( f(x) \). However, this is of no concern as long as \( W - \hat{W} \) is bounded, as the proof guarantees. This guarantees bounded control inputs \( \pi(t) \) so that the tracking objective can be achieved.

### 4.4.2 Modified unsupervised backpropagation through time tuning

In adaptive control the possible unboundedness of the weight (e.g. ‘parameter’) estimates when PE fails to hold is known as ‘parameter drift’. This phenomenon has been called ‘weight overtraining’ in the NN literature. The PE condition in Theorem 2 is meant to ensure that drift does not occur. PE is especially difficult to ensure in large unstructured NN, and is difficult to ensure even in RBF [13]. To correct the parameter drift problem without requiring the PE condition, one may modify the NN weight tuning algorithm using techniques from adaptive control, including \( \sigma \)-modification [18], \( \epsilon \)-modification [41], or dead-zone techniques [26]. Lifting of the PE condition results in a more robust NN controller that is stable under a wide variety of unmodeled dynamics and unforeseen situations.

The next theorem derives the tuning law for a FLNN controller that does not require PE. The tuning law is augmented by the \( \epsilon \)-mod term in [41].

It must now be assumed that the ideal weights \( W \) are constant and bounded so that

\[
\left| W \right|_F \leq W_b ,
\]

with the bound \( W_b \) known. In [43] it is shown how to use standard adaptive robust techniques to avoid the assumption that the bound \( W_b \) is known.

**Theorem 4.** (Augmented NN Weight Tuning Algorithm). Given the hypotheses of Theorem 2, assume the ideal NN target weights are bounded by \( W_b \) as in (62). Let the control input for the robot arm be given by (50) with \( u(t) = 0 \) and gain

\[
K_{v_{\min}}^{F} = \frac{(\kappa W_b / 4 + \epsilon_\nu + d_b)(c_0 + c_2)}{b_1 - q_b} .
\]

Let the weight tuning be modified as

\[
\dot{W} = F \phi r^T - \kappa F \left| W \right|_F ,
\]

with \( F = F^T > 0 \) and \( \kappa > 0 \) a small design parameter. Make no assumptions of any sort of PE requirements on \( \phi(x) \). Then the filtered tracking error \( r(t) \) and the NN weight estimates \( \dot{W} \) are UUB with practical bounds given respectively by the right-hand sides of (67), (68). Moreover, the tracking error may be made as small as desired by increasing the tracking gain \( K_v \).

**Proof.** Let the NN approximation property (19) hold for the function \( f(x) \) given in (39) with a given accuracy \( \epsilon \), for all \( x \) in the compact set \( S_x \equiv \{ x \mid x < h_1 \} \) with \( h_1 > q_b \). Let \( r(0) \in S_x \). Then the approximation property holds at time 0.

Select the Lyapunov function candidate (54) and obtain (56). Then, using tuning rule (64) yields

\[
L = -r^T K_{vF} r + \kappa \left| r \right|_{F} \left| tr\{ W^T (W - W) \} + r (r + \tau_d) \right|_{F} .
\]

Since

\[
tr\{ W^T (W - W) \} = \left| W \right|_F \left| W \right|_F - \left| W \right|_F^2 ,
\]

there results

\[
L \leq -K_{v_{\min}} \left| r \right|_{F}^2 + \kappa \left| r \right|_{F} \left| W \right|_F \left( W_b - W_b \right) + \left( \epsilon_\nu + d_b \right) \left| W \right|_F \\
= -\left| r \right|_{F} \left| K_{v_{\min}} \right|_{F} \left| r \right| + \kappa \left| W \right|_F \left| W - W_b \right| - \left( \epsilon_\nu + d_b \right) \left| W \right|_F ,
\]

(66)
which is negative as long as the term in braces is positive. Completing the square yields

\[ K_{\nu_{\min}}\|r\| + \kappa_0^2 \| \bar{W} \|_{F} (\bar{W} - W_{B}) - (\epsilon_x + d_{b}) \]

\[ = \kappa \| \bar{W} \|_{F} (W_{B} / 2)^{2} - \kappa W_{B}^{2} / 4 + K_{\nu_{\min}}\|r\| - (\epsilon_x + d_{b}) \]

which is guaranteed positive as long as

\[ \|r\| > \frac{\kappa W_{B}^{2} / 4 + (\epsilon_x + d_{b})}{K_{\nu_{\min}}} \equiv b_{r} \]

or

\[ \| \bar{W} \|_{F} > W_{B} / 2 + \sqrt{\kappa W_{B}^{2} / 4 + (\epsilon_x + d_{b}) / \kappa} \equiv b_{w} . \]

(67)

(68)

Thus, \( L \) is negative outside a compact set. Selecting the gain according to (63) ensures that the compact set defined by \( \|r\| \leq b_{r} \) is contained in \( S_{\alpha} \), so that the approximation property holds throughout. This demonstrates the UUB of both \( \|r\| \) and \( \| \bar{W} \|_{F} \).

This proof is similar to [41] but is modified to reflect the fact that the NN approximates only on a compact set. In this result one now requires the bound \( W_{B} \) on the ideal weights \( W \), which appears in the PD gain condition (63). Now, bounds are discovered both for the tracking error \( r(t) \) and the weights \( \bar{W}(t) \). PE is not needed to establish the bounds on \( W \) with the modified weight tuning algorithm.

The NN weight tuning algorithm (64) consists of a backprop through time term plus a new second term. The importance of the \( \kappa \) term, called the \( e \)-modification, is that it adds a quadratic term in \( \| \bar{W} \|_{F} \) in (66), so that it is possible to establish that \( L \) is negative outside a compact set in the \( \{ \|r\|, \| \bar{W} \|_{F} \} \) plane [41]. The \( e \)-mod term makes the tuning law \textit{robust} to unmodelled dynamics so that the PE condition is not needed. In [44] a projection algorithm is used to keep the NN weights bounded. In [4,5] a deadzone technique is employed.

4.5 Tuning algorithms for nonlinear-in-the-parameters NN

Now suppose that a 2-layer NN is used to approximate the robot function according to (5) with \( \|e\| \leq \epsilon_{x} \) on a compact set, and the ideal approximating weights \( W, V \) constant. Then the control law (41) becomes

\[ \tau = W^{T} \sigma(V^{T} x) + K_{f} \sigma - v . \]

(69)

The proposed NN control structure is shown in Fig. 5.

The NLIP NN controller is far more powerful than the 1-layer FLNN as it is not necessary to select a basis of activation functions. In effect, the weights \( V \) are adjusted on-line to automatically provide a suitable basis. The control is nonlinear in the first-layer weights \( V \), which presents complications in deriving suitable weight tuning laws. Auxiliary signal \( v(t) \) is a function to be detailed subsequently that provides robustness in the face of higher-order terms in the Taylor series arising from the NLIP problem.

Using the result (13) to properly address the NLIP problem, the closed-loop error dynamics (42) can be written as

\[ M \dot{r} = -(K_{\alpha} + V_{w})r + W^{T}(\dot{\sigma} - \dot{\sigma} \dot{V}^{T} x) + W^{T} \dot{\sigma} \dot{V}^{T} x + w + v \]

(70)

where the disturbance terms are

\[ w(t) = \dot{W}^{T} \sigma \dot{V}^{T} x + \dot{W}^{T} O(V^{T} x)^{2} + \epsilon + \tau_{\epsilon} . \]

(71)

Define

\[ Z = \begin{bmatrix} W & 0 \\ 0 & V \end{bmatrix} \]

(72)

and \( Z, \dot{Z} \) equivalently. It is not difficult to show that

\[ \|w(t)\| \leq C_{v} + C_{w} \|Z_{r}\| + C_{z} \|Z_{w}\| \]

(73)

with \( C_{v} \) known positive constants.

It is assumed that the ideal weights are bounded so that

\[ \|Z\|_{F} \leq Z_{B} \]

(74)

with \( Z_{B} \) known. (Standard adaptive robust control techniques can be used to lift the assumption that \( Z_{B} \) is known [43].)

4.5.1 Augmented backprop tuning for the NLIP case

The next theorem presents the most powerful NN controller in this section.

**Theorem 5.** (Augmented Backprop Weight Tuning for Multilayer NN Controller). Let the desired trajectory \( q_{d}(t) \) be bounded by \( q_{B} \) and the initial tracking error \( r(0) \) be in \( S_{\alpha} \). Let the ideal target NN weights be bounded by known \( Z_{B} \). Take the control input for the robot dynamics (35) as (69) with PD gain satisfying

\[ K_{\nu_{\min}} > \frac{(C_{B} + \kappa C_{v}^{2} / 4)(c_{a} + c_{e})}{b_{s} - q_{B}} \]

(75)
where \( C_3 \) is defined in the proof. Let the robustifying term be

\[
\nu(t) = -K_3 \left| Z \right|_F + Z_B r
\]

with gain

\[
K_3 > C_2.
\]

(77)

Let NN weight tuning be provided by

\[
\dot{W} = F \dot{\sigma} r^T - F \dot{\sigma} \dot{v} r^T - \kappa F r^T \dot{\bar{V}}
\]

(78)

\[
\dot{\bar{V}} = G \dot{\sigma}^T \dot{w} r - \kappa G r^T \dot{\bar{V}}
\]

(79)

with any constant matrices \( F = F^T > 0, G = G^T > 0, \) and \( \kappa > 0 \) a small scalar design parameter. Then the filtered tracking error \( r(t) \) and NN weight estimates \( \bar{V}, \bar{W} \) are UUB, with the bounds given specifically by (84) and (85). Moreover, the tracking error may be kept as small as desired by increasing the gains \( K_\omega \) in (69).

**Proof.** Let the NN approximation property (5) hold for the function \( f(x) \) given in (39) with a given accuracy \( e_n \) for all \( x \) in the compact set \( S_\nu = \{ x \mid \left| x \right| < b_1 \} \) with \( b_1 > q_B \). Let \( r(0) \in S_\nu \). Then the approximation property holds at time 0.

Define the Lyapunov function candidate

\[
L = \frac{1}{2} r^T M(q) r + \frac{1}{2} \text{tr} \{ \bar{W}^T F^{-1} \bar{W} \} + \frac{1}{2} \text{tr} \{ \bar{V}^T G^{-1} \bar{V} \}
\]

(80)

Differentiating yields

\[
\dot{L} = r^T M r + \frac{1}{2} r^T M r + \text{tr} \{ \bar{W}^T F^{-1} \bar{W} \} + \text{tr} \{ \bar{V}^T G^{-1} \bar{V} \} .
\]

(81)

Substituting now from the error system (70) yields

\[
\dot{L} = -r^T K_\omega r + \frac{1}{2} r^T (M - 2V_\omega) r
\]

\[
+ \text{tr} \{ \bar{W}^T (F^{-1} \bar{W} + \sigma r^T - \sigma \bar{V}^T x r^T) \}
\]

\[
+ \text{tr} \{ \bar{V}^T (G^{-1} \bar{V} + x r^T \bar{W}^T \sigma^T) \}
\]

(82)

The tuning rules give

\[
\dot{L} = -r^T K_\omega r + \kappa \left| r \right| \text{tr} \{ \bar{W}^T (W - \bar{W}) \}
\]

\[
+ \kappa \left| r \right| \text{tr} \{ \bar{V}^T (V - \bar{V}) \} + r^T (w + \nu)
\]

\[
= -r^T K_\omega r + \kappa \left| r \right| \text{tr} \{ \bar{Z}^T (Z - \bar{Z}) \} + r^T (w + \nu).
\]

Since

\[
\text{tr} \{ Z^T (Z - \bar{Z}) \} = < Z, Z > - \left| Z \right|_F^2 \leq \left| Z \right|_F \left| Z \right|_F - \left| Z \right|_F^2,
\]

there results

\[
L \leq -r^T K_\omega r + \kappa \left| r \right| \left| Z \right|_F (Z_B - \left| Z \right|_F)
\]

\[
- K_\omega \left| Z \right|_F + Z_B \left| r \right|^2 + \left| r \right| \left| w \right|
\]

\[
\leq -K_\omega \left| r \right| \left[ -K_\omega \left| Z \right|_F (Z_B - \left| Z \right|_F)
\]

\[
- K_\omega \left| Z \right|_F + Z_B \left| r \right|^2
\]

\[
+ \left| r \right| \left| C_0 + C_1 \right| \left| Z \right|_F + C_1 \left| r \right| \left| Z \right|_F^2
\]

\[
\leq -K_\omega \left| r \right| \left[ \kappa \left| Z \right|_F (Z_B - \left| Z \right|_F) - C_0 - C_1 \right] \left| Z \right|_F
\]

where \( K_\omega \) is the minimum singular value of \( K_\omega \), and the last inequality holds due to (77).

\( L \) is negative as long as the term in braces is positive. Defining \( C_3 = Z_B + C_1 / \kappa \) and completing the square yields

\[
K_\omega \left| r \right| - \kappa \left| Z \right|_F (Z_B - \left| Z \right|_F) - C_0 - C_1 \left| Z \right|_F
\]

\[
= \kappa \left| Z \right|_F (Z_B / 2)^2 + K_\omega \left| r \right| - C_0 - \kappa C_3^2 / 4
\]

(83)

which is guaranteed positive as long as either

\[
\left| r \right| > C_0 + \kappa C_3^2 / 4 \equiv b_r
\]

(84)

or

\[
\left| Z \right|_F > C_3 / 2 + \sqrt{C_0 / \kappa + C_3^2 / 4} \equiv b_Z.
\]

(85)

Thus, \( L \) is negative outside a compact set. According to the LaSalle extension [30] this demonstrates the UUB of both \( \left| r \right| \) and \( \left| Z \right|_F \), as long as the control remains valid within this set. However, the PD gain condition (75) shows that the compact set defined by \( \left| r \right| \leq b_r \) is contained in \( S_\nu \), so that the approximation property holds throughout.

The first terms in the tuning algorithms (78), (79) are nothing but backpropagation of the tracking error through time (c.f. (16)). The last terms are the Narendra \( e \text{-mod}, \) and the second term in (78) is a novel second-order term needed due to the NLIP of the NN. The robustifying term \( \nu(t) \) is required also due to the NLIP problem.

The comments appearing after Theorem 3 are germane here. Specifically, there is no preliminary off-line tuning phase and NN weight tuning occurs on-line in real-
time simultaneously with control action. Weight initialization is not an issue in the proof and it is not necessary to provide initial stabilizing weights; the weights are simply initialized so that the NN output is equal to zero, for then the PD loop keeps the system stable until the NN begins to learn. However, practical experiments [14] show that it is important to initialize the weights suitably. A good choice is to select $\tilde{V}(0)$ randomly and $\tilde{W}(0)$ equal to zero.

In practice it is not necessary to compute the constants $c_0, c_1, c_2, r_0, C, C_0, Z_B$ nor determine $S$. The size $L$ of the NN and the PD gains $K$ are simply selected large and a simulation is performed. The simulation is then repeated with a different $L$ and $K$. Based on the difference in behavior between the two simulations, $L$ and $K$ can be modified.

**Straight Backpropagation Weight Tuning**. The backprop algorithm (usually in discrete-time form) has been proposed innumerable times in the literature for closed-loop control. It can be shown that if the NN approximation error $e$ is zero, the disturbances $x(t)$ are zero, and there are no higher-order terms in the Taylor series (12), then the straight backprop algorithm (c.f. (16))

$$\dot{W} = F \tilde{\sigma} r^T$$

$$\dot{V} = Gx(\tilde{\sigma}^TWr r)^T$$

(86)

may be used for closed-loop control instead of (78), (79). PE is required to show that the NN weights are bounded using straight backprop. The conditions mentioned are very restrictive and do not occur in practice; they essentially require the robot arm to be linear (e.g. 1 = link). Moreover, PE conditions are not easy to guarantee in NN.

### 4.5.2 Simplified hebbian tuning

The next result shows how to use the simplified Hebbian NN weight tuning rule (17) in closed-loop control. In this derivation, the problem of nonlinearity in the tunable weights $V$ is overcome without using the Taylor series expansion in (12)—the proof relies on the error dynamics in the form

$$M\dot{r} = -(K_x + V) r + \tilde{W}^T \tilde{\sigma} + W^T \tilde{\sigma} + (\varepsilon + \tau_0) + \nu$$

(87)

**Theorem 6.** (Augmented Hebbian Weight Tuning). Let the desired trajectory $q_d(t)$ be bounded by $q_\beta$ and $r(0)$ be in $S$. Let the ideal target NN weights $Z$ be bounded by $Z_B$. Take the control input for the robot dynamics as (69) with PD gain satisfying

$$K_{v_{min}} > \frac{D(c_3 + c_2)}{b_\beta - q_\beta} ,$$

(88)

where

$$D = \kappa C \frac{1}{2} + \sqrt{L}Z_B + d_\beta + \varepsilon_N$$

with $L$ the number of hidden-layer neurons and

$$C = \frac{Z_B}{2} + \frac{1}{2} \kappa .$$

Let the robustifying term be

$$u(t) = -K_c(\|Z\|_F^2 + Z_B) r ,$$

(89)

with gain

$$K_c > c_2 .$$

(90)

Let NN weight tuning be provided by

$$\dot{W} = F \tilde{\sigma} r^T - \kappa F r |\tilde{W}|$$

(91)

$$\dot{V} = G |r| \tilde{\sigma} - \kappa G r |V|$$

(92)

with any constant matrices $F = F^T > 0$, $G = G^T > 0$, and $\kappa > 0$ a small scalar design parameter. Then the filtered tracking error $r(t)$ and NN weight estimates $\tilde{V}, \tilde{W}$ are UUB, with practical bounds given by

$$|r| > \frac{D}{K_{v_{min}}} = b_r ,$$

(93)

$$|Z| > C_1 + \sqrt{\frac{D \kappa}{K}} = b_z .$$

(94)

Moreover, the tracking error may be kept as small as desired by increasing the gains $K_c$.

**Proof.** See [31].

See the remarks at the end of Theorem 5. Note that the tuning rules are of Hebbian form (17), with each layer of weights tuned using the outer product of its input signal and its output signal. To prove convergence, the standard Hebbian rules must be modified by adding the robustifying $e$-mod terms, and also by multiplying the first term in (92) by $|r|$. The backprop-based tuning algorithm in Theorem 5 generally gives better performance than the Hebbian rule since it uses more information, including the Jacobians.

### 4.6 Passivity properties of NN controllers

The NN used in this paper are static feedforward nets, but since they appear in a closed feedback loop and are tuned using differential equations, they are dynamical systems. Therefore, one can discuss the **passivity** of these
NN. In general a NN cannot be guaranteed to be passive. However, the NN controllers in this paper have some important passivity properties that result in robust closed-loop performance. Passivity notions for NN are defined herein. The idea of SSP (see (34) and following comments) is pivotal in this development.

4.6.1 Passivity of the robot tracking error dynamics

The error dynamics in this paper (e.g. (70)) have the form

$$M \dot{r} = -(K_v + V_m)r + \xi_0$$

where \(r(t)\) is the filtered tracking error and \(\xi_0(t)\) is appropriately defined. This system satisfies the following strong passivity property.

**Theorem 7.** (SSP of Robot Error Dynamics) The dynamics (95) from \(\xi_0(t)\) to \(r(t)\) are a state strict passive system.

**Proof.** Take the nonnegative function

$$L = \frac{1}{2} r^T M(q)r$$

so that, using (95), one obtains

$$L = r^T Mr + \frac{1}{2} r^T M r = - r^T K_v r + \frac{1}{2} r^T (M - 2V_m)r + r^T \xi_0$$

whence the skew-symmetry property yields the power form

$$L = r^T \xi_0 - r^T K_v r .$$

This is the power delivered to the system minus a quadratic term in \(|r|\), verifying state strict passivity.

4.6.2 Passivity properties of 2-layer NN controllers

The closed-loop system is in Fig. 5, where the NN appears in an inner feedback linearization loop. The error dynamics for the 2-layer NN controller are given by (70). The closed-loop error system appears in Fig. 6, with the signal \(\xi_1\) defined as

$$\xi_1(t) = - \hat{W}^T (\hat{\sigma} - \hat{\sigma}' V^T x)$$

(96)

Note the role of the NN in the error dynamics, where it is decomposed into two effective blocks appearing in a typical feedback configuration.

We now reveal the passivity properties engendered by straight backpropagation tuning (86). To prove this algorithm, one uses the error dynamics in a different form than (70), so that in Fig 6 one has \(\dot{\xi}_1(t) = - \hat{W}^T \hat{\sigma}\).

**Theorem 8.** (Passivity of Backprop NN Tuning Algorithm). The simple backprop weight tuning algorithm (86) makes the map from \(r(t)\) to \(- \hat{W}^T \hat{\sigma}\) and the map from \(r(t)\) to \(- \hat{W}^T \hat{\sigma}' V^T x\), both passive maps.

**Proof.** The dynamics with respect to \(\hat{W}, \hat{V}\) are

$$\dot{\hat{W}} = - F \hat{\sigma} r^T$$

$$\dot{\hat{V}} = - G x (\hat{\sigma}^T \hat{W} r)^T$$

(97)

(98)

1. Selecting the nonnegative function

$$L = \frac{1}{2} tr \{ \hat{W}^T F^{-1} \hat{W} \}$$

and evaluating \(L\) along the trajectories of (97) yields

$$\dot{L} = tr \{ \hat{W}^T F^{-1} \hat{W} \} = - tr \{ \hat{W}^T \hat{\sigma} r^T \} = r^T (- \hat{W}^T \hat{\sigma}) ,$$

which is in power form (32).

2. Selecting the nonnegative function

$$L = \frac{1}{2} tr \{ \hat{V}^T G^{-1} \hat{V} \}$$

and evaluating \(L\) along the trajectories of (98) yields

$$\dot{L} = tr \{ \hat{V}^T G^{-1} \hat{V} \} = - tr \{ \hat{V}^T x (\hat{\sigma}^T \hat{W} r)^T \} = r^T (- \hat{W}^T \hat{\sigma}' V^T x),$$

which is in power form.

Thus, the robot error system in Fig. 6 is state strict passive (SSP) and the weight error blocks are passive; this guarantees the passivity of the closed-loop system. Using the passivity theorem [50] one may now conclude that the input/output signals of each block are bounded as long as the external inputs are bounded.

Unfortunately, though passive, the closed-loop system cannot be guaranteed to be SSP, so that when distur-
bances are nonzero, this does not yield boundedness of the internal states of the weight blocks (i.e., $\hat{W}$, $\hat{V}$) unless those blocks are observable, that is persistently exciting (PE).

The next result shows why a PE condition is not needed with the modified weight update algorithm given in Theorem 5.

**Theorem 9.** (SSP of Augmented Backprop NN Tuning Algorithm). The modified weight tuning algorithms in Theorem 5 make the map from $r(t)$ to $-\hat{W}'(\bar{\sigma} - \bar{\sigma}'\hat{V}^T x)$, and the map from $r(t)$ to $-\hat{W}'(\bar{\sigma} - \bar{\sigma}'\hat{V}^T x)$, both state strict passive (SSP) maps.

**Proof.** The dynamics relative to $\hat{W}$, $\hat{V}$ using the tuning algorithms in Theorem 5 are given by

$$\dot{\hat{W}} = -F \hat{V} + F \hat{V} x r + \kappa F \hat{W}$$

$$\dot{\hat{V}} = -G x (\hat{w} T \hat{w}) + \kappa G \hat{V}.$$ 

1. Selecting the nonnegative function

$$L = \frac{1}{2} tr(\hat{W}^T F^{-1} \hat{W})$$

and evaluating $L$ yields

$$L = tr(\hat{W}^T F^{-1} \hat{W}) = tr([- \hat{W}^T (\bar{\sigma} - \bar{\sigma}' \hat{V}^T x)] r^T$$

$$+ \kappa \hat{V} r \hat{W})$$

Since

$$tr(\hat{W}^T (W - \hat{W}) = \langle \hat{W} W, r \rangle - \left\| \hat{W} \right\|_r^2$$

$$\leq \left\| \hat{W} \right\|_r \cdot \left\| W \right\|_r - \left\| \hat{W} \right\|_r^2,$$

there results

$$L \leq r T [- \hat{W}^T (\bar{\sigma} - \bar{\sigma}' \hat{V}^T x)] - \kappa \hat{V} r \cdot (\left\| \hat{W} \right\|_r - \left\| \hat{W} \right\|_r W),$$

which is in power form with the last function quadratic in $\left\| \hat{W} \right\|_r$.

2. Selecting the nonnegative function

$$L = \frac{1}{2} tr(\hat{V}^T G^{-1} \hat{V})$$

and evaluating $L$ yields

$$L = tr(\hat{V}^T G^{-1} \hat{V}) = r^T (- \hat{W}^T \bar{\sigma}' \hat{V}^T x)$$

$$- \kappa \left\| \hat{V} \right\|_r^2 - < \hat{V}, V >_r$$

$$\leq r^T (- \hat{W}^T \bar{\sigma}' \hat{V}^T x) - \kappa \left\| \hat{V} \right\|_r^2 - V_B \hat{V}$$

which is in power form with the last function quadratic in $\left\| \hat{V} \right\|_r$.

The SSP of both the robot dynamics and the weight tuning blocks does guarantee SSP of the closed-loop system, so that the norms of the internal states are bounded in terms of the power delivered to each block. Then, boundedness of input/output signals assures state boundedness without any sort of observability or PE requirement.

**Definition of Passive NN and Robust NN.** We define a dynamically tuned NN as passive if, in the error formulation, the tuning guarantees the passivity of the weight tuning subsystems. Then, an extra PE condition is needed to guarantee boundedness of the weights. This PE condition is generally in terms of the outputs of the hidden layers of the NN (see Theorem 2). We define a dynamically tuned NN as robust if, in the error formulation, the tuning guarantees the SSP of the weight tuning subsystem. Then, no extra PE condition is needed for boundedness of the weights. Note that (1) SSP of the open-loop plant error system is needed in addition for tracking stability, and (2) the NN passivity properties are dependent on the weight tuning algorithm used.

### 4.6.3 Passivity properties of 1-layer NN controllers

In a similar fashion, it is shown that the FLNN controller tuning algorithm in Theorem 2 makes the system passive, so that an additional PE condition is needed to verify internal stability of the NN weights. On the other hand, the augmented tuning algorithm in Theorem 4 yields SSP, so that no PE is needed.

### 4.7 Partitioned neural networks and preprocessing of NN inputs

To improve performance the NN can be partitioned into simplified subnets, and the input $x(t)$ can be preprocessed. This simplifies the design, gives added controller structure, allows more accurate function approximation, and makes for faster weight tuning algorithms.

**Partitioned Neural Nets.** The unknown nonlinear robot function (39) is

$$f(x) = M(q) \zeta_3 (i) + V_m(q, \dot{q}) \zeta_2 (i) + G(q) + F(q)$$ (99)
where $\xi_i(t) \equiv \dot{q}_i = \Lambda \dot{e}, \quad \zeta_i(t) \equiv \dot{q}_i + \Lambda \dot{e}$. Taking the four terms in $f(x)$ one at a time, use separate NN to reconstruct each term so that

$$M(q)\xi_i(t) = W_M^T \sigma_M(V_{m}^T x_M) + \epsilon_M,$$

$$V_a(q, \dot{q}) \zeta_i(t) = W_v^T \sigma_v(V_{v}^T x_v) + \epsilon_v,$$

$$G(q) = W_c^T \sigma_c(V_{c}^T c) + \epsilon_c,$$

$$F(q) = W_f^T \sigma_f(V_{f}^T f) + \epsilon_f. \quad (100)$$

This procedure results in four neural subnets, one for estimating the inertia terms, one for the coriolis/centripetal terms, one for gravity, and one for friction. This is called a structured or partitioned NN, as shown in Fig. 7. It is direct to show that the individual partitioned NNs can be separately tuned using the algorithms given in the Theorems, 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 compute them. NNs can be used to reconstruct only the unknown terms or those too complicated to compute, which will probably include the friction $F(q)$ and the coriolis/centripetal terms $V_a(q, \dot{q})$.

**Preprocessing of Neural Net Inputs.** In selecting suitable NN inputs $x(t)$ for computation, some preprocessing of signals yields a more advantageous choice than (40) since it can explicitly introduce some of the nonlinearities inherent to robot arm dynamics. This reduces the burden of expectation on the NN and also reduces the reconstruction error $\epsilon$.

Let an $n$-link robot have $n_r$ revolute joints with joint variables $q_r$, and $n_p$ prismatic joints with joint variables $\theta_p$ so that $n = n_r + n_p$. Since the only occurrences of the revolute joint variables are as sines and cosines, transform $q = [q_r^T, q_p^T]^T$ by preprocessing to $[\cos(q_r)^T \sin(q_r)^T q_p^T]^T$ to be used as arguments for the basis functions. Then the NN input vector $x$ can be taken as

$$x = [\xi_1^T \xi_2^T \cos(q_r)^T \sin(q_r)^T q_p^T \dot{q}_r \text{sgn}(q)^T]^T, \quad (101)$$

where the signum function is needed in the friction terms.

**V. INNER FEEDBACK LOOPS, APPLICATIONS, AND EXTENSIONS**

A NN controller for rigid-link robot manipulators is given in Fig. 5. Continuous-time NN weight tuning algorithms were given that guarantee closed-loop stability. However, for actual implementation, discrete-time NN weight tuning algorithms are more convenient. Moreover, 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 specified forces or torques as well as perform position trajectory following (e.g. robotic grinding or milling). In such cases, the NN controller in Fig. 5 still works if it is modified to include additional inner feedback loops to deal with the additional plant or performance complexities. Here we give without proofs some extensions of the NN controllers that deal with these issues. For more details see [31,32] and other listed references.

### 5.1 Discrete-time tuning for NN feedback control

Since most controllers requiring the computation of nonlinear terms are implemented using digital signal processors or microprocessors, it is important to design NN controllers with discrete-time weight update algorithms, where the weights may be tuned only at the sample times. Proposed discrete-time NN tuning algorithms for feedback control abound in the literature, but until recently were ad hoc modifications of open-loop gradient-based algorithms such as the delta rule, and could not guarantee any sort of stability or tracking in closed-loop feedback controls applications.

Using rigorous nonlinear stability methods based on Lyapunov techniques, exactly as in deriving the continuous-time NN controllers, it is possible though much more involved to derive digital NN controllers [19]. The structure of the digital NN controller is the same as Fig. 5. Tuning algorithms for the 2-layer NN are given by

$$W_{k+1} = W_k + \alpha \delta_{k} \dot{\theta}_{k} + \kappa \left[ I - \alpha \delta_{k} \dot{\theta}_{k} \right] W_{k} \quad (102)$$

![Fig. 7. Partitioned neural net, which has more structure and is faster to tune than unpartitioned neural network.](image-url)
\[ V_{k+1} = V_k - \alpha_k x_k [V_k^T x_k + K_{\nu} x_k]^T - \kappa [I - \alpha_k x_k x_k^T] V_k , \]  

(103)

where \( \alpha_k = \sigma [V_k^T x_k] \), \( \alpha_k = -\frac{\xi_1}{\zeta_1 + \| x_k \|} \), \( \alpha_k = -\frac{\xi_2}{\zeta_1 + \| x_k \|} \) and 

\[ 0 < \kappa < 0, \rho < \xi_1 < 2, 0 < \xi_2 < 1, \kappa < \xi_2 . \]

These tuning algorithms which hold for NLIP nets are familiar from discrete-time adaptive control and STR [1,12], and are a form of delta rule with the first terms very similar to a discrete-time Hebbian rule with some extra terms involving the tracking error \( r_k \). The last terms are similar to what have been called ‘forgetting factors’ in computer science and are equivalent to a discrete-time version [20] of Narendra’s \( e \)-modification. These terms are required to make the NN controller robust to unknown unmodelled dynamics by ensuring that the NN weights remain bounded. To speed up learning for NN with a large number \( L \) of hidden-layer neurons, one has defined the \( \alpha_k \) based on a projection algorithm well known in adaptive control [12,1].

The proof of this algorithm is complex and relies on a discrete-time Lyapunov function that includes both the tracking error and the weight estimation errors. Consequently no ‘certainty equivalence’ condition is needed. Nor is PE needed with these tuning algorithms due to the last terms.

5.2 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 this case, the NN force/position controller in Fig. 8 can be derived [27]. The structure of the NN force controller is the same as the multiloop NN controller in Fig. 5, with however the addition of an inner loop for force control. This multiloop intelligent control topology appears to be very versatile and powerful.

The equations describing the NN force/position controller are as follows. The control input is

\[ \tau = W^T \sigma (V^T x) + K_{\nu} (Lr) - J^T (\lambda_r - K_f \lambda_r) - \nu , \]  

(104)

where the robustifying signal is

\[ \nu(t) = -K_c (\| Z \| + Z_{\nu}) r . \]  

(105)

The NN Weight/Threshold Tuning Algorithms are

\[ \dot{W} = F \sigma (V^T x) (Lr)^T - F \tilde{\sigma} \dot{V} x (Lr)^T - \kappa F \| (Lr) \| W , \]  

(106)

\[ \dot{V} = G x (\sigma^T \tilde{W} (Lr))^T - \kappa G \| (Lr) \| V \]  

(107)

where \( F, G \) are positive definite matrices and \( \kappa > 0 \).

This controller has guaranteed performance in that both the position tracking error \( r(t) \) and the force error \( \lambda(t) \) are kept small while all the NN weights are kept bounded.

The selection matrix \( L \) and jacobian \( J \) are computed based on the decomposition of the joint variable \( q(t) \) into two components— the component \( q_t \) (e.g. tangential to the given surface) in which position tracking is desired and the component \( q_n \) (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. The filtered position tracking error in \( q_t(t) \) is \( r(t) \), that is, \( r = q_{td} - q_t \), with \( q_{td} \) the desired trajectory in the plane of the surface. The desired force is described by \( \lambda_n \) and the force exertion error is captured in \( \lambda = \lambda - \lambda_n \), with \( \lambda_n \) describing the actual measured force exerted by the manipulator. The position tracking gain is \( K_v \) and the force tracking gain is \( K_f \).

5.3 NN controller for electrically-driven robot using backstepping

Robot manipulators are driven by actuators which may be electric, hydraulic, pneumatic, and so on. The actuators are coupled to the links through coupling mechanisms which may contain gears. Particularly in the case of high-speed performance requirements, the coupling shafts may exhibit appreciable compliance that cannot be disregarded. Many real-world systems in industrial and military applications also have flexible modes and vibratory effects. In all these situations, the NN controller in Fig. 5 must be modified. Two design techniques that are particularly useful for this purpose are singular perturbations and backstepping [23,24,31].

A typical example of a real robotic system is the robot arm with electric actuators, or rigid-link electrically-driven (RLED) manipulator. The dynamics of a \( n \)-link rigid robot arm with motor electrical dynamics are given by

\[ M(q) \ddot{q} + V_{\nu}(q, \dot{q}) \dot{q} + F(q) + G(q) + \tau_d = K_{\nu} \dot{q} , \]  

(108)

\[ Li + R(i, \dot{q}) + \tau_e = u_e , \]  

(109)
with \( q(t) \in \mathbb{R}^q \) the joint variable, \( i(t) \in \mathbb{R}^q \) the motor armature currents, \( K^T \) a diagonal electro-mechanical conversion matrix, \( L \) a matrix of electrical inductances, \( R(i, q) \) representing both electrical resistance and back emf, \( \tau_d(t) \) and \( \tau_e(t) \) the mechanical and electrical disturbances, and motor terminal voltage vector \( u_c(t) \in \mathbb{R}^n \) the control input.

This plant has unknown dynamics in both the robot subsystem and the motor subsystem. The NN tracking controller in Fig. 9 was designed using the backstepping technique [28]. 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. The NN weight tuning algorithms are similar to the ones presented in Theorem 5 but with some extra terms. 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 motor dynamics. This multiloop controller is typical of NN control systems designed using rigorous system theoretic techniques.

### 5.4 Output feedback control using dynamic neural networks

If all the states of the controlled plant are available as measurements, then the static NN controllers presented heretofore can be used. (It is noted that, though the NN are static in themselves, the closing of a feedback loop around them turns them into dynamic NN in conjunction with the plant dynamics.) In the case of output feedback one must use an additional NN with its own internal dynamics in the controller [22]. The function of the NN dynamics is effectively to provide estimates of the unmeasurable plant states, so that the dynamic NN functions as an observer.

Taking the representative Lagrangian mechanical dynamics (35), let there be available now only measurements of the joint variable vector \( q(t) \in \mathbb{R}^q \). Specifically, the joint velocities \( \dot{q}(t) \) are not measured. This is a typical situation in actual industrial applications, where optical encoders are used to measure \( q(t) \).

It can be shown that the following dynamic NN observer can provide estimates of the entire state \( x = [x_1^T \ x_2^T]^T \equiv [\dot{q}^T \ \dot{q}^T]^T \) given measurements of only \( x_1(t) = \dot{q}(t) \). This observer has a more or less standard structure known in the field of nonlinear system observers, but a NN is used to estimate a certain nonlinear function, which somewhat changes the structure and complicates the proofs.

\[
\dot{x}_1 = \ddot{x}_1 + k_p x_1, \\
\dot{x}_2 = M^{-1}(x_1)[\tau - W_o^T \sigma_o(x) + k_m x_1 + u_e] \\
\dot{x}_2 = \dot{x}_2 + k_{p_2} x_1, \\
\]

In this system, \( \hat{\cdot} \) denotes estimates, and \( \tilde{\cdot} \) denotes estimation errors, e.g., \( \hat{x}_1 = x_1 - \ddot{x}_1, \hat{x}_2 = x_2 - \dot{x}_2 \). It is assumed that the inertia matrix \( M(q) \) is known, but all other nonlinearities are estimated by the observer NN \( W^o_o \sigma_o(x) \), which has output-layer weights \( W^o_o \) and activation functions \( \sigma_o(\cdot) \). This system is a dynamic NN of a special structure since it has its own dynamics in the integrators corresponding to the estimates \( \ddot{x}_1, \dot{x}_2 \). Signal \( u_e(t) \) is an observer robustifying term, and the observer gains \( k_p, k_m, k_{p_2} \) are positive design constants.

The NN output-feedback tracking controller shown in Fig. 10 uses the dynamic NN observer to reconstruct the missing measurements \( x_1(t) = \dot{q}(t) \), and then employs a second 1-layer static NN for tracking control, as in Fig. 5. Since neither the joint velocities \( x_2(t) = \ddot{q}(t) \) nor the tracking error \( r(t) \) is directly measurable, the control input (50) must be modified so it becomes

\[
\tau = W_o^T \sigma_o(x) + K_{\tau} r + \Lambda e - u_e \\
\]

where the estimated or measurable portion of the tracking error is

\[
r = (q_d - \ddot{x}_2) + \Lambda e = r + \ddot{x}_2, \\
\]

![Fig. 9. Multiloop NN backstepping controller, showing inner backstepping loop with a second NN.](image)

![Fig. 10. Dynamic NN tracking controller with reduced measurements, showing second dynamic NN loop required for state estimation.](image)
with \( e(t) = q(t) - x(t) \) the tracking error. The control NN has weights \( W_c \) and activation functions \( \sigma(\cdot) \), and \( u(t) \) is a control robustifying signal. Note that the outer tracking PD loop structure has been retained.

In this dynamic NN controller, two NN must be tuned. Note that this formulation shows both the observer NN and the control NN as 1-layer FLNN; therefore, both \( \sigma(\cdot) \) and \( \sigma(t) \) must be selected as bases. A more complex derivation shows that both can in fact be taken as 2-layer NN. It can be shown [22] that both the static control NN weights \( W_c \) and the dynamic observer NN weights \( W_o \) should be tuned using variants of the algorithm presented in Theorem 4.

This design is in keeping with the finding that if the plant has additional complications or uncertainties, more hierarchical structure must be added to the control system.

VI. CONCLUSION

The Nonlinear Network structures discussed here include both neural networks (NN) and fuzzy logic systems. Nonlinear network controller design algorithms were given for a general class of industrial Lagrangian motion systems characterized by the rigid robot arms. The design procedures are based on rigorous nonlinear derivations and stability proofs, and yield a multiloop intelligent control structure with NN in some of the loops. NN weight tuning algorithms were given that do not require complicated initialization procedures or any off-line learning phase, work on-line in real-time, and offer guaranteed tracking and bounded NN weights and control signals. The NN controllers given here are model-free controllers in that they work for any system in a prescribed class without the need for extensive modeling and preliminary analysis to find a ‘regression matrix’. Unlike standard robot adaptive controllers, they do not require linearity in the parameters (see also [7] which does not need LIP). Proper design allows the NN controllers to avoid requirements such as persistence of excitation and certainty equivalence. NN passivity and robustness properties are defined and studied here.

As the uncertainty in the controlled plant increases or the performance requirements become more complex, it is necessary to modify the NN controller by adding additional feedback loops. A force controller and a backstepping controller for a combined electro-mechanical system were given that needed additional inner control loops. A dynamic NN controller for output feedback was discussed.

REFERENCES


**F. L. Lewis** was born in Würzburg, Germany, subsequently studying in Chile and Gourounstoun School in Scotland. He obtained the Bachelor’s Degree in Physics/Electrical Engineering and the Master’s of Electrical Engineering Degree at Rice University in 1971. He spent six years in the U.S. navy, serving as Navigator aboard the frigate USS Trippe (FF-1075), and Executive Officer and Acting Commanding Officer aboard USS Salinan (ATF-161). In 1977, he received the Master’s of Science in Aeronautical Engineering from the University of West Florida. In 1981 he obtained the Ph.D. degree at The Georgia Institute of Technology in Atlanta, where he was employed as a professor from 1981 to 1990 and is currently an Adjunct Professor. He is a Professor of Electrical Engineering at The University of Texas at Arlington, where he was awarded the Moncrief-O’Donnell Endowed Chair in 1990 at the Automation and Robotics Research Institute.

Dr. Lewis has studied the geometric, analytic, and structural properties of dynamical systems and feedback control automation. His current interests include robotics, intelligent control, neural and fuzzy systems, nonlinear systems, and manufacturing process control. He is the author/co-author of 2 U.S. patents, 124 journal papers, 20 chapters and encyclopedia articles, 210 refereed conference papers, seven books: Optimal Control, Optimal Estimation, Applied Optimal Control and Estimation, Aircraft Control and Simulation, Control of Robot Manipulators, Neural Network Control, High-Level Feedback Control with Neural Networks and the IEEE reprint volume Robot Control. Dr. Lewis is a registered Professional Engineer in the State of Texas and was selected to the Editorial Boards of International Journal of Control, Neural Computing and Applications, and Int. J. Intelligent Control Systems. He is the recipient of an NSF Research Initiation Grant and has been continuously funded by NSF since 1982. Since 1991 he has received $1.8 million in funding from NSF and upwards of $1 million in SBIR/industry/state funding. He received a Fulbright Research Award, the American Society of Engineering Education F. E. Terman Award, three Sigma Xi Research Awards, the UTA Halliburton Engineering Research Award, the UTA University-Wide Distinguished Research Award, the ARRI Patent Award, various Best Paper Awards, the IEEE Control Systems Society Best Chapter Award (as Founding Chairman), and the National Sigma Xi Award for Outstanding Chapter (as president). He was selected as Engineer of the year in 1994 by the Ft. Worth IEEE Section and is a fellow of the IEEE. He was appointed to the NAE Committee on Space Station in 1995 and to the IEEE Control Systems Society Board of Governors in 1996. In 1998 he was selected as an IEEE Control Systems Society Distinguished Lecturer. He is a Founding Member of the Board of Governors of the Mediterranean Control Association.