NSL-910303, March 1991
Also appears: Proc. ACC, June 1991

Direct Adaptive Control Using Gaussian Networks

Robert M. Sanner and Jean-Jacques E. Slotine
Nonlinear Systems Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139

Abstract

A direct adaptive tracking control architecture is proposed and evaluated for a class of continuous-time nonlinear dynamic systems for which an explicit linear parameterization of the uncertainty in the dynamics is either unknown or impossible. The architecture employs a network of gaussian radial basis functions to adaptively compensate for the plant nonlinearities. Under mild assumptions about the degree of smoothness exhibited by the nonlinear functions, the algorithm is proven to be stable, with tracking errors converging to a neighborhood of zero.

A constructive procedure is detailed, which directly translates the assumed smoothness properties of the nonlinearities involved into a specification of the network required to represent the plant to a chosen degree of accuracy. A stable weight adjustment mechanism is then determined using Lyapunov theory. The network construction and performance of the resulting controller are illustrated through simulations with an example system.

1 Introduction

The current interest in neural networks, and their ability to construct complex, nonlinear mappings based upon examples, has produced a variety of algorithms for incorporating these models into nonlinear control systems [18, 20, 23, 24, 27, 28, 34]. Most of these designs use off-line, iterative techniques to adjust the variable parameters of the network so as to develop discrete-time models of the direct or inverse plant dynamics. Gradient techniques are usually employed for the parameter adjustment mechanisms. The resulting trained network is then incorporated as the basis of a nonlinear control law.

These methods of exploiting neural networks for control have several drawbacks. While exact representations of continuous linear systems can be obtained by properly discretizing the plant dynamics, the same is not true for discretizations of continuous, nonlinear physical systems. Further, as a practical matter, neural networks will be most useful for solving real-time control problems if they are implemented in hardware, instead of modeled in software by discrete approximations. These facts suggest examining neural models for control which are amenable to a direct, physical implementation, for example, as the output of the continuous, analog electronic networks suggested by researchers in the field of early

vision [14, 32]. Our goal is hence to develop adaptive architectures capable of exploiting continuous analog networks for the control of continuous-time, nonlinear dynamic systems. Our approach is to treat the entire problem in the context of adaptive systems theory, avoiding iterative training procedures in favor of provably stable techniques for tuning the network parameters.

While gradient methods for adjustment of the parameters have proven to be quite effective in practice, problems arise in attempting to prove the stability of these methods when the adaptation is carried out on-line. These problems come from the fact that the network itself is parameterized nonlinearly, and current adaptive systems theory is well developed only for linearly parameterized models.

There are three critical questions which must be answered in attempting to design neural networks for direct adaptive control: do there exist tuned values of the network parameters which will fulfill the control objectives (this is sometimes called the *matching condition*); how large does the network have to be, in terms of the numbers of neurons and interconnecting weights required, and how does this scale with the assumed structure of the control objectives; and how can the network weights be adjusted toward their tuned settings in a fashion which guarantees the boundedness of all time-varying signals in the system.

In this paper, we attempt to systematically answer these questions by exploiting recent insights in both neural network and adaptive systems theory. Section 2 describes the setting of the control problem to be examined and illustrates the structure of conventional adaptive methods for its solution. Section 3.1 reviews the use of regularization theory as a method of translating assumed smoothness properties of the plant nonlinearities into a representation capable of uniformly approximating the plant over a compact set; the conditions under which these representations can be discretized and mapped onto a neural network with a finite number of components is discussed. Section 4 illustrates how these networks may be used as elements of an adaptive tracking control algorithm, and draws upon recent work with adaptive integral transforms to prove the stability and tracking convergence of the resulting closed loop system. Section 5 illustrates the method with an example.

2 Problem Statement

Assume that the plant dynamics are given in canonical form by:

$$x^{(n)}(t) + f(x(t), \dot{x}(t), \ldots, x^{(n-1)}(t)) = bu(t) \tag{2.1}$$

where $u(t)$ is the control input, f is an unknown linear or nonlinear function, and for simplicity in the initial discussion, take $b = 1$; this constraint will be removed in the subsequent development. The control objective is to force the plant state vector, $\mathbf{x} = [x, \dot{x}, \ldots, x^{(n-1)}]^T$, to follow a specified desired trajectory, $\mathbf{x}_d = [x_d, \dot{x}_d, \ldots, x_d^{(n-1)}]^T$. Defining the tracking error vector, $\tilde{\mathbf{x}}(t) = \mathbf{x}(t) - \mathbf{x}_d(t)$, the problem is thus to design a control law $u(t)$ which ensures that $\tilde{\mathbf{x}}(t) \to 0$ as $t \to \infty$.

Assume that $u(t) = u_{pd}(t) + u_{ad}(t) + r(t)$ where u_{pd} is a "PD" type control component consisting of a linear combination of tracking error states, $u_{ad}(t)$ is an adaptive control law which will attempt to recover, and cancel, the unknown function $f(\mathbf{x})$, and $r(t)$ is a feedforward of the nth derivative of the desired trajectory, that is $r(t) = x_d^{(n)}$. We then have:

$$\tilde{x}^{(n)}(t) = -\mathbf{k}^T \tilde{\mathbf{x}}(t) + (u_{ad}(t) - f(\mathbf{x}(t)))$$

where $\mathbf{k}$ is a vector of gains chosen to correspond to the coefficients of a Hurwitz polynomial. The complete closed loop dynamic equation is thus:

$$\dot{\tilde{\mathbf{x}}}(t) = \Lambda\tilde{\mathbf{x}}(t) + \mathbf{b}(u_{ad}(t) - f(\mathbf{x}(t))) \tag{2.2}$$

where Λ is a matrix in control canonical form, with eigenvalues at the roots of the Hurwitz polynomial which generates $\mathbf{k}$, and $\mathbf{b} = [0, 0, \ldots, 1]^T$.

If it is known *a priori* that $f(\mathbf{x})$ admits an expansion in terms of a set of known (linear or nonlinear) *basis functions*, $Y_i(\mathbf{x})$, i.e.:

$$f(\mathbf{x}) = \sum_{i=1}^{N} a_i Y_i(\mathbf{x})$$

then the adaptive controller can be chosen as:

$$u_{ad}(t) = \sum_{i=1}^{N} \hat{a}_i(t) Y_i(\mathbf{x}(t))$$

where $\hat{a}_i(t)$ is an approximation to the ith coefficient in the expansion of f [37, 7]. Hence,

$$u_{ad}(t) - f(\mathbf{x}(t)) = \sum_{i=1}^{N} \tilde{a}_i(t) Y_i(\mathbf{x}(t))$$

where $\tilde{a}_i(t) = \hat{a}_i(t) - a_i$. The tracking problem can thus be solved if a law for stably adjusting $\hat{a}_i(t)$ can be created which guarantees $\tilde{\mathbf{x}}(t) \to 0$.

The design of such a law is possible if a measure of the tracking error can be found which is sufficiently correlated with the approximation errors $\tilde{a}_i(t)$. Given the structure of the error dynamics (2.2), any linear combination of the error states:

$$s(t) = \boldsymbol{\lambda}^T \tilde{\mathbf{x}}(t)$$

satisfies this correlation condition provided that the transfer function $\boldsymbol{\lambda}^T(pI - \Lambda)^{-1}\mathbf{b}$ is *strictly positive real* [26, 38], where p is the Laplace variable.

These facts form the basis for most of the stable adaptive tracking controller designs. The drawback is the relatively large amount of prior information which must be incorporated into these designs in the form of the exact structure of the required basis functions.

3 Representation Issues

3.1 Regularization

When exact basis functions are not known, there are iterative methods by which the unknown function may be approximated. Memory based methods and their variants, [6, 24], partition the state space $\mathcal{R}^n$ into small hypercubes and attempt to iteratively estimate the values assumed in each hypercube by f or, more generally, when the plant dynamics are not as in (2.1), the values required for $u_{ad}(\mathbf{x})$ [3, 4, 17]. Potential function methods [2, 23] similarly partition $\mathcal{R}^n$ but provide an explicit means of "spreading" the information learned on a given partition to neighboring points. Neural networks offer yet a third approach, attempting to reconstruct f from compositions and superpositions of very simple nonlinearities [11, 13, 9].

All of these methods implicitly assume a certain smoothness of the unknown function (memory based methods usually smooth the piecewise continuous approximations with some form of local interpolation), but represent essentially heuristic approaches to the modeling task. Developing these techniques formally requires a precise description of the assumed smoothness properties, for example, requiring a number of the partial derivatives of the function to be bounded in absolute value. A useful alternative description involves creating a strictly positive measure of the smoothness, such as (if $\mathbf{x} \in \mathcal{R}^2$):

$$r_f(\mathbf{x}) = (a_{00}(\mathbf{x})f(\mathbf{x}))^2 + (a_{01}(\mathbf{x})\frac{\partial f}{\partial x_1}(\mathbf{x}))^2 + (a_{10}(\mathbf{x})\frac{\partial f}{\partial x_2}(\mathbf{x}))^2 + \cdots + (a_{mn}(\mathbf{x})\frac{\partial^{m+n} f}{\partial x_1^m \partial x_2^n}(\mathbf{x}))^2$$

(i.e. a weighted linear combination of the squares of the function and its partial derivatives), which the required function, $f(\mathbf{x})$, is assumed to minimize.

This description is central to regularized approximation theory [40, 41, 31]. Given (direct or indirect) information about the correct values for f at a one-dimensional subset of points in $\mathcal{R}^n$ parameterized by t, $\{\mathbf{x}(t) \mid 0 \leq t \leq T\}$, let $e_z(t)$ be a measure of the error obtained by using the values of an arbitrary function $z(\mathbf{x}(t))$ to try to match the given information about f. Regularization attempts to find the function $z(\mathbf{x})$ which both satisfies the assumed smoothness constraints on f and matches the known information as closely as possible. This can formally be accomplished by expressing z as the function which minimizes

$$J[z] = \frac{1}{2}\left[\int_0^T e_z(t)^2 dt + \alpha \int_{D_f} r_z(\mathbf{x}) d\mathbf{x}\right], \tag{3.1}$$

where, D_f is the domain on which f (and hence z) is defined, and α is a free parameter which controls the tradeoff between the assumed smoothness properties and the accuracy of the approximation.

Note that if $a_{00}(\mathbf{x})$ is a nonzero constant and $D_f = \mathcal{R}^n$, then f is assumed to be an element of the Hilbert space $L^2(\mathcal{R}^n)$. By expressing the smoothness constraints, $r_z(\mathbf{x})$, in terms of a linear differential operator, P, on $L^2(\mathcal{R}^n)$, such that $(Pz)^2(\mathbf{x}) = r_z(\mathbf{x})$, the structure of this space can be conveniently exploited, producing the necessary conditions for z to be a minimizing function:

$$\alpha \hat{P} P z + e_z(\xi)\delta(\xi - \mathbf{x}(t)) = 0$$

where $\hat{P}$ is the Hilbert-adjoint of the operator P [21, 19] and $\delta(\xi)$ is the Dirac distribution.

Poggio [31, 32] has studied representations arising from this equation in detail, and shows that the required solution can be represented in terms of a superposition integral using the *kernel* or *Green's function*, K, of the self-adjoint operator $\hat{P}P$ [39, 10, 25]. In the current context, this method suggests:

$$z(\mathbf{x}) = \int_{\mathbf{Y}} K(\mathbf{x}, \xi) c(\xi) d\xi \tag{3.2}$$

where the integral is to be carried out over the parameterized 1-manifold Y corresponding to the trajectory $\mathbf{x}(t)$, and the kernel is a solution (in the distributional sense) of the partial differential equation:

$$\hat{P} P z(\mathbf{x}) = \delta(\mathbf{x}).$$

The function $c(\xi)$ is a weighting function which depends upon both the values $\{e_z(\xi) \mid \xi \in Y\}$ and the structure of the kernel. The regularized solution can thus be envisioned as a continuous superposition of potential functions centered along Y; physically, this is analogous

to approximating $f(\mathbf{x})$ as the electric field strength induced at $\mathbf{x}$ from a charged filament Y with a charge density $c(\xi)$. Of course, the actual physical interpretation depends upon the structure of K and hence upon P.

If the known information about f is exact, and f does in fact minimize the smoothness properties specified by P, then representation (3.2) can be exact on Y; however, this representation poses a problem if it is to be used as the basis of an adaptive control algorithm, where the values assumed by the function f, and hence the information contained in $e(t)$, depend upon the states of the plant. In this setting, one cannot know *a priori* the manifold Y, since this will depend upon the exact trajectory followed by the plant state vector during operation of the controller; hence, it is impossible to know where to center the kernels. To address this difficulty, extend the superposition in (3.2) to a compact set Γ, containing Y, such that every point within Γ may potentially contribute to a representation of f which is valid on this set:

$$f(\mathbf{x}) = \int_{\mathbf{\Gamma}} K(\mathbf{x}, \xi) c(\xi) d\xi \tag{3.3}$$

3.2 Discretization

Using (3.3) as a representation for f requires determination of the entire influence function $c(\xi)$, and hence an uncountable number of values. In order for (3.3) to provide a useful basis for approximating f, it is necessary to investigate the conditions under which the integral can be broken into a *finite* number of evaluations.

Assume that this representation of f is sufficiently well-behaved that it admits a multidimensional spatial Fourier transform [5, 1] (a mild assumption given the compactness of Γ). If in addition this transform is bounded and vanishes outside a compact set in Ω^n, f admits an *exact* expansion in terms of its samples on an appropriately defined lattice in $\mathcal{R}^n$ [29]. To see this, define a sequence of basis vectors for the lattice as $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$, and choose the lattice points to be:

$$\xi_I = i_1 \Delta_1 \mathbf{v}_1 + i_2 \Delta_2 \mathbf{v}_2 + \cdots + i_n \Delta_n \mathbf{v}_n$$

where $I = \{i_1, \ldots, i_n\}$ is an n-tuple of integers from $(-\infty, \infty)$, and the Δ_i are the sample spacings along the lattice basis vectors. For simplicity, we will consider uniform sampling on a rectangular lattice, hence $\{\mathbf{v}_i\}_{i=1}^n = \{\mathbf{e}_i\}_{i=1}^n$, where the $\mathbf{e}_i$ are the standard basis vectors for $\mathcal{R}^n$, and $\Delta_1 = \cdots = \Delta_n = \Delta$.

The sampling operation can be viewed as a modulation of the hypersurface, f, with a field of Dirac distributions centered at the lattice points, i.e.

$$f_s(\mathbf{x}) = \psi(\mathbf{x}) f(\mathbf{x}), \tag{3.4}$$

where

$$\psi(\mathbf{x}) = \sum_{I \in \mathcal{I}} \delta(\mathbf{x} - \xi_I)$$

and $\mathcal{I} = \mathcal{Z}^n$ where $\mathcal{Z}$ is the set of all integers. Fourier transforming (3.4), one obtains:

$$F_s(\nu) = F(\nu) * \Psi(\nu)$$

where $*$ indicates convolution and,

$$\Psi(\nu) = \frac{1}{\Delta^n} \sum_{I \in \mathcal{I}} \delta(\nu - \nu_I).$$

Hence

$$F_s(\nu) = \frac{1}{\Delta^n} \sum_{I \in \mathcal{I}} F(\nu - \nu_I), \tag{3.5}$$

implying that the spectrum of the sampled waveform consists of copies of the original spectrum centered at the lattice points in Ω^n, $\nu_I = \Delta^{-2}\xi_I$.

Let S_f= Support(f), B be the smallest n-cube, centered at the origin, which completely encloses S_f, and β = Radius(B). If Δ is chosen to be less than or equal to $1/(2\beta)$, the copies of $F(\nu)$ in the sampled spectrum will be non-overlapping, and the canonical reconstruction function, whose spectrum is Δ^n on B and zero elsewhere, can be used to assemble $f(\mathbf{x})$ on the basis of its samples, since, under these conditions:

$$F(\nu) = F_s(\nu)\, G_c(\nu) \tag{3.6}$$

where $G_c(\nu)$ is the spectrum of the canonical reconstructor. Hence, inverse transforming:

$$f(\mathbf{x}) = \sum_{I \in \mathcal{I}} f(\xi_I)\, g_c(\mathbf{x} - \xi_I) \tag{3.7}$$

The structure of equality (3.6), however, suggests that many other functions can potentially be used for the reconstruction. Notice that if the hypersurface is oversampled, i.e. if $\Delta < 1/(2\beta)$, then, defining $C_\Delta = [-(2\Delta)^{-1}, (2\Delta)^{-1}]^n \supset B$, the spectrum of the reconstructor is arbitrary on $C_\Delta - B$. In fact, as long as its spectrum is bounded, real valued and strictly positive on B, and vanishes outside of C_Δ, an arbitrary (Fourier transformable) function can be used in place of g_c in expansion (3.7), although the coefficients will no longer be samples of f itself.

To see this, assume that $g_1(\mathbf{x})$ is such a function, and using its spectrum, define a new function $c(\mathbf{x}) = \mathcal{F}^{-1}(C(\nu))$, where:

$$C(\nu) = \Delta^n F(\nu) G_1^{-1}(\nu) \quad \text{if } \nu \in B$$

and $C(\nu) = 0$ otherwise. Since, by assumption $G_1(\nu)$ is strictly positive and real on B, this new spectrum is well-defined; further, $C(\nu)$ is bounded since both $F(\nu)$ and $G_1^{-1}(\nu)$ are on B. Hence $C(\nu)$ is absolutely integrable, and its inverse transform $c(\mathbf{x})$ is bounded.

Consider applying the above sampling algorithm to $c(\mathbf{x})$. It is then straightforward to verify from the definitions of $C(\nu)$ and $G_1(\nu)$ that:

$$F(\nu) = C_s(\nu)\, G_1(\nu) \tag{3.8}$$

where $C_s(\nu)$ is the sampled spectrum of $c(\mathbf{x})$, defined as in (3.5). Inverse transforming this equation reveals an exact expansion of $f(\mathbf{x})$ in terms of the samples of the bounded function $c(\mathbf{x})$:

$$\begin{aligned} f(\mathbf{x}) &= \mathcal{F}^{-1}\left[C_s(\nu) G_1(\nu)\right] \\ &= c_s(\mathbf{x}) * g_1(\mathbf{x}) = (c(\mathbf{x})\psi(\mathbf{x})) * g_1(\mathbf{x}) \end{aligned}$$

thus,

$$f(\mathbf{x}) = \sum_{I \in \mathcal{I}} c(\xi_I) g_1(\mathbf{x} - \xi_I). \tag{3.9}$$

Often in practice the spectrum of f approaches zero only asymptotically, and the same could be true of the reconstructing function. Under these conditions, expansion (3.9) cannot

be exact; however, if the rate of decrease of $|F(\nu)|$ is sufficiently fast as $\|\nu\| \to \infty$, then choosing a sufficiently small sampling mesh on $\mathcal{R}^n$ permits reduction of the aliasing to an arbitrarily small level, and results in an expansion, call it $f_e(\mathbf{x})$, which can uniformly approximate the actual function to a chosen accuracy on a given compact set, i.e. $|f - f_e| \leq \epsilon/2$.

Since the partial sums of (3.9) converge, for each $\mathbf{x}$ one can ignore all the terms in the summation corresponding to samples ξ_I which lie outside an n-ball of radius ρ surrounding $\mathbf{x}$, with the resulting truncation error less than $\epsilon/2$. To achieve this level of reconstruction everywhere on a particular, compact set, A, take the union of all such n-balls centered at each point $\mathbf{x} \in A$. This union defines a new set Γ (with deliberate similarity to (3.3)). Note that the number of mesh points contained within Γ is finite.

Let $f_a(\mathbf{x})$ be the truncated series obtained by restricting the summation in (3.9) to Γ, i.e.:

$$f_a(\mathbf{x}) = \sum_{I \in \mathcal{I}_o} c(\xi_I) g_1(\mathbf{x} - \xi_I) \tag{3.10}$$

where the index set $\mathcal{I}_o = \{I \mid \xi_I \in \Gamma\}$. It follows that, for all $\mathbf{x} \in A$:

$$\begin{aligned} |f(\mathbf{x}) - f_a(\mathbf{x})| &\leq |f(\mathbf{x}) - f_e(\mathbf{x})| + |f_e(\mathbf{x}) - f_a(\mathbf{x})| \\ &\leq \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon \end{aligned} \tag{3.11}$$

Thus, a function observing the above smoothness constraints can be approximated to a prespecified accuracy everywhere on a compact subset of $\mathcal{R}^n$ using only a finite number of components. If the kernel satisfies the conditions discussed above for a reconstructing function, (3.10) can be used as the basis for producing a finite discretization of the integral representation (3.3).

4 Adaptive Control with Gaussian Networks

If f has a transform with compact support, then f is infinitely differentiable [5] and judicious choice of the coefficients for the expansion of the operator P in the regularization functional produces radial gaussian kernels in representation (3.3) of f [32]:

$$K(\mathbf{x}, \xi) = g_\sigma(||\mathbf{x} - \xi||^2) = \exp(-\frac{(\mathbf{x} - \xi)^T(\mathbf{x} - \xi)}{\sigma^2}) \tag{4.1}$$

This kernel is particularly attractive for many reasons: it is bounded, absolutely integrable on $\mathcal{R}^n$, and it is its own Fourier transform (modulo a scale factor). Further, Girosi and Poggio [12] have proven that any continuous function, not necessarily infinitely smooth, can be uniformly approximated by linear combinations of gaussians. This agrees with the above observation regarding approximations of functions whose spectra vanish only asymptotically.

The gaussian thus fulfills the criteria discussed in Section 3.2, and hence can be used as the reconstructing function in expansion (3.10). We conclude that if f has a bounded, (essentially) bandlimited, spatial Fourier transform, it can be uniformly approximated it on a chosen compact set using a finite number of evaluations of the radial gaussian:

$$f_a(\mathbf{x}) = \sum_{I \in \mathcal{I}_o} c_I \, g_\sigma(||\mathbf{x} - \xi_I||^2) \tag{4.2}$$

where $c_I = c(\xi_I)$.

As Poggio [32, 33] points out, the resulting expansion maps directly onto a class of three layer neural networks employing a single layer of "neurons" (or nodes) with radial gaussian input-output characteristics. The current point, $\mathbf{x}$, at which the approximation $f_a(\mathbf{x})$ is required, forms the input to the network. There are a number of nodes in the network equal to the number of elements in the set $\mathcal{I}_o$, and the weights connecting the network inputs to each gaussian node encode the points ξ_I in the sampling grid. Each node computes a quadratic "activation energy" from its inputs and feed-in weights, given by:

$$r_I^2 = \|\mathbf{x} - \xi_I\|^2 = (\mathbf{x} - \xi_I)^T(\mathbf{x} - \xi_I)$$

and outputs the gaussian of this activation, $g_\sigma(r_I^2)$. The network output is formed by weighting each nodal output by the corresponding c_I, and summing together all the weighted outputs. The result represents the approximation $f_a(\mathbf{x})$.

The variance and sampling density are chosen from the relation between the spectrum of the radial gaussian, $G_\sigma(\nu)$, and the assumed spectrum of f. For convenience in the design, the required variance of the gaussian can be computed from the representation of $G_\sigma(\nu)$:

$$G_\sigma(\nu) = \exp(-\frac{\pi \nu^T \nu}{\sigma_\nu^2})$$

and by inverse transforming one obtains [8]:

$$g_\sigma(r_I^2) = k \exp(-\pi \sigma_\nu^2 r_I^2)$$

which is (4.1) with $\sigma^2 = (\pi \sigma_\nu^2)^{-1}$. The constant k is a scale factor which, without loss of generality, can be absorbed into the coefficients c_I in (4.2).

The sampling density and variance are chosen such that:

$$\exp(-\frac{\pi}{4\Delta^2 \sigma_\nu^2}) \approx 0 \quad \text{and} \quad \exp(-\frac{\pi \beta^2}{\sigma_\nu^2}) > \gamma$$

where γ is any positive number in $(0, 1)$. These conditions ensure that the gaussian's spectrum is strictly positive on S_f, yet approaches zero quickly enough that the high frequency components in $F_s(\nu)$ contribute negligibly in the inverse transform. As a practical matter, the following combination has been found to work well:

$$\sigma_v^2 = \beta^2 \pi \quad \text{and} \quad \Delta^2 = \frac{1}{8\sigma_v^2}. \tag{4.3}$$

4.1 Controller Structure

Expansion (4.2) provides a network structure which can approximate a (sufficiently smooth) unknown plant nonlinearity with ϵ accuracy on a particular compact set. If the desired trajectories are contained in a compact subset, A_d, of the state space, in principle the tracking problem could be solved by reconstructing the unknown function f everywhere on A_d. However, it is possible that, in the initial stages of adaptation, the tracking errors may be sufficiently large that the state would leave this set. As will be shown below, with a minimum of additional prior knowledge one can construct a control law for (2.1) which guarantees that, if the desired trajectories are contained in A_d, the actual plant state will

never leave an n-cube, A, surrounding A_d. Expansion (4.2) can then be used as the basis of an adaptive tracking control architecture for trajectories confined to A_d.

The control law consists of four components:

$$u(t) = u_{pd}(t) + u_{ad}(t) + u_{aux}(t) + r(t) \tag{4.4}$$

where $r(t)$ is feedforward of the nth derivative of the reference trajectory, and $u_{pd}(t)$ is a constant feedback gain component which stabilizes the error dynamics (2.2). The term $u_{aux}(t)$ is an auxiliary controller which ensures that the plant state remains within A. This control component takes the form of a modulated sliding controller; the action of the sliding term is sufficient to enforce a contraction of the tracking error as the state approaches the boundary of A, hence keeping the plant state inside this set, while the modulation restricts activity of this component to $A - A_d$, rendering it inactive when $\mathbf{x}(t) \in A_d$. As will be shown in Section 4.2.2, it is possible to design this component so that it does not interfere with the adaptation mechanism.

The adaptive control law, $u_{ad}(t)$, is implemented as the output of the radial gaussian network described above, whose inputs are the measured values of the plant states. The variance and mesh size of the network is designed considering assumed upper bounds on the spectral properties of $f(\mathbf{x})$. Hence:

$$u_{ad}(t) = \hat{f}_a(\mathbf{x}(t)) = \sum_{I \in \mathcal{I}_o} \hat{c}_I(t)\, g_\sigma(||\mathbf{x}(t) - \xi_I||^2) \tag{4.5}$$

The input weights, ξ_I, which encode the sampling mesh, are *fixed* in this architecture, while the output weights, $\hat{c}_I(t)$, will be adjusted to attempt to match their tuned values, c_I. Section 4.2.3 derives an adjustment mechanism which guarantees the stability of the entire system.

4.2 Controller Design

4.2.1 PD Control Law

A convenient method for shaping the "PD" controller involves defining a time-varying *sliding surface* $s(t) = 0$ where

$$s(t) = (\frac{d}{dt} + \lambda)^{n-1} \tilde{x}(t) \quad \text{with} \quad \lambda > 0$$

which can be rewritten as $s(t) = \boldsymbol{\lambda}^T \tilde{\mathbf{x}}(t)$ with $\boldsymbol{\lambda}^T = [\lambda^{n-1}, (n-1)\lambda^{n-2}, \ldots, 1]$. This equation defines a time-varying hyperplane in $\mathcal{R}^n$ on which the tracking error vector decays exponentially to zero, so that perfect tracking can be obtained by maintaining $s(t) = 0$ [42, 36]. To this end, the "PD" law is chosen as:

$$u_{pd}(t) = -k_d s(t) - \boldsymbol{\lambda}_v^T \tilde{\mathbf{x}}(t) \tag{4.6}$$

where $k_d > 0$, and $\boldsymbol{\lambda}_v^T = [0, \lambda^{n-1}, (n-1)\lambda^{n-2}, \ldots, \lambda]$. The first term drives the error towards the instantaneous location of the sliding surface, while the second anticipates the future motion of the surface.

It is straightforward to verify that the above control law results in a stable, overdamped error system (2.2); Λ has one eigenvalue at $-k_d$, and (n-1) eigenvalues at $-\lambda$. In addition, the transfer function of the error system from $u_{ad}(t)$ to $s(t)$ is strictly positive real, suggesting that $s(t)$ can be used to derive stable adaptive laws for the feed-out weights of the neural network; a fact which is proven below.

4.2.2 Auxiliary Control Law

The validity of the approximation (3.11) for the tuned values of the network parameters depends upon the assumption that the plant state never leaves the compact set A on which f is approximated. If this assumption is violated, it will not be possible to guarantee convergence of the tracking error. The purpose of the auxiliary component of the control law is to ensure that the magnitude of the current plant state vector remains always less than the radius of the set A. Ensuring these bounds will require both an appropriately designed control law, $u_{aux}(t)$, and some constraints on the desired trajectories which can be commanded for a given A.

To see this, let r_a and r_d be the radii of A and A_d respectively. Since $\|\mathbf{x}\| \leq \|\tilde{\mathbf{x}}\| + \|\mathbf{x}_d\|$, in order to ensure that $\mathbf{x}$ remains confined to A it is necessary (but *not* sufficient, as shown below) that the auxiliary controller enforce $\frac{d}{dt}(\tilde{\mathbf{x}}^T\tilde{\mathbf{x}}) < 0$ as $\|\tilde{\mathbf{x}}\| \to (r_a - \|\mathbf{x}_d\|)$.

Since the error system is strictly positive real, the Kalman-Yakubovitch lemma (see, e.g. [38]) provides a symmetric, positive definite matrix P such that

$$P\Lambda + \Lambda^T P = -Q$$

and $Pb = \boldsymbol{\lambda}$. Defining a Lyapunov function candidate,

$$V_0 = \tilde{\mathbf{x}}^T P \tilde{\mathbf{x}}$$

and evaluating its time derivative along the state trajectories of the error system, one obtains:

$$\dot{V}_0 = -\tilde{\mathbf{x}}^T Q \tilde{\mathbf{x}} + 2s(\hat{f}_a - f + u_{aux}). \tag{4.7}$$

Since f is assumed continuous, let M_0 be any upper bound for $|f|$ on the compact set A, and let $M(t) = M_0 + |\hat{f}_a(t)|$. Choice of the switching control law $u_{aux}(t) = -M(t)\mathrm{sgn}(s(t))$, would clearly produce $\dot{V}_0 \leq 0$, however this would introduce discontinuous control action. It is possible to approximate this switched control law to any desired degree of accuracy using a smoothly saturating function, such as the sigmoid:

$$u_{aux}(t) = -M(t)\,[-1 + 2\,\mathrm{sig}(k_1 s(t))] \tag{4.8}$$

where $\mathrm{sig}(k_1 s) = (1 + e^{-k_1 s})^{-1}$. It is easy to confirm that the bracketed quantity approaches the sign function as $k_1 \to \infty$. For any fixed value of k_1, use of (4.8) in (4.7) yields:

$$\dot{V}_0 \leq -\|\tilde{\mathbf{x}}\|(\underline{\lambda}_Q\|\tilde{\mathbf{x}}\| - \epsilon_1(k_1))$$

where $\underline{\lambda}_Q$ is the smallest eigenvalue of Q and $\epsilon_1(k_1)$ is a monotone decreasing function of its argument such that $\epsilon_1(k_1) \to 0$ as $k_1 \to \infty$. Hence, the auxiliary controller is capable of reducing the norm of the tracking error whenever $\underline{\lambda}_Q\|\tilde{\mathbf{x}}\| > \epsilon_1(k_1)$, and k_1 can be chosen to make $\epsilon_1(k_1)$ as small as desired. In particular, if $r_a - r_d > \epsilon_1/\underline{\lambda}_Q$, the auxiliary controller will ensure that the state remains within A.

The action of the auxiliary controller is required only when there is a danger the the plant state might leave the set A; it should be inactive at other times, particularly when the plant state is confined to A_d (which will occur if the adaptive component can ensure $\tilde{\mathbf{x}}(t) \to 0$ as $t \to \infty$). In order to decide when u_{aux} should be active, define $d_0(t)$ as an upper bound on the instantaneous distance between the norm of the plant state and the boundary of A:

$$d_0(t) = \inf_{\tilde{\mathbf{x}}(t)} (r_a - \|\tilde{\mathbf{x}}(t)\| - \|\mathbf{x}_d(t)\|)$$

Since $||\tilde{\mathbf{x}}(t)||$ is upper bounded by $(V_0/\underline{\lambda}_P)^{\frac{1}{2}}$, where $\underline{\lambda}_P$ is the smallest eigenvalue of P this becomes:

$$d_0(t) = r_a - \left(\frac{V_0}{\underline{\lambda}_P}\right)^{\frac{1}{2}} - \|\mathbf{x}_d(t)\| \tag{4.9}$$

The activity of the auxiliary controller is then dictated by $d_0(t)$. Specifically, $u_{aux}(t)$ must ensure

$$\dot{d}_0(t) \geq 0 \quad \text{as} \quad d_0(t) \to 0 \tag{4.10}$$

Noting that:

$$\dot{d}_0 = -\frac{1}{2}\left(\frac{V_0}{\underline{\lambda}_P}\right)^{-\frac{1}{2}} \dot{V}_0 - \frac{d}{dt}\|\mathbf{x}_d(t)\| \tag{4.11}$$

it is clear that ensuring (4.10) requires not only the ability to force contraction of $\|\tilde{\mathbf{x}}(t)\|$, but also requires constraints on the rate of growth of $||\mathbf{x}_d(t)||$. Evaluating the first term in (4.11) at $d_0 = 0$, and using some straightforward re-arrangements, these constraints can be expressed as:

$$(\frac{d}{dt} + \eta)||\mathbf{x}_d|| \leq k r_a$$

where η is a positive constant depending only upon the relative magnitudes of the eigenvalues of P and Q. This condition can be interpreted as a constraint on the vector field describing the evolution of $\mathbf{x}_d$, expressing the intuitive idea that the desired trajectories must turn smoothly away from the boundary of A_d. It is easily ensured by generating the desired trajectories as the output of a stable, linear time-invariant system with a sufficiently small bandwidth. Of course, these constraints can be made arbitrarily loose by increasing r_a or decreasing r_d.

For continuity, it is desirable to introduce the actions of the auxiliary controller smoothly as $d_0 \to 0$. To this end, define a thin boundary layer of width ϕ around the inside of the n-ball with radius r_a, and modulate the u_{aux} above with the output of another sigmoid function:

$$u_{aux}(t) = -M(t)\,[-1 + 2\,\mathrm{sig}(k_1 s(t))]\,\mathrm{sig}(k_2(\frac{\phi}{2} - d_0(t))) \tag{4.12}$$

The second sigmoid gain, k_2, and the boundary layer width should be chosen so that:

$$\mathrm{sig}(\frac{k_2\phi}{2}) \approx 1 \quad \text{and} \quad \mathrm{sig}(k_2(\frac{\phi}{2} + r_d - r_a)) \approx 0$$

This makes the auxiliary controller completely active when $d_0 = 0$, (by symmetry, this also ensures the inactivity of the auxiliary controller outside the boundary layer, i.e. when $d_0 \geq \phi$), while guaranteeing that the auxiliary controller is never active when it is certain that $\mathbf{x} \in A_d$.

4.2.3 Adaptive Control Law

With the $u_{aux}(t)$ designed above, $\mathbf{x}(t)$ is guaranteed to remain within A and hence for the tuned values of adjustable network parameters, approximation (3.11) is valid. If this approximation error is viewed as a uniformly bounded disturbance driving the plant dynamics, it is possible to design an update law for the parameters which will guarantee the stability of all time varying signals in the system, and also ensure convergence of the tracking error to a ball whose radius depends continuously upon the approximation error ϵ.

Horowitz and Messner [15, 16, 22] have studied a class of adaptive control architectures based upon integral transforms similar to (3.3). Insight from both this research, and Peterson and Narendra's [30] research into adaptive control in the presence of bounded disturbances, suggests the Lyapunov function candidate:

$$V_1 = \frac{1}{2}\max\left(D, \tilde{\mathbf{x}}^T P \tilde{\mathbf{x}}\right) + \frac{1}{2k_a}\sum_{I\in\mathcal{I}_o}\tilde{c}_I^2$$

where $\tilde{c}_I(t) = \hat{c}_I(t) - c_I$, $k_a > 0$ is an adaptation gain, and the constant D defines the size of a deadzone in the adaptation law, given by

$$D = \left(\frac{4\overline{\lambda}_P}{\underline{\lambda}_Q^2}\right)\epsilon^2 \boldsymbol{\lambda}^T\boldsymbol{\lambda}$$

where $\overline{\lambda}_P$ is the largest eigenvalue of P, and $\underline{\lambda}_Q$ is the smallest eigenvalue of Q. This deadzone reflects the fact that no useful information can be gained about the quality of the current approximation $\hat{f}_a$ when the tracking errors are less than a threshold determined by the best case approximation error, ϵ, to the nonlinear function, f.

Note that V_1 is continuous, and lower bounded by a positive definite function of the states of the adaptive system:

$$V_1 \geq \underline{\lambda}_P\|\tilde{\mathbf{x}}\|^2 + \frac{1}{2k_a}\sum_{I\in\mathcal{I}_o}\tilde{c}_I^2$$

where $\underline{\lambda}_P$ is the minimum eigenvalue of P. Computing the time derivative of V_1 along state trajectories of the adaptive system, during times when $\tilde{\mathbf{x}}^T P\tilde{\mathbf{x}} > D$:

$$\begin{aligned}\dot{V}_1 &= -\frac{1}{2}\tilde{\mathbf{x}}^T Q\tilde{\mathbf{x}} + \tilde{\mathbf{x}}^T\boldsymbol{\lambda}(\hat{f}_a - f + u_{aux}) + k_a^{-1}\sum_{I\in\mathcal{I}_o}\tilde{c}_I\dot{\hat{c}}_I \qquad (4.13)\\ &= -\frac{1}{2}\tilde{\mathbf{x}}^T Q\tilde{\mathbf{x}} + \tilde{\mathbf{x}}^T\boldsymbol{\lambda}(f_a - f) + s\,u_{aux} + s\sum_{I\in\mathcal{I}_o}\tilde{c}_I g_\sigma(||\mathbf{x}-\xi_I||^2) + k_a^{-1}\sum_{I\in\mathcal{I}_o}\tilde{c}_I\dot{\hat{c}}_I\end{aligned}$$

Note that $|f - f_a| < \epsilon$ by construction if $\mathbf{x}(t) \in A$, and that $u_{aux}(t)$ is designed to enforce this inclusion. It is straightforward from (4.12) to verify that $s\,u_{aux} \leq 0$ for all $t \geq 0$, indicating that the actions of the auxiliary controller do not disrupt the stability of the adaptive process. Using these facts, if the adaptation law is chosen to be:

$$\dot{\hat{c}}_I = -k_a s\, g_\sigma(||\mathbf{x}-\xi_I||^2) \qquad (\text{when} \quad \tilde{\mathbf{x}}^T P\tilde{\mathbf{x}} > D) \qquad (4.14)$$

equation (4.13) becomes:

$$\dot{V}_1 \leq -\frac{1}{2}\|\tilde{\mathbf{x}}\|(\underline{\lambda}_Q\|\tilde{\mathbf{x}}\| - 2\epsilon\,\|\boldsymbol{\lambda}\|).$$

Since

$$\tilde{\mathbf{x}}^T P\tilde{\mathbf{x}} > D \quad\Rightarrow\quad \|\tilde{\mathbf{x}}\| > \left(\frac{D}{\overline{\lambda}_P}\right)^{\frac{1}{2}} = \frac{2\epsilon\,\|\boldsymbol{\lambda}\|}{\underline{\lambda}_Q}$$

V_1 is assured decreasing whenever $\tilde{\mathbf{x}}^T P\tilde{\mathbf{x}} > D$.

When $\tilde{\mathbf{x}}^T P \tilde{\mathbf{x}} \leq D$, choosing the adaptation law

$$\dot{\hat{c}}_I = 0 \qquad (\text{when} \quad \tilde{\mathbf{x}}^T P \tilde{\mathbf{x}} \leq D) \tag{4.15}$$

produces $\dot{V}_1 = 0$. Hence $\dot{V}_1 \leq 0$ for all time, and all states in the adaptive system remain bounded. Following the argument in [30], since V_1 is decreasing whenever the tracking error is outside the deadzone, after some finite time, T, the tracking errors will be confined to the neighborhood of zero defined by the deadzone, for all $t > T$. By the definition of D this neighborhood can be made arbitrarily small as $\epsilon \to 0$; in other words, the best case tracking performance is directly controlled by the magnitude of the uniform bound on the accuracy of the tuned approximation to f implemented by the gaussian network. Reducing this bound improves the guaranteed steady state tracking ability of the controller, with perfect tracking possible if the approximation can be made exact on A.

4.3 Non-Unity Control Gains

This design procedure can easily be extended to plants with non-unity control gains, for example:

$$x^{(n)}(t) + f(\mathbf{x}(t)) = g(x(t), \dot{x}(t), \ldots, x^{(n-2)}(t))u(t).$$

In this case, the unknown function which must be approximated by the network will be a function of *both* the measured and the desired plant states. Indeed, if $0 < m \leq g^{-1}(\mathbf{x}) \leq M_1$, and $|\dot{g}^{-1}(\mathbf{x})| \leq M_2$ on the compact set A, then taking $u_{pd} = -k_d s$, one obtains:

$$\frac{d}{dt}(g^{-1}s^2) = (\dot{g}^{-1} - k_d)s^2 + s(g^{-1}\boldsymbol{\lambda}_v^T \tilde{\mathbf{x}} - g^{-1}f - g^{-1}r + u_{ad}).$$

Defining the unknown function:

$$h(\mathbf{x}, \mathbf{x}_d, r) = g^{-1}(x, \ldots, x^{(n-2)}) \left(f(\mathbf{x}) - \boldsymbol{\lambda}_v^T \tilde{\mathbf{x}} + r\right)$$

and choosing $u_{ad}(t) = \hat{h}(\mathbf{x}(t), \mathbf{x}_d(t), r(t))$, where $\hat{h}$ is the output of a gaussian network with inputs $\mathbf{x}$, $\mathbf{x}_d$, and r, results in:

$$\frac{d}{dt}(g^{-1}s^2) = (\dot{g}^{-1} - k_d)s^2 + s\tilde{h}.$$

Choice of gain $k_d \geq M_2$ thus produces a passive mapping $s \mapsto \tilde{h}$, implying that the above adaptation laws can be used for stable adjustment of the output weights in $\hat{h}(\mathbf{x}, \mathbf{x}_d, r)$. A more refined version of this approach might also allow for state dependent bounds, i.e. choosing $k_d(\mathbf{x}) \geq M_2(\mathbf{x})$.

The technique displayed here can be used to extend the above ideas to a much more general class of systems. If the perfect tracking can be accomplished through reconstruction of the values assumed by an unknown nonlinear function on a compact set, and a measure of the tracking error can be found which is related to the function reconstruction error by a passive operator, then a gaussian network of the type described above can be used to adaptively compensate for the nonlinearity, and reduce tracking errors to a neighborhood of the zero.

4.4 Scaling issues

In the above, the Euclidean norm was used in the computing the activation level in each gaussian node, i.e. $\|x\|^2 = x_1^2 + x_2^2 + \ldots + x_n^2$. This implicitly assumes a certain scaling relation among each of the input signals. In his research with radial gaussian networks, Poggio [33] has found it useful to employ a weighted Euclidean norm, of the form $\|\mathbf{x}\|_W^2 = \mathbf{x}^T W \mathbf{x}$; the weighting matrix W is then iteratively adjusted to account for the relations among the input signals.

From the control standpoint, a (heuristic) measure of the scale relations between the signals in the system is already available in terms of the tracking error metric s. Using this metric, for instance when $n = 2$, $1/\lambda$ units of velocity error are as significant as a unit of position error, and similarly for higher dimensional systems. This suggests choosing a *fixed* diagonal matrix W whose diagonal consists of the squares of the coefficients in $\boldsymbol{\lambda}$. Potentially, such a scheme would allow the same network to approximate the required function regardless of the scaling of the measurements fed into it; further, recalling the scaling properties of the Fourier transform, this method might also permit the design of networks which provide the same approximating power with fewer nodes than the above algorithm would predict. These are intriguing extensions which deserve further investigation.

5 Example

In this section the above ideas are illustrated on a two dimensional plant. The desired trajectories were confined to a ball of radius $r_d = 2.75$, and the reconstruction set was chosen as $A = [-3.0, 3.0]^2$. The true plant was modeled by (2.1) with $n = 2$ and:

$$f(x, \dot{x}) = 6\sin(4\dot{x}) - 3\exp(-2|x|) - x - \dot{x}$$

The plant dynamics were simulated using a fourth order, fixed stepsize Runge-Kutta algorithm with $\Delta t = 0.001$ sec.

PD Component: The "PD" component of the control law is given by equation (4.6). Values of $k_d = 5$ and $\lambda = 10$ were used.

Auxiliary Component: The auxiliary component of the control law is given by equation (4.12). The boundary layer controlling the usage of this controller was chosen to be $\phi = 0.5$, and a value of $k_1 = k_2 = 20$ was used in both sigmoids. $|f|$ was assumed to be upper bounded by 20 on A, so the value $M_0 = 20$ was used.

Adaptive Controller: For the purpose of bounding the frequency domain properties of f, it can be assumed that the magnitude of f goes smoothly to zero outside of A. That is, the adaptive controller is designed to approximate a function which is exactly equal to f on A, but which vanishes smoothly outside this set. For this plant, the adaptive control component was designed by assuming the support of $F(\nu)$ was contained in a cube of radius 1. The resulting network consisted of nodes with a variance of $\sigma^2 = 3.14$, and a sampling grid with mesh $\Delta = 0.2$. A value of $\rho = 0.4$ was used in the truncation of expansion (4.2). This discretization was assumed to provide an accuracy of at least $\epsilon = 10^{-4}$ on A.

With these parameters, $\Gamma = [-3.4, 3.4]^2$ and there were hence a total of 1225 nodes in the network. The Euclidean norm was used for computing the activation level of each node. The adaptation gains were chosen as $k_a = 250$ in (4.14), and, outside the deadzone, the parameters were updated at a rate of 100 Hz. using a trapezoidal approximation to the integral.

Figure 1 illustrates the results when the desired trajectory is given by a 0.4 Hz. unity amplitude square wave filtered through an overdamped third order system with a bandwidth of 10 rad/sec. For comparison, the tracking errors obtained using just the "PD" and auxiliary components of the control law are also shown. Use of the neural adaptive controller produces approximately a sevenfold reduction in the worst case tracking errors.

6 Concluding Remarks

By assuming that the nonlinear function driving the plant dynamics is sufficiently smooth, we have shown that a network of neurons with a radial gaussian input-output characteristic can be designed which is capable of uniformly approximating this function on a compact set. By considering the feed-in weights to each node of this network as representing a fixed sampling mesh on $\mathcal{R}^n$, and adjusting only the feed-out weights during operation, we have demonstrated that a suitable choice of adaptation law, coupled with a constant gain and a modulated sliding controller, results in a stable closed loop system, and tracking errors which converge to a neighborhood of zero.

The method described can be straightforwardly extended to multi-input plants, Extensions of the technique indicated in Section 4.3 implies that the large body of existing adaptive control results for parameterized linear and nonlinear dynamics can be exploited to design stable adaptation laws when the parameterization is replaced with an appropriately designed gaussian network.

The method we have described by no means suggests that more conventional adaptive architectures should be discarded. Every piece of information available to the designer should be employed in constructing a control system, and in most cases, good estimates of basis functions driving the plant dynamics will be available. A neural controller can then be employed in parallel with these basis functions, to account for uncertainties which cannot be so easily parameterized.

Finally, note that to a close approximation, the sigmoid function used in back propagation networks is the integral of a gaussian. This suggests that the spatial frequency domain ideas suggested above could also be used in the design of three-layered back propagation networks.

References

[1] Adams, M. and Guillemin, V., *Measure Theory and Probability*, Wadsworth and Brooks/Cole, Monterey, CA, 1986.

[2] Aiserman, M. A., Braverman, E. M., and Rozonoer, L. I., "Potential functions technique and extrapolation in learning systems theory", *Proc. 3rd IFAC Congress*, 14G.1-14G.12, London, 1966.

[3] Arimoto, S., Kawamura, S., and Miyazaki, F., "Bettering operation of robots by learning", *Jour. Rob. Sys.*, 1(2), 123-140, 1984.

[4] Arimoto, S., Kawamura, S., Miyazaki, F., and Tamaki, S., "Learning control theory for dynamical systems", *Proc. 24th CDC*, 1375-1380, Ft. Lauderdale, FL, 1985.

[5] Arsac, J., *Fourier Transforms and the Theory of Distributions*, Prentice-Hall, Englewood Cliffs, N.J., 1966.

[6] Atkeson, C. G., "Memory-based approaches to approximating continuous functions", *Proc. 6th Yale Workshop on Adaptive and Learning Systems*, 212-217, Yale, New Haven, CT, 1990.

[7] Balestrino, A., DeMaria, G., and Zinober, A., "Nonlinear adaptive model following control", *Automatica*, 20, 559, 1984.

[8] Bracewell, R., *The Fourier Transform and Its Applications*, McGraw-Hill, New York, NY, 1965.

[9] Broomhead, D. S., and Lowe, D., "Multivariable functional interpolation and adaptive networks", *Complex Systems*, 2, 321-355, 1988.

[10] Courant, R., Hilbert, D., *Methods of Mathematical Physics, vol. I*, John Wiley and Sons, New York, NY, 1989.

[11] Funahashi, K., "On the approximate realization of continuous mappings by neural networks", *Neural Networks*, 2, 183-192, 1989.

[12] Girosi, F., and Poggio, T., "Networks and the best approximation property", *Artificial Intelligence Lab. Memo*, No. 1164, MIT, Cambridge, MA, October 1989.

[13] Hornik, K., Stinchcombe, M., and White, H., "Multilayer feedforward networks are universal approximators", *Neural Networks*, 2, 359-366, 1989.

[14] Horn, B.K.P., "Parallel networks for machine vision," *Artificial Intelligence Lab. Memo*, No. 1071, MIT, Cambridge, MA, December 1988.

[15] Horowitz, R., Messner, W., Moore, J., Kao, W., "Learning motion controllers for robot manipulators", *Proc. 6th Yale Workshop on Adaptive and Learning Systems*, 218-224, Yale, New Haven, CT, 1990.

[16] Horowitz, R., Messner, W., Moore, J., Kao, W., "Convergence properties of learning controllers for robot manipulators", *1990 Japan-USA Symp. on Flex. Autom.*, 635-642, Kyoto, Japan, 1990.

[17] Kawamura, S., Miyazaki, F., and Arimoto, S., "Realization of robot motion based on a learning method", *IEEE Trans. Sys., Man, and Cyber.*, 18, 126-133, 1988.

[18] Kawato, M., Furukawa, K., and Suzuki, R., "A hierarchical neural-network model for control and learning of voluntary movement", *Biological Cybernetics*, 57, 169-185, 1987.

[19] Kreyszig, E., *Introductory Functional Analysis with Applications*, John Wiley and Sons, New York, NY, 1978.

[20] Li, W., and Slotine, J. J. E., "Neural network control of unknown nonlinear systems," *Proc. 1989 ACC*, 1136-1141, 1989.

[21] Luenberger, D., *Optimization by Vector Space Methods*, John Wiley and Sons, New York, NY, 1969.

[22] Messner, W., Horowitz, R., Kao, W.-W., and Boals, M., "A New Adaptive Learning Rule", *IEEE Trans. Autom. Cont.*, 36,2, 188-197, 1991.

[23] Miller, W. T., An, E., and Glanz, F., "The design of CMAC neural networks for control", *Proc. 6th Yale Workshop on Adaptive and Learning Systems*, 218-224, Yale, New Haven, CT, 1990.

[24] Miller, W. T., Glanz, F. H., and Kraft, L.G., "Application of a general learning algorithm to the control of robotic manipulators", *Intl. Jour. Rob. Res.*, 6, 84-98, 1987.

[25] Munkres, J. R., *Analysis On Manifolds*, Addison-Wesley, Redwood City, CA, 1991.

[26] Narendra, K. S., and Annaswamy, A., *Stable Adaptive Systems*, Prentice Hall, Englewood Cliffs, NJ, 1989.

[27] Narendra, K. S., and Parthasarathy, K., "Identification and control of dynamical systems using neural networks, Part II", *Proc. 6th Yale Workshop on Adaptive and Learning Systems*, 218-224, Yale, New Haven, CT, 1990.

[28] Narendra, K. S., and Parthasarathy, K., "Identification and control of dynamical systems using neural networks", *IEEE Trans. on Neural Networks*, 1, 1, 4-27, March 1990.

[29] Petersen, D. P., and Middleton, D., "Sampling and reconstruction of wave number limited functions in n-dimensional Euclidean spaces", *Information and Control*, 5, 279-323, 1962.

[30] Peterson, B. B., and Narendra, K. S., "Bounded Error Adaptive Control", *IEEE Trans. Autom. Cont.*, 27,6, 1161-1168, 1982.

[31] Poggio, T. and Torre, V., "Ill-posed problems and regularization analysis in early vision", *Artificial Intelligence Lab. Memo*, No. 773, MIT, Cambridge, MA, April 1984.

[32] Poggio, T., and Girosi, F., "A theory of networks for approximation and learning", *Artificial Intelligence Lab. Memo*, No. 1140, MIT, Cambridge, MA, July 1989.

[33] Poggio, T., and Girosi, F., "Extensions of a theory of networks for approximation and learning: dimensionality reduction and clustering", *Artificial Intelligence Lab. Memo*, No. 1167, MIT, Cambridge, MA, April 1990.

[34] Sanner, R. M., and Akin, D. L., "Neuromorphic regulation of an underwater telerobot," *I.E.E.E Control Sys. Mag.*, April 1990.

[35] Sastry, S., and Bodson, M., *Adaptive Control: Stability, Convergence, and Robustness*, Prentice Hall, Englewood Cliffs, NJ, 1989.

[36] Slotine, J.-J. E., "Sliding controller design for nonlinear systems", *Intl. Journal of Control*, 40, 421, 1984.

[37] Slotine, J.-J. E. and Coetsee, J.A., "Adaptive sliding controller synthesis for nonlinear systems", *Intl. Journal of Control*, 1986.

[38] Slotine, J.-J. E. and Li, W., *Applied Nonlinear Control*, Prentice-Hall, Englewood Cliffs, NJ, 1991.

[39] Stakgold, I., *Boundary Value Problems of Mathematical Physics, vol. I*, Macmillan, Toronto, Ontario, 1967.

[40] Tikhonov, A. N., "Solutions of incorrectly formulated problems and the regularization method", *Soviet Math. Dokl.*, 4, 1035-1038, 1963.

[41] Tikhonov, A. N. and Arsenin, V. Y., *Solutions of Ill-Posed Problems*, Winston and Sons, Washington, D.C., 1977.

[42] Utkin, V. I., "Variable structure systems with sliding mode: a survey", *IEEE Trans. Auto. Contr.*, 22, 212, 1977.

Non-PostScript figures omitted.

Figure 1: Tracking errors versus time