NSL-910901, Sept. 1991
To appear: IEEE CDC, Dec. 1991

Stable Adaptive Control and Recursive Identification Using Radial Gaussian Networks

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

Abstract

Previous work has provided the theoretical foundations of a constructive design procedure for uniform approximation of smooth functions to a chosen degree of accuracy using networks of gaussian radial basis functions. This construction and the guaranteed uniform bounds were then shown to provide the basis for stable adaptive neurocontrol algorithms for a class of nonlinear plants.

This paper details and extends these ideas in three directions: first some practical details of the construction are provided, explicitly illustrating the relation between the free parameters in the network design and the degree of approximation error on a particular set. Next, the original adaptive control algorithm is modified to permit incorporation of additional prior knowledge of the system dynamics, allowing the neurocontroller to operate in parallel with conventional fixed or adaptive controllers. Finally, it is shown how the gaussian network construction may also be utilized in recursive identification algorithms with similar guarantees of stability and convergence.

1 Introduction

The current algorithms for stable adaptive control and recursive system identification can be seen as subsets of the more general learning problem. Each of these methods is attempting to sufficiently recover, in some manner, an unknown, possibly nonlinear function, $f(\mathbf{x})$, of a set of input signals, $\mathbf{x}$.

In order that this problem be practical, something must be assumed about the nature of f, for example that it is continuous, or differentiable, or periodic, etc. Full-state, linear adaptive control and identification algorithms assume that f can be written as a *linear combination* of the input signals x_i; or in other words the function $f(\mathbf{x})$ lies in the span of the basis functions $Y_i(\mathbf{x}) = x_i$. This intuition also allows the linear adaptive schemes to be extended to nonlinear systems for which a set of spanning basis functions is assumed known *a priori* [17].

In the more general case where f is perhaps only known to be continuous, such basis function expansions are still possible, but require generally an infinite number of $Y_i(\mathbf{x})$. For example, the classical Weierstrass theorem states that a univariate $f(x)$ can be represented as a linear combination of $Y_i(x) = x^i$, for $i = 0, \ldots, \infty$; the theorem extends naturally to n-dimensional input spaces. Of course, any application of these infinite dimensional expansions must acknowledge that a practical implementation will be capable of evaluating only a finite (possibly very large) number of the basis functions. The resulting expansions will thus be capable of only *approximating* the actual function f on a particular subset of the input space. In this context, it is necessary to ensure that the infinite expansion converges *uniformly* on the chosen set as the number of basis functions increases, and to quantify the relation between the size of the set, the magnitude of the uniform approximation error, and the number of basis functions employed.

Exploiting these ideas for identification and control is not new, see especially [1, 19], and more recently [9]; however, such ideas have not generally taken hold in the adaptive community, largely because of the computational expense (and resulting control loop delay) incurred computing the required basis functions. However, the recent results demonstrating the uniform approximation capabilities of neural networks, coupled with the extremely efficient computational paradigm offered by these models, especially if implemented in electronic or optical hardware, suggests that it is useful to re-examine the basis function approach, and rigorously derive stable adaptation algorithms which explicitly account for the limited region of validity of the approximation, and the unavoidable approximation errors even within this region.

Any additional prior information about f can be used to select the set of basis functions employed. Under the assumption that f is infinitely differentiable, [13] show that, for a class of least squares learning problems, gaussian radial functions are natural basis functions, and map directly onto a class of three-layered, feedforward neural networks [3, 5] By assuming further that f can be well *approximated by a function with compact spectral support*, [14] demonstrate a procedure for explicitly designing a gaussian network whose uniform approximation errors can be guaranteed to be within any prespecified bound; this construction and the guaranteed bound are then used to derive stable, closed-loop adaptation laws for a class of nonlinear systems controlled by the resulting networks.

In this paper, we consider a number of extentions and clarifications to the ideas in [14, 15]. First, practical details of the construction are provided, explicitly illustrating the relation between the free parameters in the network design and the degree of approximation error on a particular set. Next, the original adaptive control algorithm is modified to permit incorporation of additional prior knowledge of the system dynamics, allowing the neurocontroller to operate in parallel with conventional fixed or adaptive controllers. Finally, it is shown how the gaussian network construction and adaptive control designs may also be utilized in recursive identification algorithms which provide similar guarantees of stability and convergence.

2 Gaussian Approximation

It is by now well known that three layer feedforward neural networks, i.e. networks with one hidden layer of nonlinear nodes, can uniformly approximate continuous functions over compact subsets of their domains, e.g. [4, 6, 5]. This holds both for networks with hidden nodes which output smoothly saturating functions of their argument, such as the standard "sigmoid" operating on the dot product of the input signals and feed-in weights, as well as the "radial basis function" nodes, which output nonlinear functions of the euclidean distance from the network inputs to the corresponding set of feed-in weights.

The approximation implemented by such three layered networks can be represented mathematically as:

$$f_A(\mathbf{x}) = \sum_{i=1}^{N} c_i \, g_i(\mathbf{x}, \xi_i) \tag{2.1}$$

where g_i is the nonlinear function implemented by node i, the vectors ξ_i represent the input weights (or "center" in the radial basis function literature) of node i, and c_i represents the output weight for that node. The above referenced theorems thus state that, given a network (2.1), with the g_i chosen from a large class of "neural" models, and a desired tolerance level ϵ_f, the inequality $|f(\mathbf{x}) - f_A(\mathbf{x})| \le \epsilon_f$ holds for all $\mathbf{x}$ in a chosen set A for *some* (unspecified) values of the free parameters N, c_i, and ξ_i. As such they are existence theorems, yielding no information about the number of nodes or required values of the feed-in or feed-out weights.

Many heuristics have been proposed for selecting these parameters, usually involving the size and distribution of the set of examples used to train the network. A key metric in evaluating the performance of these heuristics is the extent to which the network can "generalize" from the examples in the training set, which is really a method of assessing the ϵ_f provided by the chosen algorithm. As will be demonstrated below, however, in order to derive adaptation laws which are guaranteed stable and convergent, it is necessary to ensure that the network structure is capable of being tuned to achieve a *prespecified* ϵ_f; good generalization is not just desirable, it is essential when using networks for estimation and control. Further, the points in the "training set" of the neurocontroller will be the continuously evolving values of the outputs of a dynamic system; what subset of these should be used to design and train the network?

Intuitively, the answers to these questions will depend upon the degree of *smoothness* exhibited by the function. Smoothness can be quantified in several ways; from an engineering standpoint one of the most familiar and useful of these is through the *spatial Fourier transform*:

$$F(\nu) = (\mathcal{F}f)(\nu) = \int_{\mathcal{R}^n} f(\mathbf{x}) \, e^{-2\pi j \nu^T \mathbf{x}} \, d\mathbf{x}$$

Note that f is being considered as a multivariate function of its the components of the vector $\mathbf{x}$ here, and *not* as an implicit function of time, as is more usual in control and filtering applications.

Assume that f is continuous and that $\mathcal{F}f$ exists and has compact support Supp $(F(\nu)) \subset B$, where $B = [-\beta, \beta]^n$. Under these conditions f can be *exactly* represented in terms of the values it assumes on a uniform, rectangular lattice over $\mathcal{R}^n$ [11]:

$$f(\mathbf{x}) = \sum_{I \in \mathcal{Z}^n} f(\xi_I) \, g_c(\mathbf{x} - \xi_I), \tag{2.2}$$

provided the lattice mesh, $\Delta \le 1/(2\beta)$. Here g_c is the canonical interpolating function, whose spectrum is Δ^n on B and zero elsewhere. The subscript I is an n-tuple of integers used to label the sampling points, ξ_I, so that if $I = \{i_1, \ldots, i_n\}$, the corresponding lattice point is given by

$$\xi_I = i_1 \Delta \mathbf{e}_1 + i_2 \Delta \mathbf{e}_2 + \cdots + i_n \Delta \mathbf{e}_n$$

where the $\mathbf{e}_i$ are the standard basis vectors for $\mathcal{R}^n$.

Starting from this "cardinal" series (2.2), the development of [14] exploited the invariance (modulo scale factors) of the radial gaussian under Fourier transformation, independent of the dimension of the underlying space, and the spatial low-pass structure of this function, to approximately recover f, uniformly on compact sets, with a truncated expansion in radial gaussians. That is, provided that f is continuous and sufficiently smooth (in a sense to be made precise below), by defining

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

and

$$g_\sigma(\mathbf{x}, \xi_I) = \frac{1}{\sigma_\nu^n} \mathcal{F}^{-1}(G_\sigma)(\mathbf{x} - \xi_I) = \exp(-\pi\sigma_\nu^2 \|\mathbf{x} - \xi_I\|^2),$$

a representation of the form

$$f_A(\mathbf{x}) = \sum_{I \in \mathcal{I}_o} c(\xi_I) \, g_\sigma(\mathbf{x}, \xi_I) \tag{2.3}$$

has the property that $|f(\mathbf{x}) - f_A(\mathbf{x})| \le \epsilon_f$ at every point in a chosen set A, and ϵ_f can be made as small as required by the appropriate prior choice of network parameters as detailed below. The summation in (2.3) is over the index set $\mathcal{I}_o = \{I \in \mathcal{Z}^n \mid \xi_I \in \Gamma\}$, where

$$\Gamma = \{\mathbf{x} \mid \|\mathbf{x} - \mathbf{y}\|_\infty \le l\Delta, \exists \mathbf{y} \in A\},$$

that is, Γ is the rectangle containing all lattice points within a distance $\rho = l\Delta$ of A, for some positive integer l. Note that since reconstruction of f is required only on the compact set A, many of the technical difficulties associated with the existence and invertibility of the Fourier transform can be avoided [15].

This radial gaussian expansion is clearly of the form (2.1), and hence also maps directly onto a class of three layer networks. There is one hidden layer node for every term in the series, and thus the number of nodes in the is network equal to the number of elements in the index set $\mathcal{I}_o$. The current point, $\mathbf{x}$, at which the approximation $f_A(\mathbf{x})$ is required, forms the input to the network, and the weights connecting these inputs to each gaussian node encode the uniform rectangular lattice ξ_I. 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 with variance σ_ν^2 of this activation, $\exp(-\pi r_I^2 \sigma_\nu^2)$. The network output is formed by weighting each nodal output by the corresponding $c_I = c(\xi_I)$, and summing together all the weighted outputs. The result represents the approximation $f_A(\mathbf{x})$. Note that each node has the same variance in this design.

While gradient methods are usually employed for determining the centers (ξ_I) and variances (σ_ν^2) of the gaussians, the one-to-one correspondence between (2.3) and the network representation permits a precise identification of the contribution of each network component to the approximation, allowing specification of a *constructive* network design procedure. In particular, given an estimate of the spectral support of f and the size of the set A, each of the free parameters in the design can be independently selected to reduce the sources of approximation error to achieve a prespecified ϵ_f.

Given a required ϵ_f, and the set, A, on which this approximation accuracy is required, the construction procedure starts by specifying the degree of smoothness which the unknown function f is assumed to exhibit. This is expressed through both an upper bound on the magnitude of the spectrum $|F(\nu)|$, and by the parameter β, representing the size of the "essential support" of the spectrum of this function: components of $F(\nu)$ for $\|\nu\|_\infty > \beta$ are assumed to contribute to f negligibly, in a sense to be quantified below, for all $\mathbf{x} \in A$.

The uniform approximation bound can be decomposed as $\epsilon_f = \epsilon_1 + \epsilon_2 + \epsilon_3$, where ϵ_1 represents the error introduced by approximating f using only frequencies of absolute value less than β, ϵ_2 is the error introduced into the cardinal series by the fact that the radial gaussian is not an ideal spatial low-pass filter, and ϵ_3 is the error introduced by truncating the cardinal series to a finite number of terms. Each of these can be bounded in terms of the prior information and the network design parameters $(\Delta, \sigma_\nu^2, l)$:

$$\epsilon_1 \leq \int_{B^c} |F(\nu)|\, d\nu \tag{2.4}$$

$$\epsilon_2 \leq \kappa_2 \Delta^n \int_{C_\Delta^c} \exp(\frac{-\pi\|\nu\|^2}{\sigma_\nu^2})\, d\nu \tag{2.5}$$

$$\epsilon_3 \leq \kappa_3 \sum_{k\in\mathcal{K}} 2^{k+1} \binom{n}{k} \int_l^\infty \alpha^k \exp(-\pi\alpha^2\Delta^2\sigma_\nu^2)\, d\alpha. \tag{2.6}$$

where the index set $\mathcal{K}$ is:

$$\mathcal{K} = \{0 \leq k \leq n-1 \mid n-k \text{ is odd }\}.$$

The first of these bounds is immediate from the definition of the Fourier transform, the second and third are derived in the appendix. Here $B^c = \mathcal{R}^n - B, C_\Delta = [-1/(2\Delta), 1/(2\Delta)]^n$, and the constants κ_2 and κ_3 are given by

$$\kappa_2 = \exp(\frac{n\pi\beta^2}{\sigma_\nu^2}) \sup_{\nu\in B} |F(\nu)|, \quad \text{and} \quad \kappa_3 = \Delta^n\, \kappa_2\, \sigma_\nu^n\, \beta^n. \tag{2.7}$$

The essential support radius, β, is assumed known sufficiently that ϵ_1 is as small as required; ϵ_2 and ϵ_3 can be upper bounded in terms of the remaining parameters in the network design. The mesh size Δ ($\leq 1/(2\beta)$) and a variance σ_ν^2 can be chosen so as to reduce ϵ_2 as much as required. Clearly these values are coupled: σ_ν^2 describes (roughly) the essential support of each gaussian in frequency, and to be an effective low-pass filter this support must be contained in C_Δ. A rule of thumb which yields good results for low dimensional networks is to take

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

Finally, l can be chosen to minimize the truncation error, ϵ_3. Since l and A define the set Γ, this choice plus the choice of Δ together determine the number of nodes in the network and the distribution of their centers. For example, if $A = [-m\Delta, m\Delta]^2$, then $\Gamma = [-(m+l)\Delta, (m+l)\Delta]^2$, and there would be a total of $N = [2(m+l)+1]^2$ nodes in the network, centered at $\xi_{\{i_1,i_2\}} = [i_1\Delta, i_2\Delta]^T$, for $i_1, i_2 = -(m+l), \ldots, (m+l)$.

The required output-layer weights c_I could be found explicitly by evaluating the function

$$c(\mathbf{x}) = \Delta^n\, \sigma_\nu^n\, \mathcal{F}^{-1}(FG_\sigma^{-1}\chi_B)(\mathbf{x})$$

at the lattice points ξ_I, where χ_B is the characteristic function of the set B. However, by assumption only the smoothness constraint, β, and an upper bound on the magnitude of the spectrum is assumed known about f; no prior information about the exact values of f or its spectrum are assumed. Hence, the goal of the training algorithms examined below will be to recursively adjust estimates, $\hat{c}_I$, of the output weights to attempt to match their correct values c_I.

Any additional information known about f can be used in sharpening the above bounds or reducing the size of the required representation. If information about the shape and orientation of the spectrum is known, the sampling lattice and mesh size can be adjusted accordingly. For example, if f is known to have a support which is much smaller in one direction in $\mathcal{R}^n$ the mesh size for the corresponding direction in the sampling grid on $\mathcal{R}^n$ can be taken proportionally larger, reducing the number of nodes required. If B is a ball in $\mathcal{R}^n$, instead of a cube, the required sampling density (and hence the number of lattice points contained in any compact Γ) can similarly be reduced by employing a hexagonal, as opposed to rectangular, sampling scheme [11, 7].

3 Adaptive Control

Consider developing an adaptive control algorithm for a class of dynamic systems whose equations of motion can be expressed in the canonical form:

$$x^{(n)}(t) + f(x(t), \dot{x}(t), \ldots, x^{(n-1)}(t)) = bu(t) \tag{3.1}$$

where $u(t)$ is the control input, f is an unknown linear or nonlinear function, and for simplicity in this paper, take $b = 1$; [15] extends the development to the general case, including state-dependent control gains. The control objective is to force the plant state vector, $\mathbf{x} = [x, \dot{x}, \ldots, x^{(n-1)}]^T$, assumed available for measurement, 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 $\tilde{\mathbf{x}}(t) \to \mathbf{0}$ as $t \to \infty$.

One approach to this problem is to take as the control input $u(t) = u_{pd}(t) + u_{ad}(t) + x_d^{(n)}$ 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 $x_d^{(n)}$ is a feedforward of the nth derivative of the desired trajectory. With use of this control law, (3.1) becomes:

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

where $\mathbf{k}$ is a vector of gains chosen to correspond to the coefficients of a Hurwitz polynomial (that is, a polynomial whose roots lie strictly in the open left half of the complex plane), and and $\tilde{f}(t)$ is the mistuning of the adaptive control component, $\tilde{f}(t) = (u_{ad}(t) - f(\mathbf{x}(t)))$.

If it is known *a priori* (as, e.g., in [17]) that $f(\mathbf{x})$ lies in the span of a set of known (linear or nonlinear) basis functions $Y_i(\mathbf{x})$, i.e.:

$$f(\mathbf{x}) = \sum_{i=1}^N a_i Y_i(\mathbf{x}) \tag{3.2}$$

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. The controller mistuning can be expressed as:

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

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

The design of such a law is possible for this system if a measure of the tracking error can be found which is sufficiently correlated, in some appropriate sense, with the approximation errors, $\tilde{a}_i(t)$. Given the structure of the error dynamics, and the linear dependence of the control mistuning $\tilde{f}(t)$ on the parameter mistuning, $\tilde{a}_i(t)$, this correlation condition is satisfied by any linear combination of the error states:

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

provided that the transfer function relating $\tilde{f}(t)$ to $s(t)$ is *strictly positive real* [10, 18]. Of course, this analysis holds unchanged if each Y_i is the ith coordinate projection function, i.e. $Y_i(\mathbf{x}) = x_i$, in which case these equations form a linear, full state feedback adaptive control algorithm.

Note that (3.2) has the same structure as the expansion (2.3) provided the input weights (lattice points) are held fixed, reflecting the confidence in the smoothness estimate, β. Further, if the desired trajectories are contained in a compact subset, A_d, of the state space, in principle the tracking problem posed by system (3.1) could be solved by a control law capable of reconstructing the unknown function f only upon on A_d. Thus a gaussian network with fixed centers and adjustable output weights could in principle be used as the adaptive component in the above adaptive tracking architecture. The small approximation error inevitably introduced could be viewed as a uniformly bounded disturbance driving the plant dynamics, and standard results of deadzone adaptation [10, 12] used to derive stable adaptive laws.

However, it is possible that, using such a scheme, the tracking errors would become sufficiently large during the initial stages of adaptation that the plant state would leave any prespecified A_d. The network design procedure outlined in the previous section provides no information about the approximation errors outside the chosen set. The possibly rapid degradation of this approximation outside a given subset of the state space thus presents an additional complexity in the design of a globally stabilizing controller. Simple deadzone adaptation will not suffice to ensure stability of the system under these conditions.

This problem can easily be overcome, however, by including in the control law an additional component which takes over from the adaptive component as its approximation ability begins to degrade, and forces the plant state back into A_d. This component takes the form of a sliding controller with a boundary layer [16], and its design is straightforward given upper bounds on the magnitudes of the functions being approximated. The action of this sliding term will be shown to be sufficient to reduce the tracking error during the times when the plant state lies outside of A_d, while the boundary layer prevents control chattering in these regions.

Previous algorithms [14] designed this sliding component to provide guarantees that the state remain within a predesignated subset of $\mathcal{R}^n$. This guarantee, however, also required constraints on the desired trajectories which could be commanded. This approach can be avoided by instead introducing a mechanism for taking the network off-line whenever the state moves outside the region on which the network approximation is valid. Similarly, it is beneficial to turn off the sliding controller when the approximation implemented by the adaptive controller is accurate, since the sliding controller relies on crude upper bounds of the plant nonlinearities to reduce tracking errors and is hence likely to require large amounts of control authority when active.

The complete control law thus has a dual character, acting as either a sliding or as an adaptive controller depending upon the instantaneous location of the plant state vector. However, discontinuously switching between adaptive and sliding components creates the possibility that the controller might chatter along the boundary between the two types of operation. To prevent this, pure sliding operation is restricted to the exterior of a slightly larger set A, containing A_d, while pure adaptive operation is restricted to the interior of the set A_d; in between, in the region $A - A_d$, the two modes are effectively *blended* using a continuous *modulation function*, which controls the degree to which each component contributes to the complete control law. The resulting controller smoothly transitions between adaptive and nonadaptive control strategies and, as will be shown below, is capable of globally stabilizing the systems under consideration.

3.1 Controller Overview

The structure of the conventional adaptive solutions to the tracking problem, combined with the above observations, suggests a control law with the general structure

$$u(t) = u_{pd}(t) + (1 - m(t))\, u_{ad}(t) + m(t)\, u_{sl}(t).$$

Here $u_{pd}(t)$ is a negative feedback term consisting of a weighted combination of both the measured tracking error states and a tracking metric $s(t)$, to be defined below. The term $u_{sl}(t)$, represents the sliding component of the control law, and similarly the adaptive component is represented by $u_{ad}(t)$. The function $m(t) = m(\mathbf{x}(t))$ is a continuous, state dependent modulation which allows the controller to smoothly transition between sliding and adaptive modes of operation, chosen so that $m(\mathbf{x}) = 0$ on A_d, $m(\mathbf{x}) = 1$ on A^c, and $0 < m(\mathbf{x}) < 1$ on $A - A_d$.

Without loss of generality, A_d can be chosen so to correspond to a unit ball with respect to an appropriate weighted norm function. Thus, for example,

$$A_d = \{\mathbf{x} \mid \|\mathbf{x}-\mathbf{x}_0\|_{p,w} \le 1\} \quad \text{and} \quad A = \{\mathbf{x} \mid \|\mathbf{x}-\mathbf{x}_0\|_{p,w} \le 1+\Psi\}.$$

Here Ψ is a positive constant representing the width of the transition region, $\mathbf{x}_0$ fixes the absolute location of the sets in the state space of the plant, and $\|\mathbf{x}\|_{p,w}$ is a weighted p-norm of the form

$$\|\mathbf{x}\|_{p,w} = \left\{ \sum_{i=1}^{n} \left(\frac{|x_i|}{w_i} \right)^p \right\}^{\frac{1}{p}},$$

or, in the limiting case $p = \infty$, $\|\mathbf{x}\|_{\infty,w} = \max_{i=1}^{n}(\frac{|x_i|}{w_i})$, for a set of strictly positive weights $\{w_i\}_{i=1}^n$. In $\mathcal{R}^2$, for example, with $p = 2$ the sets A_d and A are ellipses, and with $p = \infty$ these sets are rectangles.

With these definitions, the modulation function can be taken as,

$$m(\mathbf{x}(t)) = \max\left(0,\ \mathrm{sat}\left(\frac{(r(t)-1)}{\Psi}\right)\right), \qquad (3.3)$$

where $r(t) = \|\mathbf{x}(t) - \mathbf{x}_0\|_{p,w}$, and sat is the saturation function ($\mathrm{sat}(y) = y$ if $|y| < 1$, and $\mathrm{sat}(y) = \mathrm{sign}\,(y)$ otherwise). When $r(t) \le 1$, meaning that $\mathbf{x} \in A_d$, the output of the saturation function is negative, hence the maximum which defines $m(t)$ is zero, as desired. When $r(t) \ge 1 + \Psi$, corresponding to $\mathbf{x} \in A^c$, the saturation function is unity, hence $m(t) = 1$, again as desired. In between, for $\mathbf{x} \in A - A_d$, it is easy to check that $0 < m(\mathbf{x}) < 1$.

The adaptive control component, $u_{ad}(t)$, for this system implements an approximation, $\hat{f}_A(\mathbf{x})$, to the plant nonlinearity $f(\mathbf{x})$, which is realized as the output of the radial gaussian network described in Section 2, whose inputs are the measured values of the plant states. The variance and mesh size of this network are designed considering assumed upper bounds on the spectral properties of the plant nonlinearity $f(\mathbf{x})$, and the required uniform approximation error (whose relation to the steady state tracking errors will be made explicit below), bounded by expressions (2.4)-(2.6). Hence:

$$u_{ad}(t) = \hat{f}_A(\mathbf{x}(t)) = \sum_{I \in \mathcal{I}_o} \hat{c}_I(t)\, g_\sigma(\mathbf{x}(t), \xi_I). \qquad (3.4)$$

The input weights, ξ_I, which encode the sampling mesh, are *fixed* in this architecture, while the output weights, $\hat{c}_I(t)$, are to be adjusted to attempt to match their tuned values, c_I.

A useful tracking error metric for both the sliding and adaptive control subsystems is defined by

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

which can be rewritten as $s(t) = \lambda^T \tilde{\mathbf{x}}(t)$ with $\lambda^T = [\lambda^{n-1}, (n-1)\lambda^{n-2}, \ldots, 1]$. The equation $s(t) = 0$ 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 asymptotically obtained by maintaining this condition [20, 16]. Further, if the magnitude of s can be shown to be bounded by a constant Φ, the actual tracking errors can be shown [16] to be asymptotically bounded by:

$$|\tilde{x}^{(i)}(t)| \le 2^i \lambda^{i-n+1} \Phi, \qquad i = 0, \ldots, n-1. \qquad (3.5)$$

Using the metric $s(t)$ and the saturation function defined above, the sliding control component can be represented as

$$u_{sl}(t) = -k_{sl}(\mathbf{x}(t))\ \mathrm{sat}(s(t)/\Phi) \qquad (3.6)$$

where $k_{sl}(\mathbf{x})$ is the gain of the controller, and Φ is the boundary layer width; the exact values required for each of these parameters will be specified in the specific design which follows.

The deadzones required in the algorithm for updating the output weights of the gaussian network reflect the fact that no useful information can be gained about the quality of the current approximations when the tracking errors are less than a threshold determined by the bounds on the approximation errors, e.g. ϵ_f. However, as originally proposed in [12], deadzone adaptation requires discontinuously starting and stopping the parametric adjustment mechanism according to the magnitude of the error signal. These discontinuities can be eliminated using the metric $s(t)$, as shown in [17], by introducing the continuous function s_Δ, defined as

$$s_\Delta(t) = s(t) - \Phi\ \mathrm{sat}(s(t)/\Phi). \qquad (3.7)$$

3.2 Controller Design and Stability

Assume that a prior upper bound $M_0(\mathbf{x})$ is known on the magnitude of f for points outside of the set A_d, i.e.

$$|f(\mathbf{x})| \le M_0(\mathbf{x}) \quad \text{when} \quad \mathbf{x} \in A_d^c.$$

Use of a given control law $u(t)$ in the system (3.1) results in the error metric having a time derivative:

$$\dot{s}(t) = a_r(t) - f(\mathbf{x}) + u(t). \tag{3.8}$$

where $a_r(t) = \lambda_v^T \tilde{\mathbf{x}}(t) - x_d^{(n)}$ with $\lambda_v^T = [0, \ \lambda^{n-1}, (n-1)\lambda^{n-2}, \dots \lambda]$, and $x_d^{(n)}$ is the nth derivative of the desired trajectory. Having chosen a set A containing A_d as outlined above, let $f_A(\mathbf{x})$ be a radial gaussian approximation to $f(\mathbf{x})$ designed such that $|f - f_A| \le \epsilon_f$ uniformly on the set A. In terms of this approximation, (3.8) can be rewritten as:

$$\dot{s}(t) = a_r(t) - f_A(\mathbf{x}) + d(\mathbf{x}) + u(t),$$

where $d(\mathbf{x}) = f_A(\mathbf{x}) - f(\mathbf{x})$ satisfies $|d| \le \epsilon_f$. This expression, and the considerations of the previous section, suggests use of the control law:

$$u(t) = -k_d s(t) - a_r(t) + (1 - m(t))\, \hat{f}_A(\mathbf{x}(t)) + m(t)\, u_{sl}(t) \tag{3.9}$$

where k_d is a constant feedback gain. The sliding component of this control law is given by (3.6) with state dependent gains $k_{sl}(\mathbf{x}(t)) = M_0(\mathbf{x}(t)) + \epsilon_f$, and the adaptive component $\hat{f}_A(\mathbf{x}(t))$ is realized as the gaussian network described by equation (3.4) above, whose tuned output implements the approximation f_A. The modulation function which blends the two controller modes is specified by equation (3.3).

If the parameters in $\hat{f}_A(\mathbf{x}(t))$ are updated as:

$$\dot{\hat{c}}_I(t) = -\, k_a\, (1 - m(t))\, s_\Delta\, g_\sigma(\mathbf{x}(t), \xi_I), \tag{3.10}$$

where k_a is a positive constant determining the adaptation rate, and if Φ is chosen so that $\Phi \ge \epsilon_f / k_d$ in both the sliding controller and the calculation (3.7) of s_Δ, then all states in the adaptive system will remain bounded, and moreover the tracking errors will asymptotically converge to a neighborhood of zero given by (3.5).

To prove this assertion consider, similarly to [17], the Lyapunov function candidate:

$$V = \frac{1}{2}({s_\Delta}^2 + \frac{1}{k_a} \sum_{I \in \mathcal{I}_o} \tilde{c}_I^2) \tag{3.11}$$

where $\tilde{c}_I(t) = \hat{c}_I(t) - c_I$. While $\dot{s}_\Delta$ is not defined for $|s| = \Phi$, $(d/dt){s_\Delta}^2$ is well defined and continuous everywhere and can be written $(d/dt){s_\Delta}^2 = 2\, s_\Delta \dot{s}$. Hence, using (3.10), $\dot{V}(t) = 0$ when $|s| \le \Phi$. When $|s| > \Phi$, since s_Δ sat$(s/\Phi) = |s_\Delta|$, one has

$$\begin{aligned} \dot{V} &= -(k_d\, {s_\Delta}^2 + |s_\Delta|\, k_d\, \Phi) - s_\Delta f + s_\Delta (1 - m)\, \hat{f}_A \\ &\quad - |s_\Delta|\, m\, k_{sl} + k_a^{-1} \sum_{I \in \mathcal{I}_o} \tilde{c}_I \dot{\hat{c}}_I \end{aligned}$$

There are now two possibilities:

(i.) $\mathbf{x} \in A^c$: Here $m = 1$, so that the output of the adaptive controller $(1 - m)\hat{f}_A$ is identically zero, and from (3.10) the time derivatives of the parameter estimates similarly vanish, hence:

$$\begin{aligned} \dot{V} &= -(k_d {s_\Delta}^2 + |s_\Delta|\, k_d\, \Phi) - s_\Delta f - |s_\Delta|\, k_{sl} \\ &\le -k_d {s_\Delta}^2 + |s_\Delta|\, (|f| - k_{sl}). \end{aligned}$$

Since the sliding controller gains have been chosen so that $k_{sl} \ge |f|$ for all $\mathbf{x} \in A^c$, one obtains

$$\dot{V} \le -k_d {s_\Delta}^2 \quad \text{when} \quad \mathbf{x} \in A^c.$$

(ii.) $\mathbf{x} \in A$: Here $0 \le m \le 1$, and by rewriting f as

$$f = (1 - m)\, f_A + m\, f_A - d$$

and using (3.10) with the fact that $|f_A| \le |f| + |d|$, yields:

$$\dot{V} \le -k_d {s_\Delta}^2 + |s_\Delta|(|d| - k_d \Phi) + |s_\Delta|\, m\, (|f| + |d| - k_{sl}). \tag{3.12}$$

Since $|d| \le \epsilon_f$ by construction when $\mathbf{x} \in A$, and Φ has been chosen so that $\Phi \ge \epsilon_f / k_d$, the first term in parentheses is less than or equal to zero. Similarly, the choice of sliding controller gains shows that, with this bound on $|d|$, $k_{sl} \ge |f| + |d|$, so the second term in parentheses is also nonpositive. Thus,

$$\dot{V} \le -k_d {s_\Delta}^2 \quad \text{when} \quad \mathbf{x} \in A.$$

In summary, the above considerations prove that $\dot{V} \le -k_d\, {s_\Delta}^2$ for all $t \ge 0$, and hence from (3.11) if s_Δ and all the $\tilde{c}_I$ are bounded at time $t = 0$, they remain bounded for all $t \ge 0$. It remains to show that $s_\Delta \to 0$ as $t \to \infty$. One can easily establish the uniform boundedness of $\dot{s}_\Delta(t)$; an application of Barbalat's lemma then establishes convergence of s_Δ to 0 [15]. Hence the inequality $|s(t)| \le \Phi$ holds asymptotically, and the asymptotic bounds on the individual tracking errors follow using (3.5).

Since the desired trajectories are contained completely within A_d, and since the tracking errors converge to a neighborhood of zero, the above proof also shows that eventually all the plant trajectories will converge to a set which is either within a small neighborhood of the set A_d, or else is completely contained inside it. In particular, the sliding subsystem will asymptotically be used less and less, and, since the negative feedback terms decay to zero as the tracking error does, the control input will eventually be dominated by the outputs of the neural network, regardless of the initial conditions on the plant or network output weights.

Simulation studies have been carried out for systems of the form (3.1) with a variety of $f(\mathbf{x})$ and are reported in [15]; all show the predicted stability and convergence properties. [15] also extends the above derivation to include systems with state-dependent control gains.

3.3 Implementation Considerations

The above developments by no means suggest 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 may be available. A neural controller can then be employed in parallel with these to account for uncertainties which cannot be so easily parameterized over a particular operating range. In fact, there may be several different regions where the dynamic structure is more complicated than the elementary model captured by the assumed basis functions would predict, and the network can be designed to approximate the difference over each of these regions; that is, the regions described above as A need not be topologically connected in the state space describing the motion.

Adding a set of known basis functions in parallel with the network produces a controller which still has an adaptive component mathematically equivalent to (3.2), but now $N = N_{nn} + N_{bf}$ where N_{nn} is the number of gaussian nodes in the network approximation, and N_{bf} is the number of basis functions in the assumed dynamic structure. Even in the absence of knowledge of a particular set of basis functions, rather than have the network attempt to synthesize affine components of f directly from gaussians, the terms $Y_i(\mathbf{x}) = x_i$, and a bias term $Y_0(\mathbf{x}) = 1$, can be directly added to the adaptive component of the control law in practical applications.

The relative weights of these known basis functions are recursively adjusted by (3.10) replacing g_σ with the corresponding Y_i. However, since these additional basis functions are generally assumed to be *globally* exact, the modulation is omitted in computing their contribution to the control law and when adjusting their relative weights. Outside the set A, the sliding component of the controller needs only offset the incremental difference between the output of the known basis functions and the effects of the unmodeled dynamics which the network attempts to eliminate when the state is within A.

Adding affine terms to the representation can be effected by adding direct connections in the original network between the input layer and the output node, plus a constant bias term on the output. Incorporating a set of known basis functions can be pictured as adding a second three-layered network in parallel with the original gaussian network; each hidden node of this new network is connected to each input with unity weight and computes the nonlinear transform $Y_i(\mathbf{x})$ of the incoming signals.

The advantages of incorporating prior known basis functions are clear: the residual (unmodelled) components of the "true" nonlinear function influencing the dynamics are likely to be either very small in magnitude, or very restricted in their domain of influence. In either case, the gaussian network required to approximate these unmodelled components should be very much smaller than the network required if no prior information were avaliable. Gaussian network approximation, or more general basis function approximation techniques, are thus seen as methods for *augmenting* existing adaptive control schemes. The networks can be used to characterize elements of the plant dynamics for which explicit models do not exist, while the known dynamic structure can be directly exploited.

A good example of this idea can be found in robotic applications, for which the basis functions governing the motion of the joints as a function of the applied torques are known with great accuracy, and for which stable adaptive schemes exploiting this knowledge are well known. What is much more difficult to accurately characterize in these applications is the impact of Coulomb friction forces, which are most noticeable only at very low velocities. A neural network of the type proposed could be designed to estimate these friction forces in a very small neighborhood of the zero joint velocity region and operate alongside a conventional adaptive controller. Once again, the sliding subsystem in this case need offset only the incremental difference between the output of the known basis functions and the neural approximation.

The above algorithm lends itself immediately to implementation on parallel electronic or optical hardware; however, until such devices are readily available, it is useful to consider modifications which render the algorithm practical if implemented in software on a conventional serial microprocessor. In order to reduce the number of computations required at each time step, the output of each gaussian may be truncated to zero when the scaled distance πr_I^2 is greater than a few multiples of the variance. This will, of course, introduce another source of approximation error, but this too can be uniformly bounded and related to the (gaussian) truncation radius using frequency domain arguments. Similarly, to avoid the need to reserve huge amounts of storage at the outset, each node can be *instantiated only when the state enters the support of the truncated gaussian.* The set Γ is thus constructed on-line, and would resemble an n dimensional "tube" surrounding the trajectory of the plant through its state space.

4 Discrete Recursive Identification

The design procedure outlined in the previous sections produces networks which form a linear regression model in the parameters $\hat{c}_I$, implying that standard system identification techniques can be used to develop models of processes whose input-output time history is available. This section provides a brief sketch and example of this idea. While the above ideas can be directly applied to produce recursive identification algorithms for physical, continuous time, nonlinear dynamic systems, traditionally identification algorithms have considered discrete time processes, and this section conforms to that convention; the discrete nature of the parameter t is emphasized in the development by using a square bracket notation, e.g. $y[t]$ instead of $y(t)$.

As opposed to iterating over, or "batch processing", a complete input-output time history, the form of the identifier considered is recursive; the network operates alongside the process as it evolves, continuously updating its model in an attempt to drive the prediction error to zero. To this end, assume that the process model has the form:

$$y[t] = f(y[t-1], \ldots, y[t-N], u[t], \ldots, u[t-M+1])$$

where $y[t]$ is the process output at time t, $u[t]$ is the process input at time t, and upper bounds on N and M are known. It is assumed also that both $y[t]$ and $u[t]$ are available for measurement for every t.

As above, given an estimate of the spatial bandwidth of f in terms of its $N+M$ arguments, a gaussian network can be constructed sufficient to uniformly approximate f on some chosen set A. For simplicity, assume that prior bounds on the range of signals $y[t]$ and $u[t]$ are known, so that an A can be chosen which will encompass the entire range of signals encountered during operation of the identifier. Note that this can be a reasonable assumption in an identification model, where the estimates, $\hat{y}[t]$ do not influence the actual process $y[t]$. In the adaptive control situation considered above, however, the estimates of f directly influence the evolution of the states $\mathbf{x}$, necessitating the more complex algorithm and analysis.

For consistency with the notation of the previous section, collect the network inputs into the single vector

$$\mathbf{x}[t] \equiv [y[t-1], \ldots, y[t-N], u[t], \ldots, u[t-M+1]];$$

the network prediction of the output at time t can then be written as

$$\hat{y}[t] = \sum_{I \in \mathcal{I}_o} \hat{c}_I[t-1] g_\sigma(\mathbf{x}[t], \xi_I).$$

Defining the prediction error as $e[t] = \hat{y}[t] - y[t]$, one has

$$e[t] = \sum_{I \in \mathcal{I}_o} \tilde{c}_I[t-1]\, g_\sigma(\mathbf{x}[t], \xi_I) + d[t]$$

where $d[t] = f_A(\mathbf{x}[t]) - f(\mathbf{x}[t])$ satisfies $d[t] \leq \epsilon_f$ everywhere on A, and hence for all t.

Similarly to (3.7), define the adaptation signal to be

$$e_\Delta[t] = e[t] - \Phi \ \mathrm{sat}(e[t]/\Phi),$$

and take as the adaptation law for the output weights

$$\hat{c}_I[t] = \hat{c}_I[t-1] - k_a e_\Delta[t] g_\sigma(\mathbf{x}[t], \xi_I), \tag{4.1}$$

where k_a is a strictly positive adaptation gain. If the deadzone is then chosen so that $\Phi > \epsilon_f$, and the adaptation gain is taken as

$$0 < \delta \equiv 2 - k_a \sup_{\mathbf{x} \in A} \sum_{I \in \mathcal{I}_o} g_\sigma(\mathbf{x}, \xi_I)^2 \tag{4.2}$$

the parameter estimates will remain bounded and the prediction error will converge asymptotically to $|e[t]| \leq \Phi$ as $t \to \infty$.

To demonstrate the convergence of this algorithm, take as a Lyapunov function candidate:

$$V[t] = \sum_{I \in \mathcal{I}_o} (\tilde{c}_I[t])^2$$

so that, using the adaptation law (4.1), $\Delta V[t] = V[t] - V[t-1]$ can be expanded as

$$\Delta V[t] = k_a^2 e_\Delta[t]^2 \sum_{I \in \mathcal{I}_o} g_\sigma(\mathbf{x}[t], \xi_I)^2 - 2k_a e_\Delta[t] \sum_{I \in \mathcal{I}_o} \tilde{c}_I[t-1] g_\sigma(\mathbf{x}[t], \xi_I).$$

But since,

$$\sum_{I \in \mathcal{I}_o} \tilde{c}_I[t-1] g_\sigma(\mathbf{x}[t], \xi_I) = e_\Delta[t] + \Phi \ \mathrm{sat}(\frac{e[t]}{\Phi}) + d[t],$$

substituting and re-arranging reveals

$$\begin{aligned}\Delta V[t] \quad &\leq \quad -k_a e_\Delta[t]^2 (2 - k_a \sum_{I \in \mathcal{I}_o} g_\sigma(\mathbf{x}[t], \xi_I)^2) - 2k_a |e_\Delta[t]| (\Phi - \epsilon_f) \\ &\leq \quad -k_a \delta e_\Delta[t]^2.\end{aligned}$$

The parameter estimates are thus bounded; since by construction the regressors $g_\sigma(\mathbf{x}[t], \xi_I)$ are uniformly bounded, one can also conclude that $e_\Delta[t] \to 0$ as $t \to \infty$ (c.f. [2]). Note that the required bound on k_a depends only upon the number and centers of the nodes in the gaussian network and can easily be computed off-line. Alternately, the error signal can be normalized by total magnitude of the regressors at each time step, in which case one requires $2 - k_a = \delta > 0$ [2].

If prior bounds are not known for the process inputs and outputs, the modulation function of Section 3 can be used to halt estimation and adaptation when the process leaves the set for which the network was designed. The parameter estimates will remain bounded in this case, but it is not possible to guarantee convergence of the prediction error.

Prediction of Chaotic Time Series

As an example of the above ideas, consider for simplicity a one dimensional autonomous process model,

$$y[t] = f(y[t-1]).$$

The identifier in this case has just the scalar input $x[t] = y[t-1]$, leading to the gaussian network identification model

$$\hat{y}[t] = \hat{f}(y[t-1]) = \sum_{I \in \mathcal{I}_o} \hat{c}_I[t-1] g_\sigma(\mathbf{x}[t], \xi_I).$$

To make the example concrete, assume the actual transition map f is spatially bandlimited with $\beta = 100$, and that the observations $y[t] \in [0,1]$ for all t. The above design procedure suggests an array of gaussian nodes with variance $\sigma_\nu^2 = 10^4\pi$ with the centers arranged on a uniform lattice of mesh $\Delta = 10^{-3}$. Choosing a truncation radius of $\rho = 0.5$, the network consists of nodes corresponding to the lattice points contained in $\Gamma = [-0.5, 1.5]$, or a total of 2001 nodes. Using the upper bounds (2.4)-(2.6) these parameters conservatively yield $\epsilon_f \leq 10^{-3}$. Accordingly, Φ is taken as 10^{-3}, and the adaptation gains are $k_a = 0.1$.

The actual process was taken as the quadratic map

$$y[t] = 3.7513\, y[t-1](1 - y[t-1])$$

which is known to be chaotic for this choice of gain [8]. Note that if $y[0] \in [0,1]$, so also is $y[t]$ for all t, agreeing with the choice of A. Figure 1 shows a typical time series for this system, and the behavior of prediction error, $e[t]$, using this gaussian network. As expected, the prediction error eventually converges to the small deadzone.

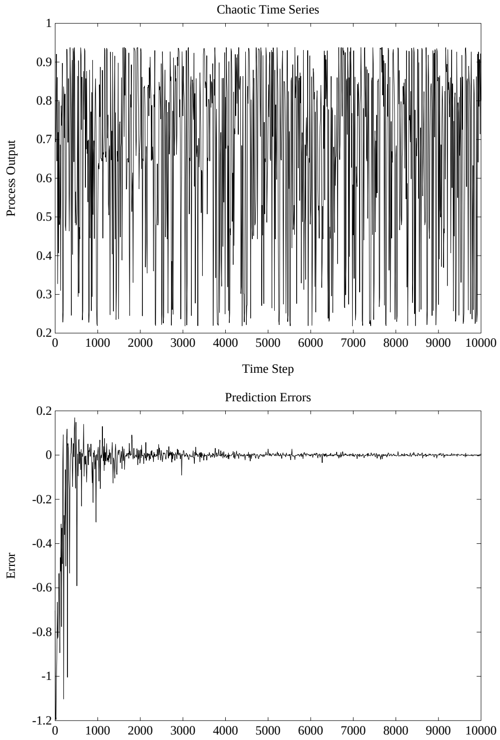

Figure 1: Recursive identification. Top: chaotic time series. Bottom: gaussian network prediction errors.

Further Applications

The above general model applies directly to the general problem of learning mappings from examples by taking $N = 0$ and $M = 1$, so that $y[t] = f(u[t])$. The stability properties of the constant gain recursive identifier are in this case exactly the stability and convergence properties of the corresponding gradient learning algorithm for these gaussian networks (again, leaving the input weights fixed). Note that arbitrary k_a cannot be chosen, but must be selected to satisfy (4.2), while the convergence guarantee depends upon stopping adaptation inside an appropriately defined deadzone in the prediction error.

The recursive identification paradigm, coupled with these constructive uniform approximation designs for gaussian networks, forms a powerful framework in which to evaluate many of the theoretical aspects of the general neural network learning problem. By casting the training algorithm as a dynamic system, one can take advantage of the many existing stability and convergence analysis techniques.

In particular, this viewpoint gives insight into the ability of a network to "generalize" beyond the examples in a given training set. Even given the uniform approximation capabilities of a particular network, good generalization for the pattern learning problem cannot be guaranteed unless the examples in the training set form a *persistently exciting* sequence. Roughly, persistency of excitation is a measure of the extent to which matching the training examples requires the full span of all the basis functions, and not just a linear subspace [10, 2, 18]. Since most learning from examples algorithms iterate to convergence over the training set, persistency can be evaluated by re-indexing time in the above models modulo the number of examples in the training set, and evaluating the excitation properties of the resulting infinite sequence of regressors as $t \to \infty$.

Finally, note that a great number of very complicated phenomenon may be accurately characterized by the above model. Stock prices, sunspot cycles, and weather patterns all may fit these assumptions, and hence the neural identification technique suggested here can be considered another tool for attempting to predict these processes. As in the previous section, the network model can be readily employed in parallel with other, especially linear, basis functions.

5 Concluding Remarks

Under the assumption that the nonlinear functions driving the plant dynamics are sufficiently smooth, we have shown that a network of neurons possessing radial gaussian input-output characteristics can be designed which is capable of uniformly approximating these functions on a compact set. By placing the elements of a gaussian network in one-to-one correspondence with elements of a modified cardinal series inspired by sampling theory, we have been able to assign a precise interpretation to each of the network parameters, and to demonstrate how these parameters are related to the properties of the function the network is required to approximate. 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, for both the control and the identification algorithms, a suitable choice of adaptation law results in a globally stable closed loop system and tracking or prediction errors which converge to a neighborhood of zero.

A unique feature of the control law is its ability to smoothly transition from adaptive to nonadaptive modes of operation, becoming a sliding controller in the regions of the state space where the network has poor approximating capability, a purely adaptive controller where the network approximating power is good, or a stabilizing blend of the two modes in an intermediate transition region. Further, prior knowledge about the structure of the plant dynamics can be directly incorporated into the algorithm, allowing the network to operate in parallel with conventional fixed or adaptive controllers, perhaps serving only to characterize those elements of the system dynamics for which no adequate physical model is known.

6 Appendix: Bound on Error Sources

In this appendix are derived crude upper bounds for the components in the uniform approximation error to f on A: the error ϵ_2, arising from the departure of the radial gaussian from the ideal low pass filter, and the error ϵ_3, introduced by truncating the infinite series expansion (2.2). In practice, of course, only order of magnitude estimates of these bounds are needed; the required deadzones can then be taken a factor of five or ten greater.

6.1 Gaussian Low-pass Filtering

In [15], this contribution is shown to be bounded by

$$\epsilon_2 \leq \left| \int_{C_\Delta^c} G_\sigma(\nu) \left[\sum_{I \in \mathcal{I}} C(\nu - \zeta_I) \right] e^{2\pi j \nu^T \mathbf{x}} d\nu \right|,$$

where $C(\nu) = \Delta^n F(\nu) G_\sigma^{-1}(\nu) \chi_B(\nu)$. Since Δ has been chosen so that $\Delta \leq 1/(2\beta)$,

$$\left| \sum_{I \in \mathcal{I}} C(\nu - \zeta_I) \right| \leq \Delta^n \sup_{\nu \in B} |F(\nu) G_\sigma^{-1}(\nu)| \leq \Delta^n \exp(\frac{n\pi\beta^2}{\sigma_\nu^2}) \sup_{\nu \in B} |F(\nu)|$$

Inequality (2.5) follows directly.

6.2 Truncation error

Consider a point $\mathbf{x} \in A$ taken to coincide with one of the points in the sampling lattice (the general case follows with only slight modifications [14]); i.e. $\mathbf{x} = \xi_J$ for some $J \in \mathcal{I}_o$. Assume for simplicity that the truncation radius ρ is taken as a multiple of the lattice mesh size, so that $\rho = l\Delta$ for some positive integer l.

To begin the analysis, picture the sampling lattice as consisting of a series of nested hypercubes surrounding the point ξ_J. Each point in the lattice thus lies on the face of exactly one of these nested hypercubes. Introducing the notation

$$R_m(\mathbf{y}) = \{\mathbf{x} \mid \|\mathbf{x} - \mathbf{y}\|_\infty = m\Delta\}$$

to denote the faces of the hypercube of radius $m\Delta$ from a point $\mathbf{y} \in \mathcal{R}^n$, and the index set

$$\mathcal{I}_m = \{I \in \mathcal{Z}^n \mid \xi_I \in R_m(\xi_J)\}$$

to denote the indices of the nodes which lie on the hypercube faces a distance $m\Delta$ from the selected point ξ_J, allows the truncation error to be expressed as:

$$\begin{aligned} \epsilon_3 &= \left| \sum_{I \in \mathcal{Z}^n} c_I\, g_\sigma(\xi_J, \xi_I) - \sum_{I \in \mathcal{I}_o} c_I\, g_\sigma(\xi_J, \xi_I) \right| && (6.3) \\ &= \left| \sum_{m=1}^{\infty} \sum_{I \in \mathcal{I}'_m} c_I\, g_\sigma(\xi_J, \xi_I) \right|, && (6.4) \end{aligned}$$

where $\mathcal{I}'_m = \mathcal{I}_m - \mathcal{I}_o$. By definition of the set Γ, $\mathcal{I}'_m = \emptyset$ so long as $m \leq l$. This equation simply states that the nodes in the sampling grid which lie on cubes of radius up to $l\Delta$ centered on the point ξ_J are by construction included in the network design and hence contribute to f_A. Thus for $m \leq l$ the inner summation vanishes identically.

From its definition, $C(\nu)$ is bounded with compact support, hence $|c(\mathbf{x})|$ is bounded for all $\mathbf{x} \in \mathcal{R}^n$ by a finite constant:

$$|c(\mathbf{x})| \leq \sigma_\nu^n \mathrm{Vol}\,(B) \sup_{\nu \in B} |C(\nu)| = \sigma_\nu^n\, \Delta^n\, \kappa_2\, \beta^n = \kappa_3,$$

where κ_2 is as defined in (2.7). The leading factor of σ_ν^n is the constant in the inverse transform of G_σ which has been absorbed into the definition of $c(\mathbf{x})$ in expansion (2.3). Substituting,

$$\epsilon_3 \leq \kappa_3 \sum_{m=l+1}^{\infty} \sum_{I \in \mathcal{I}'_m} g_\sigma(\xi_J, \xi_I).$$

For $m > l$, the inner summation no longer vanishes identically for every $J \in \mathcal{I}_o$, reflecting the outlying nodes omitted in the network construction. Since $g_\sigma(\xi_J, \xi_I)$ is always positive, it is certainly true, although somewhat conservative, to write

$$\epsilon_3 \leq \kappa_3 \sum_{m=l+1}^{\infty} \sum_{I \in \mathcal{I}_m} g_\sigma(\xi_J, \xi_I).$$

The conservatism here arises since, for $l+1 \leq m \leq m_0$, $\mathcal{I}'_m \subset \mathcal{I}_m$, where m_0 depends upon the diameter of A and the location within A of ξ_J; hence this inequality counts as omitted some of the nodes actually present in the network design.

Noting that each hypercube of radius $m\Delta$ aligned with the sampling lattice includes $(2m+1)^n$ lattice points, where n is the dimension $\mathbf{x}$, it is easy to confirm that $p(m) \equiv \mathrm{Card}\,(\mathcal{I}_m) = (2m+1)^n - (2m-1)^n$ is the number of terms in the inner summation for each m. Since $\min_{I \in \mathcal{I}_m} \|\xi_J - \xi_I\| = m\Delta$, and g_σ is a monotone decreasing function of this norm, the error bound can be rewritten as

$$\epsilon_3 \leq \kappa_3 \sum_{m=l+1}^{\infty} p(m) \exp(-\pi m^2 \sigma_\nu^2 \Delta^2).$$

The gaussian decays to zero faster than any power of m, hence for l sufficiently large the general term of this series is positive and monotone decreasing. With a shift of coordinates, the series represents a Riemannian lower sum bounded above by the integral:

$$\epsilon_3 \leq \kappa_3 \int_l^\infty p(\alpha) \exp(-\pi\alpha^2 \sigma_\nu^2 \Delta^2) d\alpha.$$

To evaluate $p(x)$ explicitly, use the binomial theorem to compute

$$p(\alpha) = \sum_{k \in \mathcal{K}} 2^{k+1} \binom{n}{k} \alpha^k,$$

where the index set $\mathcal{K}$ is as defined in Section 2. Hence,

$$\epsilon_3 \leq \kappa_3 \sum_{k \in \mathcal{K}} 2^{k+1} \binom{n}{k} \int_l^\infty \alpha^k \exp(-\pi\alpha^2 \sigma_\nu^2 \Delta^2) d\alpha \qquad (6.5)$$

and the convergent integrals can be evaluated by elementary methods.

To answer how large l must be taken for the transition to the integral representation to be valid, note that for positive α, direct calculation indicates that the product becomes decreasing at:

$$\frac{d}{d\alpha}\left[\alpha^k \exp(-\pi\alpha^2 \sigma_\nu^2 \Delta^2)\right] = 0 \quad \Rightarrow \quad \alpha = \sqrt{\frac{k}{2\pi\sigma_\nu^2}}$$

Thus the integrand is certainly monotone decreasing as long as l is taken larger than

$$l \geq \sqrt{\frac{n-1}{2\pi\sigma_\nu^2}}.$$

In practice, this constraint is likely to be much smaller than the value of l required to achieve the desired minimization of the integral in (2.6).

References

[1] 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.

[2] Astrom, K. J., and Wittenmark, B., *Adaptive Control*, Addison-Wesley, Reading, MA, 1989.

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

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

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

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

[7] Marks, R., J., *Introduction to Shannon Sampling and Interpolation Theory*, Springer-Verlag, New York, 1991.

[8] May, R. M., "Simple mathematical models with very complicated dynamics", *Nature*, 261, 459-467, 1976.

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

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

[11] 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.

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

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

[14] Sanner. R., and Slotine, J.-J. E, "Gaussian networks for direct adaptive control", *Proc. 1991 ACC*, Boston, vol. 3, pp. , June, 1991.

[15] Sanner. R., and Slotine, J.-J. E, "Gaussian networks for direct adaptive control", *NSL Report*, No. 910503, MIT, Cambridge, MA, May, 1991; submitted to *IEEE Trans. Neural Networks*

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

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

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

[19] Tsypkin, Y. A., *Adaptation and Learning in Automatic Systems*, Academic Press, New York, 1971.

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