DYNAMIC STRUCTURE NEURAL NETWORKS FOR STABLE ADAPTIVE CONTROL OF NONLINEAR SYSTEMS

Simon Fabri and **Visakan Kadirkamanathan**

Department of Automatic Control & Systems Engineering
University of Sheffield
PO Box 600, Mappin Street, Sheffield S1 4DU, UK
Email: visakan@acse.sheffield.ac.uk

Research Report No. **536**

Department of Automatic Control & Systems Engineering
The University of Sheffield

October 1994

DYNAMIC STRUCTURE NEURAL NETWORKS FOR STABLE ADAPTIVE CONTROL OF NONLINEAR SYSTEMS

Simon Fabri
Automatic Control & Systems Engineering Department
University of Sheffield, UK

Visakan Kadirkamanathan
Automatic Control & Systems Engineering Department
University of Sheffield, UK

Abstract

An adaptive control technique, using dynamic structure Gaussian radial basis function neural networks, that grow in time according to the location of the system's state in space is presented for the affine class of nonlinear systems having unknown or partially known dynamics. The method results in a network that is 'economic' in terms of network size, for cases where the state spans only a small subset of state-space, by utilising less basis functions than would have been the case if basis functions were centred on discrete locations covering the whole, relevant region of state-space. Additionally, the system is augmented with sliding control so as to ensure global stability if and when the state moves outside the region of state-space spanned by the basis functions, and to ensure robustness to disturbances that arise due to the network inherent approximation errors and to the fact that for limiting the network size, a minimal number of basis functions are actually being used. Adaptation laws and sliding control gains that ensure system stability in a Lyapunov sense are presented, together with techniques for determining which basis functions are to form part of the network structure. The effectiveness of the method is demonstrated by experiment simulations.

1 Introduction

The recent years have seen a development in the use of *neural networks* for the analysis and design of controlling nonlinear systems [21], [24], [2], [29], [33], [40], [19], [16], [32]. While the role of neural networks in these techniques are to learn some underlying functional related to the system, the methodologies differ in how they are utilised. They range from learning control action by reinforcement signals [1], learning the inverse dynamics by using trial and error responses [31], [21] and hence without any guarantee of stability, to *indirect adaptive control* based on neural network identification [24] and recently, *direct adaptive control* that guarantees stability of the overall system [29], [33], [40], [16], [32]. The neural networks used for control are the *multilayer perceptron* (MLP) with sigmoidal units [20], as in [31], [21], [24], [16], [32] and the *radial basis function* (RBF) networks [4], [28], as in [29], [33], [40], [1].

The neural direct adaptive controllers are designed for the *affine* class of nonlinear systems having unknown or uncertain dynamics. The control signals are generated based on approximate *feedback linearisation* techniques [36] using the neural network approximation to the functions (of the n-dimensional system state) representing dynamics [33], [40], [16]. Recently, neural direct adaptive control scheme for non-feedback linearisable systems have also been developed [32]. *Adaptation* in these direct adaptive controllers involves the on-line or sequential adjustment of the parameters of the neural networks based on adaptive laws derived in a manner similar to the classical Lyapunov-based *model reference adaptive control* (MRAC) design [3], [18], [22], [34] where stability of the system in the presence of adaptation is ensured. This guarantees the control aim of forcing the tracking error to tend to zero or at least remain bounded.

The radial basis function (RBF) network is more suitable for on-line or sequential adaptation, being insensitive to the order of presentation of the signals used for adaptation. With the use of Gaussian activation functions, the RBF network forms a local representation (as opposed to the sigmoidal MLP) where each basis function responds only to inputs in the neighbourhood of a reference vector stored as the *unit centre*. The spread of the neighbourhood is determined by the *unit variance* parameter.

In using RBF networks, one typically places the centre of the basis functions on regular points of say, a square mesh covering a relevant region of space where the state is known to be contained, denoted by a compact set $\chi_n \subset \Re^n$. This region therefore is the network approximation region, which in general is known for a given system. The distance between the points, say μ, affects the number of basis functions required to span the region χ_n and hence determines the size of the neural network. On the other hand, μ depends on the level of accuracy of approximation required of the neural network, where a large μ would lead to a coarser approximation while a small μ would lead to a finer approximation. In fact, as developed in [33], use of a systematic procedure is possible for determining the variances and centres of the basis functions that ensure the network approximation accuracy to be uniformly bounded everywhere within a relevant and finite region of state-space.

Systems with order higher than one ($n > 1$), typically exhibit the property that during operation, the state $\mathbf{x}(t)$ will scan only a subset of the state-space in χ_n [33], [40]. For the Gaussian RBF network, with the basis functions having a localised receptive field, most of the basis functions lie outside this subset and are not used for function approximation and their parameters are barely changed during adaptation. This implies that placing basis functions on all mesh points within χ_n results in a lot of redundancy. An efficient scheme to overcome this problem has already been developed for time-series analysis and system identification [27], [15], [13] and for classification [14], where the Gaussian RBF network is 'grown' by placing units in regions of state-space visited by the system during operation giving rise to a *dynamic network structure*. This results in a much smaller network in which all of the basis functions are utilised effectively and with much reduced storage and computational requirements.

A dynamic structure neural network has been developed for reinforcement learning controller design [1], which is based on the ideas of [27] used in time-series analysis. Similar suggestions for the direct adaptive controller have been made in [33], and in this paper, these suggestions and the growing network ideas from identification will be used in developing the dynamic structure neural network stable adaptive control scheme. The scheme developed here follows closely and makes use of the theory developed in [33], [40] with the ideas from identification [27], [12], [15] and makes appropriate modifications.

The use of neural networks with finite number of basis functions will inherently result in the presence of approximation errors which gives rise to a *disturbance* signal [33], [40], which in turn might lead to *parameter drift* [23], [22], [34]. The robust adaptive control scheme used in [33] is restricted in that to ensure stability, at any one time a minimum number of nodes or units needs to be activated in the vicinity of the state trajectory. This minimum number increases with the desired tracking accuracy. This is due to that fact that to ensure robustness to parameter drift [33] uses simple *dead-zone adaptation* [22], [36], [29], which requires the disturbance signal to be uniformly bounded with the size of the bound determining the size of the dead-zone [6], [26]. On the other hand, the dead-zone size also affects the tracking accuracy; the higher the tracking accuracy the smaller the dead-zone and hence smaller must be the bound on the disturbance to ensure robustness. The latter implies that more nodes need activation at any one time so that for the dynamic network scheme, the need for network growth becomes stringent if a greater tracking accuracy is desired. This in turn results in networks that grow unacceptably large, reducing the benefits of the dynamic network scheme.

In this paper, we offer a solution to the above problem by combining the ideas proposed in [39], [40], where the control law is augmented with a low gain *sliding control* [42], [41] term whilst $\mathbf{x} \in \chi_n$ to ensure robustness to the disturbance. The use of low gain sliding control results in a situation where the adaptation dead-zone, and hence the desired accuracy, is independent of the size of the disturbance term, since the disturbance is compensated for by sliding control. Thus, it appears as though there is no restriction on the minimum number of nodes requiring activation at any one time, so that the network growth rate becomes a design parameter whose choice does not affect the tracking accuracy. In practice however,

this argument must be modified slightly because as we shall see later, both the dead-zone width and the sliding gain affect the bandwidth of the control signal [36], [40]. The bandwidth allowed is bounded above so that the excitation of high frequency unmodelled dynamics is avoided. The limitation on the bandwidth forces a compromise to be struck between the tracking accuracy and the growth rate of the neural networks against the maximum control bandwidth.

In addition to the low gain sliding control, a high gain sliding control as in [33], is used for the additional property of forcing the state back into the network approximation region, whenever the state drifts outside it, *ie.*, $\mathbf{x} \notin \chi_n$. These techniques ensure convergence of the tracking error to a neighbourhood of zero and global stability, respectively. The effectiveness of the proposed neural adaptive control scheme is demonstrated for the class of systems considered in [39], [40].

In summary, a dynamic network scheme with node activation techniques is proposed for the neural network direct adaptive control scheme developed in [33], [40]. Appropriate modifications for the sliding control gain necessary to ensure global stability and convergence of the tracking error to a neighbourhood of zero are developed. The neural adaptive controller is optimal in terms of network size while guaranteeing stability and robustness in the presence of disturbances and approximation errors.

The background for the development of the dynamic structure scheme is provided in section 2, based largely on the system presented in [39], [40]. In section 3, concepts dealing with the dynamic network structure and the corresponding activation and adaptation laws are introduced. Stability of the system is analysed and the conditions necessary to ensure robustness of the closed loop system are determined. Implementation issues are also addressed, where a technique is suggested that speeds up the process of detecting if a node has been previously activated. Section 4 illustrates the operation of the adaptive control scheme using two example simulations, demonstrating the effectiveness of the system in keeping the size of the networks limited to smaller values when compared to a full-sized network scheme. Finally, the conclusions are provided in section 5.

2 Control of affine nonlinear systems

The control task is to ensure that the affine nonlinear system output tracks a desired output. The development of the ideas will be restricted to the case of single input single output (SISO) systems, noting that the extension to multiple input multiple output (MIMO) systems is straight forward.

Let the system output be $y(t)$. The system output vector $\mathbf{y}(t) = [y\ y^{(1)} \dots y^{(r-1)}]^T$ needs to track the desired output vector $\mathbf{y}_d(t) = [y_d\ y_d^{(1)} \dots y_d^{(r-1)}]^T$. The affine class of nonlinear

SISO systems can be expressed by equations in an affine form:

$$y^{(r)} = f(\mathbf{x}) + g(\mathbf{x})u(t) \tag{1}$$

where:

- $f(\mathbf{x}), g(\mathbf{x})$ represent nonlinear functions (namely the Lie derivatives of the system dynamics) of the state vector $\mathbf{x}(t) : \Re_+ \mapsto \Re^n$.
- r represents the relative degree of the system.
- $y^{(r)}$ represents the r^{th} derivative with respect to time of the system output.
- $y_d^{(i)}$ represents the i^{th} derivative with respect to time of the desired output, assumed bounded $\forall\ i,\ 0 \leq i \leq r$.
- $u(t)$ represents the input to the system, *ie.*, the control signal.

Theoretical descriptions of such systems are found in [11], [34], [36].

2.1 The control law

A state diffeomorphism $\mathbf{T}$ converting the states $\mathbf{x}$ to the so called *normal states* $\mathbf{z}$ *ie.*, $\mathbf{z} = \mathbf{T}(\mathbf{x})$, exist near any given point $\mathbf{x} = \mathbf{x}_o$ when the relative degree r of the system is defined and $r < n$, such that in the neighbourhood of $\mathbf{x}_o$, the system (1) can be expressed in the normal form as follows [11], [34], [25], [36]:

$$\begin{aligned}
\dot{z}_{1_1} &= z_{1_2} \\
&\vdots \\
\dot{z}_{1_{(r-1)}} &= z_{1_r} \\
\dot{z}_{1_r} &= f(\mathbf{x}) + g(\mathbf{x})u \\
\dot{\mathbf{z}}_2 &= \mathbf{I}(\mathbf{z}_1, \mathbf{z}_2) \\
y &= z_{1_1}
\end{aligned} \tag{2}$$

where $\mathbf{z} = [\mathbf{z}_1\ \mathbf{z}_2]^T$, $\mathbf{z}_1(t) : \Re_+ \mapsto \Re^r$ represents the output and its $(r-1)$ derivatives $(\mathbf{z} \equiv \mathbf{y}(t))$, $\mathbf{z}_2(t) : \Re_+ \mapsto \Re^{n-r}$ represents the internal dynamics and $\mathbf{I}(\mathbf{z}_1, \mathbf{z}_2)$ is a vector function of $\mathbf{z}_1$, $\mathbf{z}_2$. As proved in [34], if the functions $f(\mathbf{x})$, $g(\mathbf{x})$ are known, the zero-dynamics $\mathbf{I}(\mathbf{0},\ \mathbf{z}_2)$ of the system are globally exponentially stable and $\mathbf{I}(\mathbf{z}_1,\ \mathbf{z}_2)$ have continuous and bounded partial derivatives in $\mathbf{z}_1$ and $\mathbf{z}_2$, good tracking performance and stability is achieved for the affine class of systems when controlled by a law that keeps $\mathbf{z}_1$ bounded, such as the feedback linearization control law:

$$u(t) = \frac{-f(\mathbf{x}) + v(t)}{g(\mathbf{x})} \tag{3}$$

where

$$v(t) = y_d^{(r)} - \alpha_r e^{(r-1)} - \ldots - \alpha_1 e \tag{4}$$

is an auxiliary input,

$$e = (y - y_d) \tag{5}$$

is the *output tracking error* and coefficients α_i are chosen such that $\Gamma(s) = (s^r + \alpha_r s^{r-1} + \ldots + \alpha_1)$ is a Hurwitz polynomial, s being the usual Laplace variable.

Equations (1) and (3) show that this control law results in input – output dynamics that are linear and stable, namely:

$$y^{(r)} = v(t) \tag{6}$$

When the nonlinear functions are known but are linearly parameterised with unknown parameters, adaptive control techniques that do not require the use of neural networks have been developed ([35], [17], [38]). However in the absence of known Lie derivatives, one must resort to adaptive control techniques whereby approximations $\hat{f}(\mathbf{x})$, $\hat{g}(\mathbf{x})$ to the actual Lie derivatives $f(\mathbf{x})$, $g(\mathbf{x})$ are determined via the neural networks, and used in the control law:

$$u_{al}(t) = \frac{-\hat{f}(\mathbf{x}) + v(t)}{\hat{g}(\mathbf{x})} \tag{7}$$

as the closest possible law to equation (3) [40]. As in [39], [40], it is assumed that:

$$g(\mathbf{x}) \geq g_l(\mathbf{x}) \geq g_l > 0, \quad \forall\, \mathbf{x} \tag{8}$$

where $g_l(\mathbf{x})$ is a state-dependent lower bound on $g(\mathbf{x})$ and g_l is a constant lower bound on $g_l(\mathbf{x})$; both bounds being known.

To ensure global stability and convergence of tracking error to zero [33], [40], [39], control law (7) is augmented by a sliding mode control term $u_{sl}(t)$ [42], [36], such that the control input $u(t)$ becomes:

$$u(t) = u_{al}(t) + u_{sl}(t) \tag{9}$$

2.2 The neural networks

During this last decade, neural networks have been used effectively in pattern recognition, optimisation, signal classification and control problems [8], [21], [2]. The two principal types of neural networks that have emerged are the sigmoidal multiple layer perceptron networks [20] and the radial basis function (RBF) [30] neural networks [4], [28].

In [40], [39], a pair of Gaussian radial basis function neural networks are used to generate the approximations $\hat{f}(\mathbf{x})$, $\hat{g}(\mathbf{x})$ for $\mathbf{x} \in \chi_n$. The overall arrangement of the adaptive control scheme using neural networks is shown in Figure ??.

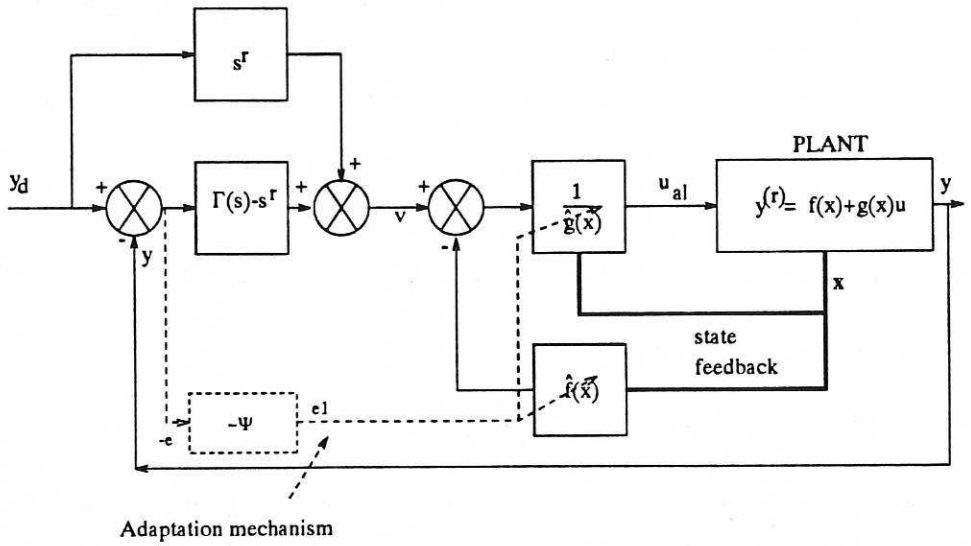

Figure 1: Neural network based adaptive control scheme.

The approximations are achieved via the functions:

$$\begin{aligned} \hat{f}(\mathbf{x}) &= \kappa(\mathbf{x})\hat{\mathbf{w}}_f^T(t)\Phi_f(\mathbf{x}) + f_o(\mathbf{x}) \\ \hat{g}(\mathbf{x}) &= \kappa(\mathbf{x})\hat{\mathbf{w}}_g^T(t)\Phi_g(\mathbf{x}) + g_o(\mathbf{x}) \end{aligned} \tag{10}$$

where,

- $\hat{\mathbf{w}}_f$, $\hat{\mathbf{w}}_g$ are k-dimensional parameter vectors
- $\Phi_f(\mathbf{x})$, $\Phi_g(\mathbf{x})$ are k-dimensional basis function vectors, whose i^{th} element is a Gaussian function of the form:

$$\begin{aligned} \Phi_{f_i} &= \exp\left\{\frac{-\|\mathbf{x}-\mathbf{m}_{f_i}\|^2}{2\sigma_f^2}\right\} \\ \Phi_{g_i} &= \exp\left\{\frac{-\|\mathbf{x}-\mathbf{m}_{g_i}\|^2}{2\sigma_g^2}\right\} \end{aligned} \tag{11}$$

  such that:

  - $\mathbf{m}_{f_i}, \mathbf{m}_{g_i}$ are n-dimensional vectors representing the *centre* of the i^{th} basis function
  - σ_f^2, σ_g^2 are the variances representing the *spread* of the basis functions.

- $f_o(\mathbf{x})$, $g_o(\mathbf{x})$ are known prior estimates of $f(\mathbf{x})$, $g(\mathbf{x})$ respectively.
- $\kappa(\mathbf{x})$ is a state-dependent function. This entails the definition of χ_n^- as a subset of the network approximation region χ_n, being slightly smaller than χ_n, so that $\kappa(\mathbf{x})$ is defined as:

$$\kappa(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \chi_n^- \\ 0 \le \bullet \le 1 & \text{if } \mathbf{x} \notin \chi_n^- \quad \text{and} \quad \mathbf{x} \in \chi_n \\ 0 & \text{if } \mathbf{x} \notin \chi_n \end{cases} \tag{12}$$

Further, it is assumed that given any uniform bounds ϵ_f, ϵ_g and the prior estimates f_o, g_o we could always find (non-unique) optimal parameter vectors $\mathbf{w}_f^*$, $\mathbf{w}_g^*$ and an optimal number k^* of basis functions, such that $\forall\ \mathbf{x} \in \chi_n$, the network approximation errors satisfy:

$$\begin{aligned} |\Delta_f| &= |f^*(\mathbf{x}) - f(\mathbf{x})| \leq \epsilon_f \\ |\Delta_g| &= |g^*(\mathbf{x}) - g(\mathbf{x})| \leq \epsilon_g \end{aligned} \tag{13}$$

where $f^*(\mathbf{x}) = \hat{f}(\mathbf{x}, \mathbf{w}_f^*, k^*)$ and $g^*(\mathbf{x}) = \hat{g}(\mathbf{x}, \mathbf{w}_g^*, k^*)$. This follows from the Universal Approximation property exhibited by feedforward neural networks [5], [7], [9], and in particular for our case, by radial basis functions [30].

The radial basis functions are placed on regular points of a square mesh in χ_n so that the mesh point separation, μ, depends directly on k^* and the size of χ_n. $\mathbf{w}_f^*$ and $\mathbf{w}_g^*$, the optimal parameter vectors that ensure the above accuracy are, however, unknown and must be estimated. This represents adaptation and is performed in an on-line manner, where the actual parameter estimates $\hat{\mathbf{w}}_f, \hat{\mathbf{w}}_g$ are adjusted recursively in time. The difference between the actual parameters and their optimal values is called the *parameter error*, denoted as:

$$\begin{aligned} \tilde{\mathbf{w}}_f(t) &= \hat{\mathbf{w}}_f(t) - \mathbf{w}_f^* \\ \tilde{\mathbf{w}}_g(t) &= \hat{\mathbf{w}}_g(t) - \mathbf{w}_g^* \end{aligned} \tag{14}$$

2.3 The adaptation laws

As derived in [39], [40] the parameter vectors $\hat{\mathbf{w}}_f, \hat{\mathbf{w}}_g$ are adapted according to the laws:

$$\begin{aligned} \frac{d\hat{\mathbf{w}}_f}{dt} &= \eta_f \kappa(\mathbf{x}) e_{1\Delta}(t) \Phi_f(\mathbf{x}) \\ \frac{d\hat{\mathbf{w}}_g}{dt} &= \eta_g \kappa(\mathbf{x}) e_{1\Delta}(t) \Phi_g(\mathbf{x}) u_{al}(t) \end{aligned} \tag{15}$$

where:

- $$e_{1\Delta}(t) = \begin{cases} 0 & \text{if} \quad -\phi < e_1 < \phi \\ e_1 - \phi \text{sign}(e_1) & \text{if} \quad |e_1| > \phi \end{cases} \tag{16}$$

  represents a dead-space function on $e_1(t)$.

- $e_1(t)$ is a filtration of the tracking error $e(t)$ such that $e_1(t) = \beta_r e^{(r-1)} + \ldots + \beta_1 e$, or in the s-domain $e_1(s) = \Psi(s)e(s)$, where the $(r-1)th$ order polynomial $\Psi(s)$ is chosen as a Hurwitz polynomial and such that $\Psi(s)\Gamma^{-1}(s)$ is strictly positive real [22], [3].

- η_f, η_g are positive adaptation rates.

- ϕ is the boundary layer thickness associated with the sliding mode control augmentation $u_{sl}(t)$ [36], [39], [40].

In addition, a parameter resetting mechanism is included that ensures that:

$$\hat{g}(\mathbf{x}) \geq g_l(\mathbf{x}) - \epsilon^*(\mathbf{x}), \quad \forall\, \mathbf{x} \in \chi_n \tag{17}$$

where $\epsilon^*(\mathbf{x})$ is a small positive function such that $g_l(\mathbf{x}) - \epsilon^*(\mathbf{x}) > 0$, $\forall\, \mathbf{x} \in \chi_n$ and $g_o(\mathbf{x}) \geq g_l(\mathbf{x})$, $\forall\, \mathbf{x} \notin \chi_n^-$.

Use of adaptation laws (15) and the control law:

$$u(t) = u_{al}(t) + u_{sl}(t) \tag{18}$$

where:

$$u_{sl}(t) = -\overline{k}(\mathbf{x}, t)\text{sat}\left(\frac{e_1}{\phi}\right) \tag{19}$$

with $\overline{k}(\mathbf{x}, t)$ being a state-dependent sliding gain and

$$\text{sat}(r) = \begin{cases} 1 & \text{if} \quad r \geq 1 \\ r & \text{if} \quad -1 < r < 1 \\ -1 & \text{if} \quad r \leq -1 \end{cases} \tag{20}$$

results in closed-loop system stability, with the filtered tracking error $e_1(t)$ converging to a value such that $|e_1(t)| \leq \phi$. This is ensured by Lyapunov stability considerations [39], [40].

3 Dynamic network structure

Dynamic structure neural networks are based on the idea that one could start with a network having no basis functions and gradually allocate basis functions in response to some conditions being satisfied. This way, the network 'grows' sequentially according to some rules which ensure that the function being approximated is learned up to the required levels of accuracy, using only a minimal number of basis functions. Such an approach contrasts with the fixed structure neural networks described in the previous section, whereby the number of basis functions is determined *a priori* and remains fixed. The use of such dynamic structure methods for Gaussian radial basis function networks in the context of system identification and function estimation is analysed in [27], [12], [15], [13]. For control applications Sanner and Slotine [33] suggest a dynamic network scheme that is still linked to the concept of a mesh as utilised in a fixed structure neural network. In this technique each mesh point represents a 'potential' location in state-space, where a basis function 'could' be placed if the need arises. Basis functions that are actually placed on such points represent *activated* basis functions, and the network 'grows' as more of the nodes are activated, contributing to the dynamic structure neural network. This technique of radial basis function placement shall be referred to as *selective node activation.*

This scheme is justified by considering adaptation rules (15), expressed in discrete form, namely:

$$\begin{aligned} \Delta\hat{\mathbf{w}}_f &= \eta_f' \kappa(\mathbf{x}) e_{1\Delta}(t) \Phi_f(\mathbf{x}) \\ \Delta\hat{\mathbf{w}}_g &= \eta_g' \kappa(\mathbf{x}) e_{1\Delta}(t) \Phi_g(\mathbf{x}) u_{al} \end{aligned} \tag{21}$$

where $\Delta\hat{\mathbf{w}}_f$, $\Delta\hat{\mathbf{w}}_g$ denote the change in the parameter vectors $\hat{\mathbf{w}}_f$, $\hat{\mathbf{w}}_g$.

Notice that if $\Phi_i(\mathbf{x})$, the i^{th} element of $\mathbf{\Phi}_f$ or $\mathbf{\Phi}_g$, is small then $\Delta\hat{\mathbf{w}}_i$ will also be small. Thus, if the i^{th} basis function is centered 'far away' from $\mathbf{x}(t)$, the change in the parameter for node i will be negligible so that no great change in performance should be noticed, if this change in parameter is ignored. Moreover, as shown in [33], if one starts with all parameters equal to zero, *ie.*, $\hat{\mathbf{w}}_f = \hat{\mathbf{w}}_g = \mathbf{0}$ at time $t = 0$, and assuming that, as noted above, one need not change the parameter of basis functions that are centered 'far away' from the current state $\mathbf{x}(t)$, we shall have radial basis functions with non-zero parameters (referred to as *active nodes*) only in the localized regions of state space where $\mathbf{x}(t)$ has ventured, referred to as *active space*. In this scheme, the network output is due to the contribution of the nodes centered on mesh points (in χ_n) that are located in regions where $\Phi_i(\mathbf{x})$ has been significantly large at some time during the interval from the start of operation up to the present instant. The rest of the basis functions, referred to as *passive* nodes, need not have existed at all since their contribution to the network output is nil. Thus, one could *activate* a basis function at any time t, only if its value is significantly large, unless otherwise it has already been activated previously. This *selective node activation* technique results in a dynamic structure basis function vector $\mathbf{\Phi}_a$ and its corresponding dynamic parameter vector $\hat{\mathbf{w}}_a$, composed of basis functions and parameters of *active* nodes only, rather than all the nodes in χ_n, as in equation (10). These vectors are *dynamic* in the sense that they can grow in size according to which nodes are activated and, if the active space is small compared to χ_n, the number of nodes in such dynamic networks is significantly less than that of the non-dynamic case, considered in the previous chapter, because the passive nodes are simply ignored.

Note that if the input is such that most regions of state-space in χ_n will be traversed, nearly all nodes in χ_n would be activated and the size of the dynamic network will be approximately equal to that of a non-dynamic scheme utilising a 'full-size' network. Hence there is no guarantee that the size of the dynamic network will be significantly less than that used in a 'full-size' network scheme. However, in a typical scenario, for systems of order higher than 1, it is the case that active space spans only a small subset of χ_n, resulting in a much smaller dynamic network when compared to a 'full-size' network.

Consider the case in which a node i (in χ_n) is activated at time t, provided that it was not previously activated and that its basis function satisfies the condition:

$$\Phi_i(\mathbf{x}(t)) \geq \delta_{min} \tag{22}$$

where δ_{min} represents an activation threshold (δ_{min} is constant and satisfies $0 < \delta_{min} \leq 1$) below which the output of the basis function is considered negligible.

Further, using the definition of Gaussian basis functions of equation (11), it follows that for a basis function centered at $\mathbf{m}_{f_i}$ or $\mathbf{m}_{g_i}$ to be activated, then respectively:

$$\begin{aligned} ||\mathbf{x}(t) - \mathbf{m}_{f_i}||^2 &\leq -2\sigma_f^2 \ln(\delta_{min}) \\ ||\mathbf{x}(t) - \mathbf{m}_{g_i}||^2 &\leq -2\sigma_g^2 \ln(\delta_{min}) \end{aligned} \tag{23}$$

In general, equation (23) represents a space enclosed within an n-dimensional ball of radius $\sqrt{-2\sigma^2 \ln(\delta_{min})}$, that is centered around $\mathbf{x}$(t), the present state. Hence, all nodes centered on mesh points within the ball must be activated, unless they have already been activated. Note that the larger the activation threshold, δ_{min}, the smaller the size of the ball, and so less nodes are activated at any one time, restricting the network growth rate further. Figure 2a shows the above ideas for a 2nd order system, where the ball is thus a circle.

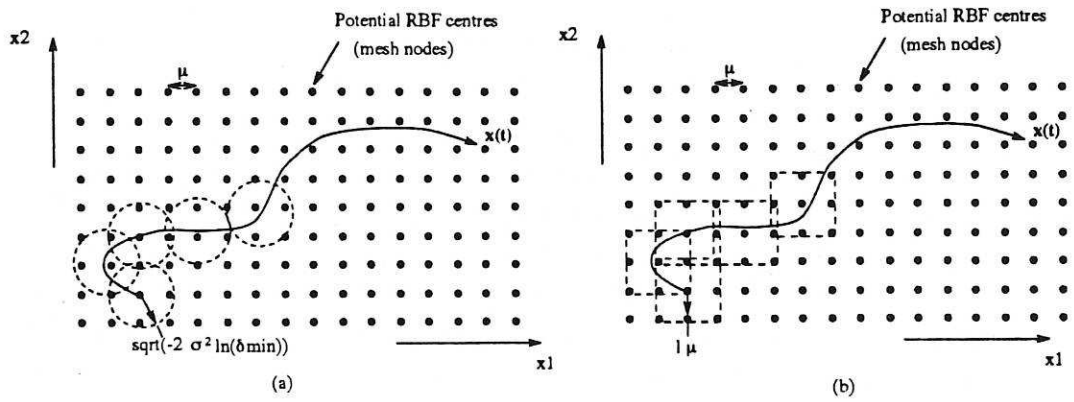

Figure 2: Selective node activation technique: using a hypersphere or a hypercube.

As will be clarified subsequently, although slightly more nodes would be activated, it is more convenient to activate nodes that are located on mesh points within a hypercube that is centred on the nearest mesh point to $\mathbf{x}(t)$ with length equal to $l\mu$, where l is the closest integer to $\sqrt{-2\sigma^2 ln(\delta_{min})}/\mu$, rather than a ball centred on $\mathbf{x}(t)$. This way, the hypercube edges coincide with the mesh points, as shown in Figure 2b. The main reason for this modification is the simplicity in evaluating the order of magnitude of that part of the disturbance term arising because of selective node activation. In addition, determination of the nodes to be activated at any one time is facilitated, an advantage whose benefits are felt more the higher the system dimensionality.

3.1 Dynamic network adaptive controller

Using a dynamic network scheme, the output of the neural network at time t consists of the contribution of the active basis functions. Denoting the basis functions and parameters of such active nodes as dynamic vectors Φ_{fa}, Φ_{ga}, $\hat{\mathbf{w}}_{fa}$, $\hat{\mathbf{w}}_{ga}$; for the $f(\mathbf{x})$ and $g(\mathbf{x})$ networks respectively and the output of the dynamic networks as $\hat{f}_a(\mathbf{x})$, $\hat{g}_a(\mathbf{x})$ then:

$$\begin{aligned} \hat{f}_a(\mathbf{x}) &= \kappa(\mathbf{x})\hat{\mathbf{w}}_{fa}^T(t)\Phi_{fa}(\mathbf{x}) + f_o(\mathbf{x}) \\ \hat{g}_a(\mathbf{x}) &= \kappa(\mathbf{x})\hat{\mathbf{w}}_{ga}^T(t)\Phi_{ga}(\mathbf{x}) + g_o(\mathbf{x}) \end{aligned} \tag{24}$$

Hence, in this scheme, the control law is:

$$u(t) = u_{al}(t) + u_{sl}(t) \tag{25}$$

where:

$$\begin{aligned} u_{al}(t) &= \frac{-\hat{f}_a(\mathbf{x})+v(t)}{\hat{g}_a(\mathbf{x})} \\ u_{sl}(t) &= -\overline{k}_a \text{sat}\left(\frac{e_1}{\phi}\right) \end{aligned} \tag{26}$$

$\overline{k}_a$ being the sliding mode gain to be determined subsequently.

Now, by definition of $f^*(\mathbf{x})$, $g^*(\mathbf{x})$ and equations (10), (13) we get:

$$\begin{aligned} f(\mathbf{x}) &= \kappa(\mathbf{x}) \mathbf{w}_f^{*T} \Phi_f - \Delta_f + f_o \\ g(\mathbf{x}) &= \kappa(\mathbf{x}) \mathbf{w}_g^{*T} \Phi_g - \Delta_g + g_o \end{aligned} \tag{27}$$

where for convenience, the argument $(\mathbf{x})$ has been dropped for Δ_f, Δ_g, f_o, g_o. Consider indexing $\mathbf{w}_f^*$, $\mathbf{w}_g^*$, the optimal, 'full-network' parameter vectors, such that the first series of elements are the optimal parameters of the active nodes, denoted by sub-vectors $\mathbf{w}_{fa}^*$, $\mathbf{w}_{ga}^*$, and the next series of elements are formed from the optimal parameters of passive nodes, denoted by sub-vectors $\mathbf{w}_{fp}^*$, $\mathbf{w}_{gp}^*$, which represent nodes not yet activated. Similarly, $\hat{\mathbf{w}}_f$, $\hat{\mathbf{w}}_g$, the full network parameter vectors are indexed accordingly into subvectors $\hat{\mathbf{w}}_{fa}$, $\hat{\mathbf{w}}_{ga}$, $\hat{\mathbf{w}}_{fp}$, $\hat{\mathbf{w}}_{gp}$. By definition, being passive, the last two subvectors and their time-derivative are zero vectors. Hence,

$$\hat{\mathbf{w}}_f = \begin{bmatrix} \hat{\mathbf{w}}_{fa} \\ \hat{\mathbf{w}}_{fp} \end{bmatrix}, \ \hat{\mathbf{w}}_g = \begin{bmatrix} \hat{\mathbf{w}}_{ga} \\ \hat{\mathbf{w}}_{gp} \end{bmatrix} \quad \text{and} \quad \mathbf{w}_f^* = \begin{bmatrix} \mathbf{w}_{fa}^* \\ \mathbf{w}_{fp}^* \end{bmatrix}, \ \mathbf{w}_g^* = \begin{bmatrix} \mathbf{w}_{ga}^* \\ \mathbf{w}_{gp}^* \end{bmatrix} \tag{28}$$

where $\hat{\mathbf{w}}_{fp}$, $\hat{\mathbf{w}}_{gp}$ are zero vectors. Using the same indices re-index similarly Φ_f, Φ_g, *ie.*,

$$\Phi_f = \begin{bmatrix} \Phi_{fa} \\ \Phi_{fp} \end{bmatrix}, \ \Phi_g = \begin{bmatrix} \Phi_{ga} \\ \Phi_{gp} \end{bmatrix} \tag{29}$$

where:

- $\Phi_{fa}(\mathbf{x})$, $\Phi_{ga}(\mathbf{x})$ contain the basis functions of active nodes.
- $\Phi_{fp}(\mathbf{x})$, $\Phi_{gp}(\mathbf{x})$ contain the basis functions of passive nodes.

Thus, substitution in equation (27) gives us:

$$\begin{aligned} f(\mathbf{x}) &= \kappa(\mathbf{x})(\mathbf{w}_{fa}^{*T} \Phi_{fa} + \mathbf{w}_{fp}^{*T} \Phi_{fp}) - \Delta_f + f_o \\ g(\mathbf{x}) &= \kappa(\mathbf{x})(\mathbf{w}_{ga}^{*T} \Phi_{ga} + \mathbf{w}_{fp}^{*T} \Phi_{gp}) - \Delta_g + g_o \end{aligned} \tag{30}$$

Hence, subtracting from equation (24) we get:

$$\begin{aligned} \hat{f}_a(\mathbf{x}) - f(\mathbf{x}) &= \kappa(\mathbf{x}) \tilde{\mathbf{w}}_{fa}^T \Phi_{fa} - \kappa(\mathbf{x}) \tilde{\mathbf{w}}_{fp}^T \Phi_{fp} + \Delta_f \\ \hat{g}_a(\mathbf{x}) - g(\mathbf{x}) &= \kappa(\mathbf{x}) \tilde{\mathbf{w}}_{ga}^T \Phi_{ga} - \kappa(\mathbf{x}) \tilde{\mathbf{w}}_{gp}^T \Phi_{gp} + \Delta_g \end{aligned} \tag{31}$$

where, $\tilde{\mathbf{w}}_{fa} = \hat{\mathbf{w}}_{fa} - \mathbf{w}_{fa}^*$, $\tilde{\mathbf{w}}_{ga} = \hat{\mathbf{w}}_{ga} - \mathbf{w}_{ga}^*$, $\tilde{\mathbf{w}}_{fp} = \hat{\mathbf{w}}_{fp} - \mathbf{w}_{fp}^* = -\mathbf{w}_{fp}^*$, $\tilde{\mathbf{w}}_{gp} = -\mathbf{w}_{gp}^*$.

Substitution of control law (25) in the system dynamics (1) and substituting for $v(t)$ and $e(t)$, implies:

$$e^{(r)} + \alpha_r e^{(r-1)} + \ldots + \alpha_1 e = f(\mathbf{x}) - \hat{f}_a(\mathbf{x}) + (g(\mathbf{x}) - \hat{g}_a(\mathbf{x}))u_{al} + g(\mathbf{x})u_{sl} \tag{32}$$

Also equation (3) implies that: $e^{(r)} + \alpha_r e^{(r-1)} + \ldots + \alpha_1 e \equiv \Gamma(s)e(s)$ in the Laplace domain, so that substitution of equation (31) in (32) results in the following error dynamics:

$$\Gamma(s)e = -\kappa\tilde{\mathbf{w}}_{fa}^T\Phi_{fa} - \kappa\tilde{\mathbf{w}}_{ga}^T\Phi_{ga}u_{al} + g(\mathbf{x})u_{sl}(t) + d_d(t) \tag{33}$$

where,

$$d_d(t) = \kappa\mathbf{w}_{fp}^{*T}\Phi_{fp} + \kappa\mathbf{w}_{gp}^{*T}\Phi_{gp}u_{al} - \Delta_f - \Delta_g u_{al} \tag{34}$$

represents a disturbance term that enters the error dynamics due to:

(i) The network inherent approximation errors $\Delta_f(\mathbf{x}), \Delta_g(\mathbf{x})$.

(ii) The selective node activation scheme, which ignores the contribution to the output of nodes that have not been activated, giving rise to errors in terms of the passive nodes.

Note that in [40], the disturbance was due only to (i), whilst in this case it is augmented by the errors due to selective node activation.

By equation (15), $e_1(t) = \Psi e(t)$ so that in terms of e_1, the error dynamics becomes:

$$e_1 = \Psi\Gamma^{-1}(-\kappa(\mathbf{x})\tilde{\mathbf{w}}_{fa}^T(t)\Phi_{fa} - \kappa(\mathbf{x})\tilde{\mathbf{w}}_{ga}^T(t)\Phi_{ga}u_{al}(t) + g(\mathbf{x})u_{sl}(t) + d_d(t)) \tag{35}$$

As in [33], [40], [39], we could use sliding control in the system with e_1 defining a sliding surface if $\Psi\Gamma^{-1}(s)$ has a first-order lag transfer function, say:

$$\Psi\Gamma^{-1}(s) = \frac{1}{s + k_d}, \qquad k_d > 0 \tag{36}$$

so that:

$$\dot{e_1} = -k_d e_1 - \kappa(\mathbf{x})\tilde{\mathbf{w}}_{fa}^T(t)\Phi_{fa} - \kappa(\mathbf{x})\tilde{\mathbf{w}}_{ga}^T(t)\Phi_{ga}u_{al}(t) + g(\mathbf{x})u_{sl}(t) + d_d(t) \tag{37}$$

where $u_{sl} = -\overline{k}_a \text{sat}\left(\frac{e_1}{\phi}\right)$.

We propose the use of adaptation laws as in [40], [39], but based only on *active* nodes, *ie.*,

$$\begin{aligned} \frac{d\hat{\mathbf{w}}_{fa}}{dt} &= \eta_f\kappa(\mathbf{x})e_{1\Delta}(t)\Phi_{fa}(\mathbf{x}) \\ \frac{d\hat{\mathbf{w}}_{ga}}{dt} &= \eta_g\kappa(\mathbf{x})e_{1\Delta}(t)\Phi_{ga}(\mathbf{x})u_{al}(t) \end{aligned} \tag{38}$$

together with a parameter-resetting mechanism:

$$\hat{\mathbf{w}}_{ga}(t^+) = \hat{\mathbf{w}}_{ga}(t) + (g_l - \hat{g}_{na})||\Phi_{ga}(\mathbf{x})||^{-2}\Phi_{ga}(\mathbf{x}) \tag{39}$$

which is activated at any time t if:

$$\hat{g}_{na}(\mathbf{x}) \leq g_l(\mathbf{x}) - \epsilon^*(\mathbf{x}), \ \forall\ \mathbf{x} \in \chi_n \tag{40}$$

where:

- $\hat{g}_{na}(\mathbf{x}) = \hat{\mathbf{w}}_{ga}^T(t)\Phi_{ga}(\mathbf{x}) + g_o$.
- $\epsilon^*(\mathbf{x})$ is a small, positive function such that $g_l(\mathbf{x}) - \epsilon^*(\mathbf{x}) > 0$, $\forall \mathbf{x} \in \chi_n$.
- $g_o(\mathbf{x}) \geq g_l(\mathbf{x})$, $\forall\ \mathbf{x} \notin \chi_n^-$.
- t^+ denotes the time just after the resetting mechanism is activated.

This resetting mechanism ensures that after a parameter reset, $\hat{g}_a(\mathbf{x}) \geq g_l(\mathbf{x})$, thus maintaining $\hat{g}_a(\mathbf{x})$ bounded below by $g_l(\mathbf{x}) - \epsilon^*(\mathbf{x})$.

3.2 Stability analysis

System stability is analysed by use of the Lyapunov function

$$V = \frac{e_{1\Delta}^2}{2} + \frac{1}{2\eta_f}\tilde{\mathbf{w}}_f^T\tilde{\mathbf{w}}_f + \frac{1}{2\eta_g}\tilde{\mathbf{w}}_g^T\tilde{\mathbf{w}}_g \tag{41}$$

where $\tilde{\mathbf{w}}_f = \hat{\mathbf{w}}_f - \mathbf{w}_f^*$, $\tilde{\mathbf{w}}_g = \hat{\mathbf{w}}_g - \mathbf{w}_g^*$.

(a) For $|e_1| < \phi$: In this case $e_{1\Delta} = 0$ so that equation (41) implies:

$$V = \frac{1}{2\eta_f}\tilde{\mathbf{w}}_f^T\tilde{\mathbf{w}}_f + \frac{1}{2\eta_g}\tilde{\mathbf{w}}_g^T\tilde{\mathbf{w}}_g > 0. \tag{42}$$

Hence, V is a suitable, positive-definite Lyapunov function candidate for the given range of e_1.

Differentiation with respect to time gives:

$$\begin{aligned} \dot{V} &= \tfrac{1}{\eta_f}\tilde{\mathbf{w}}_f^T\tfrac{d\hat{\mathbf{w}}_f}{dt} + \tfrac{1}{\eta_g}\tilde{\mathbf{w}}_g^T\tfrac{d\hat{\mathbf{w}}_g}{dt} \\ &= \tfrac{1}{\eta_f}\begin{bmatrix} \tilde{\mathbf{w}}_{fa}^T & \tilde{\mathbf{w}}_{fp}^T \end{bmatrix}\begin{bmatrix} \frac{d\hat{\mathbf{w}}_{fa}}{dt} \\ \frac{d\hat{\mathbf{w}}_{fp}}{dt} \end{bmatrix} + \tfrac{1}{\eta_g}\begin{bmatrix} \tilde{\mathbf{w}}_{ga}^T & \tilde{\mathbf{w}}_{gp}^T \end{bmatrix}\begin{bmatrix} \frac{d\hat{\mathbf{w}}_{ga}}{dt} \\ \frac{d\hat{\mathbf{w}}_{gp}}{dt} \end{bmatrix} \end{aligned} \tag{43}$$

where re-indexing of terms into active and passive nodes as before has been performed. Being a dynamic structure network, $\frac{d\hat{\mathbf{w}}_{fp}}{dt} = \mathbf{0}$, $\frac{d\hat{\mathbf{w}}_{gp}}{dt} = \mathbf{0}$, by definition of passive nodes, and equation (38) implies that for this range of e_1, $\frac{d\hat{\mathbf{w}}_{fa}}{dt}$, $\frac{d\hat{\mathbf{w}}_{ga}}{dt} = \mathbf{0}$, hence

$$\dot{V} = 0 \tag{44}$$

(b) For $|e_1| \geq \phi$: In this case $e_{1\Delta} = (e_1 - \phi\text{sign}(e_1))$ so that equation (41) implies:

$$V = \frac{(e_1 - \phi\text{sign}(e_1))^2}{2} + \frac{\tilde{\mathbf{w}}_f^T\tilde{\mathbf{w}}_f}{2\eta_f} + \frac{\tilde{\mathbf{w}}_g^T\tilde{\mathbf{w}}_g}{2\eta_g} > 0 \tag{45}$$

Hence, V is a suitable, positive-definite Lyapunov function candidate.

Differentiation of V and equation (37) gives:

$$\begin{aligned}\dot{V} &= -k_d e_{1\Delta}^2 - |e_{1\Delta}|(k_d\phi + \overline{k}_a g(\mathbf{x}) + \dot{\phi}) + e_{1\Delta} d_d(t) \\ &+ \tilde{\mathbf{w}}_{fa}^T(\frac{1}{\eta_f}\frac{d\hat{\mathbf{w}}_{fa}}{dt} - e_{1\Delta}\kappa\Phi_{fa}) + \tilde{\mathbf{w}}_{ga}^T(\frac{1}{\eta_g}\frac{d\hat{\mathbf{w}}_{ga}}{dt} - e_{1\Delta}\kappa\Phi_g u_{al}(t))\end{aligned} \tag{46}$$

Using adaptation laws (38) and letting ϕ be a constant, we get:

$$\dot{V} = -k_d e_{1\Delta}^2 - |e_{1\Delta}|(k_d\phi + \overline{k}_a g(\mathbf{x})) + e_{1\Delta} d_d(t) \tag{47}$$

Choosing

$$\overline{k}_a = \max\left\{0, \frac{\overline{d}_d - k_d\phi}{g_l}\right\} \tag{48}$$

implies that $\dot{V} \leq -k_d e_{1\Delta}^2$, where $\overline{d}_d(\mathbf{x})$ represents the bound on $d_d(t)$, the disturbance term of equation (34), *ie.*, $|d_d(t)| \leq \overline{d}_d(\mathbf{x})$. Evaluation of this bound shall be expanded on subsequently. Hence, adaptation laws (38) and sliding gain (48) ensure that for all $e_1(t)$, in the absence of parameter resets,

$$\dot{V} \leq -k_d e_{1\Delta}^2 \leq 0 \tag{49}$$

This, in turn, implies boundedness of $e_{1\Delta}(t)$, $\tilde{\mathbf{w}}_f$, $\tilde{\mathbf{w}}_g$ by Lyapunov stability theory.

In the event of parameter resets, the change in the Lyapunov function V of equation (41) following a reset, denoted by ΔV, can be deduced to be:

$$\Delta V = \frac{1}{2\eta_g}(\hat{g}_{na} - g_l + 2(g_l - g_{na}^*))(g_l - \hat{g}_{na})||\Phi_{ga}||^{-2} \tag{50}$$

where $g_{na}^* = \mathbf{w}_{ga}^{*T}\Phi_{ga}(\mathbf{x}) + g_o$ and $\hat{g}_{na}$ is as defined before. Defining g_n^* as the optimal full-network approximation to $g(\mathbf{x})$ when $\mathbf{x} \in \chi_n$;

$$\begin{aligned}g_n^* &= g_o + \mathbf{w}_g^{*T}\Phi_g(\mathbf{x}) \\ &= g_o + \mathbf{w}_{ga}^{*T}\Phi_{ga}(\mathbf{x}) + \mathbf{w}_{gp}^{*T}\Phi_{gp}(\mathbf{x}) \\ &= g_{na}^* + \mathbf{w}_{gp}^{*T}\Phi_{gp}(\mathbf{x})\end{aligned} \tag{51}$$

where the definitions of $\mathbf{w}_{ga}^*$, $\mathbf{w}_{gp}^*$, g_{na}^* have been used. Then, it follows that

$$g_n^*(\mathbf{x}) - g_{na}^*(\mathbf{x}) \leq \max(\mathbf{w}_{gp}^{*T}\Phi_{gp}(\mathbf{x})) \tag{52}$$

where $\max(\mathbf{w}_{gp}^{*T}\Phi_{gp}(\mathbf{x}))$ is an upper bound on $\mathbf{w}_{gp}^{*T}\Phi_{gp}(\mathbf{x})$.

If we assume that $g_l(\mathbf{x})$ is such that

$$g_n^*(\mathbf{x}) - g_l(\mathbf{x}) \geq \max(\mathbf{w}_{gp}^{*T}\Phi_{gp}(\mathbf{x})) \tag{53}$$

a fact that can be ascertained by choosing the lower bound $g_l(\mathbf{x})$ lower than $g_n^*(\mathbf{x})$ by at least $max(\mathbf{w}_{gp}^{*T}\Phi_{gp}(\mathbf{x}))$, whose value is to be discussed later, then:

$$g_l(\mathbf{x}) - g_{na}^*(\mathbf{x}) \leq 0 \tag{54}$$

This ensures that

$$\Delta V \leq -\frac{1}{2\eta_g}(g_l - \hat{g}_{na})^2 ||\Phi_{ga}||^{-2} \leq 0 \tag{55}$$

so that the Lyapunov function is *reduced* as a result of parameter resetting, if (54) is satisfied.

The above assumption necessary to ensure inequality (54) is different from that in [40], where a full-size network is used, makes it sufficient to assume that $g_n^*(\mathbf{x}) - g_l(\mathbf{x}) \geq 0$ for maintaining $\Delta V \leq 0$. This difference in assumptions though, is necessary to ensure stability in the presence of parameter resets for the case of a dynamic structure network.

Hence adaptation laws (38), sliding gain law (48) and the proposed parameter resetting mechanism, ensure boundedness of $e_{1\Delta}(t)$, $\tilde{\mathbf{w}}_f$, $\tilde{\mathbf{w}}_g$. In turn, the following considerations apply as in [40], [39]:

1. Boundedness of $e_{1\Delta}(t)$ implies boundedness of $e_1(t)$.
2. Boundedness of $\tilde{\mathbf{w}}_f, \tilde{\mathbf{w}}_g$ implies boundedness of $\tilde{\mathbf{w}}_{fa}$, $\tilde{\mathbf{w}}_{ga}$.
3. Boundedness of $\tilde{\mathbf{w}}_{fa}$, $\tilde{\mathbf{w}}_{ga}$ implies boundedness of $\hat{\mathbf{w}}_{fa}$, $\hat{\mathbf{w}}_{ga}$.
4. Tracking error $e(t)$ and its derivatives up to $(r-1)$ are bounded since $e_1(t)$ is bounded and $\Psi(s)$ is Hurwitz. Thus, assuming the system zero dynamics to be globally exponentially stable, the function $\mathbf{I}(\mathbf{z}_1, \mathbf{z}_2)$ has continuous and bounded partial derivatives in $\mathbf{z}_1$ and $\mathbf{z}_2$ and that the desired output and its derivatives up to order r are bounded. It follows that $\mathbf{x}(t)$ is bounded [34].
5. Assuming $f(\mathbf{x})$, $g(\mathbf{x})$ to be continuous, boundedness of $\mathbf{x}$ implies that $f(\mathbf{x})$, $g(\mathbf{x})$ are bounded.
6. $\Phi_{fa}(\mathbf{x})$, $\Phi_{ga}(\mathbf{x})$ are bounded since $\mathbf{x}(t)$ is bounded and so, together with boundedness of $\hat{\mathbf{w}}_{fa}$, $\hat{\mathbf{w}}_{ga}$ we conclude that $\hat{f}_a$, $\hat{g}_a$ are bounded.
7. $v(t)$ is bounded since $e(t)$ and its derivatives up to order $(r-1)$ and $y_d(t)$ and its r derivatives, are bounded. Also, parameter resetting ensures that $\hat{g}_a(\mathbf{x})$ is bounded away from zero so that u_{al} is bounded.
8. Disturbance $d_d(t)$ is bounded (via $\overline{d}_d$) since all terms on the right of equation (34) are bounded and $u_{sl}(t)$ is bounded since $\overline{k}_a$ is bounded. Hence, equation (37) implies that $\dot{e}_1$ is bounded, since all terms on its right hand side are so.

Thus $\dot{e}_{1\Delta}$ is bounded as well, so that $e_{1\Delta}$ is uniformly continuous. Since $\dot{V} \leq -k_d e_{1\Delta}^2$ then $\int_0^t \dot{V} + k_d e_{1\Delta}^2 d\tau \leq 0$. Letting $V_1(t) = V(t) - \int_0^t \dot{V}(\tau) + k_d e_{1\Delta}^2 d\tau$, the above condition implies that, $V_1(t) \geq 0$ so that $V_1(t)$ is *bounded below.* Also, time differentiation of $V_1(t)$ implies that $\dot{V}_1 = -k_d e_{1\Delta}^2$, so that $\dot{V}_1(t) \leq 0$, making it *semi negative-definite.* Finally, since $e_{1\Delta}$ is uniformly continuous, $\dot{V}_1$ *is also uniformly continuous.*

Hence, these three conditions allow use of Barbalat's Lemma [36], [22], [34] to deduce that:

$$\dot{V}_1(t) \to 0 \quad \text{as} \quad t \to \infty \quad \text{so that} \quad e_{1\Delta}(t) \to 0 \quad \text{as} \quad t \to \infty \tag{56}$$

This implies that $e_1(t)$ converges to somewhere within $\pm\phi$ as $t \to \infty$ or:

$$|e_1(t)| \leq \phi \quad \text{as} \quad t \to \infty \tag{57}$$

Since $e(t)$ is a stable filtration of $e_1(t)$ via $\Psi^{-1}(s)$, boundedness of $e_1(t)$ implies boundedness of $e^{(i)}(t)$ for $i = 0, 1, \ldots, (r-1)$. Thus ϕ determines the value of the steady-state tracking error. Note that if $\Psi(s) = (s+\lambda)^{r-1}$, where λ is a positive constant, then as shown in [36], in the steady-state:

$$|e^{(i)}| \leq 2^i \lambda^{i-r+1} \phi \qquad 0 \leq i \leq r-1 \tag{58}$$

Hence, use of selective node activation optimises on network size whilst ensuring stable closed-loop dynamics, if equations (38), (48) and the relevant assumptions are adhered to. Let us now take a look at the function representing the disturbance bound, required for evaluating the sliding mode gain $\overline{k}_a$ given in equation (48).

3.3 The disturbance bound

According to equation (34), the disturbance is bounded by:

$$|d_d| \leq \kappa |\mathbf{w}_{fp}^{*T} \Phi_{fp}| + \kappa |\mathbf{w}_{gp}^{*T} \Phi_{gp}||u_{al}| + |\Delta f(\mathbf{x})| + |\Delta g(\mathbf{x})||u_{al}|$$

In χ_n, the network approximation region; $\kappa(\mathbf{x}) = 1, |\Delta f(\mathbf{x})| \leq \epsilon_f$ and $|\Delta g(\mathbf{x})| \leq \epsilon_g$ by equation (13). Hence,

$$|d_d| \leq |\mathbf{w}_{fp}^{*T} \Phi_{fp}| + |\mathbf{w}_{gp}^{*T} \Phi_{gp}||u_{al}| + \epsilon_f + \epsilon_g |u_{al}| \tag{59}$$

Outside χ_n, $\kappa(\mathbf{x}) = 0$ so that by equations (10), (24): $f^*(\mathbf{x}) = \hat{f}(\mathbf{x}) = \hat{f}_a(\mathbf{x}) = f_o$ and $g^*(\mathbf{x}) = \hat{g}(\mathbf{x}) = \hat{g}_a(\mathbf{x}) = g_o$. Hence by equation (13) we get:

$$\begin{aligned} \Delta_f &= |f_o - f(\mathbf{x})| \\ \Delta_g &= |g_o - g(\mathbf{x})| \end{aligned} \tag{60}$$

As in [40], assume that the above are bounded by known functions $\overline{f}_o(\mathbf{x}), \overline{g}_o(\mathbf{x})$ such that:

$$|f_o - f(\mathbf{x})| \leq \overline{f}_o(\mathbf{x}) \quad \text{and} \quad |g_o - g(\mathbf{x})| \leq \overline{g}_o(\mathbf{x}) \tag{61}$$

Hence,

$$|d_d| \leq \overline{f}_o + \overline{g}_o |u_{al}| \tag{62}$$

Combining equations (59) and (62), we thus obtain bounds for *both*, inside and outside of χ_n,

$$|d_d| \leq \kappa(\mathbf{x})(|\mathbf{w}_{fp}^{*T} \Phi_{fp}| + |\mathbf{w}_{gp}^{*T} \Phi_{gp}||u_{al}| + (\epsilon_f + \epsilon_g |u_{al}|)) + (1 - \kappa(\mathbf{x}))(\overline{f}_o + \overline{g}_o |u_{al}|) \tag{63}$$

Whilst ϵ_f, ϵ_g, $\overline{f}_o$, $\overline{g}_o$ are assumed to be known and $\kappa(\mathbf{x})$, $|u_{al}(t)|$ could be determined since these are used in the controller; terms $|\mathbf{w}_{fp}^{*T}\Phi_{fp}(\mathbf{x})|$ and $|\mathbf{w}_{gp}^{*T}\Phi_{gp}(\mathbf{x})|$ are not yet known. In fact they depend on the number of passive nodes, which depends, amongst other things, on the *activation threshold* δ_{min}.

Note that

$$|\mathbf{w}_{fp}^{*T}\Phi_{fp}(\mathbf{x})| = |\sum_i w_{fp_i}^{*}\Phi_{fp_i}(\mathbf{x})| \leq \overline{w}_{fb}\sum_i \Phi_{fp_i}(\mathbf{x}) \tag{64}$$

where $\overline{w}_{fb} \geq \max(|w_{fp_i}^{*}|)$ represents a bound on the optimal parameters of the full network. The term $\sum \Phi_{fp_i}$ is indexed by i, whose range is equal to the number of passive nodes, *ie.*, the size of the passive nodes vector Φ_{fp}, denoted by R. This value varies with time, however it is always given by the difference between the total number of mesh points in χ_n, N, and the number of active nodes, A, *ie.*, $R = N - A$.

Thus, the term,

$$\sum_{i=1}^{R}\Phi_{fp_i}(\mathbf{x})$$

is largest when R is a maximum denoted by R_{max}, occurring when A is at its smallest value, A_{min}. This corresponds to the case at time $t = 0$, when only the nodes in one activation hypercube H_1 of length $l\mu$, centred around $\mathbf{x}_o$, the nearest mesh point to $\mathbf{x}(0)$, are activated. This is shown in Figure 3. Recall that l denotes the nearest integer to $\sqrt{-2\sigma^2\ln(\delta_{min})}/\mu$.

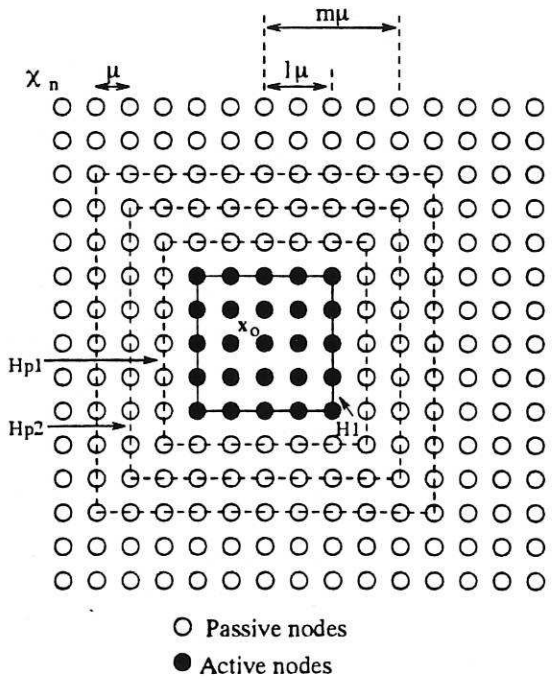

Figure 3: Initial hypercube.

The number of nodes in this hypercube is given, in general by $A_{min} = (1 + 2l)^n$, where n is

the dimension of the state-space. Thus, we may say that

$$\sum_{i=1}^{R} \Phi_{fp_i}(\mathbf{x}(t)) \leq \sum_{i=1}^{R_{max}} \Phi_{fp_i}(\mathbf{x}(0)) \tag{65}$$

The right-hand term reflects a conservative bound for the left-hand term, for all times except at $t = 0$.

As in [33], the passive nodes can be viewed as forming the sides of nested hypercubes, H_{p_i}, surrounding the activation hypercube H_1, as shown by the dashed squares in Figure 3. This is why a hypercube activation zone centred around $\mathbf{x}_o$ was chosen, rather than a ball centred around $\mathbf{x}(t)$. The number of nodes on the perimeter of any hypercube of length $m\mu$ is thus given by $[(2m+1)^n - (2m-1)^n]$. On a given hypercube H_{p_i} of length $m\mu$, the radial basis functions contributing the largest value for an input at $\mathbf{x} = \mathbf{x}_o$, are those located at the mid-point of the sides and the corresponding function will be $\exp\left(\frac{-m^2\mu^2}{2\sigma^2}\right)$. Because all of the other nodes on the perimeter of H_{p_i} contribute less than this value, the summation of all basis functions for these nodes is surely less than $[(2m+1)^n - (2m-1)^n] \exp\left(\frac{-m^2\mu^2}{2\sigma^2}\right)$. Again, this represents a conservative bound. Repeating for all H_{p_i} such that all passive nodes in χ_n are covered, we obtain:

$$\sum_{i=1}^{R} \Phi_{fp_i}(\mathbf{x}(t)) \leq \sum_{i=1}^{R_{max}} \Phi_{fp_i}(\mathbf{x}(0)) \leq \sum_{m=l+1}^{M} [(2m+1)^n - (2m-1)^n] \exp\left(\frac{-m^2\mu^2}{2\sigma^2}\right) \tag{66}$$

where it is assumed that $\Phi_{fp_i}(\mathbf{x}(0)) \approx \Phi_{fp_i}(\mathbf{x}_o)$. This introduces some error but its value is small, especially for a mesh having small μ. In any case, the conservativeness of the bounds introduced previously more than compensates for this small error.

The upper limit of the last summation term reflects the radius (in terms of the number of mesh spacings) of the largest hypercube H_{p_i} required to scan all χ_n. Clearly, this depends on the location of the initial centre $\mathbf{x}_o$. However, being conservative once again, M surely cannot exceed the number of nodes along the longest side of χ_n, so that M can be set to this value. In any case, the series on the right of the last inequality converges rapidly [33], so that the conservativeness of M does not really affect the value of this bound.

As shown in [33], the term $[(2m+1)^n - (2m-1)^n]$ can be expressed via the binomial theorem, as:

$$\sum_{j=0,\ldots,(n-1)|(n-j) \text{ is odd}} 2^{j+1}\, {}^nC_j m^j$$

so that

$$\sum_{i=1}^{R} \Phi_{fp_i} \leq \sum_{j=0,\ldots,(n-1)|(n-j) \text{ is odd}} 2^{j+1}\, {}^nC_j \sum_{m=l+1}^{M} m^j exp\left(\frac{-m^2\mu^2}{2\sigma^2}\right) := \overline{\Phi}_{fb} \tag{67}$$

Similar considerations apply for $\sum \Phi_{gp_i}(\mathbf{x})$, the bound denoted by $\overline{\Phi}_{gb}$.

Whilst [33] provides a relation for determining the bound on the magnitude of the optimal parameters, $|w^*_{fp_i}|$, this is quite conservative, resulting in an unnecessarily larger value for $\overline{k}_a$.

This goes against the aim of keeping the sliding control to its minimum whilst $\mathbf{x} \in \chi_n$. One way of determining the order of magnitude of the maximum value of $|w^*_{fp_i}|$ would be to subject the system to a *short* period of a persistently exciting input [3] [22] [34], so as to force the parameters to a value that is close to their optimal. From these values, one could obtain a hint on the order of magnitude of this bound, even though this would not be perfectly accurate due to the short period of application of the input. In practice, one could then use an overestimate based on this value, so as to ensure that $\max(|w^*_{fpi}|)$ is less than this estimate and use this value for $\overline{w}_{fb}$. Hence,

$$\begin{aligned} |\mathbf{w}^{*T}_{fp}\Phi_{fp}(\mathbf{x})| &\leq \overline{w}_{fb}\overline{\Phi}_{fb} &:= \delta_f \\ &\text{and similarly} & \\ |\mathbf{w}^{*T}_{gp}\Phi_{gp}(\mathbf{x})| &\leq \overline{w}_{gb}\overline{\Phi}_{gb} &:= \delta_g \end{aligned} \tag{68}$$

Thus, equation (63) implies:

$$|d_d| \leq \kappa(\delta_f + \delta_g|u_{al}|) + \kappa(\epsilon_f + \epsilon_g|u_{al}|) + (1-\kappa)(\overline{f}_o + \overline{g}_o|u_{al}|) := \overline{d}_d \tag{69}$$

The first term represent errors due to the fact that the contribution to the output by passive nodes is being ignored in the dynamic network scheme. The second term represents errors due to the fact that the network can only approximate the desired function to a particular accuracy that depends on μ, σ. The last term represents the error when the state falls outside the network approximation region, this having a typically large value making $\overline{d}_d$ large enough to force the state back into χ_n as quickly as possible. Note that whilst the state is in χ_n, $|\overline{d}_d|$ is limited by the size of the first two terms. This is typically much smaller than for the previously-mentioned case, so that use of crude, high-gain sliding control is avoided, keeping only a lower gain sliding control that is sufficient to ensure that parameter drift is avoided, thus guaranteeing robustness.

3.4 Choice of the boundary layer

Boundary layer sliding control is used to limit the bandwidth of the control signal so that high-frequency, unmodelled dynamics are not excited [37], [36]. Following the same line of development as in [40], with appropriate modifications for our dynamic network case, using equation (37) and the fact that the system bandwidth is predominantly determined by the dynamics inside the boundary layer, where $|e_1| \leq \phi$, we get for this range of e_1:

$$\dot{e}_1 = -e_1\left(k_d + g(\mathbf{x})\overline{k}_a\frac{1}{\phi}\right) + h \tag{70}$$

where,

$$h = -\kappa\tilde{\mathbf{w}}^T_{fa}\Phi_{fa}(\mathbf{x}) - \kappa\tilde{\mathbf{w}}^T_{ga}\Phi_{ga}(\mathbf{x})u_{al} + d_d(t) \tag{71}$$

These dynamics have a low-pass filter structure, with corner frequency $\left(k_d + \frac{\overline{k}_a g}{\phi}\right)$, that could be taken as a measure of the control signal bandwidth, ν_c. The smaller the ϕ, the smaller is the tracking error, but the higher is the control bandwidth. This increases the chance of

exciting higher frequency dynamics. For example, if the system is sampled with frequency ν_s, one can only ignore the effects of sampling if $\nu_s > 5\nu_c$ [33]. Hence, a trade-off between keeping a small tracking error and limiting the magnitude of the control bandwidth, must be settled for. If the maximum possible value of ν_c is ν_{max}, then

$$\left(k_d + \frac{\overline{k}_a g}{\phi}\right)_{max} \leq \nu_{max} \tag{72}$$

Substitution for $\overline{k}_a$ implies that:

$$\left(k_d + \frac{\overline{k}_a g}{\phi}\right)_{max} \leq \left(\frac{g\overline{d}_d}{g_l \phi}\right)_{max} \tag{73}$$

so that the above is satisfied if $\left(\frac{g\overline{d}_d}{g_l\phi}\right)_{max} = \nu_{max}$. Substituting for $\overline{d}_d$ and assuming that $\kappa(\mathbf{x}) = 1$ when $|e_1| \leq \phi$, we obtain:

$$\frac{\overline{g}_h}{g_l \phi_{min}} (\delta_f + \delta_g |u_{al}|_{max} + \epsilon_f + \epsilon_g |u_{al}|_{max}) = \nu_{max} \tag{74}$$

where:

- $\overline{g}_h$ is an upper bound on $g_h(\mathbf{x})$, a function such that $g(\mathbf{x}) \leq g_h(\mathbf{x})$, $\forall\ \mathbf{x}$.
- g_l is a lower bound on $g_l(\mathbf{x})$.
- $|u_{al}|_{max} = \frac{\hat{f}_{hm} + \overline{v}}{g_l}$ by definition of u_{al}, where,
  - $\hat{f}_{hm}$ is an upper bound on $\hat{f}_h(\mathbf{x})$, where $\hat{f}_h(\mathbf{x}) \geq |\hat{f}_a(\mathbf{x})|$,
  - $\overline{v}$ is an upper bound on $v(t)$. As in [40], [39], $\overline{v} \approx \overline{y}_d^{(r)}$, the latter being an upper bound on $y_d^{(r)}$.

Hence,

$$\phi_{min} = \frac{g_h}{g_l \nu_{max}} (\delta_f + \delta_g |u_{al}|_{max} + \epsilon_f + \epsilon_g |u_{al}|_{max}) \tag{75}$$

gives the smallest possible value of the boundary layer ϕ, ensuring a control bandwidth that does not exceed ν_{max}. Note that this ϕ_{min} is larger than that derived in [40] where a full-size network has been used. This is due to the additional terms δ_f, $\delta_g |u_{al}|_{max}$ introduced via the error in neglecting the contribution of the passive nodes in our networks. The difference obviously depends on the sizes of δ_f, δ_g which depend principally on the activation threshold δ_{min}. From equation (75), it is straight forward to deduce that ϕ_{min} for a dynamic scheme is greater than that for a full network scheme by a factor of $\frac{|u_{al}|_{max}(\epsilon_g + \delta_g) + \epsilon_f + \delta_f}{|u_{al}|_{max}\epsilon_g + \epsilon_f} \approx \left(1 + \frac{\delta_g}{\epsilon_g}\right)$ in most cases. This implies a larger tracking error for the same ν_{max}, and reflects the price we have to pay in using smaller dynamic networks, for the high frequency unmodelled dynamics not to be excited. Note also, that if a boundary layer less than ϕ_{min} is actually used, the system is still guaranteed to be robust against parameter drift, but one gets a larger control bandwidth.

3.5 Comments

1. The larger the activation threshold, δ_{min}, the smaller is the size of the hypercube in which nodes are activated, resulting in a smaller network. However, this results in a larger $\overline{\Phi}_{fb}$, $\overline{\Phi}_{gb}$ (due to a smaller m in the exponential term of equation (67)) and so larger δ_f, δ_g. In turn, these contribute to a larger $\overline{d}_d$ and hence bigger sliding gains, necessary to compensate for the larger disturbance arising from activation of a smaller number of nodes. Thus, activation threshold δ_{min}, can be viewed as a design parameter that determines the balance between the extent to which the network size is limited and the sliding gain that is required to ensure robustness to the disturbance term. The stronger the limitation on network growth (via large δ_{min}), the larger is the sliding gain; resulting in 'cruder' control having a larger closed-loop bandwidth and higher control activity. In all cases though, error convergence to the boundary layer and robustness to parameter drift are ensured.

2. ϕ appears both in the sliding control term as the width of the boundary layer, and also as the dead-zone width in the adaptation laws. Because of the latter, ϕ affects directly the steady-state tracking error. As in [40], if the effects of high frequency unmodelled dynamics are neglected, the boundary layer and hence the steady-state tracking error, can be made arbitrarily small without risking parameter drift, because of the use of sliding control within χ_n, which compensates for the disturbance term. Indeed as seen in equation (75), if no limitation on the bandwidth were imposed (*i.e.* $\nu_{max} \rightarrow \infty$), ϕ_{min} could be set to zero. This contrasts with those schemes which use simple dead-zone adaptation (without sliding control in χ_n) to avoid parameter drift, where the minimum dead-zone width is determined by the size of the disturbance.

   In practice though, the bandwidth must always be constrained, so that even in our case ϕ_{min} cannot be zero. Note however, that this limitation is imposed by the need to ensure a finite bandwidth and not to ensure robustness to the disturbance term. The argument applies equally well to the activation threshold δ_{min}: the larger it is, the bigger must be the sliding gain and so the bandwidth increases. If no limitation on the bandwidth existed, ϕ_{min} and δ_{min} could take arbitrary values within their ranges, *ie.*, $\phi_{min} \geq 0$, $0 < \delta_{min} \leq 1$. Ideally, ϕ_{min} would be set to zero implying zero steady-state tracking error and δ_{min} close to one to give the greatest possible restriction on dynamic network growth, respectively. This reflects an improvement over the dynamic scheme suggested in [33], whereby to ensure robustness to the disturbance term, the activation threshold that limits network growth depends directly on the tracking accuracy required, because simple dead-zone adaptation is used. Hence, the advantages of using sliding control within χ_n as proposed in [40], [39], are preserved in our dynamic network scheme.

   From the preceding arguments, the steady-state tracking accuracy and the activation threshold are limited only by the need to ensure that the closed-loop system bandwidth is less than ν_{max}, to avoid excitation of high frequency unmodelled dynamics. Thus, in actual design, one must seek a compromise between the steady-state tracking accuracy and the size of the dynamic structure neural network against the control bandwidth.

This is reflected in equation (75), showing the interaction between these three variables via the terms ϕ_{min}, δ_f, δ_g and ν_{max}.

3. In theory, if the entire state-space χ_n is being excited then the dynamic structure network would still grow to a size that is nearly equal to a full network, so that the use of a full non-dynamic network is preferred because of lesser computational effort being required (due to the absence of activation schemes) and the gain in terms of network size will not be significant. In practice though, the proposed dynamic structure network scheme is particularly effective when the state dimensionality $n \geq 2$, because use of a full non-dynamic structure scheme would result in an excessively large network size, a feature often referred to as the *curse of dimensionality*.

3.6 Implementation

For convenience, in this section, the arguments for implementation are developed in terms of the $\hat{f}$ network. Exactly similar considerations apply to the $\hat{g}$ network. The radial basis function network required to approximate $f(\mathbf{x})$ essentially consists of the active parameters vector, $\hat{\mathbf{w}}_{fa}$, storing the output weights and a matrix $\mathbf{M}$, storing the coordinates of the centres of *active nodes*, $\mathbf{m}_{fa_i}$, *ie.*,

$$\mathbf{M} = \begin{bmatrix} \mathbf{m}_{fa_1}^T \\ \mathbf{m}_{fa_2}^T \\ \vdots \end{bmatrix}, \qquad \hat{\mathbf{w}}_{fa} = \begin{bmatrix} \hat{\mathbf{w}}_{fa_1} \\ \hat{\mathbf{w}}_{fa_2} \\ \vdots \end{bmatrix}$$

The length of $\mathbf{M}$ and $\hat{\mathbf{w}}_{fa}$ represents the number of *active* nodes, *ie.*, the network size, and these vectors are *dynamic*, *ie.*, they start from a size of zero and their length increases as nodes are activated. The activation occurs when the system state $\mathbf{x}(t)$ traverses 'new' mesh points in χ_n, whose basis function is currently in a hypercube, centered around the nearest mesh point to $\mathbf{x}(t)$, with length of $\sqrt{-2\sigma^2 ln(\delta_{min})}$.

Thus, when a state $\mathbf{x}(t)$ is observed, mesh nodes with coordinates $\mathbf{m}_j$ in the relevant hypercube must be considered for activation. It is a straight forward procedure to determine these coordinates. However before activating them, *ie*, storing them in $\mathbf{M}$ and augmenting $\hat{\mathbf{w}}_{fa}$ accordingly, one must check if any of these nodes have already been activated before due to state $\mathbf{x}(t)$ passing 'near' them at some previous time. If this is the case, the coordinates of the node would already be stored in $\mathbf{M}$ and clearly, they must not be re-entered. Hence, each mesh node coordinate $\mathbf{m}_j^T$, must be matched with the rows in $\mathbf{M}$ to check for its presence, in which case it represents prior activation. Although this is a straight forward serial search routine, its execution is time consuming, especially when $\mathbf{M}$ grows appreciably large and if the system is to be used on-line.

However, a different technique can be used to check for the presence of $\mathbf{m}_j^T$ in $\mathbf{M}$. The idea centres on using the neural network itself to indicate if $\mathbf{m}_j$ is one of the centres of the current

network basis functions, by temporarily buffering the current network output generating $\hat{f}_a(\mathbf{x})$ and applying consecutively, each coordinate $\mathbf{m}_j$ as input to the network. If one $\mathbf{m}_j$ is already present in $\mathbf{M}$ as the centre of say, the i^{th} active node, then

$$\text{for} \quad \mathbf{m}_j = \mathbf{m}_i, \qquad \Phi_{fi} = \exp\left(\frac{-||\mathbf{m}_j - \mathbf{m}_{fa_i}||^2}{2\sigma_f^2}\right) = 1 \tag{76}$$

If $\mathbf{m}_j$ is not present in $\mathbf{M}$, *none* of the basis functions will be equal to 1. Hence, if one of the basis functions responds with a 1 when the input is $\mathbf{m}_j$, the node centered at $\mathbf{m}_j$ has already been activated and the network need not grow. On the other hand, if *all* the basis functions are less than one, the node centred at $\mathbf{m}_j$ must be activated by augmenting $\mathbf{M}$ with $\mathbf{m}_j^T$ and $\hat{\mathbf{w}}_{fa}$ with 0, resulting in network growth.

The advantage of this method is that given a parallel neural network architecture, detection of the presence of each $\mathbf{m}_j$ in $\mathbf{M}$ takes place in parallel, without having to perform a serial search of all the rows in $\mathbf{M}$ to test for a match with $\mathbf{m}_j^T$. This is achieved by augmenting the usual neural network structure with a Threshold Logic Unit (TLU) connected to the output of each radial basis function unit. The TLU outputs a 1 only if its input is greater than or equal to one. Hence, if one of the basis function outputs is 1, only its own TLU will output a 1 and the rest will be at 0. Performing a logical OR of these TLU outputs will thus result in a logic 1 output only if $\mathbf{m}_j$ is already active and 0 if not. Hence, this output (A) is a binary signal indicating that the network needs to grow (by augmenting it with a node centered at $\mathbf{m}_j$) if A is 0, or need not otherwise. This activation-decision technique shall be referred to as Φ-*based activation* and is shown in figure 4.

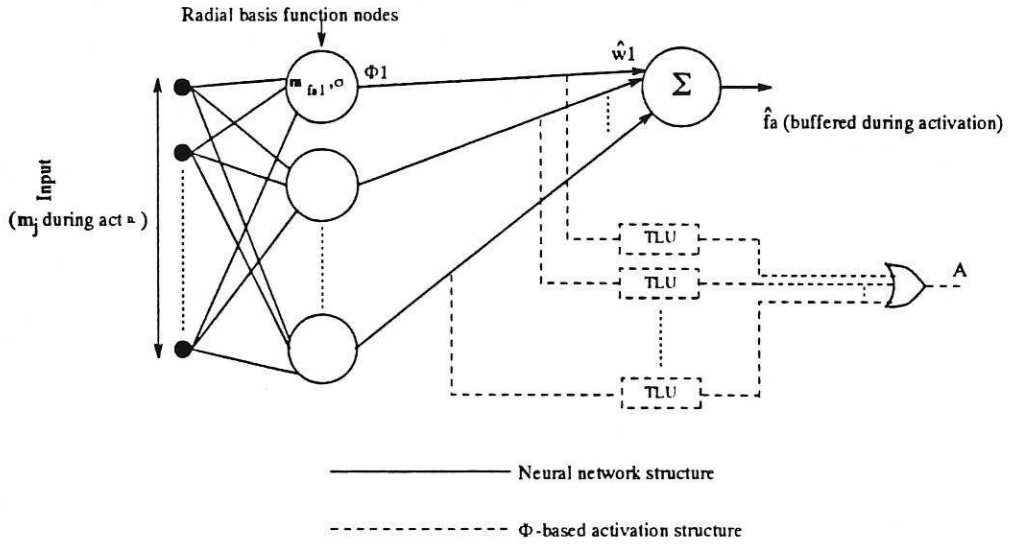

Figure 4: Φ-based activation technique.

4 Simulation and results

Two simulations, based on examples in [40] and [39] are carried out. The examples were chosen to evaluate the effectiveness of selective node activation by comparing the performances of the dynamic and non-dynamic structure networks. In the examples, with orders being 2 and 4 respectively, the input scans a relatively small subset of χ_n and hence a significant savings in terms of the size of network would be expected. What is of interest also is the tracking performances for the two systems in the face of parsimony in neural network basis functions.

4.1 Simulation 1

The example used in the first simulation is taken from [39]. The dynamical system in this example is given by the equation,

$$\begin{aligned} \dot{x}_1 &= f(\mathbf{x}) + g(\mathbf{x})u \\ \dot{x}_2 &= x_1 - x_2 \\ y &= x_1 \end{aligned} \tag{77}$$

where,

$$\begin{aligned} \mathbf{x} &= [x_1\ x_2]^T \\ f(\mathbf{x}) &= \cos(7(x_1^2 + x_2^2))\exp(-(x_1^2 + x_2^2)) \\ g(\mathbf{x}) &= 2 + \cos(7x_1x_2) \end{aligned} \tag{78}$$

This system is of order $n = 2$ and has degree $r = 1$. These dynamics are actually already expressed in global normal form and $\mathbf{I}(x_1, x2) = x_1 - x_2$ has continuous and bounded partial derivatives in x_1, x_2. The zero dynamics, $\dot{x}_2 = \mathbf{I}(0, x_2)$ are thus given by $\dot{x}_2 = -x_2$, which are globally exponentially stable. The reference input, y_d, is obtained by filtering a zero-average, 0.9 amplitude, 0.4 Hz square wave by a filter $1/\left(1 + \frac{s}{10}\right)^3$, so that the derivative of the filter output is bounded. As shown in [39], for the given system and reference input, the desired state is bounded well within the interval $[-1,\ 1]\times[-1,\ 1]$ (along x_1, x_2 respectively). To cater for the fact that during the transient period the actual state may overshoot these bounds, the network approximation region is taken to be larger, namely $\chi_n = [-1.5,\ 1.5] \times [-1.5,\ 1.5]$. This ensures that the state is bounded within the approximation region, thus avoiding the use of crude, high-gain sliding control. χ_n^-, by definition, must be a subset of χ_n so that it is set to $\chi_n^- = [-1,\ 1] \times [-1,\ 1]$.

It is assumed that the known prior estimates to the functions being approximated are $f_o = 0$, $g_o = 2$ and as shown in [39], full network inherent approximation error bounds $\epsilon_f = \epsilon_g = 0.005$ would be obtained via basis functions having a standard deviation σ_f, σ_g of 0.03 located on a mesh of spacing $\mu = 0.05$ within χ_n. Bounds $\overline{f}_o$, $\overline{g}_o$ are set to 2 and $g_l(\mathbf{x})$ is set equal to $g_l = 0.895$. The activation threshold is set to 0.2, resulting in an activation hypercube of length μ.

The control law is given by $v(t) = \dot{y}_d - 5(y - y_d)$ and $e_1(t) = e(t)$. η_f, η_g are set to 25 and ϵ^* to 0.1. Using equation (67) and the above values, results in $\overline{\Phi}_{fb} = \overline{\Phi}_{gb} = 0.062$. The system was initially subjected to a 40s persistently exciting input to obtain the order of magnitude for bounds of the optimal parameters, from which over-estimates $\overline{w}_{fb} = \overline{w}_{gb} = 1.6$ were deduced. Using equations (68), we obtain $\delta_f = \delta_g = 0.1$. Using $|v| = |y_d^{(1)}| \leq 5$, $\hat{f}_{hm} = 2$, $\overline{g}_h = 4$, $\nu_s = 2000\pi\ rad/s$, $\nu_{max} = 400\ rad/s$, gives $\phi_{min} \approx 0.01$ which was used as the boundary layer. Note that the persistently exciting input is used to obtain δ_f, δ_g only; once these values are deduced, network activation is started afresh, with initially no nodes being activated.

The results of the simulation are shown in figure 5. Note that the system is stable and that the error asymptotically converges to the expected value of 0.01, namely the size of the boundary layer. Moreover, after 150 seconds, the networks contained 556 nodes each, which is 15% of the 'full-size' networks having 3721 nodes used in [39]. This follows from the fact that the desired input forces the state to follow a subset of the space in χ_n, and thus the selective node activation technique places basis functions in the neighbourhood of this subset only. This can be seen in Figure 6 which shows χ_n^-, the region where the state trajectory is contained, marking the positions of active nodes forming the dynamic networks with a cross and those of passive nodes, which were present in a 'full-size' network scheme, with a dot.

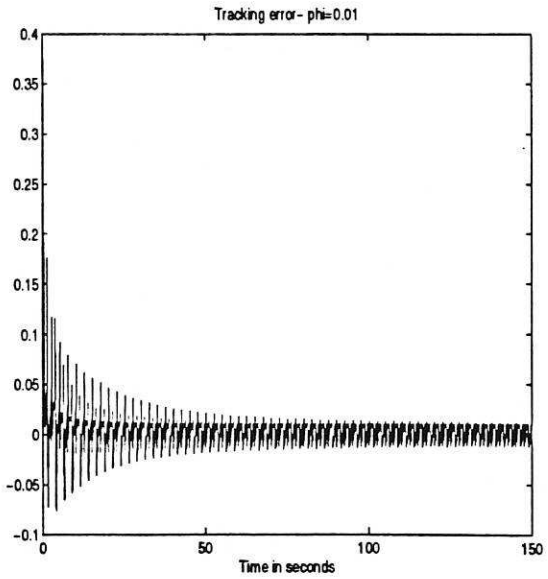

Figure 5: Simulation 1: Tracking error.

Moreover, as demonstrated in [40], even if a full-size network were used, the parameter vectors $\hat{\mathbf{w}}_f$, $\hat{\mathbf{w}}_g$ would only approach their optimal values w_f^*, w_g^* if Φ_f, $\Phi_g u_{al}$ are persistently excited. Otherwise, the network approximation accuracy is good only in state space regions visited during the operation of the system. This is the case for the dynamic structure network scheme as well. The role of the neural network is in learning the system dynamics while the

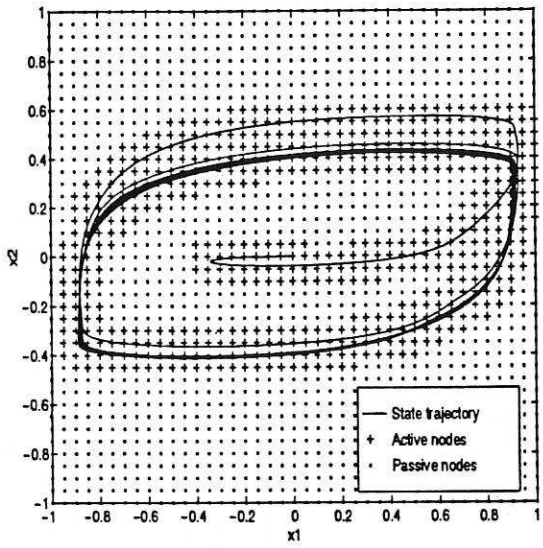

Figure 6: Simulation 1: Activated nodes.

adaptive control scheme guarantees stability and robustness. The advantage of using neural networks to learn the dynamics of the system are demonstrated in [39] in which it is shown for the dynamic system used in this simulation, the tracking errors are about a factor 20 higher for the scheme that does not use the neural network adaptive controller.

4.2 Simulation 2

The system for this second example, taken from [40], is a robotic manipulator having 2 degrees of freedom in a horizontal plane, as shown in figure 7.

The manipulator can rotate about axis O by application of torque u_2 which controls angle y_2 and move radially by application of force u_1, controlling the radial distance y_1. C is the centre of mass of the link, M_c is the link mass, M is the load mass and a is the distance from C to the load. J_1, J_2 are the moments of inertia of the link with respect to C and O respectively. The system dynamics are given by,

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \frac{((M_c+M)x_1+Ma)x_4^2}{M_c+M} + \frac{u_1}{M_c+M} \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= \frac{-2[(M_c+M)x_1+Ma]x_2x_4}{J_1+J_2+M_cx_1^2+M(x_1+a)^2} + \frac{u_2}{J_1+J_2+M_cx_1^2+M(x_1+a)^2} \\
y_1 &= x_1 \\
y_2 &= x_3
\end{aligned} \tag{79}$$

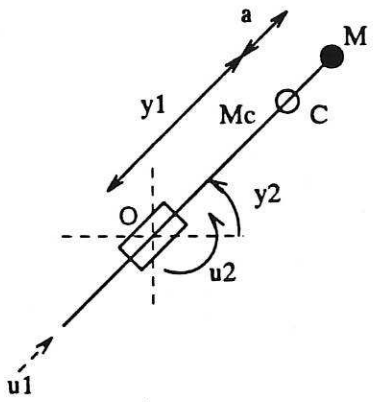

Figure 7: Two degrees of freedom robotic manipulator.

which could be re-written in terms of the radial and angular dynamics respectively, as,

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= f_1(x_1, x_4) + g_1 u_1 \\ y_1 &= x_1 \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= f_2(x_1, x_2, x_3) + g_2(x_1) u_2 \\ y_2 &= x_3 \end{aligned} \tag{80}$$

so that the unknown dynamics are given by,

$$\begin{aligned} f_1(x_1, x_4) &= \frac{((M_c+M)x_1 + Ma)x_4^2}{M_c+M} \\ g_1 &= \frac{1}{M_c+M} \\ f_2(x_1, x_2, x_3) &= \frac{-2[(M_c+M)x_1 + Ma]x_2 x_4}{J_1+J_2+M_c x_1^2 + M(x_1+a)^2} \\ g_2(x_1) &= \frac{1}{J_1+J_2+M_c x_1^2 + M(x_1+a)^2} \end{aligned} \tag{81}$$

Functions f_1, f_2, g_1, g_2 represent the 'unknown' system dynamics. Note that the two subsystems concerning radial movement (y_1) and angular movement (y_2), can be considered as two separate SISO affine systems, since the control gain matrix has a diagonal structure. Hence, two separate controllers are used, one for each subsystem.

The desired output is given by,

$$\begin{aligned} y_{1d} &= 1 + 0.5\sin(2t) \\ y_{2d} &= \tfrac{\pi}{4}(1 + \cos(2t)) \end{aligned} \tag{82}$$

so that the network approximation region, χ_n, is given by the interval $[0.3,\ 1.7] \times [-1.2,\ 1.2] \times [-0.1\pi,\ 0.6\pi] \times [-0.6\pi,\ 0.6\pi]$ along x_1, x_2, x_3, x_4 respectively. The prior estimates for the $\hat{f}_1$, $\hat{f}_2$ functions are 0 and those for the $\hat{g}_1$, $\hat{g}_2$ networks are 4.5×10^{-3} and 1×10^{-3}, respectively.

As in [40], full network inherent approximation error bounds ϵ_{f1}, ϵ_{f2}, ϵ_{g1}, ϵ_{g2} (for the networks approximating f_1, f_2, g_1, g_2 respectively) equal to 0.1, 0.05, 2×10^{-5}, 1×10^{-5}

would be obtained by basis functions having standard deviations of 0.06, 0.09, 0.14 with respect to states x_1, x_2, x_4 for the $\hat{f}_1$, $\hat{f}_2$ functions, and 0.024 for the $\hat{g}_1$, $\hat{g}_2$ functions if mesh spacing of 0.1, 0.15, 0.23 in the directions of x_1, x_2, x_4 for the $\hat{f}_1$, $\hat{f}_2$ networks and 0.04 for the $\hat{g}_1$, $\hat{g}_2$ networks along x_1 are used. Since the $\hat{g}$ networks have the same input space, mesh spacing and standard deviation they could share the same nodes as that of $\hat{f}$. As mentioned before, not all of these basis functions are necessarily used in the dynamic scheme, and the mesh positions just indicate where a basis function could be placed if required.

The lower bounds on $\hat{g}_1$, $\hat{g}_2$ are set at $g_{l1} = 4.48 \times 10^{-3}$ and $g_{l2} = 8.4 \times 10^{-4}$, respectively. The activation threshold is set to 0.2. The control law is given by $\Psi(s) = (s + 5)$ and k_d is set to 1 in the transfer function $\Psi\Gamma^{-1}$. η_{f1}, η_{f2}, η_{g1}, η_{g2} are set to 20, 20, 5×10^{-5}, 5×10^{-5} respectively and ϵ^* to 1×10^{-4}. Assuming that the functions are bounded by $\overline{g}_{h1} = 12.5 \times 10^{-3}$, $\overline{f}_{h1} = 5.18$, $\overline{g}_{h2} = 5.6 \times 10^{-3}$, $\overline{f}_{h2} = 6.2$, with the initial approximations used, the bounds follow as $\overline{f}_{o1} = 5.18$, $\overline{f}_{o2} = 6.2$, $\overline{g}_{o1} = 8 \times 10^{-3}$, $\overline{g}_{o2} = 5 \times 10^{-3}$ for $\hat{f}_1$, $\hat{f}_2$, $\hat{g}_1$, $\hat{g}_2$ respectively. From these values and use of equation (67) adapted accordingly to accommodate for the fact that the mesh spacing and standard deviations are different along x_1, x_2, x_4, one obtains: $\overline{\Phi}_{f1b} = 0.065$, $\overline{\Phi}_{f2b} = 0.392$, $\overline{\Phi}_{g1b} = \overline{\Phi}_{g2b} = 0.0077$. Using overestimates for bounds of the optimal parameters as: $\overline{w}_{f1b} = 2$, $\overline{w}_{f2b} = 2$, $\overline{w}_{g1b} = 0.007$, $\overline{w}_{g2b} = 0.004$, one obtains $\delta_{f_1} = 0.13$, $\delta_{f_2} = 0.78$, $\delta_{g_1} = 5.4 \times 10^{-5}$, $\delta_{g_2} = 3.1 \times 10^{-5}$ for the $\hat{f}_1$, $\hat{f}_2$, $\hat{g}_1$, $\hat{g}_2$ networks, respectively. Using $\overline{y}_{1d}^{(r)} = 2$, $\overline{y}_{2d}^{(r)} = \pi$ (where for these dynamics $r = 2$), sampling frequency $\nu_s = 1000\pi$ *rad/s*, $\nu_{max} = 130$ *rad/s*; one obtains $\phi_{min1} = 0.0075$, $\phi_{min2} = 0.066$. Hence, boundary layers set to $\phi = 0.015$ and $\phi = 0.08$ were used for the radial and angular subsystems, respectively. Using equation (58), these should result in steady-state error bounds of 0.003 metres and 0.016 radians respectively. The final design parameters are summarised in Table 1 for convenience.

Subsystem	f_o	g_o	η_f	η_g	ϵ_f	ϵ_g	σ_f	σ_g
Radial	0.06,0.09,0.14 0	0.024 4.5×10^{-3}	20	5×10^{-5}	0.1	2×10^{-5}	0.06 (x_1) 0.09 (x_2) 0.14 (x_4)	0.024
Angular	0	1.0×10^{-3}	20	5×10^{-5}	0.05	1×10^{-5}	0.06 (x_1) 0.09 (x_2) 0.14 (x_4)	0.024
Subsystem	δ_f	δ_g	ϕ_{min}	g_l	$\overline{f}_o$	$\overline{g}_o$	μ_f	μ_g
Radial	0.13	5.4×10^{-5}	0.0075	4.48×10^{-3}	5.18	8×10^{-3}	0.1 (x_1) 0.15 (x_2) 0.23 (x_4)	0.04
Angular	0.78	3.1×10^{-5}	0.066	8.4×10^{-4}	6.2	5×10^{-3}	0.1 (x_1) 0.15 (x_2) 0.23 (x_4)	0.04

Table 1: Design parameters for $\nu_s = 500\ Hz$, $\nu_{max} = 130\ rad/s$, $\delta_{min} = 0.2$

The results of the simulation are shown in Figure 8. Parts (a) and (b) of Figure 8 show the radial and angular tracking errors respectively. Note that during the 100s simulation time, no instability was exhibited and the errors remained bounded. Furthermore, parts (c) and (d) of the same figure show that the radial and angular tracking errors converge to within their expected bounds in the steady-state. During the given simulation interval, the network parameters remained bounded, indicating robustness to parameter drift. For the given desired

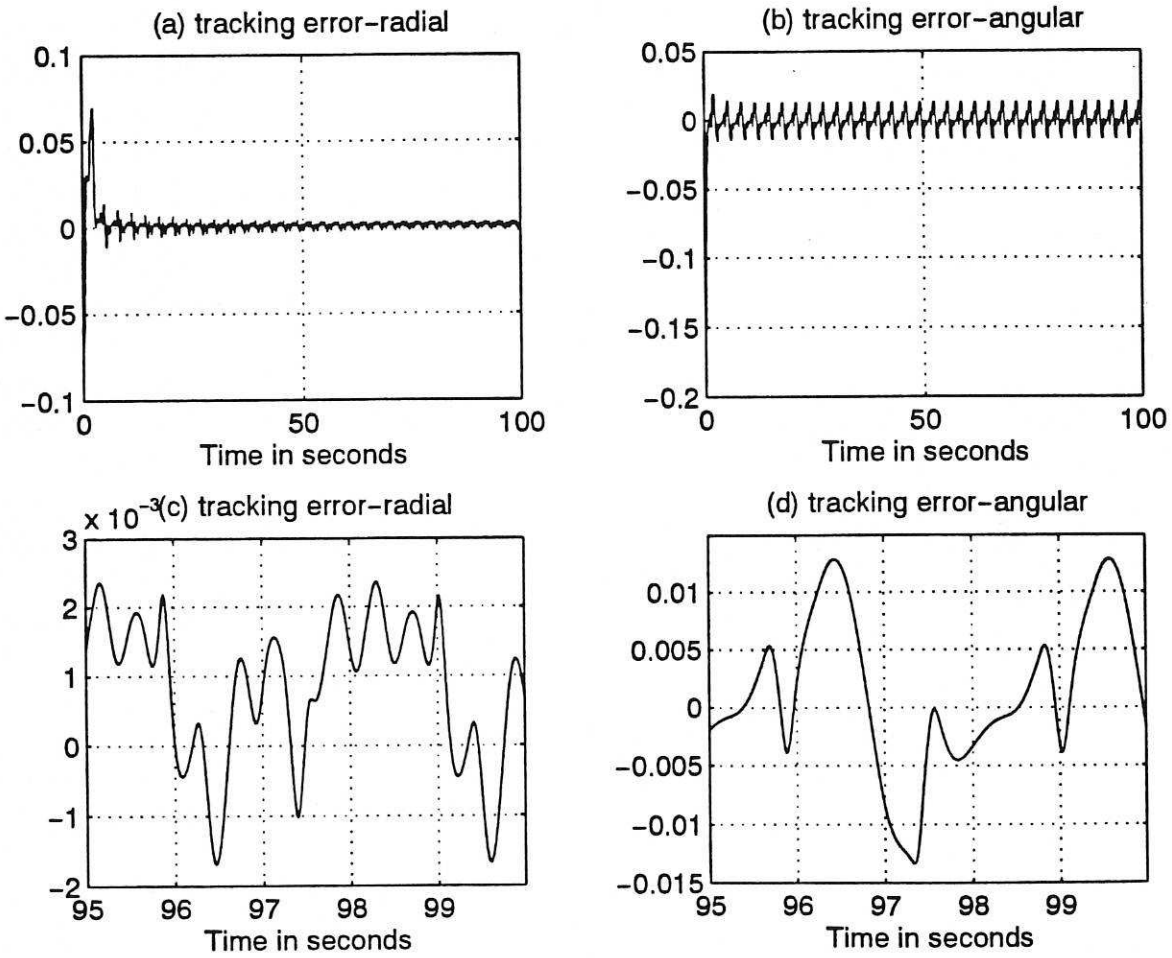

Figure 8: Simulation 2: Tracking error.

input, the size of the networks grew up to 89, 828 and 29 nodes for the $\hat{f}_1$, $\hat{f}_2$ and $\hat{g}$ networks, respectively. These represent a usage of 33%, 18% and 80.5% respectively, when compared to the corresponding non-dynamic structure (full-size) network scheme, demonstrating the effectiveness of the scheme. Note that the greatest saving is on the $\hat{f}_2$ network, since its input is 3 dimensional, and thus the active region represents a much smaller proportion of the complete approximation region, than for say, $\hat{g}$ networks whose nodes scan a one dimensional space. The tracking performance confirm that there were no performance degradation due to the use of the dynamic structure in preference to the non-dynamic one.

5 Conclusion

A method, based on Lyapunov stability considerations, for using dynamic structure Gaussian radial basis function neural networks for adaptive control of affine nonlinear systems has been presented. The objective of the control scheme is to achieve good tracking performance in the absence of known system dynamics. The scheme is also expected to be robust against parameter drift and approximation errors. The task is carried out by using dynamic structure neural networks to approximate the underlying functions and generate control signals based on the feedback linearisation control law. The work presented is based on the system developed in [40], [39], which was extended and modified to be suitable for the dynamic structure neural networks, an idea suggested in [33] for adaptive control, but had been developed for system identification [27], [12].

The scope behind the use of dynamic structure networks is to keep the size of the neural networks limited, by ignoring potential basis functions that would be centred on mesh points located beyond a chosen distance from the active regions of state-space. The result is a network that 'grows' whilst the system is evolving, by activating basis functions that are centred on mesh points in appropriate locations. The adaptation laws that adjust the parameters of the neural networks are so designed to guarantee stability in the sense of Lyapunov.

The system is augmented by low-gain sliding control (as in [40]) and by high-gain sliding control (as in [33]) to ensure global stability of the tracking error and robustness to the presence of a disturbance term. The latter arises because of the network inherent approximation errors and those basis functions that are ignored in the dynamic structure scheme. This method allows for a certain degree of flexibility regarding the choice of the tracking accuracy and the node activation threshold; a flexibility that is limited only by the maximum allowed control bandwidth. A technique for the efficient implementation of the dynamic structure network within the adaptive control scheme that avoids time involving serial search by exploiting the parallelism of the neural networks, is also developed.

The effectiveness of the dynamic structure scheme was put to test with two examples, used in [40], [39]. The performance results demonstrate that the scheme performs as predicted, by forming network sizes that are compact while achieving equally good tracking accuracy, evidenced by convergence to the predicted bounds, compared to the non-dynamic structure full-sized network scheme. The computational savings were very high where the order of the system was two or higher and where the input to the system was not persistently exciting over the entire network approximation region. The significant computational advantages offered by this scheme, with guaranteed stability and robustness to disturbances enable the use of neural networks for adaptive control of a variety of physical systems in practice. In summary, a dynamic structure neural network stable adaptive controller scheme has been derived for nonlinear systems and its advantages demonstrated in two experiments.

References

[1] C. W. Anderson. "Q–Learning with hidden-unit restarting". In *Advances in Neural Information Processing Systems 5*, Morgan Kaufmann, CA: San Mateo, 1993.

[2] P. J. Antsaklis (Ed). Special Issue on Neural Networks in Control Systems, *IEEE Control Systems Magazine*, Vol.10, No.3, April 1990.

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

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

[5] G. Cybenko. "Approximations by superpositions of a sigmoidal function", *Mathematics of Control, Signals and Systems*, Vol.2, pp.303-314, 1989.

[6] B. Egardt. *Stability of Adaptive Controllers.* Springer-Verlag, Berlin, 1979.

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

[8] J. Hertz, A. Krogh and R. G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley Publishing Co., 1991.

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

[10] P. A. Ioannou and A. Datta. "Robust Adaptive Control: Design, Analysis and Robustness Bounds", In P. V. Kokotovic (Ed.), *Foundations of Adaptive Control*, Springer-Verlag, Berlin, pp.71-152, 1991.

[11] A. Isidori. *Nonlinear Control Systems: An Introduction.* Springer-Verlag, Berlin, 1989.

[12] V. Kadirkamanathan. *Sequential Learning in Artificial Neural Networks.* PhD Thesis, Cambridge University Engineering Department, Cambridge, UK, September 1991.

[13] V. Kadirkamanathan. "A statistical inference based growth criterion for the RBF network", In *Proceedings of the IEEE Workshop in Neural Networks for Signal Processing IV*, pp.12-21, 1994.

[14] V. Kadirkamanathan and M. Niranjan. "Application of an architecturally dynamic network for speech pattern classification", In *Proceedings of the Institute of Acoustics*, Vol.14, Part 6, pp.343-350, 1992.

[15] V. Kadirkamanathan and M. Niranjan. "A function estimation approach to sequential learning with neural networks", *Neural Computation*, Vol.5, pp.954-975, 1993.

[16] A. Karakasoglu, S. L. Sudharsanan and M. K. Sundareshan. "Identification and decentralized adaptive control using neural networks with application to robotic manipulators", *IEEE Transactions on Neural Networks*, Vol.4, No.6, pp.919-930, 1993.

[17] P. V. Kokotovic, I. Kanellakopoulos and A. S. Morse. "Adaptive Feedback Linearization of Nonlinear Systems", In P. V. Kokotovic (Ed.), *Foundations of Adaptive Control*, Springer-Verlag, Berlin, pp.311-346, 1991.

[18] Y. D. Landau. *Adaptive Control: The Model Reference Approach.* Marcel Dekker Inc., New York, 1979.

[19] A. Levin and K. S. Narendra. "Control of nonlinear dynamical systems using neural networks: Controllability and stabilization", *IEEE Transactions on Neural Networks*, Vol.4, No.2, pp.192-206, 1993.

[20] D. E. Rumelhart, G. E. Hinton and Williams. "Learning internal representations by error propagation", In J. L. McClelland and D. E. Rumelhart, (Eds). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Bradford Books/MIT Press, Cambridge, MA, Vol.1, pp.318-361, 1986.

[21] W. T. Miller, R. S. Sutton and P. J. Werbos, (Eds). *Neural Networks for Control.* MIT Press, Cambridge, Massachusetts, 1990.

[22] K. S. Narendra and A. M. Annaswamy. *Stable Adaptive Systems.* Prentice-Hall, Englewood Cliffs, New Jersey 1989.

[23] K. S. Narendra and A. M. Annaswamy. "Robust adaptive control", In K. S. Narendra (Ed.), *Adaptive and Learning Systems: Theory and Applications*, Plenum Press, New York, pp.3-31, 1986.

[24] K. S. Narendra and K. Parthasarathy. "Identification and control of dynamical systems using neural networks", *IEEE Transactions on Neural Networks*, Vol.1, No.1, pp.4-27, 1990.

[25] H. Nijmeijer and A. J. Van Der Schaft. *Nonlinear Dynamical Control Systems.* Springer-Verlag, New York, 1990.

[26] B. B. Peterson and K. S. Narendra. "Bounded error adaptive control", *IEEE Transactions on Automatic Control*, Vol.27, No.6, pp.1161-1168, December 1982.

[27] J. C. Platt. "A resource allocating network for function interpolation", *Neural Computation*, Vol.3, No.2, pp.213-225, 1991.

[28] T. Poggio and F. Girosi. "Networks for Approximation and Learning", *Proceedings of the IEEE*, Vol.78, No.9, September 1990.

[29] M. Polycarpou and P. Ioannou. "Identification and Control of Nonlinear Systems Using Neural Network Models:Design and Stability Analysis", Technical Report 91-09-01, Department of Electrical Engineering-Systems, University of Southern California, Los Angeles, USA, September 1991.

[30] M. J. D. Powell. "Radial basis functions for multivariable interpolation: A review", In J. C. Mason and M. G. Cox (Eds.), *Algorithms for Approximation*, Oxford University Press, Oxford, pp.143-167, 1987.

[31] D. Psaltis, A. Sideris and A. A. Yamamura. "A multilayered neural network controller", *IEEE Control Systems Magazine*, Vol.8, pp.17-21, 1988.

[32] N. Sadegh. "A perceptron network for functional identification and control of nonlinear systems", *IEEE Transactions on Neural Networks*, Vol.4, No.6, pp.982-988, 1993.

[33] R. M. Sanner and J. J. E. Slotine. "Gaussian Networks for Direct Adaptive Control", *IEEE Transactions on Neural Networks*, Vol.3, No.6, November 1992.

[34] S. Sastry and M. Bodson. *Adaptive Control:Stability, Convergence and Robustness.* Prentice-Hall International, Englewood Cliffs, New Jersey 1989.

[35] S. Sastry and A. Isidori. "Adaptive Control of Linearizable Systems", *IEEE Transactions on Automatic Control*, Vol.34, No.11, pp.1123-1131, November 1989.

[36] J. J. E. Slotine and W. Li. *Applied Nonlinear Control.* Prentice-Hall International, Englewood Cliffs, New Jersey 1991.

[37] J. J. Slotine and S. S. Sastry. "Tracking control of non-linear systems using sliding surfaces, with application to robot manipulators", *International Journal of Control*, Vol.38, No.2, pp.465-492, 1983.

[38] A. Teel, R. Kadiyala, P. Kokotovic and S. Sastry. "Indirect techniques for adaptive input-output linearization of non-linear systems", *International Journal of Control*, Vol.53, pp.193-222, 1991.

[39] E. Tzirkel-Hancock and F. Fallside. "Stable Control of Nonlinear Systems using Neural Networks", *International Journal of Robust and Nonlinear Control*, Vol.2, pp.63-86, May 1992.

[40] E. Tzirkel-Hancock. *Stable Control of Nonlinear Systems using Neural Networks.* PhD Thesis, Cambridge University Engineering Department, Cambridge, UK, June 1992.

[41] V. I. Utkin. *Sliding Modes and their Application in Variable Structure Systems.* MIR Publishers, Moscow, 1978.

[42] V. I. Utkin. "Sliding mode and its applications to variable structure systems", *IEEE Transactions on Automatic Control*, Vol.22, No.2, pp.212-222, April 1977.