2005 American Control Conference
June 8-10, 2005. Portland, OR, USA

Getting Weights to Behave Themselves: Achieving Stability and Performance in Neural-Adaptive Control When Inputs Oscillate

C.J.B. Macnab

***Abstract*— Local basis functions offer computational efficiency when used in nonlinear adaptive control schemes. However, commonly used robust weight (parameter) update methods do not result in acceptable performance when applied to underdamped systems. This is because persistent oscillation in the inputs encourages severe weight drift, in turn requiring large robust terms that significantly limit the performance. In particular, the methods of leakage, e-modification, deadzone, and weight projection sacrifice performance to halt this weight drift. In contrast, it is observed (in simulations) that application of the proposed method halts the weight drift without sacrificing the performance.**

***Index Terms*— direct adaptive control, nonlinear approximate control, neural network control, fuzzy control, Lyapunov stability, local basis functions, β-splines, CMAC**

I. Introduction

THE methods of direct adaptive control have been used to derive stable nonlinear approximate adaptive controls, where a weighted sums of basis functions are used to model the nonlinearities. Example basis functions used in such schemes include β-splines [1], radial basis functions [2], multilayer neural networks [3], the Cerebellar Model Articulation Controller (CMAC) [4], and fuzzy sets [5]. Others have emphasized the common framework of these schemes, including [6], [7], [8]. The types of basis functions used can be classified as either global or local. Local basis functions have a value of zero at some distance away from their centers, and examples include β-splines, fuzzy triangular membership functions, and the CMAC. Normally hypercube areas are covered by each function so that table look-up methods can be used for indexing. In this paper, the non-zero area of the basis function will be referred to as a *cell*. The advantage to using local basis functions is that only *activated* (indexed) cells need to have their functions computed, making on-line computation much more efficient. However, the literature is devoid of examples where local basis functions are used in direct adaptive control of systems that exhibit persistent oscillations. In this case, large jumps in error can occur unexpectedly, due to the drift of the weights to large magnitudes. The traditional robust methods used to deal with this problem of *weight drift* include leakage [9], deadzone [10], e-modification [11] and parameter projection [10] and these may work well for global basis functions (assuming a large number of very wide functions). However, for local basis functions the weight drift is much more severe due to oscillations between local cells: two cells that are beside each other can have weights drift toward infinity and negative infinity respectively. The aforementioned robust methods result in very limited performance when attempting to halt this kind of severe weight drift.

In this paper an alternative to the standard robust methods is proposed that addresses the problem identified above. This method takes advantage of one of the proprieties of neural networks: many different sets of weights are capable of uniformly approximating the same nonlinear function. The general idea in this approach is to find an alternate set of weights that approximates the function, and to try and force the weights used in the control (*control weights*) toward the finite alternative rather than to zero (as in leakage). Each *alternate weight* is continuously updated and, in turn, kept close to a weight that has been identified as the best choice found so far, as determined by a local cost-functional. One of these *choice weights* may only be changed when its corresponding cell is deactivated. It is shown through simulation that this way of eliminating the weight drift does not result in a dramatic loss in performance.

The motivation for this research comes from CMAC direct adaptive control of flexible-joint robots. It is observed that the error jumps to very large values (appearing to go unstable) as soon as the highest level of performance is reached. The solution proposed in this paper is presented using the simplest possible simulation; two local basis functions where the disturbance, instead of the nonlinearity, is specified. The purpose is examine this problem, and its proposed solution, in the simplest most generic manner possible. The intention is to find fundamental understanding and thus wider application of the technique. The solution has also been successfully applied to a flexible-joint robot model, but those results are beyond the scope of this paper.

II. Background on robust methods

Consider that n nonlinearities in $\mathbf{f}(\mathbf{x}) \in \mathcal{R}^n$ are approximated using weighted basis functions. Each nonlinearity is approximated with m weights multiplying n-dimensional basis functions. The basis functions are arranged in the following matrix

$$\boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} \boldsymbol{\phi}_1(\mathbf{x}) & 0 & \dots & 0 \\ 0 & \boldsymbol{\phi}_2(\mathbf{x}) & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & \boldsymbol{\phi}_n(\mathbf{x}) \end{bmatrix}, \quad (1)$$

This work was supported by the Alberta Ingenuity Fund.

C.J.B. Macnab is with Department of Electrical and Computer Engineering, University of Calgary, Alberta, Canada, cmacnab@ucalgary.ca.

where $\boldsymbol{\phi}(\mathbf{x}) \in \mathcal{R}^{n\times nm}$ and for each $i = 1 \dots n$ there is a row vector of basis functions

$$\boldsymbol{\phi}_i(\mathbf{x}) = \left[\begin{array}{cccc} \phi_{i,1}(\mathbf{x}) & \phi_{i,2}(\mathbf{x}) & \dots & \phi_{i,m}(\mathbf{x}) \end{array} \right]. \quad (2)$$

If each row of basis functions is the same, $\boldsymbol{\phi}_i(\mathbf{x}) = \boldsymbol{\phi}_j(\mathbf{x})$ for $i, j = 1, .., n$, this describes one basis function network with n outputs. Consider an ideal (constant) set of weights in column vector

$$\mathbf{w} = \left[\begin{array}{cccc} w_1 & w_2 & \dots & w_{nm} \end{array} \right]^T. \quad (3)$$

The outputs of the (ideal) basis function network are given by $\boldsymbol{\phi}(\mathbf{x})\mathbf{w}$. Note that in the case of $n = 1$ there is a scalar output. Consider the local region $\mathcal{D} \in \mathcal{R}^n$ where the network has basis functions. The functions in $\boldsymbol{\phi}$ will qualify as basis functions if, given large enough m, $\mathbf{f}(\mathbf{x})$ can be approximated

$$\mathbf{f}(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})\mathbf{w} + \mathbf{d} \qquad \forall \mathbf{x} \in \mathcal{D}, \quad (4)$$

where the modeling error $\mathbf{d}$ is bounded $\|\mathbf{d}\| < d_{\max}$ *i.e.* the basis functions can uniformly approximate $\mathbf{f}(\mathbf{x}) \in \mathcal{D}$.

The error between the ideal weights $\mathbf{w}$ and the weight estimates $\hat{\mathbf{w}}$ (actual weights) is denoted

$$\tilde{\mathbf{w}} = \mathbf{w} - \hat{\mathbf{w}}. \quad (5)$$

Then to find a stabilizing control for the nonlinear system with state $\mathbf{x} \in \mathcal{R}^n$ represented by

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{u}, \quad (6)$$

where $\mathbf{u} \in \mathcal{R}^n$ are control inputs, choose (adaptive control) Lyapunov function

$$V = \frac{1}{2}\mathbf{x}^T\mathbf{x} + \frac{1}{2\beta}\tilde{\mathbf{w}}^T\tilde{\mathbf{w}}, \quad (7)$$

where β is a positive constant (learning gain). Then taking the derivative

$$\dot{V} = \mathbf{x}^T(\mathbf{f}(\mathbf{x}) + \mathbf{u}) - \frac{1}{\beta}\tilde{\mathbf{w}}^T\dot{\hat{\mathbf{w}}}, \quad (8)$$

$$= \mathbf{x}^T(\boldsymbol{\phi}\mathbf{w} + \mathbf{d} + \mathbf{u}) - \frac{1}{\beta}\tilde{\mathbf{w}}^T\dot{\hat{\mathbf{w}}}, \quad (9)$$

and choosing control

$$\mathbf{u} = -\boldsymbol{\phi}\hat{\mathbf{w}} - \mathbf{G}\mathbf{x}, \quad (10)$$

with $\mathbf{G}$ a positive-definite matrix of control gains results in

$$\dot{V} = \mathbf{x}^T(\mathbf{d} - \mathbf{G}\mathbf{x}) + \tilde{\mathbf{w}}^T(\boldsymbol{\phi}^T\mathbf{x} - \frac{1}{\beta}\dot{\hat{\mathbf{w}}}). \quad (11)$$

The question is how to update the weights, keeping the system Lyapunov-stable in spite of (bounded) disturbance $\mathbf{d}$.

A. Non-robust update

The weight update

$$\dot{\hat{\mathbf{w}}} = \beta\boldsymbol{\phi}^T\mathbf{x} \quad (12)$$

would result in Lyapunov derivative

$$\dot{V} = -\mathbf{x}^T\mathbf{G}\mathbf{x} + \mathbf{x}\mathbf{d}, \quad (13)$$

$$\leq \|\mathbf{x}\|(-\|\mathbf{x}\|\lambda_{\min}(\mathbf{G}) + d_{\max}), \quad (14)$$

where $\lambda_{\min}(\mathbf{G})$ is the minimum singular value of $\mathbf{G}$. In this case the weights could drift freely to infinite magnitudes when

$$\|\mathbf{x}\| < \frac{d_{\max}}{\lambda_{\min}(\mathbf{G})}. \quad (15)$$

This drift normally causes a sudden, large jump in state error when the weight magnitudes have grown large enough to adversely affect the output $\boldsymbol{\phi}\hat{\mathbf{w}}$.

B. Leakage

The method of leakage (or σ-modification) [9] uses the update

$$\dot{\hat{\mathbf{w}}} = \beta(\boldsymbol{\phi}^T\mathbf{x} - \sigma\hat{\mathbf{w}}), \quad (16)$$

where σ is a positive constant. The term containing σ in (16) is the leakage term. By subbing in (10),(16) into (11) and using $\hat{\mathbf{w}} = \mathbf{w} - \tilde{\mathbf{w}}$ the derivative becomes

$$\dot{V} = \mathbf{x}^T(\mathbf{d} - \mathbf{G}\mathbf{x}) + \sigma\tilde{\mathbf{w}}^T(\mathbf{w} - \tilde{\mathbf{w}}), \quad (17)$$

$$\leq \|\mathbf{x}\|d_{\max} - \|\mathbf{x}\|^2\lambda_{\min}(\mathbf{G}) + \sigma\|\mathbf{w}\|\|\tilde{\mathbf{w}}\| - \sigma\|\tilde{\mathbf{w}}\|^2. \quad (18)$$

There is an elliptical surface defined on the $(\|\mathbf{x}\|, \|\tilde{\mathbf{w}}\|)$ plane found by setting (18) equal to zero, outside of which $\dot{V} < 0$. By standard Lyapunov theorems, this implies the states and weight errors are Semi-Globally Uniformly Ultimately Bounded (SGUUB). One advantage over other methods is that information about the nonlinearities is not needed to ensure stability, only to quantify the bounds. In addition, $\|\mathbf{x}\| \to 0$ as $\lambda_{\min}(\mathbf{G}) \to \infty$.

The advantage to using local basis functions is that they need not be calculated on-line when the cell is not activated (when the function value is zero). But the leakage term in (16) must be applied at all times. To accomplish this without increasing on-line computations, note the i'th weight update when the i'th basis function is zero is given by $\dot{\hat{w}} = -\nu\hat{w}_i$. This can be solved for $\hat{w}_i(t)$ during the time period Δt_i during which the basis function is zero. The solution is applied as a discrete update the instant the basis function regains a non-zero value,

$$\Delta\hat{w}_i = \left(e^{-\beta\sigma\Delta t_i} - 1\right)\hat{w}_i. \quad (19)$$

There is extra memory required to store the values of these time periods. Note that this results in resetting the weights to zero if the cell has not been activated for a long time ($\Delta\hat{w}_i \to -\hat{w}_i$ as $\Delta t_i \to \infty$). Since cells must be frequently visited, the method cannot keep previously learned trajectories in memory for long.

C. e-modification

The update known as e-modification [11] is given by

$$\dot{\hat{\mathbf{w}}} = \beta(\boldsymbol{\phi}^T\mathbf{x} - \sigma\|\mathbf{x}\|\hat{\mathbf{w}}). \tag{20}$$

When the i'th basis function is zero the i'th weight update is $\dot{\hat{w}}_i = -\nu\|\mathbf{x}\|\hat{w}_i$. There is no way to calculate this without doing on-line calculations at all times. The advantage of using local basis functions is lost, since deactivated weights still need to be updated. Thus e-modification is not considered in this paper.

D. Deadzone

The method of deadzone [10] simply stops updating the weights in the nonstable region

$$\dot{\hat{\mathbf{w}}} = \begin{cases} \beta\boldsymbol{\phi}^T\mathbf{x} & \text{if} \|\mathbf{x}\| > \frac{d_{\max}}{\lambda_{\min}(\mathbf{G})}, \\ 0 & \text{otherwise.} \end{cases} \tag{21}$$

The error will not go to zero, but stability (SGUUB) is assured. However, an accurate estimate of $d_{\max}$ is required, which requires accurate measurements of the nonlinearities. Outside disturbances must also be taken into account, making a conservative estimate of $d_{\max}$ more likely to be used.

E. Projection

The method of *parameter projection* [10] is often used in adaptive control. A simplified version of the projection algorithm for the i'th weight update is

$$\dot{\hat{w}}_i = \begin{cases} 0 & \text{if} \quad \hat{w}_i = \|\mathbf{w}\|_\infty \quad \text{and} \quad (\boldsymbol{\phi}^T\mathbf{x})_i > 0, \\ 0 & \text{if} \quad \hat{w}_i = -\|\mathbf{w}\|_\infty \quad \text{and} \quad (\boldsymbol{\phi}^T\mathbf{x})_i < 0\ , \\ \beta\boldsymbol{\phi}^T\mathbf{x} & \text{otherwise.} \end{cases} \tag{22}$$

This prevents the weight magnitudes from growing beyond $\|\mathbf{w}\|_\infty$. Again, SGUUB stability is assured. However, the quantity $\|\mathbf{w}\|_\infty$ is ill-defined. Thus, a conservative estimate would normally be used.

III. A NEW METHOD

In practice, there are severe shortcomings with the previous methods when applied to local basis functions in the presence of persistent oscillations. The persistent oscillations between cells greatly increases the tendency of the weights to drift to large magnitudes. To overcome this problem without sacrificing performance, a new method is proposed that uses two additional sets of weights. As a first step, the control weights will be kept finite by introducing a term to keep them close to a set of alternate weights $\hat{\mathbf{p}}$. The error between the alternate weights and the ideal weights $\mathbf{w}$ is denoted

$$\tilde{\mathbf{p}} = \mathbf{w} - \hat{\mathbf{p}}. \tag{23}$$

The alternate weights are, in turn, kept finite by keeping them close to a set of choice weights $\hat{\mathbf{o}}$, as identified by a cost functional. The i'th choice weight will be subject to the constraint

$$|\o_i| < o_{\max}, \tag{24}$$

where $o_{\max}$ is chosen as a (rough) estimate of $\|\mathbf{w}\|_\infty$ and we define the parameter γ

$$\gamma = \|\mathbf{w}\| + o_{\max} \geq \|\mathbf{w} - \hat{\mathbf{o}}\|. \tag{25}$$

A. Controller derivation

Choose the (adaptive control) Lyapunov function

$$V = \frac{1}{2}\mathbf{x}^T\mathbf{x} + \frac{1}{2\beta}\tilde{\mathbf{w}}^T\tilde{\mathbf{w}} + \frac{1}{2\beta}\tilde{\mathbf{p}}^T\tilde{\mathbf{p}}. \tag{26}$$

Define the *control output* as

$$\hat{\mathbf{c}}(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})\hat{\mathbf{w}}, \tag{27}$$

and the *alternate output* as

$$\hat{\mathbf{a}}(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})\hat{\mathbf{p}}, \tag{28}$$

and define the output errors

$$\tilde{\mathbf{c}} = \boldsymbol{\phi}(\mathbf{x})\mathbf{w} - \hat{\mathbf{c}}, \qquad \tilde{\mathbf{a}} = \boldsymbol{\phi}(\mathbf{x})\mathbf{w} - \hat{\mathbf{a}}. \tag{29}$$

Taking the derivative of (26) to get $\dot{V}$ and then subbing in (27) results in

$$\dot{V} \;=\; \mathbf{x}^T(\mathbf{c} \;+\; \mathbf{d} \;+\; \mathbf{u}) \;-\; \frac{1}{\beta}\tilde{\mathbf{w}}^T\dot{\hat{\mathbf{w}}} \;-\; \frac{1}{\beta}\tilde{\mathbf{p}}^T\dot{\hat{\mathbf{p}}}, \tag{30}$$

where $\mathbf{c} = \boldsymbol{\phi}(\mathbf{x})\mathbf{w}$ is a vector of ideal outputs. The following control and weight updates are proposed

$$\mathbf{u} = -\hat{\mathbf{c}} - \mathbf{G}\mathbf{x}, \tag{31}$$

$$\dot{\hat{\mathbf{w}}} = \beta\left(\boldsymbol{\phi}^T\mathbf{x} + \alpha\boldsymbol{\phi}^T(\hat{\mathbf{a}} - \hat{\mathbf{c}}) + \zeta(\hat{\mathbf{p}} - \hat{\mathbf{w}})\right), \tag{32}$$

$$\dot{\hat{\mathbf{p}}} = \beta\left(\alpha\boldsymbol{\phi}^T(\hat{\mathbf{c}} - \hat{\mathbf{a}}) + \nu(\hat{\mathbf{o}} - \hat{\mathbf{p}})\right). \tag{33}$$

with α, ζ, ν all positive constants. The Lyapunov derivative becomes

$$\begin{aligned} \dot{V} = \mathbf{x}^T(-\mathbf{G}\mathbf{x} + \tilde{\mathbf{c}} + \mathbf{d}) - \tilde{\mathbf{w}}^T\boldsymbol{\phi}^T\mathbf{x} \\ - \tilde{\mathbf{w}}^T\left(\alpha\boldsymbol{\phi}^T(\hat{\mathbf{a}} - \hat{\mathbf{c}}) + \zeta(\hat{\mathbf{p}} - \hat{\mathbf{w}})\right) \\ - \tilde{\mathbf{p}}^T\left(\alpha\boldsymbol{\phi}^T(\hat{\mathbf{c}} - \hat{\mathbf{a}}) + \nu(\hat{\mathbf{o}} - \hat{\mathbf{p}})\right) \end{aligned} \tag{34}$$

To analyze $\dot{V}$ make the following substitutions

$$\hat{\mathbf{w}} = \mathbf{w} - \tilde{\mathbf{w}}, \quad \hat{\mathbf{c}} = \mathbf{c} - \tilde{\mathbf{c}}, \quad \hat{\mathbf{p}} = \mathbf{w} - \tilde{\mathbf{p}}, \quad \hat{\mathbf{a}} = \mathbf{c} - \tilde{\mathbf{a}},$$

and then

$$\begin{aligned} \dot{V} = \mathbf{x}^T(-\mathbf{G}\mathbf{x} + \mathbf{d}) - \tilde{\mathbf{w}}^T\left(\alpha\boldsymbol{\phi}^T(\tilde{\mathbf{c}} - \tilde{\mathbf{a}}) + \zeta(\tilde{\mathbf{w}} - \tilde{\mathbf{p}})\right) \\ - \tilde{\mathbf{p}}^T\left(\alpha\boldsymbol{\phi}^T(\tilde{\mathbf{a}} - \tilde{\mathbf{c}}) + \nu(\hat{\mathbf{o}} + \tilde{\mathbf{p}} - \mathbf{w})\right). \end{aligned} \tag{35}$$

Expanding the terms gives

$$\begin{aligned} \dot{V} = -\mathbf{x}^T\mathbf{G}\mathbf{x} + \mathbf{x}^T\mathbf{d} - \alpha(\tilde{\mathbf{c}} - \tilde{\mathbf{a}})^T(\tilde{\mathbf{c}} - \tilde{\mathbf{a}}) \\ + \zeta\tilde{\mathbf{w}}^T\tilde{\mathbf{p}} - \zeta\tilde{\mathbf{w}}^T\tilde{\mathbf{w}} + \nu\tilde{\mathbf{p}}^T(\mathbf{w} - \hat{\mathbf{o}}) - \nu\tilde{\mathbf{p}}^T\tilde{\mathbf{p}}. \end{aligned} \tag{36}$$

The derivative can be bounded

$$\begin{aligned} \dot{V} \leq -\lambda_{\min}(\mathbf{G})\|\mathbf{x}\|^2 + \|\mathbf{x}\|d_{\max} + \zeta\|\tilde{\mathbf{w}}\|\|\tilde{\mathbf{p}}\| \\ + \nu\gamma\|\tilde{\mathbf{p}}\| - \zeta\|\tilde{\mathbf{w}}\|^2 - \nu\|\tilde{\mathbf{p}}\|^2, \end{aligned} \tag{37}$$

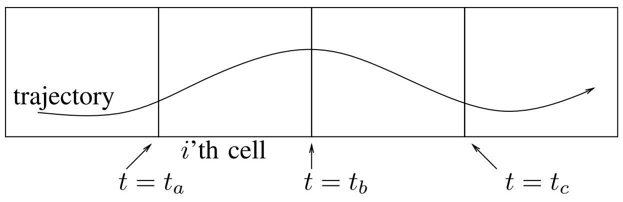

Fig. 1. Determining the cost functional (four local cells shown)

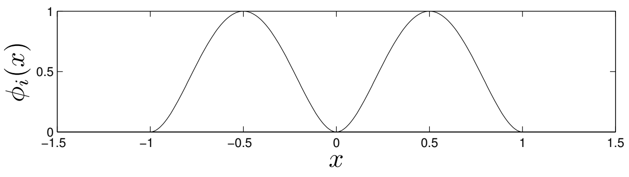

Fig. 2. Two local spline basis functions

$$\dot{V} \leq \begin{bmatrix} \|\mathbf{x}\| \\ \|\tilde{\mathbf{w}}\| \\ \|\tilde{\mathbf{p}}\| \end{bmatrix}^T \begin{bmatrix} -\lambda_{\min}(\mathbf{G}) & 0 & 0 \\ 0 & -\zeta & \zeta/2 \\ 0 & \zeta/2 & -\nu \end{bmatrix} \begin{bmatrix} \|\mathbf{x}\| \\ \|\tilde{\mathbf{w}}\| \\ \|\tilde{\mathbf{p}}\| \end{bmatrix} + \begin{bmatrix} \|\mathbf{x}\| \\ \|\tilde{\mathbf{w}}\| \\ \|\tilde{\mathbf{p}}\| \end{bmatrix}^T \begin{bmatrix} d_{\max} \\ 0 \\ \nu\gamma \end{bmatrix}, \quad (38)$$

and if the parameters are chosen such that

$$\zeta - 4\nu < 0, \quad (39)$$

then $\dot{V} < 0$ outside of a (bounded) region near the origin, implying SGUUB stability.

B. Identifying the choice weights

A cost functional for the i'th cell is defined that incorporates the average error in the i'th cell, the magnitude of the i'th weight, and the average error in the next cell activated in time. The i'th cell is activated at time t_a, deactivated at time t_b, whereas the next cell is activated at time t_b and deactivated at time t_c. This notation is illustrated in Figure 1 where a trajectory is shown moving through four cells, and the middle two will be used to compute the cost-functional. The cost functional is defined at time t_c as

$$f_i(t_c) = \frac{\int_{t_a}^{t_c} \|\mathbf{x}\|}{t_c - t_a} + \begin{cases} (\kappa_1 - \kappa_2(t_b - t_a))|\hat{w}_i| & \text{if } (t_b - t_a) > \frac{\kappa_1}{\kappa_2}, \\ 0 & \text{otherwise.} \end{cases} \quad (40)$$

If this is the minimal value found so far, then the i'th weight is kept in memory as a choice weight $\hat{o}_i$. Thus, the following assignment occurs at time t_c

$$\hat{o}_i \leftarrow \hat{w}_i(t_b) \quad \text{if} \quad f_i(t_c) < f_i(\tau), \quad \forall\, \tau \leq t_a. \quad (41)$$

The parameters κ_1 and κ_2 are positive constants chosen to ensure that cells activated for short periods of time are penalized much more for large weight magnitude. They are chosen with knowledge of the period of oscillation T such that $\frac{\kappa_1}{\kappa_2} > T$. Thus, cells that have less than a full period of oscillation occur within them will always be penalized for the weight magnitude. Only cells that contain more than one period of oscillation will be penalized solely on average state error.

C. Discrete updates

The terms multiplied by ζ and ν in (32) and (33) respectively must be applied regardless of the value of ϕ_i. They must be applied even when the cell is deactivated. To accomplish this, identify the weight updates when a cell is deactivated as $\dot{\hat{w}}_i = \zeta(\hat{p}_i - \hat{w}_i)$ and $\dot{\hat{p}}_i = \nu(\hat{o}_i - \hat{p}_i)$. The value of choice weight $\hat{o}_i$ will be a constant during deactivation ($\dot{\hat{o}}_i = 0$ for $t_{i,o} \leq t \leq t_{i,a}$). Thus the solutions for $w_i(t)$ and $p_i(t)$ can be calculated using standard techniques and upon reactivation of the cell the weights are set to the following values (assuming $\zeta \neq \nu$)

$$\begin{aligned} \hat{w}_i(t_{i,a}) =& \hat{w}_i(t_{i,o})e^{-\zeta\Delta t_i} \\ &+ \hat{p}_i(t_{i,o})\left[\left(\frac{e^{-\zeta\Delta t_i}}{\nu - \zeta}\right) + \left(\frac{e^{-\nu\Delta t_i}}{\zeta - \nu}\right)\right] \\ &+ \hat{o}_i\zeta\left[1 - e^{-\zeta\Delta t_i} - \left(\frac{e^{-\zeta\Delta t_i}}{\nu - \zeta}\right) - \left(\frac{e^{-\nu\Delta t_i}}{\zeta - \nu}\right)\right] \end{aligned} \quad (42)$$

$$\hat{p}_i(t_{i,a}) = \hat{p}_i(t_{i,o})e^{-\nu\Delta t_i} + \hat{o}_i(1 - e^{-\nu\Delta t_i}) \quad (43)$$

where $\Delta t_i = t_{i,a} - t_{i,o}$, the length of time the i'th cell was deactivated. Note that if a trajectory has been learned but not implemented for a long time ($\Delta t_i \to \infty$) then the weights will be set to the choice weights upon reactivation ($\hat{w}_i(t_{i,a}) \leftarrow \hat{o}_i$ and $\hat{p}_i(t_{i,a}) \leftarrow \hat{o}_i$). This is an immediate advantage over leakage where the weights would be reset to zero.

IV. Simulation Results

A. Using local basis functions

For demonstration only two local basis functions, with one state input, are used on a generic system. In this way the problem and the proposed solution can be understood at the simplest level. The basis functions used in the simulation are local spline polynomials. Each function is localized to a region. To this end define an activation function for the i'th cell as

$$\mathcal{C}_i(x) = \begin{cases} 1 & \text{if} \quad \min_i \leq x \leq \max_i, \\ 0 & \text{otherwise,} \end{cases} \quad (44)$$

where $\min_i$ and $\max_i$ are the lower and upper boundaries of the cell respectively. Possible basis function types include rectangular (binary CMAC), triangular (fuzzy set), or spline. A continuous differentiable function is often (mathematically) required, so here the following spline is used:

$$g_i(x) = 16(h_i(x)^2 - 2h_i^3(x) + h_i^4(x)), \quad (45)$$

where $0 \leq h_i(x) \leq 1$ gives the position inside the cell calculated

$$h_i(x) = \mathcal{C}_i(x) \left(\frac{x - \min_i}{\max_i - \min_i} \right). \tag{46}$$

The two splines used in the simulation are illustrated in Figure 2.

To examine performance the differential equations resulting from applying the control and update rules are simulated. In this hypothetical case the ideal weights are chosen and the disturbance $\mathbf{d}$ is then the input to the system. Since it is now the nonlinearity that is unknown, the advantage is that the performance can be inspected independent of the type of nonlinearity, and it becomes easier to see the effects of disturbances and modeling errors. The system can be simulated with only two basis functions (and two weights) so that the behavior of the weights can be understood completely. We are interested in the effect of oscillation of the state about a cell boundary, and the common parameters used to investigate this were chosen to be

$$\mathbf{w} = \begin{bmatrix} -1 & 1 \end{bmatrix}^T \qquad \beta = 1$$

$$d = 0.1 \sin(2\pi t) \qquad G = 1$$

and the two basis functions shown in Figure 2 were used. The simulations are run for $t = 500$ seconds. If the weight drift is continuing after this time we consider the stability from the robust modification to be inadequate (even though weight drift will be halted as $t \to \infty$). The justification is that in practice, with more complex systems, this continued weight drift will eventually result in a large leap in state error, often appearing to go unstable. Even with this simple two-function system a leap in state error is sometimes observed, depending on basis function placement and weight value. If, on the other hand, weight drift has been stopped in $t = 500$ seconds we consider this adequate stability. The goal is to halt the weight drift without sacrificing performance. The measure of error used to evaluate performance is the amplitude of the steady-state oscillation

$$\text{error} = \|\mathbf{x}(t > 400)\|_\infty. \tag{47}$$

B. Simulation using robust weight updates

1) Leakage: The control (10) applied to the system (6) with $n = 1$ and weight updates (16) result in the differential equations

$$\dot{x} = \boldsymbol{\phi}\tilde{\mathbf{w}} + d - Gx, \tag{48}$$

$$\dot{\tilde{w}}_i = \beta \left[-\phi_i x + \sigma(w_i - \tilde{w}_i) \right] \quad \text{for } i = 1, 2. \tag{49}$$

In Figure 3 the parameter σ is varied and the maximum error measure is plotted, and the observation on whether weight drift is prevented or not is marked with different symbols. It can be seen that with leakage there is a direct trade-off between performance and stability. In order to halt the weight drift, performance must be sacrificed.

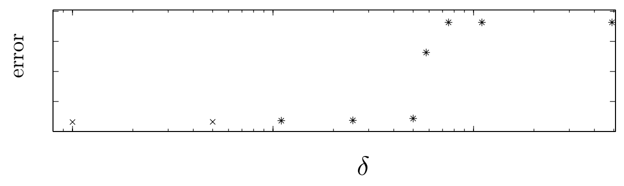

Fig. 4. Deadzone: with weight drift, 'x', weight drift prevented, '*'.

2) Deadzone: The same system with the same control and weight updates (21) result in the differential equations

$$\dot{x} = \boldsymbol{\phi}\tilde{\mathbf{w}} + d - Gx, \tag{50}$$

$$\dot{\tilde{w}}_i = \begin{cases} -\beta\phi_i x & \text{if } |x| > \delta \\ 0 & \text{otherwise} \end{cases} \qquad i = 1, 2, \tag{51}$$

where the parameter δ is an estimate of the disturbance bound. For SGUUB stability we require $\delta > d_{\max}/G = 0.1$. It can be seen in Figure 4 that there is a small region $(0.1 < \delta < 0.5)$ where the weight drift has been prevented without sacrificing performance. Thus, deadzone may be appropriate for local basis functions. However, the value of $d_{\max}$ must also include any external disturbances, so it may be difficult to estimate in practice. One can see in the graph that an inaccurate estimate may lead immediately to poor performance. The transition from good to poor performance is very steep, unlike leakage, so that the error increases without warning. This type of behavior will be inappropriate for critical systems where $d_{\max}$ is uncertain.

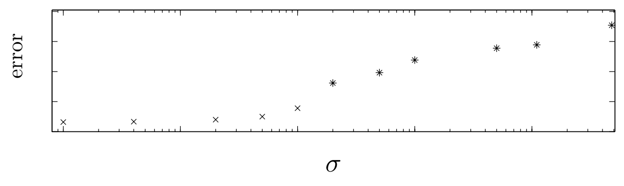

Fig. 3. Leakage: with weight drift, 'x', weight drift prevented, '*'.

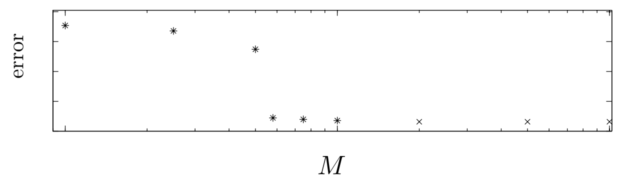

Fig. 5. Projection: with weight drift, 'x', weight drift prevented, '*'.

3) Projection: The same system with the same control and weight updates (22) result in the differential equations

$$\dot{x} = \boldsymbol{\phi}\tilde{\mathbf{w}} + d - Gx, \tag{52}$$

$$\dot{\tilde{w}}_i = \begin{cases} 0 & \text{if} \quad w_i - \tilde{w}_i = M \quad \text{and} \quad \phi_i x > 0, \\ 0 & \text{if} \quad w_i - \tilde{w}_i = -M \quad \text{and} \quad \phi_i x < 0, \\ -\beta\phi_i x & \text{otherwise,} \end{cases} \tag{53}$$

where M is an estimate of the maximum weight, and must chosen such that $M > \|\mathbf{w}\|_\infty = 1$ for SGUUB stability. Normally, the value of $\|\mathbf{w}\|_\infty$ is ill-defined, and must be estimated. In this case, we have the luxury of having it as a specified parameter. In Figure 5 we see that in order to eliminate weight drift we must choose $M < 1$, negating the stability result. Thus, projection is not an appropriate method for training local basis functions.

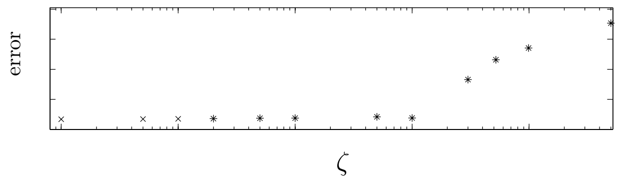

Fig. 6. New Method: with weight drift, 'x', weight drift prevented, '*'.

C. Simulation using new method

Given the same system with control applied as before but with new proposed weight updates (32),(33), and (41) the resulting differential equations are

$$\dot{x} = \boldsymbol{\phi}\tilde{\mathbf{w}} + d - Gx, \tag{54}$$

$$\dot{\tilde{w}}_i = \beta\left(-\phi_i x + \alpha\phi_i(\tilde{a} - \tilde{c}) + \zeta(\tilde{p}_i - \tilde{w}_i)\right), \tag{55}$$

$$\dot{\tilde{p}}_i = \beta\left(\alpha\phi_i(\tilde{c} - \tilde{a}) + \nu(-\hat{o}_i - \tilde{p}_i)\right), \tag{56}$$

where $\beta = 1$, $\alpha = 1$ and the ratio of ζ and ν is kept constant at $\zeta = 2\nu$. In Figure 6 we see that there is a large region $(0.002 < \zeta < 0.1)$ where the weight drift has been prevented without sacrificing performance. Above that, there is gradual increase in error. Thus, it is relatively easy to pick values of ζ and ν such that both performance and stability are maximized, by some measure.

V. Conclusions

An alternative method for robust weight updates has been proposed that is appropriate for direct adaptive control of nonlinear systems using local basis functions. It is demonstrated that the traditional robust weight update methods of leakage, e-modification, deadzone, and projection are inappropriate in this case if persistent oscillations are present in the inputs. Leakage resulted in a direct trade-off between performance and stability. The e-modification method requires on-line computation of all basis functions, preventing the efficient indexing of local basis functions. The method of parameter (weight) projection resulted in poor performance for a Lyapunov-stable solution. The method of deadzone can result in acceptable performance if the disturbance bounds are known with certainty, but can suddenly lead to poor performance otherwise.

The new method proposed in this paper can prevent weight drift and appears to do so without sacrificing performance. There are also other advantages. Unlike leakage, the weights do not get reset to zero if a local basis function has been deactivated for a length of time. Unlike deadzone, certain knowledge of disturbance bounds is not needed. When disturbances start to affect performance, the decline is gradual like leakage, not sudden like deadzone. Also unlike deadzone, the error can reach exactly zero.

Since local basis functions are very computationally efficient compared to global ones, the proposed method has the possibility of wide application. Adaptive control using β-splines, neural networks, and fuzzy sets have been widely proposed, but practical computer power has limited their implantation to relatively simple systems: either those of low order or those that do not exhibit persistent oscillations. The ability to use local basis functions will greatly increase the applicability of such methods and make nonlinear control more feasible for a wider variety of systems.

References

[1] R. S. Minhas and S. A. Bortoff, "Adaptive feedback linearization using multivariable spline functions," in *Proc. American Control Conference*, Seattle, Washington, 1995, pp. 3207–3211.
[2] R. Sanner and J. Slotine, "Guassian networks for direct adaptive control," *IEEE Trans. Neural Networks*, vol. 3, no. 6, pp. 837–863, 1992.
[3] F. Lewis, S. Jagannathan, and A. Yesildirek, *Neural Network Control of Robot Manipulators and Nonlinear Systems*. Philidelphia, PA: Taylor and Francis, 1999.
[4] S. Commuri and F. Lewis, "CMAC neural networks for control of nonlinear dynamical systems: structure, stability and passivity," in *Proc. IEEE International Symposium on Intelligent Control*, Monetery, Aug. 1995, pp. 123–129.
[5] L. X. Wang, "Stable adaptive fuzzy control of nonlinear systems," vol. 1, pp. 146–155, May 1993.
[6] M. Brown and C. Harris, *Neurofuzzy Adaptive Modelling and Control*. New Jersey: Prentice-Hall, 1995.
[7] S. Ge, T. Lee, and C. Harris, *Adaptive Neural Network Control of Robotic Manipulators*. New Jersey: World Scientific, 1998.
[8] M. French, C. Szepaesvari, and E. Rogers, *Performance of Nonlinear Approximate Adaptive Controllers*. West Sussex, England: Wiley, 2003.
[9] P. Ioannuou and P. Kokotovic, "Instability analysis and improvement of robustness of adaptive control," *Automatica*, vol. 20, no. 5, pp. 583–594, 1984.
[10] B. Egardt, *Stability of Adaptive Controllers*. New York: Springer-Verlag, 1979.
[11] K. Narendra and A. Annaswamy, "A new adpative law for robust adaptation without persistant excitation," *IEEE Trans. Automat. Contr.*, vol. AC-32, no. 2, pp. 134–145, Feb. 1989.