Available online at www.sciencedirect.com

SCIENCE DIRECT®

Neural Networks 18 (2005) 71–90

Neural Networks

www.elsevier.com/locate/neunet

Composite adaptive control with locally weighted statistical learning

Jun Nakanishi[a,b,*], Jay A. Farrell[c,1], Stefan Schaal[a,d,2]

[a]ATR Computational Neuroscience Laboratories, Department of Humanoid Robotics and Computational Neuroscience, 2-2 Hikaridai, Seiko-cho, Soraku-gun, Kyoto 619-0288, Japan
[b]Computational Brain Project, ICORP, Japan Science and Technology Agency, Kyoto 619-0288, Japan
[c]Department of Electrical Engineering, University of California, Riverside Riverside, CA 92521, USA
[d]Department of Computer Science and Neuroscience, University of Southern California, Los Angeles, CA 90089-2520, USA

Received 2 April 2004; revised 30 August 2004; accepted 30 August 2004

Abstract

This paper introduces a provably stable learning adaptive control framework with statistical learning. The proposed algorithm employs nonlinear function approximation with automatic growth of the learning network according to the nonlinearities and the working domain of the control system. The unknown function in the dynamical system is approximated by piecewise linear models using a nonparametric regression technique. Local models are allocated as necessary and their parameters are optimized on-line. Inspired by composite adaptive control methods, the proposed learning adaptive control algorithm uses both the tracking error and the estimation error to update the parameters.

We first discuss statistical learning of nonlinear functions, and motivate our choice of the locally weighted learning framework. Second, we begin with a class of first order SISO systems for theoretical development of our learning adaptive control framework, and present a stability proof including a parameter projection method that is needed to avoid potential singularities during adaptation. Then, we generalize our adaptive controller to higher order SISO systems, and discuss further extension to MIMO problems. Finally, we evaluate our theoretical control framework in numerical simulations to illustrate the effectiveness of the proposed learning adaptive controller for rapid convergence and high accuracy of control.

Keywords: Adaptive control; Statistical learning of nonlinear functions; Composite adaptation; Locally weighted learning; Receptive field weighted regression

1. Introduction

From the viewpoint of statistical learning, model-based adaptive control can be conceived of as a function approximation process where the objective is to adjust some parameters of the control system's model such that a cost criterion is minimized. Model-based adaptive control is well-studied if the control system is linear, similarly as statistical learning is well-understood for linear systems. For nonlinear systems, a common practice in learning is to expand the input space of the original learning data by means of nonlinear basis functions such that the resulting, usually higher dimensional representation becomes linear in the learning parameters again. The same idea has been explored in nonlinear model-based adaptive control (Narendra & Annaswamy, 1989; Slotine & Li, 1991) if the control system's dynamics are amenable to a transformation where the learning parameters appear linearly in the equations of motion. For instance, globally stable model-based adaptive controllers for robot arms have been proposed which exploit the properties of linear inertial parameters of rigid-body dynamics (Slotine & Li, 1987; Whitcomb, Rizzi, & Koditschek, 1993). However, if the structure of the system dynamics is unknown, learning methods are needed to approximate the unknown functions. For this purpose, multi-layer sigmoidal neural networks (Chen & Khalil, 1995; Levin & Narendra, 1993) were

* Corresponding author. Address: ATR Computational Neuroscience Laboratories, Department of Humanoid Robotics and Computational Neuroscience, 2-2 Hikaridai, Seiko-cho, Soraku-gun, Kyoto 619-0288, Japan. Tel.: +81 774 95 2408; fax: +81 774 95 1236.
[1] Tel.: +1 951 827 2159; fax: +1 951 827 2425.
[2] Tel.: +1 213 740 9418; fax: +1 213 740 5687.

0893-6080/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2004.08.009

suggested. However, nonlinear parameterized neural networks make global stability proofs difficult, may contain local minima, and often require off-line training. Thus, function approximators that are linear in learning parameters are preferable, as has been demonstrated with radial basis function (RBF) networks (Sanner & Slotine, 1992; Seshagiri & Khalil, 2000), and piecewise linear approximators (Choi & Farrell, 2000) in tracking error-based adaptive control.

To our knowledge, while many publications on nonlinear adaptive control with function approximation focus on the control theoretic part of the topic, few of them stress the problems of function approximation in this context. It is important, however, to point out that incremental learning of systems with unknown complexity still remains an open issue in the statistical learning literature. An ideal algorithm needs to avoid potential numerical problems from redundancy in the input data, eliminate irrelevant input dimensions, keep the computational complexity of learning updates low while remaining data efficient, allow for real-time learning in high dimensional spaces, and, of course, achieve accurate function approximation and adequate generalization. Additionally, a particular problem of learning control is that the domain of operation of the plant is usually only known in terms of some upper bounds. Creating a function approximator in such an overestimated domain can become computationally rather expensive due to allocating too many learning parameters, and it also bears the danger of fitting noise in the data if these parameters are not properly constrained by the learning data. In general, given that the complexity of the function to be approximated is unknown, allocating the right number of learning parameters is a difficult problem, in particular if learning is to proceed on-line.

In this paper, we will suggest an approach to model-based nonlinear adaptive control that employs function approximation with automatic structure adaptation, i.e. the learning system grows incrementally with the size of the domain of operation and the complexity of the functions to be learned. For this purpose, we will employ a learning framework from the nonparametric statistics literature, derived from methods of kernel regression and also discussed under the name of *locally weighted learning* (Atkeson, Moore, & Schaal, 1997). The essence of our learning algorithms is to accomplish function approximation with piecewise linear models, where local models are allocated as needed without the danger of over parameterizing the learning system. Such automatic structure adaptation of the function approximator is particularly desirable when the domain of operation and complexity of the function to be approximated are not known in advance, as is often the case in high dimensional control systems where only unrealistic upper bounds of such values can be derived. Based on this framework, we will propose a provably stable learning adaptive controller inspired by the idea of composite adaptive control which uses both the tracking error and the prediction error to update the parameters (Slotine & Li, 1989, 1991). Stability analyses and numerical simulations are provided to illustrate the effectiveness of the proposed controller and demonstrate very rapid convergence to accurate tracking performance.

This paper is organized as follows: In Section 2, we will first discuss statistical learning of nonlinear functions, and motivate our choice of the locally weighted learning framework. Second, Section 3.1 introduces theoretical development of our learning adaptive control framework for first order SISO systems, and present stability proof including a parameter projection method that is needed to avoid potential singularities during adaptation. Then, Section 5, we generalize our adaptive controller to higher order SISO systems, and discuss further extension to MIMO problems in Section 6. Finally, in Section 7, we evaluate our theoretical control framework in numerical simulations to illustrate the effectiveness of the proposed learning adaptive controller for rapid convergence and high accuracy of control.

2. Statistical learning of nonlinear functions

The general structure of the control system of interest is a class of nonlinear MIMO systems of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{G}(\mathbf{x})\mathbf{u} \tag{1}$$

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) \tag{2}$$

where $\mathbf{x} \in \Re^n$ is a state, $\mathbf{z} \in \Re^q$ is an output, $\mathbf{u} \in \Re^m$ is an input, $\mathbf{f}$: $\Re^n \to \Re^n$, $\mathbf{G}$: $\Re^q \to \Re^{n \times m}$ are unknown functions, and $\mathbf{h}$: $\Re^n \to \Re^q$ denotes a mapping from the state to the output. Our goal in this paper is to design a provably stable adaptive controller for such a class of nonlinear systems with on-line learning of the unknown nonlinearities of the dynamics in order to achieve accurate tracking and rapid adaptation.

In this section, we first outline the learning algorithm, *receptive field weighted regression* (RFWR), which falls into a class of *locally weighted learning framework* (LWL) (Atkeson et al., 1997; Schaal & Atkeson, 1998). LWL advocates that each local model in the function approximator minimizes the *locally weighted* error criterion independently. Then, we discuss function approximation for the plant dynamics (1). We motivate our choice of this locally weighted learning framework in comparison to cooperative learning strategies.

2.1. Receptive field weighted regression

For the development of adaptive learning controllers herein, we focus on a particular locally weighted learning (LWL) algorithm, receptive field weighted regression (RFWR) (Schaal & Atkeson, 1998). RFWR is an incremental version of the locally weighted regression (LWR) algorithm with automatic structure adaptation.[3]

Nonlinear function approximation in RFWR is accomplished by covering the input space of the training data with locally linear models and by forming a final prediction from the weighted average of the individual predictions

$$\hat{y} = \frac{\sum_{k=1}^{N} w_k \hat{y}_k}{\sum_{k=1}^{N} w_k} \tag{3}$$

where

$$\hat{y}_k = \bar{\mathbf{x}}_k^T \hat{\boldsymbol{\theta}}_k \text{ and } \bar{\mathbf{x}}_k = [(\mathbf{x}-\mathbf{c}_k)^T, 1]^T \tag{4}$$

and $\mathbf{c}_k$ is the center of the k-th linear model. The weight w_k is a measure of how much a data point $\mathbf{x}$ falls into the region of validity of each linear model, and is characterized by a kernel function. In (Schaal & Atkeson, 1998), a Gaussian kernel

$$w_k = \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{c}_k)^T \mathbf{D}_k (\mathbf{x}-\mathbf{c}_k)\right) \tag{5}$$

is used for analytical convenience, where $\mathbf{D}_k$ is a positive definite matrix, called a distance metric. For adaptive control, the infinite support of the Gaussian is less suitable. Therefore, we employ a biquadratic kernel instead, which has compact support[4].

$$w_k = \begin{cases} (1-d^2)^2 & \text{if } |d| < 1 \\ 0 & \text{otherwise} \end{cases} : \text{biquadratic kernel} \tag{6}$$

where d is the Mahalanobis distance defined by

$$d^2 = (\mathbf{x}-\mathbf{c}_k)^T \mathbf{D}_k (\mathbf{x}-\mathbf{c}_k). \tag{7}$$

The biquadratic kernel belongs to a family of possible kernels for LWL that empirically demonstrate comparable function approximation performance and only differ in subtle asymptotic properties (see (Atkeson et al., 1997) for more the details).

In learning, each locally linear model is trained independently in order to minimize the locally weighted error criterion

$$J_k = \sum_{i=1}^{p} w_{k,i} (y_i - \hat{y}_{k,i})^2 \tag{8}$$

where y_i is a target for learning, and $\hat{y}_{k,i}$ and $w_{k,i}$ are given by (4) and (5) respectively for each input data point $\mathbf{x}_i$. When a given training point ($\mathbf{x}$, y) falls within the support of a local model, i.e. $w_k > 0$, the regression parameters $\hat{\boldsymbol{\theta}}_k$ are updated by weighted recursive least squares, which accomplishes fast Newton-like learning:

$$\hat{\boldsymbol{\theta}}_k^{n+1} = \hat{\boldsymbol{\theta}}_k^{n} + w_k \mathbf{P}_k^{n+1} \bar{\mathbf{x}}_k e_{pk} \tag{9}$$

where

$$\mathbf{P}_k^{n+1} = \frac{1}{\lambda}\left(\mathbf{P}_k^n - \frac{\mathbf{P}_k^n \bar{\mathbf{x}}_k \bar{\mathbf{x}}_k^T \mathbf{P}_k^n}{\frac{\lambda}{w_k} + \bar{\mathbf{x}}_k^T \mathbf{P}_k^n \bar{\mathbf{x}}_k}\right) \text{ and}$$
$$e_{pk} = y - \hat{y}_k = y - \bar{\mathbf{x}}_k^T \hat{\boldsymbol{\theta}}_k^n. \tag{10}$$

As suggested in standard recursive system identification (Ljung and Söderström, 1986), we use a forgetting factor $\lambda \in [0, 1]$ such that the sufficient statistics $\mathbf{P}_k$ will not retain old training data for too long a time, as adaptation of $\mathbf{D}_k$ (see Section 3.5) will render older data points contributing to $\mathbf{P}_k$ obsolete.

The distance metric $\mathbf{D}_k$ is learned by gradient descent in a locally weighted leave-one-out cross validation criterion that approximates the statistical expectation of the cost function (8) (Schaal & Atkeson, 1998). We will explain this criterion in more detail in Section 3.5.

Learning in RFWR is initialized with no local model at all. Whenever a training point $\mathbf{x}$ does not activate any local model by more than a certain threshold w_{gen}, a new local model is added with $\mathbf{c} = \mathbf{x}$. The threshold w_{gen} essentially determines the overlap between local models and indirectly regulates how many local models will be maintained. Increased overlap improves the smoothness of the total prediction in (3). Whenever a training point falls within the support of a local model, both distance metric and regression parameters are updated. We would like to emphasize that these updates can be computed for each local model completely independently of all other models, which creates, as shown in the next section, robustness of structure adaption towards the total number of local models in the learning system; for more details, see (Schaal & Atkeson, 1998).

2.2. Cooperative vs local learning

Before turning to developing an adaptive control framework with RFWR, we will first focus on motivating why this particular statistical learning approach offers several benefits with respect to previous work in the literature. At the heart of this issue is the difference between cooperative and local learning, explained in the following.

2.2.1. Generalized linear function approximation

The standard approach of function approximation using linear parametrization with nonlinear basis functions can be

[3] The original notion of locally weighted learning (LWL) and locally weighted regression (LWR) refer to memory based learning techniques, however, in this paper we refer to a nonmemory based spatially localized learning technique.

[4] Note that $w_k(x) = 0$ for $d > 1$. The support of $w_k(x)$ is $\mathcal{S}_k = \text{Cl}\{x \in \mathcal{D} | w_k(x) > 0\}$ where Cl denotes closure. Since $\mathbf{D}_k$ is positive definite, $\mathcal{S}_k$ is closed and bounded.

formalized as

$$\hat{y} = \hat{f}(\mathbf{x}) = \sum_{k=1}^{N} \boldsymbol{\phi}_k^T(\mathbf{x})\hat{\boldsymbol{\theta}}_k \tag{11}$$

where $\hat{y}$ denotes the prediction of the true function $f(\mathbf{x})$ given the n-dimensional input vector $\mathbf{x}$. $\boldsymbol{\phi}_k(\mathbf{x})$ is a vector of nonlinear basis functions. The parameters $\hat{\boldsymbol{\theta}}_k$ need to be estimated from data, either arriving as pairs of $(\mathbf{x}_i, y_i)$ or as $(\mathbf{x}_i, e_i)$, where y_i is a target for learning and e_i is an error signal that approximates the prediction error $e_{p,i} = f(\mathbf{x}_i) - \hat{f}(\mathbf{x}_i)$. y_i and e_i are assumed to be contaminated by mean-zero noise. A simple choice for $\boldsymbol{\phi}_k(\mathbf{x})$ is a Gaussian function, e.g. $\phi_k(\mathbf{x}) = \exp(-0.5(\mathbf{x}-\mathbf{c}_k)^T\mathbf{D}_k(\mathbf{x}-\mathbf{c}_k))$, as used in radial-basis function networks. In general, the kernel should be selected according to the class of function to be approximated (Vijayakumar & Ogawa, 1999).

For training, the learning system's objective is to minimize the least squares criterion

$$J = \sum_{i=1}^{p} (y_i - \hat{f}_i)^2 \tag{12}$$

over all p available training data points such that future predictions are as close as possible to the true target function[5], where $\hat{f}_i$ is the prediction for each data point $\mathbf{x}_i$ given in (11). When inserting (11) into (12), one can recognize that this form of error criterion causes cooperation among all the basis functions in approximating every $\hat{f}_i$, i.e. each basis function contributes a fraction towards reducing the approximation error. This cooperative strategy needs to be contrasted with another learning framework, *locally weighted learning* (LWL) (Atkeson et al., 1997). As mentioned in Section 2.1, LWL advocates that each basis function and its parameters should be conceived of as an *independent* local model that minimizes the *locally weighted* error criterion

$$J_k = \sum_{i=1}^{p} w_{k,i}(y_i - \hat{y}_{k,i})^2 \text{ where } \hat{y}_{k,i} = \boldsymbol{\phi}_k^T(\mathbf{x}_i)\hat{\boldsymbol{\theta}}_k. \tag{13}$$

The weight $w_{k,i}$ is computed from a spatially localized kernel function, often chosen to be a Gaussian kernel (5) although many alternatives exist (Atkeson et al., 1997). The final prediction $\hat{y}$ for a point $\mathbf{x}$ is formed from the normalized weighted average of the predictions $\hat{y}_k$ of all local models (cf. (3)), using (5) to determine the weighting factors. There appears to be an initial similarity between LWL and the radial basis function example above as both approaches use Gaussian kernels and a linear parameterization. However, there is a crucial difference between cooperative learning and LWL. In LWL, each local model is trained entirely independently of all other local models such that the total number of local models in the learning system does not directly affect how complex a function can be learned—complexity can only be controlled by the level of adaptability of each local model, but not by the total number of local models. This property avoids over-fitting if a robust learning scheme exists for training the individual local model (Schaal & Atkeson, 1998). In contrast, in cooperative learning, adding a new basis function (equivalent to adding a new local model in LWL) allows fitting some y_i more accurately due to the increased number of parameters that now contribute to fitting each y_i. Thus, if too many basis functions are added, the learning system is likely to fit noise in the data. Importantly, adding a new basis function also affects changes in all previously learned parameters.

The difference between cooperative learning and LWL is crucial for structure adaptation during incremental learning, i.e. the gradual adding of more and more local models as a function of the complexity of the training data and the size of the workspace that is encountered as considered in RFWR. The example in the following section will give an intuitive illustration of this difference.

2.2.2. Empirical example

This section compares the following four different learning schemes for a simple function approximation problem to illustrate the difference of the properties of cooperative and locally weighted learning methods:

1. Locally Weighted Regression (LWR): $\boldsymbol{\phi}_k(\mathbf{x}) = [\mathbf{x}^T 1]^T$ in (8) with w_k defined by (5)
2. LWR with normalized weights (nLWR): as LWR, but use the normalized weight $\bar{w}_k = w_k / \sum_{k=1}^{N} w_k$ for (8).
3. Radial Basis Function Networks with locally linear models (RBF): $\boldsymbol{\phi}_k(\mathbf{x}) = w_k[\mathbf{x}^T 1]^T$ in (11), where w_k is computed from a Gaussian RBF that has the same kernel as (5).
4. RBF with normalized weights (nRBF): as RBF, but using the same normalized weights as nLWR.

RBF and nRBF are both cooperative learning methods, while LWR and nLWR are typical LWL systems. Learning algorithms for RBF networks can be found in (Bishop, 1995). For LWL methods, the learning algorithm described in Section 2.1 is used.

Fig. 1 shows the results of function approximation of the function $y = 2x^3 + 0.5x^2 - 2x$ using 50 training points distributed in $x \in [-1.5, 1.5]$ with added Gaussian noise ($N(0, 0.16)$) in the outputs. Each plot in Fig. 1 compares the fit for 3, 7, and 13 basis functions—for 3 and 7 basis functions, the regular and normalized weighting kernels are shown in Fig. 3. For all four methods, initial approximation results using three basis functions are not good enough due to an insufficient number of basis functions; thus, four more basis functions are added, leading to improved performance for all algorithms. From the view point of structure learning, it is interesting to examine how much the individual learning systems had to change the parameters of the three-basis-function system when the four additional basis

[5] Note that a statistically precise formulation would be to minimize the expectation of J, not just the sum squared error of the training data.

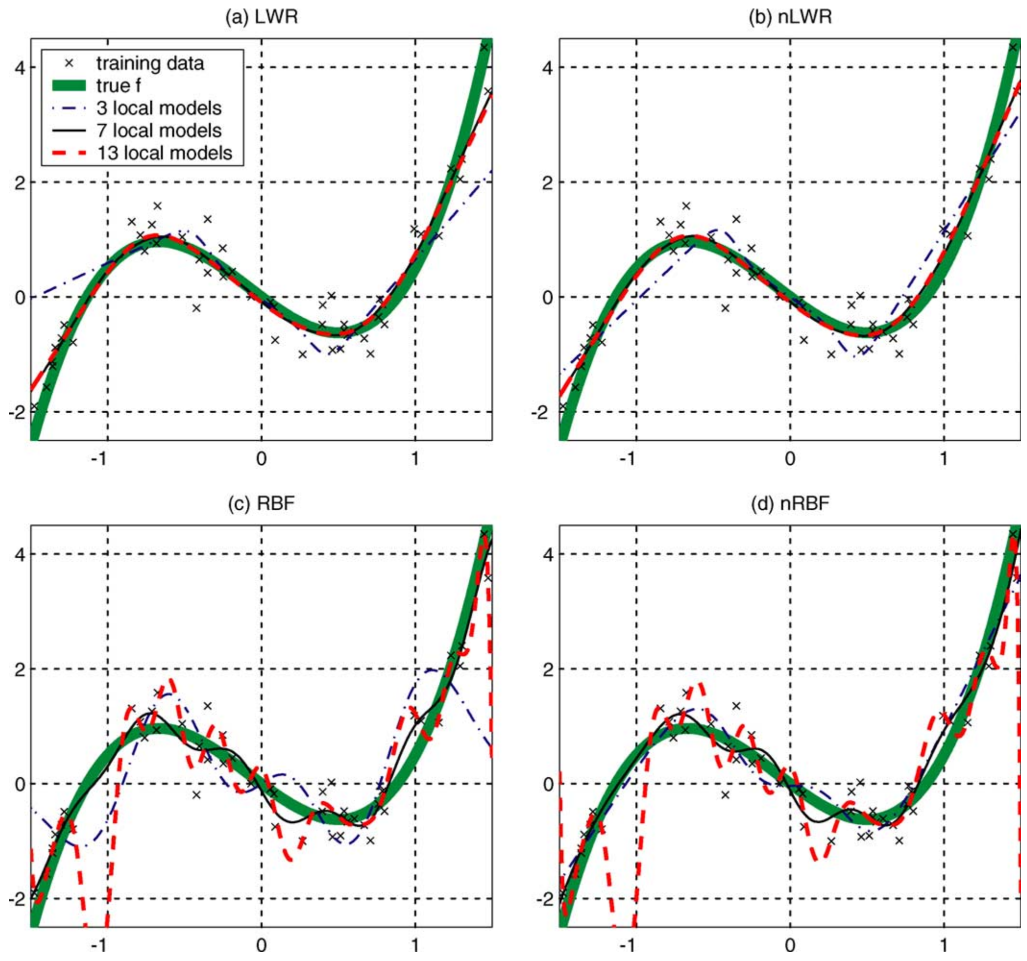

Fig. 1. Results of function approximation of the function $y=2x^3+0.5x^2-2x$ by four different approximation schemes with 3, 7, and 13 local models. (a) locally weighted regression. (b) locally weighted regression with normalized weights. (c) cooperative learning using a RBF network. (d) cooperative learning using a RBF network with normalized weights. With three local models, oversmoothing can be seen because of the insufficient number of local models. With seven local models, fitting is improved in each case. With 13 local models, RBF and nRBF start overfitting while locally linear learning exhibits almost no change.

functions were added. This change is depicted in Fig. 2. As can be seen in this figure, LWR shows virtually no change of the learned parameters of the initial three basis functions, nLWR has a modest change, while both RBF methods have drastic changes. This behavior in cooperative learning and local learning with normalized weights is not desirable since it requires too much re-organization of previously learned parameters during structure learning. Such reorganization will lead to a transient decrease of performance of the function approximator, and is particularly dangerous for on-line learning adaptive control. Moreover, the strong cooperation of RBF networks can also lead to negative interference over the domain of the approximator, i.e. the unlearning of previously useful knowledge, when an outlier data point is added in some part of the workspace, or when training data distributions change over time. In LWL, such negative interference always remains strictly localized (Schaal & Atkeson, 1998). As a last point, RBF networks and nLWR make it hard to learn the optimal distance metric $\mathbf{D}_k$ in (5), since in this case the partial derivatives of the cost functions (8) and (12), respectively, w.r.t. the distance metric have to be taken through all basis functions—in LWR, this derivative remains localized to each local model.

When adding six more local models to the learning systems, i.e. the total number of 13 models, the advantages of LWR for structure learning become even more apparent (Fig. 1). RBF and nRBF both start overfitting[6], while LWL methods improve the approximation result slightly due to averaging over more models. Thus, this empirical example nicely demonstrates how LWL methods differ from cooperative learning, and why the properties of LWL are very useful for structure learning for on-line adaptive control.

[6] It should be noted that in the example above, rather small kernels are used in the RBF systems and we can improve the fit if the kernels are widened. However, as we increase the kernel size, the change of the coefficients (cf. Fig. 2) becomes even more drastic and learning becomes increasingly numerically ill-defined due to an ill-conditioned matrix inversion in the RBF learning algorithm. Large kernels also entail a large danger of negative interference during incremental learning (Schaal and Atkeson, 1998).

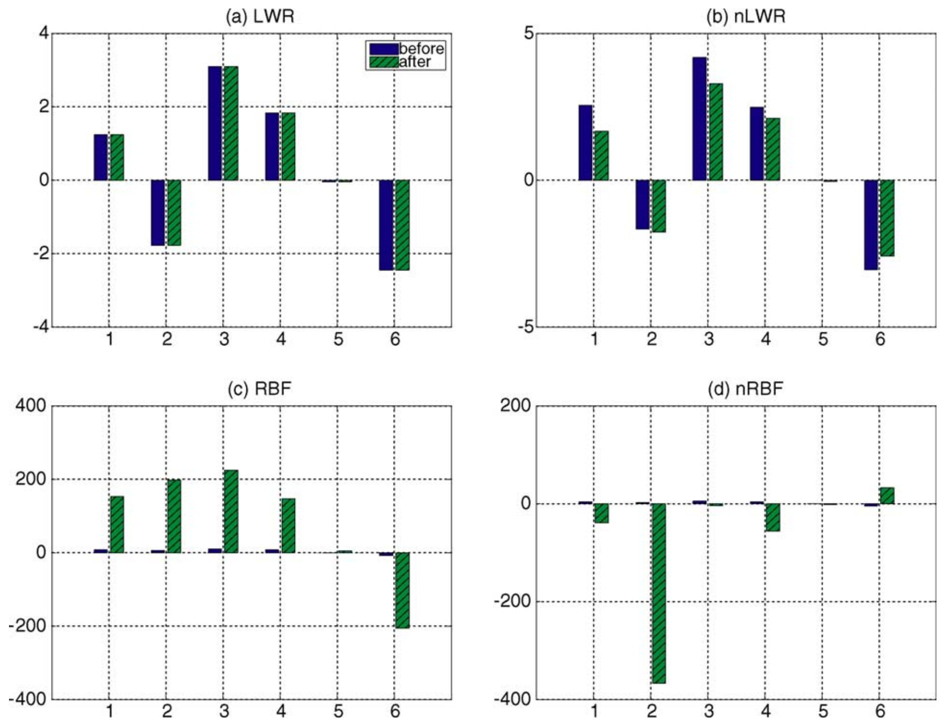

Fig. 2. Comparison of the coefficients of the local models centered at $c_k = -0.8$, 0 and 0.8 before and after adding four new local models. The left bar in each index shows the coefficient before adding the new local models and the right bar in each index shows the coefficient after doing so. The indices 1–3 correspond to the slope parameter b_k of each local model, while the indices 4–6 correspond to the intercept $b_{0,k}$. Only LWR (a) shows virtually no changes of coefficients, while nLWR (b) shows a modest change and RBF (c) and nRBF (d) methods show drastic changes (notice the difference in the scale of the vertical axis).

3. Nonlinear learning adaptive control

We will now turn to the application of RFWR in learning nonlinear adaptive control. As we have mentioned, our objective is to design a provably stable adaptive controller with on-line learning of unknown nonlinearities of the system dynamics which achieves accurate tracking and rapid adaptation.

In this section, we first present locally linear approximation of the nonlinear functions of the plant dynamics. Then, we review tracking error-based learning adaptive control, and finally we generalize to composite learning adaptive control with the locally weighted learning framework.

3.1. System dynamics and locally linear approximation

In this section, we present initial mathematical developments of our learning adaptive control for a class of first order SISO systems of the form

$$\dot{x} = f(x) + g(x)u \tag{14}$$

where $x \in \Re$ is a state and $u \in \Re$ is a control input. In Sections 5 and 6, we generalize the proposed algorithm to higher order SISO systems and discuss the application to MIMO problems, respectively.

Using a linear parameterization, for $x \in \mathcal{D}$, $f(x)$ and $g(x)$ in the system dynamics (14) can be represented as

$$f(x) = \frac{\sum_{k=1}^{N} w_k(x) y_{f,k}(x)}{\sum_{k=1}^{N} w_k(x)} + \Delta_f(x) \tag{15}$$

$$g(x) = \frac{\sum_{k=1}^{N} w_k(x) y_{g,k}(x)}{\sum_{k=1}^{N} w_k(x)} + \Delta_g(x) \tag{16}$$

where $w_k(x)$ is the biquadratic kernel (6) as its compact support $\mathcal{S}_k$ yields computational savings due to the fact that points outside the support of the local model do not contribute to the local model. $\Delta_f(x)$ and $\Delta_g(x)$ are the approximation errors for the approximation structure above. We assume that $\mathcal{D}$ is compact and that f and g are smooth on $\mathcal{D}$. We also assume that the sign of $g(x)$ is positive and $g(x)$ is lower bounded by a known constant g_l such that

$$g(x) \geq g_l > 0 \quad \forall x \in \mathcal{D}. \tag{17}$$

An equivalent assumption for the case when $g(x)$ is negative can be considered.

The functions $y_{f,k}(x)$ and $y_{g,k}(x)$ are locally linear models such that

$$y_{f,k}(x) = \bar{\mathbf{x}}_k^T \boldsymbol{\theta}_{f,k}, \quad y_{g,k}(x) = \bar{\mathbf{x}}_k^T \boldsymbol{\theta}_{g,k} \tag{18}$$

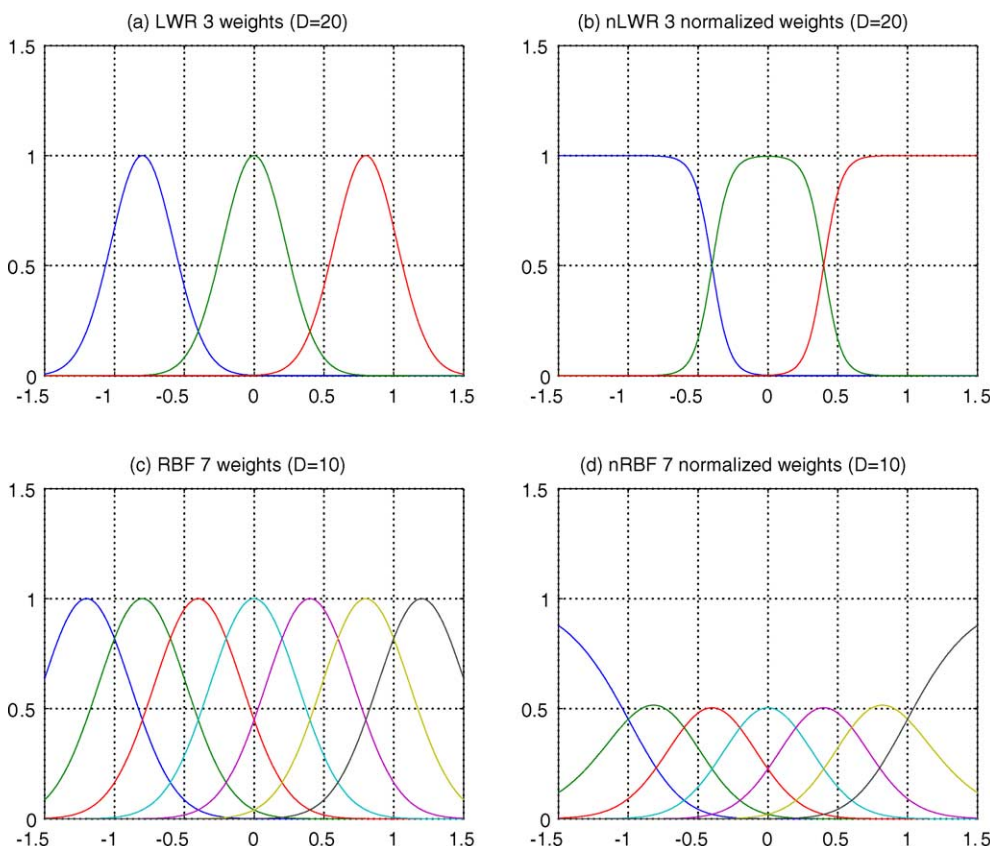

Fig. 3. Examples of the weighting functions. (a) three Gaussian weights (**D** = 20 in (5)) used for LWR. (b) three normalized Gaussian weights (**D** = 20) used for nLWR. (c) seven Gaussian functions (**D** = 10) used for RBF fitting. (d) seven normalized Gaussian functions (**D** = 10) used for RBF fitting with normalized weights.

where

$$\bar{\mathbf{x}}_k = \begin{bmatrix} x - c_k \\ 1 \end{bmatrix}, \quad \boldsymbol{\theta}_{f,k} = \begin{bmatrix} b_{f,k} \\ b_{0f,k} \end{bmatrix}, \quad \boldsymbol{\theta}_{g,k} = \begin{bmatrix} b_{g,k} \\ b_{0g,k} \end{bmatrix}. \tag{19}$$

For future notational convenience, $f(x)+g(x)u$ can be reformulated as

$$\begin{aligned} f(x) + g(x)u &= \frac{\sum_{k=1}^N w_k(\bar{\mathbf{x}}_k^T \boldsymbol{\theta}_{f,k} + u\bar{\mathbf{x}}_k^T \boldsymbol{\theta}_{g,k})}{\sum_{k=1}^N w_k} + \Delta_f + \Delta_g u \\ &= \frac{\sum_{k=1}^N w_k \boldsymbol{\chi}_k^T \boldsymbol{\Theta}_k}{\sum_{k=1}^N w_k} + \Delta_f + \Delta_g u \end{aligned} \tag{20}$$

where

$$\boldsymbol{\chi}_k = \begin{bmatrix} \bar{\mathbf{x}}_k \\ u\bar{\mathbf{x}}_k \end{bmatrix}, \quad \boldsymbol{\theta}_k = \begin{bmatrix} \boldsymbol{\theta}_{f,k} \\ \boldsymbol{\theta}_{g,k} \end{bmatrix}. \tag{21}$$

With the definition of the normalized weights $\bar{w}_k$

$$\bar{w}_k(x) = \frac{w_k(x)}{\sum_{k=1}^N w_k(x)}, \tag{22}$$

the function f also can be expressed as

$$\begin{aligned} f(x) &= \frac{\sum_{k=1}^N w_k(x) y_{f,k}(x)}{\sum_{k=1}^N w_k(x)} + \Delta_f(x) \\ &= \sum_{k=1}^N \bar{w}_k(x) y_{f,k}(x) + \Delta_f(x), \end{aligned} \tag{23}$$

$$f(x) = \boldsymbol{\phi}^T(x)\boldsymbol{\theta}_f + \Delta_f(x) \tag{24}$$

where $\boldsymbol{\phi}$ is the vector of normalized weighted basis functions defined by

$$\boldsymbol{\phi}(x) = \begin{bmatrix} \bar{w}_1(x)\bar{\mathbf{x}}_1 \\ \vdots \\ \bar{w}_N(x)\bar{\mathbf{x}}_N \end{bmatrix}, \tag{25}$$

and $\boldsymbol{\theta}_f = [\boldsymbol{\theta}_{f,1}^T, \ldots, \boldsymbol{\theta}_{f,N}^T]^T$. Similar definitions apply to $g(x)$ and its corresponding components. If we define the augmented basis function and parameter vector, respectively, by

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\phi} \\ u\boldsymbol{\phi} \end{bmatrix}, \quad \text{and } \boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}_f \\ \boldsymbol{\theta}_g \end{bmatrix} \tag{26}$$

then $f(x)+g(x)u$ in (20) can be further expressed as

$$f(x) + g(x)u = \mathbf{\Phi}^T\mathbf{\Theta} + \Delta_f + \Delta_g u. \tag{27}$$

For the following analyses, it will be useful to derive an error bound on the locally linear approximation of each model. We define the approximation error that we would like to achieve asymptotically as

$$\varepsilon_{f,k}(x) = \begin{cases} f(x) - y_{f,k}(x) & \text{on } \mathcal{S}_k \\ 0 & \text{otherwise} \end{cases} \tag{28}$$

where

$$\boldsymbol{\theta}_{f,k} = \underset{\hat{\boldsymbol{\theta}}_{f,k}}{\operatorname{argmin}} \left(\int_{-\infty}^{+\infty} w_k(x) \| f(x) - \tilde{\mathbf{x}}_k^T \hat{\boldsymbol{\theta}}_{f,k} \|^2 \, dx \right). \tag{29}$$

By the asymptotic properties of locally weighted regression (Schaal & Atkeson, 1997), the upper bound of the asymptotic locally linear fitting error (28) can be derived under the assumption that terms higher than of quadratic order in a Taylor series expansion of $f(x)$ around $x=c_k$ can be locally neglected in $\mathcal{S}_k$. This upper bound can be kept small if the shape and size of local model is chosen to be narrow relative to the local curvature the function (second derivative). As we will discuss in Section 3.5, the shape and size of the local models is optimized on-line according to the local curvature of the function to be approximated, i.e. the shape of the receptive field becomes narrow in the direction of high second derivatives (Hessian) of the function. Solving (23) for the approximation error and exploiting the property $\sum_{k=1}^N \bar{w}_k(x) = 1$ of the normalized weights results in

$$\begin{aligned} |\Delta_f(x)| &= \left| f(x) - \sum_{k=1}^N \bar{w}_k(x) y_{f,k}(x) \right| \\ &= \left| \sum_{k=1}^N \bar{w}_k(x)(f(x) - y_{f,k}(x)) \right| \\ &\le \sum_{k=1}^N \bar{w}_k(x) |\varepsilon_{f,k}(x)| \end{aligned} \tag{30}$$

$$|\Delta_f(x)| \le \max_k (|\varepsilon_{f,k}|) \overset{\text{def}}{=} \bar{\varepsilon}_f. \tag{31}$$

Thus, if each local linear model has accuracy $\varepsilon_{f,k}$, then the global fit on $\mathcal{D}$ also achieves at least accuracy $\bar{\varepsilon}_f$ (Choi & Farrell, 2000), and let $\bar{\Delta}_f$ denote the upper bound of $|\Delta_f|$. Similar definitions and properties presented above apply to $g(x)$ and a corresponding $\varepsilon_{g,k}$ and $\bar{\varepsilon}_g$.

3.2. Tracking error-based learning adaptive control

3.2.1. Control law

Tracking error-based learning control solely employs the tracking error $e=x-x_d$ as the means to adjust the parameters of the adaptive controller, where $x_d(t)$ denotes a desired trajectory that is smooth and also has a smooth derivative $\dot{x}_d$. If the function $f(x)$ and $g(x)$ in the nonlinear dynamics (14) were known, the control law

$$u = \frac{1}{g(x)}(-f(x) + \dot{x}_d - K(x - x_d)) \tag{32}$$

could achieve asymptotic perfect tracking, with tracking error dynamics

$$\dot{e} = -Ke \tag{33}$$

where K is a positive constant. Our goal is to design an on-line approximation-based controller capable of accurate tracking when nonlinearities in the dynamical system are unknown. The following development of a tracking error-based learning adaptive controller follows (Choi & Farrell, 2000; Slotine & Li, 1991).

When f and g are unknown, consider the control law

$$u = \frac{1}{\hat{g}(x)}(-\hat{f}(x) + \dot{x}_d - K(x - x_d)) \tag{34}$$

where $\hat{f}$ and $\hat{g}$ are estimates of f and g, respectively, defined by

$$\hat{f}(x) = \frac{\sum_{k=1}^N w_k(x)\hat{y}_{f,k}(x)}{\sum_{k=1}^N w_k(x)}, \quad \hat{g}(x) = \frac{\sum_{k=1}^N w_k(x)\hat{y}_{g,k}(x)}{\sum_{k=1}^N w_k(x)} \tag{35}$$

and

$$\hat{y}_{f,k}(x) = \tilde{\mathbf{x}}_k^T \hat{\boldsymbol{\theta}}_{f,k}, \ \hat{y}_{g,k}(x) = \tilde{\mathbf{x}}_k^T \hat{\boldsymbol{\theta}}_{g,k} \tag{36}$$

where

$$\hat{\boldsymbol{\theta}}_{f,k} = \begin{bmatrix} \hat{b}_{f,k} \\ \hat{b}_{0f,k} \end{bmatrix}, \quad \hat{\boldsymbol{\theta}}_{g,k} = \begin{bmatrix} \hat{b}_{g,k} \\ \hat{b}_{0g,k} \end{bmatrix}. \tag{37}$$

Define an estimate of the augmented parameter vector of (21) and (26), respectively, as

$$\hat{\mathbf{\Theta}}_k = \begin{bmatrix} \hat{\boldsymbol{\theta}}_{f,k} \\ \hat{\boldsymbol{\theta}}_{g,k} \end{bmatrix} \text{and} \quad \hat{\mathbf{\Theta}} = \begin{bmatrix} \hat{\boldsymbol{\theta}}_f \\ \hat{\boldsymbol{\theta}}_g \end{bmatrix} \tag{38}$$

for future use. Note that the adaptation law must ensure that $\hat{g}(x) \ge g_l > 0$ to avoid $\hat{g}(x)=0$. This issue will be discussed in Section 3.3.2. With the choice of the control law (34), the tracking error dynamics for $e=x-x_d$ become

$$\begin{aligned} \dot{e} &= -Ke + (f - \hat{f}) + (g - \hat{g})u \\ &= -Ke - \boldsymbol{\phi}^T(\tilde{\boldsymbol{\theta}}_f + \tilde{\boldsymbol{\theta}}_g u) + \Delta_f + \Delta_g u \end{aligned} \tag{39}$$

where $\tilde{\boldsymbol{\theta}}_f$ is the parameter error vector defined as $\tilde{\boldsymbol{\theta}}_f = \hat{\boldsymbol{\theta}}_f - \boldsymbol{\theta}_f$, and a similar definition applies to $\tilde{\boldsymbol{\theta}}_g$.

3.2.2. Tracking error based adaptation law

If we select the tracking error-based adaptation law

$$\dot{\hat{\mathbf{\Theta}}} = \mathbf{\Gamma}\mathbf{\Phi}(x)e(t) \tag{40}$$

where $\mathbf{\Gamma}$ is a positive definite adaptation gain matrix, and the Lyapunov function

$$V = \frac{1}{2}e^2 + \frac{1}{2}\tilde{\mathbf{\Theta}}^T \mathbf{\Gamma}^{-1}\tilde{\mathbf{\Theta}} \tag{41}$$

where $\tilde{\mathbf{\Theta}}$ is the parameter error vector $\tilde{\mathbf{\Theta}} = \hat{\mathbf{\Theta}} - \mathbf{\Theta}$, it is possible to prove the following desirable properties:[7]

1. If $\Delta_f(x)$ and $\Delta_g(x)$ are identically zero, then $e \in L_2$ with e, $\hat{\mathbf{\Theta}}, \tilde{\theta} \in L_\infty$.
2. When $\Delta_f(x)$ and $\Delta_g(x)$ are not the zero function, then e is uniformly ultimately bounded with a small bound determined in part by $\bar{\varepsilon}_f$ and $\bar{\varepsilon}_g$. When there is approximation error, (40) must be modified by a deadzone, epsilon or sigma modification (Choi & Farrell, 2000) to maintain stability.
3. The parameter $K > 0$ determines the rate of decay of the tracking error (e.g. due to initial conditions or disturbances).
4 The uniform ultimate bound on $|e|$ is inversely related to K. Note that the designer can decrease the bound on $|e|$ by (a) increasing K or (b) decreasing Δ_f and Δ_g. Increasing K is not necessarily desirable as this increases both the closed loop bandwidth and the magnitude of control signal. Decreasing Δ_f and Δ_g can be achieved by on-line adaptation of the approximator structure, which will be discussed in Section 3.5. This measure effects neither the closed loop bandwidth or the magnitude of the control signal.

The proof of these comments is a special case of the proof presented in Section 3.4; therefore, it is not included here. Note that the adaptation law must ensure that $\hat{g}(x) \geq g_l > 0$ to avoid $\hat{g}(x) = 0$. For this purpose, we discuss how to constrain the parameter update law using the parameter projection method in Section 3.3.2. If $\mathbf{\Gamma}$ is block diagonal, with blocks denoted by $\mathbf{\Gamma}_k$, then the tracking error adaptation law becomes a local update:

$$\dot{\hat{\mathbf{\Theta}}} = \mathbf{\Gamma}\mathbf{\Phi}(x)e(t) \rightarrow \dot{\hat{\mathbf{\Theta}}}_k = \mathbf{\Gamma}_k \mathbf{\Phi}_k(x)e(t) = \mathbf{\Gamma}_k \bar{w}_k \boldsymbol{\chi}_k e(t). \tag{42}$$

where $\mathbf{\Phi}_k = \begin{bmatrix} \boldsymbol{\phi}_k \\ \boldsymbol{\phi}_k u \end{bmatrix}$. In this case, the Lyapunov function is equivalently given by

$$V = \frac{1}{2}e^2 + \sum_{k=1}^{N} \frac{1}{2}\tilde{\mathbf{\Theta}}_k^T \mathbf{\Gamma}_k^{-1}\tilde{\mathbf{\Theta}}_k, \tag{43}$$

where $\tilde{\mathbf{\Theta}}_k^T = \hat{\mathbf{\Theta}}_k - \mathbf{\Theta}_k$, which yields the same properties listed above.

[7] Given a function u: $[0, \infty) \rightarrow \Re$, L_2 denotes the space of piecewise continuous, square-integrable functions with the norm $\|u\|_{L_2} = \sqrt{\int_0^\infty u^T(t)u(t)\mathrm{d}t} < \infty$ and L_∞ denotes the space of piecewise continuous, uniformly bounded functions with the norm $\|u\|_{L_\infty} = \sup_{t \geq 0}\|u(t)\| < \infty$ (Khalil, 1996).

3.3. Composite learning adaptive control

While tracking error-based learning adaptive control is a theoretically sound and successful technique, it does not take full advantage of the power of a modern statistical learning framework such as RFWR: the previous section had no component that distinguished using an RFWR approach from a cooperative learning approach (cf. Section 2). The main thrust of this paper is to make use of some additional statistical information that can be collected during learning such that convergence of the adaptive controller is accelerated and structure adaptation of the learning system can be accomplished in a stable and principled way. For this purpose, we need to turn to a composite learning adaptive control approach (Slotine & Li, 1991) where, besides the tracking error, the prediction error is explicitly incorporated in the parameter update law of the controller.

3.3.1. Continuous time parameter estimation with RFWR

The key idea of composite adaptive control is to augment the parameter update law of the tracking error-based controller with a term that is explicitly driven by the prediction error. For the purpose of stability proofs, we will first develop a continuous time parameter update for RFWR in this subsection, before incorporating this result in the final adaptive control law in the next subsection. We will assume that both the state and state derivative are measurable, which could be relaxed by some pre-filtering or identifier techniques (Narendra & Annaswamy, 1989; Slotine & Li, 1991) in future work.

The basic idea of RFWR is to minimize the individual weighted squared prediction error for each locally linear model as given in (8). In a continuous time formulation, this cost function becomes

$$J_k = \int_0^t w_k(x(\tau))\|y(\tau) - \hat{y}_k(x(\tau), u(\tau))\|^2 \mathrm{d}\tau, \tag{44}$$

for $k = 1, \ldots, K$,

where, $y(t) = \dot{x}(t) = f(x(t)) + g(x(t))u(t)$ and $\hat{y}_k = \hat{\mathbf{\Theta}}_k^T \boldsymbol{\chi}_k$. For notational compactness, we will avoid writing the explicit time or state dependence of variables in the following unless needed.

The minimization of J_k yields a gradient

$$\frac{\partial J_k}{\partial \hat{\mathbf{\Theta}}_k} = -2\int_0^t w_k(\tau)(y(\tau) - \hat{\mathbf{\Theta}}_k^T(t)\boldsymbol{\chi}_k(\tau))\boldsymbol{\chi}_k^T(\tau)\mathrm{d}\tau. \tag{45}$$

By setting $\partial J_k / \partial \hat{\mathbf{\Theta}}_k = 0$ and rearranging the equation, we obtain

$$\int_0^t w_k(\tau)\boldsymbol{\chi}_k(\tau)^T y(\tau)\mathrm{d}\tau = \hat{\mathbf{\Theta}}_k^T(t)\int_0^t w_k(\tau)\boldsymbol{\chi}_k(\tau)\boldsymbol{\chi}_k^T(\tau)\mathrm{d}\tau \tag{46}$$

Defining $\mathbf{P}_k^{-1}(t) = \int_0^t w_k(\tau)\boldsymbol{\chi}_k(\tau)\boldsymbol{\chi}_k^T(\tau)\mathrm{d}\tau$, one can write[8] an update law for $\mathbf{P}$ as

$$\dot{\mathbf{P}}_k = -w_k(\tau)\mathbf{P}_k\boldsymbol{\chi}_k(\tau)\boldsymbol{\chi}_k^T(\tau)\mathbf{P}_k \tag{47}$$

where we used

$$\frac{d}{\mathrm{d}t}(\mathbf{P}_k^{-1}) = w_k(\tau)\boldsymbol{\chi}_k(\tau)\boldsymbol{\chi}_k^T(\tau). \tag{48}$$

Finally, taking the transpose and the time derivative of (46) and solving for $\dot{\hat{\boldsymbol{\Theta}}}_k$ yields the parameter update

$$\dot{\hat{\boldsymbol{\Theta}}}_k = w_k\mathbf{P}_k\boldsymbol{\chi}_k(y-\hat{y}_k) = w_k\mathbf{P}_k\boldsymbol{\chi}_k e_{pk} \tag{49}$$

where e_{pk} is the prediction error of the k-th linear model defined by

$$e_{pk} = y - \hat{y}_k \tag{50}$$

which, as mentioned before, assumes that $y = \dot{x}$ is explicitly measured.

Exponential forgetting as in (10) can be locally incorporated by using the modified update law

$$\dot{\mathbf{P}}_k = \lambda\mathbf{P}_k - w_k\mathbf{P}_k\boldsymbol{\chi}_k\boldsymbol{\chi}_k^T\mathbf{P}_k, \text{ where } \lambda > 0, \tag{51}$$

applied only when x falls in the support of the local model, i.e. $w_k(x) > 0$. Thus, Eqs. (49) and (51) form the continuous time equivalent of Eqs. (9) and (10), respectively.

Using the Lyapunov function

$$V = \frac{1}{2}\sum_{k=1}^{N}\tilde{\boldsymbol{\Theta}}_k^T\mathbf{P}_k^{-1}\tilde{\boldsymbol{\Theta}}_k, \tag{52}$$

we can show the following important properties:

1. In the case that Δ_f and Δ_g are identically zero, the total prediction error $e_p = \sum_{k=1}^N w_k e_{pk} / \sum_{k=1}^N w_k$ approaches zero asymptotically and $\tilde{\boldsymbol{\Theta}}_k \in L_\infty\ \forall k$.
2. In the case where Δ is not identically zero, the prediction error asymptotically converges to a region of uniform boundedness. This bound is determined in part by $\bar{\varepsilon}_f$ and $\bar{\varepsilon}_g$.

$$\mathrm{Proj}\{\tau\} = \begin{cases} \tau & \text{if } \hat{\boldsymbol{\Theta}}_k \in \mathring{\Pi}_k \text{ or } (\nabla_{\hat{\boldsymbol{\Theta}}_k}\wp_k)^T\tau \le 0 \\ \left(I - c(\hat{\boldsymbol{\Theta}}_k)\mathbf{P}_k\dfrac{(\nabla_{\hat{\boldsymbol{\Theta}}_k}\wp_k)(\nabla_{\hat{\boldsymbol{\Theta}}_k}\wp_k)^T}{(\nabla_{\hat{\boldsymbol{\Theta}}_k}\wp_k)^T\mathbf{P}_k(\nabla_{\hat{\boldsymbol{\Theta}}_k}\wp_k)}\right)\tau & \text{if } \hat{\boldsymbol{\Theta}}_k \in \Pi_{\varepsilon,k}\backslash\mathring{\Pi}_k \text{ and } (\nabla_{\hat{\boldsymbol{\Theta}}_k}\wp_k)^T\tau > 0 \end{cases} \tag{57}$$

These claims are provable as a special case of the proofs included in the next section.

3.3.2. Composite adaptation law with parameter projection

We are now equipped to formulate the parameter update law for a composite learning adaptive controller using RFWR and to present its stability analysis. Consider a parameter update for the control law (34) in which the tracking error-based adaptation (42) and RFWR-based adaptation (49) and (51) are combined.

$$\dot{\hat{\boldsymbol{\Theta}}}_k = \mathbf{P}_k\boldsymbol{\chi}_k(\bar{w}_k e + w_k e_{pk}) \tag{53}$$

$$\dot{\mathbf{P}}_k = \lambda\mathbf{P}_k - w_k\mathbf{P}_k\boldsymbol{\chi}_k\boldsymbol{\chi}_k^T\mathbf{P}_k \tag{54}$$

Note that forgetting in (54) is localized, i.e. $\mathbf{P}_k$ in (54) is updated only when $w_k(x) > 0$.

The parameter update needs to be constrained within the following convex set, Π_k, for each locally linear model to satisfy the assumption for $g(x)$ in (17)

$$\Pi_k = \left\{\hat{\boldsymbol{\Theta}}_k \,|\, \wp_k(\hat{\boldsymbol{\Theta}}_k) \overset{\text{def}}{=} g_l - \left(-\frac{|\hat{b}_{g,k}|}{\sqrt{D_k}} + \hat{b}_{0g,k}\right) \le 0\right\} \tag{55}$$

where $\wp_k(\hat{\boldsymbol{\Theta}}_k)$ is a convex function, and D_k is the distance metric for a scalar state x in our SISO system. This set is derived considering the minimum of each locally linear model at the boundary of the local model by solving the constrained optimization problem outlined in Appendix A where $g(x) = \hat{b}_{g,k}(x-c_k) + \hat{b}_{0g,k}$ and $\mu = 1$ in (A1). Let us denote the interior of Π_k by $\mathring{\Pi}_k$ and the boundary by $\partial\Pi_k$. In order to ensure that $\hat{g}(x) \ge g_l > 0$, the update law is modified by the parameter projection (Krstić, Kanellakopoulos, & Kokotović, 1995) as

$$\dot{\hat{\boldsymbol{\Theta}}}_k = \mathrm{Proj}\{\mathbf{P}_k\boldsymbol{\chi}_k(\bar{w}_k e + w_k e_{pk})\} \tag{56}$$

where the standard projection operator $\mathrm{Proj}\{\boldsymbol{\tau}\}$ projects the vector $\boldsymbol{\tau}$ onto the hyperplane tangent to the $(\partial\Pi_k$ at $\hat{\boldsymbol{\Theta}}_k$ when $\hat{\boldsymbol{\Theta}}_k$ is at the boundary with $\boldsymbol{\tau}$ pointing outward. Since this standard projection operator is generally discontinuous, we specifically use the smooth parameter projection (Krstić et al., 1995) defined by

$$c(\hat{\boldsymbol{\Theta}}_k) = \min\left\{1, \frac{\wp_k(\hat{\boldsymbol{\Theta}}_k)}{\varepsilon}\right\} \tag{58}$$

where

$$\Pi_{\varepsilon,k} = \{\hat{\boldsymbol{\Theta}}_k \,|\, \wp_k(\hat{\boldsymbol{\Theta}}_k) \le \varepsilon\} \tag{59}$$

which is a union of the set Π and an $O(\varepsilon)$-boundary layer around it. Note that $(\nabla_{\hat{\boldsymbol{\Theta}}_k}\wp_k)$ represents an outward normal vector at $\hat{\boldsymbol{\Theta}}_k \in \partial\Pi_k$.

By the properties of the projection operator (Krstić et al., 1995) (see Lemma E.1 (*iii*) on page 512), the solution of $\dot{\hat{\boldsymbol{\Theta}}}_k = \mathrm{Proj}\{\tau\}$ is guaranteed to remain in $\Pi_{\varepsilon,k}$ assuming that $\hat{\boldsymbol{\Theta}}_k(0)$ is chosen to be in $\Pi_{\varepsilon,k}$. The complete proof requires that these update laws be modified by appropriate deadzones to prevent parameter adaptation when the prediction

[8] Note the following identity: $\frac{d}{dt}(\mathbf{P}\mathbf{P}^{-1}) = \dot{\mathbf{P}}\mathbf{P}^{-1} + \mathbf{P}\dot{\mathbf{P}}^{-1} = 0$, which implies that $\dot{\mathbf{P}}^{-1} = -\mathbf{P}^{-1}\dot{\mathbf{P}}\mathbf{P}^{-1}$.

and tracking errors become sufficiently small as discussed the next section.

3.4. Stability analysis

For our stability analyses, we choose the Lyapunov function

$$V = \frac{1}{2}e^2 + \sum_{k=1}^{N} \frac{1}{2}\tilde{\mathbf{\Theta}}_k^T \mathbf{P}_k^{-1} \tilde{\mathbf{\Theta}}_k. \tag{60}$$

which makes it clear that there are $N+1$ terms in the Lyapunov function, one for the tracking error and one for each local model The weights w_k appear in the parameter update law, but not in the Lyapunov function. The weights ensure that the parameter update law adjusts only those linear models appropriate to the present operating point so that only the corresponding terms in the Lyapunov function are decreasing. All the remaining terms of the Lyapunov function $\tilde{\mathbf{\Theta}}_k^T \mathbf{P}_k^{-1} \tilde{\mathbf{\Theta}}_k$ for which $w_k(x)=0$ will be held constant.

The following provides a stability analysis of our RFWR-based composite learning controller for the case where the state derivative is directly measured. Using the parameter adaptation law with projection (56) and (54) (outside the deadzone defined in (63)), the time derivative of the Lyapunov function is[9]

$$\begin{aligned}
\dot{V} &= e\dot{e} + \sum_{k=1}^{N} \tilde{\mathbf{\Theta}}_k^T \mathbf{P}_k^{-1} \dot{\tilde{\mathbf{\Theta}}}_k + \frac{1}{2}\sum_{k=1}^{N} \tilde{\mathbf{\Theta}}_k^T \dot{\mathbf{P}}_k^{-1} \tilde{\mathbf{\Theta}}_k \\
&= e\dot{e} + \sum_{k=1}^{N} \tilde{\mathbf{\Theta}}_k^T \mathbf{P}_k^{-1} \operatorname{Proj}\{\mathbf{P}_k \boldsymbol{\chi}_k (\bar{w}_k e + w_k e_{pk})\} \\
&\quad + \frac{1}{2}\sum_{k=1}^{N} \tilde{\mathbf{\Theta}}_k^T \dot{\mathbf{P}}_k^{-1} \tilde{\mathbf{\Theta}}_k \\
&\le e\dot{e} + \sum_{k=1}^{N} \tilde{\mathbf{\Theta}}_k^T \mathbf{P}_k^{-1} \mathbf{P}_k \boldsymbol{\chi}_k (\bar{w}_k e + w_k e_{pk}) \\
&\quad + \frac{1}{2}\sum_{k=1}^{N} \tilde{\mathbf{\Theta}}_k^T \dot{\mathbf{P}}_k^{-1} \tilde{\mathbf{\Theta}}_k \\
&= -Ke^2 - \frac{1}{2}\lambda \sum_{k=1}^{N} \tilde{\mathbf{\Theta}}_k^T \mathbf{P}_k^{-1} \tilde{\mathbf{\Theta}}_k - \frac{1}{2}\sum_{k=1}^{N} w_k e_{pk}^2 \\
&\quad + \frac{1}{2}\sum_{k=1}^{N} w_k (\varepsilon_{f,k} + \varepsilon_{g,k} u)^2 + e(\Delta_f + \Delta_g u)^2 \\
&\le -\frac{K}{2}e^2 - \frac{1}{2}\lambda \sum_{k=1}^{N} \tilde{\mathbf{\Theta}}_k^T \mathbf{P}_k^{-1} \tilde{\mathbf{\Theta}}_k - \frac{1}{2}\sum_{k=1}^{N} w_k e_{pk}^2 \\
&\quad + \frac{1}{2}\sum_{k=1}^{N} w_k (\varepsilon_{f,k} + \varepsilon_{g,k}|u|)^2 + \frac{1}{2K}(\Delta_f + \Delta_g |u|)^2
\end{aligned} \tag{61}$$

[9] The proof uses the identities $2e\Delta = -(\alpha e - \frac{1}{\alpha}\Delta)^2 + \alpha^2 e^2 + (\frac{1}{\alpha})^2 \Delta^2$, $\forall \alpha \ne 0$, and $\alpha^2 = K$.

We can summarize these stability results as follows:

Perfect approximation. When perfect approximation is assumed such as $\bar{\varepsilon}_f = \bar{\varepsilon}_g = 0$, the time derivative of the Lyapunov function simplifies to

$$\dot{V} \le -\frac{K}{2}e^2 - \frac{1}{2}\lambda \sum_{k=1}^{N} \tilde{\mathbf{\Theta}}_k^T \mathbf{P}_k^{-1} \tilde{\mathbf{\Theta}}_k - \frac{1}{2}\sum_{k=1}^{N} w_k e_{pk}^2. \tag{62}$$

Thus, we have $\dot{V} \le 0$, which implies that e and e_p converge to zero and that e, $e_p \in L_2$ by Barbalat's Lemma (Slotine & Li, 1991).

Imperfect approximation. When $\bar{\varepsilon}_f \ne 0$ or $\bar{\varepsilon}_g \ne 0$, we have ultimate boundedness of the prediction and tracking errors. The region of ultimate boundedness can be decreased either by increasing K or by decreasing $\bar{\varepsilon}_f$ and $\bar{\varepsilon}_g$. Since K is related to the bandwidth of the control law, there can be undesirable consequences (e.g. exciting unmodelled dynamics) if it is increased. Alternatively, $\bar{\varepsilon}_f$ and $\bar{\varepsilon}_g$ are decreased by improving the structure of the approximator. This can be accomplished on-line, as described in Section 3.5.

In the Lyapunov analysis in (61), there are terms associated with the approximation error of $g(x)$ multiplied by the control input such as $\varepsilon_{g,k}u$ and $\Delta_{g,k}u$. In (Choi & Farrell, 2000), this is treated by sliding mode control with a sufficiently large gain. However, we prefer not to use the sliding control since it could cause a chattering problem. Alternatively, we can design a time varying adaptation deadzone such that the parameter is updated only when

$$Ke^2 + \sum_{k=1}^{N} w_k e_{pk}^2 > \sum_{k=1}^{N} w_k (\bar{\varepsilon}_f + \bar{\varepsilon}_g |u(t)|)^2 + \frac{1}{K}(\bar{\Delta}_f + \bar{\Delta}_g |u(t)|)^2. \tag{63}$$

3.5. Structure adaptation of the function approximator

Given that both the input domain and the complexity of the function to be learned are unknown, we will exploit RFWR's automatic structure adaptation mechanisms to improve the quality of function approximation (Schaal & Atkeson, 1998), thus decreasing $\bar{\varepsilon}_f$ and $\bar{\varepsilon}_g$ above in order to tighten the bounds on the tracking error. Two mechanisms of structure adaptation are available, either by adding new local models or by adjusting the distance metric $\mathbf{D}_k$ of existing models. A new locally linear model is added if the input of a training point $\mathbf{x}$ does not activate any of the existing local models more than a threshold w_{gen}, and the center of the new receptive field is set to $\mathbf{c}_k = \mathbf{x}$.

The distance metric $\mathbf{D}_k$ of each local model, which is decomposed as the inner product of the upper triangular matrix $\mathbf{M}_k$ (i.e. $\mathbf{D}_k = \mathbf{M}_k^T \mathbf{M}_k$) to ensure positive definiteness, can be optimized independently of all other local models on-line by minimizing the penalized weighted mean squared error criterion

$$J_k = \frac{1}{W_k}\int_0^t w_k \|y - \hat{y}_k\|^2 \, \mathrm{d}\tau + \gamma \sum_{i,j=1}^{n} D_{k,ij}^2, \tag{64}$$

by gradient descent

$$\dot{\mathbf{M}}_k = -\alpha \frac{\partial J_k}{\partial \mathbf{M}_k} \tag{65}$$

where $W_k = \int_0^t w_k \, \mathrm{d}\tau$. The penalty term is introduced to avoid the problem of continuously shrinking receptive fields as the number of training data increases—while this shrinking property is statistically useful to approach zero error function approximation, it is computationally too expensive in practical implementations as an infinite number of local models would have to be maintained. As a result of distance metric adaptation, the shape and size of the receptive field is adjusted according to the local curvature of the function to be approximated, i.e. the shape of the receptive field becomes narrow in the direction of high second derivatives (Hessian) of the function. See (Schaal & Atkeson, 1997) for the details of the asymptotic properties.

By following (Schaal & Atkeson, 1998), the gradient descent derivative can be approximated as

$$\frac{\partial J_k}{\partial \mathbf{M}_k} \approx \frac{\partial w}{\partial \mathbf{M}} \int_0^t \frac{\partial J_{k,1}}{\partial w} \mathrm{d}\tau + \frac{w}{W}\frac{\partial J_{k,2}}{\partial \mathbf{M}} \tag{66}$$

where

$$J_{k,1} = \frac{1}{W_k}(w_k \|y - \hat{y}_k\|^2) \quad \text{and } J_{k,2} = \gamma \sum_{i,j=1}^{n} D_{k,ij}^2. \tag{67}$$

With memory traces W_k, E_k and $\mathbf{H}_k$

$$W_k = \int_0^t e^{-\lambda(t-\tau)} w_k \, \mathrm{d}\tau \tag{68}$$

$$E_k = \int_0^t e^{-\lambda(t-\tau)} w_k e_{pk}^2 \, \mathrm{d}\tau \tag{69}$$

$$\mathbf{H}_k = \int_0^t e^{-\lambda(t-\tau)} w_k \bar{\mathbf{x}}_k e_{pk} \, \mathrm{d}\tau \tag{70}$$

the derivative terms in (66)—under a biquadratic weighting kernel—become

$$\frac{\partial w_k}{\partial M_{k,rl}} = -2\sqrt{w_k}(\mathbf{x}_k - \mathbf{c}_k)^T \frac{\partial \mathbf{D}_k}{\partial M_{k,rl}}(\mathbf{x}_k - \mathbf{c}_k),$$
$$\frac{\partial J_{k,2}}{\partial M_{k,rl}} = 2\gamma \sum_{i,j=1}^{n} D_{k,ij} \frac{\partial D_{k,ij}}{\partial M_{k,rl}} \tag{71}$$

$$\frac{\partial D_{k,ij}}{\partial M_{k,rl}} = \delta_{rj} M_{k,il} + \delta_{ir} M_{k,jl} \; (\delta \text{ is the Kronecker operator}) \tag{72}$$

and

$$\int_0^t \frac{\partial J_{k,1}}{\partial w} \mathrm{d}\tau \approx -\frac{E_k}{W_k^2} + \frac{1}{W_k}(e_{pk}^2 - 2\bar{\mathbf{P}}_k \bar{\mathbf{x}}_k e_{pk} \otimes \mathbf{H}_k) \tag{73}$$

where the operator $\otimes$ denotes an element-wise multiplication of two homomorphic matrices or vectors with a subsequent summation of all coefficients such as $\mathbf{Q} \otimes \mathbf{V} = \sum Q_{ij} V_{ij}$, and $\bar{\mathbf{P}}_k$ denotes the inverted covariance matrix associated with the input $\bar{\mathbf{x}}_k$. The update equations for the memory terms result from integrating the following first order differential equation:

$$\dot{W}_k = -\lambda W_k + w_k \tag{74}$$

$$\dot{E}_k = -\lambda E_k + w_k e_{pk}^2 \tag{75}$$

$$\dot{\mathbf{H}}_k = -\lambda \mathbf{H}_k + w_k \bar{\mathbf{x}}_k e_{pk} \tag{76}$$

Structure adaptation affects the stability proofs of the previous section in different ways, depending on which kind of structure adaptation is considered: adding new local models or adapting distance metric of existing local models. Adding new local models can simply be regarded as starting to update some of the model parameters in a Lyapunov function (60) that has all possible local models allocated in advance—as mentioned in the previous section, due to the compact support of the local models, the Lyapunov function is only decreased in very few terms at a time, depending on where the current operating point of the control system is; all other terms are unchanged. Structure adaptation in terms of the distance metric is a nonlinear parameter optimization process, and only convergence of gradient descent to a local optimal solution may be shown with a local quadratic approximation of the cost function around its minimum. As noted in (Schaal & Atkeson, 1998), it is necessary to choose an appropriate initial distance metric such that the receptive field does not grow to span the entire input domain, which is an undesirable local minimum from the viewpoint of local learning. We prefer to consider structure adaptation due to the distance metric as a disturbance to the control system that happens on a much slower time scale than the adaptation of the parameters $\mathbf{\Theta}_k$—gradient descent in $\mathbf{D}_k$ is usually much slower than the second order recursive least squares methods for learning $\mathbf{\Theta}_k$.

4. Discrete-time formulation for practical implementation

So far, we have been concerned with the continuous-time formulation and theoretical analyses of the proposed learning adaptive controller. In continuous-time domains, the parameter update law is derived in the form of differential equations. Given that actual implementations are usually in discrete time, we will briefly present the discrete-time adaptation law.

The error criterion for locally weighted regression in continuous-time

$$J_k = \int_0^t w_k \|y - \hat{y}\|^2 \, d\tau \tag{77}$$

can equivalently be written in terms of a summation as

$$J_k = \lim_{\Delta t \to 0} \sum_{i=1}^{p} w_{k,i} \|y_i - \hat{y}_{k,i}\|^2 \Delta t, \text{ where } \Delta t = \frac{t}{p}. \tag{78}$$

Note that $p \to \infty$ as $\Delta t \to 0$.

When Δt is nonzero, the solution of locally weighted regression is given by

$$\hat{\boldsymbol{\Theta}}_k^n = \mathbf{P}_k^n \sum_{i=1}^{p} w_{k,i} \boldsymbol{\chi}_{k,i} y_i \tag{79}$$

where

$$\mathbf{P}_k^n = \left(\sum_{i=1}^{p} w_{k,i} \boldsymbol{\chi}_{k,i} \boldsymbol{\chi}_{k,i}^T \right)^{-1} \tag{80}$$

Note that Δt does not appear in these equations as it is canceled out. When a given sampled training data point ($\mathbf{x}$, y) at time t falls with in the support of a local model, i.e. $w_k > 0$, the incremental update law can thus be shown to yield

$$\hat{\boldsymbol{\Theta}}_k^{n+1} = \hat{\boldsymbol{\Theta}}_k^n + w_k \mathbf{P}_k^{n+1} \boldsymbol{\chi}_k e_{pk} \tag{81}$$

where

$$\mathbf{P}_k^{n+1} = \frac{1}{\lambda} \left(\mathbf{P}_k^n - \frac{\mathbf{P}_k^n \boldsymbol{\chi}_k \boldsymbol{\chi}_k^T \mathbf{P}_k^n}{\frac{\lambda}{w_k} + \boldsymbol{\chi}_k^T \mathbf{P}_k^n \boldsymbol{\chi}_k} \right) \text{ and } e_{pk} = y - \hat{\boldsymbol{\Theta}}_k^{nT} \boldsymbol{\chi}_k \tag{82}$$

with the inclusion of a forgetting factor $\lambda \in [0, 1]$. Not surprisingly, these formulae exactly correspond to the RFWR parameter update law given by (9) and (10) in Section 2.1. For distance metric adaptation, we use the leave-one cross validation error criterion with a penalty term

$$J_k = \frac{1}{W_k} \sum_{i=1}^{p} \frac{w_{k,i} \|y_i - \hat{y}_{k,i}\|^2}{(1 - w_{k,i} \boldsymbol{\chi}_{k,i}^T \mathbf{P}_k \boldsymbol{\chi}_{k,i})^2} + \gamma \sum_{i,j=1}^{n} D_{k,ij}^2, \tag{83}$$

for statistical reasons so that overfitting can be avoided for finite sampling frequencies Δt when training data contain measurement noise (Schaal & Atkeson, 1998).

In summary, the following adaptation law should be used for discrete-time implementations with time step Δt when a training data point falls in the support of the local model, i.e. $w_k(x) > 0$:

Parameter update

$$\hat{\boldsymbol{\Theta}}_k^{n+1} = \hat{\boldsymbol{\Theta}}_k^n + \mathbf{P}_k^{n+1} \boldsymbol{\chi}_k (\bar{w}_k e \, \Delta t + w_k e_{pk}) \tag{84}$$

Inverted covariance matrix update

$$\mathbf{P}_k^{n+1} = \frac{1}{\lambda} \left(\mathbf{P}_k^n - \frac{\mathbf{P}_k^n \boldsymbol{\chi} \boldsymbol{\chi}^T \mathbf{P}_k^n}{\frac{\lambda}{w_k} + \boldsymbol{\chi}^T \mathbf{P}_k^n \boldsymbol{\chi}} \right) \tag{85}$$

Distance metric update

$$\mathbf{M}_k^{n+1} = \mathbf{M}_k^n - \alpha \frac{\partial J_k}{\partial \mathbf{M}_k} \tag{86}$$

using the penalized leave-one-out cross validation error criterion (83). (see (Schaal & Atkeson, 1998) for the derivation of the gradient equations for the discrete-time case.) Note that the parameter update law and inverted covariance matrix update law above need to be modified by a deadzone introduced in (63) when the prediction and tracking errors become sufficiently small.

5. *n*th order SISO systems

This section briefly presents the generalization to higher order SISO systems. Consider the nth order SISO system with relative degree n (input-state linearizable) of the form

$$\begin{aligned} \dot{x}_1 &= x_2 \\ &\vdots \\ \dot{x}_{n-1} &= x_n \\ \dot{x}_n &= f(\mathbf{x}) + g(\mathbf{x})u \end{aligned} \tag{87}$$

where $x = x_1$, $\mathbf{x} = [x_1, \ldots, x_n]^T \in \Re^n$, and $u \in \Re$. We assume that f and g are smooth on the domain $\mathcal{D} \subset \Re^n$, and also assume that the sign of $g(\mathbf{x})$ is positive and $g(\mathbf{x})$ is lower bounded similar to (17). Note that a general class of input-state linearizable SISO systems

$$\dot{\mathbf{x}}_0 = \mathbf{f}_0(\mathbf{x}_0) + \mathbf{g}_0(\mathbf{x}_0)u \tag{88}$$

where $\mathbf{x}_0 \in \Re^n$, can be transformed into (87) with a nonlinear state transformation $\mathbf{x} = \mathbf{T}(\mathbf{x}_0)$. Our results can be generalized to a class of SISO systems having relative degree $r < n$ with a minimum-phase assumption (Choi & Farrell, 2000).

5.1. Control law

Consider a control law

$$u = \frac{1}{\hat{g}(\mathbf{x})} (-\hat{f}(\mathbf{x}) + x_d^{(n)} - \mathbf{K}\boldsymbol{e}) \tag{89}$$

where $x^{(n)}$ denotes the nth time derivative of x, $\mathbf{K} = [K_1, K_2, \ldots, K_n]$ is the feedback gain row vector, chosen such that the polynomial $s^n + \sum_{i=1}^{n} K_i s^{i-1} = 0$ has all roots in the left half of the complex number plane, and $\mathbf{e} = [e, \dot{e}, \ldots, e^{(n)}]^T$ is the tracking error vector with $e = x - x_d$. When a perfect approximation is assumed such as $\Delta_f = \Delta_g = 0$ and $\varepsilon_{f,k} = \varepsilon_{g,k} = 0$, the tracking error dynamics can be expressed in the controllable canonical form of the state space representation as:

$$\dot{\mathbf{e}} = \mathbf{A}\boldsymbol{e} + \mathbf{b}(f - \hat{f} + (g - \hat{g})u)$$
$$= \mathbf{A}\boldsymbol{e} + \mathbf{b}(-\boldsymbol{\phi}^T(\tilde{\boldsymbol{\theta}}_f + \tilde{\boldsymbol{\theta}}_g u)) \quad (90)$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & 1 \\ -K_1 & -K_2 & \cdots & -K_n \end{bmatrix} \text{ and } \mathbf{b} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \quad (91)$$

Note that $\mathbf{A}$ is Hurwitz. Define the sliding surface

$$e_1 = \mathbf{ce} \quad (92)$$

where $\mathbf{c} = [\Lambda_1, \ldots, \Lambda_n]$ is chosen such that ($\mathbf{A}$, $\mathbf{b}$, $\mathbf{c}$) is minimal (controllable and observable) and $H(s) = \mathbf{c}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{b}$ is strictly positive real. This filtered tracking error will be used in the tracking error-based parameter update (Choi & Farrell, 2000) and the strictly positive real assumption will be necessary in the Lyapunov stability analysis.

5.2. Composite parameter update law with projection

The composite parameter adaptation law with projection for the nth order SISO system becomes:

$$\dot{\hat{\boldsymbol{\Theta}}}_k = \text{Proj}\{\mathbf{P}_k \boldsymbol{\chi}_k (\bar{w}_k e_1 + w_k e_{pk})\} \quad (93)$$

$$\dot{\mathbf{P}}_k = \lambda \mathbf{P}_k - w_k \mathbf{P}_k \boldsymbol{\chi}_k \boldsymbol{\chi}_k^T \mathbf{P}_k \quad (94)$$

Note that the complete proof with imperfect approximation requires that (93) and (94) be modified by appropriate deadzones to prevent parameter adaptation when the prediction and tracking errors become sufficiently small.

5.3. Stability analysis

Consider the Lyapunov function

$$V = \frac{1}{2}\mathbf{e}^T \mathbf{S}\mathbf{e} + \frac{1}{2}\sum_{k=1}^{N} \tilde{\boldsymbol{\theta}}_k \mathbf{P}_k^T \tilde{\boldsymbol{\theta}}_k. \quad (95)$$

By the Lefschetz-Kalman-Yakubovich lemma (Tao & Ioannou, 1990), with the controllability and strictly positive real assumption of ($\mathbf{A}$, $\mathbf{b}$, $\mathbf{c}$), there exist real symmetric positive definite matrices $\mathbf{S}$ and $\mathbf{L}$, a real vector $\mathbf{q}$, and $\mu > 0$ such that

$$\mathbf{A}^T\mathbf{S} + \mathbf{S}\mathbf{A} = -\mathbf{q}\mathbf{q}^T - \mu\mathbf{L} \quad (96)$$

$$\mathbf{S}\mathbf{b} = \mathbf{c}^T \quad (97)$$

Similar to the derivation in Section 3.4, the time derivative of V can be calculated as

$$\dot{V} \le -\beta_1 \mathbf{e}^T\mathbf{e} - \frac{1}{2}\lambda \sum_{k=1}^{K} \tilde{\boldsymbol{\theta}}_k^T \mathbf{P}_k^{-1} \tilde{\boldsymbol{\theta}}_k - \frac{1}{2}\sum_{k=1}^{K} w_k e_{pk}^2 \quad (98)$$

where

$$\beta_1 = \frac{1}{2}\lambda_{\min}(\mathbf{q}\mathbf{q}^T + \mu\mathbf{L}) > 0.$$

This implies asymptotic convergence of the tracking error and the approximation error. With imperfect approximation, we need to treat the magnitude of u and the terms associated with the function approximation error by introducing a deadzone.

6. MIMO Systems

This section sketches a further generalization of the proposed learning adaptive controller to a class of MIMO systems of the form:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{G}(\mathbf{x})\mathbf{u} \quad (99)$$

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) \quad (100)$$

where $\mathbf{x} \in \Re^n$, $\mathbf{z} \in \Re^q$, $\mathbf{u} \in \Re^m$, $\mathbf{f}: \Re^n \to \Re^n$, $\mathbf{G}: \Re^n \to \Re^{n \times m}$, and $\mathbf{h}: \Re^n \to \Re^q$

Our ideas presented in this paper can be generalized if the system is square (having as many inputs as outputs such that $m = q$), minimum phase, can be transformed into a decoupled linear system via static state feedback (Sastry & Bodson, 1989; Sastry & Isidori, 1989), and the Lie derivative of the output function, $\mathbf{h}(\mathbf{x})$ is linearly parameterizable. Under similar conditions, our results also extend to $m \ge q$.

7. Empirical evaluations

We present two numerical simulations to demonstrate the effectiveness of the proposed learning controller. First, we apply the proposed control algorithm to a second order SISO system, and evaluate the tracking performance and on-line function approximation with structure adaptation. Then, we illustrate an application to a planar 2-DOF robot arm as an example of a nonlinear MIMO system.

7.1. SISO example

First, we consider a second order SISO system which was used in (Choi & Farrell, 2000; Sanner & Slotine, 1992):

$$\dot{x}_1 = x_2, \quad \dot{x}_2 = f(\mathbf{x}) + u \quad (101)$$

where $\mathbf{x}=[x_1, x_2]^T$ and

$$f(\mathbf{x}) = 4\left(\frac{\sin(4\pi x_1)}{\pi x_1}\right)\left(\frac{\sin(\pi x_2)}{\pi x_2}\right)^2. \tag{102}$$

This is an interesting plant since the nonlinear function $f(\mathbf{x})$ has a sharp peak at the origin while the remaining function is relatively flat as plotted in Fig. 7 (left). On-line learning of a function with such nonlinear complexity is a difficult problem without prior knowledge of the nonlinearity. The plant dynamics are integrated using the Runge-Kutta algorithm with a time step of 0.001s, and the parameters of local models are updated every 0.02s without deadzone. We use the same PD gain $\mathbf{K}=[K_1, K_2]^T=[100, 20]^T$ and filtered error $e_1=\mathbf{ce}$ where $\mathbf{c}=[15, 1]^T$ as in (Choi & Farrell, 2000). Initial conditions of the dynamical system are set to zero.

7.1.1. Tracking performance

This section presents the tracking performance of the proposed learning adaptive controller in comparison to a (nonadaptive) PD controller and the tracking error-based adaptive controller of Section 3.2. We use the same desired trajectory $x_d(t)$ as in (Choi & Farrell, 2000; Sanner & Slotine, 1992), which is generated from an output of a third order prefilter with a bandwidth of 10 rad/s

$$\dot{\mathbf{x}}_r = \mathbf{A}_r\mathbf{x}_r + \mathbf{b}_r u_r \tag{103}$$

$$x_d = \mathbf{C}_r\mathbf{x}_r \tag{104}$$

where

$$\mathbf{A}_r = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\alpha^3 & -3\alpha^2 & -3\alpha \end{bmatrix}, \quad \mathbf{b}_r = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{105}$$

$$\mathbf{c}_r = [1,0,0], \quad \alpha = 10$$

and

$$u_r(t) = 0.5\mathrm{sgn}(\sin(2\pi f t)) + u_{\mathrm{mean}} \tag{106}$$

where $f=0.4$ and $u_{\mathrm{mean}}=0.5$. Fig. 4 depicts the desired trajectories for two periods.

For the tracking error based adaptive controller, the location of the centers of the kernels are predetermined on a grid over the region of $[-0.5, 1.5]\times[-3, 3]$ with mesh size 0.5. Thus, 65 local models are used. For the RFWR composite learning adaptive controller, a new local model is added on-line as necessary if a training point does not activate any local model by more than a threshold $w_{gen}=0.2$. $\mathbf{P}_k$ is initialized with $\mathbf{P}_k=250\mathbf{I}$. As a result, for the specified desired trajectory 18 local models are created. In both cases, the initial distance metric for the kernel is chosen to be $\mathbf{M}=2.3\mathbf{I}$ and, all the initial parameters of the locally linear models $\hat{\boldsymbol{\theta}}_k$ are set to zero.

Fig. 5 compares the tracking error of the (nonadaptive) PD controller, the tracking error-based adaptive controller with two different adaptation rates, $\boldsymbol{\Gamma}_k=10\mathbf{I}$ and $250\mathbf{I}$, and the proposed RFWR composite learning adaptive controller. The convergence rate of the tracking error based adaptive controller largely depends on the adaptation rate $\boldsymbol{\Gamma}_k$: a larger $\boldsymbol{\Gamma}_k$ yields faster convergence. Ideally, well-tuned tracking error based adaptive controllers could perform comparable to the proposed RFWR composite adaptive controller in terms of convergence rate and tracking error bound as can be seen in Fig. 5 ($\boldsymbol{\Gamma}_k=250\mathbf{I}$ case). However, there is a practical limitation on the magnitude of $\boldsymbol{\Gamma}_k$ due to finite sampling times and danger of exciting unmodelled dynamics in the presence of measurement noise, which may lead to instability of the control system. For example, when a Gaussian noise of $N(0, 0.01)$ is added to the measurement, the performance of the tracking error based adaptive controller with $\boldsymbol{\Gamma}_k=250\mathbf{I}$ is much degraded while

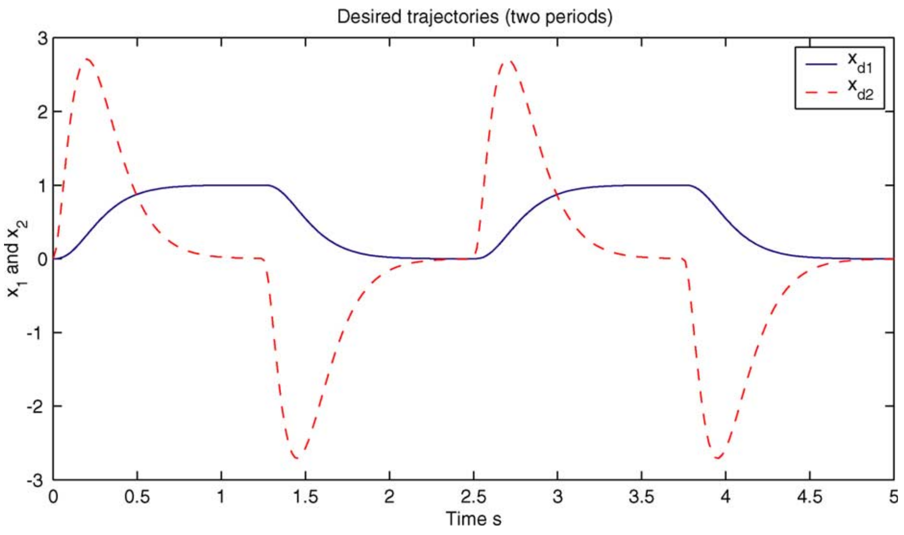

Fig. 4. The desired trajectories, $\mathbf{x}_d$ over two periods.

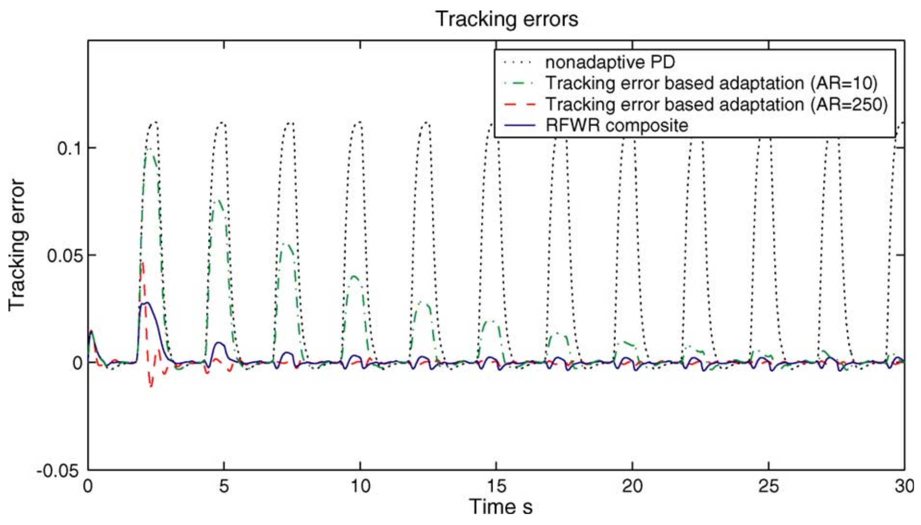

Fig. 5. Comparison of the tracking errors among (nonadaptive) PD controller (dotted), tracking error based adaptive controller with $\mathbf{\Gamma}_k = 10\mathbf{I}$ (dash-dot), $\mathbf{\Gamma}_k = 250\mathbf{I}$ (dashed), and the RFWR composite adaptive controller (solid).

the RFWR composite adaptive controller achieves stable and rapid learning (see Fig. 6).

7.1.2. On-line function approximation with structure adaptation

We now turn to RFWR composite learning control with structure adaptation of the function approximator. This property is particularly useful when the input domain and complexity of the function to be approximated are not known. As mentioned before, a new receptive field is added as necessary and the distance metric of the receptive field is optimized by minimizing a leave-one-out cross validation error criterion.

In the following example, we use the same square wave to drive the desired trajectory generator, but its mean, u_{mean} in (106), is varied randomly between -1.0 and 1.0 every 2.5 s to collect training data distributed over the region which roughly covers $[-2, 2]\times[-2, 2]$. The parameters $w_{\text{gen}} = 0.2$, $\mathbf{P}_k = 250\mathbf{I}$, are used for the RFWR update. For distance metric optimization, second order gradient descent (Schaal & Atkeson, 1998) is incorporated to accelerate the distance metric adaptation speed. Penalty $\gamma = 10^{-7}$ is used in (83). The first local model is initialized with the distance metric $\mathbf{M} = 2.3\mathbf{I}$, and when a new local model is added, its initial distance metric is set to be the same value as that of the closest existing local model.

Fig. 7 shows the target function to be approximated (left) and its approximation after 400 s of training (center). As a result of distance metric adaptation, the number of local models grew to 99, and initially large receptive fields were adjusted during learning according to the local curvature of the function, i.e. they became narrow in the region of

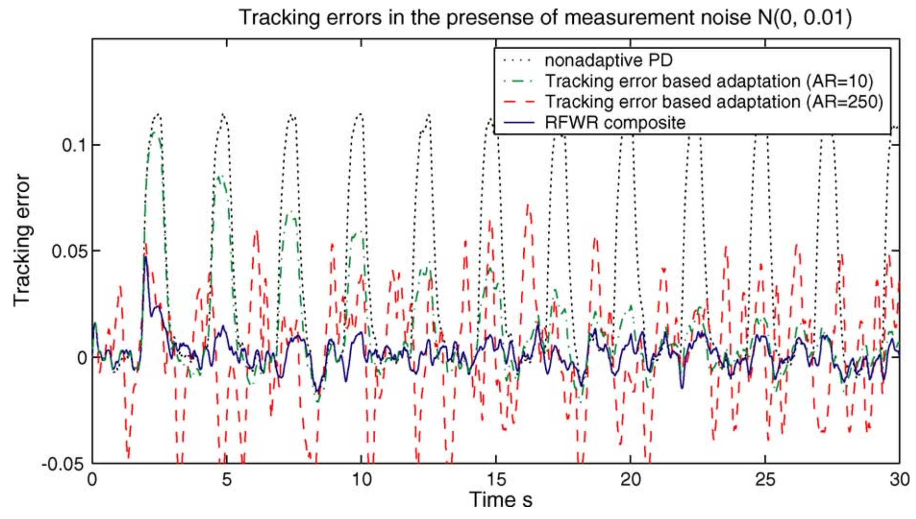

Fig. 6. Comparison of the tracking errors among different controllers in the presence of measurement noise: (nonadaptive) PD controller (dotted), tracking error based adaptive controller with $\mathbf{\Gamma}_k = 10\mathbf{I}$ (dash-dot), $\mathbf{\Gamma}_k = 250\mathbf{I}$ (dashed), and the RFWR composite adaptive controller (solid). Note that the performance of the tracking error based adaptive controller with $\mathbf{\Gamma}_k = 250\mathbf{I}$ is much degraded.

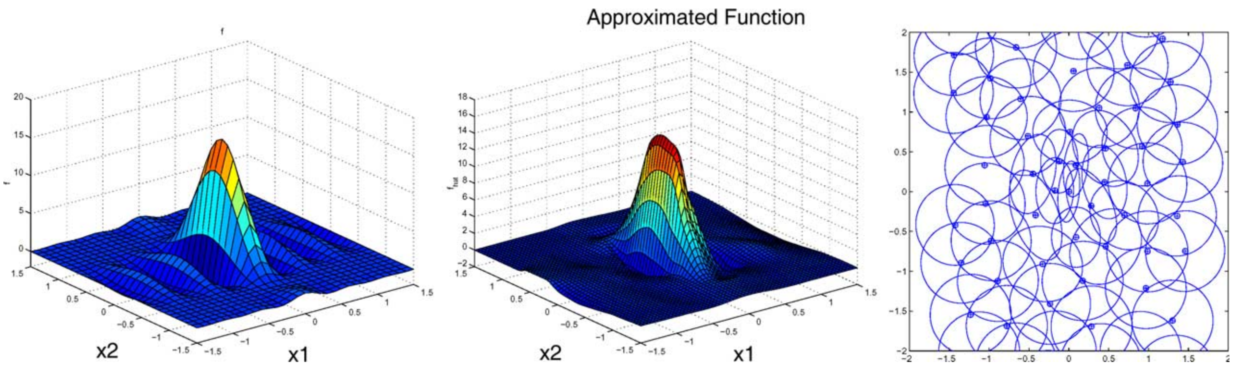

Fig. 7. Left: the true function $f(\mathbf{x})$. Center: the approximated function, $\hat{f}(\mathbf{x})$, after 400 s of learning. Right: Organization of the receptive fields around the origin after 400 s of training given by contour lines of 0.1 isoactivation and a ⊕ mark denotes the centers. Size and shape of the receptive fields are adjusted according to the local curvature of the function.

the bump at the origin and remained large where the function was flat (see Fig. 7, right).

Note that for the tracking error based adaptive controller, the structure of the function approximator (center allocation, and size and shape of receptive fields of local models) needs to be predetermined prior to on-line adaptation. For this simulation, the tracking error based adaptive controller would require 207 local models to cover the estimated region of operation $[-2, 2]\times[-3, 8]$ distributed on a grid with mesh size 0.5. In contrast, the proposed RFWR composite adaptation does not require prior information of the size of the domain of operation.

7.2. MIMO example: 2-DOF robot arm

As a second example, we present numerical simulations of the proposed algorithm on a planar 2-DOF arm as an application to a nonlinear MIMO system. The plant dynamics take the form of a standard two-link planar manipulator:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{V}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{k}(\mathbf{q}) + \mathbf{D}\dot{\mathbf{q}} = \boldsymbol{\tau} \quad (107)$$

where $\mathbf{q}=[\theta_1, \theta_2]^T$, $\boldsymbol{\tau}=[\tau_1, \tau_2]^T$ is the torque input, $\mathbf{M}(\mathbf{q})$ is the inertia matrix, $\mathbf{V}(\mathbf{q}, \dot{\mathbf{q}})$ is the Coriolis/centrifugal vector, $\mathbf{k}(\mathbf{q})$ is the gravity vector, and $\mathbf{D}$ denotes the viscous friction coefficient matrix.

$$\mathbf{M} = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}$$

$$\begin{aligned} m_{11} &= m_1 l_{c1}^2 + I_1 + m_2(l_1^2 + 2l_1 l_{c2}\cos\theta_2 + l_{c2}^2) + I_2, \\ m_{12} &= m_{21} = m_2(l_{c2}^2 + l_1 l_{c2}\cos\theta_2) + I_2, \\ m_{22} &= m_2 l_{c2}^2 + I_2 \end{aligned}$$

$$\mathbf{V}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = -m_2 l_1 l_{c2}\sin\theta_2 \begin{bmatrix} 2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2 \\ -\dot{\theta}_1^2 \end{bmatrix}$$

$$\mathbf{k}(\mathbf{q}) = \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} m_1 g l_{c1}\sin\theta_1 + m_2 g(l_1\sin\theta_1 + l_{c2}\sin(\theta_1+\theta_2)) \\ m_2 g l_{c2}\sin(\theta_1+\theta_2) \end{bmatrix}$$

$$\mathbf{D} = \mathrm{diag}\{d_i\}$$

m_i and I_i are the mass and the moment of inertia of each link, respectively, and l_i is the link length. The center of mass of each link is located on the center line, which passes through adjacent joints at a distance l_{ci}. d_i is the coulomb friction coefficient. The link parameters used in the simulation are $m_1=m_2=1$, $l_1=l_2=1$, $l_{c1}=l_{c2}=0.5$, $I_1=I_2=0.02$ and $d_1=d_2=0.01$.

The task is to draw a periodic Figure 8 pattern in Cartesian space at a 1 Hz base frequency under the influence of gravity as depicted in Fig. 8. Both links of the arm are 1 m long as specified above, and the Figure 8 had a height of 0.8 m and width of 0.5 m. Due to these large dimensions and the double frequency in the horizontal direction that is needed to draw the figure-8, the nonlinearities of the dynamics of the 2-DOF arm are significant. Desired trajectories in joint space were computed from standard

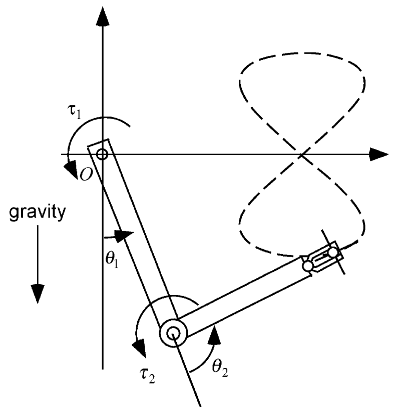

Fig. 8. 2-DOF planar arm and Figure 8 task.

analytical inverse kinematics formulae. The plant dynamics is simulated on Matlab Simulink using an adaptive step size Runge-Kutta algorithm, and the learning adaptive controller is incorporated using a compiled S-function based on C code. The parameters of the local models are updated every 0.00ls.

The proposed learning adaptive control is implemented as follows: First, we rewrite the dynamics of the robot arm (107) in the forward model representation

$$\ddot{\mathbf{q}} = \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q})\boldsymbol{\tau} \tag{108}$$

where

$$\mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}) = -\mathbf{M}^{-1}(\mathbf{V} + \mathbf{k} + \mathbf{D}\dot{\mathbf{q}}), \text{ and } \mathbf{G}(\mathbf{q}) = \mathbf{M}^{-1} \tag{109}$$

and we approximate **f** and **G** using locally linear models for f_i and g_{ij} where

$$\mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}, \quad \text{and } \mathbf{G} = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix}. \tag{110}$$

Essentially, this formulation requires two independent function approximators for

$$\ddot{\theta}_1 = f_1(\mathbf{q}, \dot{\mathbf{q}}) + g_{11}(\mathbf{q})\tau_1 + g_{12}(\mathbf{q})\tau_2 \tag{111}$$

$$\ddot{\theta}_2 = f_2(\mathbf{q}, \dot{\mathbf{q}}) + g_{21}(\mathbf{q})\tau_1 + g_{22}(\mathbf{q})\tau_2 \tag{112}$$

with input vector $\boldsymbol{\chi}_k$ (cf. (21)) for each local model as

$$\boldsymbol{\chi}_k = [\bar{\mathbf{x}}_k^T, \bar{\mathbf{x}}_k^T\tau_1, \bar{\mathbf{x}}_k^T\tau_2]^T, \quad \text{and } \bar{\mathbf{x}}_k = \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix} - \mathbf{c}_k. \tag{113}$$

Then, we design a feedback linearizing controller with the estimates **f** and **G**

$$\boldsymbol{\tau} = \hat{\mathbf{G}}^{-1}(-\hat{\mathbf{f}} + \ddot{\mathbf{q}}_d - \mathbf{K}_d\dot{\mathbf{e}} - \mathbf{K}_p\mathbf{e}) \tag{114}$$

where

$$\hat{\mathbf{f}} = \begin{bmatrix} \hat{f}_1 \\ \hat{f}_2 \end{bmatrix}, \quad \text{and } \hat{\mathbf{G}} = \begin{bmatrix} \hat{g}_{11} & \hat{g}_{12} \\ \hat{g}_{21} & \hat{g}_{22} \end{bmatrix}, \tag{115}$$

$\mathbf{e} = \mathbf{q} - \mathbf{q}_d$, and $\mathbf{K}_p$ and $\mathbf{K}_d$ are PD gain matrices. Note that, in practice, to ensure the numerical stability of the matrix inversion associated with the feedback linearization (in this case $\hat{\mathbf{G}}^{-1}$), we employ ridge regression (Schaal & Atkeson, 1998) for matrix inversion. In this paper, we are concerned with the forward model representation of the plant dynamics for general control applications. However, in our future work, for the particular application of rigid body dynamics including robot arms, we will address an inverse model formulation, which does not require matrix inversion and would be made suitable for such plant dynamics.

Fig. 9 illustrates the performance of the proposed learning adaptive controller in comparison to a low gain PD controller. In Fig. 9, X_{des} is the desired end-effector trajectory, and X_{PD} is the result of PD control. $X_{\text{composite}}$ denotes tracking with the proposed controller after 60 s of learning on a slowly drifting Figure 8 pattern around the desired trajectory. For our learning adaptive controller, learning started from scratch with no local models, and new local models were added as necessary if a training point did not activate any local model by more than a threshold. The distance metric of the local model was also learned on-line. While the low gain PD controller has very large tracking errors that lead to a strongly distorted Figure 8 pattern, the proposed controller achieved almost perfect tracking results after 60 seconds of learning. As a measure of the tracking performance, the L_2 norm of the tracking errors in joint space is 4.80×10^{-1} rad for the low gain PD controller and 1.37×10^{-3} rad for the proposed controller, which is defined by $L_2[\mathbf{e}(t)] = \sqrt{(1/T)\int_{t_0}^{t_f} \|\mathbf{e}(t)\|^2 \mathrm{d}t}$ where $T = t_f - t_0$. For the proposed control algorithm, 166 local models are created. These simulation results demonstrate rapid learning of the system dynamics and convergence of the tracking error of the proposed learning adaptive controller.

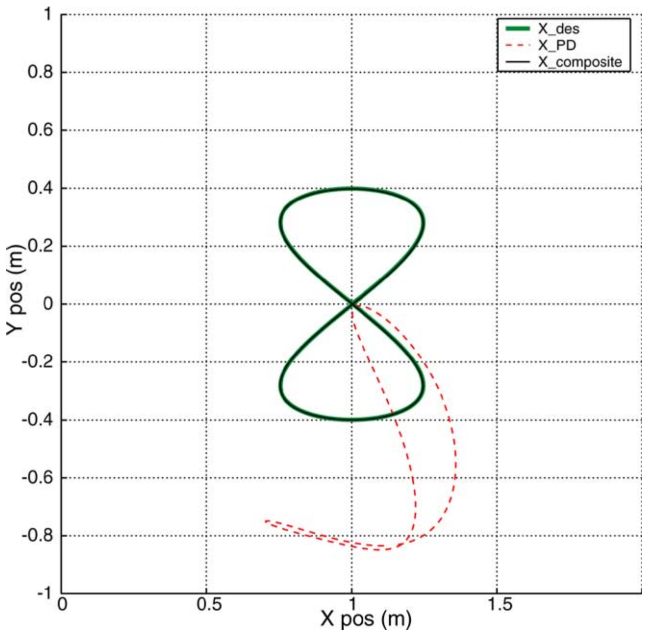

Fig. 9. Trajectories of robot end-effector. X_{des} is the desired trajectory (thick line), X_{PD} is the result using a low gain PD controller (dashed line), $X_{\text{composite}}$ is the result using the proposed composite adaptive controller after 60 s of learning (thin solid line). While the low gain PD controller yields very large tracking error, the proposed controller achieves good tracking performance-$X_{\text{composite}}$ coincides almost perfectly with the desired trajectory.

8. Conclusion

In this paper, we presented a comprehensive development of a provably stable learning adaptive controller that covers a large class of nonlinear systems. Motivated from the viewpoint of statistical learning, we employed a locally weighted learning framework in which unknown functions of the control system are approximated by piecewise linear models. New local models are allocated automatically as necessary and the shape and size of the local models, characterized by a distance metric, are optimized on-line. Such automatic structure adaptation of the function approximator is particularly desirable given that the domain of operation and complexity of the function to be approximated are usually not known in advance. Our learning adaptive control algorithm uses both the tracking error and the prediction error to update the learning parameters, inspired by composite adaptive control in which tracking error-based adaptation and recursive least squares updates are combined. Stability analyses and numerical simulations were provided to illustrate the effectiveness of the proposed controller.

Future work will address developments of theoretically sound learning and control algorithms toward real-time high-dimensional system control including humanoid robots. Our current locally weighted learning algorithm for function approximation with piecewise linear models (RFWR) will become computationally very expensive for learning in high-dimensional input spaces. As a replacement, we consider using an advanced statistical learning algorithm, locally weighted projection regression (LWPR), proposed in (Vijayakumar & Schaal, 2000) which achieves low computational complexity and efficient learning in high-dimensional spaces. In this paper, from a control theoretic point of view, we considered a general forward dynamics representation. However, for the particular application to rigid body dynamics, we are more interested in an inverse model representation. We will adapt our framework to this special case of nonlinear systems and also compare it with biologically inspired internal model learning such as feedback error learning (Gomi & Kawato, 1993).

Acknowledgements

We would like to thank Professor Chris Atkeson at Carnegie Mellon University and Professor Jean-Jacques E. Slotine for their helpful comments on this work. This research was supported in part by National Science Foundation grants ECS-0325383, IIS-0312802, IIS-0082995, ECS-0326095, ANI-0224419, a NASA grant AC#98-516, an AFOSR grant on Intelligent Control, the Communications Research Laboratory of Japan, the ERATO Kawato Dynamic Brain Project funded by the Japan Science and Technology Agency, and the ATR Computational Neuroscience Laboratories.

Appendix A. Constrained optimization using lagrange multiplier method

Consider the following constrained optimization problem of the form:

$$\text{minimize } g(\mathbf{x}) = \mathbf{x}^T\mathbf{b}, \qquad \text{subject to the constraint } \mathbf{x}^T\mathbf{D}x - \mu = 0 \tag{A1}$$

where $\mathbf{x}\in\Re^n$, $\mathbf{b}\in\Re^n$ is the parameter vector of the linear model, $\mathbf{D}$ is positive definite distance metric, and μ is a constant which is representative of the local model boundary. Define an objective function

$$J = \mathbf{x}^T\mathbf{b} + \lambda(\mathbf{x}^T\mathbf{D}\mathbf{x} - \mu) \tag{A2}$$

where λ is the Lagrange multiplier. Then, at the stationary points of J, we have

$$\frac{\partial J}{\partial \mathbf{x}} = \mathbf{b}^T + 2\lambda\mathbf{x}^T\mathbf{D} = 0 \tag{A3}$$

and solving this equation for $\mathbf{x}$ yields

$$\mathbf{x} = -\frac{1}{2\lambda}\mathbf{D}^{-1}\mathbf{b}. \tag{A4}$$

Substituting (A4) into (A1), we have

$$\lambda = \pm\frac{1}{2\sqrt{\mu}}\sqrt{\mathbf{b}^T\mathbf{D}^{-1}\mathbf{b}}. \tag{A5}$$

Thus, we obtain

$$\mathbf{x} = \pm\frac{\sqrt{\mu}}{\sqrt{\mathbf{b}^T\mathbf{D}^{-1}\mathbf{b}}}\mathbf{D}^{-1}\mathbf{b}. \tag{A6}$$

The value of $f(\mathbf{x})$ at (A6) is given by

$$\mathbf{x}^T\mathbf{b} = \pm\sqrt{\mu\mathbf{b}^T\mathbf{D}^{-1}\mathbf{b}}. \tag{A7}$$

Thus, the minimum of $\mathbf{x}^T\mathbf{b}$ subject to the constraint (A1) is $-\sqrt{\mu\mathbf{b}^T\mathbf{D}^{-1}\mathbf{b}}$ at $\mathbf{x} = -\frac{\sqrt{\mu}}{\sqrt{\mathbf{b}^T\mathbf{D}^{-1}\mathbf{b}}}\mathbf{D}^{-1}\mathbf{b}$.

Appendix B. Lefschetz-Kalman-Yakubovich lemma and positive real transfer function

Lemma 1. *Lefschetz-Kalman-Yakubovich lemma* (*Tao & Ioannou, 1990*)

Given $\mu > 0$, *a matrix* $\mathbf{A}$ *such that* $det(s\mathbf{I} - \mathbf{A})$ *has only zeros in the open left half plane, a real vector* $\mathbf{b}$ *such that* $(\mathbf{A}, \mathbf{b})$ *is completely controllable, a real vector* $\mathbf{c}$, *a scalar* d, *and an arbitrary real symmetric positive definite matrix* $\mathbf{L}$; *then a real vector* $\mathbf{q}$ *and a real matrix* $\mathbf{P} = \mathbf{P}^{\mathrm{T}} > 0$ *satisfying*

$$\mathbf{A}^T\mathbf{P} + \mathbf{P}A = -\mathbf{q}\mathbf{q}^T - \mu\mathbf{L} \tag{B1}$$

$$\mathbf{P}\mathbf{b} - \mathbf{c}^T = \sqrt{(2d)}\mathbf{q} \tag{B2}$$

exist if and only if $h(s) = \mathbf{c}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{b} + d$ *is a strictly positive real matrix and* μ *is sufficiently small.*

Definition 1. Positive real transfer function (Krstić et al., 1995, p. 509))

A rational transfer function *G(s)* is said to be positive real if *G(s)* is real for all real *s*, and *Re*{*G(s)*}≥*0* for all

$Re\{s\} \geq 0$. If, in addition, $G(s-\varepsilon)$ is positive real for some $\varepsilon > 0$, then $G(s)$ is said to be strictly positive real.

Note that the scalar positive real transfer function $G(s)$ is stable, minimum-phase and of relative degree not exceeding one (Kaufman, Bar-Kana, & Sobel, 1993). In addition, any scalar transfer function of a relative degree higher than one is not positive real (Khalil, 1996).

Consider the following linear time-invariant system

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}, \quad \mathbf{y} = \mathbf{Cx} + \mathbf{Du} \tag{B3}$$

where $\mathbf{x} \in \Re^n$, $\mathbf{u} \in \Re^m$ and $\mathbf{y} \in \Re^m$. $\mathbf{H}(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}$ is the transfer matrix of the system above.

The following theorem states the necessary and sufficient condition of strictly positive real system matrices for a special case of the dynamical system given by (B3) (Tao & Ioannou, 1990).

Theorem 1. (*Tao & Ioannou*, 1990) *In* (B3), *let* $n=1$ *and* $m=1$ *or* $n=2$ *and* $m=1$ *and let* $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ *be minimal*, $\mathbf{D}=0$ *and* $\mathbf{B} \neq 0$, *then* $\mathbf{H}(s)$ *is a strictly positive and real matrix if and only if the following conditions hold*

1. All eigenvalues of $\mathbf{A}$ have negative real parts
2. $\mathbf{CB} = (\mathbf{CB})^T > 0$
3. $\mathbf{CAB} + (\mathbf{CAB})^T < 0$

References

Atkeson, C. G., Moore, A. W., & Schaal, S. (1997). Locally weighted learning. *Artificial Intelligence Review*, *11*(1–5), 11–73.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford: Oxford University Press.

Chen, F.-C., & Khalil, H. K. (1995). Adaptive control of a class of nonlinear discrete-time systems using neural networks. *IEEE Transactions on Automatic Control*, *40*(5), 791–801.

Choi, J. Y., & Farrell, J. A. (2000). Nonlinear adaptive control using networks of piecewise linear approximators. *IEEE Transactions on Neural Networks*, *11*(2), 390–401.

Gomi, H., & Kawato, M. (1993). Neural network control for a closed-loop system using feedback-error-learning. *Neural Networks*, *6*, 933–946.

Kaufman, H., Bar-Kana, I., & Sobel, K. (1993). *Direct adaptive control algorithm*. Berlin: Springer.

Khalil, H. K. (1996). *Nonlinear systems* (2nd ed). Englewood Cliffs, NJ: Prentice Hall.

Krstić, M., Kanellakopoulos, I., & Kokotović, P. (1995). *Nonlinear and adaptive control design*. New York: Wiley.

Levin, A. U., & Narendra, K. S. (1993). Control of nonlinear dynamical systems using neural networks: Controllability and stabilization. *IEEE Transactions on Neural Networks*, *4*(2), 192–206.

Ljung, L., & Söderström, T. (1986). *Theory and practice of recursive identification*. Cambridge, MA: MIT Press.

Narendra, K. S., & Annaswamy, A. M. (1989). *Stable adaptive systems*. Englewood Cliffs, NJ: Prentice Hall.

Sanner, R., & Slotine, J.-J. E. (1992). Gaussian networks for direct adaptive control. *IEEE Transactions on Neural Networks*, *3*(6), 837–863.

Sastry, S., & Bodson, M. (1989). *Adaptive control: Stability convergence, and robustness*. Englewood Cliffs, NJ: Prentice Hall.

Sastry, S., & Isidori, A. (1989). Adaptive control of linearizable systems. *IEEE Transactions on Automatic Control*, *34*(11), 1123–1131.

Schaal, S., & Atkeson, C.G. (1997). Receptive field weighted regression (Technical Report RE-H-209). ATR Human Information Processing Laboratories.

Schaal, S., & Atkeson, C. G. (1998). Constructive incremental learning from only local information. *Neural Computation*, *10*(8), 2047–2084.

Seshagiri, S., & Khalil, H. K. (2000). Output feedback control of nonlinear systems using RBF neural networks. *IEEE Transactions on Neural Networks*, *11*(1), 69–79.

Slotine, J.-J. E., & Li, W. (1987). On the adaptive control of robot manipulators. *International Journal of Robotics Research*, *6*(3), 49–59.

Slotine, J.-J. E., & Li, W. (1989). Composite adaptive control of robot manipulators. *Automatica*, *25*(4), 509–519.

Slotine, J.-J. E., & Li, W. (1991). *Applied nonlinear control*. Englewood Cliffs, NJ: Prentice Hall.

Tao, G., & Ioannou, P. A. (1990). Necessary and sufficient conditions for strictly positive real matrices. *IEE Proceedings G, Circuits, Devices and Systems*, *137*(5), 360–366.

Vijayakumar, S., & Ogawa, H. (1999). RKHS based functional analysis for exact incremental learning. *Neurocomputing*, *29*(1–3), 85–113.

Vijayakumar, S., & Schaal, S. (2000). *Locally weighted projection regression: An O(n) algorithm for incremental real time learning in high dimensional space In International conference in machine learning (ICML), Stanford, USA*.

Whitcomb, L. L., Rizzi, A. A., & Koditschek, D. E. (1993). Comparative experiments with a new adaptive controller for robot arms. *IEEE Transactions on Robotics and Automation*, *9*(1), 59–70.