Lazy Learning for Local Modeling and Control Design*

Gianluca Bontempi and Mauro Birattari and Hugues Bersini

Iridia - CP 194/6

Université Libre de Bruxelles

email: {gbonte, mbiro, bersini}@ulb.ac.be

Abstract. This paper presents local methods for modeling and control of discrete-time unknown nonlinear dynamical systems, when only a limited amount of input-output data is available. We propose the adoption of *lazy learning*, a memory-based technique for local modeling. The modeling procedure uses a query-based approach to select the best model configuration by assessing and comparing different alternatives. A new recursive technique for local model identification and validation is presented, together with an enhanced statistical method for model selection. Also, three methods to design controllers based on the local linearization provided by the *lazy learning* algorithm are described. In the first method the *lazy* technique returns the forward and inverse models of the system which are used to compute the control action to take. The second is an indirect method inspired to self-tuning regulators where recursive least squares estimation is replaced by a local approximator. The third method combines the linearization provided by the local learning techniques with optimal linear control theory, to control nonlinear systems about regimes which are far from the equilibrium points. Simulation examples of identification and control of nonlinear systems starting from observed data are given.

* The work of Gianluca Bontempi was supported by the European Union TMR Grant FMBICT960692. The work of Mauro Birattari was supported by the F.I.R.S.T. program of the Région Wallonne, Belgium.

1 Introduction

The problem of modeling a process from a limited amount of observed data has been the object of several disciplines from nonlinear regression to machine learning and system identification. In the literature dealing with this problem, two main opposing paradigms have emerged: local memory-based versus global methods.

Global modeling builds a single functional model of the dataset. This has traditionally been the approach taken in neural network modeling and other form of nonlinear statistical regression. The available dataset is used by a learning algorithm to produce a model of the mapping and then the dataset is discarded and only the model is kept.

Local memory-based algorithms defer processing of the dataset until they receive request for information (e.g. prediction or local modeling). The classical nearest neighbor method is the original approach to local modeling. A database of observed input-output data is always kept and the estimate for a new operating point is derived from an interpolation based on a neighborhood of the query point. Local techniques are an old idea in time series prediction (Farmer & Sidorowich, 1987), classification (Cover & Hart, 1967) and regression (Cleveland, 1979). The idea of local approximators as alternative to global models originated in non-parametric statistics (Epanechnikov, 1969), (Benedetti, 1977) to be later rediscovered and developed in the machine learning field (Aha, 1989), (Bottou & Vapnik, 1992). Recent work on *lazy learning* (a.k.a just-in-time learning) gave a new impetus to the adoption of local techniques for modeling (Atkeson *et al.*, 1997a), (Stenman *et al.*, 1996) and control problem (Tolle *et al.*, 1992), (Schaal & Atkeson, 1994), (Atkeson *et al.*, 1997b). The new promising feature of this local paradigm is the adoption of enhanced statistical procedures to identify the local approximator. One example is the PRESS statistic (Myers, 1990) which is a simple, well-founded and economical way to perform *leave-one-out* cross validation (Efron & Tibshirani, 1993) and to assess the performance in generalization of local linear models.

The aim of this paper is to extend the idea of local memory-based learning in several directions. First, we propose a model identification methodology based on the use of an iterative optimization procedure to select the best local model among a set of different candidates. In classical local modeling, an amount of options had to be designed by the data analyst according to heuristic criteria and a priori assumptions. Here we propose an automatic selection procedure which searches for the optimal model configuration, by returning for each candidate model its parameters and a statistical description of its generalization properties.

We introduce a new algorithm to estimate in a recursive way the model performance in cross-validation. In particular, we propose a technique based on recursive least squares methods to compute the PRESS in an incremental way. Moreover, a powerful and well-founded statistical test is used to compare the performance of two alternative candidates on the basis of their cross-validation error sampling distributions.

The contribution of the paper in the control domain is a set of nonlinear control design techniques which extensively use analysis and design tools from linear control. The idea of employing linear techniques in a nonlinear setting is not new in control literature but had recently gained a new popularity thanks to methods for combining multiple estimators and controllers in different operating regimes of the system (Murray-Smith & Johansen, 1997). Gain scheduling (Shamma & Athans, 1992), fuzzy inference systems (Takagy & Sugeno, 1985) and local model networks (Johansen & Foss, 1993), are well-known examples of control techniques for nonlinear systems based on linear control. A comparative review of these approaches is provided in section 3.1. Here, we propose three methods for discrete-time control based on the local linear description returned by a *lazy learning* approximator.

The first method is an example of gradient-based controller which exploits the nice properties of the *lazy learning* algorithm as a nonlinear approximator. It is inspired to neural controllers and combines an inverse with a forward model to select the control action which, according to the available dataset, is supposed to produce the desired output of the controlled system. The

control algorithm has a one time-step horizon and is implemented as a gradient based optimization algorithm where the *lazy* model computes the value and the gradient of the cost function to be minimized. The algorithm has been introduced by (Atkeson *et al.*, 1997b) but was tested simply on a static control task. Here we analyze its dynamic stability properties and we compare them with the other proposed local control techniques.

The second controller is derived from the self-tuning regulator (STR) architecture (Astrom, 1983) and combines discrete-time conventional control (e.g. generalized minimum variance, pole placement) with local model identification. The control system can be thought of as composed of two loops. The inner one consists of the process and a feedback regulator. The parameters of the regulator are adjusted by the outer loop, that is in this case an adaptive *lazy learning* estimator. The main differences with conventional adaptive control techniques lie in the parameter estimation scheme. In the STR, identification is performed by a recursive parameter estimator which updates the same linear model when a new input-output sample is observed. In our approach there is no global linear model description but at each time-step the system dynamics is linearized in the neighborhood of the current operating regime. The adaptive feature of the *lazy* model is due not to a sequential parameter estimation but simply to the database updating.

The third control system we propose is designed with optimal control techniques parameterized with the values returned by the linear local estimator. The combination of linear quadratic regulators (LQR) with local modeling is not new in literature (Tanaka, 1995), (Atkeson *et al.*, 1997b), (Passino & Yurkovich, 1996). In these papers, however, the authors make two strong assumptions: an analytical description of the locus of equilibrium points is available, and the system is supposed to evolve in a sufficiently restricted neighborhood of the desired regime. Here, the idea is that a combination of a local estimator with a time-varying optimal control can take into account the nonlinearity of a system over a wider range than conventional linearized quadratic regulators. We propose a receding horizon controller which returns at each sampling period the first action of the optimal sequence found by a gradient based optimization procedure. The op-

timization problem is formulated as a minimization of a quadratic cost function where the cost is returned by a forward simulation of the identified model and the gradient is returned by the discrete-time Riccati equation for linear time-varying systems.

It is worth saying that this paper will not focus on algorithmic computational issues. On this subject, the reader is invited to refer to the literature dealing with efficient implementations of memory based algorithms (Nene & Nayar, 1997), (Moore *et al.*, 1997).

The remainder of the paper is organized as follows. In section 2 we will introduce our modeling technique based on an iterative selection procedure. In section 2.1 we present the recursive algorithm for PRESS statistic computation. In section 2.2 the method for statistical comparison between models assessed by the recursive PRESS evaluation is introduced. Algorithmic details and theoretical analysis of the three algorithms for local control can be found in section 3. Finally, in section 4 simulation examples of identification and of control are given.

2 Local modeling as an optimization problem

Modeling from data involves integrating human insight with learning techniques. In many real cases the analyst faces a situation where a limited set of data is available and an accurate prediction is required. Often, information about the order, the structure or the set of relevant variables is missing or not reliable. The process of learning consists in a trial and error procedure during which the model is properly tuned on the available data. In the *lazy learning* approach, the estimation of the value of the unknown function is performed giving the whole attention to the region surrounding the point where the estimation itself is required.

Let us consider an unknown mapping $f\colon \Re^m \to \Re$ of which we are given a set of N samples $\big\{(\boldsymbol{\varphi}_1, y_1), (\boldsymbol{\varphi}_2, y_2), \ldots, (\boldsymbol{\varphi}_N, y_N)\big\}$. These examples can be collected in a matrix $\boldsymbol{\Phi}$ of dimensionality $[N \times m]$, and in a vector $\mathbf{y}$ of dimensionality $[N \times 1]$.

Given a specific query point $\boldsymbol{\varphi}_q$, the prediction of the value $y_q = f(\boldsymbol{\varphi}_q)$ is computed as follows. First, for each sample $(\boldsymbol{\varphi}_i, y_i)$ a weight w_i is computed as a function of the distance $d(\boldsymbol{\varphi}_i, \boldsymbol{\varphi}_q)$ from

the query point $\boldsymbol{\varphi}_q$ to the point $\boldsymbol{\varphi}_i$. Each row of $\boldsymbol{\Phi}$ and $\mathbf{y}$ is then multiplied by the corresponding weight creating the variables $\mathbf{Z} = \mathbf{W}\boldsymbol{\Phi}$ and $\mathbf{v} = \mathbf{W}\mathbf{y}$, with $\mathbf{W}$ diagonal matrix having diagonal elements $\mathbf{W}_{ii} = w_i$. Finally, a locally weighted regression model (LWR) is fitted solving the equation $(\mathbf{Z}^T\mathbf{Z})\boldsymbol{\beta} = \mathbf{Z}^T\mathbf{v}$ and the prediction of the value $f(\boldsymbol{\varphi}_q)$ is obtained evaluating such a model in the query point:

$$\hat{y}_q = \boldsymbol{\varphi}_q^T(\mathbf{Z}^T\mathbf{Z})^{-1}\mathbf{Z}^T\mathbf{v}. \tag{1}$$

For an analysis of this method in terms of approximation properties, see Cybenko (1996).

Here, we will focus mainly on the procedural aspects of the modeling technique. Typically, the data analyst who adopts a local regression approach, has to take a set of decisions related to the model (e.g. the number of neighbors, the weight function, the parametric family, the fitting criterion to estimate the parameters). We extend the classical approach with a method that automatically selects the adequate configuration. To this aim, we simply import tools and techniques from the field of linear statistical analysis. The most important of these tools is the PRESS statistic (Myers, 1990), which is a simple, well-founded and economical way to perform *leave-one-out* cross validation (Efron & Tibshirani, 1993) and therefore to assess the performance in generalization of local linear models. Due to its short computation time which allows its intensive use, it is the key element of our approach to modeling data. Assessing the performance of each linear model, alternative configurations can be tested and compared in order to select the best one. This same selection strategy is indeed exploited to select the training subset among the neighbors, as well as various structural aspects like the features to treat and the degree of the polynomial used as a local approximator. The general ideas of the approach can be summarized as follows.

1. The task of learning an input-output mapping is decomposed in a series of linear estimation problems.
2. Each single estimation is treated as an optimization problem in the space of alternative model configurations.

3. The estimation ability of each alternative model is assessed by the cross-validation performance computed using the PRESS statistic.

In order to make these operations more effective, we propose two innovative algorithms in the *lazy learning* method:

– a recursive algorithm for the parametric estimation and the cross-validation of each local model. This method avoids having to restart each model evaluation from scratch and decreases noticeably the computational cost

– a more rigorous statistical test to compare the performance of two alternative candidate models. The test does not just consider the average values of the cross-validation errors but also their sampling distributions.

In the next two sections we will discuss these algorithms in detail.

2.1 The PRESS statistic and the recursive method

We will focus on the *leave-one-out* cross-validation procedure (Efron & Tibshirani, 1993), which returns a reliable estimation of the prediction error in $\boldsymbol{\varphi}_q$. We define the i-th *leave-one-out* error $e^{\mathrm{cv}}(i)$ as the difference between y_i and the prediction given by the local model centered in $\boldsymbol{\varphi}_q$ and fitted using all the examples available but the i-th. An estimation of the prediction error in $\boldsymbol{\varphi}_q$ is given by the average of the errors $e^{\mathrm{cv}}(i)$ each weighted according to the distance $d(\boldsymbol{\varphi}_i, \boldsymbol{\varphi}_q)$. When considering local linear model, the *leave-one-out* cross-validation can be performed without recalculating the regression parameter for each excluded example thanks to the local version of the PRESS statistic (Atkeson *et al.*, 1997a):

$$\begin{aligned} \mathrm{MSE}^{\mathrm{cv}}(\boldsymbol{\varphi}_q) &= \frac{1}{\sum_i w_i^2} \sum_i \left(\frac{v_i - z_i^T (\mathbf{Z}^T\mathbf{Z})^{-1}\mathbf{Z}^T\mathbf{v}}{1 - z_i^T (\mathbf{Z}^T\mathbf{Z})^{-1} z_i} \right)^2 \\ &= \frac{1}{\sum_i w_i^2} \sum_i \left(w_i \frac{y_i - \boldsymbol{\varphi}_i^T (\mathbf{Z}^T\mathbf{Z})^{-1}\mathbf{Z}^T\mathbf{v}}{1 - z_i^T (\mathbf{Z}^T\mathbf{Z})^{-1} z_i} \right)^2 \\ &= \frac{1}{\sum_i w_i^2} \sum_i \left(w_i \, e^{\mathrm{cv}}(i) \right)^2. \end{aligned} \tag{2}$$

In our modeling procedure the performance of a model in cross validation is the criterion adopted to choose the best local model configuration. One of the most important parameters to be tuned in a local model configuration is the size of the region surrounding $\boldsymbol{\varphi}_q$, in which the function $f(\cdot)$ can be conveniently approximated by a linear local model. Such a parameter can be related to the number of training examples which fall into the region of linearity. The task of identifying the region of linearity is therefore akin to the task of finding, among the examples available, the number n of neighbors of $\boldsymbol{\varphi}_q$ to be used in the local regression fit. Thus, we consider different models, each fitted on a different number of examples, and we use the *leave-one-out* cross-validation to compare them and to select the one for which the predicted error is smaller.

To make the procedure faster and to avoid repeating for each model the parameter and the PRESS computation, we adopt an incremental approach based on recursive linear techniques. Recursive algorithms have been developed in model identification and adaptive control literature (Goodwin & Sin, 1984) to identify a linear model when data are not available from the beginning but are observed sequentially. Here we employ these methods to obtain the parameters of the model fitted on $n+1$ nearest neighbors by updating the parameters of the model with n examples. Furthermore, the *leave-one-out* errors $e^{\mathrm{cv}}(i)$ are obtained exploiting partial results from the least square method and do not require additional computational overload. Once adopted as the weighting kernel the indicator function which assigns $w_i = 1$ to the examples used to fit the model and $w_i = 0$ to the others, the recursive *lazy learning* algorithm is described as follows:

$$\begin{cases} \mathbf{P}_{n+1} = \mathbf{P}_n - \dfrac{\mathbf{P}_n \boldsymbol{\varphi}_{n+1} \boldsymbol{\varphi}_{n+1}^T \mathbf{P}_n}{1 + \boldsymbol{\varphi}_{n+1}^T \mathbf{P}_n \boldsymbol{\varphi}_{n+1}} \\ \boldsymbol{\gamma}_{n+1} = \mathbf{P}_{n+1} \boldsymbol{\varphi}_{n+1} \\ e_{n+1} = y_{n+1} - \boldsymbol{\varphi}_{n+1}^T \boldsymbol{\beta}_n \\ \boldsymbol{\beta}_{n+1} = \boldsymbol{\beta}_n + \boldsymbol{\gamma}_{n+1} e_{n+1} \end{cases} \tag{3}$$

$$e_{n+1}^{\mathrm{cv}}(i) = \frac{y_i - \boldsymbol{\varphi}_i^T \boldsymbol{\beta}_{n+1}}{1 + \boldsymbol{\varphi}_i^T \mathbf{P}_{n+1} \boldsymbol{\varphi}_i}$$

where $\left(\boldsymbol{\varphi}_{n+1}, y_{n+1}\right)$ is the $n+1$-th nearest neighbor of the query point, $\mathbf{P}_n$ is the recursive approximation of the matrix $(\mathbf{Z}^T\mathbf{Z})^{-1}$, $\boldsymbol{\beta}_n$ denotes the optimal least squares parameters of the model fitted on the n nearest neighbor, and $e_n^{\text{cv}}(i)$, with $1 \leq i \leq n$, is the vector $\mathbf{E}_n^{\text{cv}}$ of *leave-one-out* errors. Once this vector is available, the formula (2) is easily computed. This value is a weighted average of the cross-validated errors and is the simplest statistic that can be used to describe the performance of the model defined by n neighbors. However, the problem of assessing the right dimension of the linearity region using a limited number of samples affected by noise requires a more powerful statistical procedure. In the following section, we will discuss in detail the method we adopt.

2.2 The statistical test for model selection

The recursive method described in the previous section returns for each size n of the neighborhood a vector $\mathbf{E}_n^{\text{cv}}$ of *leave-one-out* errors. In order to select the best model, our procedure (described in detail in Fig. 1) consists in increasing the number of neighbors considered when identifying the local model, until the model performance deteriorates and a departure from the region of local linearity is detected. This requires a statistical test to evaluate when the enlarged model is significantly worse than those already considered. In terms of hypothesis testing, we formulate the null hypothesis H_0 that $\mathbf{E}_n^{\text{cv}}$ and $\mathbf{E}_{n+1}^{\text{cv}}$ belong to the same distribution. To evaluate this hypothesis we use a nonparametric permutation test (Siegel & N.J. Castellan, 1988) which does not require any assumptions about normality, homogeneity of variance, or about the shape of the underlying distribution. We adopt a paired version of the permutation algorithm (see appendix 6.1) because of the correlation between the two error vectors .

The permutation test shows one of the main advantage of a local modeling procedure: with low computational effort it is possible to return along with the prediction and the linear local parameters also a statistical description of the uncertainty affecting these results. This property can result useful both for prediction and for control problems.

1. $n = 0$: initialize the parameter of the local model (e.g. conventional initialization: $\boldsymbol{\beta}_0 = 0$, $\mathbf{P}_0 = \lambda \mathbf{I}$ with a large λ).

2. Add the example $\big(\boldsymbol{\varphi}_{n+1}, y_{n+1}\big)$. The parameter $\boldsymbol{\beta}$ is updated and the vector $\mathbf{E}_n^{\mathrm{cv}}$ is evaluated using Equation (3).

3. Check for a departure from the local linear region comparing the new model with the one previously accepted. If the new model is significantly worse goto 4, else accept the model and goto 2 to consider adding further examples.

4. Stop the recursive identification procedure and return the parameters of the last model accepted.

Fig. 1. The procedure adopted to identify recursively local linear model and to define the largest region of linearity.

3 Lazy learning for control design

To illustrate the different approaches to *lazy learning* control design, we will first define some notations for the single input single output (SISO) case. Consider a class of discrete-time dynamic systems whose equations of motion can be expressed in the form:

$$y(k) = f\big(y(k-1), \ldots, y(k-ny), u(k-d), \ldots, u(k-d-nu), e(k-1), \ldots, e(k-ne)\big) + e(k), \quad (4)$$

where k denotes the time, $y(k)$ is the system output, $u(k)$ the input, $e(k)$ is a zero-mean disturbance term, $d > 0$ is the relative degree and $f(\cdot)$ is some nonlinear function. This model is known as the NARMAX model (Leontaritis & Billings, 1985). Let us assume we have no physical description of the function $f(\cdot)$ but a limited amount of pairs $\big[u(k), y(k)\big]$ is available. Defining the information vector as

$$\boldsymbol{\varphi}(k-1) = \big[y(k-1), \ldots, y(k-ny), u(k-d), \ldots, u(k-d-nu), e(k-1), \ldots, e(k-ne)\big], \quad (5)$$

the system (4) can be written in the form:

$$y(k) = f\left(\boldsymbol{\varphi}(k-1)\right) + e(k). \tag{6}$$

It is demonstrated that models (4) can describe the input-output behavior of general state-space nonlinear dynamic systems:

$$\begin{aligned} \mathbf{x}(k+1) &= g\left(\mathbf{x}(k), u(k)\right) + \mathbf{v}(k) \\ y(k) &= h\left(\mathbf{x}(k)\right) + w(k), \end{aligned} \tag{7}$$

where $\mathbf{x} \in \Re^p$ is the state vector, $\mathbf{v} \in \Re^p$ and $w \in \Re$ are zero-mean disturbance and noise, and $g\colon \Re^{p+1} \to \Re^p$, $h\colon \Re^p \to \Re$ are some nonlinear functions.

3.1 Lazy learning and local linear control: a comparative analysis

Although nonlinearity characterizes most real control problems, methods for analysis and control design are considerably more powerful and theoretically founded for linear systems than for nonlinear ones. In nonlinear control literature we find therefore many approaches based on the extension of linear techniques to nonlinear problems. In the following, a short survey of these techniques and a comparison with the *lazy* approach are provided.

Linearization about an equilibrium point. A point $\boldsymbol{\varphi}^* = \boldsymbol{\varphi}(k)$ is called an equilibrium point of the plant (6) if, for each time step k, $\boldsymbol{\varphi}(k) = \boldsymbol{\varphi}(k-1)$, where $y(k) = f(\boldsymbol{\varphi}(k-1))$ is the first term in the information vector (5). Assuming that $f(\cdot)$ is continuously differentiable at $\boldsymbol{\varphi}^*$, we can linearize the equation (6) by performing a multi-variable Taylor series expansion. The outcome is a linear time invariant system that describes locally the nonlinear dynamics. Under some conditions (Slotine & Li, 1991) this linear model provides informations about the local stability properties of the global system. Furthermore, starting from its parametric form, a linear controller can be designed to stabilize (6) around $\boldsymbol{\varphi}^*$. A major drawback of this procedure consists in a not accurate modeling when the system is operating away from the equilibrium point. An alternative is provided by linearizing along a trajectory.

Linearization about a trajectory. The idea is to study the behavior of the system near a prescribed trajectory. Let $\boldsymbol{\varphi}^*(k)$ denote a specific trajectory of the nonlinear system (4), that is $y^*(k) = f(\boldsymbol{\varphi}^*(k-1))$ with $\boldsymbol{\varphi}^*(k)$ information vector at time k. Assuming that $f(\cdot)$ is continuously differentiable, the system (6) may be approximated near the trajectory $\boldsymbol{\varphi}^*(k)$ by a linear time-varying system. Let us remark that the time-varying property makes the control design process more difficult. Moreover, this approach requires the knowledge of the trajectory in advance, condition that unfortunately is not always satisfied.

Gain scheduling. It is one of the most widely and successfully applied technique for the design of nonlinear controllers. This method breaks the control design process in two steps. First, one designs local linear controllers based on linearizations at several different equilibrium points. Then, a scheme is implemented for interpolating (scheduling) the parameters at intermediate conditions. For a formal analysis of this approach see (Rugh, 1991).

Adaptive control. Here an identification algorithm (RLS) operates all the time to update a linear approximation to the system dynamics, whose parameters are used to adjust the coefficients of the controller (Goodwin & Sin, 1984). Such an approximation can provide satisfactory performance only if the local linearization changes slowly. Some variants of the approach (*forgetting factor*) make the recursive algorithm more sensitive to recent data in order to better track variations in the plant transfer function.

Local model networks (LMN). This approach extends the concept of operating point by introducing the notion of *operating regime*. An operating regime is a set of operating points where the system behaves approximately linearly (Johansen & Foss, 1993), (Johansen & Foss, 1995). To each of them a validity region and a local description of the system behavior are associated. The function $f(\cdot)$ is then approximated with a set of interpolated local models. The use of local model networks in identification and control has been proposed by several authors (Murray-Smith & Hunt, 1995). One major advantage of this approach is the possibility to integrate a priori knowledge with parametric learning procedures. Related nonlinear modeling approaches

are Takagi-Sugeno (1985) fuzzy inference systems and radial basis functions (Moody & Darken, 1989).

Multiple model adaptive control. Like LMN, it is a model-based control strategy that uses a weighted combination of model/controller pairs (Narendra & Balakrishnan, 1997). The main difference lies in the procedure used to combine the models. Unlike LMN where the operating domain is partitioned by considering a priori knowledge or pre-processing observed data, this approach combines the local model/controllers according to some local measures of accuracy (e.g. maximum a posteriori probability and horizon-based error tracking) estimated on-line during the control process (Schott & Bequette, 1997). In addition, the weighting measures are used to select which local models to update with the incoming sample.

Let us now discuss the main differences between the above mentioned approaches and the *lazy learning* technique.

Lazy learning vs. Linearization. A linearization approach requires an a priori knowledge of the system, in order to have an analytical characterization of the equilibrium points. *Lazy learning* does not require an analytical model, but returns the best linear approximation that can be extracted from observed data. Linearization techniques return a local linear model whose range of validity is restricted to a neighborhood of the linearization points. Memory-based techniques can adaptively change the local description as a function of the current system state.

Lazy learning vs. Adaptive control. The standard recursive procedure embedded in the adaptive controller estimates only one linear model which is the best linear approximation on the basis of past observations. The adoption of a forgetting factor aims to make the algorithm able to deal both with nonlinear and time-varying configurations by giving more weight to the most recent data. On the contrary, the *lazy* approach treats separately nonlinear and time-varying systems. The set of data considered to estimate the local regression parameters is the set of nearest data in the input space domain, i.e. the information vector space in the case of (6). This allows different local models for different operating regimes and then

prevents from the problem of *data interference* (Jacobs *et al.*, 1991) (Salganicoff, 1997) — a.k.a. *stability-plasticity dilemma* (Carpenter & Grossberg, 1988). Consider, as illustrative example, the simple nonlinear dynamics $y(k) = f(u(k-1))$ of which we plot six samples in Fig. 2. Let the numbers represent the temporal order with which the samples have been observed. If the system is identified with a forgetting factor recursive approach, when the example no. 6 is encountered, the estimated model (dotted line) has lost memory of the dynamics existing in the neighborhood of the points 1 and 2. As a result, the accuracy of the RLS approximation (dotted line) in 6 is poor. On the other hand, the *lazy* approach is not affected by any interference phenomenon (from data 4 and 5) and returns a better local approximation (solid line). Finally, the *lazy* technique can deal with time-varying configurations with minor changes. It is sufficient to extend the input space by adding to the input features the current time variable k. Once a prediction is required, the nearest samples in space and time will be the candidate neighbors.

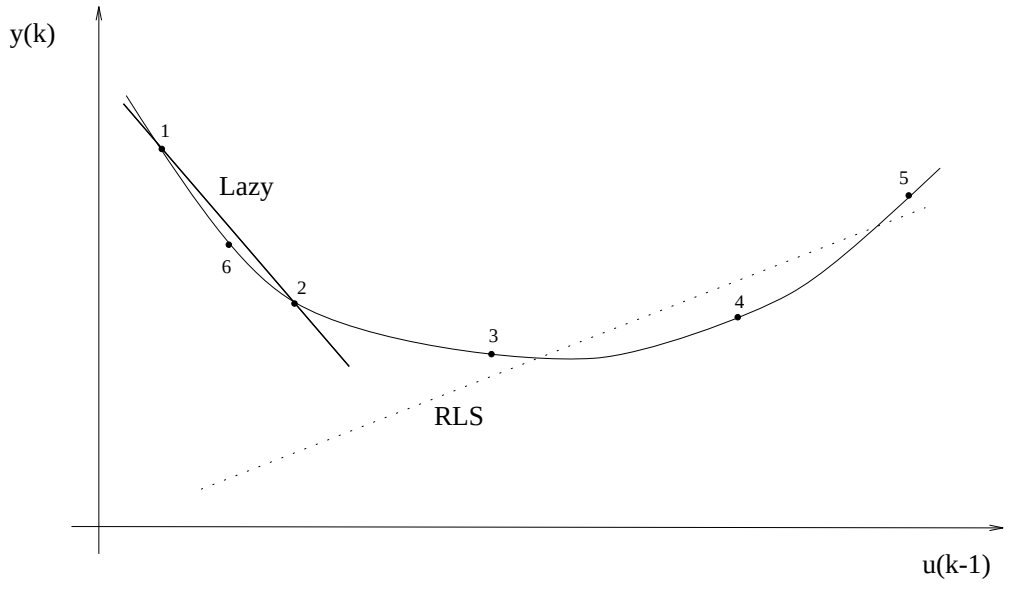

Fig. 2. Recursive vs. Lazy learning identification

Lazy learning vs. Gain scheduling. Here the same remarks made about the linearization approach are valid. A further major difficulty in the gain scheduling approach is the selection of appro-

priate scheduling variables. The *lazy learning* selects instead the input features on the basis of the information vector (5) which represents the most recent operating condition of the system.

Lazy learning vs. Local model networks. The two approaches share the common idea of decomposing a difficult problem in simpler local problems. The main differences concern the model identification procedure. Local model networks aim to estimate a global description to cover the whole system operating domain, whereas memory based techniques focus simply on the current operating point. LMN result more time consuming in identification but faster in prediction. However when new data are observed, model update may require to perform the whole LMN modeling process from scratch. On this matter *lazy learning* takes an advantage from the absence of a global model: once a new input-output example is observed, it is enough to update the database which stores the set of input-output pairs.

Lazy learning vs. Multiple model adaptive control. Lazy learning is situated in a middle ground between LMN and multiple model control. Like in LMN, in *lazy learning* the model scheduling obeys a criterion of locality in the space of state variables, and not a performance measure over the preceding time instants. Like in multiple model control, the *lazy learning* model is sequentially adapted to the observed data, even though this is done indirectly by updating the database.

3.2 The lazy learning gradient-based controller

As discussed above, *lazy learning* can approximate complex nonlinear mappings using a limited amount of data. The idea of the *lazy* gradient-based controller is to use this property to solve a one step horizon control problem as an optimization problem. Consider a dynamic system of which only a set of observed input-output samples is available. For clarity, we assume $d = 1$. Suppose that the system, formulated in the NARMAX form (4), is required to reach at the next time time step a reference value y_{ref}. Since at time $k-1$ the value $u(k-1)$ is not available yet,

we introduce the vector

$$\boldsymbol{\varphi}_s(k-1) = \big[y(k-1), \ldots, y(k-ny), u(k-2), \ldots, u(k-nu), e(k-1), \ldots, e(k-ne)\big], \quad (8)$$

as a subset of the information vector (5). The *lazy* model (1) can be used to predict the response of the system to the control action u^i:

$$\hat{y}_{u^i}(k) = \hat{f}\big(y(k-1), \ldots, y(k-ny), u^i, u(k-2), \ldots, u(k-nu), e(k-1), \ldots, e(k-ne)\big). \quad (9)$$

In addition, the linearization returned by the local description provides an estimate of the gradient of the system output $\frac{\mathrm{d}\hat{y}_{u^i}(k)}{\mathrm{d}u^i}$ with respect to the control action (Atkeson *et al.*, 1997a). The control problem can therefore be formulated as a constraint gradient based optimization problem:

$$u^{\mathrm{opt}} = arg \min_{u^i} J(u^i) = arg \min_{u^i} (y_{ref} - \hat{y}_{u^i}(k))^2. \quad (10)$$

In order to speed up the optimization resolution, the algorithm can be initialized with the value returned from the model of the inverse dynamics (Jordan & Rumelhart, 1992):

$$u^0 = \hat{f}_{inv}\big(y_{ref}, y(k-1), \ldots, y(k-ny), u(k-2), \ldots, u(k-nu), e(k-1), \ldots, e(k-ne)\big). \quad (11)$$

A detailed version of the *lazy* gradient-based control algorithm is presented in Fig. 3.

3.2.1 Lazy gradient-based control analysis

It is well known that the parameterization of a plant is extremely important to prove its stability. Even if powerful stability results have been obtained in the control of linear time-invariant systems using the Lyapunov method, the same techniques cannot be applied in a generic nonlinear case. In the method discussed above, *lazy learning* is used as a black-box approximator of the input-output representation of a nonlinear dynamical system. The result is a nonlinear controller for which the stability of the closed loop feedback system cannot be guaranteed theoretically for a generic nonlinear plant.

Instability can be a consequence of non minimum-phase configurations. In this case the inverse dynamics is unstable and methods like those described before cannot prevent the control signal

1. Initialization of the algorithm with the value u^0 provided by the inverse mapping (11).

2. Prediction of the outcome $\hat{y}_{u^i}$ of the system forced by the input u^i.

3. Computation of the gradient vector $\frac{\mathrm{d}J(u^i)}{\mathrm{d}u^i}$.

4. Updating of the control sequence $u^i \rightarrow u^{i+1}$. The optimization step is performed by a constrained gradient based algorithm implemented in the Matlab function `constr` (Grace, 1994).

5. If the minimum has been reached ($u^i = u^{i+1}$) goto 6 else goto 2.

6. Control action execution. The action $u(k-1) = u^{\mathrm{opt}}$ is applied to the real system.

7. Updating of the database by storing the new input-output observation.

Repeat these steps at each sampling period.

Fig. 3. The lazy gradient-based controller algorithm

from growing without limit, making the closed loop system unstable. This problem is demonstrated by the simulation examples in sections 4.2 and 4.3. While the technique can successfully control a complex minimum-phase nonlinear system, it is not able to regulate a system whose linearization about some operating regimes is non minimum-phase.

The concept of minimum-phase is significantly more complex in the nonlinear case than in the linear one. Hence, a possible solution may come from the adoption of linear techniques for solving locally the non minimum-phase problem. In the next section we present a control technique which can avoid these instability phenomena by using conventional linear techniques in a nonlinear context.

3.3 The Lazy learning self-tuning controller

In this section we propose a hybrid architecture for the indirect control of nonlinear discrete-time plants from their observed input-output behavior. An indirect control scheme (Astrom & Witten-

mark, 1990), (Narendra & Annaswamy, 1989) combines a parameter estimator, which computes an estimate $\boldsymbol{\vartheta}$ of the unknown parameters, with a control law $u(k) = K(\boldsymbol{\varphi}(k), \boldsymbol{\vartheta}^*)$ implemented as a function of the plant parameters. In the adaptive version, the estimator generates the estimate $\boldsymbol{\vartheta}(k)$ at each sampling period k by processing the observed input-output behavior. This estimate is then assumed to be a specification of the real plant and used to compute the control law $u(k) = K(\boldsymbol{\varphi}(k), \boldsymbol{\vartheta}(k))$ (certainty equivalence paradigm). In conventional adaptive control theory, to make the problem analytically tractable, the plant is assumed to be a linear time-invariant system with unknown parameters.

Our approach combines the local learning identification procedure described in the section 2 with conventional linear control techniques. We adopt the minimum-variance (MV) and the pole-placement (PP) control technique, borrowed from linear self-tuning controllers (Astrom & Wittenmark, 1990).

The MV control algorithm was first formulated in (Astrom, 1967). Since then the MV technique has had many practical applications and significant theoretical developments. The reasons for MV popularity lie in its simplicity and ease of interpretation and implementation. Let us consider a linear discrete-time process described in input-output form by the equation:

$$A(z)y(k) = z^{-d}B(z)u(k) + C(z)e(k), \tag{12}$$

and suppose we want to regulate it to $y_{ref} = 0$. The MV control problem can be stated as finding the control law which minimizes the variance of the output. The MV controlled closed loop system is stable only if B has all of its roots inside the unit circle (minimum phase). However, more complex formulations are available in the case of a tracking problem or in the case of non minimum-phase systems (Generalized MV or GMV). In these cases it is possible to select properly the controller parameters in order to make the closed loop system asymptotically stable.

Pole placement design is an alternative technique to deal with non minimum-phase configurations. The procedure requires first to choose the desired closed loop pole positions and then to calculate the appropriate controller.

Both these design techniques require a model formulation in the form (12). However in our approach the linearization is performed also in configurations which are far from the equilibrium locus. As a consequence, the relation between the input u and the output y is given by

$$A(z)y(k) = z^{-d}B(z)u(k) + C(z)e(k) + b \tag{13}$$

where b is an offset term. This requires a slight modification to the formulas for GMV and PP controller design (see appendices 6.2 and 6.3).

The proposed control algorithm is described in detail in Fig. 4. Note that the selection of neighbors is made considering only the subset vector (8) of the information vector. In fact $u(k-1)$ cannot be available as it is the expected outcome of the procedure. Anyway, the local weighted regression is performed in the space of the complete information vector (5).

1. Acquisition of the the vector (8).
2. Linearization of the function $f(\cdot)$ about (8). The linearization is computed by the *lazy learning* algorithm.
3. Derivation of the polynomials A, B, C and the offset b of (13) from the linearized model.
4. Design of a MVG/PP controller for (13) which satisfies the required properties (stability, accuracy, speed ...) of the closed loop behavior.
5. Computation of the control signal.
6. Updating of the database by storing the new input-output observation.

Repeat these steps at each sampling period.

Fig. 4. The lazy self-tuning controller algorithm

3.3.1 Lazy self-tuning control analysis

In the *lazy* self-tuning regulator the nonlinear plant (6) is parameterized as a linear system where parameters are changing with the observable state. This means that the nonlinear model can be written as a linear model where parameters vary with the state of the system. This configuration recalls the linear parameter varying (LPV) configuration introduced by (Shamma & Athans, 1992) in his analytical analysis of gain scheduling controllers, or the state-dependent models presented by (Priestley, 1988). These models can be represented as follows:

$$A(\boldsymbol{\varphi}(k))y(k) = z^{-d}B(\boldsymbol{\varphi}(k))u(k) + C(\boldsymbol{\varphi}(k))e(k). \tag{14}$$

Let us now assume that there exists a LPV model which represents in a sufficiently accurate manner the nonlinear system (6). If we make the hypothesis of complete controllability and observability, the closed-loop system may be put into the state-space form

$$\mathbf{x}(k+1) = \mathbf{F}\big(\boldsymbol{\varphi}(k)\big)\mathbf{x}(k) + \mathbf{v}(k) \tag{15}$$

This representation allows us to analyze the stability of our controller. If the regulator is designed such that the eigenvalues of $\mathbf{F}(\boldsymbol{\varphi}(k))$ are stable, then the system (15) will be asymptotically stable for any fixed value of $\boldsymbol{\varphi}$ (frozen time stability). However, this is not a sufficient condition for uniform asymptotic stability of the system. If we consider the set of values assumed by the matrix $\mathbf{F}(\boldsymbol{\varphi}(k))$ a sufficient condition for uniform stability (Tanaka & Sugeno, 1992) is that it exists a common matrix $\mathbf{P} > 0$ such that:

$$\mathbf{F}(\boldsymbol{\varphi}(k))^T\mathbf{P}\mathbf{F}(\boldsymbol{\varphi}(k)) - \mathbf{P} < 0 \qquad \text{for all k.} \tag{16}$$

With the pole placement technique we can impose the same stable closed loop transfer function for all k. It follows that there exists a matrix $\mathbf{P}$ that satisfies the equation (16). Then under the following assumptions:

– the system is completely controllable and observable;

– the system (6) can be put in the form (14);

– the approximation error of the *lazy learning* identifier is negligible;

the equilibrium of the nonlinear system (6) controlled by the *lazy* PP self-tuning controller is globally asymptotically stable.

3.4 The lazy learning optimal controller

Consider the optimal control problem of the nonlinear system (7) over a finite horizon time. Using a quadratic cost function, the solution to an optimal control problem is the control sequence U that minimizes

$$J = \frac{1}{2}\mathbf{x}(t_f)^T\mathbf{P}_f\mathbf{x}(t_f) + \frac{1}{2}\sum_{k=0}^{t_f}\left|\mathbf{x}(k)^T\mathbf{u}(k)^T\right|\begin{vmatrix}\mathbf{Q}_k & \mathbf{M}_k\\ \mathbf{M}_k^T & \mathbf{R}_k\end{vmatrix}\left|\mathbf{x}(k)\mathbf{u}(k)\right|, \tag{17}$$

with $\mathbf{Q}_k$, $\mathbf{M}_k$, $\mathbf{R}_k$, $\mathbf{P}_f$ weighting terms designed a priori. While analytic results are not available for a generic nonlinear configuration, optimal control theory (Stengel, 1986) provides the solution for the linear case. In the following, we will present the nonlinear problem in a linear time varying setting.

Consider the trajectory of the dynamical system once forced by an input sequence $U = \big[\mathbf{u}(0), \mathbf{u}(1), \ldots, \mathbf{u}(t_f - 1)\big]$. Assume that the system can be linearized about each state of the trajectory. Neglecting the residual errors due to the first order Taylor series approximation, the behavior of the linear system along a generic trajectory is the behavior of a linear time varying system whose state equations can be written in the form

$$\begin{aligned}\mathbf{x}(k+1) &= \mathbf{F}\big(\boldsymbol{\varphi}(k)\big)\mathbf{x}(k) + \mathbf{G}\big(\boldsymbol{\varphi}(k)\big)\mathbf{u}(k) + \mathbf{K}\big(\boldsymbol{\varphi}(k)\big)\\ &= \mathbf{F}_k\mathbf{x}(k) + \mathbf{G}_k\mathbf{u}(k) + \mathbf{K}_k,\end{aligned} \tag{18}$$

with $\mathbf{F}_k$, $\mathbf{G}_k$, $\mathbf{K}_k$ parameters of the system linearized about the query point $\boldsymbol{\varphi}(k)$. $\mathbf{K}_k$ is an offset term that equals zero in equilibrium points. This term requires a slight modification in the linear controller formulation. However, in order to simplify the notation, in the following we will neglect the constant term.

Optimal control theory provides the solution for the linear time-varying system (18). At each time step the optimal control action is

$$\mathbf{u}(k) = -\left(\mathbf{R}_k + \mathbf{G}_k^T \mathbf{P}_{k+1} \mathbf{G}_k\right)^{-1} \left(\mathbf{M}_k^T + \mathbf{G}_k^T \mathbf{P}_{k+1} \mathbf{F}_k\right) \mathbf{x}(k), \tag{19}$$

where $\mathbf{P}_k$ is the solution to the backward Riccati equation.

$$\mathbf{P}_k = \mathbf{Q}_k + \mathbf{F}_k^T \mathbf{P}_{k+1} \mathbf{F}_k - \left(\mathbf{M}_k + \mathbf{F}_k^T \mathbf{P}_{k+1} \mathbf{G}_k\right) \left(\mathbf{R}_k + \mathbf{G}_k^T \mathbf{P}_{k+1} \mathbf{G}_k\right)^{-1} \left(\mathbf{M}_k^T + \mathbf{G}_k^T \mathbf{P}_{k+1} \mathbf{F}_k\right), \tag{20}$$

having as final condition

$$\mathbf{P}(t_f) = \mathbf{P}_f. \tag{21}$$

The piecewise-constant optimal solution is obtained by solving the Euler-Lagrange equations which are the three necessary and sufficient conditions for optimality when the final time is fixed.

$$0 = \frac{\partial H_k}{\partial \mathbf{u}_k} = \mathbf{x}_k^T \mathbf{M}_k + \mathbf{u}_k^T \mathbf{R}_k + \lambda_{k+1}^T \mathbf{G}_k \tag{22}$$

$$\lambda_k^T = \frac{\partial H_k}{\partial \mathbf{x}_k} = \mathbf{x}_k^T \mathbf{Q}_k + \mathbf{u}_k^T \mathbf{M}_k^T + \lambda_{k+1}^T \mathbf{F}_k \tag{23}$$

$$\lambda_f^T = \mathbf{x}_f^T \mathbf{P}_f \tag{24}$$

with $\lambda_k = \mathbf{P}_k \mathbf{x}_k$ adjoint term in the augmented cost function (Hamiltonian):

$$H_k = J + \lambda_{k+1}^T \big(\mathbf{F}_k \mathbf{x}(k) + \mathbf{G}_k u(k)\big). \tag{25}$$

The Euler-Lagrange equations do not hold for nonlinear systems. Anyway, if the system can be represented in the form (18), formula (22) can be used to compute the derivative of the cost function (17) with respect to a control sequence U. This requires at each time k the matrices $\mathbf{F}_k$ and $\mathbf{G}_k$ that can be obtained by linearizing the system dynamics along the trajectory forced by the input sequence.

As discussed in section 2, our modeling procedure performs system linearization with minimum effort, no a priori knowledge and only a reduced amount of data. Hence, we propose an algorithm for nonlinear optimal control, formulated as a gradient based optimization problem and based on the local system linearization.

The algorithm searches for the sequence of input actions

$$U^{\text{opt}} = arg \min_{U^i} J(U^i), \tag{26}$$

that minimizes the finite-horizon cost function (17) along the future t_f steps. The cost function $J(U^i)$ for a generic sequence U^i is computed simulating forward for t_f steps the model identified by the local learning method. The gradient of $J(U^i)$ with respect to U^i is returned by (22).

These are the basic operations of the optimization procedure (described in detail in Fig. 5) executed each time a control action is required:

- forward simulation of the *lazy* model forced by a finite control sequence U^i of dimension t_f;
- linearization of the simulated system about the resulting trajectory;
- computation of the resulting finite cost function $J(U^i)$;
- computation of the gradient of the cost function with respect to simulated sequence;
- updating of the sequence with a gradient based algorithm.

Once the search algorithm returns U^{opt}, the first action of the sequence is applied to the real system (*receding horizon* control strategy (Clarke, 1994)). Let us remark how the *lazy learning* model has a twofold role in the algorithm in Fig. 5: (i) at step 2 it is an estimator which predicts the behavior of the system once forced with a generic input sequence, (ii) at step 3 it returns a linear approximation of the system's dynamics.

3.4.1 Lazy optimal control analysis

Atkeson et al. (1997b), Tanaka (1995) and (Passino & Yurkovich, 1996) applied infinite-time LQR regulator to nonlinear systems linearized with *lazy learning* and neuro-fuzzy models. The drawback of these approaches is that an equilibrium point or a reference trajectory is required. Moreover, they made the strong assumption that the state of the system will remain indefinitely in a neighborhood of the linearization point. As discussed above, the advantage of the proposed approach is that these requirements do not need to be satisfied. First, *lazy learning* is able to

1. Initialization of the algorithm with a random sequence of actions U_s^0.

2. Forward simulation of the system forced by the sequence $U_s^i = \left[\mathbf{u}_s^i(k), \mathbf{u}_s^i(k+1), \ldots, \mathbf{u}_s^i(k+t_f-1)\right]$, where $\mathbf{u}_s^i(j)$ denotes the action applied to the simulated system at time j. The system behavior is predicted using the model identified by the local learning method.

3. Formulation of the nonlinear system in the time-varying form. The parameters $\mathbf{F}_j$, $\mathbf{G}_j$, $\mathbf{K}_j$, with $j = k, \ldots, k+t_f-1$, are returned by the local model identification.

4. Backward resolution of the discrete-time Riccati equation (20) for the resulting time-varying system.

5. Computation of the cost function (17).

6. Computation of the gradient vector $\frac{\partial J}{\partial U_s^i} = \left[\frac{\partial J}{\partial \mathbf{u}_s^i(k)}, \frac{\partial J}{\partial \mathbf{u}_s^i(k+1)}, \ldots, \frac{\partial J}{\partial \mathbf{u}_s^i(k+t_f-1)}\right]$ by using formula (22).

7. Updating of the control sequence $U_s^i \rightarrow U_s^{i+1}$. The optimization step is performed by a constrained gradient based algorithm implemented in the Matlab function `constr` (Grace, 1994).

8. If the minimum has been reached ($U_s^i = U_s^{i+1}$) goto 9 else goto 2.

9. Control action execution. The first action $\mathbf{u}_s^{\text{opt}}(k)$ of the sequence U_s^{opt} is applied to the real system.

10. Updating of the database by storing the new input-output observation.

Repeat these steps at each sampling period.

Fig. 5. The lazy learning optimal controller algorithm

linearize a system in points far from equilibrium. Second, the time varying approach makes possible the use of a linear control strategy even though the system operates within different linear regimes.

There are no demonstrated stability properties for linear time-varying optimal control. However, with respect to the other approaches this control strategy presents some nice properties. Firstly, it can easily deal with MIMO (multi-input multi-output) systems, as shown in the simulation example 4.4. Secondly, it allows control policies on a longer time horizon than the simple relative degree of the system. Finally, this formalism can keep into consideration the uncertainty affecting the model. In our controller we make the assumption that the parameters returned by the local models are a real description of the local behavior (certainty equivalence principle). However, this is a restricting assumption which requires a sufficient degree of accuracy in the approximation. Optimal control theory can represent a possible solution to this limitation. In fact, stochastic optimal control theory provides a formal solution to the problem of parameter uncertainty in control systems — dual control (Fel'dbaum, 1965). Furthermore, our modeling procedure can return at no additional cost a statistical description of the estimated parameters (see Section 2.2). Future work will focus on the extension of this technique to the stochastic control case.

4 Simulation studies

4.1 The identification of a nonlinear discrete-time system

Our approach has been applied to the identification of a complex nonlinear benchmark proposed by Narendra and Li (1996). The discrete-time equations of the system are:

$$\begin{cases} \mathbf{x}_1(k+1) &= \left(\dfrac{\mathbf{x}_1(k)}{1+\mathbf{x}_1^2(k)}+1\right)\sin\big(\mathbf{x}_2(k)\big) \\ \mathbf{x}_2(k+1) &= \mathbf{x}_2(k)\cos\big(\mathbf{x}_2(k)\big)+\mathbf{x}_1(k)\,\mathrm{e}^{-\frac{\mathbf{x}_1^2(k)+\mathbf{x}_2^2(k)}{8}} \\ &\quad +\dfrac{u^3(k)}{1+u^2(k)+0.5\cos\big(\mathbf{x}_1(k)+\mathbf{x}_2(k)\big)} \end{cases} \tag{27}$$

$$y(k) = \frac{\mathbf{x}_1(k)}{1+0.5\sin\big(\mathbf{x}_2(k)\big)}+\frac{\mathbf{x}_2(k)}{1+0.5\sin\big(\mathbf{x}_1(k)\big)}$$

where $(\mathbf{x}_1, \mathbf{x}_2)$ is the state and only the input u and the output y are accessible. The system is modeled in the input-output form $y(k+1) = f\big(y(k), y(k-1), y(k-2), y(k-3), u(k)\big)$. We

use an initial empty database which is updated all along the identification. The identification is performed for 1500 time steps with a test input $u(k) = \sin\left(\frac{2\pi k}{10}\right) + \sin\left(\frac{2\pi k}{25}\right)$. The plot in Fig. 6a shows the model and the system output in the last 200 points, while the plot in Fig. 6b shows the identification error. We obtain a good performance in modeling this complex system. These

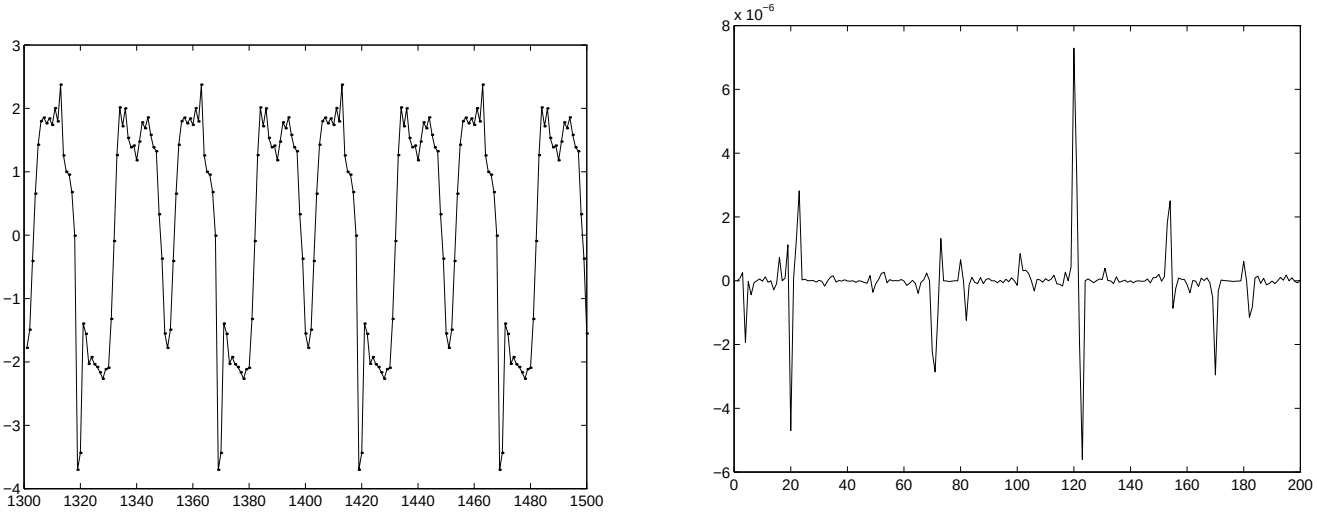

Fig. 6. Nonlinear system identification results: a) system (solid) and model (dotted) outputs b) identification error

results outperform those obtained by (Narendra & Li, 1996) with a much wider training set of 500,000 points and a complex architecture (4-layer feed-forward neural network).

4.2 The lazy gradient-based control of a nonlinear discrete-time system

In this simulation we consider the control of the plant described by the difference equations (27), by using the control algorithm described in Fig. 3. The system is represented in the minimum-phase input-output form $y(k+1) = f\big(y(k), y(k-1), y(k-2), y(k-3), u(k)\big)$. The reference output $y_{ref}(k)$ is given by

$$y_{ref}(k+1) = 0.75 \sin\left(\frac{2\pi(k+1)}{50}\right) + 0.75 \sin\left(\frac{2\pi(k+1)}{25}\right) \tag{28}$$

We use an initial empty database which is updated all along the identification. The system is controlled for 1300 time steps. The plot in Fig. 7a shows the model and the system output in the

last 300 points, while the plot in Fig. 7b shows the control action. These results outperform those

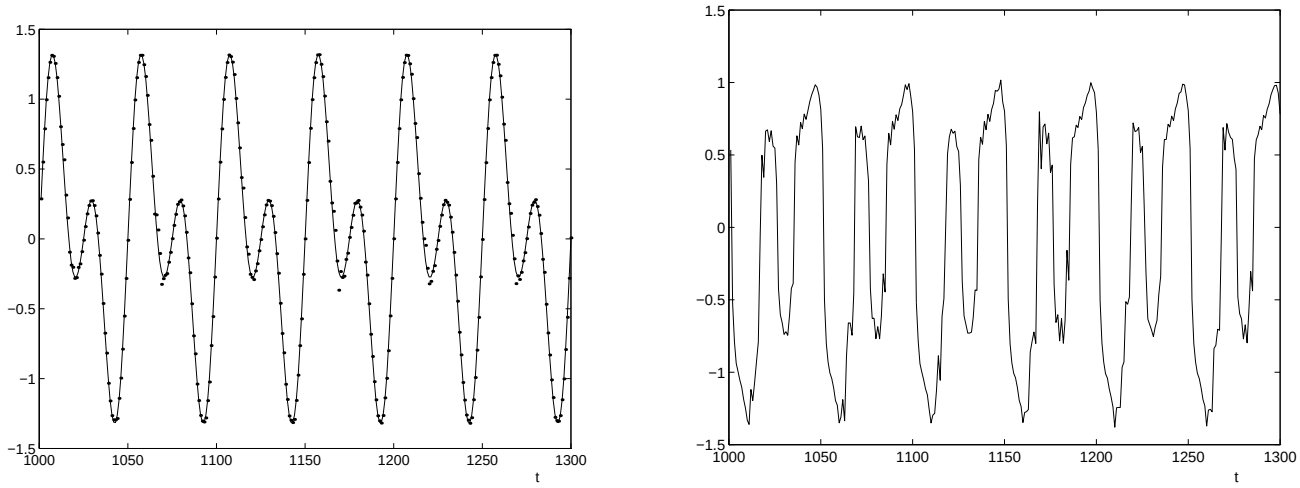

Fig. 7. Gradient-based system control: a) reference (solid) and system (dotted) outputs b) control action

obtained by (Narendra & Li, 1996) after 2,000,000 steps of on-line adjustments and a complex architecture (4-layer feed-forward neural network).

4.3 The lazy self-tuning control of a nonlinear discrete-time system

In this simulation we consider the control of the nonlinear SISO system described by the difference equation:

$$y(k+1) = \frac{y(k)y(k-1)y(k-2)\big(y(k-2)-1\big)u(k-1)+u(k)}{1+y^2(k-1)+y^2(k-2)} + \varepsilon. \tag{29}$$

The system is represented in the input-output form $y(k+1) = f\big(y(k), y(k-1), y(k-2), u(k), u(k-1)\big)$. The reference output $y_{ref}(k)$ is given by a periodic square wave. The *lazy* gradient-based algorithm is not able to control the system over this reference trajectory. On the contrary, a self-tuning regulator based on a pole placement algorithm is able to track the trajectory. We initialize the *lazy learning* database with a set of 5000 points collected by preliminarly exciting the system with a random uniform input. The database is then updated on-line each time a new input-output pairs is returned by the simulated system. The plot in Fig. 8a shows the reference and the system output when $\varepsilon = 0$, while the plot in Fig. 8b shows the effect of adding band-limited white noise

(peak-to-peak=0.35 and $\sigma_\varepsilon^2 = 0.1$) to the plant output and to the manipulated variable. In Fig. 9

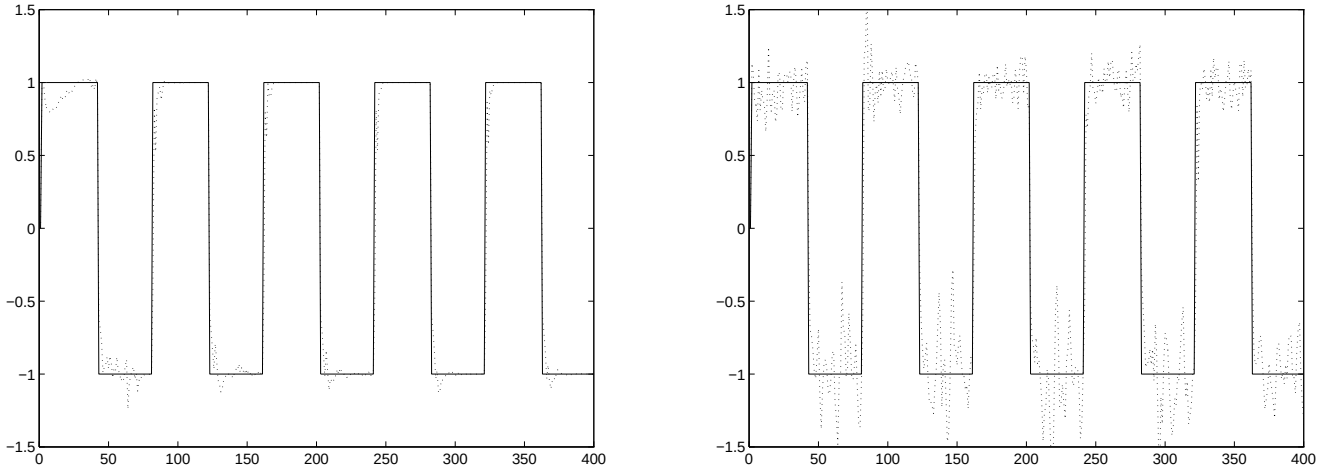

Fig. 8. Lazy self-tuning control: a) no noise: reference (solid) and system (dotted) outputs b) measurement and input noise: reference (solid) and system (dotted) output.

we plot the value of the zero of the open loop system identified by the *lazy learning* during the simulation with $\varepsilon = 0$. It is worthy noting how the system is non minimum-phase (i.e. absolute value of the zero greater than one) when the system variable y is in the neighborhood of $y = -1$ (e.g. see the time interval $50 - 100$). This is indeed the region where the gradient-based controller fails to control the system by making the feedback loop unstable.

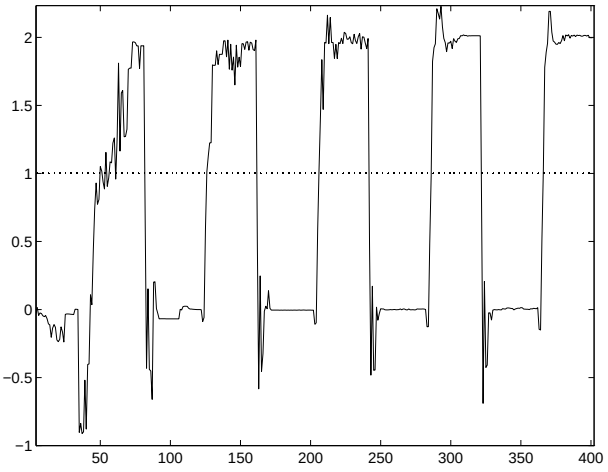

Fig. 9. Zero of the open loop identified system.

4.4 The lazy optimal control of the bioreactor

Consider, as third control example, the bioreactor system, a well-known benchmark in nonlinear control (MillerIII *et al.*, 1990), (Bersini & Gorrini, 1996). The bioreactor is a tank containing water, nutrients, and biological cells. Nutrients and cells are introduced into the tank where they mix. The state of this process is characterized by the number of cells (c_1) and the amount of nutrients (c_2). The equations of motion of the bioreactor are the following:

$$\begin{cases} \frac{dc_1}{dt} &= -c_1 u + c_1(1-c_2)\,\mathrm{e}^{\frac{c_2}{\gamma}} \\ \frac{dc_2}{dt} &= -c_2 u + c_1(1-c_2)\,\mathrm{e}^{\frac{c_2}{\gamma}} \left(\frac{1+\beta}{1+\beta-c_2}\right) \end{cases} \tag{30}$$

with $\beta = 0.02$ and $\gamma = 0.48$. In our experiment the goal was to stabilize the multi-variable system about the unstable state $(c_{ref1}, c_{ref2}) = (0.2107, 0.726)$ by performing a control action each 0.5 seconds.

We use the control algorithm described in Fig. 5. The horizon of the control algorithm is set to $t_f = 5$. The initial state conditions are set by the random initialization procedure defined in (MillerIII *et al.*, 1990). We initialize the *lazy learning* database with a set of 1000 points collected by preliminarly exciting the system with a random uniform input. The database is then updated on-line each time a new input-output pair is returned by the simulated system. The plot in Fig. 10a shows the output of the two controlled state variables, while the plot in Fig. 10b shows the control action. The bioreactor is considered as a challenging problem for its nonlinearity and because small changes in value of the parameters can cause the bioreactor to become unstable. These results show how using local techniques it is possible to control complex systems on a wide nonlinear range, with only a limited amount of examples and no a priori knowledge about the underlying dynamics.

5 Conclusions and future developments

Local memory-based techniques are powerful techniques for learning from a limited amount of data and for gaining insight on the local behavior of nonlinear systems. Furthermore, together with

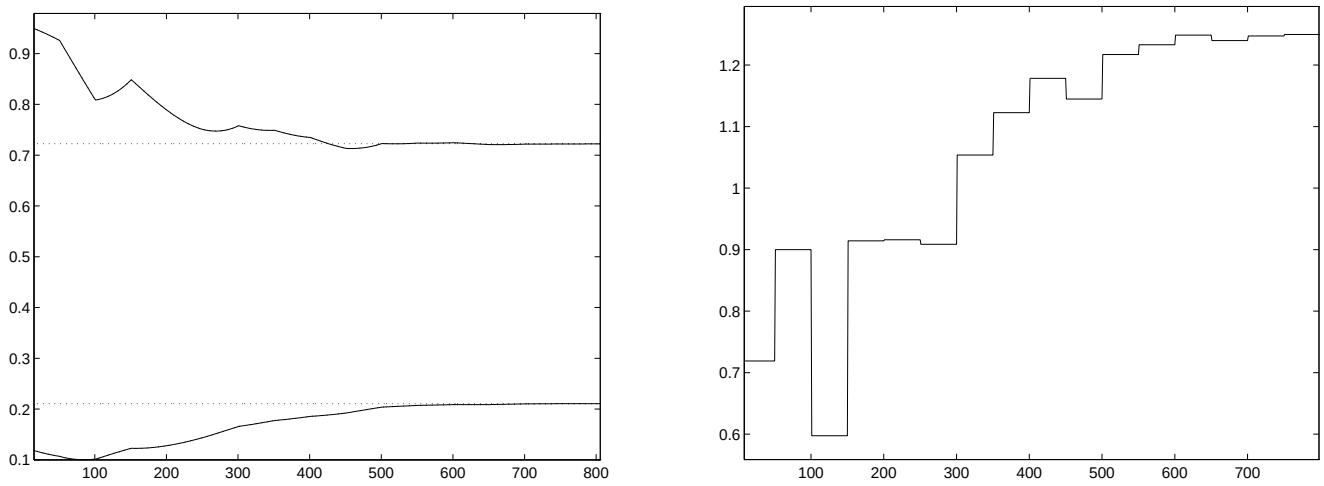

Fig. 10. Control results a) system outputs (solid) and references (dotted) b) control action (Each discrete-time step corresponds to 0.01 sec.)

the required prediction and/or parametric description, they return a statistical distribution of the uncertainty affecting this information. In modeling, this paper proposed an innovative algorithm to improve the performance of memory-based techniques which is based on a recursive version of the cross-validation and a statistical model selection. In control we illustrated and analyzed three control systems, which make an extensive use of local modeling. The *lazy* gradient-based control system, inspired by neural control, makes use of forward and inverse approximations of the system dynamics to select the control action. Like in neuro-control, properties of stability cannot be guaranteed in a general case. However, we showed how this approach obtains performances more accurate than neural networks even using a smaller set of training examples. The *lazy* self-tuning architecture adopts a linear control technique which eases the analysis in terms of stability properties and provides a useful insight in the dynamic properties of the nonlinear system. In addition, from a computational point of view, it is the less expensive among the proposed methods. Finally, the *lazy* optimal controller extends the local approaches to control strategies with extended time horizons. Moreover, the algorithm is well-suited for regulation of MIMO systems.

Future developments will mainly concern the problem of model uncertainty in control. To this aim, we will extend our analysis to robust and stochastic dual control. In fact, *lazy learning* is one of the few algorithms for nonlinear modeling which returns a statistical description of the uncertainty affecting the prediction and the model parameters with minimal additional computational cost.

There are still several concerns about how lazy learning can model real systems fast enough when the size of the database grows. This paper did not treat this issue but several researchers are exploring fast ways to find relevant data for local approximation using efficient software algorithms and special purpose hardware. The integration of these methods with the modeling and control techniques we presented can make of *lazy learning* one of the most promising tools for nonlinear control.

6 Appendix

6.1 The paired permutation test

Consider the null hypothesis H_0 that the vector of leave-one-out errors $\mathbf{E}_n^{\mathrm{cv}} = [e_n^{\mathrm{cv}}(i)]_{i=1}^{n}$ and the first n elements of the vector $\mathbf{E}_{n+1}^{\mathrm{cv}} = \left[e_{n+1}^{\mathrm{cv}}(i)\right]_{i=1}^{n+1}$, belong to the same distribution.

The paired permutation test assumes that, for each i, the observed values can be assigned to one of the two vectors $\mathbf{E}_n^{\mathrm{cv}}$ and $\mathbf{E}_{n+1}^{\mathrm{cv}}$ with the same probability. This means that, if H_0 is true, the difference $d_i = e_n^{\mathrm{cv}}(i) - e_{n+1}^{\mathrm{cv}}(i)$ between paired errors is to be just as likely negative as positive.

The null hypothesis H_0 is tested against some H_1 computing the value $D = \sum_{i=1}^{n} d_i$ and assuming that D is an instance of the random variable D^*. The sampling distribution of D^* is found by a randomization procedure (Cohen, 1995), a computer-intensive statistical method to derive the sampling distribution of a statistic by simulating the process of sample extraction. In the permutation test, this is done by creating a high number of pseudo-samples D^b, with $b = 1, \ldots, B$, derived from the actual sample D by substituting randomly a difference d_i with

$-d_i$. Once the sampling distribution of D^* is generated, a one-tailed test determines whether the null hypothesis has to be rejected.

If D is in the rejection region i.e. the right tail consisting of the most extreme values of D^*, the two leave-one-out vectors are assumed not to be extracted from the same distribution. Hence, the generalization error of the linear model using $n+1$ neighbors is assumed to be significantly greater than the one of the model fitted on n neighbors.

6.2 Generalized minimum variance design with offset term

Clarke and Gawthrop (1975) developed the Generalized Minimum-Variance Controller (GMVC) by introducing the reference signal and the control variable into the performance index

$$J = E\left[\left(P(z)y(k+d) + Q(z)u(k) - y_{ref}(k)\right)^2\right], \tag{31}$$

where $P(z) = \frac{P_N(z)}{P_D(z)}$ and $Q(z) = \frac{Q_N(z)}{Q_D(z)}$. Suppose that data are generated according to model (13). Multiplying both sides of (13) by P we obtain

$$\frac{P_N(z)}{P_D(z)}y(k) = \frac{P_N(z)}{P_D(z)}\left(\frac{B(z)}{A(z)}u(k-d) + \frac{b}{A(z)} + \frac{C}{A(z)}e(k)\right). \tag{32}$$

By setting $\tilde{y} = Py$, $\tilde{y}_{ref} = y_{ref} - Qu$, $\tilde{A} = P_D A$, $\tilde{B} = P_N B$, $\tilde{C} = P_N C$ and $\tilde{b} = P_N b$:

$$\tilde{A}(z)\tilde{y}(k) = z^{-d}\tilde{B}(z)u(k) + \tilde{C}e(k) + \tilde{b}. \tag{33}$$

Let the polynomial $E(z)$ and $\tilde{F}(z)$ be the solution of the Diophantine equation:

$$\tilde{C}(z) = \tilde{A}(z)E(z) + z^{-d}\tilde{F}(z). \tag{34}$$

Multiplying both sides of (33) by $z^d E(z)$ gives:

$$\tilde{A}(z)E(z)\tilde{y}(k+d) = \tilde{B}(z)E(z)u(k) + \tilde{C}e(k+d) + \tilde{b}E(z). \tag{35}$$

From (34) we have:

$$\tilde{C}(z)\tilde{y}(k+d) = \tilde{B}E(z)u(k) + \tilde{C}e(k+d) + \tilde{b}E(z) + \tilde{F}(z)\tilde{y}(k). \tag{36}$$

The optimal control law is then

$$\tilde{C}(z)\left(y_{ref} - \frac{Q_N}{Q_D}u(k)\right) = \tilde{B}E(z)u(k) + \tilde{b}E(z) + \tilde{F}(z)\tilde{y}(k), \tag{37}$$

that is equivalent to

$$P_D(z)Q_D(z)C(z)y_{ref} - Q_N(z)P_D(z)C(z)u(k) = \\ Q_D(z)\tilde{F}(z)y(k) + Q_D(z)P_D(z)B(z)E(z)u(k) + Q_D(z)P_D(z)bE \tag{38}$$

in the plant polynomials. The control law is then

$$u(k) = \frac{H(z)y_{ref} - F(z)y(k) - Q_D(z)P_D(z)bE(z)}{G(z)} \tag{39}$$

with $G = Q_N P_D C + Q_D P_D BE$ and $H = P_D Q_D C$, $F = Q_D \tilde{F}$. The result for the basic minimum variance controller can be obtained by setting $P_N = P_D = Q_D = 1$ and $Q_N = 0$.

6.3 Pole placement design with offset term

In the pole-placement formulation the desired closed-loop function is given by:

$$H_m(z) = \frac{B_m(z)}{A_m(z)} \tag{40}$$

The regulator has one output u and two inputs, the reference signal y_{ref}, and the measured output y. A general structure for the regulator may be represented by

$$u(k) = \frac{T(q)}{R(q)}y_{ref}(k) - \frac{S(q)}{R(q)}y(k) - G(q) \tag{41}$$

where R, T, G and S are polynomials in the forward-shift operator q. The input-output relationship for the closed-loop system is obtained by eliminating u between Equations (13) and (40). Hence:

$$y = \frac{B_d T}{AR + B_d S}y_{ref} + \frac{R(b - GB_d)}{AR + B_d S} + \frac{CR}{AR + B_d S}e. \tag{42}$$

with $B_d = z^{-d}B$. Requiring that this input-output relation is equivalent to (40) gives

$$\frac{B_d T}{AR + BS} = \frac{B_m}{A_m}, \tag{43}$$

$$b = GB_d. \tag{44}$$

The pole-placement design problem with offset term is then equivalent to the conventional one, once the additional requirement (44) is satisfied.

References

Aha D.W. 1989. Incremental, instance-based learning of independent and graded concept descriptions. *Pages 387–391 of: Sixth International Machine Learning Workshop*. San Mateo, CA: Morgan Kaufmann.

Astrom K.J. 1967. Computer Control of a Paper Machine. An application of linear stochastic control theory. *IBM J. Res. Dev.*, **11**, 389–404.

Astrom K.J. 1983. Theory and Applications of Adaptive Control - A Survey. *Automatica*, **19**(5), 471–486.

Astrom K.J. & Wittenmark B. 1990. *Computer-controlled Systems: Theory and Design*. Prentice-Hall International Editions.

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

Atkeson C.G. , Moore A.W. & Schaal S. 1997b. Locally weighted learning for control. *Artificial Intelligence Review*, **11**(1–5), 75–113.

Benedetti J.K. 1977. On the non parametric estimation of regression functions. *Journal of the Royal Statistical Society, Series B*, 248–253.

Bersini H. & Gorrini V. 1996. A Simplification of the Back-Propagation-Through-Time Algorithm for Optimal Neurocontrol. *IEEE Trans. on Neural Networks*, **8**(3), 437–441.

Bottou L. & Vapnik V.N. 1992. Local learning algorithms. *Neural Computation*, **4**(6), 888–900.

Carpenter G.A. & Grossberg S. 1988. The art of adaptive pattern recognition by a self-organising neural network. *IEEE Computer*, **21**(3), 77–88.

Clarke D.W. 1994. *Advances in Model-Based Predictive Control*. Oxford University Press.

Clarke D.W. & Gawthrop P.J. 1975. Self tuning controller. *Proceedings IEEE*, **122**(9), 929–934.

Cleveland W.S. 1979. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, **74**, 829–836.

Cohen P.R. 1995. *Empirical Methods for Artificial Intelligence.* Cambridge, Massachusetts: The MIT Press.

Cover T. & Hart P. 1967. Nearest neighbor pattern classification. *Proc. IEEE Trans. Inform. Theory*, 21–27.

Cybenko G. 1996. Just-in-Time Learning and Estimation. *Pages 423–434 of:* Bittanti S. & Picci G. (eds), *Identification, Adaptation, Learning. The Science of Learning Models from data.* NATO ASI Series. Springer.

Efron B. & Tibshirani R.J. 1993. *An Introduction to the Bootstrap.* New York: Chapman and Hall.

Epanechnikov V.A. 1969. Non parametric estimation of a multivariate probability density. *Theory of Probability and Its Applications*, 153–158.

Farmer J.D. & Sidorowich J.J. 1987. Predicting chaotic time series. *Physical Review Letters*, **8**(59), 845–848.

Fel'dbaum A.A. 1965. *Optimal control systems.* New York: Academic Press.

Goodwin G.C. & Sin K. S. 1984. *Adaptive Filtering Prediction and Control.* Prentice-Hall.

Grace Andrew . 1994. *Optimization Toolbox ForUse with MATLAB.* The MATHWORKS Inc.

Jacobs R.A. , Jordan M.I. & Barto A.G. 1991. Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. *Cognitive Science*, **15**(2).

Johansen T.A. & Foss B.A. 1993. Constructing NARMAX models using ARMAX models. *International Journal of Control*, **58**, 1125–1153.

Johansen T.A. & Foss B.A. 1995. Semi-Empirical Modeling of Nonlinear Dynamic Systems through Identification of Operating Regimes and Local Models. *Pages 105–126 of:* Hunt K.J. , Irwin G.R. & Warwick K. (eds), *Neural Network Engineering in dynamic control systems.* Springer.

Jordan M.I. & Rumelhart D.E. 1992. Forward Models: Supervised Learning with a Distal Teacher. *Cognitive Science*, **16**, 307–354.

Leontaritis I.J. & Billings S.A. 1985. Input-output parametric models for non-linear systems. *International Journal of Control*, **41**(2), 303–344.

MillerIII W.T. , Sutton R.S. & Werbos P.J. (eds) 1990. *Neural Networks for Control*. The MIT Press.

Moody J. & Darken C.J. 1989. Fast learning in networks of locally-tuned processing units. *Neural Computation*, **1**(2), 281–294.

Moore A.W. , Schneider J. & Deng K. 1997. Efficient Locally Weighted Polynomial Regression Predictions. *In: Proceedings 0f the 1997 International Machine Learning Conference*. Morgan Kaufmann Publishers. ftp at http://www.cs.cmu.edu/ awm/papers.html.

Murray-Smith R. & Hunt K. 1995. Local Model Architectures for Nonlinear Modelling and Control. *Pages 61–82 of:* Hunt K.J. , Irwin G.R. & Warwick K. (eds), *Neural Network Engineering in dynamic control systems*. Springer.

Murray-Smith R. & Johansen T.A. (eds) 1997. *Multiple Model Approaches to Modelling and Control*. Taylor and Francis.

Myers R.H. 1990. *Classical and Modern Regression with Applications*. Boston, MA: PWS-KENT.

Narendra K.S. & Annaswamy A.M 1989. *Stable Adaptive Systems*. Englewood Cliffs, NJ: Prentice Hall.

Narendra K.S. & Balakrishnan J. 1997. Adaptive Control Using Multiple Models. *IEEE Transactions on Automatic Control*, **42**(2), 171–187.

Narendra K.S. & Li S.M. 1996. Neural Networks in Control Systems. *Chap. 11, pages 347–394 of:* Paul Smolensky M.C. Mozer & Rumelhart D.E. (eds), *Mathematical Perspectives on Neural Networks*. Lawrence Erlbaum Associates.

Nene S. A. & Nayar S.K. 1997. A Simple Algorithm for Nearest Neighbor Search in High Dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **19**(9), 989–1003.

Passino K.M. & Yurkovich S. 1996. Fuzzy Control. *Pages 1001–1017 of:* Levine W. S. (ed), *The Control Handbook*. IEEE Press.

Priestley M.B. 1988. *Non-linear and Non-stationary time series analysis*. Academic Press.

Rugh W.J. 1991. Analytical framework for gain scheduling. *IEEE Control Systems*, **11**(1), 79–84.

Salganicoff M. 1997. Tolerating concept and sampling shift in lazy learning using prediction error context switching. *Artificial Intelligence Review*, **11**(1–5), 133–155.

Schaal S. & Atkeson C. G. 1994. Robot Juggling: Implementation of Memory-Based Learning. *IEEE Control Systems*, February, 57–71.

Schott K.D. & Bequette B.W. 1997. Multiple Model Adaptive Control. *Pages 269–291 of:* Murray-Smith R. & Johansen T.A. (eds), *Multiple Model Approaches to Modeling and Control*. Taylor and Francis.

Shamma J.S. & Athans M. 1992. Gain scheduling: Potential hazards and possible remedies. *IEEE Control Systems Magazine*, June, 101–107.

Siegel S. & N.J. Castellan Jr. 1988. *Non Parametric Statistics for the Behavioral Sciences*. 2nd edn. McGraw-Hill International.

Slotine J.J. & Li W. 1991. *Applied Nonlinear Control*. Prentice-Hall International.

Stengel Robert F. 1986. *Stochastic optimal control: theory and application*. New York: John Wiley and Sons.

Stenman A. , Gustafsson F. & Ljung L. . 1996. Just in time models for dynamical systems. *In: 35th IEEE Conference on Decision and Control*.

Takagy T. & Sugeno M. 1985. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on System, Man and Cybernetics*, **15**(1), 116–132.

Tanaka K. 1995. Stability and Stabilizability of Fuzzy-Neural-Linear Control Systems. *IEEE Transactions on Fuzzy Systems*, **3**(4), 438–447.

Tanaka K. & Sugeno M. 1992. Stability analysis and design of fuzzy control systems. *Fuzzy Sets and Systems*, **45**, 135–156.

Tolle H. , Parks P.C. , Hormel E. Ersuand M. & Militzer J. 1992. Learning control with interpolating memories - general ideas, design-lay-out, theoretical approaches and practical applications. *International Journal of Control*, **56**(2), 291–317.