INT. J. CONTROL, 1999, VOL. 72, NO. 7/8, 643–658

Lazy learning for local modelling and control design

GIANLUCA BONTEMPI†*, MAURO BIRATTARI† and HUGUES BERSINI†

This paper presents local methods for modelling and control of discrete-time unknown non-linear dynamical systems, when only input–output data are available. We propose the adoption of *lazy learning*, a memory-based technique for local modelling. The modelling 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 by 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 non-linear systems about regimes which are far from the equilibrium points. Simulation examples of identification and control of non-linear systems starting from observed data are given.

1. Introduction

The problem of modelling a process from observed data has been the object of several disciplines from non-linear 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 modelling builds a single functional model of the dataset. This has traditionally been the approach taken in neural network modelling and in other forms of non-linear 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 precessing of the dataset until a prediction or a local model are required. The classical nearest neighbour method is the original approach to local modelling. 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 neighbourhood of the query point. Local techniques are an old idea in time series prediction (Farmer and Sidorowich 1987), classification (Cover and 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 and Vapnik 1992). Recent work on *lazy learning*, also known as just-in-time learning, gave a new impetus to the adoption of local techniques for modelling (Stenman *et al.* 1996, Atkeson *et al.* 1997 a) and control problem (Schaal and Atkeson 1994, Atkeson *et al.* 1997 b, Rhodes *et al.* 1997). For an up-to-date review on lazy learning methods see the special issue of *Artificial Intelligence Review* (Aha 1997).

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 and 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, through cross-validation (Stoica *et al.* 1986), the best local model among a set of different candidates. In classical local modelling, an amount of options had to be designed by the data analyst according to heuristic criteria and *a priori* assumptions. More recently, several authors have considered automatic selection techniques of the parameters of local polynomial models (Fan and Gijbels 1995, Ruppert *et al.* 1995). However, these works are based on a separate estimation of the bias and variance components of the mean square approximation error, making the resulting procedure rather complicated. 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. To this aim, 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

† Iridia–CP 194/6, Université Libre de Bruxelles, 50, av. Franklin Roosevelt, 1050, Bruxelles, Belgium.
* Author for correspondence. e-mail: gbonte@ulb.ac.be

International Journal of Control ISSN 0020–7179 print/ISSN 1366–5820 online © 1999 Taylor & Francis Ltd
http://www.tandf.co.uk/JNLS/con.htm
http://www.taylorandfrancis.com/JNLS/con.htm

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 non-linear control design techniques which extensively use analysis and design tools from linear control. The idea of employing linear techniques in a non-linear 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 and Johansen 1997). Gain scheduling (Shamma and Athans 1992), fuzzy inference systems (Takagi and Sugeno 1985) and local model networks (Johansen and Foss 1993), are well-known examples of control techniques for non-linear systems based on linear control. 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 a gradient-based controller which exploits the nice properties of the *lazy learning* algorithm as a non-linear 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.* (1977 b) but was tested simply on a static control task. Here we analyse its dynamic stability properties and we compare it with the other proposed local control techniques.

The second controller is derived from the self-tuning regulator (STR) architecture (Åström 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 neighbourhood 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 modelling is not new in literature (Tanaka 1995, Passino and Yurkovich 1996, Atkeson *et al.* 1977 b). 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 neighbourhood 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 non-linearity 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 optimization 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 (Moore *et al.* 1997, Nene and Nayar 1997).

The remainder of the paper is organized as follows. In §2 we will introduce our modelling technique based on an iterative selection procedure. In §2.1 we present the recursive algorithm for PRESS statistic computation. In §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 §3. Finally, in §4 simulation examples of identification and of control are given.

2. Local modelling as an optimization problem.

Modelling from data involves integrating human insight with learning techniques. In many real cases the analyst faces a situation where a 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 $f : \Re^m \rightarrow \Re$ of which we are given a set of N samples $\{(\boldsymbol{\varphi}_1, y_1), (\boldsymbol{\varphi}_2, y_2), \ldots, (\boldsymbol{\varphi}_N, y_N)\}$. 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 φ_q, the prediction of the value $y_q = f(\varphi_q)$ is computed as follows. First, for each sample (φ_i, y_i) a weight w_i is computed as a function of the distance $d(\varphi_i, \varphi_q)$ from the query point φ_q to the point φ_i. Each row of $\mathbf{\Phi}$ and $\mathbf{y}$ is then multiplied by the corresponding weight creating the variables $\mathbf{Z} = \mathbf{W}\mathbf{\Phi}$ 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 $f(\varphi_q)$ is obtained evaluating such a model in the query point

$$\hat{y} = \varphi_q^T(\mathbf{Z}^T\mathbf{Z})^{-1}\mathbf{Z}^T\mathbf{v} \tag{1}$$

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 modelling technique. Typically, the data analyst who adopts a local regression approach, has to take a set of decisions related to the model as, for example, the number of neighbours, the weight function, the parametric family, and 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 and 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 modelling 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 can be exploited to select the training subset among the neighbours, as well as various structural aspects like the features to treat and the degree of the polynomial used as a local approximator (Bersini *et al.* 1998). 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 and Tibshirani 1993), which returns a reliable estimation of the prediction error in φ_q. We define the ith *leave-one-out* error $e^{cv}(i)$ as the difference between y_i and the prediction given by the local model centred in φ_q and fitted using all the examples available but the ith. An estimation of the prediction error in φ_q is given by the average of the errors $e^{cv}(i)$ each weighted according to the distance $d(\varphi_i, \varphi_q)$. When considering a 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.* 1997 a)

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

In our modelling 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 of linearity* surrounding φ_q, defined as the region 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 itself. The task of identifying the region of linearity is therefore akin to the task of finding, among the examples avail-

able, the number n of neighbours of φ_q to be used in the local regression fit. In terms of generalization properties, the choice of this parameter is related to the problem of finding the proper bias/variance trade-off (Geman *et al.* 1992). Adding points reduces the variances and increases the bias; on the other hand reducing the number of neighbours makes the bias smaller at the cost of a greater variance of the estimator. 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 computation of the parameters and of the PRESS statistic, we adopt an incremental approach based on recursive linear techniques. Recursive algorithms have been developed in model identification and adaptive control literature (Goodwin and 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 neighbours by updating the parameters of the model with n examples. Furthermore, the *leave-one-out* errors $e^{cv}(i)$ are obtained exploiting partial results from the least squares 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 others, the recursive *lazy learning* algorithm is described as

$$\left.\begin{aligned} \mathbf{P}_{n+1} &= \mathbf{P}_n - \frac{\mathbf{P}_n \varphi_{n+1} \varphi_{n+1}^{\mathrm{T}} \mathbf{P}_n}{1 + \varphi_{n+1}^{\mathrm{T}} \mathbf{P}_n \varphi_{n+1}} \\ \gamma_{n+1} &= \mathbf{P}_{n+1} \varphi_{n+1} \\ e_{n+1} &= y_{n+1} - \varphi_{n+1}^{\mathrm{T}} \boldsymbol{\beta}_n \\ \boldsymbol{\beta}_{n+1} &= \boldsymbol{\beta}_n + \gamma_{n+1} e_{n+1} \end{aligned}\right\} \qquad (3)$$

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

where (φ_{n+1}, y_{n+1}) is the $n+1$th nearest neighbour of the query point, $\mathbf{P}_n$ is the recursive approximation of the matrix $(\mathbf{Z}^{\mathrm{T}}\mathbf{Z})^{-1}$, $\boldsymbol{\beta}_n$ denotes the optimal least squares parameters of the model fitted on the n nearest neighbour, and $e_n^{cv}(i)$, with $1 \leq i \leq n$, is the vector $\mathbf{E}_n^{cv}$ of *leave-one-out* errors. Once this vector is available, equation (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 mode defined by n neighbours. However, the problem of assessing the right dimension of the linearity region using a number of samples affected by noise requires a powerful statistical procedure. In the following section, we will discuss in detail the method we adopt.

1. $n = m$: initialize the parameter of the local model (e.g. conventional initialization: $\boldsymbol{\beta}_0 = 0$, $\mathbf{P}_0 = \lambda \mathbf{I}$ with large λ).
2. Add the example (φ_{n+1}, y_{n+1}). The parameter $\boldsymbol{\beta}$ is updated and the vector $\mathbf{E}_n^{cv}$ 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 go to 4, else accept the model and go to 2 to consider adding further examples.
4. Stop the recursive identification procedure and return the parameters of the last model accepted.

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

2.2. *The statistical test for model selection*

The recursive method described in the previous section returns for each size n of the neighbourhood a vector $\mathbf{E}_n^{cv}$ of *leave-one-out* errors. In order to select the best model, our procedure, described in detail in figure 1, consists in increasing the number of neighbours considered when identifying the local model, until the model performance deteriorates. 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 $\mathbf{E}_n^{cv}$ and $\mathbf{E}_{n+1}^{cv}$ belong to the same distribution. To evaluate this hypothesis we use a non-parametric permutation test (Siegel and 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 A1) because of the correlation between the two error vectors.

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

3. Lazy learning for control design

To illustrate the different approaches to *lazy learning* control design, we will first define some notation 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

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

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 input–output time delay and $f(\cdot)$ is some non-linear function. This model is known as the NARMAX model (Leontaritis and Billings 1985). Let us assume we have no physical description of the function $f(\cdot)$ but an amount of pairs $[u(k), y(k)]$ is available. Defining the information vector as

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

system (4) can be written in the form

$$y(k) = f[\boldsymbol{\varphi}(k-1)] + e(k) \quad (6)$$

It is demonstrated that models (4) can describe the input–output behaviour of general state-space non-linear dynamic systems

$$\mathbf{x}(k+1) = g[\mathbf{x}(k), u(k)] + \mathbf{v}(k)$$
$$y(k) = h[\mathbf{x}(k)] + w(k) \quad (7)$$

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 $g : \Re^{p+1} \to \Re^p$, $h : \Re^p \to \Re$ are some non-linear functions.

3.1. *Lazy learning and local linear control: a comparative analysis*

Although non-linearity characterizes most real control problems, methods for analysis and control design are considerably more powerful and theoretically better founded for linear systems than for non-linear ones. In non-linear control literature we therefore find many approaches based on the extension of linear techniques to non-linear 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 equation (6) by performing a multi-variable Taylor series expansion. The outcome is a linear time invariant system that describes locally the non-linear dynamics. Under some conditions (Slotine and Li 1991) this linear model provides information 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 of inaccurate modelling 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 behaviour of the system near a prescribed trajectory. Let $\boldsymbol{\varphi}^*(k)$ denote a specific trajectory of the non-linear system (4), that is $y^*(k) = f[\boldsymbol{\varphi}^*(k-1)]$ with $\boldsymbol{\varphi}^*(k)$ an information vector at time k. Assuming that $f(\cdot)$ is continuously differentiable, 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, a condition that unfortunately is not always satisfied.

Gain scheduling: It is one of the most widely and successfully applied techniques for the design of non-linear 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 and 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 an operating point by introducing the notion of an *operating regime*. An operating regime is a set of operating points where the system behaves approximately linearly (Johansen and Foss 1993, 1995). To each of them a validity region and a local description of the system behaviour 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 and Hunt 1995). One major advantage of this approach is the possibility to integrate *a priori* knowledge with parametric learning procedures. Related non-linear modelling approaches are Takagi and Sugeno (1985) fuzzy inference systems and radial basis functions (Moody and 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 and 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 and 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 neighbourhood 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 non-linear and time-varying configurations by giving more weight to the most recent data. On the contrary, the *lazy* approach treats separately non-linear 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 protects them from the problem of *data interference* (Jacobs *et al.* 1991, Salganicoff 1997)—also known as the *stability-plasticity dilemma* (Carpenter and Grossberg 1988). Consider, as an illustrative example, the simple non-linear dynamics $y(k) = f[u(k-1)]$ of which we plot six samples in figure 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 example no. 6 is encountered, the estimated model (dotted line) has lost memory of the dynamics

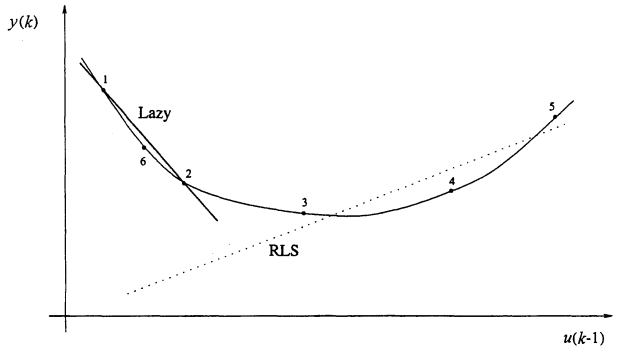

Figure 2. Recursive vs. lazy learning identification.

existing in the neighbourhood of points 1 and 2. As a result, the accuracy of the RLS approximation (dotted line) in 6 is poor due to its limited tracking speed. 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 neighbours.

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 appropriate 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 into simpler local problems. The main differences concern the model identification procedure. Local model networks aim to estimate a functional description to cover the whole system operating domain, whereas memory based techniques focus simply on a value estimation about the current operating points. Local model networks are more time consuming in identification but faster in prediction. However, when new data are observed, model update may require us to perform the whole LMN modelling process from the beginning. On this matter *lazy learning* takes advantage of 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 the middle ground between LMN and multiple model control. As 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. As 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 non-linear mappings using observed 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 step a reference value y_{ref}. The *lazy* model (1) can be used to predict the response of the system to the action u^i

$$\hat{y}_{u^i}(k) = \hat{f}\,[y(k-1),\ldots,y(k-ny),u^i,u(k-2),\ldots, u(k-nu),e(k-1),\ldots,e(k-ne)] \quad (8)$$

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

$$u^{\text{opt}} = \arg\min_{u^i} J(u^i) = \arg\min_{u^i}[y_{\text{ref}} - \hat{y}_{u^i}(k)]^2 \quad (9)$$

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 and Rumelhart 1992)

$$u^0 = \hat{f}_{\text{inv}}[y_{\text{ref}},y(k-1),\ldots,y(k-ny),u(k-2),\ldots, u(k-nu),e(k-1),\ldots,e(k-ne)] \quad (10)$$

A detailed version of the *lazy* gradient-based control algorithm is presented in figure 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 non-linear case. In the method discussed above, *lazy learning* is used as a black-box approximator of the input–output representation of a non-linear dynamical system. The result is a non-linear controller for which the stability of the closed loop feedback system cannot be guaranteed theoretically for a generic non-linear plant.

1. Initialization of the algorithm with the value u^0 provided by the inverse mapping (10).
2. Prediction of the outcome $\hat{y}_{u^i}$ of the system forced by the input u^i.
3. Computation of the gradient vector $(\mathrm{d}J(u^i))/\mathrm{d}u^i$.
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 ($u^i = u^{i+1}$) go to 6 else go to 2.
6. Control action execution. The action $u(k-1) = u^{\text{opt}}$ 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.

Figure 3. The lazy gradient-based controller algorithm.

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 from growing without limit, making the closed loop system unstable. This problem is demonstrated by the simulation examples in §4.2 and §4.3. While the technique can successfully control a complex minimum-phase non-linear 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 non-linear 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 non-linear context.

3.3. *The lazy learning self-tuning controller*

In this section we propose a hybrid architecture for the indirect control of non-linear discrete-time plants from their observed input–output behaviour. An indirect control scheme (Narendra and Annaswamy 1989, Åström and Wittenmark 1990) combines a parameter estimator, which computes an estimate $\hat{\vartheta}$ of the unknown parameters, with a control law $u(k) = K[\varphi(k),\vartheta^*]$ implemented as a function of the plant parameters. In the adaptive version, the estimator generates the estimate $\hat{\vartheta}(k)$ at each sampling period k by processing the observed input–output behaviour. Following the certainty equivalence paradigm, this estimate is assumed to be a specification of the real plant and is then used to compute the control law $u(k) =$

$K[\varphi(k), \vartheta(k)]$. 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 § 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 (Åström and Wittenmark 1990).

The MV control algorithm was first formulated in Åström (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

$$A(z)y(k) = z^{-d}B(z)u(k) + C(z)e(k) \qquad (11)$$

and suppose we want to regulate it to $y_{\text{ref}} = 0$. 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 first requires us 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 of equation (11). However, in our approach the linearization is also performed in configurations which are far from the equilibrium locus. As a consequence, the relationship between the input u and the output y is given by

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

where b is an offset term. This requires a slight modification to the formulas for GMV (see Appendix A2) and PP controller design (see Appendix A3 and Hunt and Johansen 1997).

The proposed control algorithm is described in detail in figure 4. Note that the selection of neighbours is made considering only the subset vector

$$\varphi_S(k-1) = [y(k-1), \ldots, y(k-ny), u(k-2), \ldots, u(k-nu), e(k-1), \ldots, e(k-ne)] \qquad (13)$$

1. Acquisition of the vector (13), and selection of neighbours.
2. Linearization of the function $f(\cdot)$ using the *lazy learning* algorithm.
3. Derivation of the polynomials A, B, C and the offset b of (12) from the linearized model.
4. Design of a MVG/PP controller for (12) which satisfies the required properties (stability, accuracy, speed, ...) of the closed loop behaviour.
5. Computation of the control signal. For details see Appendix A2 and A3.
6. Updating of the database by storing the new input–output observation.

Repeat these steps at each sampling period.

Figure 4. The lazy self-tuning controller algorithm.

of the information vector (5). In fact $u(k-1)$ is not 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).

3.3.1. *Lazy self-tuning control analysis.* In the *lazy* self-tuning regulator the non-linear plant (6) is parameterized as a linear system where parameters are changing with the observable state. This means that the non-linear 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 and Athans (1992) in their analytical analysis of gain scheduling controllers, or the state-dependent models presented by Priestley (1988). These models can be represented as

$$A[\varphi(k)]y(k) = z^{-d}B[\varphi(k)]u(k) + C[\varphi(k)]e(k) \qquad (14)$$

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

$$\mathbf{x}(k+1) = \mathbf{F}[\varphi(k)]\mathbf{x}(k) + \mathbf{v}(k) \qquad (15)$$

This representation allows us to analyse the stability of our controller. If the regulator is designed so that the eigenvalues of $\mathbf{F}[\varphi(k)]$ are stable, then the system (15) will be asymptotically stable for any fixed value of φ (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 $\mathbf{F}[\varphi(k)]$ a sufficient condition for uniform stability (Tanaka and Sugeno, 1992) then a common matrix $\mathbf{P} > 0$ exists such that

$$\mathbf{F}[\varphi(k)]^{\mathrm{T}}\mathbf{P}\mathbf{F}[\varphi(k)] - \mathbf{P} < 0 \qquad \text{for all } k \qquad (16)$$

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 equation (16). Then under the 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 non-linear 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 non-linear 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

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

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 non-linear configuration, optimal control theory (Stengel 1986) provides the solution for the linear case. In the following, we will present the non-linear problem in a linear time varying setting.

Consider the trajectory of the dynamical system once forced by an input sequence $U = [\mathbf{u}(0), \mathbf{u}(1), \ldots, \mathbf{u}(t_f - 1)]$. 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 behaviour of the linear system along a generic trajectory is the behaviour of a linear time varying system whose state equations can be written in the form

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

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

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

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

$$\begin{aligned}\mathbf{P}_k = \mathbf{Q}_k + \mathbf{F}_k^{\mathrm{T}}\mathbf{P}_{k+1}\mathbf{F}_k - (\mathbf{M}_k + \mathbf{F}_k^{\mathrm{T}}\mathbf{P}_{k+1}\mathbf{G}_k)\\ \times (\mathbf{R}_k + \mathbf{G}_k^{\mathrm{T}}\mathbf{P}_{k+1}\mathbf{G}_k)^{-1}(\mathbf{M}_k^{\mathrm{T}} + \mathbf{G}_k^{\mathrm{T}}\mathbf{P}_{k+1}\mathbf{F}_k) \quad (20)\end{aligned}$$

having as final condition

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

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.

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

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

$$\lambda_f^{\mathrm{T}} = \mathbf{x}_f^{\mathrm{T}}\mathbf{P}_f \quad (24)$$

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

$$H_k = J + \lambda_{k+1}^{\mathrm{T}}[\mathbf{F}_k\mathbf{x}(k) + \mathbf{G}_k u(k)] \quad (25)$$

The Euler–Lagrange equations do not hold for non-linear 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 §2, our modelling procedure performs system linearization with minimum effort, no *a priori* knowledge and only observed data. Hence, we propose an algorithm for non-linear 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

$$U^{\mathrm{opt}} = \arg\min_{U^i} J(U^i) \quad (26)$$

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 figure 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 the simulated sequence;
- updating of the sequence with a gradient based algorithm.

Once the search algorithm has returned to U^{opt}, the first action of the sequence is applied to the real system—*receding horizon* control strategy (Clarke 1994). Let us remark that the *lazy learning* model has a twofold role in the algorithm in figure 5: (i) at step 2 it is an estimator which predicts the behaviour 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.* (1997 b), Tanaka (1995), and Passino and Yurkovich (1996), applied infinite-time LQR regulator to non-linear 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 neighbourhood of the linearization point. Sjöberg and Agarwal (1997) proposed a control algorithm where a linear time-varying description is used for non-linear control problems. However, their approach requires an initial controller that stabilizes the plant.

The advantage of the lazy approach is that the above requirements do not need to be satisfied. First, *lazy learning* is able to 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 delay of the system. Finally, this formalism can take 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 behaviour (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 modelling procedure can return at no additional cost a statistical description of the estimated parameters (see §2.2). Future work will focus on the extension of this technique to the stochastic control case.

1. Initialization: for $k = 0$, U^0 is set to a random sequence; for $k > 0$, U^0 is set equal to U^{opt} at the previous time step, with the first component removed.
2. Forward simulation of the system forced by the sequence
$$U^i = [\mathbf{u}^i(k), \mathbf{u}^i(k+1), \ldots, \mathbf{u}^i(k+t_f-1)]$$
where $\mathbf{u}^i(j)$ denotes the action applied to the simulated system at time j. The system behaviour is predicted using the model identified by the local learning method.
3. Formulation of the non-linear 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
$$\frac{\partial J}{\partial U^i} = \left[\frac{\partial J}{\partial \mathbf{u}^i(k)}, \frac{\partial J}{\partial \mathbf{u}^i(k+1)}, \cdots, \frac{\partial J}{\partial \mathbf{u}^i(k+t_f-1)}\right]$$
by using formula (22).
7. 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).
8. If the minimum has been reached ($U^i = U^{i+1}$) goto 9 else goto 2.
9. Control action execution. The first action $\mathbf{u}^{\text{opt}}(k)$ of the sequence U^{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

Figure 5. The lazy learning optimal controller algorithm.

4. Simulation studies

4.1. *The identification of a non-linear discrete-time system*

Our approach has been applied to the identification of a complex non-linear benchmark proposed by

Narendra and Li (1996). The discrete-time equations of the system are

$$\left.\begin{aligned} \mathbf{x}_1(k+1) &= \left(\frac{\mathbf{x}_1(k)}{1+\mathbf{x}_1^2(k)}+1\right)\sin[\mathbf{x}_2(k)] \\ \mathbf{x}_2(k+1) &= \mathbf{x}_2(k)\cos[\mathbf{x}_2(k)] \\ &\quad + \mathbf{x}_1(k)\exp\left(-\frac{\mathbf{x}_1^2(k)+\mathbf{x}_2^2(k)}{8}\right) \\ &\quad + \frac{u^3(k)}{1+u^2(k)+0.5\cos[\mathbf{x}_1(k)+\mathbf{x}_2(k)]} \end{aligned}\right\} \quad (27)$$

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

where $(\mathbf{x}_1, \mathbf{x}_2)$ is the state and only the input u and the output y are accessible. The system is modelled in the input–output form $y(k+1) = f[y(k), y(k-1), y(k-2), y(k-3), u(k)]$. 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 $u(k) = \sin(2\pi k/10) + \sin(2\pi k/25)$. The plot in figure 6(*a*) shows the model and the system output in the last 200 points, while the plot in figure 6(*b*) shows the identification error. We obtain a good performance in modelling this complex system. These results outperform those obtained by Narendra and Li (1996) with a much wider training set of 500 000 points and a complex architecture (four-layer feed-forward neural network).

4.2. *The lazy gradient-based control of a non-linear discrete-time system*

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

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

We use an initial empty database which is updated all along the identification. The system is controlled for 1300 time steps. The plot in figure 7(*a*) shows the model and the system output in the last 300 points, while the plot in figure 7(*b*) shows the control action. These results outperform those obtained by Narendra and Li (1996) after 2 000 000 steps of on-line adjustments and a complex architecture (four-layer feed-forward neural network).

4.3. *The lazy self-tuning control of a non-linear discrete-time system*

In this simulation we consider the control of the non-linear SISO system described by the difference equation

$$y(k+1) = \frac{y(k)y(k-1)y(k-2)[y(k-2)-1]u(k-1)+u(k)}{1+y^2(k-1)+y^2(k-2)} + \varepsilon \quad (29)$$

The system is represented in the input–output form $y(k+1) = f[y(k), y(k-1), y(k-2), u(k), u(k-1)]$. The reference output $y_{\text{ref}}(k)$ 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 preliminarily exciting the system with a ran-

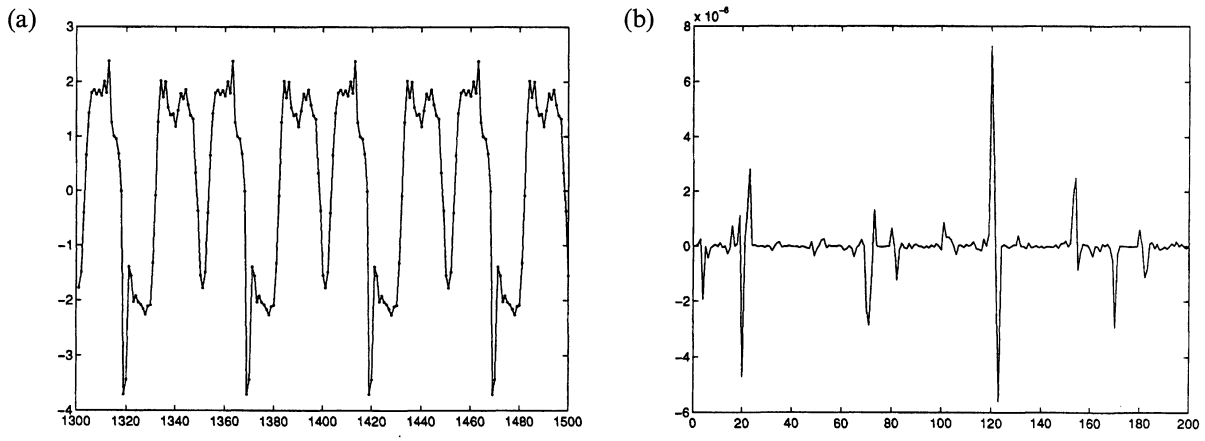

Figure 6. Non-linear system identification results: (a) system (solid) and model (dotted) outputs; (b) identification error.

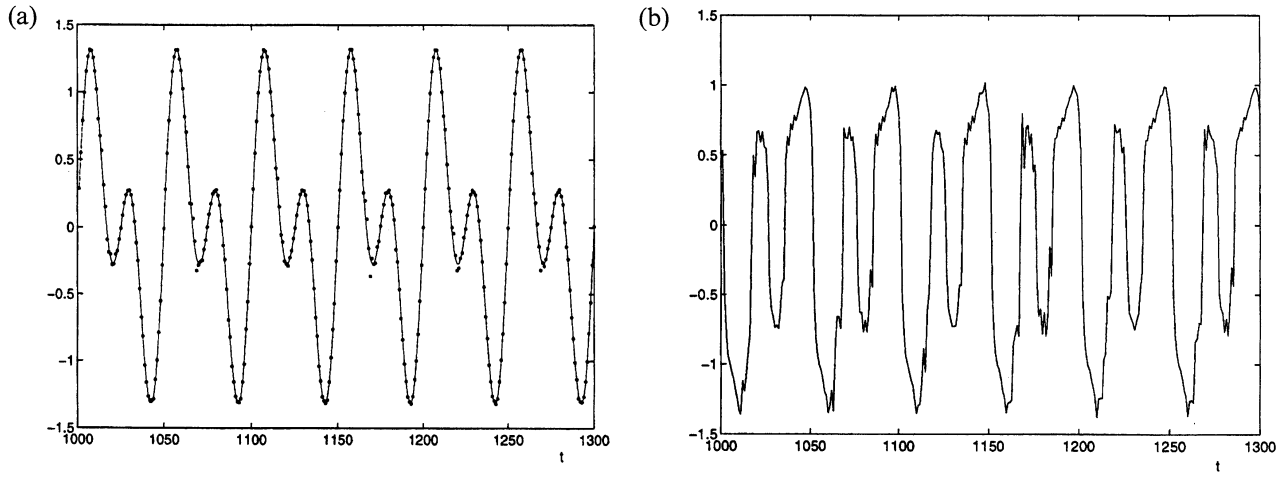

Figure 7. Gradient-based system control: (a) reference (solid) and system (dotted) outputs; (b) control action.

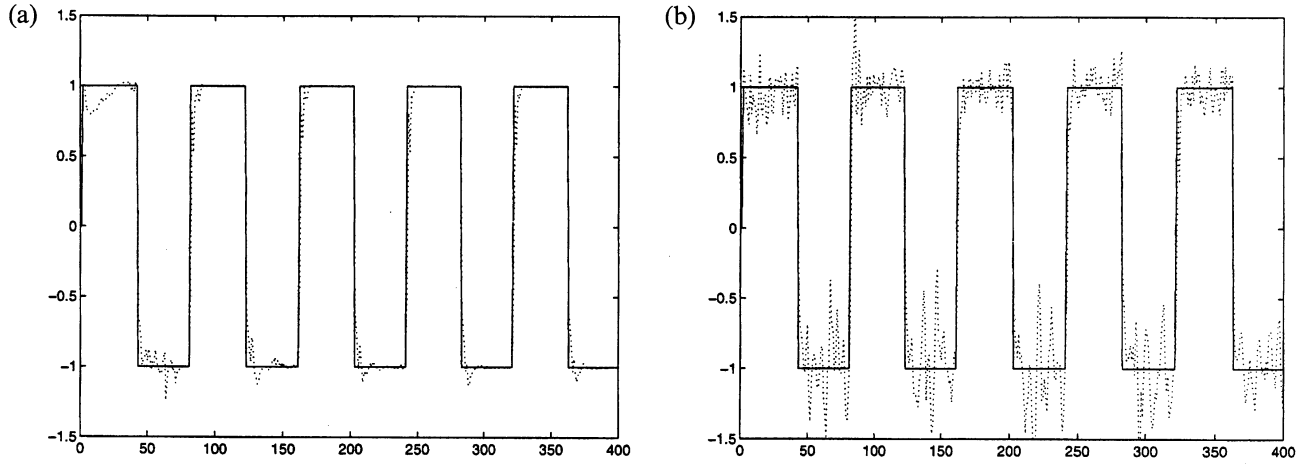

Figure 8. Lazy self-tuning control: (a) no noise: reference (solid) and system (dotted) outputs; (b) measurement and input noise: reference (solid) and system (dotted) output.

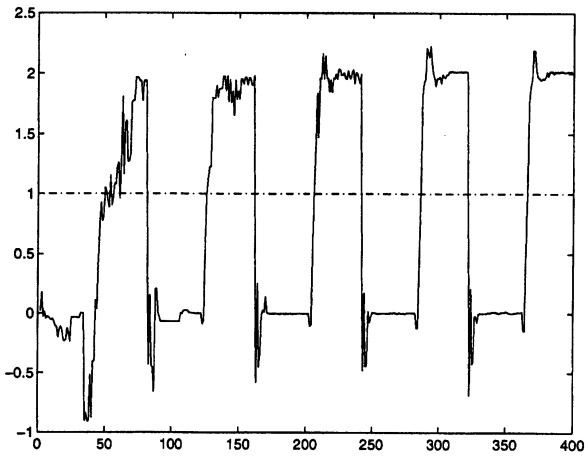

Figure 9. Zero of the open loop identified system.

dom uniform input. The database is then updated online each time a new input–output pair is returned by the simulated system. The plot in figure 8(a) shows the reference and the system output when $\varepsilon = 0$, while the plot in figure 8(b) 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 input variable. In figure 9 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 worth noticing 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 neighbourhood 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.

4.4. *The lazy optimal control of the bioreactor*

Consider, as third control example, the bioreactor system, a well-known benchmark in non-linear control (Miller *et al.* 1990, Bersini and Gorrini 1997). 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

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

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 s.

We use the control algorithm described in figure 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 Miller *et al.* (1990). We initialize the *lazy learning* database with a set of 1000 points collected by preliminarily exciting the system with a uniformly distributed random input. The database is then updated on-line each time a new input–output pair is returned by the simulated system. The plot in figure 10(*a*) shows the output of the two controlled state variables, while the plot in figure 10(*b*) shows the control action. The bioreactor is considered as a challenging problem for its non-linearity and because small changes in the value of the parameters can cause the bioreactor to become unstable. These results show that using local techniques it is possible to control complex systems on a wide non-linear range, with only a set of examples and no *a priori* knowledge about the underlying dynamics.

5. Conclusions and future development

Local memory-based techniques are powerful techniques for learning from observed data and for gaining insight on the local behaviour of non-linear systems. Furthermore, together with the required prediction and/or parametric description, they return a statistical distribution of the uncertainty affecting this information. As far as modelling is concerned, 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 the control domain, we illustrated and analysed three control systems which make extensive use of local modelling. 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. As in neuro-control, properties of stability cannot be guaranteed in a general case. We showed, however, that this approach obtains performances more accurately than neural networks even when 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 into the dynamic properties of the non-linear system. In addition, from a computational point of view, it is the least 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

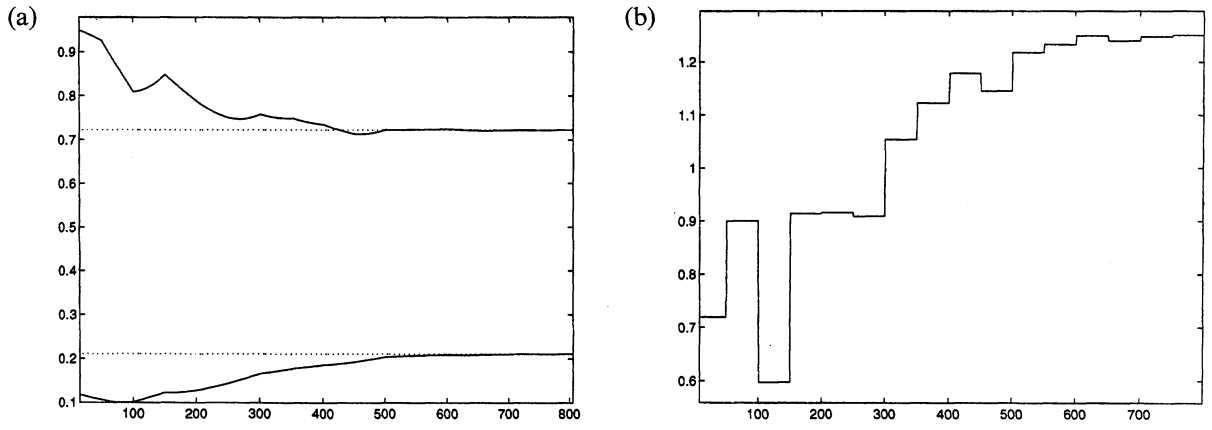

Figure 10. Control results: (a) system outputs (solid) and references (dotted); (b) control action (each discrete-time step corresponds to 0.01 s).

extend our analysis to robust and stochastic dual control. In fact, *lazy learning* is one of the few algorithms for non-linear modelling which returns a statistical description of the uncertainty affecting the prediction and the model parameters with minimal additional computational cost.

There are still two main concerns for the adoption of lazy learning for modelling and control. The first is the well known problem of the *curse of dimensionality*, that is the sparseness of data in situations of high dimension of the query space. Lazy learning shares this problem with all other non-linear modelling techniques but unlike model-based approaches lazy methods can take advantage from their feature of updating continuously the set of observed samples (Atkeson *et al.* 1997 b). Techniques of active learning (Cohn *et al.* 1995), which detect regions where data are missing in order to collect additional samples, could be found useful in this context.

A second major concern is how can *lazy learning* model real systems fast enough when the size of the database grows. This paper does not deal with this issue but several researchers are fast exploring ways to find relevant data for local approximation using efficient software algorithms and special purpose hardware. The integration of these methods with the modelling and control techniques we presented here makes *lazy learning* one of the most promising tools for non-linear control.

Acknowledgments

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

Appendix

A1. *The paired permutation test*

Consider the null hypothesis H_0 that the vector of leave-one-out errors $\mathbf{E}_n^{cv} = [e_n^{cv}(i)]_{i=1}^n$ and the first n elements of the vector $\mathbf{E}_{n+1}^{cv} = [e_{n+1}^{cv}(i)]_{i=1}^{n+1}$, 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 $\mathbf{E}_n^{cv}$ and $\mathbf{E}_{n+1}^{cv}$ with the same probability. This means that, if H_0 is true, the difference $d_i = e_n^{cv}(i) - e_{n+1}^{cv}(i)$ 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 $n+1$ neighbours is assumed to be significantly greater than the one of the model fitted on n neighbours.

A2. *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

$$J = E\{[P(z)y(k+d) + Q(z)u(k) - y_{ref}(k)]^2\} \quad (A1)$$

where $P(z) = P_N(z)/P_D(z)$ and $Q(z) = Q_N(z)/Q_D(z)$. Suppose that data are generated according to model (12). Multiplying both sides of (12) by P we obtain

$$\frac{P_N(z)}{P_D(z)}y(k) = \frac{P_N(z)}{P_D(z)}\left(\frac{B(z)}{A(z)}u(k-d) + \frac{b}{A(z)} + \frac{C}{A(z)}e(k)\right) \quad (A2)$$

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 $\tilde{b} = P_N b$

$$\tilde{A}(z)\tilde{y}(k) = z^{-d}\tilde{B}(z)u(k) + \tilde{C}e(k) + \tilde{b} \quad (A3)$$

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

$$\tilde{C}(z) = \tilde{A}(z)E(z) + z^{-d}\tilde{F}(z) \quad (A4)$$

Multiplying both sides of (A 3) by $z^d E(z)$ gives

$$\tilde{A}(z)E(z)\tilde{y}(k+d) = \tilde{B}(z)E(z)u(k) + \tilde{C}e(k+d) + \tilde{b}E(z) \quad (A5)$$

From (A 4) we have

$$\tilde{C}(z)\tilde{y}(k+d) = \tilde{B}E(z)u(k) + \tilde{C}e(k+d) + \tilde{b}E(z) + \tilde{F}(z)\tilde{y}(k) \quad (A6)$$

The optimal control law is then

$$\tilde{C}(z)\left(y_{ref} - \frac{Q_N}{Q_D}u(k)\right) = \tilde{B}E(z)u(k) + \tilde{b}E(z) + \tilde{F}(z)\tilde{y}(k) \quad (A7)$$

that is equivalent to

$$P_D(z)Q_D(z)C(z)y_{\text{ref}} - Q_N(z)P_D(z)C(z)u(k) = Q_D(z)\tilde{F}(z)y(k) + Q_D(z)P_D(z)B(z)E(z)u(k) + Q_D(z)P_D(z)bE \quad \text{(A 8)}$$

in the plant polynomials. The control law is then

$$u(k) = \frac{H(z)y_{\text{ref}} - F(z)y(k) - Q_D(z)P_D(z)bE(z)}{G(z)} \quad \text{(A 9)}$$

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 $Q_N = 0$.

A3 *Pole placement design with offset term*

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

$$H_m(z) = \frac{B_m(z)}{A_m(z)} \quad \text{(A 10)}$$

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

$$u(k) = \frac{T(q)}{R(q)}y_{\text{ref}}(k) - \frac{S(q)}{R(q)}y(k) - G(q) \quad \text{(A 11)}$$

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 (12) and (A 10) Hence:

$$y = \frac{B_d T}{AR + B_d S}y_{\text{ref}} + \frac{R(b - GB_d)}{AR + B_d S} + \frac{CR}{AR + B_d S}e \quad \text{(A 12)}$$

with $B_d = z^{-d}B$. Requiring that this input–output relation is equivalent to (A 10) gives

$$\frac{B_d T}{AR + BS} = \frac{B_m}{A_m} \quad \text{(A 13)}$$

$$b = GB_d \quad \text{(A 14)}$$

The pole-placement design problem with offset term is then equivalent to the conventional one, once the additional requirement (A 14) is satisfied.

References

AHA, D. W., 1989, Incremental, instance-based learning of independent and graded concept descriptions. (San Mateo, CA: Morgan Kaufmann), pp. 387–391.

AHA, D. W., 1997, Editorial. *Artificial Intelligence Review*, **11**, 1–6. (*Special Issue on Lazy Learning*).

ÅSTRÖM, K. J., 1967, Computer control of a paper machine. An application of linear stochastic control theory. *IBM Journal of Research and Development* **11**, 389–404.

ÅSTRÖM, K. J., 1983, Theory and applications of adaptive control—a survey. *Automatica*, **19**, 471–486.

ÅSTRÖM, K. J., and WITTENMARK, B., 1990, *Computer-controlled Systems: Theory and Design.* (Prentice-Hall International Editions).

ATKESON, C. G., MOORE, A. W., and SCHAAL, S., 1997a, Locally weighted learning. *Artificial Intelligence Review*, **11**, 11–73.

ATKESON, C. G., MOORE, A. W., and SCHAAL, S., 1997b, Locally weighted learning for control. *Artificial Intelligence Review*, **11**, 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., and GORRINI, V., 1997, A simplification of the back-propagation-through-time algorithm for optimal neurocontrol. *IEEE Transactions on Neural Networks*, **8**, 437–441.

BERSINI, H., BIRATTARI, M., and BONTEMPI, G., 1998, Adaptive memory-based regression methods. *Proceedings of the 1998 IEEE International Joint Conference on Neural Networks*, 2102–2106.

BOTTOU, L., and VAPNIK, V. N., 1992, Local learning algorithms. *Neural Computation*, **4**, 888–900.

CARPENTER, G. A., and GROSSBERT, S., 1988, The art of adaptive pattern recognition by a self-organising neural network. *IEEE Computer*, **21**, 77–88.

CLARKE, D. W., 1994, *Advances in Model-Based Predictive Control* (Oxford University Press).

CLARKE, D. W., and GAWTHROP, P. J., 1975, Self tuning controller. *Proceedings IEEE*, **122**, 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, MA: The MIT Press).

COHN, D. A., Ghahramani, Z., and JORDAN, M. I., 1995, Active learning with statistical models. In G. Tesauro, D. Touretzky, and T. Leen (Eds) *Advances in Neural Information Processing Systems*, p. 7

COVER, T., and HART, P., 1967, Nearest neighbor pattern classification. *Proceedings of the IEEE Transactions by Information Theory*, 21–27.

CYBENKO, G., 1996, Just-in-time learning and estimation. In S. Bittanti and G. Picci (Eds), *Identification, Adaptation, Learning. The Science of Learning Models from Data* (NATO ASI Series, Springer), pp. 423–434.

EFRON, B., and TIBSHIRANI, R. J., 1993, *An Introduction to the Bootstrap* (New York, NY: Chapman and Hall).

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

FAN, J., and GIJBELS, I., 1995, Data-driven bandwidth selection in local polynomial fitting: variable bandwidth and spatial adaptation. *Journal of the Royal Statistics Society, Series B*, **57**, 371–394.

FARMER, J. D., and SIDOROWICH, J. J., 1987, Predicting chaotic time series. *Physical Review Letters*, **8**, 845–848.

FEL'DBAUM, A. A., 1965, *Optimal Control Systems* (New York, NY: Academic Press).

GEMAN, S., BIENENSTOCK, E., and DOURSAT, R., 1992, Neural networks and the bias/variance dilemma. *Neural Computation*, **4**, 1–58.

GOODWIN, G. C., and SIN, K. S., 1984, *Adaptive Filtering Prediction and Control* (Prentice-Hall).

GRACE, A., 1994, *Optimization Toolbox For Use with MATLAB* (The MATHWORKS Inc.).

HUNT, K. J., and JOHANSEN, T. A., 1997, Design and analysis of gain-scheduled control using local controller networks. *International Journal of Control*, **66**, 619–651.

JACOBS, R. A., JORDAN, M. I., and BARTO, A. G., 1991, Task decomposition through competition in a modular connections architecture: The what and where vision tasks. *Cognitive Science*, **15**, 219–250.

JOHANSEN, T. A., and FOSS, B. A., 1993, Constructing NARMAX models using ARMAX models. *International Journal of Control*, **58**, 1125–1153.

JOHANSEN, T. A., and FOSS, B. A., 1995, Semi-empricial modeling of non-linear dynamic systems through identification of operating regimes and local models. In K. H. Hunt, G. R. Irwin, and K. Warwick (Eds) *Neural Network Engineering in Dynamic Control Systems* (Springer).

JORDAN, M. I., and RUMELHART, D. E., 1992, Forward models: supervised learning with a distal teacher. *Cognitive Science*, **16**, 307–354.

LEONARITIS, I. J., and BILLINGS, S. A., 1985, Input–output parametric models for non-linear systems. *International Journal of Control*, **41**, 303–344.

MILLER, W. T., SUTTON, R. S., and WERBOS, P. J. (Eds), 1990, *Neural Networks for Control* (The MIT Press).

MOODY, J., and DARKEN, C. J., 1989, Fast learning in networks of locally-tuned processing units. *Neural Computation*, **1**, 281–294.

MOORE, A. W., SCHNEIDER, J., and DENG, K., 1997, Efficient locally weighted polynomial regression predictions. In *Proceedings of the 1997 International Machine Learning Conference*, vol. 15 (Morgan Kaufmann Publishers), pp. 219–250 (ftp at http://www.cs.cmu.edu/~awm/papers.html).

MURRAY-SMITH, R., and HUNT, K., 1995, Local model architectures for non-linear modelling and control. In K. J. Hunt, G. R. Irwin and K. Warwick (Eds) *Neural Network Engineering in Dynamic Control Systems* (Springer), pp. 61–82.

MURRAY-SMITH, R., and JOHANSEN, T. A. (Eds), 1997, *Multiple Model Approaches to Modelling and Control* (Taylor & Francis).

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

NARENDRA, K. S., and ANNASWAMY, A. M., 1989, *Stable Adaptive Systems* (Englewood Cliffs, NJ: Prentice Hall).

NARENDRA, K. S., and BALAKRISHNAN, J., 1997, Adaptive control using multiple models. *IEEE Transactions on Automatic Control*, **42**, 171–187.

NARENDRA, K. S., and LI, S. M., 1996, Neural networks in control systems. In P. Smolensky, M. C. Mozer and D. E. Rumelhart (Eds) *Mathematical Perspectives on Neural Networks* (Lawrence Erlbaum Associates), Ch. 11, pp. 347–394.

NENE, S. A., and NAYAR, S. K., 1997, A simple algorithm for nearest neighbor search in high dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **19**, 989–1003.

PASSINO, K. M., and YURKOVICH, S., 1996, Fuzzy control. In W. S. Levine (Ed). *The Control Handbook* (IEEE Press), pp. 1001–1017.

PRIESTLEY, M. B., 1988, *Non-linear and Non-stationary Time Series Analysis*. (Academic Press).

RHODES, C., MORARI, M., TSIMRING, L. S., and RULKOV, N. F., 1997, Data-based control trajectory planning for non-linear systems. *Physical Review E*, **56**.

RUGH, W. J., 1991, Analytical framework for gain scheduling. *IEEE Control Systems*, **11**, 79–84.

RUPPERT, D., SHEATHER, S. J., and WAND, M. P., 1995, An effective bandwidth selector for local least squares regression. *Journal of the American Statistical Association*, **90**, 1257–1270.

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

SCHAAL, S., and ATKESON, C. G., 1994, Robot juggling: implementation of memory-based learning. *IEEE Control Systems*, February, 57–71.

SCHOTT, K. D., and BEQUETTE, B. W., 1997, Multiple model adaptive control. In R. Murray-Smith and T. A. Johansen (Eds) *Multiple Model Approaches to Modeling and Control* (Taylor and Francis), pp. 269–291.

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

SIEGEL, S., and CASTELLAN, N. J, JR., 1988, *Non Parametric Statistics for the Behavioral Sciences*, 2nd edn (McGraw-Hill International).

SJOBERG, J., and AGARWAL, M., 1997, Non-linear controller tuning based on linearized time-variant model. *Proceedings of American Control Conference*.

SLOTINE, J. J., and LI, W., 1991, *Applied Non-linear Control* (Prentice-Hall International).

STENGEL, R. F., 1986, *Stochastic Optical Control: Theory and Application* (New York, NY: John Wiley and Sons).

STENMAN, A., GUSTAFSSON, F., and LJUNG, L., 1996, Just in time models for dynamical systems. *35th IEEE Conference on Decision and Control*, pp. 1115–1120.

STOICA, P., EYKHOFF, P., JANSSEN, P., and SÖDERSTRÖM, T., 1986, Model-structure selection by cross-validation. *International Journal of Control*, **43**, 1841–1878.

TAKAGI, T., and SUGENO, M., 1985, Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on System, Man and Cybernetics*, **15**, 116–132.

TANAKA, K., 1995, Stability and stabilizability of fuzzy-neural-linear control systems. *IEEE Transactions on Fuzzy systems*, **3**, 438–447.

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