DATA-DRIVEN TECHNIQUES FOR DIVIDE AND CONQUER ADAPTIVE CONTROL

Edy Bertolissi [*] **Mauro Birattari** [*] **Gianluca Bontempi** [*]
Antoine Duchâteau [*] **Hugues Bersini** [*]

[*] *IRIDIA, Université Libre de Bruxelles*
50, av. Franklin Roosevelt, 1050 Brussels, Belgium
{eberto, mbiro, gbonte, aduchate, bersini}@ulb.ac.be

Abstract: This paper presents an approach for modeling and controlling discrete-time non-linear dynamical system. The controller consists of a multiple step ahead direct adaptive controller. At each time step a forward simulation of the system composed by the controller and the plant model is performed. This dynamic information is then used to adapt the parameters of the controller. In order to obtain good results it is necessary to have a good model of the process to control. Takagi-Sugeno fuzzy systems and *lazy learning*, are two approaches which can be successfully used to model the controller plant. This paper focuses on case when a model of the plant is not given *a priori* and has to be learned starting from a limited amount of data and it is necessary to add some adaptation capabilities to perform on-line learning. Simulation examples of the control of the manifold pressure of a car engine using adaptive and non-adaptive versions of Takagi-Sugeno and Lazy models are given. *Copyright © 2000 IFAC*

Keywords: Adaptive control, non-parametric identification, fuzzy systems.

1. INTRODUCTION

A system can be modeled in many different ways, and the choice of a particular modeling and identification approach depends on the particular application. The idea of using linear techniques in a nonlinear setting is not new in control literature, but had recently gained popularity thanks to methods for combining multiple estimators and controllers in different operating regimes of the system (Murray-Smith and Johansen, 1997; Bontempi *et al.*, 1999).

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 paradigms have emerged: global versus *divide and conquer*. Global modeling builds a single functional model on the basis of the dataset. This is the traditional approach used in neural networks and other form of nonlinear statistical regression. The available dataset is used by a learning algorithm to produce a model of the mapping. Then the dataset is discarded and only the functional description is kept. Multiple model approaches (Murray-Smith and Johansen, 1997), also known as *divide and conquer* techniques, partition a complex problem into simpler ones, whose solutions can be combined to provide a solution of the original problem. An example of these approaches are the *modular* architectures where different modules are composed in order to cover the input space. Even if these approaches use the combination of local models, the learning procedure remains a functional estimation problem, and requires the same procedures used for generic global models. Takagi-Sugeno fuzzy systems (Takagi and Sugeno, 1985) represent a well known example of this approach. Another example of multiple model methods are the so called *local modeling* techniques, where the problem of function estimation is transformed into one of value estimation. In these approaches the goal is not to find a model which explains the whole process, but to find the best output for a specific given input (called *query*). Local techniques renounce to a complete description of the input-output relation, and aim at approximating the function only in the neighborhood of the point to be predicted. The objective of these techniques is to improve the prediction accuracy at the cost of a reduced readability of the resulting model. Since an approximation function is not calculated, it is necessary to keep into memory the whole dataset for each prediction, and therefore the quantity of memory required for these approaches is much larger than the one necessary in the other cases. *Lazy learning* (Aha, 1997) is one of these local modeling techniques.

The goal of this paper is to compare a Takagi-Sugeno fuzzy systems with *lazy learning* on the task of modeling an unknown dynamic system in the framework of the implementation of a direct adaptive controller (DAC). In this type of controllers, at each time step a forward simulation of the system composed by the controller and plant pair is performed. The results of this computation are then used to tune the controller by adapting its parameters. In order to perform the forward simulation a model of the plant is required. Its quality is directly related with the performance of the controller, since errors in the model could lead to incorrect updates of its parameters. This paper focuses on case when initially only a limited amount of process data is available, but further examples can be collected on line. In this case it is useful to add adaptation capabilities to model of the plant in order to capture its unmodeled dynamics. Takagi-Sugeno fuzzy systems and *lazy learning* can be successfully used within DAC for the modeling of the controlled plant, and they can be modified to be suitable for performing on line adaptation.

2. LOCAL MODELING AS AN OPTIMIZATION PROBLEM

In Takagi-Sugeno fuzzy systems the computation of the value of the unknown function is performed by the fuzzy interpolation of a set of linear systems which locally approximate the desired function. These systems define a set of fuzzy rules which partition the input space:

$$\mathcal{R}^{(j)}: \text{ IF } \varphi_1 \text{ IS } A_1^{(j)} \text{ AND } \dots \text{ AND } \varphi_n \text{ IS } A_n^{(j)}$$
$$\text{THEN } y = a_0^{(j)} + a_1^{(j)}\varphi_1 + \cdots + a_n^{(j)}\varphi_n$$

The system output is a weighted average of the individual rule outputs:

$$y = \sum_{j=1}^{M} \frac{\mu_{A^{(j)}}(\varphi)\left(a_0^{(j)} + a_1^{(j)}\varphi_1 + \cdots + a_n^{(j)}\varphi_n\right)}{\sum_{k=1}^{M}\mu_{A^{(k)}}(\varphi)} \tag{1}$$

where the weights $\mu_{A^{(j)}}(\varphi)$ are computed according to:

$$\mu_{A^{(j)}}(\varphi) = \prod_{i=1}^{n} \mu_{A_i^{(j)}}(\varphi_i)$$

This approach allows to model a system by means of the decomposition of a nonlinear mapping into a collection of local linear models. Since this approach imposes to structure the problem in a series of local models, Takagi-Sugeno models can be easily constructed from numerical data, which is used to identify the number and the attributes of the fuzzy rules.

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 is required. Let us consider an unknown mapping $f: \Re^m \to \Re$ of which we are given a set of N samples $\{(\varphi_{[i]}, y_{[i]})\}_{i=1}^{N}$. These examples can be collected in a matrix Φ 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 $(\varphi_{[i]}, y_{[i]})$ 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 $\varphi_{[i]}$. Each row of Φ and $\mathbf{y}$ is then multiplied by the corresponding weight creating the variables $\mathbf{Z} = \mathbf{W}\Phi$ 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 $(\mathbf{Z}^T\mathbf{Z})\beta = \mathbf{Z}^T\mathbf{v}$ and the prediction of the value $f(\varphi_q)$ is obtained evaluating such a model in the query point:

$$\hat{y}_q = \varphi_q^T(\mathbf{Z}^T\mathbf{Z})^{-1}\mathbf{Z}^T\mathbf{v}. \tag{2}$$

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). In this paper we take advantage of the method described in (Birattari *et al.*, 1999), which automatically selects, for each query point, the adequate configuration. This is done by importing tools and techniques from the field of linear statistical analysis. The most important of these tools is the PRESS

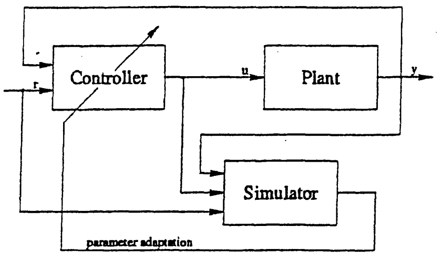

Fig. 1. Schematic representation of the system

statistic (Myers, 1994), which is a simple, well-founded and economical way to perform *leave-one-out* cross validation 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 the *lazy learning* approach to modeling data. In this modeling procedure the performance of a model in cross validation is the criterion adopted to choose the best local model configuration (Birattari *et al.*, 1999). One of the most important parameters to be tuned in a local model configuration is the size of the region surrounding φ_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 k of neighbors of φ_q to be used in the local regression fit. Thus, different models are considered, each fitted on a different number of examples, and the *leave-one-out* cross-validation is used 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, an incremental approach based on recursive linear techniques is adopted (Birattari *et al.*, 1999).

3. SYSTEM DESCRIPTION

Figure 1 shows the structure of a closed loop control system which is composed by: the plant to be controlled, the simulator, and the controller. Assume that the n input m output plant is expressed in terms of its input-output representation:

$$y_i(k+1) = F_i(\mathbf{y}(k),\dots,\mathbf{y}(k-p_y+1), \mathbf{u}(k),\dots,\mathbf{u}(k-p_u+1)), \quad i=1,\dots,m \quad (3)$$

where the scalar $y_i(k)$ is the i^{th} output of the plant at time k, $\mathbf{u}(k)$ is the input vector $[u_1(k),u_2(k),\dots,u_n(k)]^T$ and $\mathbf{y}(k)$ is the output vector $[y_1(k),y_2(k),\dots,y_m(k)]^T$, F_i is an unknown nonlinear function, and p_y and p_u are the known structure orders of the system for the output i. If the regressor vector $\phi(k)$ is defined as follows:

$$\phi(k) = [\mathbf{y}(k),\dots,\mathbf{y}(k-p_y+1), \mathbf{u}(k-1),\dots,\mathbf{u}(k-p_u+1)] \quad (4)$$

equation (3) can be rewritten as:

$$y_i(k+1) = F_i(\phi(k),\mathbf{u}(k)), \qquad i=1,\dots,m \quad (5)$$

which in vector notation becomes:

$$\mathbf{y}(k+1) = \mathbf{F}(\phi(k),\mathbf{u}(k)) \quad (6)$$

The control action $\mathbf{u}(k)$ is fed into the plant:

$$\mathbf{u}(k) = \mathcal{F}(\mathbf{r}(k),\psi(k);\mathbf{w}) \quad (7)$$

where $\psi(k)$ is a regressor vector obtained from time delayed values of the inputs and of the outputs, $\mathbf{w}$ is the set of parameters describing the controller and $\mathbf{r}(k)$ is the vector of the reference signals $[r_1(k),r_2(k),\dots,r_m(k)]^T$. The controller will produce a control action $\mathbf{u}(k)$ which will drive the plant outputs $\mathbf{y}(k+1)$ at the values specified by the vector $\mathbf{r}(k)$. Since the dynamics of the controller is much faster than the one of the plant to be controlled it is supposed that the control action at time k is influenced by the output of the plant at time k, without any delay.

The predictor performs a forward simulation of the system composed by the plant and the controller. Each predicted output of the plant is equal to:

$$\hat{y}_i(k+1) = \hat{F}_i(\phi(k),\mathbf{u}(k)), \qquad i=1,\dots,m \quad (8)$$

where $\hat{F}_i$ is the estimate of F_i, learned on the basis of the available dataset. The results of the forward simulation are then used to perform the parametric adaptation of the controller.

4. MULTIPLE STEP AHEAD ADAPTIVE CONTROL

Using the structure of the system depicted in figure 1 it is possible to design a learning algorithm, based on the principles defined in generalized predictive control theory (Clarke *et al.*, 1987), which performs an adaptation of the weights of the controller using information about the future behavior of the system. The predictor will provide the controller with information regarding the futures values of the $\hat{\mathbf{y}}$ and $\mathbf{u}$ up to the prediction horizon. On the basis of these predictions, we can design an adaptation algorithm based on the gradient descent:

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \eta\frac{\partial J}{\partial \mathbf{w}} \quad (9)$$

In order to train the parameters $\mathbf{w}$ the following cost function is selected:

$$J = \frac{1}{2}\sum_{t=k}^{k+H_p} (\mathbf{r}(t)-\hat{\mathbf{y}}(t))^T Q(\mathbf{r}(t)-\hat{\mathbf{y}}(t)) + \frac{1}{2}\sum_{t=k-1}^{k+H_c} (\Delta\mathbf{u}(t))^T R(\Delta\mathbf{u}(t)) \tag{10}$$

where $\Delta\mathbf{u}(t) = \mathbf{u}(t) - \mathbf{u}(t-1)$ and $\mathbf{u}(k-2) = 0$. The matrix $Q \in \Re^{n\times n}$ weights the errors $(\mathbf{r}(t) - \hat{\mathbf{y}}(t))$ while the matrix $R \in \Re^{m\times m}$ has the effect to penalize the large variations of $\Delta\mathbf{u}(t)$ which could destabilize the system. Substituting this expression in equation 9 and recalling equation 6 it is possible to obtain:

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \eta\left(\sum_{t=k}^{k+H_p} (\mathbf{r}(t)-\hat{\mathbf{y}}(t))^T Q\frac{\partial\hat{\mathbf{y}}(t)}{\partial\mathbf{w}} + \sum_{t=k-1}^{k+H_c} (\Delta\mathbf{u}(t))^T R\frac{\partial\Delta\mathbf{u}(t)}{\partial\mathbf{w}}\right) \tag{11}$$

This expression can be calculated by recursively compute the following two expressions:

$$\frac{\partial\mathbf{u}(t-1)}{\partial\mathbf{w}} = \frac{\partial\mathcal{F}(\mathbf{r}(t-1),\psi(t-1);\mathbf{w})}{\partial\mathbf{w}} = \frac{\partial\mathcal{F}(\cdots)}{\partial\mathbf{w}} + \frac{\partial\mathcal{F}(\cdots)}{\partial\psi(t)}\frac{\partial\psi(t)}{\partial\mathbf{w}} \tag{12}$$

and

$$\frac{\partial\hat{\mathbf{y}}(t)}{\partial\mathbf{w}} = \frac{\partial\hat{\mathbf{F}}(\mathbf{u}(t-1),\phi(t-1))}{\partial\mathbf{w}} = \frac{\partial\hat{\mathbf{F}}(\cdots)}{\partial\phi(t-1)}\frac{\partial\phi(t-1)}{\partial\mathbf{w}} + \frac{\partial\hat{\mathbf{F}}(\cdots)}{\partial\mathbf{u}(t-1)}\left(\frac{\partial\mathcal{F}(\cdots)}{\partial\mathbf{w}} + \frac{\partial\mathcal{F}(\cdots)}{\partial\psi(t-1)}\frac{\partial\psi(t-1)}{\partial\mathbf{w}}\right) \tag{13}$$

The full description of this algorithm can be found in (Bertolissi *et al.*, 2000*b*).

The multiple step algorithm is a quite complex algorithm. Unlike the one step ahead version of this algorithm, it allows to control system which are not minimum phase and where any desired output is not reachable in one step. The approach looks on a longer time horizon and takes into account a series of control actions in order to derive a control policy. The drawbacks of the method are a higher computational load and the need for a more precise model. The first cost is easily understandable and is due to the simulation at each step of the closed loop behavior over a long time horizon. The second cost comes from the fact that the long term predictions are more difficult to achieve and need better precision in order to avoid errors accumulation. The quality of the model is therefore fundamental for the implementation of a good controller. Takagi-Sugeno fuzzy systems and *lazy learning*, can both be used to produce a model of the plant to be controlled. However in all the cases when the dynamics of the controller plant cannot be completely defined from the available training data, or its dynamics is time variant, it is useful to add adaptation capabilities to the model of the plant in order to improve the performance of the system.

In the case of fuzzy systems, when new data becomes available, a new identification procedure is usually necessary to incorporate this new knowledge in the model. However an approach which allows the implementation of on-line learning consists in adapting only the linear consequences of the fuzzy rules, without modifying the position of the centers and the shape of the fuzzy rules. Given

$$y^j(\varphi) = a_0^{(j)} + a_1^{(j)}\varphi_1 + \cdots + a_n^{(j)}\varphi_n = l^{(j)}\varphi$$

$$A^{(j)} = \frac{\mu_{A^{(j)}}(\varphi)}{\sum_{k=1}^{M}\mu_{A^{(k)}}(\varphi)}$$

equation 1 can be rewritten as:

$$y = \sum_{j=1}^{M} A^{(j)}y^j(\varphi) = \sum_{j=1}^{M} A^{(j)}l^{(j)}\varphi = \boldsymbol{L}^T\boldsymbol{\Phi}$$

with $\boldsymbol{L}^T = [l^{(1)}\ldots l^{(M)}]$ and $\boldsymbol{\Phi} = [A^{(1)}\varphi\ldots A^{(M)}\varphi]^T$. This is a linear system whose parameters $\boldsymbol{L}$ are adapted using a recursive least mean square algorithm. This adaptation procedure is not equivalent to a new identification procedure, since it affects only the parameters which are used to describe the consequences of the fuzzy model. A real identification procedure would add rules or move them in order to capture the new unknown underlying dynamics.

In the case of *lazy learning* models, adaptation capabilities are implemented by adding the new data, as it becomes available, to the database of examples used to calculate the subsequent predictions. This means that the lazy learning simulator can be easily updated on line, while the controller is running. Moreover the addition of new points associated with unmodeled dynamics do not change the prediction accuracy of the model in other areas.

5. SIMULATION STUDIES

In this paper we propose, as an example, a subsection of one of the benchmarks defined within the FAMIMO project (ESPRIT LTR Project 21911). The aim of the controller is to set the position of the electrical throttle of a direct injection engine (GDI) which determines the the manifold pressure *pman* $\in$ [100 1024]. The process in a

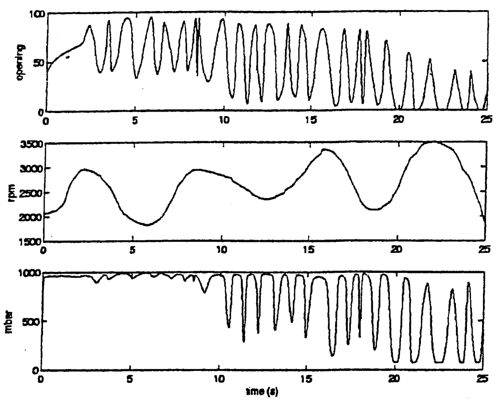

Fig. 2. Training sequence used to identify the model of the manifold pressure

second order highly nonlinear system. The control action is evaluated on the basis of the fresh air throttle control $MTC \in [0\ 100]$, which indicates the position of the throttle, and the engine speed $N \in [900\ 3400]$, which indicates the number of revolutions per minute (rpm) of the engine.

Since the aim of the paper is to show that it is possible to use a *lazy learning* approximator when an insufficient number of data is available for the modeling of the process, the plant has been identified using a reduced data set (5000 points) covering sparsely the state space. In this way the data provides information only on a part of the system dynamics. The training data used for the identification procedures is displayed in figure 2. A Takagi-Sugeno and a lazy learning model of the plant have been identified starting from the data. For the lazy learning simulator, a combination of the 6 best (according to the PRESS estimate) linear local models built using between 30 and 60 neighbors has been used (Birattari *et al.*, 1999). For Takagi-Sugeno fuzzy system an incremental identification procedure has been used (Bersini *et al.*, 1997). This procedure automatically generates a fuzzy partition of the process input space by letting grow the number of fuzzy rules until an optimum (here 13 rules) is achieved with respect to the given performance criteria.

In this particular experience the controller itself has been implemented as a Takagi-Sugeno fuzzy system. It is worth noting that any other (differentiable) parametric controller could have been used as well and that there is no correlation between the choice of a particular controller and the choice of the type of simulator. The control action is the position of the throttle (MTC), and it is calculated on the basis of the current value of the manifold pressure (pman), the desired value of the value of the manifold pressure (pman_d), and the number of revolutions per minute of the engine (N). The sampling time of the system is equal to

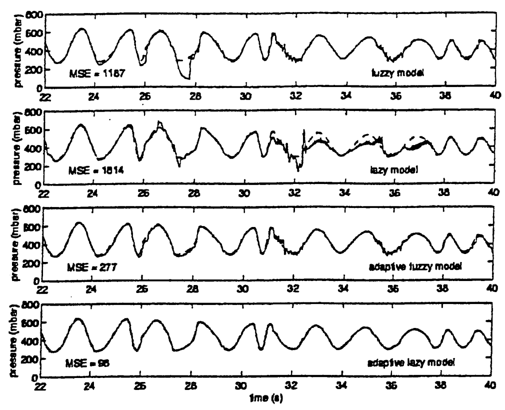

Fig. 3. Starting from the top, performance of the direct adaptive controller based on a fuzzy model, a lazy model, a fuzzy adaptive model, and a lazy adaptive model (bottom). The solid line is the actual output of the plant, while the reference is represented by the dashed line.

5 milliseconds. The controller is described by 108 Takagi-Sugeno fuzzy rules which cover the input space. At the beginning the consequences of all the rules have been initialized to 0. The adaptation algorithm uses a 5 step ahead prediction horizon and a learning rate $\eta = 10^{-8}$.

Two sets of experiments were performed. The first two experiences assessed the control capabilities of the simulator based on the fuzzy and lazy model without adaptation capabilities, while in the following set of experiences adaptation capabilities have been added to the models. In all the experiments the direct adaptive controller has to follow a pseudo sinusoidal signal which pushes the plant in areas where its dynamic are not well described by the training data. Figure 3 shows the performance of the direct adaptive controller adaptive (solid line) after 22 seconds of training. The first two graphs show the performance of the fuzzy and lazy models built starting from the initial 5000 input-output pairs. It is possible to notice that in both cases the adaptive controller shows a poor performance. In particular it is possible to notice that both controllers tend to show deficiencies in the same areas, but the output of the controller based on the fuzzy model is smoother. This means that the prediction and the values of the derivatives in the lazy model are less reliable, leading the system to exhibit high frequency components. The smoother performance of the controller based on the fuzzy model may be caused by the fact that when the estimation of the model is done in an area where limited information is available, the fuzzy system offers better extrapolation abilities than the lazy system. However the main point of these two graphs is to highlight the fact that the given initial data is insufficient to build a model of the plant required to perform a good forward simulation. The two bottom graphs of figure 3 show the performance of the direct adaptive controller when used in connection with the adaptive versions of the fuzzy and the lazy model of the plant. In both cases it is possible to see an improvement of the performance of the system, however it is clear from the image that the controller based on the adaptive lazy system outperforms the one based on the fuzzy model. In the case of the fuzzy model the recursive least mean square algorithm is used to constantly update the parameters of the consequences of the rules of the fuzzy model. In the case of the lazy model the input-output regressor is added to the database each time the prediction differs more than 0.02% from the output of the system. In the case of controller based on the lazy system it is possible to see that the ringing signal which characterized the controller build in conjunction with its non adaptive version disappears. This is because as new points are added to the database of examples (a 15% increase in this particular simulation) the lazy model can find neighbors that are closer to the query point, and does not need to extrapolate: it increases the stability and the accuracy of the derivatives.

All the experiments have been performed taking advantage of the NLMIMO Toolbox (Bertolissi *et al.*, 2000*a*), which integrates the Lazy Learning Toolbox[1] (Birattari and Bontempi, 1999) as one of its components.

6. CONCLUSIONS

Divide and conquer techniques are powerful techniques for learning from a limited amount of data. We illustrated and analyzed the performance of an adaptive controller used in conjunction with four different approaches for modeling the plant to be controlled, starting from a limited amount of input-output data. When the model of the plant is not adaptive it has been seen that the use of a Takagi-Sugeno model leads the system to a smoother response. However when adaptation capabilities are added to the lazy and fuzzy models the performance of the system improves, and the controller based on the lazy adaptive system outperforms the one based on the fuzzy adaptive model. The adaptation procedures performed on the two models of the plant are not equivalent: in the case of the fuzzy system the model of the plant is stretched to take into account the new dynamics, in the case of lazy system new points which describe a new dynamics are added to the existing model. The two adaptation strategies also require a different level of computational effort, in the case of the adaptive fuzzy system it is necessary to perform a recursive least mean square adaptation of the parameters of the linears of the fuzzy rules, while in the case of the lazy adaptive system it is just necessary to add point to the database of examples without additional computations.

[1] `http://iridia.ulb.ac.be/-lazy/`

7. ACKNOWLEDGEMENTS

This research has been supported by the European FAMIMO project (LTR Project 21911 Reactive Scheme, Task 4.2) and by a Marie Curie Fellowship awarded to Edy Bertolissi (CEC-TMR Contract No. ERBFMBICT972495).

8. REFERENCES

Aha, D. W. (1997). Editorial. *Artificial Intelligence Review* **11**(1–5), 1–6.

Bersini, H., A. Duchateau and N. Bradshaw (1997). Using incremenmtal learning algorithms in the search for minimal and effective fuzzy models. In: *Proceedings of the FUZZ-IEEE '97*. Barcelona, Spain. pp. 1417–1422.

Bertolissi, E., A. Duchateau and H. Bersini (2000*a*). Nlmimo, non-linear multi-input multi-output toolbox. In: *Proc. of the MATHMOD 2000 Conference*. Vienna, Austria.

Bertolissi, E., A. Duchateau, H. Bersini and F. Vanden Berghen (2000*b*). Direct fuzzy control for MIMO processes. In: *Proceedings FUZZ-IEEE 2000*. San Antonio, Texas.

Birattari, M. and G. Bontempi (1999). The lazy learning toolbox, for use with matlab. Technical Report TR/IRIDIA/99-7. IRIDIA-ULB. Brussels, Belgium.

Birattari, M., G. Bontempi and H. Bersini (1999). Lazy learning meets the recursive least-squares algorithm. In: *Advances in Neural Information Processing Systems 11* (M. S. Kearns, S. A. Solla and D. A. Cohn, Eds.). MIT Press. Cambridge. pp. 375–381.

Bontempi, G., M. Birattari and H. Bersini (1999). Lazy learning for modeling and control design. *International Journal of Control* **72**(7/8), 643–658.

Clarke, D. W., C. Mohtadi and P. S. Tuffs (1987). General predictive control - part 1. The basic algorithm. *Automatica* **23**(2), 137–148.

Murray-Smith, R. and Johansen, T. A., Eds.) (1997). *Multiple Model Approaches to Modeling and Control*. Taylor and Francis.

Myers, R. H. (1994). *Classical and Modern Regression with Applications*. second ed.. PWS-KENT Publishing Company. Boston, MA.

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