Int. J. Engng Ed. Vol. 22, No. 1, pp. 205–215, 2006
Printed in Great Britain.

0949-149X/91 $3.00+0.00
© 2006 TEMPUS Publications.

A Control Engineering Laboratory Based on a Low-Cost Non-linear Plant*

JUAN A. MÉNDEZ, SANTIAGO TORRES, LEOPOLDO ACOSTA, MARTA SIGUT and LORENZO MORENO
Department of Applied Physics, University of La Laguna, 38271 La Laguna, Tenerife, Spain.
E-mail: jamendez@ull.es

A real-time control laboratory is presented here. The experiments are based on a non-linear plant and are oriented to undergraduate courses on digital control and intelligent control. The strategies proposed are both classical and intelligent techniques. Thus, the students can compare, on the real plant, the results obtained with both approaches. The control problem is to eliminate the load oscillations in a transport crane. To solve this problem the students will apply a classical state feedback controller and then will improve this algorithm with intelligent techniques based on neural networks (NN) and fuzzy logic (FL). The experiments are carried out on a low-cost scale prototype.

Keywords: control laboratory; low cost plant; fuzzy logic; student motivation

INTRODUCTION

IN THE CONTEXT of control engineering, the theoretical concepts sometimes reach such a degree of complexity that the students have problems following the lectures. In these circumstances, a well-planned laboratory is crucial to complement the theoretical lectures. In this way, the control concepts are reinforced and, at the same time, students have to deal with problems derived from the implementation of the control algorithms in real systems: such as application of A/D and D/A converters, control actions in real-time, systems constraints on commands, etc.

This paper presents some experiments for undergraduate courses on digital control and intelligent control. The first three experiments are oriented to complement the theoretical lectures of a digital control course. Before the students take this course on digital control, they have attended a previous one on linear systems: Laplace transform theory, time and frequency response, stability, etc. In the digital control course, topics such as Z-transform theory, digital filtering, state feedback control and pole-placement control are studied. Thus, with the proposed experiments they have the opportunity to see in practice some of the methods explained in the lectures, such as data acquisition and actuation issues, selection of sampling rate and state feedback control. The second set of experiments is oriented to an intelligent control course. The program of this course is basically neurocontrol, fuzzy control and real-time expert systems. With the proposed experiments, students can gain practice with intelligent tools based on neural networks and fuzzy logic. They can also observe the advantages and disadvantages of both classical and intelligent techniques. The training of the students is completed with a module on advanced control, where stochastic, predictive and optimal control is studied. For this subject, a complete laboratory is also proposed in [1] and [2].

To implement the different strategies, a non-linear system with four degrees of freedom is considered. The plant, which is very typical in the process industry, is an overhead crane [3]. Its task is to transport containers between two points on a plane. Due to the acceleration of the crab, the load mass suffers oscillations along the trajectory. These oscillations are especially undesirable at the arrival. The control problem will be focused on eliminating these oscillations. The laboratory is provided with a low-cost scale model of an overhead crane. In this prototype the students can check the proposed algorithms.

The first experiment proposed is to validate the model of the real plant. To do this, the students will make a set of open-loop experiments to compare the real-time results with the simulated ones and to verify whether the model parameters are correct. Then, they implement a control strategy based on state variable feedback (SVF). The controller parameters are calculated by means of the pole placement method. They can also be computed solving a linear quadratic optimization problem. In these experiments, an adequate choice of the closed-loop poles in the pole placement controller, or of the cost function matrices in the optimal controller, is critical in order to get good results.

As an alternative to these classical strategies, an intelligent subsystem is added to the controller. It consists of a neural network (NN) that provides the adequate values for the controller parameters in order to optimize the system response. This operation is performed on-line. Finally, the

* Accepted 19 December 2004.

design of a controller based on fuzzy logic (FL) is proposed. In view of the growing success of these strategies [4], which are applied in many industrial fields, it is important for the students to understand their methodology. This does not seem difficult, since the fuzzy controllers build the control rules according to linguistic reasoning. In this paper, the main design phases are described and application to the prototype is performed.

OVERVIEW OF INTELLIGENT TECHNIQUES

Neural networks basics

Artificial neural networks are mathematical algorithms designed using the biological principles of brain activities. Basically, a NN performs a mapping between an input space and an output space. The objective is to train the network, in such a way that, when an input is applied, the network produces the desirable response. Learning is accomplished by adjusting the interconnecting weights in response to training data sets. In this work the NN used are the most common family of feed-forward networks: the multilayer perceptron (MLP). An MLP is based on cascading neurones in layers. The MLP is trained with the backpropagation algorithm to adjust the weights of the net. This well-known algorithm uses a gradient descent technique [5]. The objective is to change the weights of the network in order to minimize a criterion function of the form:

$$J = \frac{1}{2}\sum_{q=1}^{N_L}(d_q(\mathbf{x}) - v_{Lq}(\mathbf{x}))^2 \qquad (1)$$

where L is the number of layers of the network, N_L is the number of nodes in layer l, $\mathbf{x}$ is the training sample, v_{Lq} is the q-th output of the network and $d_q(\mathbf{x})$ is the desired response of the q-th output. That is, the aim is to minimize the error between the output of the net and a pre-specified value. Because all the nodes in all the layers of the network contribute to the output v_{Lq}, it is necessary to update all these weights to minimize J. To accomplish this objective, the gradient descent method states that the weights must be updated in each iteration according to

$$w_{l,j,i}(k+1) = w_{l,j,i}(k) - \mu\frac{\partial J(k)}{\partial w_{l,j,i}} \qquad (2)$$

where $w_{l,j,i}$ is the weight which connects the output of the i-th node in layer $l-1$ to the input of the j-th node in layer l and μ is a constant called *learning rate*. The backpropagation algorithm can be summarized as follows:

Step 1) INITIALIZE weights to small random values.
Step 2) PROPAGATE the input signal forward throughout the network.
Step 3) COMPUTE the gradient of the cost function for each weight of the network.
Step 4) UPDATE weights according to the gradient descent law.
Step 5) GOTO step 2 and REPEAT the procedure until end condition is reached.

Fuzzy logic basics

Fuzzy logic (FL) theory applied to control engineering has developed greatly in recent years. The reason for this is that it offers easy and robust solutions to complex problems, allowing human reasoning to be applied to the control of systems. This reasoning is often qualitative, so the resultant actions are vague or ambiguous (soft, high, . . .), in contrast to the precise mathematics used by computers. FL solves this problem by considering a variable as having a membership degree to a fuzzy set. Fuzzy sets have been defined covering the full range of the variable, each value of the variable having a membership degree to each of the fuzzy sets. Expert rules will be applied to these fuzzy sets (inference). Due to the non-existence of fuzzy actuators or sensors (this is, with their inputs or outputs being fuzzy), it is necessary to add two steps to the control scheme: fuzzification and defuzzification. Therefore, there are three different elements to a scheme of fuzzy control:

- Fuzzification: The precise values of the inputs must be transformed into their correspondent fuzzy variables, with the corresponding membership degree to each fuzzy number.
- Inference: Direct application of the rules to the inputs. As a result the outputs are obtained with the correspondent membership degrees.
- Defuzzification: The imprecise values of the outputs must be transformed into precise ones.

Fuzzification

A fuzzy set $A \subset X$, X being the universe of discourse of A, is characterized by its membership function

$$\mu_A : X \rightarrow [0,1]$$

where $\mu_A(x)$ is the membership degree of the element x in the fuzzy set A for each $x \in X$. The set A can also be defined as a set of ordered pairs $(x, \mu_A(x))$. The fuzzification consists of obtaining the value $\mu_A(x)$ from x. If the universe of discourse belongs to real numbers, the fuzzy set is called a fuzzy number. The most employed fuzzy numbers are triangular and trapezoidal shaped. The human expert defines the magnitudes in vague terms, with expressions such as 'vigorously', 'little', 'far', 'very cold', etc., using these fuzzy numbers.

Inference

In a fuzzy design, a human expert expresses, by means of rules like

$$IF \begin{bmatrix} \textit{inputs verify} \\ \textit{some premises} \end{bmatrix} THEN \begin{bmatrix} \textit{output is a certain} \\ \textit{control action} \end{bmatrix},$$

the decisions s/he would take in specific situations. Each inference rule is applied to the inputs with their membership degrees. The result of this inference process is an output for each rule, with a membership degree obtained applying a specified *t*-norm on the inputs. All these outputs (fuzzy sets) are aggregated using a specified *s*-norm, thus obtaining the output fuzzy number.

Basically a *t*-norm is an operator that models the logical connective *and* is applied to fuzzy sets. Usually the *t*-norm is taken to be a *min* operator. On the contrary, an *s*-norm is an operator that models the logical connective *or* is applied to fuzzy sets, and is usually taken to be a *max* operator.

Defuzzification

The output fuzzy number resulting from the aggregation process will be transformed into a crisp value. For this, a defuzzification method like center-of-gravity, middle-of-maxima, height-defuzzification, etc., is applied to the output fuzzy number to get a precise value.

Application to a specific system

Consider a system with two inputs and one output and fuzzy rules extracted from the expert's knowledge:

R$_1$: if (x is A_1 and y is B_1) then (z is C_1)
R$_2$: if (x is A_2 and y is B_2) then (z is C_2)
. . .
R$_n$: if (x is A_n and y is B_n) then (z is C_n)

where $\{A_i\}_{i=1,\dots,n}$ are the fuzzy numbers in U of the first input variable of the rule's antecedent; $\{B_i\}_{i=1,\dots,n}$ in V the fuzzy numbers of the second one; and $\{C_i\}_{i=1,\dots,n}$ the fuzzy numbers in W of the output variable of the rule's consequences. Let $x_o \in U$ and $y_o \in V$ be two crisp input values; C_i' the fuzzy number inferred from x_o and y_o applying the rule R_i; and C' the fuzzy number given by the aggregation of each C_i'. Usually, Mamdani's method (or min-max-gravity method) [6] is used for the inference. With this method, the *t*-norm that operates on the fuzzy inputs is the *min* operator. Then, the membership degree of the value consequence of the rule R_i can be expressed generally as:

$$\mu_{C_i',\max} = \min\{\mu_{A_i}(x_o), \mu_{B_i}(y_o)\} \tag{3}$$

The membership degree to the inferred fuzzy number C_i' is:

$$\mu_{C_i'}(z) = \min\{\mu_{C_i}(z), \mu_{C_i',\max}\}, \quad z \in W \tag{4}$$

On the other hand, the *s*-norm that operates on the C_i' fuzzy results is the max operator

$$\mu_{C'}(z) = \max\{\mu_{C_1'(Z)}, \mu_{C_2'}(z), \dots, \mu_{C_n'}(z)\}, \quad z \in W \tag{5}$$

And the center-of-gravity method is employed for defuzzification:

$$z_o = \frac{\int_W z \cdot \mu_{C'}(z)\, dz}{\int_W \mu_{C'}(z)\, dz} \tag{6}$$

where z_o is the defuzzified value of the fuzzy set C'.

PLANT DESCRIPTION

The plant on which the experiments will be developed is shown in Fig. 1. This prototype consists of the following units:

- Crab: A vehicle that transports the load mass. It is powered by two DC motors joined to the driving wheels. The input voltage to each DC motor can vary between −8.5 V and +8.5 V. The voltage applied on a motor is proportional to the angular velocity reached by the motor. The maximum speed is about 60 rpm for an 8.5 V.
- Load mass: A container joined to the crab by a steel rope.
- Metal bridge of about 1.2 m length where the crab can move horizontally.

The values of the crab mass, the length of the rope

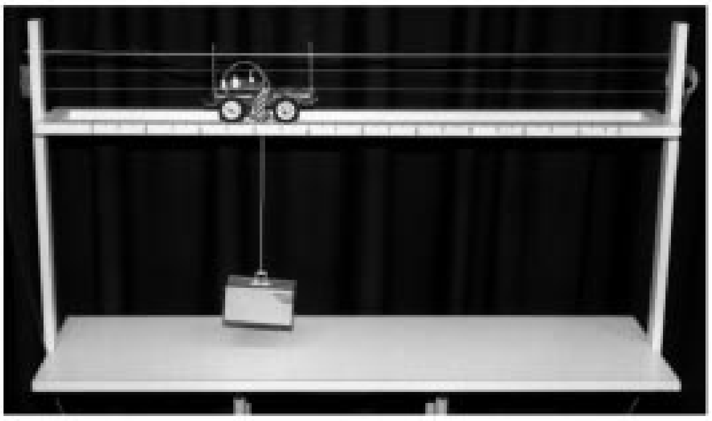
Fig. 1. General view of the crane prototype.

and the mass of the container used in the experiments are as follows:

- Crab mass: 0.3 kg.
- Load mass: 0.15 kg.
- Length of the rope: 0.5 m.

This scale model is complemented by an A/D D/A converter board, which sends the information to the control unit and the commands to the crab. The board is located on a PC, with the necessary control software.

CONTROL OBJECTIVE

The experiments proposed in this laboratory try to complete the theoretical background of the students. The experiments are divided into two groups: one is related to the design of classical control strategies and the other is related to the design of intelligent controllers. The objective of the proposed control strategies is to take the crab from an initial point to a desired one, arriving with no oscillations in the load mass. For this plant, a simple PID controller is not able to reach the control objective because of the multivariable specification.

For this reason, other control strategies are designed. The first two approaches are based on a state variable feedback (SVF) controller. To adjust the gains, two of the most common methods are used. First, the students try a pole placement strategy based on the Ackerman's formula. The second strategy consists of applying an LQ optimization procedure. The third experiment introduces an intelligent scheme of control consisting of the design of an NN to automatically adjust the SVF gains. The last experiment is the design of a controller based on fuzzy logic. Once the dynamics of this system is known (e.g. by observing the behaviour of the system under the previous controllers), the students are prepared to define the fuzzy variables and their subsets, and also the inference rules to achieve the control objective.

EXPERIMENTS

Model validation

This experiment is planned to take three hours. A schematic view of this overhead crane plant is shown in Fig. 2. The task of the crane is to transport containers with different masses from an initial point to a final one. There are four degrees of freedom: the position of the crab (x), the linear speed ($\dot{x}$), the rope angle (θ) and its angular speed ($\dot{\theta}$). It is assumed that the crab has a mass M, the length of the rope is l and the load mass is m. A simplification is done assuming that the rope has a negligible mass. The input of the plant, u, represents the horizontal force applied to the crab. The application of a non-zero force on the plant produces undesirable oscillations in the

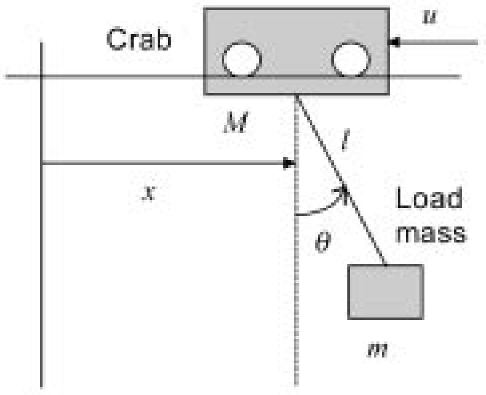

Fig. 2. Schematic view of the plant.

load mass. The control objective is to make the crab arrive at the final position with θ and $\dot{\theta}$ equal to zero. Defining the state variables $x_1 = x$, $x_2 = \dot{x}$, $x_3 = \theta$ and $x_4 = \dot{\theta}$ that control objective can be seen as taking the crane from the point $x_1 = x_{1i}$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$ to the set point $x_1 = x_{1\text{ref}}$, $x_2 = 0$, $x_3 = 0$ y $x_4 = 0$.

The dynamics of the plant can be described [7] by the following two non-linear second-order equations

$$\left.\begin{aligned} &ml(\dot{x}_4 \cos x_3 - x_4^2 \sin x_3) + (M+m)\dot{x}_2 = u \\ &\dot{x}_2 \cos x_3 + l\dot{x}_4 = -g \sin x_3 \end{aligned}\right\} \quad (7)$$

Some effects due to non-linearities must be considered. They are actuator saturation, dead zone and mechanical friction. The first two effects can be inserted in the model by considering the non-linear function:

$$V' = \begin{cases} 0, & \textit{if } |V| < 1.8 \\ V, & \textit{if } 1.8 \le |V| < 8.5 \\ 8.5, & \textit{if } |V| \ge 8.5 \end{cases} \qquad (8)$$

where V is the voltage (in volts) applied to the plant and V' the resultant voltage obtained by considering command constraints and dead zone. The voltage V is proportional to the speed in the system. However, the input signal u appearing in (7) is expressed in force units (kg m/s^2) and it is proportional to the acceleration. The following relation is found between the signals:

$$u = \beta \frac{dV'}{dt}$$
$$\beta = 0.015\,(\text{kg}\cdot\text{m}/(\text{s}\cdot\text{Volt})) \qquad (9)$$

The first experiment proposed to the students is to validate this model. This step is necessary to the later realization of any control experiment on the plant. They can do it by applying step voltage inputs and ramp voltage inputs to get the open-loop response of the real plant, and then comparing the results with those obtained by simulation of the model (7), (8) and (9) using the masses and

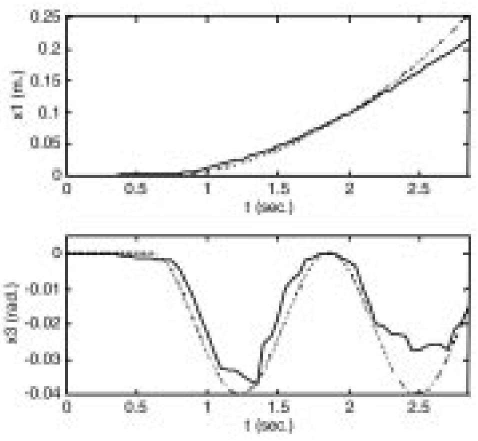

Fig. 3. Open-loop response using a voltage ramp input with slope 3 V. Theoretical response (dashed line) and measured response (continuous line).

lengths of the real plant. In Fig. 3 the theoretical (dashed line) and the measured (continuous line) evolution of the x_1 and the x_3 variables for a voltage ramp input with slope equal to 3 V is shown. As can be observed the theoretical model fits the real plant response relatively well. Observe that the amplitude of the load mass oscillations decreases as a consequence of the mechanical friction effects (not incorporated in the model). In view of this, the model (7) with the considerations given by (8) and (9) can be chosen as a representation of the plant dynamics.

After model validation, the real-time implementation of the different controllers in the prototype is performed. These controllers are described in the following sections.

Pole placement controller

The students are asked to design a pole placement controller. This will take approximately three hours. The procedure is based on obtaining a feedback control law so that the closed-loop system presents specified dynamics. The resultant control action has the following expression:

$$u(k) = k_1 e_1(k) - k_2 x_2(k) - k_3 x_3(k) - k_4 x_4(k) \tag{10}$$

being

$$e_1(k) = x_{1ref} - x_1(k) \tag{11}$$

where x_{1ref} is the reference value for the position x_1, while the reference value for the linear speed x_2, angle x_3 and angular speed x_4 is zero.

The k_i values are adjusted to get the desired closed-loop response in the system. So, the procedure is, first, to choose the desired closed-loop poles, and then to calculate the gains. As is well known, this is accomplished by means of the Ackerman's formula [8]:

$$K = [0 \quad 0 \quad \ldots \quad 1] M_c^{-1} P(A) \tag{12}$$

M_c being the controllability matrix of the system, A the matrix of the linearized system and $P(A)$ the characteristic polynomial of the closed-loop system with the matrix A substituted for the complex variable s.

Obviously this algorithm assumes a linear model for the plant. So the first step in the controller synthesis is the linearization of the model. The easiest way is to linearize around the equilibrium point. This can be done assuming small angles and ignoring the products of angular variables. In this way the model (7) is simplified to

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \frac{mg}{M} x_3 + \frac{1}{M} u \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= -\frac{(m+M)g}{lM} x_3 - \frac{1}{lM} u
\end{aligned} \tag{13}$$

The validity of the linearized model can be easily verified by the students comparing the response of the non-linear model (7) with the linearized model (13) when small angles are considered. The algorithm for the real-time implementation of this experiment is the following:

```
Step 1) FIX the measuring period T.
        CHOOSE the closed-loop poles and
        COMPUTE $k_1$, $k_2$, $k_3$, $k_4$ in Ackerman's
        formula.
Step 2) READ the inputs $x_1(k)$, $x_2(k)$, $x_3(k)$,
        $x_4(k)$.
Step 3) COMPUTE the output $u(k)$.
Step 4) APPLY $u(k)$ and WAIT until
        $t = (k+1)T$.
Step 5) $k \leftarrow k+1$.
Step 6) GOTO step 1.
```

The students in this experiment have to propose the closed-loop poles for satisfactory performance of the plant. Several guidelines are provided to the students: the poles at the origin have to be moved out of it; and the complex poles should have to have a given damping factor. Once they have an appropriate set of gains, the students have to implement the controller on the plant. They can implement it easily using any real-time software package, first programming the input/output routines for the specific A/D D/A converter.

As is well known, damping factors between 0.4 and 0.7 usually give good closed-loop performance. However, for the crane system this range of values implies large feedback gains and the response of the system is poor. In practice, to get better results it is necessary to reduce the damping factor of the complex poles. In Fig. 4 a real-time experiment is shown where the complex poles are selected to have a damping factor of 0.14. The sample period taken for the real-time implementation was T = 0.03 sec. This value is several times smaller than the fastest time constant of the system. It can be observed that the crane attains

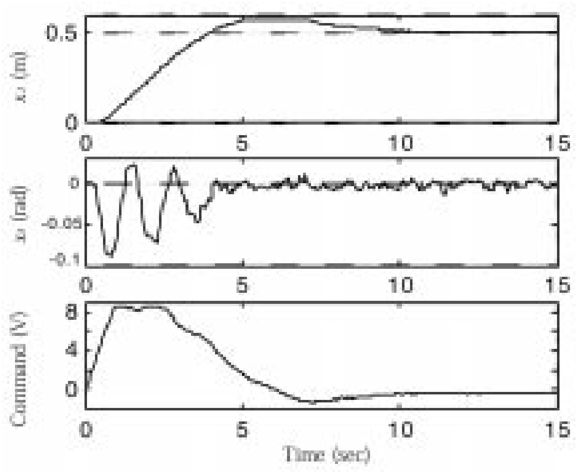

Fig. 4. Real-time experiment for S.V.F. controller. Poles at −0.5, −178, −0.76 5.28i. The damping factor for the complex poles is reduced to 0.14. The corresponding gains are $k_1 = 19.36$, $k_2 = 39.79$, $k_3 = -18.30$ and $k_4 = -3.54$. Set point at 0.5 m.

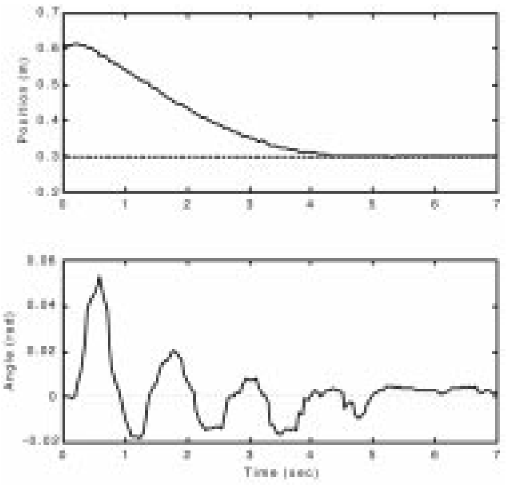

Fig. 5. Real-time experiment for optimal controller. The gains are $k_1 = 32.4$, $k_2 = 56.0$, $k_3 = -1.45$ and $k_4 = 0.03$. Set point at 0.3 m.

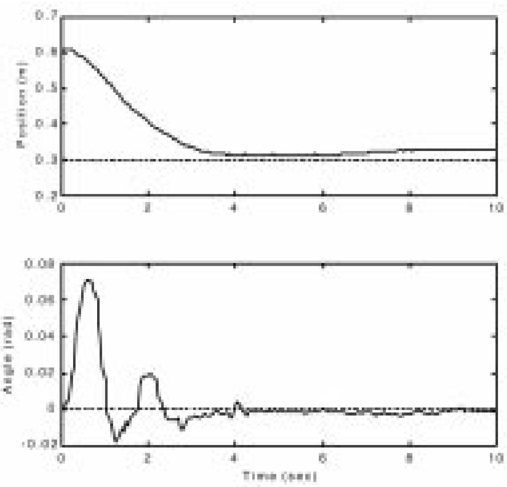

Fig. 6. Real-time experiment for optimal controller. The gains are $k_1 = 32.4$, $k_2 = 61.0$, $k_3 = -90.0$ and $k_4 = -0.02$. Set point at 0.3 m.

the set point smoothly but with overshooting. Simultaneously, the variable x_3 starts oscillating until the crane is near to the final position. At this point the mass oscillation decreases and tends towards the equilibrium point $x_3 = 0$, $x_4 = 0$. The settling time of the closed-loop system is about 10 sec.

The main drawback of this control strategy is related to the choice of the closed-loop poles. If the right values for the poles are not encountered, the feedback gains may take inappropriate values and the performance of the closed-loop system would decrease notably.

Optimal controller

In this section another classical control strategy is proposed to the students: an optimal controller. This will take approximately three hours. This strategy uses a control law like (10) where the feedback gains are chosen to optimize a linear quadratic function. A typical cost function is:

$$J = \sum_{k=1}^{\infty} (x_k^T Q x_k + u_k^T R u_k) \qquad (14)$$

where x is the 4×1 state vector. Q is, in this case, a 4×4 matrix that weighs the state variables and R is a factor that weighs the applied command. The choice of these matrixes, Q and R, will determine the values of the resulting feedback gain, and hence the performance of the closed-loop system.

The algorithm for the real implementation of this strategy is similar to the previous controller. The students should try different values for Q and R and compare the results obtained. Figure 5 shows an experiment with the gains obtained from the matrices:

$$Q = \begin{pmatrix} 105 & 0 & 0 & 0 \\ 0 & 310 & 0 & 0 \\ 0 & 0 & 0.05 & 0 \\ 0 & 0 & 0 & 0.015 \end{pmatrix}; \quad R = 0.1$$

With this choice, the weight in the variables corresponding to the position and velocity of the crab is more important in the cost function than the corresponding angle and the angular speed. This will produce higher gains in the position variables of the crab than in the angular variables. As can be seen, the position is attained efficiently but the angle is not corrected until 6 seconds.

If an appropiate weight is considered for the angle in the cost function, the oscillations of the load mass are cancelled more efficiently. In Fig. 6 an experiment corresponding to this situation is presented. The matrices chosen in this case are the following: $Q = \text{diag}[105 \;\; 310 \;\; 740 \;\; 2.5]$ and $R = 0.1$. Now the oscillations are cancelled in a shorter time than in the previous experiment. As a consequence, the position of the crab will not attain the set point and the response of the system loses efficacy.

Therefore, an adequate choice of the matrices of the cost function will be necessary if appropriate gain parameters are to be obtained. This problem is similar to the definition of the closed-loop poles of the pole placement controller seen in the previous section. In the following experiment, an intelligent control strategy is introduced.

State variable feedback controller using neural networks

The aim of this experiment is to propose to students a method for adaptively adjusting the parameters in a controller. This will take five hours in the laboratory. Three of these hours will be spent on simulation experiments and the real-time experiments will take two hours. Basically the goal of these real experiments is to show the students an application of neural networks to the control of a real plant. The idea is to use NN to adjust the parameters of a conventional controller by using information on the system state. The NN proposed acts as an implicit self-tuner directly providing the parameters to the controller. Actually, there is no estimation and control design stage. The NN, at every moment, simply looks for the optimal parameters to minimize a cost function using information on the system state.

This control scheme is shown in Fig. 7 [7]. As can be observed, an SVF controller similar to (10) is applied on the x_2 and x_3 state variables and on the x_1 error variable. Thus, the applied command is

$$u(k) = -[k_1 \quad k_2 \quad k_3] \times [-e_1(k) \quad x_2(k) \quad x_3(k)]^T \tag{15}$$

So there are three parameters to tune: the gains k_1, k_2 and k_3 corresponding to e_1, x_2 and x_3 respectively. The gain k_4 is fixed previously by students. One NN (NN1) is used to tune the parameters related to the crab variables, k_1 and k_2, and another (NN2) to provide the parameter related to the load mass variable, k_3. The inputs to the net NN1 are the x_1 error and the x_1 derivative and for NN2 the x_3 error and the x_3 derivative. This

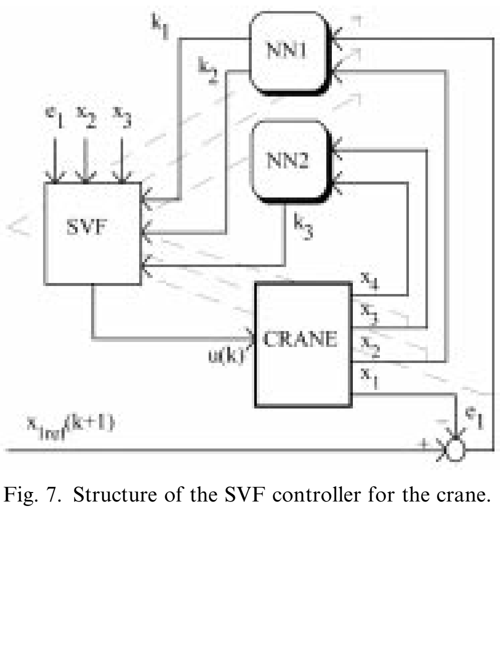

Fig. 7. Structure of the SVF controller for the crane.

structure based on two NNs is expected to work better than a one-NN structure due to less parameters affecting the cost performance for each NN. Thus it is less difficult for each NN to find the optimal values for the gains.

Obviously during an experiment the set-point x_{1ref} keeps constant until the crane completes the task. This also assures the convergence of the NNs. The training of the network is done on-line. At each stage of the process the NNs are trained to find the values for k_1, k_2 and k_3 that minimize a pre-specified cost function. This function is generally chosen as quadratic. It is proposed to the students to be in the following form:

$$J(k) = \tfrac{1}{2}p_1(k)(x_{1ref} - x_1(k))^2 + \tfrac{1}{2}p_2(k)x_2^2(k) + \tfrac{1}{2}p_3(k)x_3^2(k) \tag{16}$$

where $p_i(k)$ are time-dependent functions that weight each variable in the cost function. Depending on the position of the crab the $p_i(k)$ variables will take the appropriate value.

The computation of the gradient is done according to

$$\frac{\partial J(k)}{\partial w_{L,j,i}} = \frac{\partial J(k)}{\partial k_j}\frac{\partial k_j}{\partial w_{L,j,i}} \tag{17}$$

The first factor of this expression is:

$$\frac{\partial J(k)}{\partial k_j} = \left(-p_1(k)e_1(k)\frac{\partial x_1(k)}{\partial u(k)} + p_2(k)x_2(k)\frac{\partial x_2(k)}{\partial u(k)} + p_3(k)x_3(k)\frac{\partial x_3(k)}{\partial u(k)}\right)\frac{\partial u(k)}{\partial k_j} \tag{18}$$

The factors $\partial x_i(k)/\partial u(k)$ are obtained immediately if the model of the plant is known. Avoiding any assumption about the plant model, these terms can be obtained by difference approximation:

$$\frac{\partial x_i(k)}{\partial u(k)} \approx \frac{x_i(k) - x_i(k-1)}{u(k-1) - u(k-2)} \tag{19}$$

The factors $\partial u(k)/\partial k_j$ are calculated from the expression (15) for the controller:

$$\frac{\partial u(k)}{\partial k_1} = e_1(k); \quad \frac{\partial u(k)}{\partial k_2} = -x_2(k); \quad \frac{\partial u(k)}{\partial k_3} = -x_3(k). \tag{20}$$

Starting from the instant k, the algorithm can be summarized as follows:

Step 1) *READ* $x_1(k)$, $x_2(k)$, $x_3(k)$ *and* $x_4(k)$.
Step 2) (Networks training)
NN1 training:

$$\frac{\partial u(k)}{\partial k_1} \leftarrow e_1(k); \quad \frac{\partial u(k)}{\partial k_2} \leftarrow -x_2(k)$$

Backpropagation

$$\left(\frac{\partial u(k)}{\partial k_1}, \frac{\partial u(k)}{\partial k_2}; e_1(k), \dot{e}_1(k); p_i(k)|_{i=1,\ldots,4}\right);$$

UPDATE k_1 and k_2
NN2 training:

$$\frac{\partial u(k)}{\partial k_3} \leftarrow -x_3(k)$$

Backpropagation

$$\left(\frac{\partial u(k)}{\partial k_3}; x_3(k), x_4(k); p_i(k)|_{i=1,\ldots,4}\right);$$

UPDATE k_3.
Step 3) COMPUTE command

$$u(k) = -(k_1 \quad k_2 \quad k_3)(-e_1(k) \quad x_2(k) \quad x_3(k))^T.$$

Step 4) APPLY $u(k)$ and WAIT until $t = (k+1)T$.
Step 5) *k k+1.*
Step 6) GOTO step 1.

Note that, although NN1 provides the value for k_1 and k_2 and NN2 provides k_3, both NNs use the same cost function (16). This is why $p_i(k)|_{i=1,\ldots,4}$ is used in both NNs. This algorithm finds the optimal values for the gains of the controller. For a fixed parameter plant, the gains of the adaptive controller will converge to a set of values that minimizes the performance index. However, if plant dynamics vary, the gains supplied by the NNs change adaptively to a new set of values.

Due to the complexity of this algorithm, the experiments will consist of two sessions. Firstly, there is a class in the computer room where they are asked to implement the same control structure on a SISO linear plant. To do this, they are provided with MATLAB® and the Neural Networks MATLAB® Toolbox. Then students have the opportunity to implement the controller on the crane prototype. The real-time experiments are carried out using the software package NNCYC developed by the Computer and Control Group of the University of La Laguna [9]. This software allows the definition of the NN structure and the weights (p_i) of the cost function.

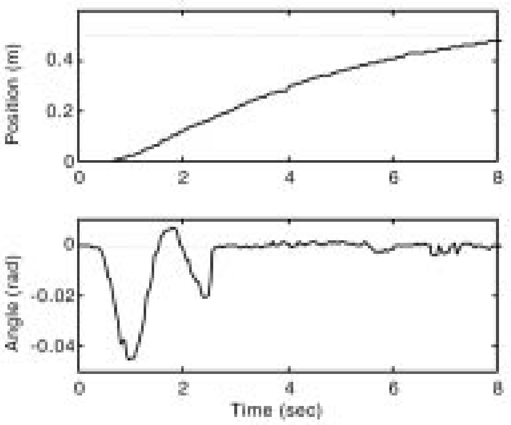

Fig. 8. Real-time implementation on the crane prototype. Evolution of the x_1 and x_3 variable.

The results obtained in one of the experiments are presented in Fig. 8. This experiment has been performed after several training trials with the crane. The gain k_4 has been fixed to the value zero. In other trials, students could vary this value in order to get better performance from the controller. The cost parameters chosen are $p_1(k) = 20$, $p_2(k) = 0.2$ and $p_3(k) = 0.1$. The results obtained seem entirely satisfactory, since both control objectives in the load angle and in the cart position are attained. Observe that, although the system achieves the control objective, the evolution is too slow. Note that in the previous experiment the settling time is about 4 sec and now this time is about 8 sec. The reason for this is the transient evolution of the gains until their convergence is achieved.

As an alternative to the SVF controller, a controller based on fuzzy logic is proposed. The advantage of fuzzy controllers is that the model of the system is not necessary in the design of the controller. Thus it is expected that a fuzzy controller would have better performance than a model-based controller. In particular, if the linear model is not available or is not accurate (because angles are not small enough), the efficiency of the SVF controller decreases, while the fuzzy controller presents a good performance. A very simple controller based on linguistic rules is presented. The rules are obtained on the basis of a human expert's knowledge. This reduces the complexity of

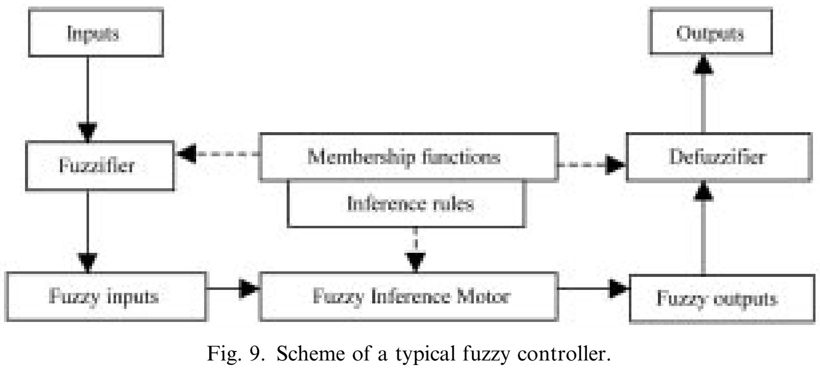

Fig. 9. Scheme of a typical fuzzy controller.

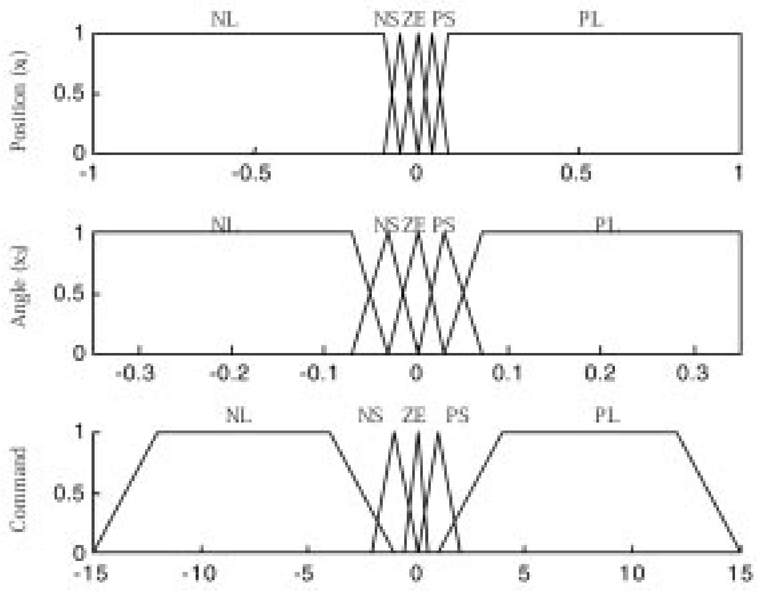

Fig. 10. Membership functions of the defined fuzzy variables.

the control algorithms. A comparison between these results and the preceding ones is proposed.

Crane control based on fuzzy logic

Students are asked to design a fuzzy controller. For this experiment five hours in the laboratory will be needed. The typical scheme of a fuzzy control system is shown in Fig. 9. The fuzzy numbers of each input and output variable and the rules that define the control action must be specified. These sets and rules are proposed to be defined intuitively by the students, once they have studied, in the previous experiments, the behaviour of the system. For this purpose, visual fuzzy software is used. This program allows easy definition of the fuzzy sets and the inference rules. These fuzzy sets and the results of the control action are graphically represented. Mamdani's method, previously proposed, is used in its discrete form, computing the output of the controller from the inputs read in the converter board.

One of the possible choices of the fuzzy variables and the inference rules that offers acceptable results is presented below. The chosen input variables to the inference motor are the error in the position ($e_1 = x_{1ref} - x_1$) and the angle of the load (θ). The output variable is the command applied to the crane (u).

Every fuzzy variable has five sets defined in their universe of discourse:

PL:	Positive Large
PS:	Positive Small
ZE:	Almost Zero
NS:	Negative Small
NL:	Negative Large

The membership functions are represented in Fig. 10. The rules of the inference motor are defined according to the following considerations:

- There is no action on the angle until the position is near the set point.
- The oscillations of the load are corrected by aligning the crab vertically with the load.
- Once the angle is corrected, the crab is taken to the set point with a small command.

With these guidelines the resultant rule table can be seen in Fig. 11. The algorithm used for the implementation of the controller can be summarized as follows:

Step 1) DEFINITION of the fuzzy subsets for the variables and inference rules.
Step 2) READ the inputs $x_1(k)$, $x_3(k)$.
Step 3) FUZZIFICATION of the input variables.
Step 4) APPLY the inference motor rules to get the output variable.

	θ				
e_1	NL	NS	ZE	PS	PL
NL	PL	PL	PL	PL	PL
NS	NL	NS	PS	PS	PL
ZE	NL	NS	ZE	PS	PL
PS	NL	NS	NS	PS	PL
PL	NL	NL	NL	NL	NL

Fig. 11. Rule table of the crane fuzzy controller.

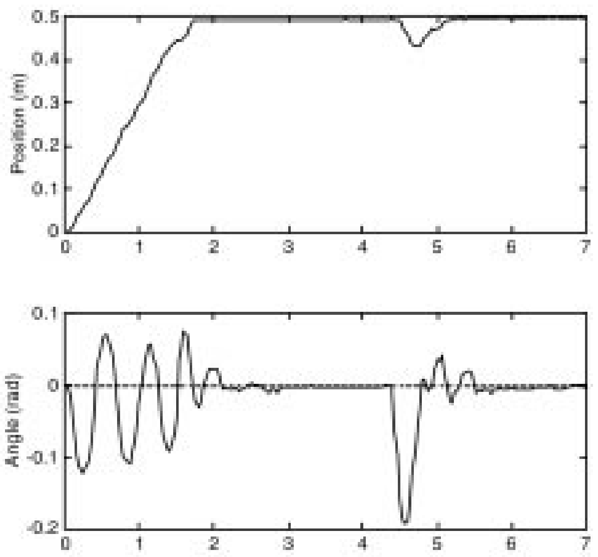

Fig. 12. Evolution of the position and the angle of the crane applying the fuzzy controller. A perturbation is applied to the load mass once the set point, situated in 0.5 m, is attained.

Step 5) DEFUZZIFICATION of the output variable.
Step 6) APPLY $u(k)$ and WAIT until $t=(k+1)T$.
Step 7) $k \leftarrow k+1$.
Step 8) GOTO step 2.

The results obtained in one of the experiments are shown in Fig. 12. Starting from any initial error, the set point is attained smoothly and with zero offset. There is no overshooting and the settling time is about 3 sec. The response to a perturbation applied on the load mass once the set point is attained can also be seen. The fuzzy controller provides the necessary commands to correct the angle and to reach the target position.

With a thinner choice of the triangular sets around the final target values, the closed-loop system is unstable. This is because small variations in the variables will produce large variations in the fuzzified variables. Thus, the resultant command will change too brusquely, causing instability.

An advantage of this kind of controller is its robust response to changes on the system parameters (mass of the crab, mass of the load or length of the rope). Moreover, even choosing the rules and the subsets intuitively, on the basis of the known performance of the system in the previous experiments, a good response of the controller was obtained.

Comparing these results and the previous ones, it can be seen that the oscillations of the load mass are cancelled in a shorter time than in the preceding experiments. This is because the controller first tries to correct the load oscillations once the crab is near the set point. Due to this fact, the variations on the applied command are higher than in the preceding experiments. In experiment 2 the command takes the maximum value to attain the desired crab position. Nearing this, it slowly decreases, suppressing the load oscillations. The command evolution in experiment 3 is very soft, but the position of the crab is attained after a longer time. With the fuzzy controller, the command evolves with sharper changes, but the angle is corrected earlier.

Due to the different values of the applied command, the amplitude of the oscillations varies for experiments with different controllers: high values of angles are obtained in experiments 2 and 4, while in experiment 3 these values decrease by roughly half. It is worth pointing out that, with the fuzzy controller, if there were later load oscillations, the system would efficiently and quickly retrieve the set point.

CONCLUSIONS

In this paper, a digital control laboratory has been proposed. The laboratory is oriented to digital control and intelligent control courses. The objective of the experiments proposed is, on the one hand, to reinforce the theoretical concepts seen in the lectures and, on the other hand, to evaluate and compare the performance of both classical and intelligent techniques in a specific control problem. An important advantage of this laboratory is that the plant on which the experiments are carried out is a relatively low-cost plant. Furthermore, the presence of this kind of system in industry is very widespread, and the associated control problem is still a matter of study for researchers. This makes the laboratory more attractive for students. Although they consider that these subjects are not easy because of the complexity of some of the approaches studied, they are now more motivated and enthusiastic than earlier generations that took the laboratory by simulation.

Acknowledgement—This work is supported by the project 'Diseño e implementación de un Robot Autónomo de Servicio, aplicación en tareas de restauración', financed by the Spanish Government.

REFERENCES

1. L. Moreno, L. Acosta, A. Hamilton, J. D. Piñeiro, J. J. Merino, J. L. Sánchez and R. M. Aguilar, Experiments on a DC motor based system digital control course, *IJEEE*, **32** (1995), pp. 163–178.
2. L. Moreno, L. Acosta, J. A. Méndez, A. Hamilton, J. D. Piñeiro and J. L. Sánchez, Stochastic optimal controllers for a DC servo motor: Applicability and performance, *Control Engineering Practice*, **4**(6) (1996), pp. 757–764.
3. C. Cheng and C. Chen, Controller design for an overhead crane system with uncertainty, *Control Engineering Practice*, **4**(5) (1996), pp. 645–653.
4. T. Munkata and Y. Jani, Fuzzy systems: An overview, *Communications of ACM*, **37** (1994), pp. 69–76.
5. D. R. Hush and B. Horne, Progress in supervised neural networks, *IEEE Signal Processing Magazine* (1993), pp. 12–14.
6. W. Pedrycz, *Fuzzy Control and Fuzzy Systems*, Wiley, New York (1989).
7. J. A. Méndez, L. Acosta, L. Moreno, A. Hamilton and G. N. Marichal, Design of a neural network based self-tuning controller for an overhead crane, *IEEE Conference on Control Applications*, Trieste, Italy (1998), pp. 168–171.
8. K. J. Aström and B. Wittenmark, *Computer-Controlled Systems*, Prentice-Hall Information and System Sciences Series (1997).
9. J. A. Méndez, Desarrollo de estrategias de control predictivas de alto rendimiento mediante la incorporación de técnicas de control robusto y de redes neuronales, Ph.D. thesis, Universidad de La Laguna, Tenerife, Spain (1998).

Lorenzo Moreno received M.Sc. and Ph.D. degrees from the Universidad Complutense de Madrid, Spain, in 1973 and 1977 respectively. From 1977 to 1979, he was an Associate Professor in the Department of Computer and Control Engineering, Universidad de Bilbao, Spain. From 1979 to 1988, he was an Associate Professor in the Department of Computer Science, Universidad Autónoma de Barcelona, Spain. He is presently a Professor in the Applied Physics Department at the Universidad de La Laguna, Tenerife, Spain. His areas of interest include control, signal processing, simulation and artificial intelligence.

Santiago Torres graduated in applied physics in 1997 and received his M.Sc. degree in applied physics in 1999 from the Universidad de La Laguna, Tenerife, Spain. He graduated in electronic engineering in 2004. He is an Associate Professor in the Applied Physics Department at the same university. His areas of interest include engineering control and robotics.

Juan A. Méndez graduated in applied physics in 1992. He received his M.Sc. degree in applied physics in 1993 and his Ph.D. degree in computer science in 1998 from the Universidad de La Laguna, Tenerife, Spain. He is a Professor in the Applied Physics Department at the same university. His areas of interest include predictive control and artificial intelligence applied to control.

Leopoldo Acosta received his M.Sc. degree in applied physics in 1987 and his Ph.D. degree in computer science in 1991 from the Universidad de La Laguna, Tenerife, Spain. He is a Professor in the Applied Physics Department at the same university. His areas of interest include robotics, control and artificial intelligence.

Marta Sigut graduated in applied physics in 1996 from the Universidad de La Laguna, Tenerife, Spain, and received an M.Sc. in applied physics in 1998 and a Ph.D. in computer science in 2003 from the Universidad de La Laguna, Tenerife, Spain. She is an Associate Professor in the Applied Physics Department at the same university. Her areas of interest include control of large-scale systems.