Int. J. Engng Ed. Vol. 21, No. 6, pp. 1059–1067, 2005
Printed in Great Britain.

0949-149X/91 $3.00+0.00
© 2005 TEMPUS Publications.

Virtual Laboratories for Control Education: a Combined Methodology*

CÉSAR FERNÁNDEZ, MARÍA ASUNCIÓN VICENTE and LUIS MIGUEL JIMÉNEZ
Universidad Miguel Hernández. Systems Engineering and Automation Division, University Miguel Hernandez,, Elche, Spain. E-mail: c.fernandez@umh.es

A methodology for control education is presented, focused in student motivation and making use of a simulation environment. The goal is to increase the number of practice sessions, making them attractive to the student and avoiding costly laboratory equipment. As a difference to other approaches like virtual laboratories, the proposed methodology is based on a combination of computer sessions and laboratory sessions, which should be complementary. The main idea is to simulate some interesting, well chosen real system in a preliminary computer session; and then performing a practical experiment with a simple low-cost laboratory equipment working under the same physical principles. This methodology reduces costs, multiplies the number of different experiments, and allows the student to manipulate and control real physical systems. The MATLAB environment is used for the development of the simulation programs. The structure followed allows an easy development of new programs, by keeping independent the user interface, the process simulation and the graphical representation modules. Two examples of simulation programs (linear and nonlinear systems) are given, and full source code is available for both of them. The methodology is being used at present at the Industrial Systems Engineering Department of the Miguel Hernandez University for two introductory subjects of control theory.

INTRODUCTION

SUBJECTS RELATED to systems and control theory in engineering degrees include strong mathematical foundations and student motivation is very difficult, particularly considering that the students have chosen a technical degree [1, 2]. This problem can be immediately solved by programming a high number of practical sessions, so that the student can apply the theoretical foundations to physical systems. This solution is very costly and usually unaffordable, particularly in control education, where physical equipments are very expensive (process control scale models, robots, etc.).

Different approaches have been proposed to solve such problems, among them virtual laboratories [3] and remote access to laboratory equipment through the Internet [4, 5], but new problems arise. Virtual laboratories are mere simulations, and the student wonders whether the control schemes that control a virtual device would be applicable to actually controlling a physical device. Besides, virtual practices may be considered by the student as a sort of computer-aided theoretical session, and not as practices. On the other hand, remote access to laboratory equipment also presents problems: students cannot work simultaneously and not all the laboratory equipments can be remote controlled.

In order to avoid these problems, a two-step methodology is proposed: first, an interesting, well-chosen real system is simulated in a computer session, and then a practical experiment is carried out with a simple low-cost laboratory equipment. This methodology reduces costs, multiplies the number of different experiments, and allows the student to manipulate and perform the closed loop control of real physical systems.

The system to be simulated has to be chosen specifically according to the equipment available at the laboratory. The goal is to find a real-life system, complex and interesting but based on the same physical principles that the simple, low-cost laboratory equipment. In this way, the student can relate the behaviour of the real-life simulated system with that of the laboratory device, even if this device is extremely simple. Both practice sessions benefit from each other: the computer session by itself is not enough as there is a lack of interaction with a real device; and the laboratory session by itself is not enough as the devices are usually too simple, and it is hard for the student to relate the work performed in the laboratory with a real-life control problem.

Taking into account the drawbacks of previous approaches, some aspects—mainly regarding the simulation environment—have been considered relevant in order to be able to successfully apply the proposed methodology:

- The simulation should allow to be run both at the university computer rooms and from the student's home computer.
- The student should not need a wideband connection and, more important, the student should not need to be connected during all the simulation experiments that can take a very long time.

* Accepted 26 July 2005.

- A graphical animation of the results is required: simulations that represent the results by simple plots are not enough to motivate the students. Moreover, the animation should run in real-time (1 s of animation should equal 1 s of system behaviour).
- Docents may be able to develop new simulation environments easily in order to create environments suited to the equipment available at their laboratories.

In this paper, a generic structure for the development of simulation programs covering the previous aspects is presented. The process of creating new programs following this structure is fast and easy, and some examples are provided. The Matlab environment [6] has been chosen due to its widespread use in universities, particularly in subjects related to control theory, where textbooks usually include MATLAB examples [7, 8]. Anyway, other similar tools could be used instead.

DESCRIPTION OF THE PLANT

The goal of the student in the proposed simulation sessions is to tune or design a control algorithm for a certain system or device (in control terms, such device is referred to as a *plant* [9]). It becomes necessary to describe the behaviour of such plant, and there are different possibilities, among them the definition via differential equations, via transfer functions, or via block diagrams, where each block may be linear or not (examples of nonlinear behaviours are very common in plants: delays, saturations, etc. [9, 10]).

In order to cover such a variety of possible representations, the following strategy is used:

- When all elements describing the system are linear, the behaviour should be expressed as a single transfer function or as a block diagram whose blocks are transfer functions (it is straightforward to reduce the diagram using Matlab functions). Some off-line processing of the initial representation may be necessary if it is originally expressed as a series of differential equations, namely obtaining the equilibrium point and transforming to the Laplace domain [9]. Such representation can be directly handled by Matlab (responses to step inputs or to arbitrary inputs can be obtained easily) and no extra tools are required. This is the simplest case, and an example is described below.
- When there are nonlinear elements, it is not possible to obtain a transfer function (linear approximations can be obtained but they may not be accurate [9]) and so MATLAB functions cannot be used directly to obtain the responses. For these examples a Simulink block diagram needs to be constructed, in which the different elements may be expressed in the time domain, in the Laplace domain or may even use hybrid representations. In order to present a uniform user interface for all the simulation environments, Simulink is invoked in the background from the MATLAB script. An example simulation program is described later.

SIMULATION PROGRAM STRUCTURE

The proposed structure for the simulation program is shown in the diagram of Fig. 1. In brief, the first steps of the simulation program are devoted to gathering data from the user. For this purpose, a user interface can be created easily by using MATLAB GUI functions. The examples provided in this paper can be used as a guide for the development of new programs.

The user interface starts asking the student for the experiment to be performed. In the examples presented, the experiments include the simulation of responses to constant inputs, step inputs, random inputs (noise or disturbances) and to previously defined input signals. Other experiments can be defined similarly. Once the experiment is selected and, depending on the particular kind of experiment, some adjustments may be necessary (step amplitude, disturbance range, etc).

When the experiment to be performed is fully described, it is the time for the student to propose a solution for the control problem. This solution may represent the tuning of a certain predefined controller (e.g. the selection of proportional, derivative and integral effects for a PID controller) or the complete design of a controller starting from scratch (e.g. the definition of the controller transfer function). Once all these data have been input to the system, the simulation starts.

As it has been mentioned earlier, depending on the linearity of the system description, MATLAB or Simulink tools are used for the simulation. The behaviour of the system (plant + controller) is simulated off-line during a certain adjustable amount of time, which may vary from seconds to minutes. (As the animation of the results will run in real-time, longer simulations are not considered.) In the examples shown in this paper, the off-line processing takes less than 0.1 s for a 30-s system simulation in a standard PC; however, more complex systems may increase this processing time.

When the simulation results have been obtained, the real-time animation of results starts. Decoupling the simulation and representation steps is a key factor for the easy development of simulation programs. The graphical representation of results is performed via MATLAB graphic functions, and real-time representations (1 s of representation equals 1 s of system behaviour) can be easily accomplished by the use of MATLAB clock functions. The listing below shows the representation algorithm in pseudo-code (it can be easily implemented using any programming language) and the source code available with the examples presented in this paper shows a full Matlab implementation.

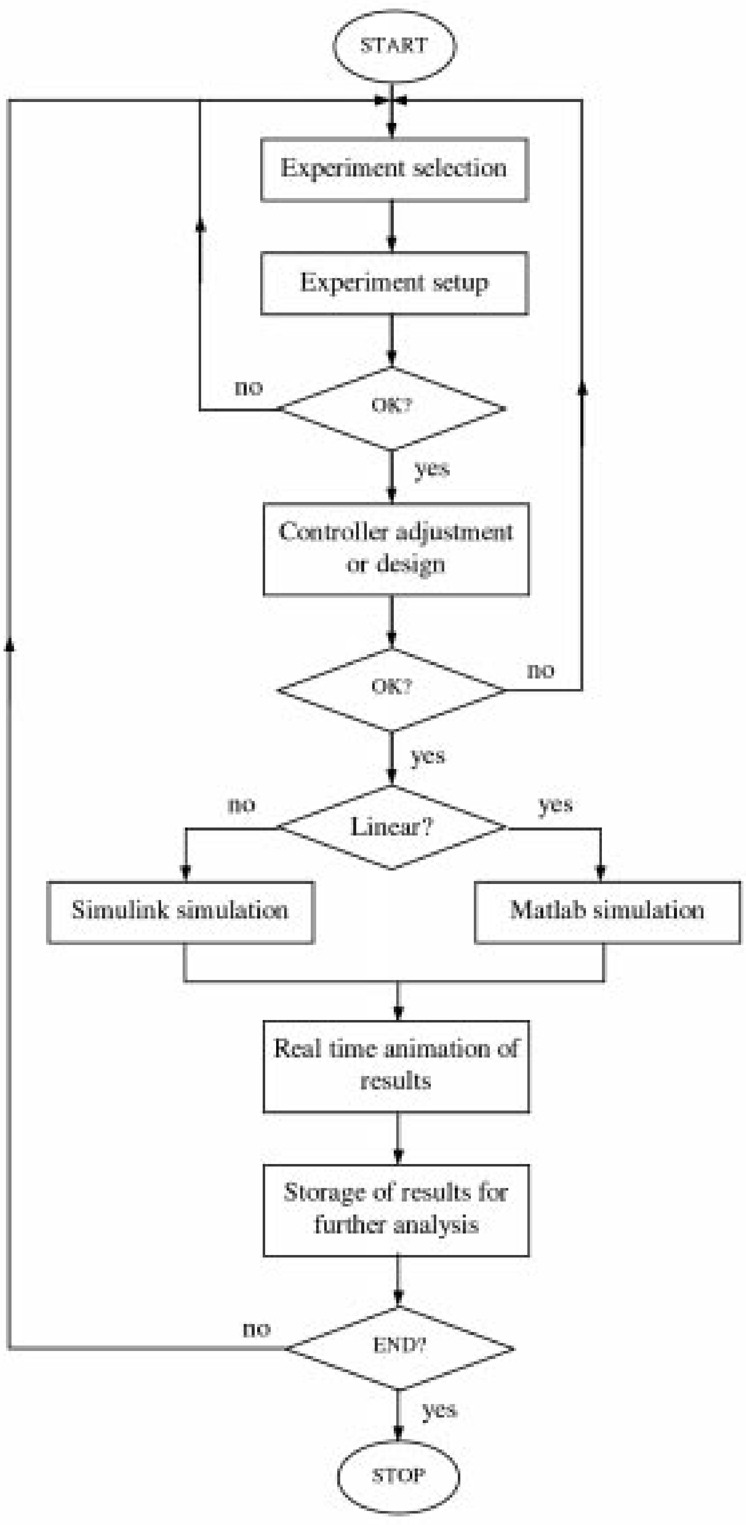


Fig. 1. Proposed structure for simulation programs.

```
SimulationTime = ReadTime()
while SimulationTime < TotalTime
  InputSignal = ReadInput(SimulationTime)
  OutputSignal = ReadOutput(SimulationTime)
  AnimateDrawing(InputSignal,      Output-
    Signal)
  SimulationTime = ReadTime()
```

In the pseudo-code shown, ReadTime() is supposed to be a function capable of accessing the system clock and measuring the time elapsed since the beginning of the animation; and AnimateDrawing() is supposed to be the function in charge of updating the display with a new graphical representation of results.

The last step in the structure of the simulation program is the storage of results. All input and output signals are stored, thus allowing the student to analyse the results of the simulation after the graphical animation has ended. In the examples provided, the results are stored as MATLAB variables, so the students can easily plot or process them.

LINEAR EXAMPLE: MAGNETIC LEVITATION TRAIN

The first example presented deals with the closed loop control of magnetic levitation systems. It was decided to build a simulation environment to make the magnetic levitation practice more attractive to the students. The equipment available at the lab is a low-cost device from Extra Dimension Technologies [11], which is shown in Fig. 2.

The magnetic levitation practice should be interesting for control students, as the system is inherently unstable (it is based on magnetic attraction, magnetic repulsion systems are stable [12]) and a closed loop control system is required in order to make it work. The purpose of the practice is to tune a PID controller for this plant. The controller measures the height and reacts to increasing or decreasing the current through the electromagnet, until the reference height is reached. However, when actually carrying out the laboratory session, students really don't show a high interest for this experiment. The reason can be found in the simplicity of the device.

In order to make the practice more interesting to the students, a simulation environment has been developed. This simulation environment shows an application of the same physical principle: magnetic levitation trains. Such trains represent an example of a system the students may find interesting to work with: they are one of the latest technologies for transportation. A magnetic levitation train prototype is shown in Fig. 3.

In order to make the simulated system as similar as possible to the existing laboratory device, a magnetic attraction system is simulated. The goal for the student is to keep the train height stable (the gap existing between the rail and the train) by adjusting a PID controller. An accident may occur if the gap exceeds certain limits: the train may contact the guideways. Some screenshots of the simulation environment are shown in Figs 5 and 6.

In this example, the student may perform two different experiments: the simulation of the step response of the train (i.e. the reference height or gap is modified and the train should react to that input) and the simulation of the noise response of the system (the noise level is adjustable). The train behaviour has been simulated via transfer functions as the system can be considered approximately linear when the gap varies in a narrow range.

Once the simulation is finished, the results may be stored as MATLAB variables. Such variables can be used by the student to perform further analysis or plots, in order to check the validity of the control system he/she has designed. As an example, Fig. 6 shows plots of the output signals (gap between track and train) for correct vs. incorrect configurations of the controller in a step response experiment. Obviously, the second plot corresponds to an unstable system. More details concerning this simulation environment can be found in our previous work [13].

The simulation environment consists of a single Matlab function, which is available for download at [14].

To be able to run the simulation, it is only necessary to download the program or to copy it from a CD or diskette (the extremely small size of the programs should be noted: 18 kB in this example) and execute it on a computer running MATLAB. The simulation runs locally, so this methodology makes the simulations available to all students, even if they don't have wideband Internet connections at home (even if they don't have Internet at all).

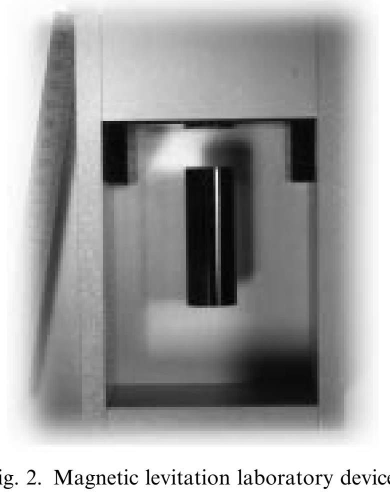


Fig. 2. Magnetic levitation laboratory device.

Fig. 3. Magnetic levitation train MLU002N (Japan).

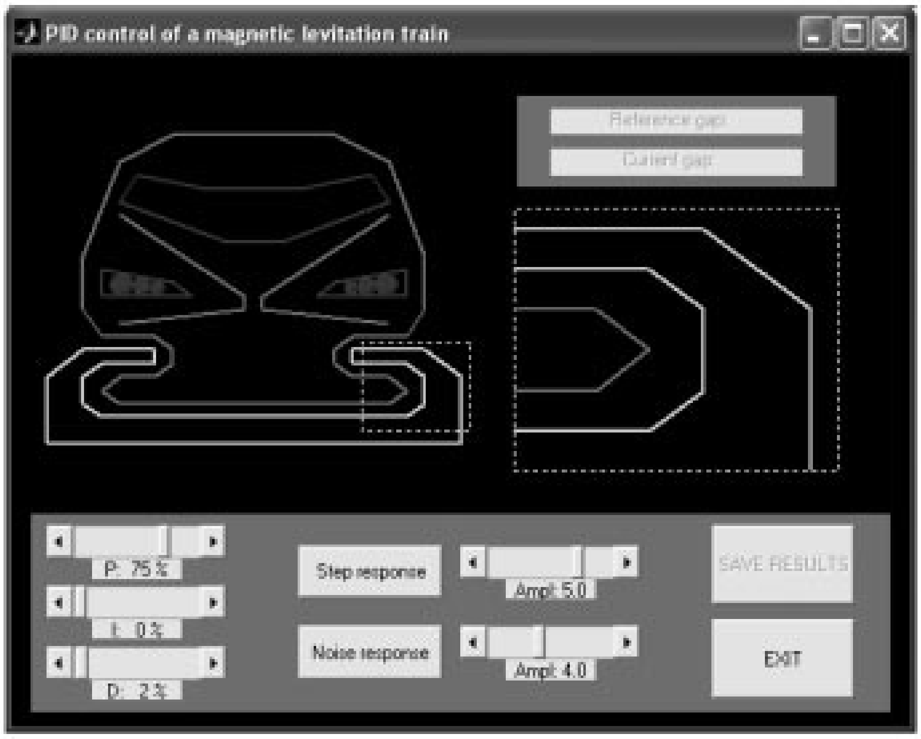
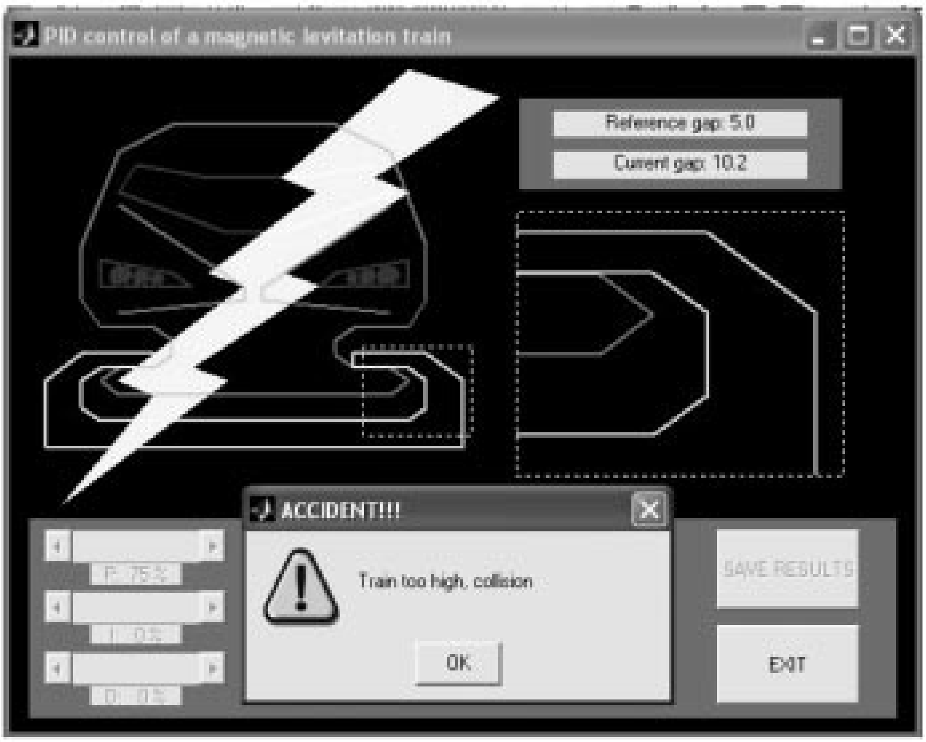


Fig. 4. Maglev simulation: main screen.

NONLINEAR EXAMPLE: CAR CRUISE CONTROL

The second example presented deals with the closed-loop speed control of a DC motor. It is a common practice session in control education: a controller is designed in order to adjust the speed of a DC electric motor. Concretely, the equipment available at the lab is one of the most common devices in most universities: the servomotor model 33-001 from Feedback [15] which can be seen in Fig. 7.



Reference gap: 5.0

Current gap: 10.2

ACCIDENT!!!

Train too high, collision

Fig. 5. Maglev simulation: bad adjustment of controller.

As controlling the speed of a DC servomotor is not a challenging task for the students, it has been decided to create also a simulation environment for this device. In this case, and in order to make the practice more attractive, the simulation does not consider an electric motor but a combustion engine: the purpose is to design an algorithm for a car cruise control, like those available in high-end cars. Even though the system is quite different, the basic principle is the same—speed control—and there is an important common problem: both the electric motor speed control and the car cruise control present stationary error unless an integral effect is included in the controller [9]. Thus, the same principle can be applied to the simulation and to the practice session with the lab equipment.

The goal for the student is to design a controller capable of keeping stable the car speed, even if the slope of the road is not uniform. The controller is supposed to be continuously measuring the speed of the car, and should react to differences with the reference speed increasing or decreasing the fuel injected to the engine (adjusting the gas pedal of the car).

A screenshot of the simulation environment is shown in Fig. 8. It can be seen that both the slope of the road and the instantaneous speed are animated during the real-time representation of results.

In this example, the behaviour of the car is highly nonlinear due to two reasons: first, the longitudinal component of the weight is related to the road slope through a sine function; and second, the resistance due to air friction is supposed to have a quadratic relationship with the speed of the car. (Even considering such nonlinearities, the car behaviour has been simplified to a high extent, as the purpose of the simulator is not to obtain accurate experimental data.) As it has been explained before, the presence of nonlinearities makes it necessary to simulate the behaviour of the system via a Simulink block diagram. Fig. 9 shows the diagram used in this example.

The student, once the controller has been

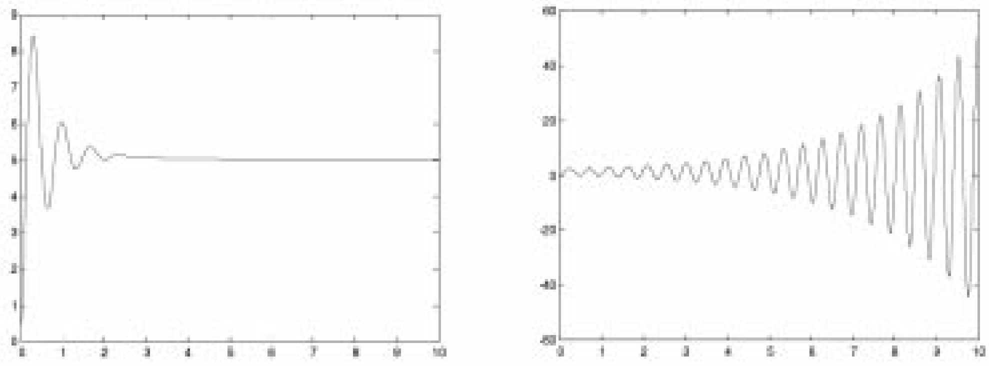


Fig. 6. Output plots for correct vs. incorrect configuration of the controller.

Fig. 7. DC servomotor model 33-001 from Feedback.

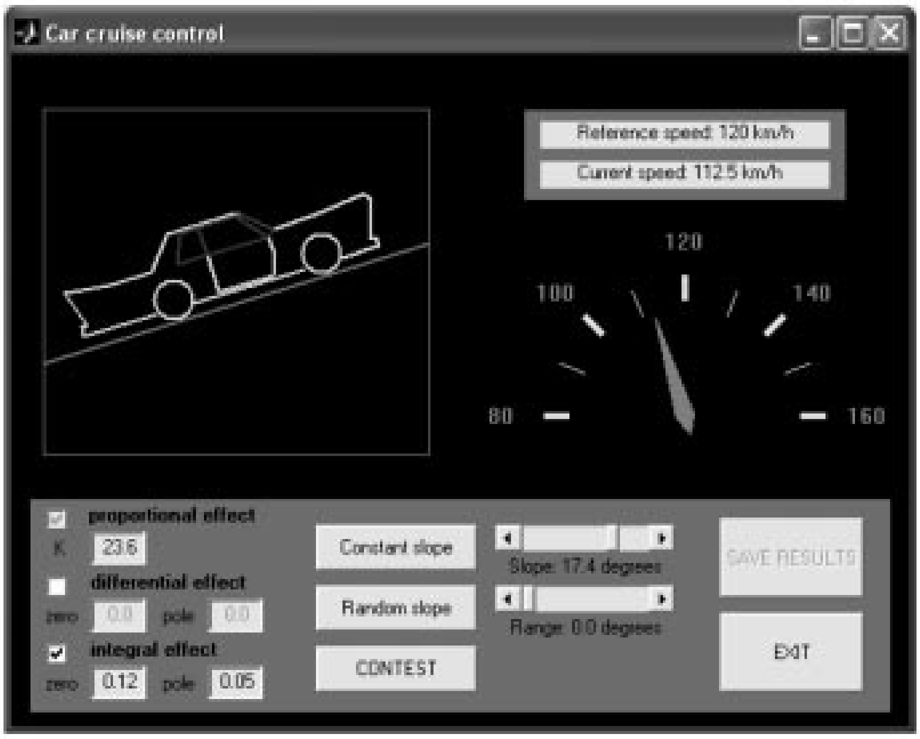


Fig. 8. Cruise control simulator: main screen.

adjusted, can perform three different experiments with this simulator:

- *Response to a constant slope.* This experiment shows whether the controller works properly under unchanging road conditions.
- *Response to a randomly varying slope.* This experiment shows the behaviour of the system when the car goes through a previously unknown road.
- *Contest.* This experiment is similar to the previous one but with a different purpose: the students should obtain the best possible behaviour for the car and then compare their results with those of other students. For this purpose, the slope does not vary randomly but under a previously fixed profile, which is the same for all the students.

Once the experiment is finished, it is possible to save the results to a file and perform further analysis, as in the previous example. Particularly, when the contest experiment is performed, it is interesting to check the behaviour of the system by plotting the input (slope) and output (speed) signals. Such data is represented in Fig. 10.

Considering the contest experiment, a comparison can be made among the results obtained by the different students, using different metrics. One of these metrics could be the integral of the absolute deviations from the reference speed; other metrics could be related to the maximum absolute value of the deviations; others may work in a different way with upper or lower deviations, etc.

For this example (and all nonlinear ones) the simulation environment consists of two files: a

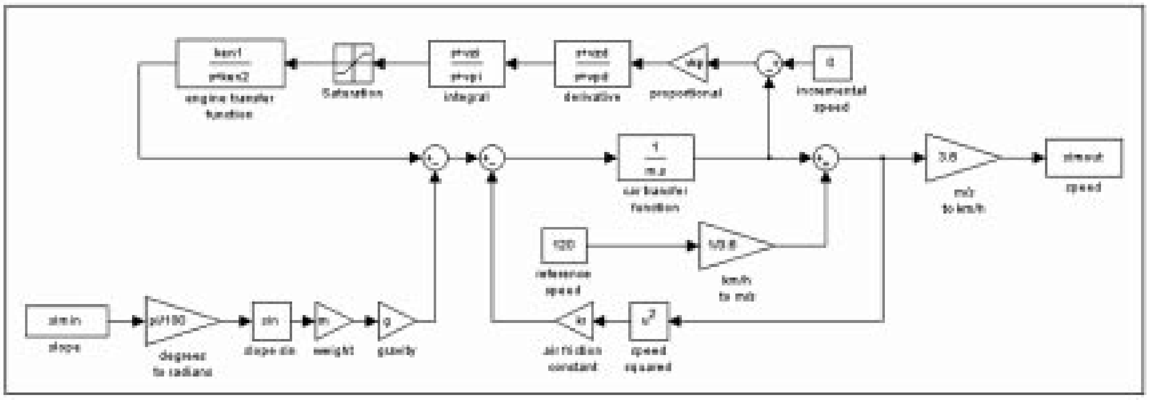


Fig. 9. Cruise control: Simulink block diagram.

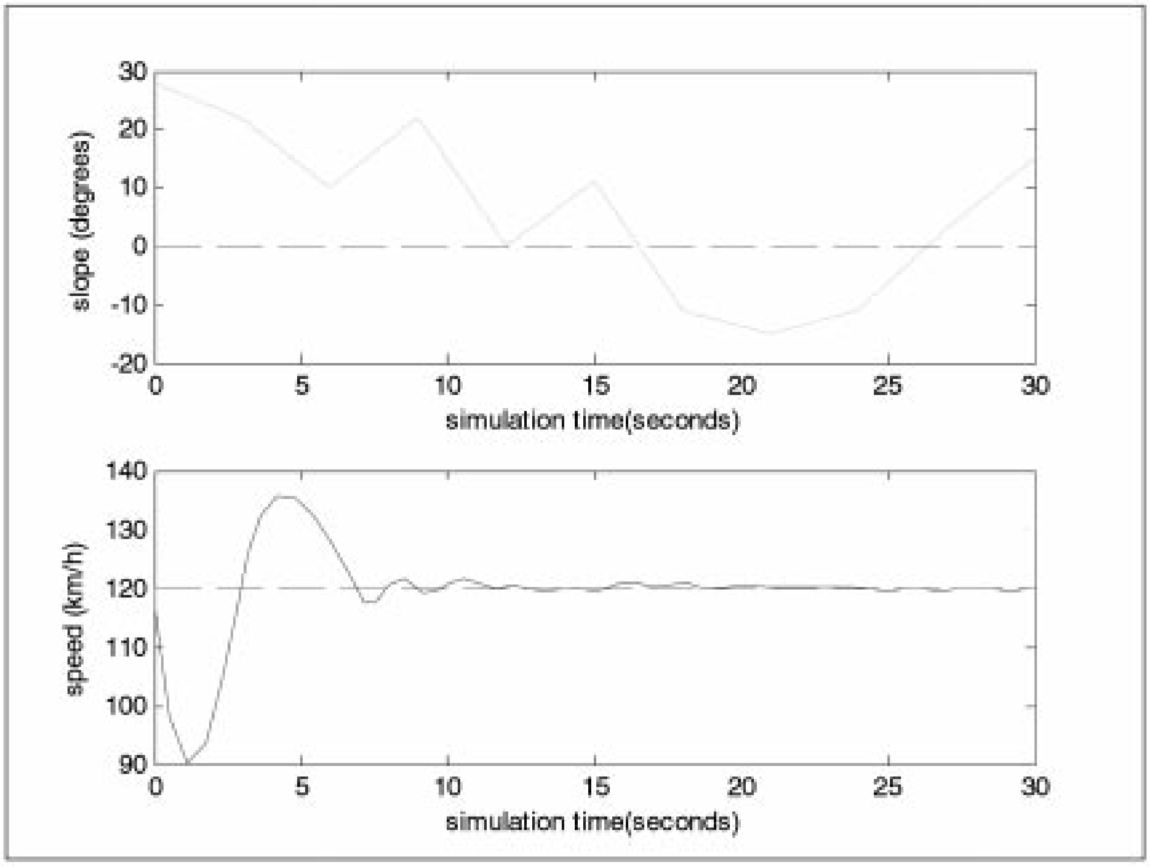


Fig. 10. Results of the contest experiment.

MATLAB script and a Simulink block diagram. These files are available for download at [16] and [17].

As in the previous example, it is only necessary to download both files to perform the simulation on a computer running MATLAB, and the program will run locally.

CONCLUSIONS

Student motivation in control engineering subjects can only be accomplished by increasing the number of practical sessions, but laboratory costs become unaffordable. Web-shared equipment or virtual laboratories help in reducing costs but present serious drawbacks.

A two-step methodology is proposed, performing a previous simulation session before each laboratory session, thus increasing student motivation and fully exploiting the equipment available at any laboratory.

The structure proposed for the development of simulation environments is based in MATLAB, and allows a real-time animated representation of results without the need for real-time toolboxes. Apart from that, the requirements (both in terms of computing power and connection bandwidth) are kept to a minimum, thus enabling the students to run the simulation from their homes.

The two examples presented in this paper (linear and nonlinear simulations) can be used by other students as a guide to developing new simulation environments using a similar interface, but adapted to other laboratory equipment.

Two main benefits are obtained by using the proposed methodology: first, the simulation sessions help to increas student motivation, as control theory is applied to solve interesting real-life problems; and second, the practice sessions with laboratory equipment are better understood by the students as they have worked with similar (simulated) devices in the previous session.

REFERENCES

1. K. J. Aström, Reflections on control engineering education, *Jornadas de Automática,* **IX**,1998.
2. R. Puerto, C. Fernandez, Training exercises on engineering courses: a practical application on digital signal filters, *10th. EAEEIE Conf. Educational Innovations in EIE,* 1999.

3. F. A. Candelas, S. T. Puente, F. Torres, F. G. Ortiz and P. Gil, Educational virtual laboratory for training of robotics, *Int. J. Eng. Educ.,* **19**(3), 2003, pp. 363–370.
4. R. Puerto, L. M. Jiménez, O. Reinoso and C. Fernández, Laboratorio vía Internet para el control de procesos*,* Ac*tas de la Conferencia Internacional sobre Educación, Formación y Nuevas Tecnologías,* 2002.
5. J. M. Sebastián, F. M. Sánchez and D. García, Sivanet: A new remote physical scenario for control self-learning through the Internet, *Learning Technology,* IEEE Computer Society (2002).
6. www.mathworks.com/
7. N. S. Sise, *Control Systems Engineering,* Wiley (2003).
8. K. Ogata, *Solving Control Engineering Problems with Matlab,* Prentice-Hall (1993).
9. K. Ogata, *Modern Control Engineering,* Prentice-Hall (2001).
10. B. C. Kuo, *Automatic Control Systems,* Wiley (2003).
11. www.xdtech.com/Business/productsbusiness/productcatalogbusiness/ productpagebusiness/ml.htm
12. http://www.teicontrols.com/notes/SeminarEE155/MaglevVehicles.pdf
13. C. Fernandez, M. A. Vicente and R. Puerto, Enseñanza en asignaturas de control apoyada en equipos experimentales virtuales, *VI Congreso de Tecnologías Aplicadas a la Enseñanza de la Electrónica,* 2004.
14. http://lorca.umh.es/isa/es/asignaturas/tcs/practicas/matlab/train.m
15. http://www.fbk.com
16. http://lorca.umh.es/isa/es/asignaturas/tcs/practicas/matlab/car.m
17. http://lorca.umh.es/isa/es/asignaturas/tcs/practicas/matlab/car_sim.mdl

Cesar Fernandez is head of the Systems Engineering and Automation Division at Miguel Hernandez University since 2003. He teaches control theory and machine learning subjects at both industrial engineering and telecommunications engineering degrees, and currently conducts research on machine learning applied to robotics and autonomous robot grasping.

Maria Asuncion Vicente is Associate Professor at Miguel Hernandez University, Systems Engineering and Automation Division, since 2000. She teaches linear circuits theory, instrumentation and sensors and microcontroller programming. Her research interests are focused on robotics and vision systems. Currently she is working in appearance based visual object recognition based on ICA features.

Luis Miguel Jimenez is Associate Professor at Miguel Hernandez University, Systems Engineering and Automation Division, since 1997. He teaches control theory and real-time computing at both industrial engineering and telecommunications engineering degrees. His main research interest is machine vision. Currently he is working in 3D computer vision algorithms.