Java Stand-Alone Application for Linear Control Systems Simulation

L. Brito Palma, J. Costa Cruz
Universidade Nova de Lisboa – FCT – DEE
Caparica, 2829-516, Portugal;
{LBP}@fct.unl.pt

P. Sousa Gil, F. Vieira Coito
Universidade Nova de Lisboa – FCT – DEE
Caparica, 2829-516, Portugal;
{PSG,FJVC}@fct.unl.pt

***Abstract*—In this paper, a Java stand-alone application for linear control systems simulation is presented. This simulator implements linear low-order process models (first order, second order and third order), open-loop architecture and closed-loop architecture with a linear PID feedback controller. Modeling and dynamical simulation are now basic tools for understanding and verifying theoretical subjects. The main contribution is a Java application that can be used by the instructor / user in a blended learning environment to teach / learn the basic notions of dynamical systems behavior, some notions of systems stability, do time domain analysis and frequency domain analysis, and also analyze the effect of PID feedback controllers in the closed-loop system.**

Keywords— learning systems, dynamical systems, automatic control, simulation, engineering education.

I. INTRODUCTION

Dynamical systems modeling and control are main topics of automatic control. Accordingly to Karl Astrom, from Lund University – Sweden, the control technology using feedback is a hidden and magic technology: a) make precise systems from imprecise components; b) keep variables constant; c) reduce effects of disturbances and component variations in processes; d) stabilize unstable systems; e) etc. The main drawback of control technology is the danger of instability situations that can provoke faults or failures. Automatic control is a major field in almost every engineering subject and is an important part of the respective engineering curricula [1, 6, 18, 19]. In the last years, high schools developed great efforts to make available new real and virtual laboratories for the students, and also some remote laboratories, in order to facilitate the teaching / learning process [1, 3, 4, 6, 9]. Currently, the new paradigm emphasizes "What students do?" shifting the center of the process to the student, instead of the teacher. This new paradigm is in line with Bologna Declaration, and supports life long learning as well. This paradigm shift may be seen as the shifting from a "faculty-centric" to a "student-centric" teaching approach. Within this context the virtual and remote laboratories concept provides a tool to support the shift towards a teaching approach centered on students [2, 5, 8, 10, 12].

Experimentation in a physical laboratory is expensive due to the need of a number of similar equipment's, and is costly to maintain. Simulators, virtual laboratories and remote laboratories present a solution to some of these problems and are available most of the time [13]. In engineering education activities, the role of experimentation is a key concept. In general, hardware setups and processes are nonlinear systems. Computer simulation plays an important role in the teaching / learning engineering process, mainly because is one of the few procedures to deal with linear processes.

In the field of automatic control there are simulators based on different languages: Matlab, Octave, Visual Basic, Java, Java Applets, etc. In the present work, the goal was to create a simulator based on a Java stand-alone application to be used in the classroom and at home. The trend was to use free and open-source software, so Matlab and other commercial software were not an option for programming.

This paper focuses on simulation of dynamical signals and systems and PID control using a Java-based simulator, named JAVA_SIST, oriented for teaching subjects related to signals and systems and automatic control. At the moment, version 2.6 is available, running as a stand-alone Java application. Two modes are available: a) administrator mode; b) user mode. Only the administrator can change the process dynamics (transfer functions). Both user and administrator can change the controller's gains. A first version (v.1.7) of this simulator was proposed in the past [23, 20].

In section II, a brief state of the art related to simulators for dynamical systems and PID control is presented. The simulation platform appears in section III, and the simulation features are detailed in section IV. The experimental results are depicted in section V, and finally the conclusions are summarized in section VI.

II. STATE OF ART

On the WWW there exist many simulators Java-based mainly in the field of physics and mechanics, some based on based on Java stand-alone applications, others based on Java applets, [14, 15]. Focused on the area of automatic control (dynamical systems, signals and control) few simulators exist, since the automatic control area is usually a course only studied on universities and high schools. Automatic control deals, mainly, with the following topics: a) dynamical systems modeling; b) time domain responses; c) frequency domain

responses; d) stability analysis; e) open-loop and closed-loop analysis and control; f) controller design. In this area the simulators facilitates the teaching / learning process.

Few simulators exist for dynamical systems analysis and PID control using Java technologies. Most of them are based on Java applets that run inside the browser, and are very limited in terms of features and functionalities. Some of them are listed below:

1) IFAC Control Resources. A list of virtual Laboratories mainly related to automatic control is available on the web and can be found on the web-link: http://controlrc.ifac-control.org/virtual-laboratories/control.

2) PID control laboratory website. The purpose of this web site (http://www.pidlab.com/en/home) is to introduce free virtual laboratories (Java applets) for PID controller design and tuning. Using these interactive tools, PID tuning based on known process model or experimental data can be done in a very short time.

3) Michigan Technological University - Department of Chemical Engineering, written by Tomas B. Co. Web-link: http://www.chem.mtu.edu/~tbco/cm416/newpidb.html. The program simulates a linear process. There are three modes of operation available: Manual, Relay and PID. Also, the user can change the set point and control parameters. The program is written in Java v1.1.7 and runs inside the browser.

4) Université Aix Marseille. The site contains Java applets to study the behavior of low order systems with a feedback controller (open-loop, PID, zero-pole compensator with proportional gain) in the loop. The web-link is the following (site in "French" language): http://www.ta-formation.com/applets/process/jav-process.htm. The author is Olivier Guédon.

5) PID trainer - PID loop simulator project ("pidloopsimulato") on "sourceforge.net", mainly focus on PID design and PID training. The web-link is http://pidloopsimulato.sourceforge.net/.

6) PID simulator using Microsoft Excel, for a first order system with time delay. Site: http://www.engineers-excel.com/Apps/PID_Simulator/Description.htm.

Authoring tools written in Java can also be found on the web to help non-programmers. One of them is Easy Java Simulations (EJS). EJS is a platform to develop Java applications and can be found in http://fem.um.es/Ejs/. EJS is part of the Open Source Physics project and has been created by Francisco Esquembre, [17].

III. Simulation Platform

Here, in this paper, the main goal was the development of a platform for simulating linear dynamical systems and feedback PID controllers, in a language (Java) that is compatible with the most popular operating systems. In each cycle of the simulation, the platform must read the user settings in order to update the simulation parameters, perform the simulation, and display the results to the user (vide Fig. 1). It is important to guarantee the sequence of these actions hence the support of a synchronized multithreading is needed. The language must allow a simple way to interact with the user by means of an intuitive graphical interface, which allows the user to parameterize, supervise and save the simulation for posterior processing. These premises along with the need for productivity and simplicity have led to choose JAVA as the development language for this project, [14, 15]. Furthermore, JAVA has some other appealing features, such as:

1) Object-oriented based: it is based on several techniques, including inheritance, modularity, polymorphism, and encapsulation;

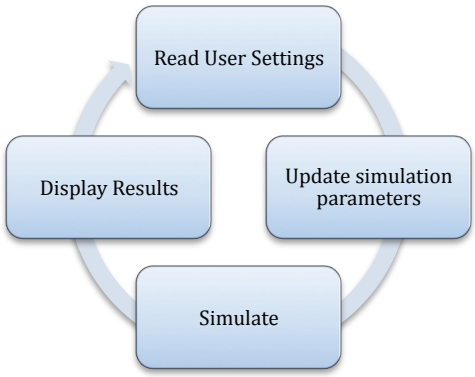

Fig. 1. Platform simulation architecture.

2) Robustness: it allows developing reliable software, and helps reducing the error probability in software applications, with catching exceptions. The internal memory management provides also an efficient programming;

3) Simplicity*:* the construction of JAVA programs does not require an extensive knowledge. A wide number of available libraries make easy its development. Also, has it is one of the most used language by the programmers community, it has a wide range of information and open source software easily accessible on-line;

4) Portability: its independency enables the possibility to execute applications in computers having any operating system, just only requiring the Java Virtual Machine (JVM) installation;

5) Distributed: this language offers a group of classes that allows communication through networks, in a transparent and simple way for the user. Also, multithreading makes possible the communication by sockets at the same time users perform other actions.

6) JAVA Native Interface (JNI): JAVA presents a native programming interface that allows the use of libraries implemented in other programming languages.

JAVA SE Runtime Environment 7 was used for the development of the simulator. The definition of classes and its attributes fits in the description of the system, and the methods defined for each class accurately represents the concept of system responses. Advantages and disadvantages of server-side and client-side technologies (Java, Ajax, Html, etc) in the design of virtual and remote laboratories can be found in [7].

CONTROL LABORATORY

In almost every engineering course the subjects of "Signals, Systems and Control" play a very important role, and are the basis for the development of new and sophisticated architectures. In order to understand and verify theoretical subjects, dynamical systems simulation is nowadays a useful tool for testing control systems, without having practical issues related to physical plant experimentation.

Fig. 2. Analog electronic simulator (HW123b setup).

Most of the existing processes and plants that are available for systems and control experimentation are nonlinear and some of them are also time variant. When the goal is to learn linear and time invariant (LTI) systems, these features difficult the student learning process. In our automation and control laboratories, many hardware setups are available, namely water tanks, thermal plants, DC motors, etc. Laboratory environments are classified according to the type of user access and equipment nature (Table 1). As they are real setups, they can be strongly non-linear and time variant. Therefore, to teach Linear Systems and Control Theory concepts, a linear system setup is required. In the past, these premises led to the development of an analog electronic simulator denominated HW123b (Fig. 2) for emulating LTI systems using operational amplifiers, resistors and capacitors. The HW123b setup can emulate dynamical systems of first-order, second-order or third-order (Fig. 3), and is accessible on-line, hence being classified as a remote lab environment [6].

The JAVA_SIST application is proposed with features similar to the HW123b setup, allowing the users to experiment in the real device and compare the results with the simulated version.

TABLE I. CLASSIFICATION OF LABORATORY ENVIRONMENTS.

		User Access	
		Local	***Distant***
Nature of Equipment	***Physical (real)***	Hands-on lab	Remote lab
	Virtual (modeled)	Virtual lab	Distributed virtual lab

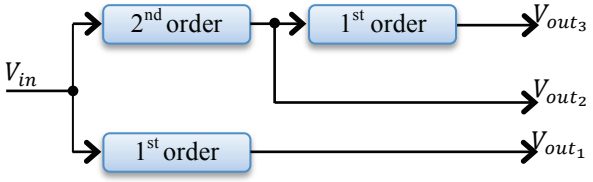

Fig. 3. Block diagram of the analog electronic simulator (HW123b).

The linear systems emulated by the HW123b are discretized in the JAVA_SIST application. This approach allows the user to safely test the designed controllers in the simulator and then verify the theory in the real hardware process HW123b.

In this work, the JAVA_SIST application / simulator is one of the resources in a student's centered approach accordingly to the proposed blended learning architecture (Fig. 5), for learning automatic control concepts. Blended learning (BL) systems combine face-to-face (F2F) instruction with computer-mediated instruction [22]. The simulator promotes the individual work, the autonomy and also an efficient time management.

Using the JAVA_SIST simulator, one possible learning process sequence is: a) theoretical concepts study; b) understand solved exercises; c) try to solve problems (with and without Matlab, Scilab or Octave); d) practice on the hardware simulator HW123b; e) practice on the software simulator JAVA_SIST.

The real pedagogical value of the JAVA_SIST application, as a virtual lab, will be tested in the course "Systems Theory (2014/15)" at our Faculty.

IV. SIMULATION FEATURES

In this section, the simulator features are presented.

At the present stage of development, the main functionalities of JAVA_SIST application are the following, depicted in Fig. 4: a) open-loop and closed-loop simulation of low order (first, second and third order) systems; b) time domain analysis (impulse response and step response); c) frequency domain analysis (frequency response for a sinusoidal input signal); d) controller testing (P, PI, PID with

and without anti-windup mechanism); e) noise addition to reference signal, control input and/or plant output; f) FFT computation for reference signal, control input and/or plant output. These application features permit the support of an introductory course on Systems and Control Theory at high schools.

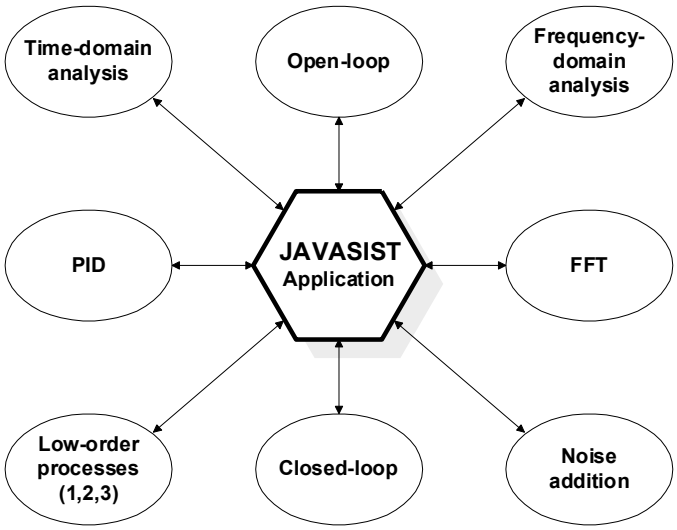

Fig. 4. Main functionalities of JAVASIST application.

In order to demonstrate the functionality of the tool JAVASIST 2.6, some experiments can be done, as depicted in the section "Simulation Results". Assuming that the tool will be used for an Introductory Course on Signals and Systems, the following plan is proposed, assuming the functionalities depicted in Fig. 4:

i) Assume a plant model (first, second or third order), in open-loop;

ii) Perform time-domain analysis of the plant model in open-loop: stability analysis based on impulse response;

iii) Perform frequency-domain analysis: build a Bode diagram based on response to sinusoidal input signals;

iv) Apply the FFT for input signals and for output signals used to build the Bode diagram;

v) Design different types of controllers (P, PI, PD and PID) using the Ziegler-Nichols methods or other approaches, and test the controllers in closed-loop, in order to analyze the control-loop performance;

vi) Add noise to reference signal, control input and/or plant output, in order to analyze the robustness of the control structure in closed-loop;

vii) Inject white noise on the input command signal in order to do system identification, i.e., discover the plant model from input-output signals, [18, 24];

viii) Change, on-line, the plant model or the controller structure, in order to evaluate the change in the control-loop performance.

A. Plant Simulation

The JAVA_SIST application is able to simulate dynamical systems of first-order, second-order and third-order [18, 19]; the respective transfer functions are the following, (1), (2), and (3). Typical physical systems that can be modeled by these transfer functions are heating systems, liquid-level systems, electrical motors, etc, [18, 21, 24].

$$G_1(s) = \frac{K_1}{\tau s+1} \quad (1)$$

$$G_2(s) = \frac{K_2\omega_n^2}{s^2+2D\omega_n s+\omega_n^2} \quad (2)$$

$$G_3(s) = G_1(s)\ G_2(s) \quad (3)$$

In equations (1, 2, 3) the coefficients of the transfer functions used in this work are the following, accordingly to the hardware setup (HW123b): $K_1 = 1.02$; $K_2 = 1.05$; $\tau = 1.05\ s$; $D = 0.57$; $\omega_n = 2.4\ rad\ s^{-1}$. The first-order system has a stable real pole and the second-order system has two stable complex poles. For implementation, the transfer functions were discretized with a sampling period of $h = 0.1\ s$ using the ZOH method.

B. Controller Simulation

The application simulates an open-loop architecture, and a closed-loop architecture with P, PI and PID controllers [18, 19, 21]; the basic transfer functions are the described in (4, 5, 6).

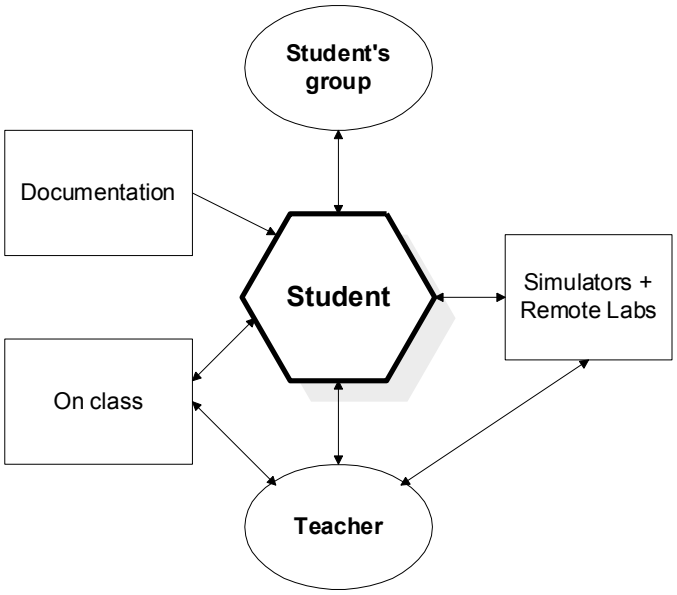

Fig. 5. Blended learning architecture student's centered.

$$C_1(s) = K_p \quad (4)$$

$$C_2(s) = K_p(1 + \frac{1}{T_i s}) \quad (5)$$

$$C_3(s) = K_p(1 + \frac{1}{T_i s} + T_d s) \quad (6)$$

In the software, discretized versions of the controllers were implemented, using an anti-windup mechanism with parameter Tt and a derivative parameter N; more details can be found in the book [21]. A simplified version of the pseudo-code of the discretized PI controller is presented in the next lines:

b_i = Kp * h / T_i; a_0 = h / Tt; % h = sampling time

e(k) = r(k) – y(k); % control error in discrete time "k"

P(k) = Kp * e(k); % proportional action

v(k) = P(k) + I(k-1);

u(k) = satur(v(k),ulow,uhigh); % control action (saturated)

aw(k) = a_0 * (u(k) – v(k)); % anti-windup

I(k) = I(k-1) + b_i * e(k) + aw(k); % integral action

C. Control Loop

The overall closed-loop control architecture used in the simulation, using switching mechanisms, can be observed in Fig. 6. The reference signal r is user-defined and can be modified any time while the simulation stems. The software, according to the selected controller and the plant, computes the control action signal u and the plant output signal y. The signals δ_c and δ_p are, respectively, the user selection of controller and plant. The user can adjust the controller gains K_p, T_i and T_d, and also the parameters Tt and N.

D. Graphical User Interface

The graphical user interface (GUI) is divided into several parts (Fig. 7). In the center there is a graphic window showing the simulation data: reference r (red), plant output y (blue) and control action u (green). The signals are shown as the simulation stems sample by sample.

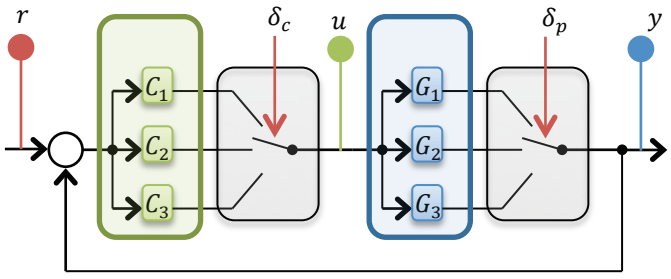

Fig. 6. Control loop architecture with switching mechanisms.

It is possible to work in an open-loop control or in closed-loop control architecture. In open-loop control the user sends a command signal to the process input. In closed–loop a feedback PID controller acts on the process input. In the middle left side, the user controls the reference to be given to the control system in closed-loop, or establish a command signal in open-loop control.

The reference/command signal can be modified any time while the simulation stems, using a numeric value or a slider, at the left side. In the middle right side are given tools for recording and saving the simulation. The user may start or stop the recording, and also export the data to a formatted text file for posterior processing, using the Matlab software or other software such as the Scilab, Octave, Excel, etc. It is also possible to import a command signal from a specific file (*.sgn).

In the lower right side the user may choose which plant should be simulated. In the lower left side the user may choose between open-loop control and closed-loop control using the available controllers (P, PI, PID and PID with anti-windup mechanism).

The information relative to the duration of the simulation and the number of samples computed are also shown in the upper right corner. Also, the user can start and stop the simulation any time, which is one of the advantages of the simulator.

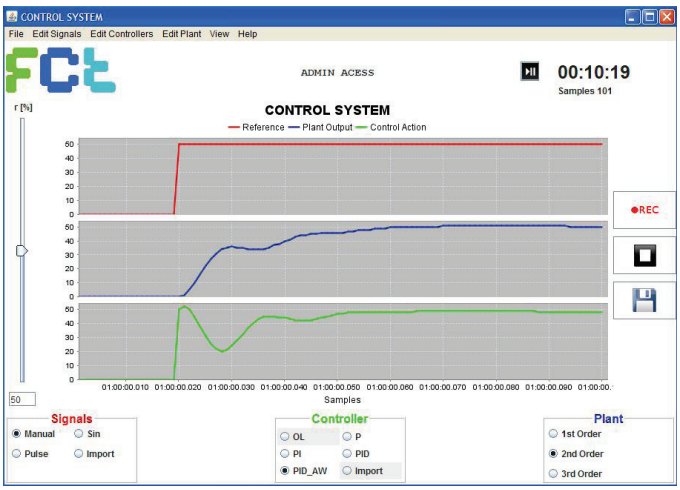

Fig. 7. Graphical User Interface (GUI): 2nd order system with PID_AW controller.

In Fig. 8 can be observed one possible control architecture, in this case a closed-loop feedback PID controller with anti-windup mechanism, and also the available signals (r: reference signal, y: output signal, u: control action).

Menus are also available in order to execute the following tasks: a) file (save and clear); b) edit signals (pulse, sinusoidal, noise, import, export); c) edit the gains for the set of controllers (P, PI, PID, PID with anti-windup, import); d) edit the plant parameters (1st order, 2nd order, 3rd order, import),

only for the administrator; e) view control loop architecture or the FFT transform; f) program help (about).

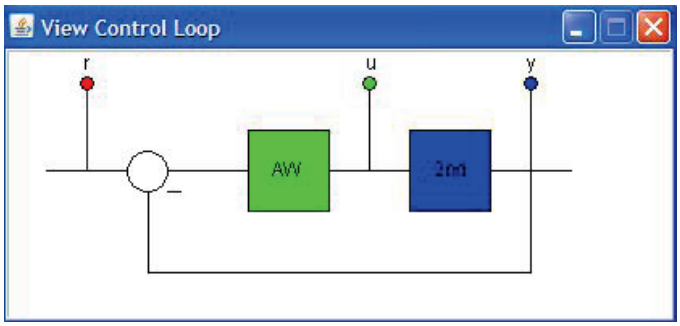

Fig. 8. Closed-loop control with feedback PID controller with anti-windup (AW) and 2nd order process.

V. Simulation Results

Some simulation results, obtained with the proposed Java simulator, are described in this section.

A sampling time of 0.1 s was used in the simulations.

In Fig. 9, the first order model defined in (1) was first selected and tested in open-loop architecture. A first-order model can represent the dynamic behavior of some real systems such as heating systems, liquid level systems, etc. The command signal applied to the control system is a step of magnitude 0.5, i.e., 50%. Fig. 9 also shows the reference signal "r" (red graphic), the plant output "y" (blue graphic) and the control action "u" (green graphic).

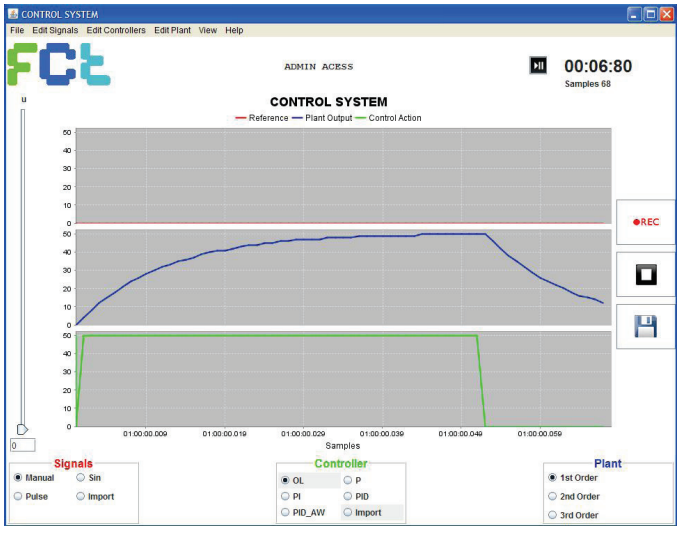

Fig. 9. Simulation result for step response with first order plant and open-loop architecture.

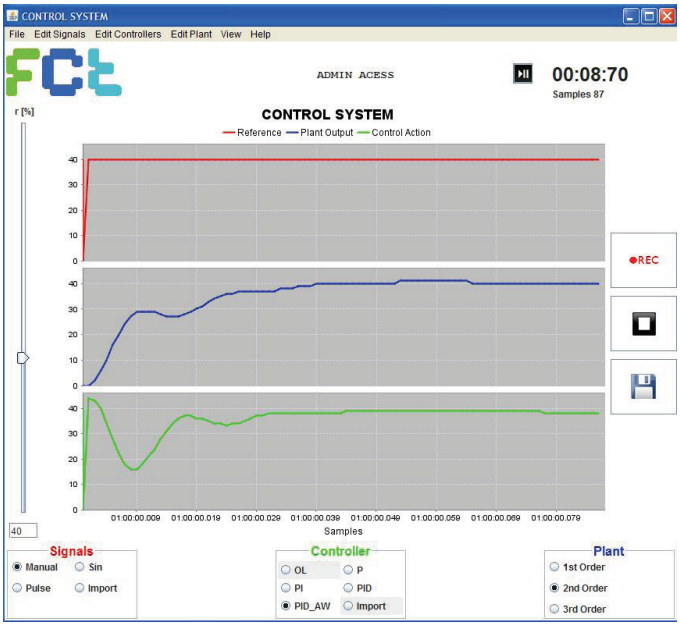

Fig. 10. Simulation result for step response, second order plant and PID controller with gains [Kp, Ti, Td, N, Tt]=[1, 1, 1, 1, 1].

In Fig. 10, the second order model defined in (2) was first selected and tested in closed-loop architecture. A set-point (reference) signal of 40% was chosen. The static error tends to zero, as expected accordingly to the specification unitary static gain for the closed-loop. The closed-loop performance could be significantly improved by fine-tuning the controller gains. The primary goal in the experiment was not actually to tune well the controller, but instead test the platform functionalities.

Another simulation was executed, this time for the third order plant model. By changing the gain Kp to 50 an oscillating plant output response is obtained, as depicted in Fig. 11. It should be emphasized that the saturation of the controller action *u* provokes a nonlinear behavior in the control system.

The ability to switch on-line between controllers and plant models, allows the user to understand the need for retuning the controller according to the selected plant model.

In Fig. 12 can be observed part of the experimental data saved on a text file, for the oscillatory behavior (Fig. 11). This data can be post-processed in order to be used for system identification, controller design, control loop performance analysis, etc.

In the future, more experiments will be available, on order to compare the data captured from the hardware setup HW123b with the data obtained from the Java simulator JAVASIST 2.6. This comparison permits to understand the main differences that exist between a real setup and a simulation model.

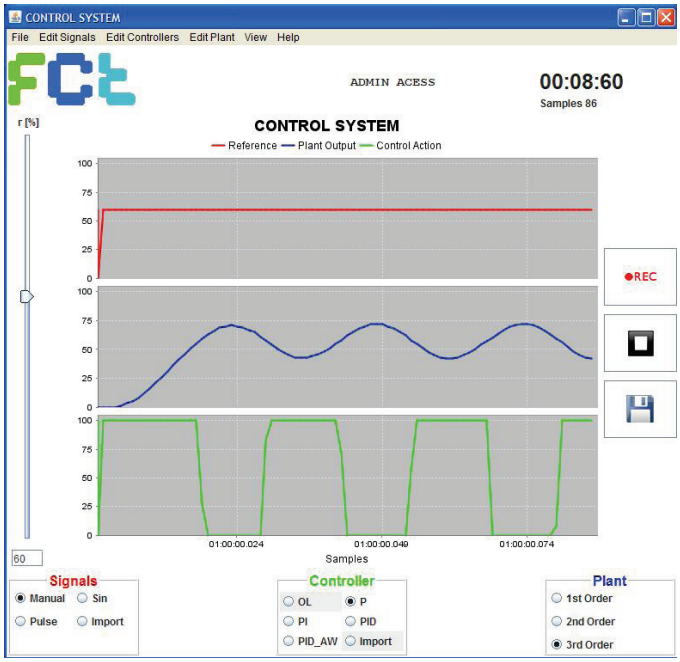

Fig. 11. Simulation result for step response, third order plant and proportional controller with gain set to 50.

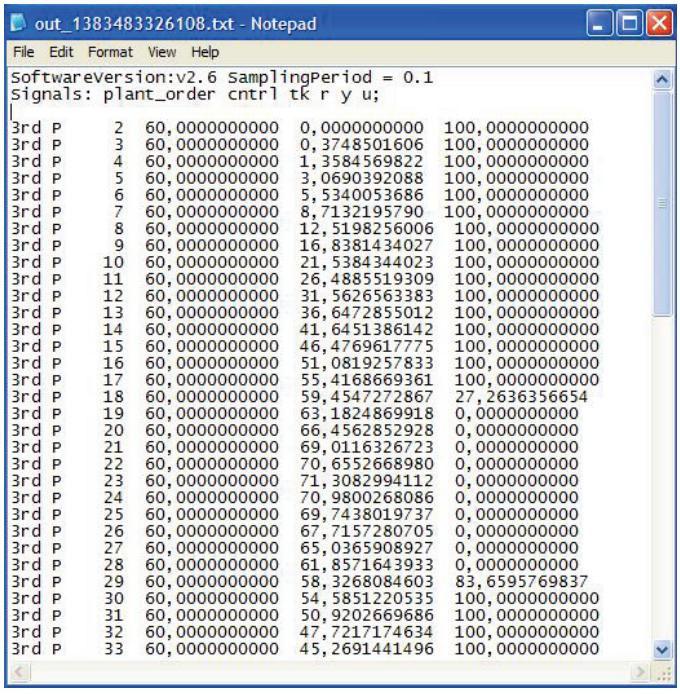

Fig. 12. Experimental data saved on a text file (Fig. 11 experiment).

The notions of closed-loop stability and instability can be also study using the simulator. Fig. 13 shows the impulse response of the 2nd order system. In Fig. 14 can be observed a closed-loop stable 2nd order system for a certain set of controller gains (PID with anti-windup); after increasing the proportional gain (1 to 10) the closed-loop system tends to the instability.

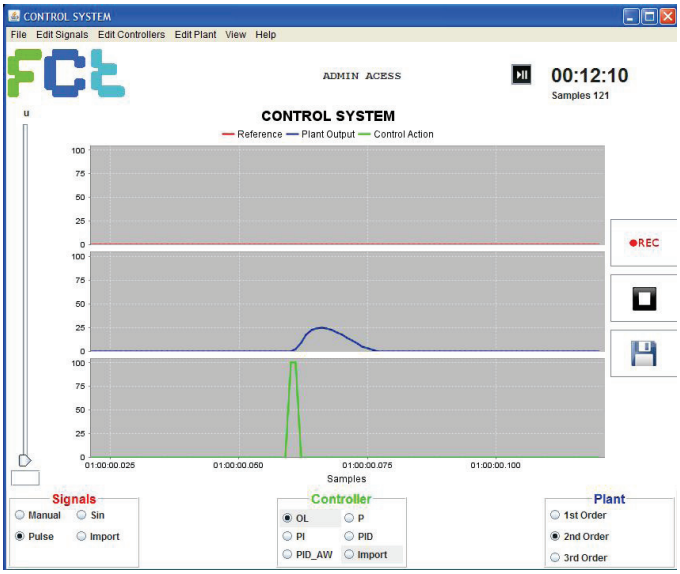

Fig. 13. Impulse response of the 2nd order system.

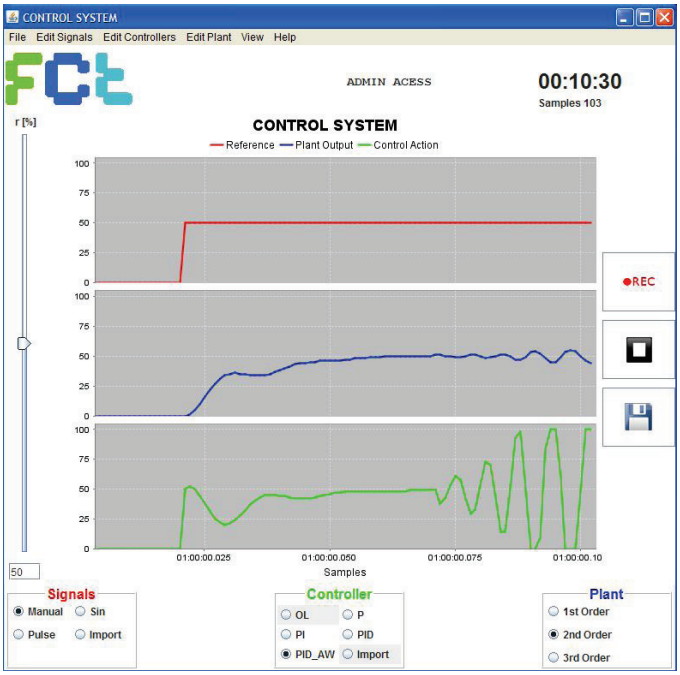

Fig. 14. Closed-loop stable and unstable system. Set of gains: [Kp, Ti, Td, N, Tt] = [1, 1, 1, 1, 1] change to [10, 1, 1, 1, 1].

In order to do a frequency response analysis of the first order system a sinusoidal input signal (with amplitude A, frequency f and DC value d given by 10, 10 Hz and 20) was

generated and the fast Fourier Transform (FFT) was applied to it, as shown in Fig. 15. The equation of the input signal is the following: u(t) = A * sin(2 * pi * f * t) + d.

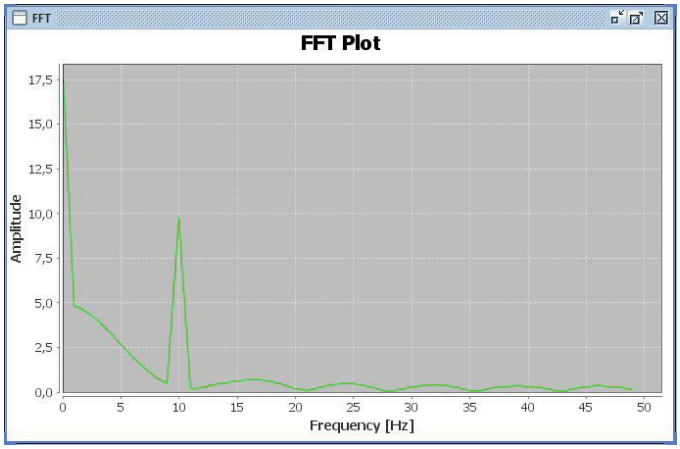

Fig. 15. Fast fourier transform applied to a sinusoidal input signal.

Finally, Fig. 16 depicts the response of the first order signal to an input command signal with DC value of 50% and with white noise. This kind of experiment can be used for system identification purposes; the plant model can be discovered using algorithms based on least-squares, principal components regression, etc, [18].

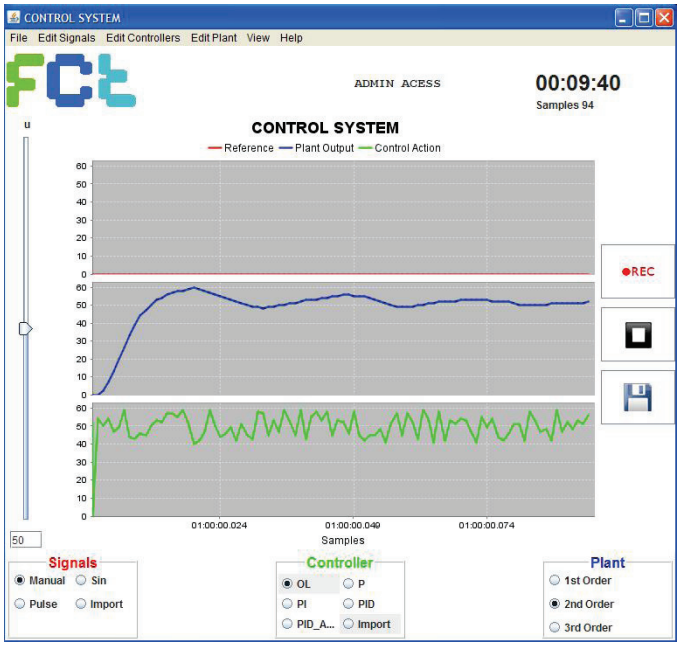

Fig. 16. Input command signal with white noise.

VI. CONCLUSIONS

In this paper, a Java stand-alone application (JAVASIST 2.6) for linear control systems simulation was described and presented.

The JAVASIST simulator presents a solution to the problem of allowing students to share expensive laboratory experiments in the systems and control field, in an efficient way.

At the moment, the simulator can only deal with low-order linear models (first, second an third orders). Open-loop control and closed-loop control architectures can be simulated. Time domain analysis and frequency domain analysis can be done. Noise and FFT transform can be applied to the signals.

This is one more contribution to the field of e-learning in the automatic control area.

The future work will be oriented to evaluate the pedagogical impact and improve the functionalities of the JavaSIST simulator 2.6, such as: a) addition of perturbations; b) allow changing the sampling period; c) implementation of algorithms for control loop performance analysis (CLPA) based on the Harris index; d) enable remote monitoring and control using the TCP/IP protocol and web services.

ACKNOWLEDGMENT

This work has been supported by Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, by Uninova-CTS, by CISUC-Coimbra and by national funds through FCT - Fundação para a Ciência e a Tecnologia (project PEst-OE/EEI/UI0066/2011). The authors would like to thank all the institutions.

REFERENCES

[1] S. Dormido. “Control learning: Present and future”, Proc. 15th IFAC World Congr. Automatic Control, Barcelona, Spain, pp. 81-103, 2002 .

[2] J. Ma, J. Nickerson. “Hands-on, simulated and remote laboratories: A comparative literature review”, ACM Computing Surveys, (38:3) article no. 7, pp. 1-24, 2006.

[3] M. Johansson, M. Gafvert, K. Astrom. “Interactive tools for education in automatic control”, IEEE Control Systems Magazine, 18, no. 3, pp. 33-40, 1998.

[4] F. Vieira Coito, P. Almeida, L. Brito Palma. “SMCRVI – A Labview/Matlab based Tool for Remote Monitoring and Control”, 10th IEEE International Conference on Emerging Technologies and Factory Automation, Sept 19-22, Catania, Italy, 2005.

[5] L. Gomes, F. Coito, A. Costa, L. Brito Palma, P. Almeida. “Remote Laboratories support within Teaching and Learning Activities”, International Conference on Remote Engineering and Virtual Instrumentation (REV’07), June 25-27, University of Porto, Porto, Portugal, 2007.

[6] F. Coito, L. Brito Palma. “A Remote Laboratory Environment for Blended Learning”, PTLIE Workshop - Pervasive Technologies in E/M Learning and Internet based Experiments, 1st ACM International Conference on Pervasive Technologies Related to Assistive Environments (PETRA 2008), July 16-18, Athens – Greece, 2008.

[7] J. Garcia-Zubia, P. Orduna, D. Lopez-de-Ipina, G. Alves, “Addressing Software Impact in the Design of Remote Laboratories”, IEEE Trans. on Industrial Electronics, vol. 56, no. 12, 2009.

[8] M. Ngolo, L. Brito Palma, F. Coito, L. Gomes, A. Costa, “Architecture for Remote Laboratories based on REST Web Services”, 35th Annual Conference of the IEEE Industrial Electronics Society, IECON / ICELIE, Nov. 3-5, Porto – Portugal, 2009.

[9] L. Brito Palma, F. Vieira Coito, A. Gomes Borracha, J. Francisco Martins, “A Platform to Support Remote Automation and Control Laboratories”, 1st Experiment@ International Conference, Nov. 17-18, Lisboa – Portugal, 2011.

[10] L. Gomes, S. Bogosyan, “Current trends in remote laboratories”. IEEE Transactions on Industrial Electronics, 56(12), 4744-4756, 2009.

[11] T. Restivo, and A. Cardoso, "The Portuguese Contribution for lab2go.pt. lab2go", http://www.lab2go.net/, International Journal of Online Engineering, vol. 9, Special Issue 1, 2013.

[12] A. Cardoso, T. Restivo, P. Cioga, M. Delgado, J. Monsanto, J. Bicker, E. Nunes, and P. Gil, "flock.uc.pt – A Web Platform for Online Educational Modules with Online Experiments", International Journal of Online Engineering, vol. 9, Special Issue 1, 2013.

[13] L. Gomes, J. Garcia-Zubía. Advances on remote laboratories and e-learning experiences. Univ. Deusto, Bilbao, 2007.

[14] R. Raposa, SCJP – Sun Certified Programmer for Java Platform, SE6 – Study Guide, 1st ed., Wiley, 2009.

[15] I. Darwin, Java Cookbook, O’ Reilly, 1st edit., 2001.

[16] G. Farias, R. Keyser, S. Dormido, F. Esquembre, “Developing networked control labs: a Matlab and easy Java simulations approach”, IEEE Transactions on Industrial Electronics, 57(10), 2010.

[17] F. Esquembre's, “Easy Java Simulations”, Univ. de Murcia, Spain, website: http://www.um.es/fem/EjsWiki/, 2013.

[18] W. Levine, The Control Handbook, CRC Press, 1996.

[19] K. Astrom, R. Murray, Feedback Systems: an introduction for scientists and engineers, Princeton University Press, 2008.

[20] L. Brito Palma, J. Costa Cruz, F. Vieira Coito, P. Sousa Gil, “Interactive Demonstration of a Java-Based Simulator of Dynamical Systems”, 2nd Experiment@ International Conference, Universidade de Coimbra, Sept. 18-20, Portugal, 2013.

[21] K. Astrom, T. Hagglund, Automatic Tuning of PID Controllers, Instrument Society of America, 1988.

[22] C. Graham, Blended learning systems: definition, current trends, future directions. In C. J. Bonk & C. R. Graham (Eds.). Handbook of blended learning. Pfeiffer Publishing, 2006.

[23] L. Brito Palma, J. Costa Cruz, F. Vieira Coito, P. Sousa Gil, “Java-Based Simulator of Dynamical Systems and PID Control”, 2nd Experiment@ International Conference, Universidade de Coimbra, Sept. 18-20, Portugal, 2013.

[24] Oppenheim, A., A. Willsky, S. Hamid, Signals and Systems, Prentice-Hall, 1983.