(/)

CCAC 2019 (/event?id=41) Presentation [F2-1-3] (/presentation?id=57336)

(/event?id=41)

Development and Control of Virtual Plants in a Co-Simulation Environment

Harold Fernando Ruiz Bravo[1], Laura María Rodríguez Rivera[1], Andrés Pantoja[1], Jhon Barco Jiménez[2]
[1]Universidad de Nariño
[2]Institución Universitaria CESMAG

Details

Category
Regular Session

Sessions

09:10 - 10:30 | Fri 18 Oct | Orinoco | F2-1
Applied control for industrial and non-industrial areas

Full Text

Abstract

Virtual plants represent one of the most versatile instruments in the teaching-learning process of control classes since they may replace expensive laboratories or real plants. This work proposes a methodology to use the 3D simulator V-REP and Matlab for the construction, communication, and control of virtual plants in a co-simulation environment. The implementation of the ball-and-plate and ball-and-beam systems shows the control possibilities and the differences on working with ideal-model simulations and using specialized platforms to include unmodeled physical phenomena. Using two simple controllers, the results show how the real-based experiments produce unexpected responses that should be corrected with distinct control strategies.

Additional Information

Detailed information about construction, implementation and communication of a virtual plant can be found in the following links

primitive model construction

https://www.youtube.com/watch?v=7zkXhFoC6QA

Solidworks-V-REP element importing

https://www.youtube.com/watch?v=UcvJevHg8JI&t=7s

V-Rep-Matlab Remote Api handling

https://www.youtube.com/watch?v=Rhe7yBWqvIU&t=3s

Video

No videos found

(/)

Log in (/login)

Contact Us (/contact-us)

Privacy Policy **(/pdf/terms-of-use.pdf)**

© 2019 Infovaya.

2019 IEEE 4th Colombian Conference on Automatic Control (CCAC)
October 15-18, 2019. Diez Hotel, Medellin, Colombia.

Development and Control of Virtual Plants in a Co-Simulation Environment

Harold Fernando Ruiz Bravo
Departamento de Electrónica
Universidad de Nariño
Pasto,Colombia
harolfernandoruiz@hotmail.com

Laura Maria Rodriguez Rivera
Departamento de Electrónica
Universidad de Nariño
Pasto,Colombia
lauritamaria_rod@hotmail.com

Andrés Pantoja
Departamento de Electrónica
Universidad de Nariño
Pasto,Colombia
ad_pantoja@udenar.edu.co

John Barco
Facultad de Ingeniería
I.U. CESMAG
Pasto,Colombia
jebarco@iucesmag.edu.co

Abstract— **Virtual plants represent one of the most versatile instruments in the teaching-learning process of control classes since they may replace expensive laboratories or real plants. This work proposes a methodology to use the 3D simulator V-REP and Matlab for the construction, communication, and control of virtual plants in a co-simulation environment. The implementation of the ball-and-plate and ball-and-beam systems shows the control possibilities and the differences on working with ideal-model simulations and using specialized platforms to include unmodeled physical phenomena. Using two simple controllers, the results show how the real-based experiments produce unexpected responses that should be corrected with distinct control strategies.**

Keywords— Co-simulation, Virtual plants, V-REP, ball-and-plate, ball-and-beam.

I. Introduction

A virtual plant is a software-based environment, where users interact on a series of graphical components that represent elements of a physical model. These basic elements are modeled in a specialized software platform forming more complex mechanisms, in such a way, that the system closely emulates the actual plant performance. Some examples of virtual plants include mechanical systems working on a simulation and remote laboratories, where the plants are real (and expensive) but operated through internet by diverse users [1].

On the other hand, these environments have had impact in many knowledge areas and engineering research, making the most of the advance of 3D simulation-based technologies and the relevance they have had in pedagogy and ludic activities [2] [3]. This penetration has facilitated the improvement of many 3D and 2D simulation environments for educational purposes, which allow users to consolidate concepts and merge theory with practice in a simple and accessible way.

One of the most widespread platforms in this type of application is V-REP, whose features support simple and complex forms (e.g., different types of robots), as well as the inclusion of 3D models from external CAD software. Moreover, V-REP has affinity with multiple programming languages, including Matlab.

In the field of educational software, especially in the technical area, this work intends to solve the lack of expensive laboratory equipment and specialized didactic prototypes through using the emerging technologies above mentioned. An interactive platform integrating a plant simulator and a control software facilitates the joint of theoretical and practical knowledge, avoiding high investments in equipment.

Several works have focused on the construction of virtual plants. For instance, the authors in [4] present a 3D virtual reality simulator for the development of controllers to follow trajectories with a 6-DOF robotic arm. In [5], it is shown the implementation of a didactic tool that allows simulating, observing, and analyzing the behavior of industrial processes, emphasizing on the communication with a real PLC. Using the main features of the V-REP environment, the proposal in [6] shows the construction, design, and simulation of a LEGO EV3 robot in the V-REP 3D simulator.

This work describes the use of a co-simulation between a free environment of virtual plants (V-REP) and a common language for the study of control (Matlab), working in a platform communicated by an API (application programming interface). To show the methodology, a ball-and-plate and a ball-and-beam plants are presented. It should be noted that the contribution of this work does not focus on the development of advanced controllers for a particular plant but the proposal of a pedagogical methodology taking advantage of virtual plants. The method can be adapted for the implementation of different plants emulating real behavior in contrast to standard model-based simulations.

To analyze the differences between the simulation results and the ones obtained in the example plants, two controllers (i.e., PID and LQR) are tuned to track simple trajectories of the ball. The comparison shows the gap between a real controller and a simulation given that the virtual plant emulates physical properties as friction, displacements, and slides, which are close to reality. This conditions allow students to face demands or real systems that are not obtained in a simulation (e.g., noise, disturbances, actuator limits, and physical barriers), improving theoretical-practical training and the actual application of control concepts.

The rest of the work is organized as follows: Section II presents the virtual environment and the co-simulation scheme, while Section III describes the development of the virtual plants. Section IV explains the results obtained, taking into account the differences of controllers within the simulation and the real plants. Finally, Section V contains the conclusions and future work.

II. Virtual Environment and Co-Simulation Scheme

The implementation of a virtual plant in V-REP presents two important processes. The first one consists of the analysis of the advantages and limitations of the software with respect to the different objects, shapes, sensors, and actuators that constitute the plant. The second one involves the communication between the objects part of the virtual plant and an external program by means of a remote API (Application Programming Interface).

A. *V-REP Generalities*

V-REP is a program with a free educational version that is used as a tool to test and develop virtual robots and mechanical systems. It is characterized by having extensive documentation, allowing importation of 3D models, and having compatibility with multiple programming languages, including Matlab.

The software has a processing engine for advanced kinematic and dynamic calculations that allows the simulation of environments and physical parts (e.g., solid pieces, motors, and sensors, among other work options) in its graphical interface. The interaction with other software platforms is carried out through more than 400 functions in the native API of the simulation environment, whose documentation is detailed in [6] and [7]. It should be noted that the simulator is capable of emulating real physical behaviors such as those produced by gravity, inertia slips of parts, possible external disturbances, as well as collisions and frictions between the different objects found in the workspace.

Physical components in V-REP are articulated with each other by joints that are classified according to the type (rotational or translational) or the operation way (passive, inverse kinematics, object dependent, motion, or torque-force). The joints are moved by motors, representing the main actuators of the plants. Furthermore, there are a great variety of sensors to complete the feedback loop in the control schemes [7].

In terms of simulation time, V-REP presents restrictions on the minimum sampling time that can be achieved, which is between 1 and 200 milliseconds. The sampling period depends mainly on the computer capacities of the used equipment, the complexity of the plant (number of simple forms, actuators, and sensors), and the use of external servers such as the remote API for interconnection with other programs.

B. *V-REP Remote API*

The API is used to control a simulation (reading and writing properties and attributes of components, actuators, and sensors) from external hardware or software. It consists of more than one hundred functions, which can be called from a C++ application, Python, Java, or a Matlab script [6], [7]. The functions interact with V-REP through Blue Zero Middleware [8] and its interface plugin with V-REP [7]. The remote API allows one or more external applications to interact with the simulator in synchronous or asynchronous mode.

In the case of a co-simulation, the client (development in Matlab) must be synchronized with the progress of the simulation in V-REP. In this mode, each step of the simulation, equivalent to a sampling period, determines a new calculation cycle based on the data sent to the actuators and collects the data provided by the sensors. In this way, the feedback is provided and it synchronizes both programs.

In the development proposed in this work, we adapt the original remote API of V-REP and Matlab [7]. For this purpose, the "remotApi" script is modified, along with the functions of bidirectional transmission of information. In consequence, the commands used in Matlab are simpler to use and have only the necessary parameters, facilitating the co-simulation execution and configuration.

The co-simulation is performed by sending commands from Matlab that, through the API, are executed in V-REP. A high amount of commands offers the versatility that allows Matlab to obtain all type of information from the objects in the plant, as well as to establish certain parameters in actuators (e.g., velocities or positions in motors). However, the original functions are quite complex given the large number of parameters they need to use. For this reason, we modify the script "remote" to ease the link between the two platforms.

III. Two Case Studies: Ball-and-Plate and Ball-and-Beam Systems

To illustrate the process of implementing virtual plants and their control through Matlab, we present the development of a ball-and-plate and a ball-and-beam experiments. The first plant consists of a surface that pivots on a central articulation moved by two servomotors that control the rotation angle on the x and y coordinates. The complete system has two degrees of freedom and the measurement of the position of the ball on the surface can be sensed by a camera or directly with the properties of the location of the sphere in the V-REP space. The second system is composed by a motor that turns a beam containing a ball. Since the motor is attached to a fixed base, the rotation produced by the torque allows the beam to locate the ball in the desired position. The system has one degree of freedom and the position of the ball can be measured by using a proximity sensor located at one end of the beam or also using the position property of the ball provided by the V-REP API.

A. *Ball-and-Plate Mechanical Design in V-REP*

To establish the structure of the plant, we use the so-called "pure forms" in V-REP, which are simple solids such as bars, plates and spheres. Each element can be defined with properties such as dimensions, mass, and position, characteristics that determine the operation within the 3D scheme. Once all the necessary elements for the construction of the system have been defined, the different pieces are put together by means of the "group select shapes" option, which joins the different shapes in a set, as the one shown in Figure 1. In this initial state, the main pivot is embedded on a rotational articulation in the middle of the table. In the same way, the arms articulated to the servomotors are joined to control the movement of the surface.

As the original shapes of V-REP are not esthetic and represent basic shapes, for the emulation of a real scheme it is necessary to import models designed in CAD tools such as Solidworks [9], [7]. To do this, the files are imported in obj, dxf, or stl formats that contain basic, but clean shapes. It is worth noting that if complex figures are used, the simulation becomes slow and the sampling times in the co-simulation increase considerably.

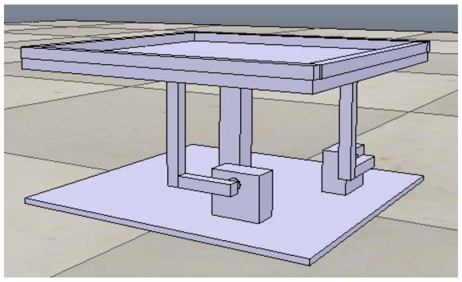
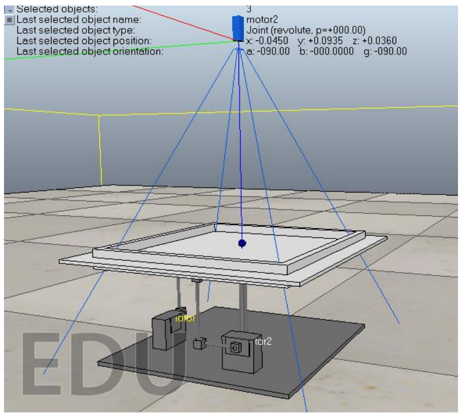


Fig. 1. Initial structural design of the ball-and-plate system in the main environment of V-REP.

The imported pieces are grouped into a tree structure so that the simulator associates each basic form to its properties in the native language. In addition, the couplings between all elements are indispensable, because if the shapes are not properly joined, the pieces fall to the floor when the simulation starts (similar to solids in a real environment). The grouping is done hierarchically by taking a "father" and associating a "son" through a simple menu. Once all the pure forms of V-REP are associated with each other and with the models imported from the CAD software, the original solids of V-REP become "invisible", producing a polished scheme as the one shown in Figure 2.

Regarding the actuators, the plant has two motors that move two arms joined to the plate with rotational joints located at the same axial distance. The actuators are assumed to be symmetrical for motion in each of the axes as illustrated in Figure 2. The joints are configured in torque-force mode, while the motors are selected in the angular-position mode to enable the servo function (i.e., the input of each motor is the angular position). To capture the position of the ball with a real sensor, V-REP can use a fixed camera, whose image is exported to Matlab estimate the target on the plate. However, to ease the measure of the ball location, we use directly the properties obtained through the API of the simple, free-moving sphere on the plate.



```
Selected objects:                3
Last selected object name:       motor2
Last selected object type:       Joint (revolute, p=+000.00)
Last selected object position:   x: -0.0450   y: +0.0935   z: +0.0360
Last selected object orientation: a: -090.00   b: -000.0000   g: -090.00
```

Fig. 2. Complete Ball-Plate System.

B. *Ball-and-plate Controllers Design*

1) Plant Model

Assuming that the ball does not slide and is always in contact with the surface, and complete symmetry in the system, a non-linear model describing the behavior of the plant is given by [10].

$$\begin{bmatrix}\dot{x_1}\\ \dot{x_2}\\ \dot{x_3}\\ \dot{x_4}\end{bmatrix} = \begin{bmatrix} x_2 \\ a\left(x_1\dot{U}_x^2 + x_3\dot{U}_x\dot{U}_y - g\,sin(U_x)\right) \\ x_4 \\ a\left(x_3\dot{U}_y^2 + x_1\dot{U}_x\dot{U}_y - g\,sin(U_y)\right)\end{bmatrix}, \qquad (1)$$

where x_1 and x_3 are the position and velocity along the x-axis, respectively, while x_2 and x_4 are the same variables on the y-axis. The inputs U_x and U_y are the angular positions of the servomotor axes, the value of $a = 5/7$, for this case, meets the physical characteristics of the plate (dimensions, weight, inertia), and $g = 9.8\ m/s^2$ is the gravity.

To design a lineal control, the linearization of the system at the origin point (0,0,0,0) results in

$$A = \begin{bmatrix} 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0\end{bmatrix} B = \begin{bmatrix} 0 & 0\\ -ag & 0\\ 0 & 0\\ 0 & -ag\end{bmatrix}, \qquad (2)$$

where the same separability characteristic of the original MIMO model is appreciated. Notice that the system can be split in two equal SISO systems of order two. The system outputs are the ball positions on the x-axis (x_1) and on the y-axis (x_3).

2) Discrete PID Controller

As an initial point of comparison, a discrete PID controller is designed using two independent loops (one for each motor). The tuning is performed in a single subsystem and replicated for the second joint given the assumed symmetry in the plant. The standard discrete control structure is defined as

$$U(z) = \left[k_P + \frac{k_I}{1 - z^{-1}} + k_D(1 - z^{-1})\right]E(z), \qquad (3)$$

where k_P , k_I y k_D are the discrete proportional, integral and derivative constants, respectively, $U(z)$ is the control signal and $E(z)$ the error of the output against the desired reference. The the controller parameters are obtained for low overshoot and a short set-up time, relative to the trajectory time in the simulation results. The final values are $k_P = -0.4626$, $k_I = -0.0029$, and $k_D = -5.3064$.

3) Discrete LQR Controller

A discrete lineal quadratic regulator is characterized by being an optimal, robust, and easy to implement strategy with state feedback. Its operation is based on minimizing the function

$$J = \sum_{k=0}^{\infty}(x_k^T Q\, x_k + u_k^T R u_k), \qquad (4)$$

where k is the index of discrete time, Q is a symmetric definite positive matrix that establishes the priorities in the states, while R establishes the importance of the control signal (energy expended) to obtain the optimization objective. To avoid steady-state error, we add an integrator for each one of the outputs, increasing the total order of the system by two. However, for the separate plant design, each subsystem with an additional integrator is chosen and the three feedback gains are calculated (two original states plus the integral error as the new state), as it is proposed in [11].

Based on simulation calibrations, the values used for the calculation of the feedback constants are presented in Table 1. All state variables are assumed measured using the properties from the remote API.

TABLE I. LQR PARAMETERS AND CONSTANTS

LQR constant and input parameters	Value
Q	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 500 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}$
R	100
K	[-0.5779 -1.1352 -0.0150]

C. Ball-and-Beam Mechanical Design in V-REP

Similarly to the previous example, we build a pure form (Figure 3) and a polished scheme with CAD-imported shapes (Figure 4) for a ball-and-beam plant. The models also take into account the grouping of the imported parts and the configuration of pure forms with the native language in V-REP.

With respect to the actuators, the plant has only a motor that moves the rail as Figure 4 shows. The articulation is configured in the torque-force mode, while the engine is arranged such that the input is speed angular (a different mode compared to the motors in the previous plant). For the measurement of the position of the ball in the rail, we use a proximity sensor located at the left end of the rail.

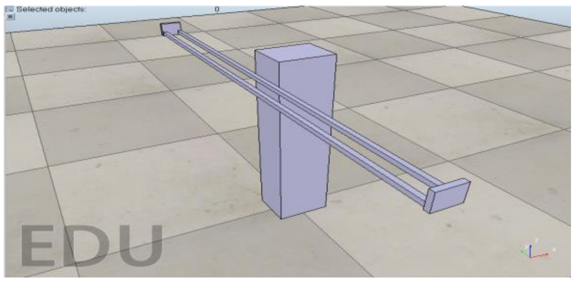
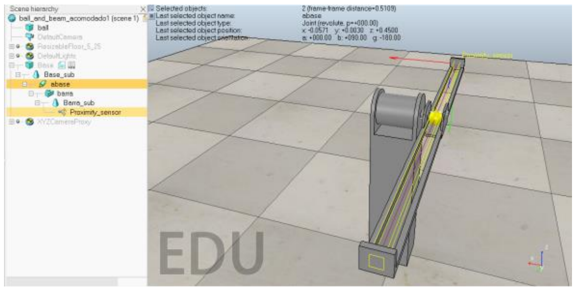


Fig. 3. Initial structural model of the ball-and-beam system in the main environment of V-REP.



Selected objects: 2 (frame-frame distance=0.5109)
Last selected object name: abase
Last selected object type: Joint (revolute, p=+000.00)
Last selected object position: x: -0.0571 y: +0.0030 z: +0.4500
Last selected object orientation: a: +000.00 b: +090.00 g: -180.00

Fig. 4. Complete ball-and-beam system.

D. Ball-and-plate Controllers Design

1) Plant Model

A typical fourth-order non-linear model is defined by [12]

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} x_3 \\ x_4 \\ \dfrac{(m_b\, x_1 x_4^2 - m_b g \sin x_2)}{\left(m_b + \left(\frac{I_b}{R^2}\right)\right)} \\ \dfrac{\left(u - (m_b g x_1 \cos x_2) - (2 m_b x_1 x_3 x_4)\right)}{\left(I_v + (m_b x_1^2)\right)} \end{bmatrix}, \qquad (5)$$

where x_1 and x_3 are the ball position and velocity, respectively, while x_2 and x_4 are the angular position and velocity of the motor. The input u is the torque of the motor, m_b and m_v are the ball and beam mass, I_b and I_v are the inertia moments of the ball and beam, L_v is the beam length, g is the gravity and R is the ball ratio. The values of the parameters are in Table 2.

TABLE II. BALL AND BEAM SYSTEM PARAMETERS

Parameter	Unit of measurement	Value
I_b	(K_g/m^2)	$4.624*10^{-5}$
I_v	(K_g/m^2)	0.0833
m_b	(K_g)	0.1
m_v	(K_g)	1
L_v	(m)	1
R	(m)	0.0340
g	(m/s^2)	9.8

The linearization of the system at (0,0,0,0) is

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \dfrac{-gm_b}{\left(m_b + \left(\frac{I_b}{R^2}\right)\right)} & 0 & 0 \\ \dfrac{-gm_b}{I_v} & 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \dfrac{1}{I_v} \end{bmatrix}. \qquad (6)$$

2) Discrete LQR Controller

For this simpler plant (compared to the previous one), we only consider the LQR controller to observe the simulation and virtual-plant results. The design of the LQR controller with integrator is similar to the one presented for the ball-and-plate system. The tuning values and the feedback constants are presented in Table 3. The constant *K(5)* corresponds to the gain for the integral of the error (the augmented state variable). All state variables are also measured in this plant with the properties obtained from the remote API.

TABLE III. LQR PARAMETERS AND CONSTANTS

LQR constant and input parameters	Value
Q	$\begin{bmatrix} 300 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 800 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0.1 \end{bmatrix}$
R	10
K	[-5.5379 10.0166 6.1597 1.2323 -0.0557]

IV. RESULTS AND DISCUSSION

The results of both plants are analyzed by comparing the response of systems to the designed controllers in two scenarios: *i)* using ideal models in a Simulink simulation, and *ii)* with the interaction of the controllers in Matlab regulating the ball position in the designed virtual plants through the co-simulation API.

A. Analysis of the ball-and-plate system.

To test this plant, we propose to follow a simple trajectory, starting in rest at position (0,0) cm, then changing set point every 40 seconds to (-5,-5), (-5,5), (5,5), (5,-5) and (-5,-5) centimeters.

The responses for the LQR controller are shown in Figures 5, 6 and 7, detailing the position of the ball in time, the control signals to the actuators, and the trajectory in the plate. Note that the position in every axis is achieved without steady-state error due to the additional integrator in the design. Moreover, the differences on the simulation and the virtual plant are not notorious in Figure 5, but they are remarkable in the control signals and in the trajectories (Figures 6 and 7).

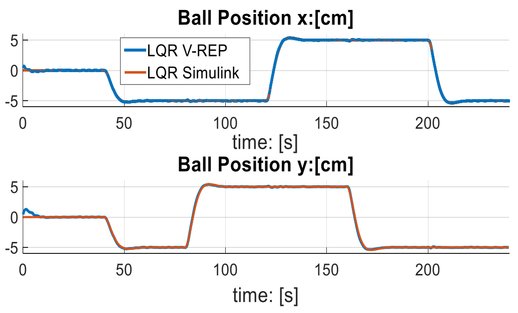


Fig. 5. Ball position response using a LQR.

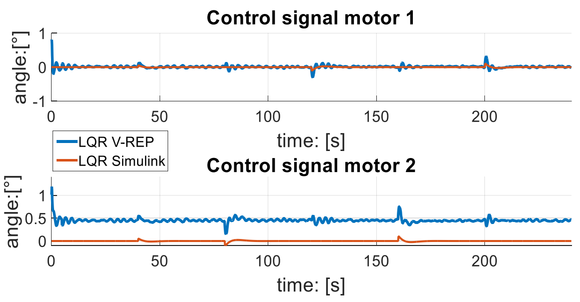


Fig. 6. Motors control signal using a LQR.

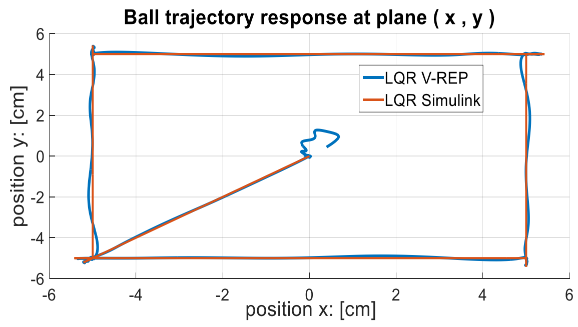


Fig. 7. Ball trajectory response using a discrete LQR.

From Figure 6, it is noticeable that the control signal of the motor 2 in the virtual plant differs clearly from the simulation. This response is due to the fact that the plate in the virtual plant is not perfectly aligned to the origin, and the controller tries to compensate this lag by remaining at a constant value to equilibrate the table in a horizontal position. This lag causes the ball not to start at (0,0) cm (Figure 7), since it tends to slide to the positive side of the plane (x, y). However, despite the parameters contemplated by the virtual environment (friction, gravity, and contact between the ball and the plate) the general behavior of the virtual plant in V-REP is similar to simulation in Simulink.

On the contrary, the responses for the discrete PID controller, are quite different as it shown in Figures 8, 9 and 10. Although in Figure 8 the simulation show smooth changes in the set points, the actual response in the virtual plant presents overshoots and oscillations. This behavior is more evident in Figure 10, where the trajectory followed by the real plant is completely distinct to the one presented in simulation.

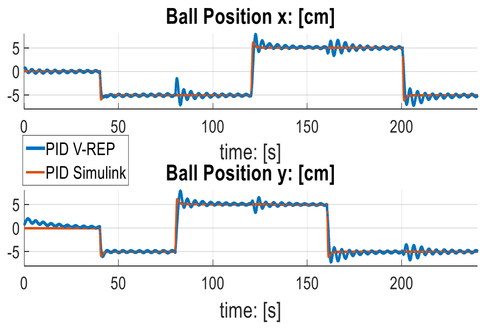


Fig. 8. Ball position response using a discrete PID.

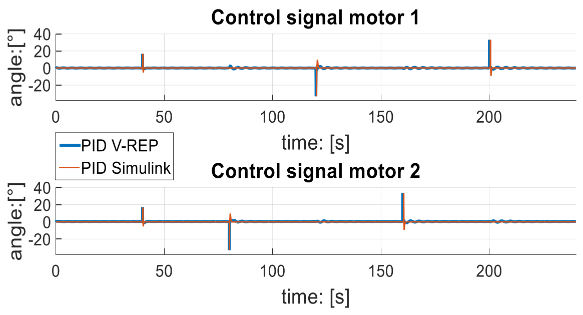


Fig. 9. Motors control signal using a PID.

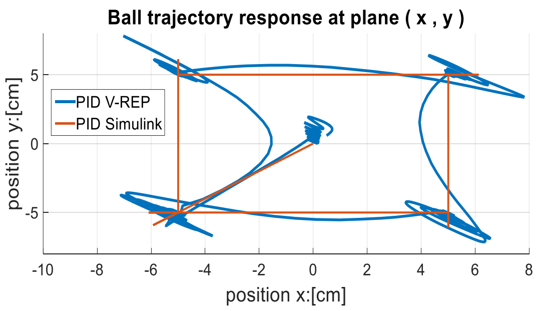


Fig. 10. Ball trajectory response using a discrete PID.

Figures 8 and 10 point out that the aforementioned initial displacement of the table is also presented, since the ball does not start at the origin. The compensation of the controller is presented in Figure 9, where the control signal for the virtual plant is higher than the one in the LQR controller (achieving peaks between 30 and -30 degrees). The PID controller in this case is more aggressive than the LQR, and then, the friction and slide of the ball in the virtual environment cause large differences with the simulation. In addition, these differences are more remarkable in this case due to the low robustness of the PID-controller design.

Comparing the controllers, the LQR controller presents a higher performance in robustness, set time, control signal and steady state error over the PID controller. The oscillations of the PID controller delay reaching the desired reference, while the abrupt changes on the control signal allow the ball to slide, increasing the errors.

B. *Analysis of the ball-and-beam.*

In this case, the tests for the simulation and the virtual plant start with the steady ball at 0 cm (in the center of the beam). Then the set point is changed every 36 seconds to 20 cm, 40 cm, -40 cm, 10 cm, and -30 cm. The results are displayed in Figures 11 and 12 for the ball position and the motor velocity (control signal), respectively.

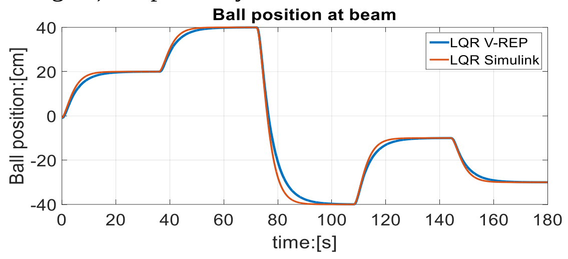


Fig. 11. Ball-Beam System Ball position response using a LQR.

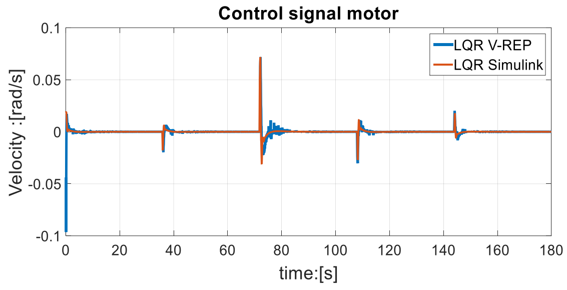


Fig. 12. Ball-Beam System motor control signal using a LQR.

From Figure 11, the virtual plant in V-REP takes a slight longer time to establish in comparison with the simulated system. However, the physical non-modeled factors as friction and contact between the ball and the beam, are more evident in the control signals shown in Figure 12. Here, the motor requires a stronger actuating effort at the beginning, and with more oscillations in the set point changes to guide the ball. The virtual plant must initially compensate for the action of gravity that caused the ball to be out of phase. However, the responses of the simulation in V-REP and Simulink are similar, showing the robustness of the LQR controller.

V. CONCLUSONS AND FUTURE WORK

This work presents a methodology to construct virtual plants in V-REP and the modification of a remote API to communicate the real environment with controllers implemented in Matlab. This co-simulation platform facilitates the teaching-learning process in control, allowing the education institutions to use different plants with most of the real interactions that usually are not present in model simulations.

To show de application of the method, we have presented the design and control of two mechanical plants (ball-and-beam and ball-and-plate). Although the differences between simulation and the co-simulation platform are remarked using simple controllers, the results can be used to re-tune the regulation strategies or to choose more appropriate control methods.

As future work, we are working on more complex plants such as a Furuta pendulum and adaptive environments (scenarios and obstacles) for trajectories planning with holonomic robots. Moreover, several V-REP features can be explored in order to propose more interactive and realistic plants for the application of distinct control techniques.

REFERENCES

[1] H. Vargas, D. Dormido, N. Duro, D. Dormido-Canto, "Creación de laboratorios virtuales y remotos usando easy java simulations y Labview", XXVII Jornadas de Automática, Almería, España, pp. 1182-1188, Sep, 2006.

[2] S.Dormido, "Control learning: Present and future" Annual Control Reviews, vol.28, pp.115-136, 2005.

[3] D.Guillet, A.Nguyen., Y.Rekik, "Collaborative Web-Based Experimentation inFlexible Engineering Education", IEEE Trans on Education, vol. 48, no. 4, 2005.

[4] V.H. Andaluz, F.A. Chicaiza, C. Gallardo, W.X. Quevedo, J. Varela, J.S. Sánchez, O. Arteaga, "Unity3D-MatLab Simulator in Real Time for Robotics Applications", Augmented Reality, Virtual Reality, and Computer Graphics: Third International Conf., Lecce, Italia, pp. 246-263, Jun, 2016.

[5] R.D. Vásquez, H.O. Sarmiento, D.S. Muñoz, "Propuesta de implementación de plantas virtuales para la enseñanza de programas de control lógico", *ACOFI*, vol.11, no.22, pp. 1-13, 2016.

[6] A. Domínguez, "Modelado y Simulación de un Robot LEGO Mindstorms EV3 mediante V-REP y Matlab", Proyecto final de carrera, Dep. Informática, Universidad de Málaga, 2016.

[7] Coppelia Roboticts V-REP, V-REP User Manual, available at: http://www.coppeliarobotics.com/helpFiles/, Consultated Apr 2019.

[8] Blueworkforce bluezero, V-REP shapes, available at: https://blueworkforce.github.io/bluezero/v1/, Consultated Apr 2019.

[9] Solidworks Corporation, "Guía del estudiante para el aprendizaje del software Soidworks", 2011 available at: https://www.solidworks.com/sw/docs/Student_WB_2011_ESP.pdf

[10] A. Cedeño, M. Gordón, L. Morales, "Control de posición y seguimiento de caminos en el sistema Bola-Plataforma", XXVII Jornadas de Automática, Quito, Ecuador, vol.27, pp. 47-53, 2017.

[11] K. Ogata, "Discrete-Time Control Systems", 2nd edition, Ed. Prentice Hall Inc., USA, 1995, pp. 596-609.

[12] J. Huang, C.F. Lin, "Robust nonlinear control of the ball and beam system," Proceedings of the American Control Conference, Seattle, USA, pp. 306-310, Jun ,1995.