REAL-TIME ROBOT CONTROL IMPLEMENTATION WITH MATLAB/SIMULINK

A. Valera, M. Vallés, J. Tornero

Dpto. Ingeniería de Sistemas y Automática
Universidad Politécnica de Valencia
Camino de Vera 14, 46022 Valencia (Spain)
E-mail: giuprog@isa.upv.es, mvalles@isa.upv.es, jtornero@isa.upv.es

Abstract: This presents a hardware platform and a programming environment for the design, analysis and implementation of industrial robots. The platform is compound with two industrial robots and a distributed computation software architecture based on PCs. For the programming environment a specific library of nonlinear robot controllers has been implemented and tested. The environment is developed over MATLAB/Simulink which provides modeling, simulating and the analyzing facilities for testing robot control strategies. The automatic generation of program code is carried out by the Real-Time Workshop and the Real-Time Windows Target toolboxes also provided by MATLAB. These toolboxes allow obtaining an implementation of the controllers over several Operating Systems. Finally, two industrial robotics have been used as physical set-up for testing the algorithms and the proposed architecture, as well as the facilities of the program environment. *Copyright © 2001 IFAC*

Keywords: Robot control, Nonlinear control, Real-time systems, Computer controlled systems, Industrial robots, Simulation

1. INTRODUCTION

Over last years, some tools have been developed in order to provide facilities for the automatic generation of code corresponding to controllers previously designed. The simulation tool MATLAB also has provided several tools for doing this: the Real-Time Workshop and the Real-Time Windows target toolboxes.

In this paper, these tools have been used to implement control strategies for two industrial robots: a gantry robot and a PUMA 560 industrial robot, but it can be applied to any other kind of robot. The only requirement is that the robot signals (such as the robot position, velocity and control actions) have to be available.

Robots are widely used in the industry mainly in order to increase productivity in many areas of manufacturing processes, as well as the quality of industrial products. Many industrial applications involve complex robot tasks, which require precise robot control. In this sense, control engineers must understand, develop and implement new control algorithms in order to fit new requirements.

Therefore, control engineers are forced to introduce new tools to better understand and solve questions related with the robotic systems, in general, and the robot control in particular.

Focused on the implementation of robot control systems, a robot nonlinear control library has been designed. This library is based on Matlab/Simulink, and it offers more than 25 different non-linear controllers, grouped in four classes: passivity, computer torque, adaptive and controller-observers. It is also very simple to test and develop new techniques to expand the library with other controllers based on the same classes or other different.

Once controllers are simulated, analyzed and validated, real-time code programs for implementation are generated using Matlab toolboxes.

2. REAL-TIME CONTROL IMPLEMENTATION WITH MATLAB

As it is well-known, Matlab is a technique computing environment for high-performance numeric computation and visualization. A family of toolboxes has extended Matlab functionality in order to solve particular classes of problems. One of these toolboxes is Simulink, that it is a software package used for modeling, simulating, and analyzing dynamical systems with graphical user interface. With Simulink, models are built as block diagrams using click-and-drag mouse operations, and it includes a comprehensive block library of sink, sources, linear and nonlinear components, etc.

In addition, Matlab/Simulink has another toolboxes for the implementation of real-time control programs directly from Simulink models. These toolboxes can automatically build programs that can be run in a variety of environments, including real-time systems and stand-alone simulations.

The first toolbox is Real-Time Workshop (RTW). RTW is an open and extensible system designed for use with a wide variety of operating environments and hardware types. With the RTW, an executable program (that can be run in real-time on a remote processor) is directly generated from the Simulink control scheme used for simulation. To do that, the user must only delete the system model, and add the S-Function blocks to access to the physical system information: the analog and digital inputs/outputs. After that, the executable program can be obtained with the push of a button: the **Build** button on the Real-Time Workshop page of the *Simulation Parameters* Simulink model dialog box.

For the automatic code generation the environment where to place the generated code must be chosen (generic real-time, DOS real-time, Tornado real-time etc.). To do this, RTW invokes the Target Language Compiler (TLC) that consists of: the entry point or main file (the system target file), a set of block target files that specify how to translate each block of the model into target-specify code and the TLC function library. To create a real-time executable program the *make* utility is invoked. This utility compiles and links the generated code. The *make* utility can be configured for your real-time system modifying the template makefile and the system target file.

The last step is to download the executable program to the target hardware and run it. When the execution is finished, a *mat* file (with the more important variables) can be created in order to analyze the

The second toolbox for the real-time control implementation is Real-Time Windows Target (RTWIN). RTWIN is a real-time kernel that interfaces with the Microsoft Windows operating system. This kernel enables to assign the highest priority of execution to the real-time executable, which is created by the Real-Time Workshop.

Normally the kernel remains idle. Only during real-time execution of the model the kernel intervenes to ensure that the model is given priority to use the CPU to execute each model update at the prescribed sample intervals. Once the model update at a particular sample interval completes, the kernel releases the CPU to run any other Windows application that may need servicing. The generated code, using I/O boards, captures a sample of data from one or more input channels, uses the data as inputs to the model, processes the data, and sends it back to the outside world using a output channel of the I/O boards.

Simulink's external mode enables to observe the behavior of the real-time system, and it also allows altering parameters by simply editing the block diagram while running Simulink in external mode. New parameter values are automatically transferred to the compiled version of the model during real-time execution.

RTWIN creates a real-time model (with the required I/O boards) by clicking the Build button on the Real-Time Workshop page of the Simulink Parameters dialog box. After that, the model can be tested directly. To start execution the **Connect to Target** of the Simulation pull-down menu must be selected. It allows starting the execution (**Start real-time code**) and stopping the execution (**Stop real-time code** or **Disconnect from target**).

To implement the real-time programs, RTWIN includes a small block library consisting of a source block (RT In), a sink block (RT Out), and an Adapter block. The purpose of this block is to provide a reference to a particular I/O board. Each instance of an Adapter block must select a I/O board and the setting parameters (the base address, gains, input and output digital ports etc.). In addition, the RT In and RT Out bocks has a entry for the particular Adapted block that it is associated with, and normally it provides a entry for the sample time and the channels of the board.

3. NONLINEAR MOTION ROBOT CONTROL

In recent years, advanced motion robot control has gained a lot of attention. Increasing demands on the robot systems performance has led to the development of various control methods. These controllers can mainly be grouped under four different approaches: passivity method, the controllers based on inverse dynamics (or computer torque), design of controller-observers and adaptive controllers.

This passivity-based approach solves the robot control problem by exploiting the robot system's physical structure, and specifically its passivity property. The design philosophy of these controllers is to reshape the system's natural energy in such a way that the tracking control objective is achieved (Paden and Panja, 1988).

In the library and for the point-to-point problem (regulation), the library controllers based on the passivity can be viewed such as particular cases of the next general control law:

$$\tau_e = -K_p e - K_d v - u \tag{1}$$

These controllers implement basically a proportional ($e = q - q_d$) and a derivative (v) action with the compensation of the gravity term (u). If the velocity signal is no available, an estimation of it can be obtained using a differential approximation or a linear compensation. In addition, we can change the gravity compensation by an integral action of the error signal.

For the passivity-tracking problem, the kinetic and potential energy must be changed in a desired way. The general expression of the controllers that can be found in the literature is the next:

$$\begin{aligned} \tau_e = {} & M(q,\theta)a + C(q,\dot{q},\theta)v_1 + \\ & + G(q,\theta) - K_p e_1 - K_d v_2 \end{aligned} \tag{2}$$

In all these controllers, the control law has two parts, a robot dynamics compensation and a proportional and differential action.

The second group of the controllers is the controllers based on the inverse dynamics (or computer torque controllers), and they can be included in the feedback linearization problem. A system is feedback linearizable (the rigid robot systems have this property) if exist a state-space transformation and a regular static state feedback that transform the non-linear system into a linear one (Slotine and Li, 1991). The general expression of this kind of controller, is the next:

$$\tau = M(q,\theta)v + C(q,\dot{q},\theta)\dot{q} + G(q,\theta) \tag{3}$$

For that implementation of the controllers, an exact knowledge of the system is required. But in the practice usually there are robot model uncertainties. In order to relax the perfect model knowledge, adaptive procedures can be included. The adaptive control approach is to design a controller that can adapt itself to the process uncertainties, so the control specifications are achieved (Bayard and Wen, 1988), (Berghuis, 1995). The adaptive controllers implemented in the library have the next general expression:

$$\begin{aligned} \tau = {} & M_0(u)a + C_0(u,v)v_1 + G_0(u) + \\ & + Y(u,v,w,x)\hat{\theta} - K_p e - K_d v_2 - K_J \|e\|^2 s_1 \end{aligned} \tag{4}$$

In these controllers, the control law has three parts, the compensation of the known part of system dynamics, the adaptive part and the linear static state feedback part.

The model-based robot controllers need access to the robot position and velocity. But in recent years, it has been a trend to omit velocity-sensing equipment in the robots. In order to control the robot systems using only position measurements, several controller-observers have been modelled (Nicosia and Tomei, 1990). As in the last case, a general expression has been obtained:

$$\begin{aligned} \tau_e = {} & M(q,\theta)a + C(q,\dot{\bar{q}},\theta)v_1 + \\ & + G(q,\theta)w - K_p e - K_d v_2 \end{aligned} \tag{5}$$

4. REAL-TIME ROBOT CONTROL IMPLEMENTATION

In order to make easy the implementation of the robot controllers, the *Robot Nonlinear Control Library* (Valera, et al., 1999) is used. This library is based on Matlab/Simulink and has more than 25 different non-linear controllers grouped in the four classes presented in the previous section. For each group of controller, a general expression has been obtained and the corresponding control law implemented (Valera, 2000).

Models of different industrial robots are also provided by the library that can be used for analyzes, simulation and control design. They obey the general equation of motion for an n-axis manipulator given in Equation (6).

$$\tau = M(q,\theta)\ddot{q} + C(q,\dot{q},\theta)\dot{q} + G(q,\theta) \qquad (6)$$

where:

M(.): symmetric inertia matrix
C(.): vector of Coriolis and centrigual torques
G(.): vector of gravitational torques
q: vector of generalized coordinates
τ: vector of generalized forces

In addition, new robots can be modeled using basically three *Matlab functions* (the *gravity.m*, *coriolis.m* and *inertia.m*) and two integrators like it can be seen in Figure 1.

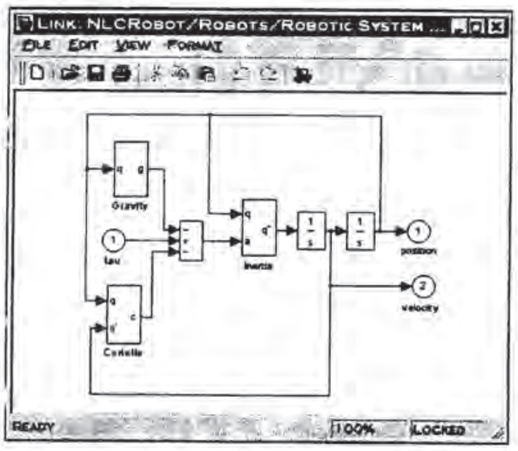


Figure 1: Robot model

Users can easily create those new model blocks and connect them with the desired controller for simulation. Some simulation parameters must be fixed, like the start and stop time, the value of the gain matrix of the nonlinear controllers.

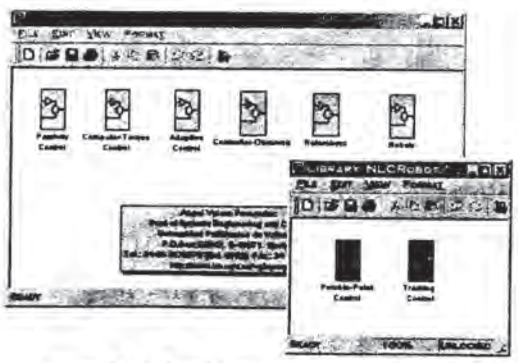


Figure 2: Robot Nonlinear Control Library

Once the good functionality of the control system has been reached and validated by simulation, an executable program can be obtained. Any programming language can be used for this purpose. However, and in order to simplify this manual process, an automatic code generation process has been implemented using the RTWIN and RTW. These toolboxes use directly the Simulink schemes, and provide a real-time development environment.

In the work, two industrial robots are now used. The first is an industrial gantry robot with three degrees of freedom: the X-Y-Z axes, with 2.5, 1.5 and 1 meters respectively. The second robotic system used in the laboratory is a PUMA 560 which is a 6-axis industrial robot widely used in the industry. The original control modules of these two robots compound several modules, like the power module, the amplifier module, the control module, I/O modules, etc.

Control units for industrial robots trend to be totally closed systems and their robot manufacturers are reluctant to provide information. In addition, controllers are difficult to modify given that they use their own operating system. So, it is impossible to modify this control strategy (not even the controller gain values). As a consequence, users have serious problems to implement conventional as well as advanced control strategies (such force control), cooperative control of several robots, automatic trajectory generation, control based on external sensing (such as vision), etc.
In order to no reproduce these limitations, the control stages of the remote laboratory industrial robots have been implemented been implemented based on PC. In this way, the PC has access to the position and velocity signals, and generates the adequate control actions to move the different elements of the robots. The new control architecture is depicted in Figure 3, and extensively described in the reference (Valera, et al., 1998).

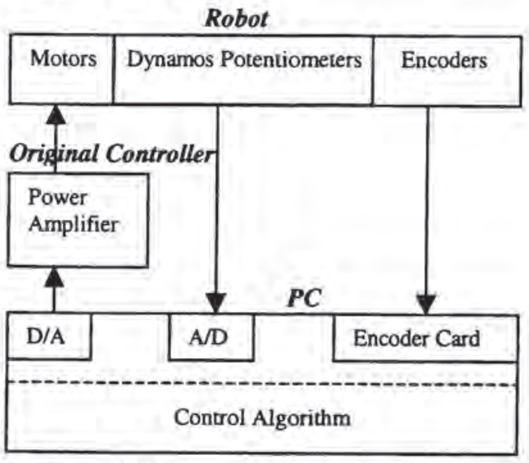
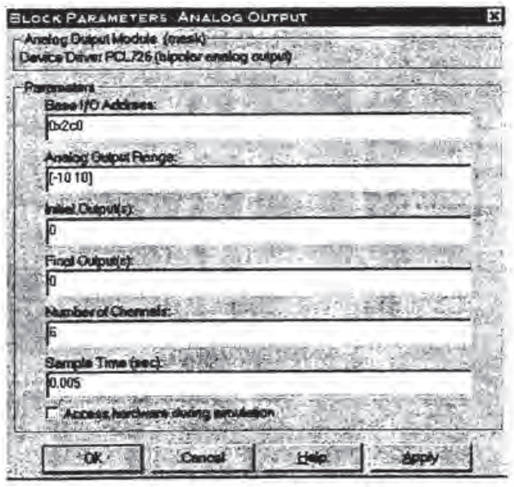


Figure 3: Open architecture control unit

To implement this control architecture, three kinds of data acquisition cards have been used: an *Advantech™ PCL-818* card in order to obtain the analog robot outputs; the *PCL-726* for supplying the control actions; and the *PCL-833* for reading the encoders. The proposed control architecture gives two advantages, first the simplicity and second the low cost

For connecting the industrial robots with the Simulink model, several *S-Functions* have been programmed in C language. These *S-Functions* read from the external data modules (the robot position and velocity, etc.) and write data into Simulink input blocks. Others *S-Functions* write data in output Simulink blocks (the control actions, etc.). In this way, the generation of real-time robot control system software is easy and simple because of the well-known Simulink properties to construct model controllers and the prototyping of these controllers using the RTWIN and RTW software. Figure 4 shows the mask for the PCL-726 card. In these masks you usually must indicate the hardware base address, the number of channels, the output ranges, the sample time, etc.


Block Parameters: Analog Output

Analog Output Module (mask)
Device Driver PCL726 (bipolar analog output)

Parameters
Base I/O Address:
0x2c0
Analog Output Range:
[-10 10]
Initial Output(s):
0
Final Output(s):
0
Number of Channels:
6
Sample Time (sec):
0.005
☐ Access hardware during simulation

Figure 4: Matlab mask for analog outputs

Before the generation of the real-time executable program, a parameters tuning stage is carried out. For that purpose, the RTWIN toolbox is very useful because the controller can be taken directly (from the Robot Nonlinear Control Library) and put it into the model with the *S-Functions* input/output interfaces. After compiling and building the model, it can be executed just pushed of two buttons: **Connect to Target** and **Start real-time code**. With this executable program, all the important signals from the robot can be observed using the Scope Simulink block. This can be used as additional information in order to modify on-line the parameters of the robot controller and analyze the effects of these changes. Figure 5 shows the RTWIN model and a real execution signal (using the scopes blocks) with the gantry robot.

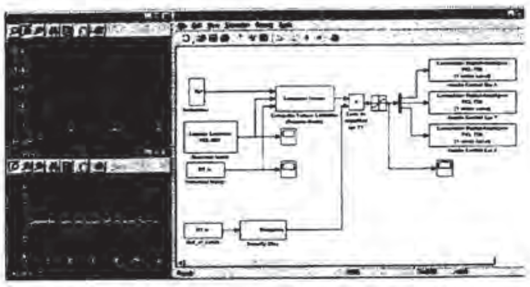


Figure 5: Robot controller with RTWIN

Once the best controller parameters are fixed, the executable program is automatically obtained. In this case the RTW toolbox provides it after pushing the **Build** button on the Real-Time Workshop page of the *Simulation Parameters* Simulink model dialog box.

A distributed computation software architecture has been implemented with Matlab package and its Simulink, Real-Time Workshop and Real-Time Windows Target extensions. In additions, the *Robot Nonlinear Control Library* is also integrated in the software structure.

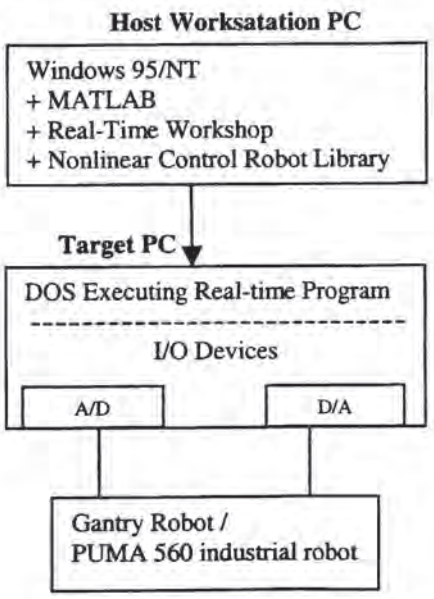


Figure 6: Robot control architecture

This architecture has a group of computers with Matlab package and its Simulink, Real-Time Workshop and Real-Time Windows Target extensions. In additions, the computers have also the *Robot Nonlinear Control Library* described before. After RTW generates the executable program, the following step is to download the program into the computer that will implement the control with the data acquisition cards previously enumerated. The program transference is very simple and direct carried out through the computer network.

While the control program is running, a data file is being generated. The file (with ".mat" format) has all the variables selected from the Simulink scheme. For example, figures 7 and 8 show the gantry robot real response (positions and velocities) with Computed-torque controller obtained with the RTW.

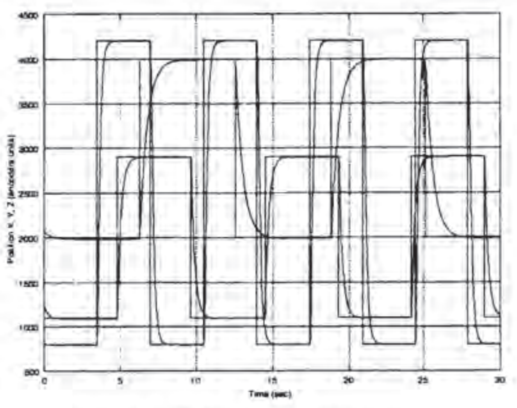


Figure 7: Robot joints positions

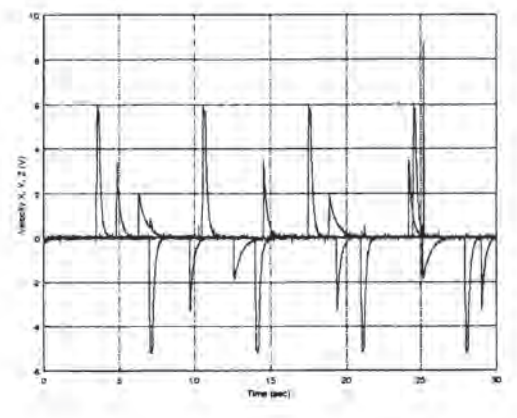


Figure 8: Robot joints velocities

5. CONCLUSIONS

This paper has presented a hardware platform and programming environment for the design, the analysis and the implementation of industrial robots. The platform is compound with two industrial robots and a distributed computation software architecture based on PC.

For the programming environment a specific library of nonlinear robot controllers has been implemented and tested. The library provides more than 25 different nonlinear controllers grouped in four classes of robot controller. The used of the Matlab/Simulink provides a very easy and interactive platform for modeling, simulating and the analyzing robot control strategies. In addition, it is very simple to implement existing controllers or even to build new ones. In the automatic generation of the program code corresponding to selected controllers, the Real-Time Workshop and the Real-Time Windows Target toolboxes provided by the MATLAB have been used.

Finally, two industrial robotics have been used as physical set-up for testing and analyzing the properties of the controllers generated using the proposed architecture, as well as the facilities of the program environment.

6. ACKOWLEDGEMENTS

This work has been partially supported by the Vicerrectorado de Investigación y Desarrollo Tecnológico (Universidad Politécnica de Valencia) PPI-5-99-19990575, and by the Spanish Government (Secretaria de Estado de Universidades, Investigacion y Desarrollo) TAP1999-1080-C04-04.

REFERENCES

Bayard, D.S. and J.T. Wen (1988). New Class of Control Laws for Robotic Manipulators: Part 2. Adaptative Case. *Int. J. of Control, Vol.47, pp.* 1387-1406

Berghuis, H. (1995). *Model-based Robot Control: from Theory to Practice.* Ph. D Thesis. Dept. of Applied Mathematics, Univertity of Twente, Enschede, The Netherlands.

Nicosia, S. and P. Tomei (1990). Robot Control by Using Only Joint Position Measurements. *IEEE Trans. on Automatic control*, **Vol. 35**, pp. 1058-1061

Paden, B., R Panja (1988). Globally Asymptotically Stable 'PD+' Controller for Robot Manipulators", *Int. J. on Control*, **Vol. 47**, pp. 1697-1712

Slotine, J.J.E. and W. Li (1991) *Applied Nonlinear Control.* Prentice-Hall Int. Editions, ISBN 0-13 040049-1

Valera, A. (2000). *Análisis Comparativo de Técnicas de Control de Robots Rígidos y Flexibles.* Ph.D Thesis, Ed. Universidad Politécnica de Valencia, ISBN 84-699-2911-9

Valera, A., J. Vergara, J. Tornero and E. Garcia (1998). Control a PUMA 500 using a new open architecture. In *3rd Portuguese Conf. On Automatic Control.* Coimbra (Portugal).

Valera, A., J. Tornero and J. Salt (1999). A Robot Control Toolbox for Matlab/Simulink. In: *13th European Simulation Multiconference*, pp 260-264