DESIGN OF DC MOTOR SPEED CONTROL THROUGH PROCESSOR-IN-THE-LOOP APPROACH

S. Lentijo, A. Monti, and R. Dougal
Department of Electrical Engineering
University of South Carolina
Columbia, SC 29208, USA
E-mail: [lentijo, monti, dougal]@engr.sc.edu

KEYWORDS
Rapid control prototyping; Processor-in-the-Loop; Virtual Test Bed.

ABSTRACT

Due to the rapid development of digital processor technology, various methodologies for design, development, implementation and testing of digital control systems have been proposed. In most cases, the systems are soft real-time or even non real-time, hardware-dependent or based on proprietary solutions. In this paper, we focus on a testing procedure for a microcontroller-based control, which will be adopted for software testing. The procedure is performed with no real-time constraints, but is capable to give information about the real-time conditions. It is called Processor-in-the-Loop (PIL) and it is illustrated by the design and implementation of a DC motor speed control system; simulation and experimental results are then presented and analyzed.

1. INTRODUCTION

Processor-in-the-Loop (PIL) allows one to test the actual control software running on a dedicated processor that controls a virtual prototype of the plant. PIL simulation provides an intermediate stage between simulation and deployment, where the Virtual Test Bed Platform (VTB) [1] models the plant, while code generated for the controller subsystem runs on the actual target hardware. The PIL simulation does not run in real-time, because the plant model runs on the host PC (Windows OS) and only the controller runs on the target hardware. The communications between them are via a serial link or USB port.

In this paper, we focus on a testing procedure to control a DC motor, where the DC motor is modeled in VTB and the control algorithm is implemented in an Infineon microcontroller. Also high level visualization is used to help the user rapidly comprehend the system performance, in this particular case the DC motor emulates the power transmission of a scale car; the user can not only set the speed reference of the car but can also control the steering using a 3-D visualization engine called VXE [2]. This on-line interaction between the user and the system is made possible by the VTB PIL features, where management of the electrical variables produces a concrete and visible mechanical effect. In other words, the VTB PIL procedure allows the user to make subtle changes during the simulation, without having to stop and reset variables before proceeding with new parameters.

2. PROCESSOR-IN-THE-LOOP

During PIL simulation (Figure 1), VTB simulates the plant model for one sample interval and exports the output data (output of the plant) to the control system under test via a serial communications link. When the target processor receives signals from the plant model, it executes the controller code for one sample step. The controller returns the output signals (output of the controller) computed during this step to VTB, via the same communications link. At this point one sample cycle of the simulation is complete and the plant model proceeds to the next sample interval. The process repeats and the simulation progresses.

![Figure 1. PIL Simulation.](image)

Implementing the PIL procedure through VTB offers a number of advantages over other platforms, such as:
The VTB blocks describing the plant do not depend on the processor used for the control. As long as there is a serial link or USB port and an interface protocol, the system can work.

The software is modular and it is easy to customize the communication to any number of inputs and outputs.

Thanks to the management of a digital output pin, a precise measurement of the computational time is also possible. This information about the real computational time in the controller can be shown as a viewable in VTB. Also, thanks to the new capabilities of VTB [2], this computational time feature can be used to create a delay in the simulation, in order to take into account hardware delays and thereby make the simulation more accurate.

Another interesting feature that PIL offers is the possibility of running the controller with an independent sample rate. This means that VTB does not have to call the control algorithm every time step; this can be done every n seconds that the user defines. The variable time step in VTB makes this possible.

3. SYSTEM STRUCTURE

The experimental system of PIL can be divided into two major parts by function.

1) Windows VTB (non real-time)
2) Control Hardware

The structure of the PIL simulation system is illustrated in Figure 2.

![Figure 2. Structure of PIL simulation system](image)

3.1 Windows VTB

The Virtual Test Bed (VTB) is a software environment that has been developed for design, analysis, and virtual prototyping of large-scale multi-technical systems.

Within the context of "virtual prototyping", we include not only simulation of system dynamics, but also solid modeling of the system and visualization of the system dynamics. As a high-level virtual prototyping tool, the VTB program addresses many challenges in a field such as power electronics, which encompasses a wide range of disciplines, including analog electronics, digital electronics, power systems, controls, electro-mechanics, and mechanical systems. The complexity of power electronics systems inevitably bridges several areas of technical expertise and the engineers in each of those technical areas traditionally work with their own set of design and simulation tools.

The VTB environment addresses these challenges by supporting:

1) Multi-formalism

Different languages can be used to build models of the different components of a system. This allows an individual to build models using the preferred language within his or her discipline.

2) A Highly interactive environment

Users can change the system topology or parameters while a simulation executes. This allows the user to investigate rapidly interactions between components or to explore the influence of design parameters on system performance.

3) High-level visualization

Visualization models of the system can be easily created and linked to live simulation data by the Visualization Extension Engine (VXE). Visualization helps the user to rapidly comprehend the system performance. Visual outputs include data-driven animation of the motion of solid objects, imposition on top of the solid objects of novel representations of abstract simulation data, and simply oscilloscope-like waveforms [2].

3.2 Control Hardware

For this particular application, an Infineon Microcontroller is chosen for running PI controller.

The Infineon C167/CR-LM, is a full featured 16-bit single chip CMOS microcontroller. It combines high CPU performance (up to 12.5/16/5 million instructions per second) with high peripheral functionality and means for power reduction. Several key features contribute to the high performance of the C167/CR-LM (CPU clock of 25/33 MHz).

First, the board can be adapted to a wide range of closed-loop applications due to its large number of I/O devices (111 I/O lines), a 10-Bit resolution A/D converter (4 input channel) with a sampling time of 9.7 µs, 4 high-
resolution PWM channels, 2 capture and compare units, and synchronous and asynchronous serial units.

Second, a 2-KByte local memory is used to store programs and data. This memory is fully cached and cannot be accessed by the host PC in standard operating mode. The host interface of the board is used to perform board setups, program downloads, and runtime data transfer.

4. APPLICATION EXAMPLE

We use the design of the DC motor speed control system using a PI digital controller to illustrate the PIL approach through a full design process. The design steps described in detail below are:
1) Controller Design
2) Simulation Results in VTB
3) PIL Simulation in VTB and the Infineon Microcontroller
4) Hardware Verification

1) Controller Design: A PI controller is designed in order to keep the response of the system equal to the required reference signal. We can define two main transfer functions. $G_c$, which corresponds to the PI controller, and $G_p$, which represents the DC motor.

The plant can be represented as:

$$G_p(s) = \frac{K_T}{s^2 L J + R J s + K_T^2}$$

The PI controller can be represented as:

$$G_c(s) = K_p + \frac{K_l}{s}$$

The open loop system transfer function is:

$$G_o(s) = G_c(s) \cdot G_p(s)$$

In the frequency domain, it is:

$$G_o(j\omega) = G_c(j\omega) \cdot G_p(j\omega)$$

To obtain stability, the open loop system should yield the following requirement,

$$G_o(j\omega)\bigg|_{\omega_{bw}} = 1 \cdot e^{-j(\pi - \varphi_m)}$$

which is equal to the following two equations:

$$\left|G_o(j\omega)\right|_{\omega_{bw}} = 1$$

$$\angle G_o(j\omega)\bigg|_{\omega_{bw}} = -\pi + \varphi_m$$

After calculation, the parameters of the PI controller are obtained:

$$K_p = \frac{\cos(-\pi + \varphi_m - \varphi_{bm})}{|G_p(j\omega_{bw})|}$$

$$K_l = -\frac{\omega_{bw} \sin(-\pi + \varphi_m - \varphi_{bm})}{|G_p(j\omega_{bw})|}$$

Using the motor parameters (Table 1) and defining a bandwidth ($\omega_{bw}$) equal to 500 rad/sec and a phase margin equal to 60˚ the values obtained for the PI controller are $K_p = 5.8$ and $K_l = 1696$

The algorithm for the PI controller was implemented in C language and is located in a single function.

Table 1. Parameters of the DC Motor.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal Voltage</td>
<td>$U_n$</td>
<td>24 V</td>
</tr>
<tr>
<td>Terminal Resistance</td>
<td>$R$</td>
<td>2 Ω</td>
</tr>
<tr>
<td>Output Power</td>
<td>$P_2$</td>
<td>70.8 W</td>
</tr>
<tr>
<td>No-load Speed</td>
<td>$n_0$</td>
<td>5300 rpm</td>
</tr>
<tr>
<td>Friction Torque</td>
<td>$M_f$</td>
<td>0.0043 N.m</td>
</tr>
<tr>
<td>Back-EMF constant</td>
<td>$K_E$</td>
<td>4.7145e-4 V.rad/s</td>
</tr>
<tr>
<td>Rotor Inductance</td>
<td>$L$</td>
<td>270 µH</td>
</tr>
<tr>
<td>Rotor Inertia</td>
<td>$J$</td>
<td>6.399e-6 Kg.m²</td>
</tr>
</tbody>
</table>

2) Simulation Results in VTB: First, the whole system is implemented in VTB as seen in Figure 3.

Figure 3. VTB schematic of the DC speed motor control system.

The simulation result for the system above is presented in Figure 4. A step reference for the speed is applied to the system. The step applied has amplitude equal to 300 rad/s (approximately 2865 rpm). The simulation result shows clearly that the system goes to the desired reference.
3) **PIL Simulation:** After testing the PI controller in the simulation in the previous step, the next step is implementing the PI controller in the target hardware, which in this case is the Infineon C167CR-LM.

For the PIL simulation, VTB models the DC motor and the power converter. The PI controller is running on the Infineon microcontroller and the communications between them is via serial link.

As mentioned in Section 2, two main features of the PIL procedure are time length of the control algorithm (Figure 5) and the possibility of running the controller with an independent sample rate. To show the latter concept we start the PIL simulation calling the PI controller every time step. We notice that the motor reaches the desired speed, 300 rad/s (approximately 2865 rpm), and stabilizes. However, when we call the PI controller every 0.01s, corresponding to the VTB simulation step of 1ms, the speed of the motor still follows the reference but never reaches the steady state, as shown in (Figure 6). Next, a sampling time equal to 0.1s is applied to the simulation and immediately we notice that the system becomes unstable. Finally, we go back and call the PI controller using a sampling time equal to the time step of VTB (1ms) and the system becomes controllable again (Figure 6). By means of this procedure, we can determine the controller's maximum and minimum sample rates.

5. **COMPARISON BETWEEN SIMULATION AND EXPERIMENTAL RESULTS**

After testing the PI controller on the actual target, we performed the final test, which consisted of controlling the real DC motor with the PI controller implemented in the Infineon C167CR-LM.

A laboratory prototype of the plant was designed and built (Figure 8). The motor parameters are provided in Table 1. The main switch used for the power converter is IRF530N.
Figure 8. Hardware Prototype.

Figure 9 and 10 show a comparison between the simulated and experimental results. These results show the transient that follows the step change of the reference speed from 0 rad/s to 300 rad/s (0 to 2865 rpm).

Figure 9. Motor Speed (Simulation).

Figure 10. Motor Speed (Experiment).

The rising time for the motor speed response represents the duration between the 10% and the 90% of the steady state value. We can calculate that the rising time of the simulated motor speed response shown in Figure 10 is equal to 1.35s. The experimental motor speed response, illustrated by the waveform in Figure 1011, the rising time is 1.4s. Comparing these two results, we can conclude that the simulation and the experimental results match perfectly.

CONCLUSION

This paper introduced a procedure for rapid control prototyping called Processor-in-the-Loop that allows the user to check the control software running together with a simulated model of the plant. Although the procedure is not performed in real-time, it allows the measurement of important quantities such as execution time and the sampling time of the microcontroller. This information can increase confidence in the control software before the final testing.

ACKNOWLEDGMENT

This work was supported by the US Office of Naval Research (ONR) under grant N00014-02-1-0623.

REFERENCES


