Proceedings of the World Congress on Engineering 2009 Vol I
WCE 2009, July 1 - 3, 2009, London, U.K.

On the Real Time Implementation of a Controller for an Electromechanical System

Ruben Salas-Cabrera, Jonathan C. Mayo-Maldonado, Erika Y. Rendon-Fraga,
Eduardo N. Salas-Cabrera and Aaron Gonzalez-Rodriguez *

Abstract— **This paper deals with the experimental control of the rotor position of a DC Motor. A real time Linux-based software is used for implementing this closed loop system. A setup for measuring the rotor position is implemented. This design consists of several hardware/software components. A discrete-time state observer is employed for calculating the transient and steady state values of the state variables. A load torque dynamic model is proposed in order to improve the closed loop performance. Several tests are carried out to identify the parameters of the augmented dynamic model that includes the load torque state equation. A custom-made Pulse Width Modulation-based Mosfet H-Bridge converter is implemented as a power interface.**

Keywords: ***real time, data acquisition, electric machine, electronics, microcontroller***

1 Introduction

The contribution of this work is not about the theoretical analysis and design of an observer-based control law but its implementation and experimental results. In particular, this work deals with the experimental implementation of a state space controller for a DC motor that has separate winding excitation. In other words, the field and armature windings of the electric machine are fed from independent sources, [1]. The output to be controlled is the rotor position.

A brief review of the literature follows. An interesting contribution is presented in [2]. It is related to the speed control of a series DC motor. Authors in [2] use a nonlinear dynamic model for deriving a backstepping-based control law. The emphasis is on the analysis of the controller. Author in [3] uses RTAI-Lab and Scilab for implementing a transfer function-based controller for a DC motor. The main contribution in [3] is related to testing several open source real time tools (RTAI, RTAI-Lab, Comedi, Scilab, xrtailab). Experimental results regarding high gain observers are presented in [4]. RTAI-Lab and Scicos/Scilab are used for the real time implementation. The system to be tested in [4] was a series DC motor. Valuable information about a problem associated with the analog outputs of the acquisition card was presented in [4]. Linux and RTAI-Lab are the open source real time platform that we employed in this implementation [5], [6]. The real time program that we implemented consists of a discrete-time state feedback including an integrator, a full order discrete-time state observer and a rotor position sensing algorithm. In this control law implementation the only state variable that is measured is the rotor position of the motor. Important components of the experimental system are: a personal computer, a National Instruments PCI-6024E data acquisition card [7], comedi driver library for Linux [8], a free open source real time platform [6], a custom-made power electronics converter, an incremental encoder and signal conditioning circuits for measuring the rotor position.

*Authors wish to thank Michael Griffin and Carlos A. Cruz-Villar for their valuable suggestions during the early stage of this work. This work was supported in part by the Instituto Tecnologico de Cd. Madero and the Fondo Mixto de Fomento a la Investigacion Cientifica y Tecnologica CONACyT-Gobierno del Estado de Tamaulipas. Authors are with the Instituto Tecnologico de Cd. Madero, Departamento de Ingenieria Electrica y Electronica and Division de Estudios de Posgrado e Investigacion, Av. 1o. de Mayo S/N, Cd. Madero, MEXICO. Email:salascabrera@aol.com

2 Modeling

The equations that establish the DC motor behavior are obtained by using fundamental electrical and mechanical laws [1], this is

$$V_f = r_f i_f + L_{ff}\frac{d}{dt}i_f \tag{1}$$

$$V_a = r_a i_a + L_{aa}\frac{d}{dt}i_a + L_{af} i_f \omega_r \tag{2}$$

$$J\frac{d}{dt}\omega_r + T_L = L_{af} i_f i_a \tag{3}$$

$$\frac{d}{dt}\theta_r = \omega_r \tag{4}$$

Notation for parameters and variables is given in Table 1. Parameters involved in (1)-(4) were calculated once we analyzed the experimental results of several transient and steady state tests.

2.1 Load Torque Model

There are different approaches for modeling the load torque in electrical machines. Some contributions use an algebraic description to represent the load torque and

Table 1: Notation

ω_r	Rotor speed
i_a	Armature current
i_f(0.46 Amps)	Field current
V_f	Field voltage
V_a	Armature voltage
r_f(309.52 Ohms)	Field resistance
r_a(6.615 Ohms)	Armature resistance
L_{aa}(0.0645 H)	Armature inductance
L_{ff}(14.215 H)	Field inductance
L_{af}(1.7686 H)	Mutual inductance
J(0.0038 kg/m^2)	Inertia
T_L	Load torque
K_0(0.20907),K_1(-9.8297)	Coefficients of Load torque

the associated terms (friction and damping terms) [9]. Other works utilize a detailed dynamic representation for the load torque-related terms [10]. In this work, we propose an experimental-based load torque dynamic model and identify the parameters involved in the new dynamic equation. The idea of proposing this model is to improve the performance of the closed loop system. Let us propose a state equation for the load torque, this is

$$\frac{d}{dt}T_L = K_0\omega_r + K_1 T_L \tag{5}$$

This model is a lineal dynamic approximation of a more detailed representation, [10]. For the purpose of obtaining parameters in (5), we rewrite equations (3) and (5) as

$$\frac{d}{dt}\begin{bmatrix}\omega_r \\ T_L\end{bmatrix} = \begin{bmatrix}0 & -\frac{1}{J} \\ K_0 & K_1\end{bmatrix}\begin{bmatrix}\omega_r \\ T_L\end{bmatrix} + \begin{bmatrix}\frac{L_{af}i_f}{J} \\ 0\end{bmatrix} i_a \tag{6}$$

where the armature current i_a and rotor speed w_r are interpreted as the input and output of this subsystem, respectively. To identify parameters in (5) an experimental test was carried out. During this test, the field winding was fed with a constant current at $i_f = 0.46$ amps. It is clear that under these conditions state equation in (6) becomes a linear time-invariant description. Basically, the test consists of applying a random sequence to the armature terminals. FIFO is a tool that is included in the real time platform and was used to obtain the experimental transient variables, [6], [11]. The experimental setup used for identifying parameters in state equation (6) consists of an ATmega8 microcontroller for defining the random sequence , a Nana Electronics SHR-100 hall effect sensor for measuring the armature current , a Pepperl+Fuchs incremental encoder for measuring the rotor speed, a data acquisition card for recording the transient variables, RTAI-Lab real time platform and a custom-made PWM H-bridge converter. Diagrams of the experimental setup for parametric identification can be found in [12]. Reference [12] also contains the source program of the microcontroller for defining the random sequence. Due to the lack of the space, reader is referred to [12], where the measured armature current and speed transient traces are shown. Once we obtained the transient experimental data, we utilized Matlab for the off-line processing of the data. In particular, prediction-error approach is used to calculate the numerical version of (6). By comparing (6) and its corresponding numerical version, load torque parameters K_0 and K_1 are defined. Before proceeding to the main contribution presented in Section 3, we have to describe briefly the space state representation, state observer and control law.

2.2 State Space Representation

Mostly of the physical systems presents non-linear dynamic behavior. However, under some conditions they may be considered to have a linear time-invariant representation. This is the case of a DC motor with a field winding that is fed from a constant source. Substituting the numerical parameters specified in Table 1 into (2),(3),(4) and (5) and employing 5 kHz as a sample rate, it is possible to obtain the nominal linear time-invariant discrete time dynamic model. Thus we have

$$\begin{bmatrix}\theta_r(k+1) \\ \omega_r(k+1) \\ i_a(k+1) \\ T_L(k+1)\end{bmatrix} = \begin{bmatrix}1 & 0.000200 & 0.000004 & -0.000005 \\ 0 & 0.000045 & 0.042383 & -0.052578 \\ 0 & -0.002497 & 0.979644 & 0.000065 \\ 0 & 0.000041 & 0.9x10^{-6} & 0.998035\end{bmatrix} \begin{bmatrix}\theta_r(k) \\ \omega_r(k) \\ i_a(k) \\ T_L(k)\end{bmatrix} + \begin{bmatrix}4.4x10^{-9} \\ 0.0000659 \\ 0.0030691 \\ 9.201x10^{-10}\end{bmatrix} V_a(k) \tag{7}$$

$$\theta_r(k) = \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\theta_r(k) & \omega_r(k) & i_a(k) & T_L(k)\end{bmatrix}^T \tag{8}$$

where (7) is the discrete time state equation and (8) is the discrete time output equation. It is clear the following definition for the state $x = \begin{bmatrix}\theta_r & \omega_r & i_a & T_L\end{bmatrix}^T$, the output to be controlled $y = \theta_r$ and the input of the system $u = V_a$. State equation (7) assumes that parameters are linear time-invariant. Similarly, the field winding is supposed to be fed by a voltage source that keeps constant the field current at 0.46 amps. This is the reason for not including the state equation of the field winding. Experimental results in Section 4 will show that the nominal model in (7)-(8) contains the fundamental dynamic characteristics of the system necessary to design a successful experimental control law. Parametric uncertainty will be addressed by using an integrator, i.e.

$$x_I(k+1) = x_I(k) + e(k) = x_I(k) + y(k) - r(k) \tag{9}$$

where x_I is the integrator state variable, r is the setpoint and $y = \theta_r$ is the output to be controlled.

2.3 State Observer

The state space equation of a standard linear time-invariant discrete time observer can be written as [13]

$$\tilde{x}(k+1) = G\tilde{x}(k) + Hu(k) + K_e[y(k) - C\tilde{x}(k)] \quad (10)$$

The observer gain K_e is a 4x1 constant vector associated with the output error $y(k) - C\tilde{x}(k) = \theta_r(k) - \tilde{\theta}_r(k)$. As it is explained in the next subsection, desired poles are basically computed by specifying the desired time constant. Once the set of desired poles are obtained, a standard procedure for calculating the gain K_e is applied. This gain can also be computed by using the Scilab/Scicos *ppol* command. In this particular case, gains in (10) were chosen such that the desired observer poles become $z_{1,2,3,4} = 0.994017964, 0.994017963, 0.994017962, 0.994017961$, this is

$$K_e = \begin{bmatrix} 0.0015523 \\ 0.1544085 \\ -0.0392419 \\ -0.0014389 \end{bmatrix} \quad (11)$$

It is important to emphasize that for the purpose of implementing the real time closed loop system the rotor position is the only state variable to be measured. The state feedback uses the state variables (including the load torque) provided by the experimental discrete-time state observer. Armature current and rotor speed were measured just for the off-line parametric identification of the load torque model.

2.4 Controller

Pole placement technique is used to calculate the gains associated with the state feedback. Armature voltage $u(k) = V_a(k)$ is now defined by the following standard state feedback

$$u(k) = -\begin{bmatrix} K_I & K \end{bmatrix} \begin{bmatrix} x_I(k) \\ x(k) \end{bmatrix} \quad (12)$$

where K_I is the gain corresponding to the integral state variable, K is the gain vector associated with the state variable of the original system x. These gains are calculated following a standard procedure [13]. Gains can also be computed by using the Matlab command *place*. On the other hand, it is well known that the relationship between a s-plane pole having a particular time constant and the corresponding z-plane pole is given by $z = e^{-\frac{1}{\tau}T}$, where τ is the time constant and T is the sampling period. If $\tau = 0.1$ seconds and $T = 0.0002$ seconds, thus the corresponding pole in the z-plane becomes $z_1 = e^{-\frac{1}{0.1}(0.0002)} = 0.998001998$. Similarly, the rest of desired controller poles can be obtained

$$z_{1,2,3,4,5} = 0.998001998, 0.998001997, 0.998001996, 0.998001995, 0.998001994$$

thus the corresponding controller gains become

$$[K_I, K] = [0.0006168, 1.2288494, -0.6467532, -4.021708, -2.4009488] \quad (13)$$

Next section presents the main contribution of this work, i.e. the design of several hardware/software components employed in the closed loop system.

3 Experimental Setup

3.1 Real-time platform

As it was established earlier, a free open source real-time platform is employed to solve the control algorithm. The resulting executable program is able to provide correctness of the calculated values and to accomplish strict time requirements [14]. In our case, the control algorithm was computed every 0.0002 seconds (5 kHz). Furthermore, this software is in charge of reading/writing digital and analog signals at the terminals of the acquisition card. Knoppix 5.0 and RTAI-Lab are used as the real-time platform. Knoppix 5.0 is a Linux distribution which includes RTAI-Lab tool-chain [6]. The Real Time Application Interface (RTAI) is a patch of the Linux kernel that is able to carry out real-time tasks. The RTAI-Lab includes *Comedi*, which is a collection of Linux drivers for a variety of common data acquisition cards [8]. RTAI-Lib and Comedi, allows the real-time program to access the signals at the terminals of the data acquisition card. In this particular work, a National Instruments PCI-6024E data acquisition card is used. Scilab/Scicos is a graphical environment that includes a library of blocks that can be employed to simulate the closed loop system, [11]. The library of real-time blocks is defined in RTAI-Lib. We use Scilab/Scicos to simulate the closed-loop system as a first step of the implementation. Then, the Scilab/Scicos simulation blocks are switched to the corresponding RTAI-Lib real-time blocks. The executable program can be created by compiling the RTAI-Lib/Scilab/Scicos-based source program by employing the RTAIcodegen tool located in Scicos. The main source program is shown in Fig. 1. In order to be able to present it in just one figure we used several subroutines that are called super blocks. They are a tool for organizing the program structure. A super block looks like any other block, however its operation is defined by a custom-made Scicos code. The main program can be divided in three parts. The first part is the super block that includes the position measurement algorithm. A general description of that algorithm is presented in the next subsection. The second part is the super block that computes the state observer. The last part calculates the state feedback and solves the integral state space equation. Closed loop gains K_I, K_1, K_2, K_3 and K_4 are defined in (13). In addition, the program includes the set-point block, the Comedi block (DAC) and a gain denoted by A. This gain scales the numerical value of the calculated armature voltage. The scaled voltage is the

numerical representation of the actual voltage applied to the power converter. It is denoted by *control voltage*, (see Fig. 5). Additional information regarding the program code can be found in [12].

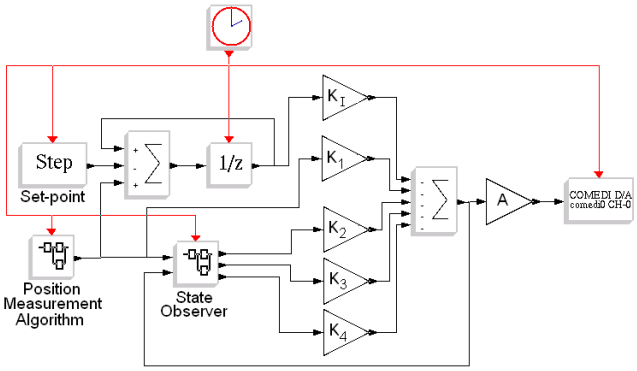

Figure 1: Main Scicos program including several super blocks (subroutines)

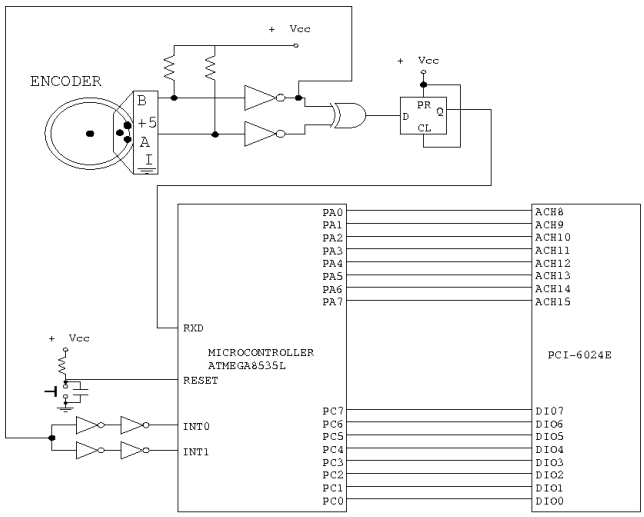

Figure 2: Rotor position measurement setup

3.2 Rotor position measurement design

A design for measuring the rotor position is implemented. There are several components of this hardware/software design. One of them is the Pepperl+Fuchs encoder that is physical attached to the motor shaft. The second one is a ATmega8535 microcontroller-based signal conditioning circuit and the third one is the real time software that calculates the rotor position from the digital signals provided by the microcontroller. The ATmega8535 microcontroller is programmed for being used as a 16-bit binary UP/DOWN counter. A signal conditioning circuit is necessary to determine the shaft direction, which is implemented by using a flip-flop integrated circuit. The flip-flop uses the pulses of channels A and B of the encoder and provides a binary 1 or 0 depending on the shaft direction. Channel A (or B) of the encoder is then connected to a micro-controller terminal without modifying the voltage as they both the encoder and the microcontroller have 5 volts as a nominal voltage. The microcontroller is in charge of counting the pulses provided by the encoder. In this case, the encoder generates 1024 pulses per revolution. The micro-controller count goes up or down depending on the flip-flop binary signal. The complete electronics diagram of the rotor position measurement setup is shown in Fig. 2. The micro-controller provides a 16-bit binary count that represents the rotor position. On the other hand, the National Instruments PCI-6024E data acquisition card has only 8 digital inputs [7]. In order to be able to read the 16-bit binary data, 8 additional digital inputs are obtained by employing 8 analog inputs of the acquisition card and using them as digital inputs. The algorithm to interpret those analog inputs as digital inputs is coded in the Scilab/Scicos source program. The routine that obtains the binary data by using digital inputs is shown in Fig. 3. The algorithm consists of multiplying each bit (1 or 0) by its corresponding value in the decimal system, then the results are added. The same algorithm is employed to convert bits that were obtained by using the analog inputs. In addition, a signal conditioning routine is employed to use those analog inputs as digital inputs, see Fig. 4. A comparator determines the binary value (1 or 0) of the signal that was read at the terminals of the analog input. This routine is implemented for each one of the 8 signals obtained by using the analog inputs. Combining the results provided by these routines a decimal measurement of the rotor position is obtained. A final super block called *Position*

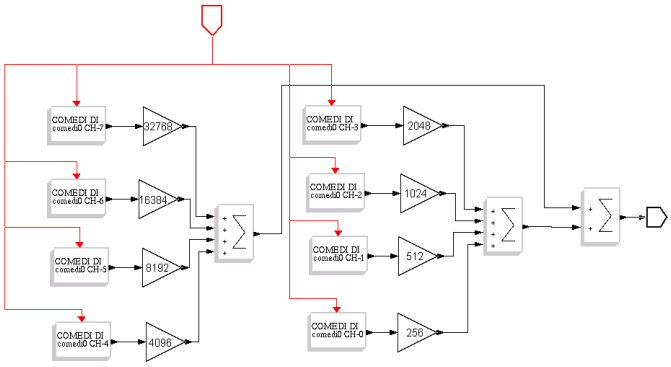

Figure 3: Subroutine associated with the digital inputs of the position measurement algorithm

Measurement Algorithm (see Fig. 1) was created to include the described code. It is important to mention that the rotor position setup is able to measure positive and negative values. In order to accomplish this feature, the microcontroller was programmed to have an initial count that is located at the middle of the 16-bit count range. Due to the lack of space, reader is referred to [12], where

information regarding the ATmega8535 micro-controller assembler code can be found.

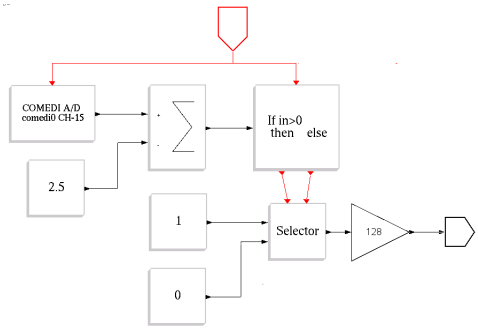

Figure 4: Software-based signal conditioning for one of the eight analog inputs used as digital inputs

3.3 Power converter

A Pulse Width Modulation-based MOSFET H-Bridge converter was designed and implemented. This type of power electronics device is commonly used to drive DC motors when bidirectional speed/position control is needed. The numerical value of the armature voltage is defined by the state feedback and calculated by the computer. This value is written by the real time software to one of the analog output channels of the data acquisition card. The actual armature voltage is applied by the H-Bridge to the DC motor and is defined by the following equation $V_a = \delta V_{cc}$, where V_{cc} is the DC power supply voltage of the converter and δ is the duty cycle of the power MOSFETs. The duty cycle is clearly associated to the signal at the analog output of the data acquisition card. This signal is denoted by *control voltage* in diagram of the power electronics converter Fig. 5. This converter consists of four parts. Part (a) is implemented for conditioning the signal provided by the data acquisition card. Part (b) generates the PWM signals depending on the voltage in Part (a). This part also implements a switching delay to provide a recovery time of the power components during every switching period. Part (c) isolates power from instrumentation Parts (a) and (b). Finally, Part (d) is the actual MOSFET H-Bridge circuit which is directly connected to the armature terminals of the DC motor.

4 Experimental Results

This section presents the experimental transient characteristics of the closed loop system. The real time program calculates the discrete-time state feedback in (12), which includes the discrete-time integrator state variable. Gains of the controller are defined in (13). In our case, all of the parts of the closed loop control algorithm was computed every 0.0002 seconds (5 kHz). Here, we implement the armature voltage as a function of estimated states excepting

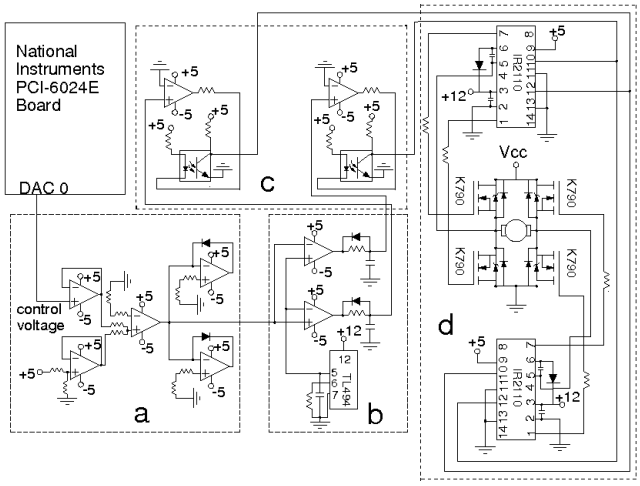

Figure 5: Power Converter. a) Signal Conditioning. b) PWM and switching delay. c) Isolation. d) H Bridge and DC motor.

the rotor position, which is the only state variable that is measured. Estimated states, including the load torque, are obtained by solving the full order discrete-time state observer in (10). By comparing (7) and (10) it is easy to define matrix G and vector H. Gains associated to the observer are specified in (11). Transient behavior of the integrator state variable was computed by solving the state equation in (9). Initial conditions of the integrator and observer state variables were set numerically equal to zero. The experimental trace shown in Fig. 6 illustrates the dynamic characteristic of the rotor position following a 25.1328 *radian* (4 *revolution*) reference command. The

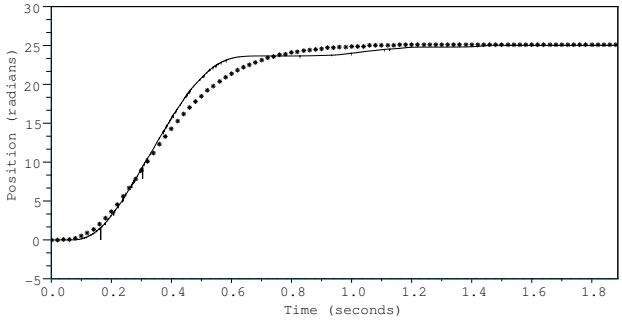

Figure 6: Rotor Position: – Experimental measured, ** Simulated model-based

simulated model-based trace of the rotor position is also shown in Fig. 6. It is clear that these signals (simulated and measured) are similar to each other. Initially, the rotor position was at zero radians. The position begins to increase immediately until the position error is close to zero, which occurs approximately at 1.2 sec. In or-

ISBN: 978-988-17012-5-1

der to save these transient data, a real-time block called FIFO was used, [6] [11] . Experimental observer-based variables were also saved by using FIFO. One of those observed states is shown in Fig. 7. For the purpose of

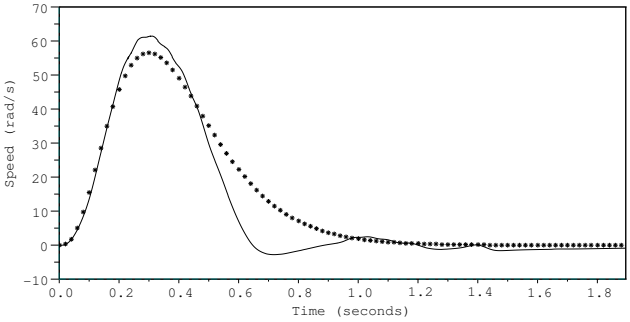

Figure 7: Rotor speed: – Experimental observer-based, ** Simulated model-based

testing the experimental closed loop system in a more demanding operating condition, we installed an inertial disk. The original value of the inertia was 0.0038 kg/m^2, see Table 1. The new inertia is 0.0398 kg/m^2. In other words, the new parameter (inertia) is about 10.5 times the original value. It means a significant increase of a particular parameter, of the mechanical subsystem, that is clearly connected to the output to be controlled (rotor position). In addition, the inertial disk was intentionally slightly loose to the shaft, therefore the parameter varied during the test around the non-original value. Even under that demanding condition the experimental closed loop system was able to follow a 21.99 *radian* (3.5 *revolution*) reference command, see Fig. 8.

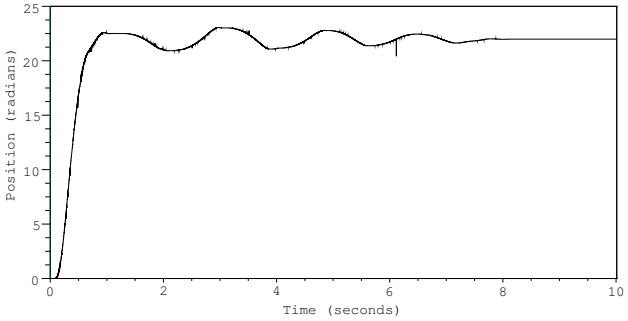

Figure 8: Experimental measured rotor position when a significant change of a parameter ocurrs (10.5 times the original value aprox.)

5 Conclusions and Future Work

An experimental setup was designed and used in this work. A fundamental part of the setup is the free open source real time software that is able to accomplish the control algorithm under strict time requirements. This setup also provides an effective framework for rapid testing of more complex control algorithms. A future work might be related to the idea of implementing a new nonlinear control scheme, keeping the hardware platform and changing just the source program running under the same real time software platform.

References

[1] P. C. Krause, O. Wasynczuk and S. D. Sudhoff, *Analysis of Electrical Machinery and Drive Systems*. John Wiley and Sons, IEEE Press Power Engineering; 2004.

[2] D. Zhao, N. Zhang. *An improved nonlinear speed controller for series DC motors. IFAC 17th World Congress*, 2008.

[3] J. Jugo. Real-time control of a DC motor using Scilab and RTAI. *Scilab INRIA Rocquencourt*, 2004.

[4] N. Boizot, E. Busvelle, and J. Sachau. High-gain observers and Kalman filtering in hard real-time. *RTL 9th Workshop*, 2007.

[5] Open Source Linux Distribution. *Knoppix 5.0*, 2004.

[6] R. Bucher, S. Mannori and T. Netter. *RTAI-Lab tutorial: Scilab, Comedi and real-time control*. 2008.

[7] National Instruments. *PCI-6024E National Instruments Manual*. 2006.

[8] D. Schleef, F. Hess, and H. Bruyninckx. *The Control and Measurement Device Interface handbook*. 2003.

[9] M. Ruderman, J. Krettek, F. Hoffmann, T. Bertram. *Optimal state space control of DC motor. IFAC 17th World Congress*, 2008.

[10] T. Tjahjowidodo, F. Al-Bender, H. Van Brussel. *Friction identification and compensation in a DC motor. IFAC 16th World Congress*, 2005.

[11] S. L. Campbell, J. P. Chancellier, and R. Nikoukhah. *Modeling and Simulation in Scilab/Scicos*. Springer, 2006.

[12] E. Y. Rendon-Fraga, *Rotor Position Control of a DC Motor using a real time platform*, MSc. thesis, in spanish. Instituto Tecnologico de Cd. Madero. Cd. Madero, Mexico; 2009.

[13] G. F. Franklin, J. D. Powell and M. Workman, *Digital Control of Dynamic Systems*, Addison Wesley, Menlo Park, CA; 1998.

[14] P. A. Laplante. *Real-Time systems design and analysis an engineer's handbook*. IEEE Computer Society Press, 1997.