
AC 2011-931: ENHANCING MECHATRONICS EDUCATION USING MODEL-BASED TECHNIQUES AND MATHWORKS TOOLS

Farzad Pourboghrat, Southern Illinois University

Farzad Pourboghrat received his Ph.D. degree in Electrical Engineering from the University of Iowa in 1984. He has since been with the department of Electrical and Computer Engineering (ECE) at Southern Illinois University, Carbondale (SIUC) where he is currently a Professor and director of the Embedded Control Systems (ECS) Lab. He is a senior member of IEEE. His research interests include control theory, real-time embedded control, mechatronics and distributed robotic systems.

Narayanan Ramachandran, Southern Illinois University Carbondale

Narayanan Ramachandran received his Bachelors degree in Electrical and Electronics Engineering from Bangalore University in 2004 and was awarded a Masters of Science degree in Electrical and Computer Engineering from Southern Illinois University Carbondale in 2010. He is currently pursuing his Ph.D. degree in Electrical and Computer Engineering at Southern Illinois University Carbondale and is a Research Assistant at Embedded Control Systems Laboratory. His main areas of research includes power electronics and control systems.

Aishwarya Vasu, Southern Illinois University Carbondale

Aishwarya Vasu received her Bachelor's degree in Electronics and Communication Engineering from Anna University, Chennai, India in 2006. She received her Master of Science degree in Electrical and Computer Engineering from Southern Illinois University Carbondale. She is currently a PhD student at Southern Illinois University Carbondale. She is a Research assistant at Embedded Controls and Systems Laboratory at SIUC. She is also a Research assistant at Materials Technology Center, SIUC.

Her research interests include Real time programming of control systems and Real time operating/Embedded Systems.

Arjun Shekar Sadahalli, Southern Illinois University Carbondale

Arjun Shekar Sadahalli is a graduate student currently pursuing his Ph.D. in Electrical and Computer Engineering at Southern Illinois University Carbondale. He received his Bachelors degree in Electronics and Communication from Visveswaraiah Technological University in 2004 and his Masters of Science degree in Electrical and Computer Engineering from Southern Illinois University Carbondale in 2009. He works as a Research Assistant in Embedded Control Systems Laboratory with control systems and navigation as his primary area of research.

Prashanthi Banala, Southern Illinois University Carbondale

She received Bachelor's degree in Electrical and Electronics Engineering from Jawaharlal Nehru Technological University in 2008. She is currently pursuing Master's in Electrical and Computer Engineering at Southern Illinois University Carbondale and working as Research Assistant in Embedded Control System Laboratory at Southern Illinois University Carbondale. Her main research interests are control systems and distributed system design.

Giampiero Campa, MathWorks

Giampiero Campa, was born in Taranto, Italy. He received both is M.S. degree in Control Engineering (1996) and his Ph.D. degree in Robotics and Automation (2000), from the University of Pisa, Italy. He has also worked at the Industrial Control Centre, Strathclyde University, UK, (1995) and at the Department of Aerospace Engineering, Georgia Institute of Technology, Atlanta, USA (1999). From 2000 to 2008 he has served as faculty in the Flight Control Group at the Department of Aerospace Engineering, West Virginia University. His research at WVU involved system identification, sensor fusion, fault tolerance, machine vision, and adaptive and nonlinear control, especially applied to UAVs. During his stay at WVU Dr. Campa has published more than 30 peer-reviewed articles in international journals, about 60 research papers for international conferences, and a couple of book chapters. He has joined the MathWorks in 2009, where he currently works as a Technical Evangelist for the US west coast area.

Enhancing Mechatronics Education using Model-Based Techniques and Mathworks Tools

Abstract- This paper describes our efforts in renovating an existing non-required mechatronics course to upgrade the contents of the course and to improve the students' learning experience. The main efforts include developing lecture materials, demo tools, and laboratory project modules employing model-based design approach, using Matlab, Simulink, and a number of other Mathworks toolboxes. These tools simplify the process of system modeling, control design, and embedded microcontroller programming for rapid prototyping and design verification.

1. Introduction

It has long been known that teaching through examples and hands-on laboratory exercises improve the students' learning experience, especially in technical multi-domain subjects such as Mechatronics. However, the development and implementation of effective examples and laboratory experiments are both time-consuming and expensive. Model-based techniques using Matlab, Simulink, and other Mathworks toolboxes¹ provide a systematic and flexible approach to teaching Mechatronic systems that can be achieved within constrained university budgets.

This paper highlights some of the developments toward renovating an existing non-required Mechatronics course, which could be taken by both undergraduate and graduate students, to improve its learning outcome. A teaching innovation grant was used to achieve the goals of the project using model-based design approach and Mathworks tools. Toward this end, both lecture and laboratory components of the course were modified in tandem. Lecture materials were revised to include a number of tutorials and webinars to introduce the use of various Mathworks toolboxes¹ for modeling and automated control design. Also, example models were developed using Simscape and Simulink for demos within lectures, including dc motor, pendulum, inverted Furuta pendulum, segway, flexible cantilever beam, and differential-drive mobile robot. Similarly, several electro-mechanical systems were developed to be used for lab experiments and rapid-prototyping using Microchip's dsPIC microcontrollers and Arduino development board. Such lab set-ups include a dc motor speed control system, an inverted pendulum balancer, a laboratory-scale segway, a differential-drive robotic system, a flexible beam vibration controller, and an acoustic particle-levitation system.

As a first trial, a number of these modules were developed and used in an exploratory model-based control design course during summer 2010 with seven graduate students enrolled in the course. The tasks of modeling, control design, and implementation on an embedded dsPIC microcontroller using Mathworks tools were considered for the speed control of a dc motor, as a typical example. It was observed that using model-based approach with Mathworks tools all students became more engaged in the course and became more confident with the processes involved in the design and implementation of embedded control systems.

These developed tools are now being used in a renovated Mechatronics course. Additional demo tools and modules are being developed and refined for continued enhancement of the course using model-based approach with Mathworks tools. A GUI is also being developed to facilitate the automatic system modeling, control design, and programming of a number of embedded microcontrollers for control implementation, including dsPIC30, dsPIC33, PIC24, and Arduino board, among others. Improvements in the technical knowledge of the students and their confidence in carrying out the laboratory experiments throughout the course are being used as the primary indicator to assess the effectiveness of these new teaching tools.

2. Model-based design approach

Model-based design is a methodology used for designing embedded software. It is used to address the challenges associated with modeling, analysis, design, implementation, testing and optimization of multi-domain motion control systems^{2,3,4}. The approach is generally implemented using an integrated software environment that is interfaced with the external world through data-acquisition systems. In this project, Matlab and Simulink are used for modeling and simulation of multi-domain systems, including electrical, mechanical, and many others, as well as for signal processing, parameter estimation, control design, optimization, and real-time embedded programming. A typical embedded control application involves: modeling, parameter tuning, system identification, control design, optimization, embedded programming, monitoring and verification^{3,4,5}. These are briefly explained in the following.

Modeling: two common approaches to modeling dynamic systems are the first principle and the data-driven techniques³. In the first principle technique, physics laws are used to derive the mathematical equations that describe the dynamics of the system. In the data-driven technique, the input and output (I/O) signals of the actual system are measured and an approximate mathematical model is found to fit the collected I/O data. Simulink can be used for modeling dynamic systems when their mathematical models are already known. Simscape components, on the other hand, can be used to model dynamic systems using block models of simpler physical components/systems³. In this work Simscape components are used for first principle modeling of multi-domain mechatronic systems to capture the essential dynamics of the system and are used for the purpose of simulation, analysis, and control design.

Parameter tuning: the Simscape model of the system can accurately represent the essential characteristics of the actual system in simulation, but in general it cannot correctly predict the response of the system to arbitrary inputs with high numerical precision. The Simulink design optimization tools can be used to automatically tune the parameters of the Simscape model of the system so that the simulated I/O data of the model matches the collected I/O data from the actual physical system. The tuned model of the system can then predict the response of the actual physical system to arbitrary inputs quite accurately if all actual system components are included in the Simscape model and their corresponding parameters are tuned.

System identification: most control design techniques require an accurate model of the system⁶. Typical linear control techniques, however, are most suitable for linear systems. Thus, in general, linearized models of the systems must be obtained for the linear control design. For this purpose, the system identification toolbox can be used for data-driven modeling to estimate a mathematical model of the system that fits the collected I/O data. The approximate model can be chosen linear or may have nonlinear components, which can subsequently be used for the purpose of control design.

Control design: control design techniques for linear systems, including root-locus, frequency-response, and state-space techniques are well established and can be used for systems with known linear models^{6,7}. Various control design techniques also exist for linear systems with uncertain models⁷. Once an approximate linearized model of the system has been found that closely matches the actual measured I/O data, the Simulink control design toolbox can be used for the design of a number of linear control strategies, including PID and lead-lag controllers. The SISO and multi-loop design tools can be used to interactively develop and adjust the control systems while making it possible to simultaneously observe and verify the system performance.

Optimization: the control design and tuning can be done automatically for both the PID and the control transfer function blocks. Simulink design optimization tools can be used to automatically tune any of the parameters or gains in the loop, including the controller gains. Thus, the adjustment of control parameters can be automated once the design specifications are defined in terms of the system-response characteristics in either time or the frequency-domain.

Embedded microcontroller programming: once the control system is designed and the system performance in simulation is satisfactory, the control algorithm must be uploaded in the actual microcontroller that is embedded in the physical system for real-time implementation. Real-time workshop and embedded target toolbox for the specific microcontrollers (e.g., dsPIC or Arduino) can be used for this process^{8,9,10}. The microcontroller must be interfaced with the physical world through analog input and analog/PWM output blocks in the control subsystem. Real-time workshop can then be used to build and automatically upload the control program in the embedded microcontroller for implementation which would run indefinitely until it is reset.

Monitoring and verification: once the control system is programmed, built, and uploaded in the embedded microcontroller, the actual system response could be measured and transmitted, via serial communication, to the host-PC for monitoring and verification of the designed embedded control performance. Both Microchip and Arduino Target Blocksets^{8,9,10} provide such serial communication blocks. When serial communication blocks are not available, the freeware, Real-Term¹¹, can be used for serial communication to host PC via RS232 USB data transmission. The transmitted data can then be stored and used offline in Matlab/Simulink for system modeling and parameter tuning and, with appropriate baud-rate, can even be used for real-time monitoring and verification of the closed-loop performance.

3. Course materials

Modifications in the course include new lecture materials, demo materials, and lab exercises. A number of recorded webinars¹² from Mathworks have been incorporated into the class lectures to introduce the various processes involved in the mechatronics system design using Matlab and Simulink tools. These include introductions to Simscape for multi-domain system modeling, parameter tuning using Simulink control design tools, linearized modeling of systems using parameter identification toolbox, control design and auto-tuning of the control parameters using both Simulink control design and Simulink design optimization tools, as well as real-time workshop and embedded target blocksets for Microchip dsPIC and Arduino.

A number of Simscape models have been developed for use as demo materials in class lectures. These include dc-motor, pendulum, inverted Furuta pendulum, segway, and differential-drive mobile robot. These models are provided to the students for exploration and for use in various homework exercises involving modeling, parameter estimation, control design, and optimization using Matlab and Simulink.

New laboratory exercises using Matlab, Simulink and various Mathworks toolboxes are among the major changes in this renovated Mechatronics course. A basic platform is developed for the speed control of an electric dc motor equipped with a tachometer for motor speed measurement, interfaced both with dsPIC and Arduino. This platform is used for teaching the essential tasks involved in the design and control of mechatronic systems using Mathworks development tools. A number of other electro-mechanical systems are also developed for the course and are used for lab exercises involving modeling and control of mechatronic systems using Mathworks tools.

The course also includes team projects. Each student is given an Arduino kit for the entire semester, consisting of the Arduino board, a small breadboard, and a few electronic components. Teams of two to three students are to build their own simple machines, such as a Segway and a differential-drive mobile robot, using motors, sensors and simple connectors, as in an Erector set (similar to Legos). These team projects must be developed and completed using Mathworks tools, and are expected to be demonstrated by the student teams. A possible final robotic competition event among the class teams could also be held at the end of the semester for bonus points.

4. DC motor speed control platform

Here, the various tasks to be performed by all students for a dc motor speed-control project using Mathworks tools are highlighted in more detail, for both dsPIC and Arduino platforms.

dsPIC platform: A schematic picture of the test platform using Microchip's dsPIC30F4012 microcontroller is shown in Figure 1, which highlights the various subsystems in the set-up as well as the interface with the host PC. This platform is to be used with dsPIC blocksets^{8,9}.

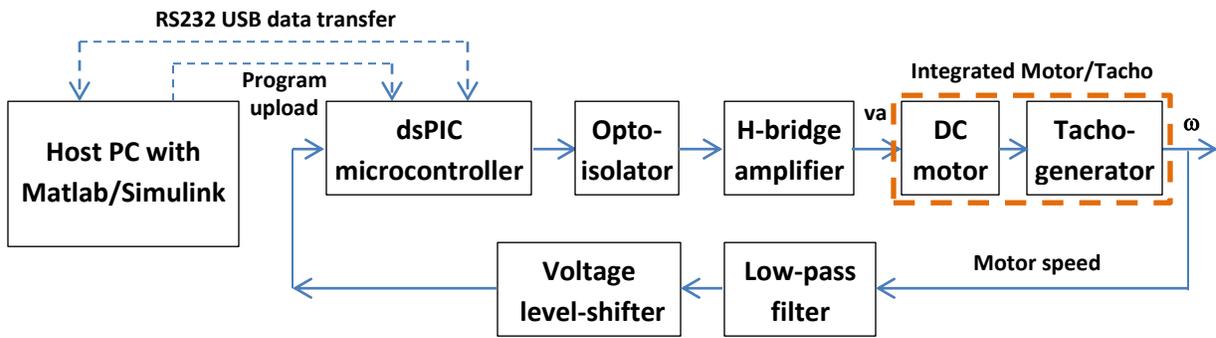


Figure 1 – Schematic of the DC motor speed control lab set-up

The photographic picture of the test platform for this lab experiment is shown in Figure 2.

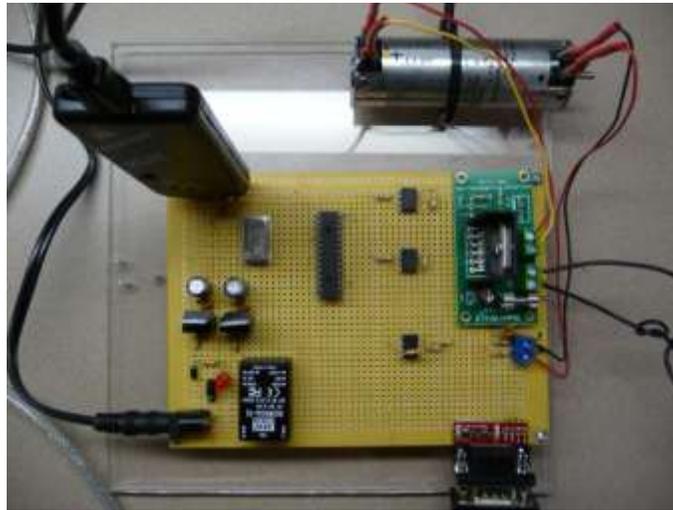


Figure 2 – Lab platform for DC motor speed control

The specs of the PMDC motor are: +/- 4000 RPM, 12V, 7mH, 10Ω, 3.45 oz-in/amp, and it has an integrated tacho-generator (speed sensor) rated at 1.9V/RPM. For the model-based control design a Simscape model of the dc motor is developed in the host PC, as shown in Figure 3.

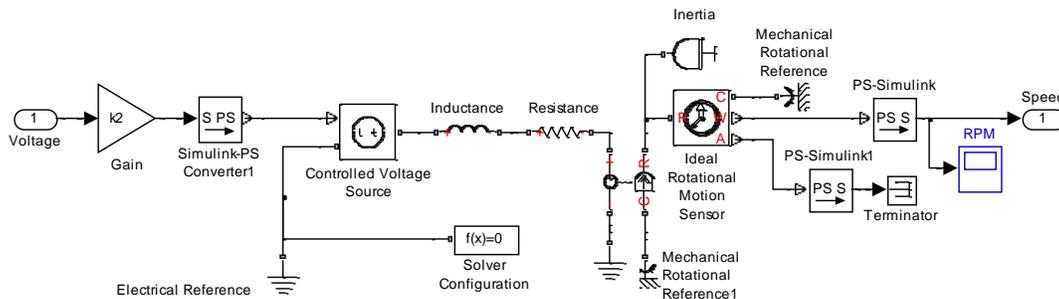


Figure 3 – Simscape model of the dc motor

serial communication with the PC because the free version of Lubin Kerhuel’s dsPIC Target Blockset only supports 6 I/O pins, which is not enough to simultaneously support multiple analog I/O for control and the serial communication with the PC. The Microchip’s Matlab Device Blocks for MPLAB IDE⁹, on the other hand, has no limitation on the number of I/O pins and hence can simultaneously support multiple analog I/O for control and serial communication with the PC.

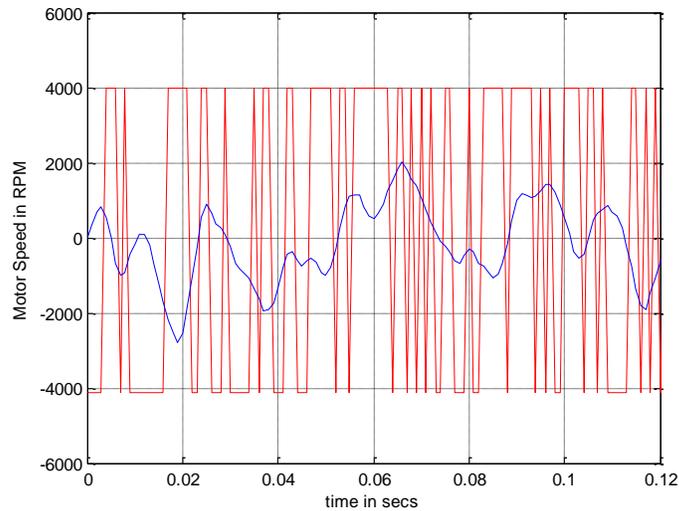


Figure 5 – I/O data collected from actual DC motor platform

The Simulink design optimization tool is then used to automatically tune the motor parameters and the scaling gains in the Simscape model of the system so that the simulation data matches the actual collected I/O data from the real system. The parameter tuning history and the system response trajectory validation after tuning of the parameters are shown in Figure 6.

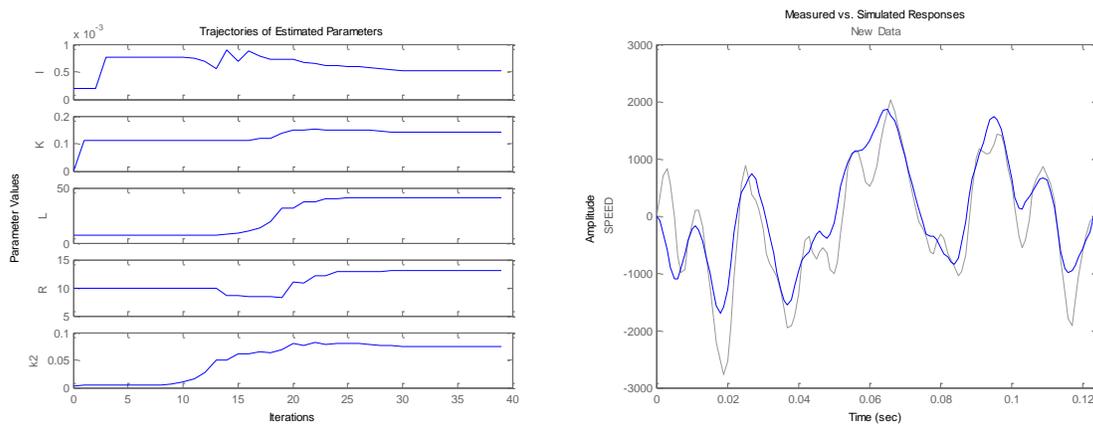


Figure 6 – Parameter tuning history and model response validation

The linearized model of the system can also be found quite accurately from the captured I/O data, using the System Identification toolbox¹, without being restricted to tuning only those model parameters that are associated with the physical parameters. These linearized black-box models can be used to verify the accuracy of the physically motivated, but idealized, Simscape models with tuned parameters. The parameter identification GUI for I/O data set-up and the model structure selection windows are shown in Figure 7.

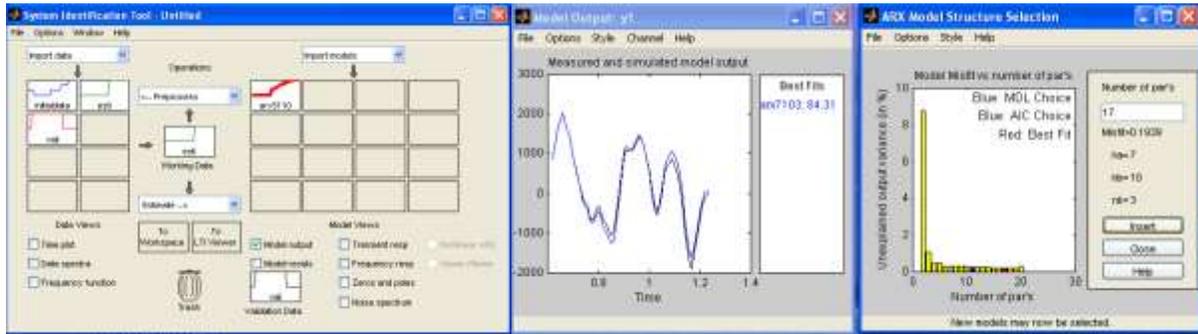


Figure 7 – System identification data set-up GUI and model structure selection windows

A discrete transfer-function model of order 12, with a sampling-time of $T_s=0.01s$, is found for the system. An equivalent continuous transfer function of the system is then determined and an approximate transfer function of order 3 is found, using balanced model-order reduction, as

$$G(s) = \frac{0.1094 s^3 - 14.87 s^2 + 689.6 s + 2565}{s^3 + 52.83 s^2 + 1528 s + 2878}$$

The linearization of the tuned Simscape model also reveals a transfer function of order 3, as

$$G(s) = \frac{0.001848 s + 4.462e7}{s^3 + 1314 s^2 + 398000 s + 8.395e7}$$

The corresponding Bode plots of the estimated transfer-functions of the DC motor platform are shown in Figure 8. The figure shows the Bode plots of the estimated 12th-order and the 3rd-order transfer functions found using the System Identification toolbox and the 3rd-order linearized transfer function of the tuned Simscape model found using the Simulink Design Optimization toolbox¹. The figure indicates that the estimated transfer function using parameter-tuning technique is not exactly equivalent to the other two estimated transfer functions using the System Identification toolbox. Nevertheless, all these approximate transfer function models are accurate enough, at least in the small frequency range of up to 20 rad/s, and can be used successfully for the speed control of the actual dc motor.

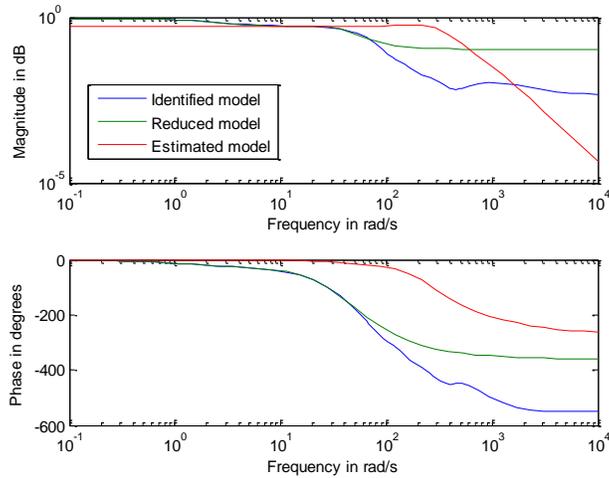


Figure 8 – Bode plot of the transfer functions of the DC motor platform

A discrete PID controller is considered for speed control of the DC motor platform, as shown in Figure 9, where the reference signal V_{ref} is chosen as a square-wave swinging between ± 10 volts with a period of 10 seconds.

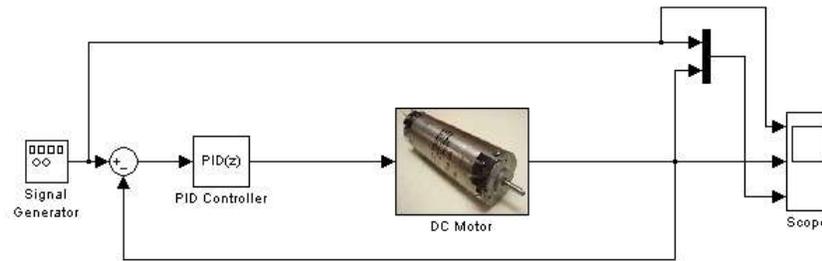


Figure 9 – Block diagram of the PID control design

The auto-tuning result for the PID control parameters and the simulated closed-loop motor speed, using the tuned PID controller with a sample time of $T_s=0.01s$, are shown in Figure 10.

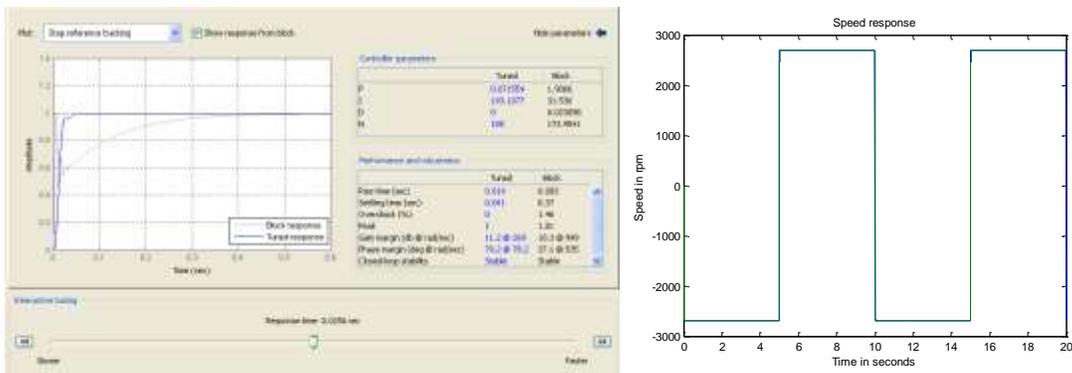


Figure 10 – Auto-tuning of PID control parameters and simulated closed-loop response

The tuned discrete PID controller block is then added to the Simulink model of the dsPIC program for actual implementation via dsPIC, as shown in Figure 11.

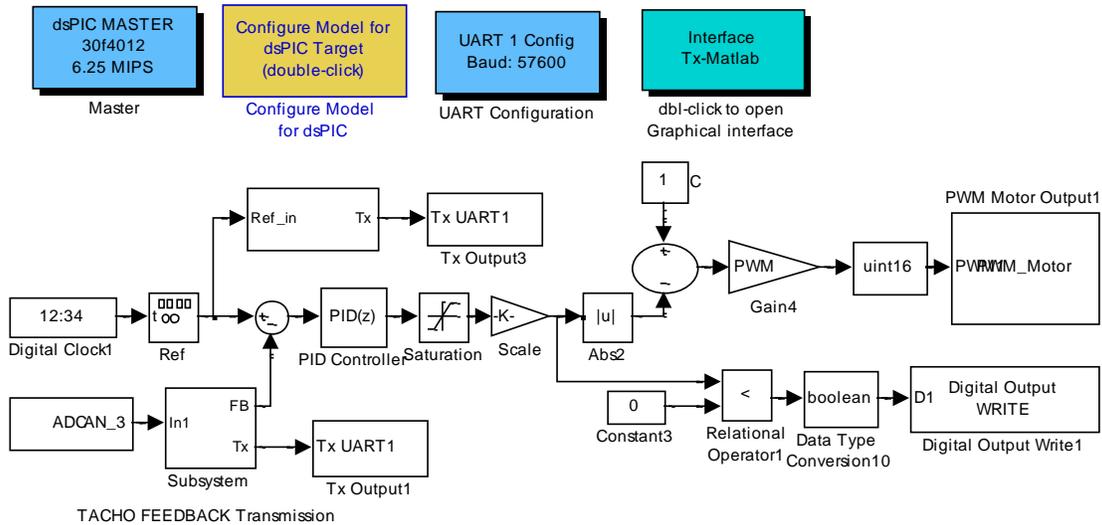


Figure 11 – Simulink model of the dsPIC program for PID control implementation

The dsPIC program for implementing the tuned PID controller is then built and uploaded in the actual embedded dsPIC microcontroller using the dsPIC Target toolbox and real-time workshop (RTW), through Microchip’s PICKit2 programming tool. The uploaded program would then run indefinitely until the microcontroller is reset. The actual reference signal and the motor speed in the closed-loop system are then captured and transmitted to the host PC, in real-time, using the freeware Real-Term, for the purpose of plotting and verification. Figure 12 shows the actual measurement of the motor speed against the reference command in the closed-loop system with the embedded PID controller. Although the actual motor speed is slightly different than the simulated response, the performance of the embedded controller seems to be satisfactory.

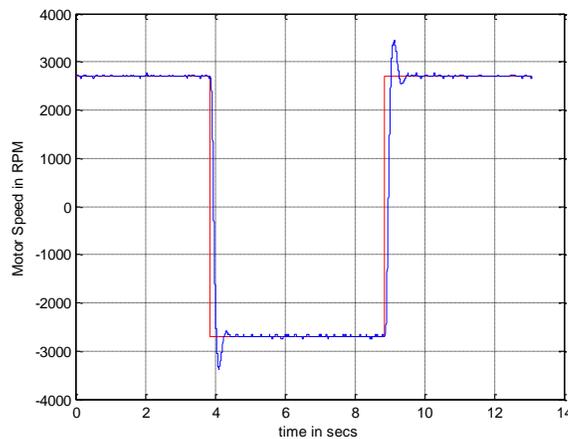


Figure 12 – Actual motor speed in closed-loop using the tuned PID controller

The steps and the results of system modeling using System Identification toolbox, for the Arduino set-up with a sampling time of $T=0.01s$, are shown in Figures 15.

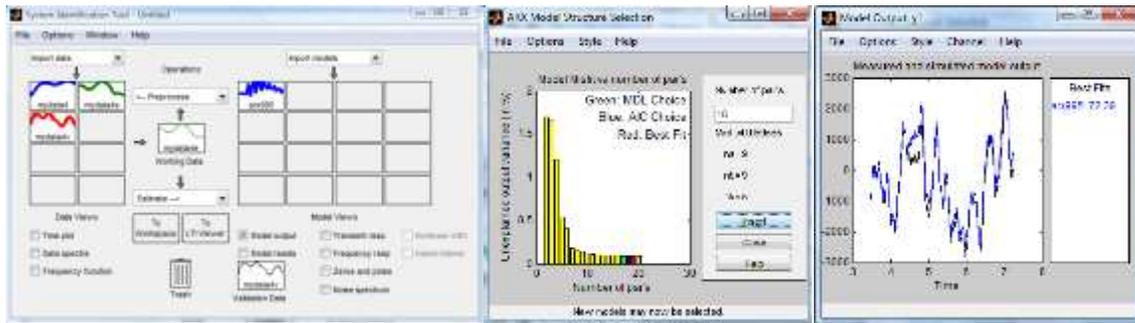


Figure 15 – System identification GUI and model selection windows using Arduino

Also, the time history of the Simscape-model’s parameter estimation and its output-estimate validation, using Simulink Design Optimization toolbox, are shown in Figure 16.

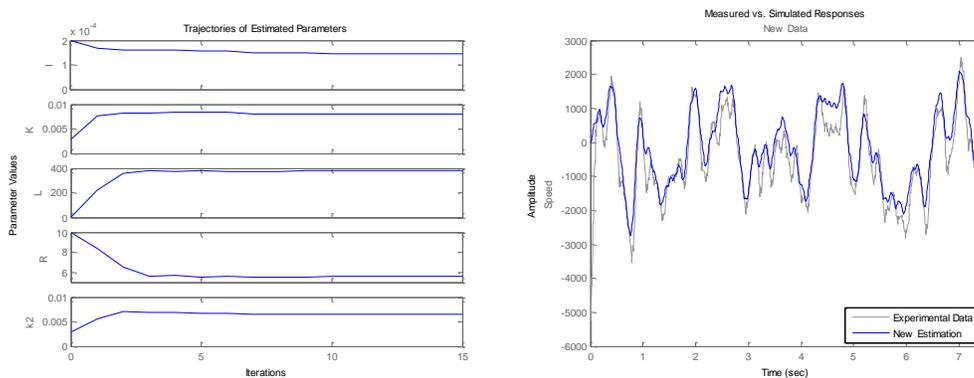


Figure 16 – Parameter estimation and validation of Simscape model using Arduino

The corresponding Bode plots of the estimated transfer-functions of the DC motor platform are shown in Figure 17.

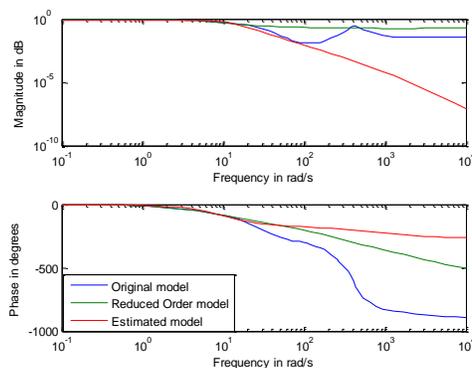


Figure 17 – Bode plot of the transfer functions of the DC motor platform using Arduino

The figure shows the Bode plots of the estimated 13th-order and the 3rd-order transfer functions found using the System Identification toolbox and the 3rd-order linearized transfer function of the tuned Simscape model found using Simulink Design Optimization toolbox. The figure indicates that all three estimated models are approximately equivalent for frequencies up to 20 rad/s.

Given the linearized model of the system, a discrete PID controller with $T_s=0.01s$ is designed for a square-wave reference signal of $\pm 10V$ with 10s period. The results for auto-tuning of the PID control parameters and the closed-loop motor speed simulation are shown in Figure 18.

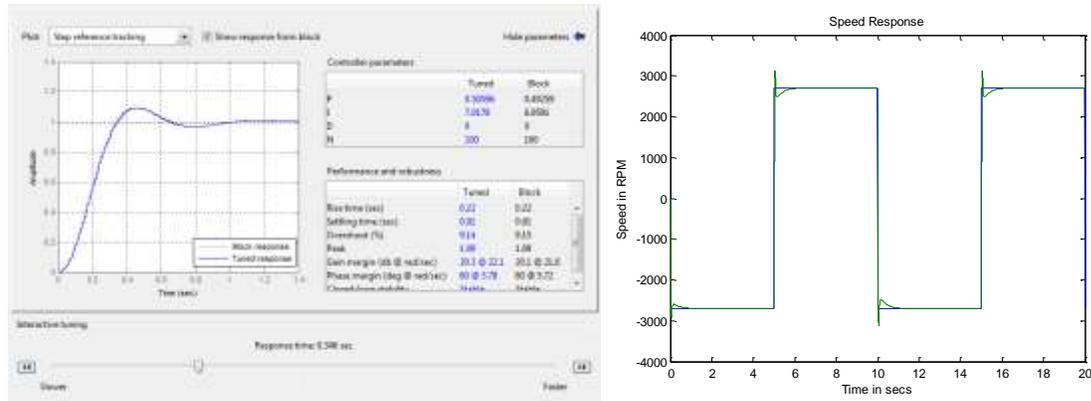


Figure 18 – Auto-tuning results for the PID control parameters using Arduino

The tuned discrete PID controller block is then added to the Simulink model of the Arduino program for actual implementation via Arduino, as shown in Figure 19.

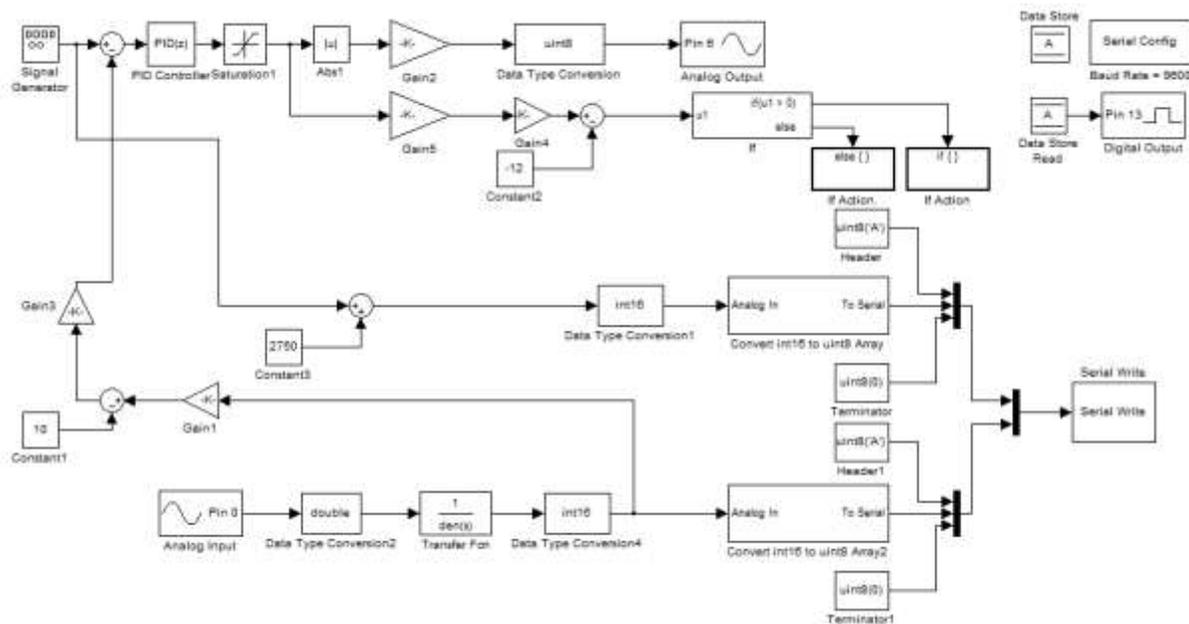


Figure 19 – Simulink model of the Arduino program for control implementation

The motor speed in actual closed-loop hardware implementation with tuned PID controller for a sample time of $T_s=0.01s$, using the Arduino board, is shown in Figure 20.

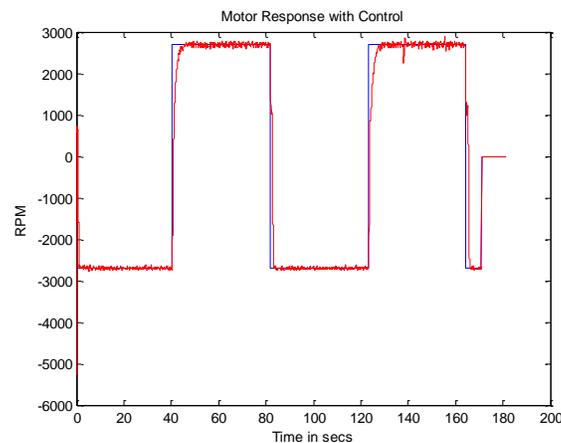


Figure 20 –Actual motor speed for tuned PID controller using Arduino

As seen from the figure, the motor speed in actual implementation is slightly different than the simulation result. However, both responses follow their given commands and are satisfactory. This experiment clearly shows that the tasks of modeling, control design, and rapid-prototyping of a simple mechatronic system can be performed quite easily in Matlab/Simulink environment, using any microcontroller of appropriate computational power, once their Simulink blockset is made available.

A preliminary implementation of these course developments in an exploratory graduate course on model-based control design revealed a noticeable improvement in the students' grasp of the concepts and their confidence in hands-on projects in embedded control and mechatronics. These developments are currently being used in the renovated mechatronics course and will be further modified for continued improvement of the course.

5. Conclusion

This paper reports the developments made in renovating an existing mechatronics course to improve its outcome. Model-based approach was chosen using Mathworks software tools to enhance the course delivery and to enrich the students learning experience. Preliminary implementation of these new developments in an exploratory course showed that all students became more engaged in the course and gained greater confidence in dealing with the challenges of embedded mechatronic system designs and rapid prototyping.

Acknowledgement

This project was funded in part through a curriculum development support by Mathworks, Inc.

References

1. Mathworks Products and Services, <http://www.mathworks.com/products/>.
2. C. Birdsong, "Undergraduate Engineers Develop Hydraulic Servo Control Systems Using Model-Based Design with Simulink," Matlab Digest, Academic Edition, http://digitalcommons.calpoly.edu/meng_fac/31/.
3. D. Eastman, P. Lambrechts, and A. Turevskiy, "Design and Verification of Motion Control Algorithms Using Simulation," Proc. ESC Conference, 2009.
4. Q. Li, W.J. Zhang, and L. Chen, "A Concurrent Engineering Approach for Mechatronic Systems Design," IEEE Trans. Mechatronics, Vol. 6, no 2, pp 161-169, 2001.
5. R. Grygiel and M. Pacholczyk, "Prototyping of Control Algorithms in Matlab/Simulink," Proc. 14th World Multi-Conf. on Systemics, Cybernetics and Informatics (WMSCI 2010), June 29-July 2, 2010, Orlando FL.
6. G.F. Franklin, J.D. Powell, and A. Emami-Naeini, Feedback Control of Dynamic Systems, 5th Ed., Prentice-Hall, 2006
7. D. Xue, Y. Quan, and D.P. Atherton, Linear Feedback Control Analysis and Design with Matlab, SIAM publishing, Philadelphia, PA, 2007.
8. Lubin Kerhuel, "Simulink Blockset for Embedded Target for dsPIC," <http://www.kerhuel.eu/#2>.
9. Microchip's Matlab Device Blocks for MPLAB IDE, http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en538347.
10. Arduino Target, <http://www.mathworks.com/matlabcentral/fileexchange/24675>.
11. Serial Terminal: RealTerm, <http://realterm.sourceforge.net/>
12. Mathworks Recorded Webinars, <http://www.mathworks.com/company/events/webinars/>.
13. Microchip PICkit 2 Development Programmer/Debugger toolkit, http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en023805.