Journal of Control Science and Engineering

Embedded-Model-Based Control

Guest Editors: Sabri Cetinkunt, Shin-ichi Nakajima,
Brad Nelson, and Salem Haggag

Embedded-Model-Based Control

Journal of Control Science and Engineering

Embedded-Model-Based Control

Guest Editors: Sabri Cetinkunt, Shin-ichi Nakajima, Brad Nelson, and Salem Haggag

Editorial Board

Contents

Hindawi Publishing Corporation
Journal of Control Science and Engineering
Volume 2013, Article ID 237897, 2 pages
http://dx.doi.org/10.1155/2013/237897

Editorial
Embedded-Model-Based Control

Sabri Cetinkunt,[1] Shin-ichi Nakajima,[2] Brad Nelson,[3] and Salem Haggag[4]

[1] *Department of Mechanical and Industrial Engineering, University of Illinois at Chicago, Chicago, IL 60607, USA*
[2] *Department of Mechanical and Control Engineering, Niigata Institute of Technology (NIIT), Niigata 9503315, Japan*
[3] *Institute of Robotics and Intelligent Systems, ETH Zurich, 8092 Zurich, Switzerland*
[4] *Faculty of Engineering, Ain Shams University, Cairo 11511, Egypt*

Correspondence should be addressed to Sabri Cetinkunt; scetin@uic.edu

Received 27 June 2013; Accepted 27 June 2013

The industrial practice of mechatronics engineering involves three phases of software development:

Phase 1: control software development and simulation in non-real-time environment;

Phase 2: hardware-in-the-loop (HIL) simulation and testing in real-time environment;

Phase 3: testing and validation on actual machine.

In Phase 1, the control software is developed increasingly by using graphical software tools as opposed to script-based software tools, such as Simulink and Stateflow. Then the developed software is simulated and analyzed on a non-real-time computer environment. The "plant model" which is the computer model of the machine controlled is a non-real-time detailed dynamic model. Simulations and analysis are done in this non-real-time environment.

In Phase 2, the "same control software" is tested on a target embedded control module (ECM). That is, the "same control software" is a C-code which is autogenerated from the graphical diagrams using autocode generation tools. That real-time controller software is run on the target embedded controller module (ECM) hardware which is connected to another computer which simulates the controlled process dynamics in real time. This is called the *hardware-in-the-loop (HIL) simulation*. This process allows the engineer to test the control software on the actual ECM hardware. The computer which simulates the plant model in real time provides the simulated I/O connections to the ECM. The fundamental challenge in HIL simulations is the to find a balance between the model accuracy (hence, more complex and detailed models) and the need for real-time simulation. As the real-time modeling capabilities are improved, virtual dynamic testing and validation of complete machines using HIL tools will become a reality in engineering design and development processes for embedded control systems. HIL is the testing and validation engineering process between pure software simulation (100% software) and pure hardware (100% actual machine) with all its hardware and embedded software, where some of the components are actual hardware and some are simulated in real-time software. The pure software-based simulation cannot capture the real-time conditions in sufficient detail to provide the necessary confidence in the overall system functionality and reliability. However, pure hardware testing is quite often too expensive due to the cost of actual hardware, its custom instrumentation for testing purposes, and team of engineers and operators involved in the testing. Furthermore, some tests (especially failure modes) cannot be tested or are very difficult to test (i.e., flight control systems) on the actual hardware.

HIL tools have been developed rapidly in recent years such that some of the hardware components of the control system are included as actual hardware (such as electronic control unit (ECU), engine, transmission, and dynamometer) and some of the components are present in software form running in real time and its results are reflected on the control system by a generic simulator (i.e., dynamometer which represent the load on the powertrain based on the machine dynamics and operating conditions). Early versions of HIL simulations were used to test the static input-output

behavior of the ECM running the intended real-time control code. Modern HIL simulation and testing are performed for dynamic testing, as well as static testing, where the I/O to ECM is driven by dynamic and detailed models of the actual machine.

In Phase 3, the ECM with the control code is tested on the actual prototype machine. First, all of the I/O hardware is *verified* for proper operation. The sensors and actuators (i.e., solenoid drives and amplifiers) are *calibrated*. The software logic is tested to make sure all contingencies for fault conditions are taken into account. Then, the control algorithm parameters are *tuned* to obtain the best possible dynamic performance based on expert operator and end-user comments. The performance and reliability of the machine is tested, compared to benchmark results, and documented in preparation for production release.

In this special edition, we have tried to put together a set of papers to address these essential aspects of the modern mechatronics engineering practice. We hope that it will be a useful collection for our readers.

Sabri Cetinkunt
Shin-ichi Nakajima
Brad Nelson
Salem Haggag

Hindawi Publishing Corporation
Journal of Control Science and Engineering
Volume 2013, Article ID 973141, 9 pages
http://dx.doi.org/10.1155/2013/973141

Research Article

Real Time Monitoring of Diesel Engine Injector Waveforms for Accurate Fuel Metering and Control

Q. R. Farooqi,[1] B. Snyder,[1] and S. Anwar[2]

[1] *Cummins Inc., Columbus, IN, USA*
[2] *Purdue School of Engineering & Technology, IUPUI, 723 W. Michigan Street, Indianapolis, IN, USA*

Correspondence should be addressed to S. Anwar; soanwar@iupui.edu

Received 27 February 2013; Accepted 24 April 2013

Academic Editor: Sabri Cetinkunt

This paper presents the development, experimentation, and validation of a reliable and robust system to monitor the injector pulse generated by an engine control module (ECM) which can easily be calibrated for different engine platforms and then feedback the corresponding fueling quantity to the real-time computer in a closed-loop controller in the loop (CIL) bench in order to achieve optimal fueling. This research utilizes field programmable gate arrays (FPGA) and direct memory access (DMA) transfer capability to achieve high speed data acquisition and delivery. This work is conducted in two stages: the first stage is to study the variability involved in the injected fueling quantity from pulse to pulse, from injector to injector, between real injector stators and inductor load cells, and over different operating conditions. Different thresholds have been used to find out the best start of injection (SOI) threshold and the end of injection (EOI) threshold that capture the injector "on-time" with best reliability and accuracy. Second stage involves development of a system that interprets the injector pulse into fueling quantity. The system can easily be calibrated for various platforms. Finally, the use of resulting correction table has been observed to capture the fueling quantity with highest accuracy.

1. Introduction

In order to further improve the fuel efficiency of a diesel engine, it is imperative to use optimal amount of fuel injection that will produce the demanded power while meeting the emission requirements. As such most diesel engine manufacturers, such as Cummins, Inc., use closed-loop test on a hardware-in-loop (HIL) bench which is a very important step in the performance testing of diesel engines. In order to carry out the systems' performance analysis, the model of the engine and all other components of the vehicle are run on a real-time computer that simulates the real vehicle. The ECM is fed with all the sensor signals it expects in a real vehicle, in real-time, from emulated sensors using required hardware. However, the real time model fails to run the closed loop real time simulation properly without the accurate information on fueling quantity being injected. The ECM calculates the desired fueling quantity with the control algorithm that takes into account all the required sensor feedbacks at each time step. Finally the injector "on-time", the amount of time the injector should inject the fuel into the cylinder, is looked up from a fuel on-time table, corresponding to the fueling quantity that is to be injected and the operating common-rail pressure. The corresponding electrical pulse is sent to the injector stators or the inductor load cells that emulates the injectors. This research investigates whether the inductors, instead of the injectors, are appropriate to be used in a closed loop bench if necessary correction measures can be taken to adopt this cheaper solution. It also investigates different thresholds in order to identify the one that works the best to capture the correct "on-time". The experimental results show that the double threshold scheme, with start of injection at 0.1 V and end of injection at 3 V, captures the on-time with the least amount of errors.

This work involves the use of FPGA-based data acquisition system having different threshold approaches with different FPGA configurations of circuitry. The FPGA hardware allows the usage of its prebuilt logic blocks and programmable routing resources to configure the silicon chips to implement custom hardware functionality [1] providing hardware-timed

speed and reliability. The real-time HIL simulation requires hardware-timed speed and reliability, which is the reason for selecting FPGA hardware. Reyneri et al. [2] presented their work with a complete HIL test bench for a common rail injection system where they demonstrated a codesign technique that integrated codesign and cosimulation of hardware (HW) and software (SW) which constituted the HIL bench. They used eight FPGA processors, one PC, one analog-to-digital (A/D), digital-to-analog (D/A) board and a data acquisition board, in the test bench, in addition to the common rail test bench and the cosimulation in CodeSimulink environment. Predefined voltage waveform computed on the basis of the required current waveform and the electrical model of the injector was sent to the injectors. Their work focused on testing the ECM performance, which requires the injected fueling quantity to be sensed and fedback to the software simulation running in the RT, which is different than the work we present her. The authors [2] used an ad-hoc hardware signal generator based on FPGA that powered the H bridges for the injectors. They used open-loop generation of current waveform. However, they tuned the inductor load cells, that is, R-L circuits with estimated R and L values. They employed neuro-fuzzy methodologies that characterized the injectors, that is, the electrical parameters, in order to tune the inductor load cells that allowed them to weight the fuel injected with cheaper load cells and still obtain desired precision. FPGA hardware and 8-channel A/D converter with about 20 kHz sample rate were used during the injector characterization process.

Saldaña-González et al. [3] presented-FPGA-based hardware implementation that takes the digitized voltage signals produced by the data acquisition electronics of the photomultiplier tubes and processes them to allow identifying events. The data was then used to determine the strength and positions of the interactions based on the logic of Anger to form a planar image that allows reconstruction of 2D image for medical diagnostics in a gamma camera in real time. Pozniak [4] presented the application of FPGA the based-multichannel, distributed, synchronous measurement systems for triggering, and data acquisition used in high-energy physics (HEP) experiments. Turqueti et al. [5] presented design and implementation of a 52 microphone MEMS array embedded in an FPGA platform with real-time processing capabilities.

The objective of this research is to investigate the variability and inaccuracy inherent in the injector monitoring process using various approaches and conclude the most cost effective and reasonably accurate system. The research investigates the variability involved in the fuel measurement system used for closing the loop between the plant models and the ECM on a CIL bench. We also investigate if the inductor load cells that emulate the injectors are adequate to be used in the CIL bench and how much tradeoff is needed to use the cheaper inductors instead of the injectors and if the inductor load cells or the injectors show certain offset that can be corrected on the benches through proper tuning. Another objective is to find how much variability is present from pulse to pulse, from injector to injector, and over different operating conditions. Finally, the system needs to be easily calibrateable to be used with different platforms. Therefore, a test sequence is needed to generate a correction table that will be able to capture the fueling quantity with best possible accuracy, within the limitation of hardware. This research also aims at reducing the latency in delivering data and improved robustness of the CIL system.

2. Experimental Setup

The performance of a diesel engine, both in terms of fuel efficiency and emission, is heavily dependent on the fuel system that delivers fuel into the engine cylinder, which takes care of precisely controlling the injection timing, correcting pressure of injection to ensure proper mixing of air and fuel taking into account proper fuel atomization, and other critical parameters. Cummins engines are controlled to ensure precise control of the fuel injection into the cylinder with the advanced fuel system that constitutes common-rail, pump, and high precision injectors. The necessity to reduce fuel consumption, exhaust gas emissions, and engine noise has led to advanced technologies being employed in the fuel systems, replacing the mechanical injection system.

In general, the common-rail architecture employs a common pressure accumulator or high pressure storage called rail. This rail is fed by a high pressure fuel pump that could be driven at crankshaft speed (engine or twice the camshaft speed). Sometimes high pressure radial pump, independently from the engine rate, generates high pressure at the rail. High pressure injection lines connect the common rail to the fuel injectors. ECM controls the pressure at the rail through inlet metering valve (IMV). The ECM generates the injection pulse, which controls the opening of the injectors with electromechanical actuators. The ECM calculates the fuel quantity needed based on a predefined characteristic curve, the engine model, the driver's intentions via the accelerator position, engine speed, torque, temperature, acceleration, and so forth. The electronic control facilitates flexibility in injection timing and metering control, reduced cycle-to-cycle and cylinder-to-cylinder variability, as well as, tighter control tolerances, and increased accuracy over very long periods of operations. Figure 1 shows the layout of the common-rail architecture of the fuel injection system [6].

The common rail system includes the following components (Figure 1):

(i) high pressure fuel pump,

(ii) rail for fuel storage and distribution,

(iii) injectors,

(iv) electronic control module (ECM).

The rail serves as a fuel accumulator to maintain a relatively constant pressure at all fueling rates used by the engine. The fuel volume in the rail also dampens pressure oscillations caused by the highpressure pump and the injection process. From the rail, the fuel is supplied at constant pressure to the injectors via high pressure pipes. The ECM generates current pulses which energize each injector solenoid valve in sequence and define the start and the end of each injection event per engine cycle. The common rail system can generate

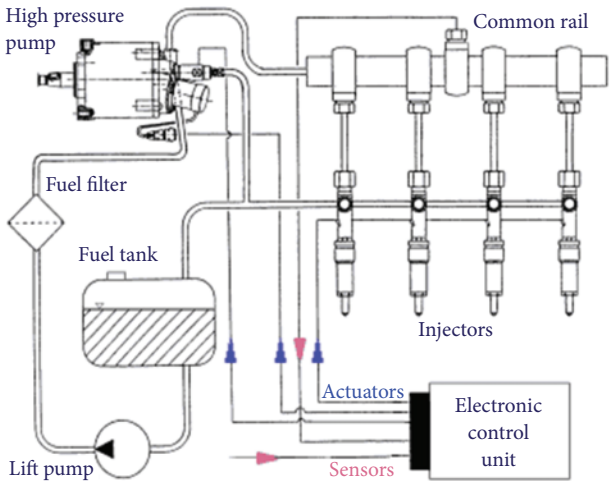


Figure 1: Schematic of the common-rail injection system.

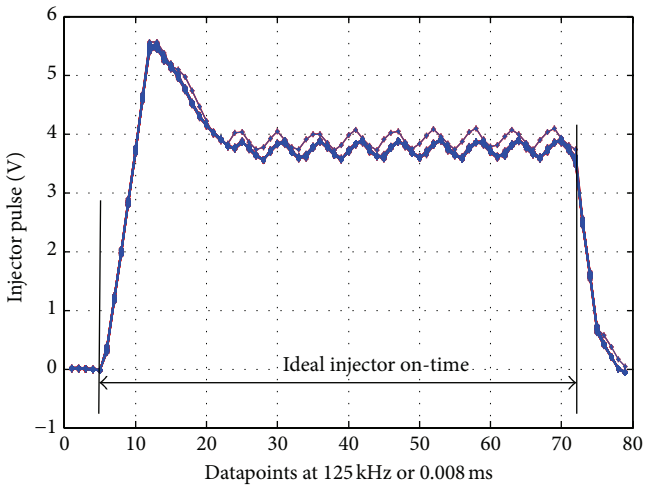


Figure 2: Injector pulse captured by the data acquisition system.

more than one injection per engine cycle and give more flexible control of the rate of injection compared to other injection system designs.

This research addresses the most important attribute of the fuel injection system, that is, metering of correct amount of fuel into the cylinder, in the application of HIL testing of the control algorithm. The control system is developed to calculate the correct amount of fuel to be injected by the fuel system in terms of fuel quantity, which is implemented by the fuel system by converting the fuel quantity into duration in time to inject the fuel at a given common-rail pressure. In order to carry out a hardware in the loop simulation, the simulation model needs the accurate measurement of the fuel being injected in order to carry out the accurate calculation to simulate the engine performance. The ECM generates the fueling signal in terms of electrical pulse to the injectors. The voltage waveform constitutes a high initial boost voltage to overcome the inertia of the injector mechanics, followed by a lower constant voltage which holds the injector nozzle to open position for the desired period of time. The hardware used in this research senses this electrical pulse and real time system that utilizes the FPGA personality and DMA transfer converts the pulse back into the fuel quantity. The electrical signal captured by the sensors does not distinctly indicate the start of injection and end of injection, which is the critical parameter to be figured out in this research, in order to calculate the most accurate measurement of injector on-time. The injector on-time, that is, the period of time the injector remains open to allow the fuel to be injected. The injection pulse captured is shown in Figure 2. Ideally, the injection on-time corresponds to the length of time between the time the injector signal starts to rise from zero value and the moment it starts to fall from the steady voltage value that is held during the injection period. Figure 2 clearly delineates the challenge involved in identifying the start of injection and end of injection.

The start of injection can be identified by the voltage value being over 0 V; however, the noise involved causes error in the identification. On the other hand, the steady value of voltage maintained at the time of injector being open is visibly noisy, and the approaches taken to identify the end of injection were to consider the slope of the voltage drop or identify a threshold value. The latter approach turned out to be more suitable, when coupled with identifying a threshold to distinguish the start of injection as well.

Another important parameter investigated in this research is the variability involved in the injector pulses captured by the proposed method. The importance of delivering the correct fuel quantity with consistency is very important in the hardware-in-the-loop test since the purpose of using simulation instead of real engine and hardware is largely the repeatability of tests, in addition to cost reduction. To determine the repeatability of the injection pulse monitoring system, standard deviation of the captured on-time was used as an indicator. The fuel quantity being injected by the ECM was overridden through the CAN bus, while capturing it by the system. The fuel quantity being identified is expected to be exactly the same as the value being overridden. However, the inherent variability was calculated by the standard deviation. In later stage of the research, injection on-time was directly overridden instead of the fuel quantity. The on-time was held at a steady value and the system logged the on-time captured by the proposed system. Different variability was obtained with different approaches of injection on-time capture.

The research aimed at identifying the optimal approach, in terms of cost of implementation, the accuracy, repeatability, and variability involved in capturing the correct fuel quantity being injected by the injector.

Analog input module NI-9205, along with Xilinx Virtex-5 Field Programmable Gate Arrays (FPGA) hardware and the direct memory access (DMA) transfer capability in the compact reconfigurable input-output (CRIO) real-time (RT) controller, has been utilized to capture the injector voltage signal generated by the ECM. Since the analog input module has a specification of ±10 V, and the peak voltage of the injector signal is 12 V, voltage dividers with 2 V : 1 V ratio were used to capture the signals. The analog signals were logged at

different data acquisition rates, and the voltage signals were postprocessed in MATLAB to obtain the on-time with various thresholding approaches in the first investigation phase. The shot-to-shot variability, that is, the variation of captured fueling quantity from pulse to pulse was compared with the standard deviation in different thresholding approaches, as well as different operating conditions. A different operating condition comprises different engine speed, common rail pressure, fueling quantity, and injector or inductor load cells on all six injectors or inductors. In the second phase, a real-time application, along with the FPGA bit-stream that imprinted the desired circuitry into the hardware, was built, compiled and deployed to the real-time target that could interpret the fueling quantity from the analog signals. The FPGA circuitry allowed the generation of engine speed signal (ESS) and engine position signal (EPS) to be generated to simulate the engine speed.

Since the injector signal generated by the ECM is of importance in this research, and the whole HIL bench for closed loop testing is not required, a separate bench was developed for this research to run tests through different static operating points of different variables in an open-loop testing environment. Figure 3 shows the schematic of the bench developed for this research. The windows host computer runs test sequence to go over different values of different variables under consideration. National Instrument's TestStand software was used to run the test sequence. At the beginning, the test sequence establishes a session through CUTY (a software interface) that allows the Windows host PC to communicate with the CAN communication. The Cummins software called Calterm was used to monitor the parameters being overridden on the CAN bus.

The electrical pulse generated by the ECM is passed through a load, either the real injector stators or the inductors that emulates the injectors in the CIL bench. In this research, the focus is to interpret the electrical signal generated by the ECM for the injector and provide the fueling quantity injected to the RT simulation. Therefore, the key issue of this research is to capture the injector pulse with highest accuracy at a reasonable cost. The research investigates if the system can continue to capture the correct fueling quantity if the ECM commands for fueling over a prolonged period of time. The analog injector signal could be converted in several ways; however, the research identified the simplest and most effective way to capture it. The fueling quantity injected or the injector "on-time" was overridden with the software CUTY and CAN bus. Therefore, the real-time computer used in this project was the National Instrument's Compact Reconfigurable Input Output (CRIO). The CRIO contains a real-time processor with a chassis, featuring built-in elemental I/O functions such as the FPGA read/write function that provides a communication interface to the highly optimized reconfigurable FPGA circuitry. The chassis contained one analog output module to generate the emulated common-rail pressure sensor signal, analog input module to capture the injector voltage signal, and digital output module to generate the EPS and ESS signals. Windows host PC communicates with the CRIO through ethernet connection. National Instruments TestStand and LabVIEW have been used to run the tests. Real-time applications were compiled, built, and deployed to the CRIO, including the FPGA bit files that imprinted the required FPGA personality. The automated test sequence on NI Teststand establishes connection with the ECM through the CAN bus, with the CUTY software for the ECM. CUTY is a Cummins proprietary software that has been used for accessing the values of the parameters on the CAN bus, as well as overriding the values of required parameters. The Teststand sequence overrides the value of the fueling quantity to be injected or the injector "on-time" through the data link. The sequence also communicates over the ethernet connection with the real-time application running on the CRIO to vary the simulated engine speed through network variables. The EPS/ESS signals corresponding to the simulated engine speed are generated by the FPGA personality, according to the crank angle of the engine. The ECM requires the common-rail pressure signal and the EPS/ESS signals in the corresponding pins to generate the injector signal. The common-rail pressure is varied over different values by the test sequence running on the NI Teststand on the host PC, through Ethernet connection to change the values in the real-time application running on the CRIO. The corresponding pressure sensor signal is generated by the analog output module, by emulating the sensor. Different test sequences were developed in different stages of the experimentation. The real-time application contained the FPGA personality that generated the desired EPS/ESS signal, corresponding to the engine speed; the RT application switched over different channels of the analog modules since the analog module had only one analog to digital converter, carrying out the DMA (direct memory access) transfer from the FPGA module to the memory of the RT computer. It created separate files for each state. "Different states" refers to different engine speeds, different common-rail pressures, different fueling quantities or the "on-time" that is overridden on the ECM, in case of injector stators or the inductors.

Figure 4 shows the injectors, the inductors, and the FPGA hardware. The research investigated if inductors are good enough for closed-loop testing, and it was found that they are not. Injector stators were used from production injectors of Cummins engines. The bench hardware made by Cummins provided the electrical protection and necessary systems to transform the line voltage to low as power DC voltage to supply electronic circuits and high power supply as well to drive the electrical injectors or load cells. The NI CRIO-9014 [7], along with NI 9111 chassis, having analog output, analog input, and digital I/O cards, is shown on the right side of the hardware in [8]. The NI 9205 [9] analog input module has been a key feature for in this research. The NI 9205 features are 32 single-ended or 16 differential analog inputs, 16-bit resolution, and a maximum sampling rate of 250 KS/s. Each channel has programmable input ranges of ±200 mV, ±1, ±5, and ±10 V. To protect against signal transients, the NI 9205 includes up to 60 V of overvoltage protection between input channels and common (COM). In addition, the NI 9205 also includes a channel-to-earth-ground double isolation barrier for safety, noise immunity, and high common-mode voltage range. The 4-slot CRIO-9111 [8] chassis has Xilinx Virtex-5 reconfigurable I/O FPGA core capable to automatically

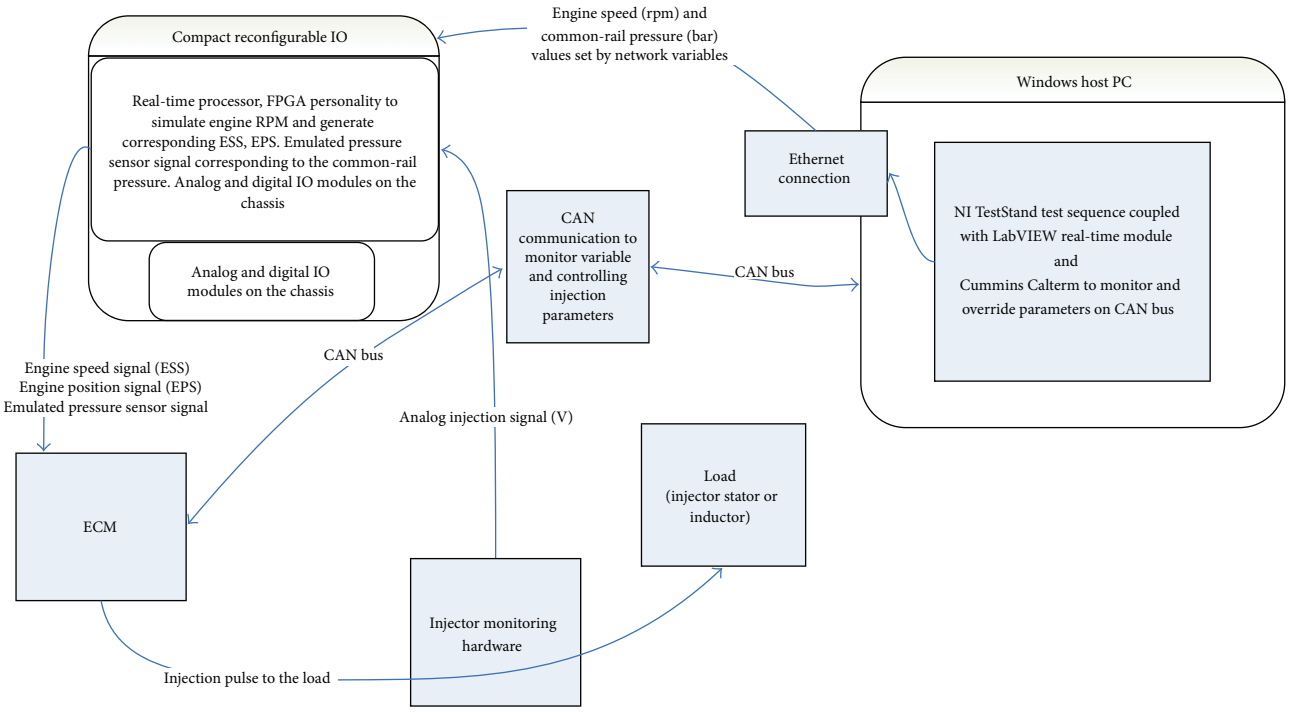


Figure 3: Layout of the bench for this research.

Figure 4: Injector stators used as load (top). Inductors that emulate real injectors as load (middle). FPGA hardware to capture the injection signal (bottom).

synthesize custom control and signal processing circuitry using LabVIEW. The research employed NI 9264 [10] analog output module to generate the pressure signal in order to emulate the pressure sensor. The ECM requires the pressure signal to calculate the injector on-time (ms) in order to inject certain amount of fuel. The research also utilized NI 9401 [11] 8 channel, 5 V/TTL high speed bidirectional digital I/O module to generate the engine position signal (EPS) and engine speed signal (ESS) to feed the ECM with the simulated engine speed. The test setup includes the six voltage dividers to accommodate the voltage provided by the hardware into the NI 9205 module [9]. Other hardware used in the bench were the inhouse power supply for the ECM and the electrical hardware, the Tektronix TDS 2024B oscilloscope, PEAK adapter to convert CAN messages and transfer into the computer, the CAN terminators to establish the CAN bus, and so forth.

This research employs a CRIO system offered by National Instruments. It contains an integrated real-time controller and a chassis, with a communication interface with a highly optimized reconfigurable FPGA circuitry, that contain slots for different modules used. National Instruments helps the users involved in the development of mechatronic control systems by providing hardware and software solutions in order to accelerate the development and testing of such systems. This supports creating real-time applications in LabVIEW, building and deploying the files into the RT system to implement real-time environment for any user-defined HIL Bench which falls into the targeted I/O criteria. The CRIO system used in this research is a real-time system for performing fast function prototyping. The CRIO-9014 runs the NI LabVIEW real-time module on the VxWorks real-time operating system (RTOS) for extreme reliability and determinism. With the CRIO-9014 real-time controller, one can use the leading VxWorks RTOS technology to quickly design, prototype, and deploy a customizable commercially available off-the-shelf (COTS) embedded system using LabVIEW graphical programming tools.

3. Experimental Results

The experimentation was carried out on the experimental bench to find the most cost effective, efficient, recalibratable, and reproducible solution to the injector monitoring problem with the following variable parameters under consideration:

(i) fuel quantity or injector on-time (ms),

(ii) engine speed,

(iii) common-rail pressure,

(iv) two different loads, that is, injectors or cost saving inductors to simulate injectors,

(v) six different injectors or inductors,

(vi) different thresholds.

In order to implement the injector monitoring system in the hardware-in-the-loop system, the system is required to maintain good accuracy in capturing the correct amount of fuel quantity over a large range of fueling, engine speed, and common-rail pressure with as little variation as possible. The research also investigates if the accuracy varies from injector to injector. Since the system, if satisfies requirements, is going to be implemented in a large number of hardware-in-the-loop benches, the cost of implementation is an important factor to consider as well.

The research begins with varying all the variables and consecutively ruling out some of the variations if found to have insignificant influence over the accuracy of the system. The data acquisition hardware available from NI had limitation in sampling rate. Therefore, initially only one NI-9205 module with 20.8 kHz sampling rate at each channel was considered to be used for all six channels.

To identify the start of injection and the end of injection, different thresholds were considered, narrowing down to the most effective approach. Initially, the end of injection was identified using the slope of injection pulse, which was not very successful due to the noise involved in the signal captured. Therefore, thresholds were used to identify the SOI and EOI, having only one threshold for both ends or two thresholds. The initial experiments show that the influence of varying common-rail pressure is comparatively insignificant. Therefore, the tests were carried out at varying engine speeds and fueling quantities with different threshold approaches on both kinds of loads. The sampling rate turned out to be the most significant factor in the accuracy of the system. Since the injector on-time remains the same with the constant fueling quantity over varying engine speed, it was expected to have the same accuracy. However, the experimental results show that the accuracy varies over different engine speeds.

Initially, tests showed that the accuracy of the system is not significantly dependent on common-rail pressure; therefore, tests were run at 1200 bar common-rail pressure over different engine speeds and fueling quantities for both injector stators and the inductors, six of them each. The injection pulses were logged in the form of discrete voltage values with 20.8 kHz sampling rate at each injector channels with a precision of 1 V, which was later increased to 0.0156 V precision value. The injection voltage values were logged in .tdms format. National Instrument's data analysis software DIAdem script was used to convert the .tdms files to .mat files in order to postprocess the data on MATLAB. The fueling "on-time" was extracted using various single thresholds or double thresholds in MATLAB. Single threshold approach uses same threshold value for both the start of injection (SOI) and end of injection (EOI). The SOI threshold is the value that determines when the injection has started; that is, as soon as the voltage value goes above the SOI threshold, the injection is considered to have started. Similarly, the EOI threshold is the value that determines when the injection has ended, that is, as soon as the voltage value goes below the EOI threshold, the injection is considered to have ended. In the first phase of the experiment, the double threshold approach considered EOI at the point where the voltage value starts to drop from a steady value; that is, instead of using a threshold to identify the EOI, the code considered five consecutive data points, and if the voltage value kept on falling through five points, the third point was considered the EOI point. The test sequence goes over different values of engine speeds and fueling quantities to be injected. The extracted pulse lengths are measured in milliseconds. The mean of all the pulse-lengths are calculated for each injector channel at each state, in both cases of injectors and inductors. The expected fueling "on-time" is the value overridden on the ECM. Therefore, the error in the fueling quantity was calculated at each of the states on the mean values using the following equation.

$$\text{error}\,(\%) = \frac{\text{ontime}_{\text{mean ontime}} - \text{ontime}_{\text{expected ontime}}}{\text{ontime}_{\text{expected ontime}}} \times 100. \tag{1}$$

Double threshold and single threshold at 2 V showed less variation from pulse to pulse; however, the variation is pronounced at 1500 rpm and 3000 rpm engine speeds. The mean of percentage errors at each state was calculated and plotted to compare the performance of the system with injectors or the inductors used as loads. Following plots show the comparison with different threshold approaches. Figures 5(a) and 5(b), and 6 show that the errors with inductors are lot more than the injectors. They signify the fact that, if cheaper solution, that is, inductors are used as the loads instead of using six production injectors for each bench, single threshold at 2 V is the best option. However, injectors show better results in double threshold approach. These experimental results pave way for further experimentation, to investigate the performance of the system with higher precision and higher sampling rate. These plots imparts us the knowledge, how much error can be expected, should we implement it. However, the error% is unacceptable for the CIL application since the CIL application requires higher accuracy at a fueling quantity as low as 10 mg/stk at a lower pressure lower than 1200 bar, which would certainly lead to a much more error.

It is evident from the previous experimental result that, using the highest sampling rate available with double threshold provide the best estimate of the calculated fueling quantity by the ECM; however, there is variability involved in the process. In order to implement this system in the HIL bench,

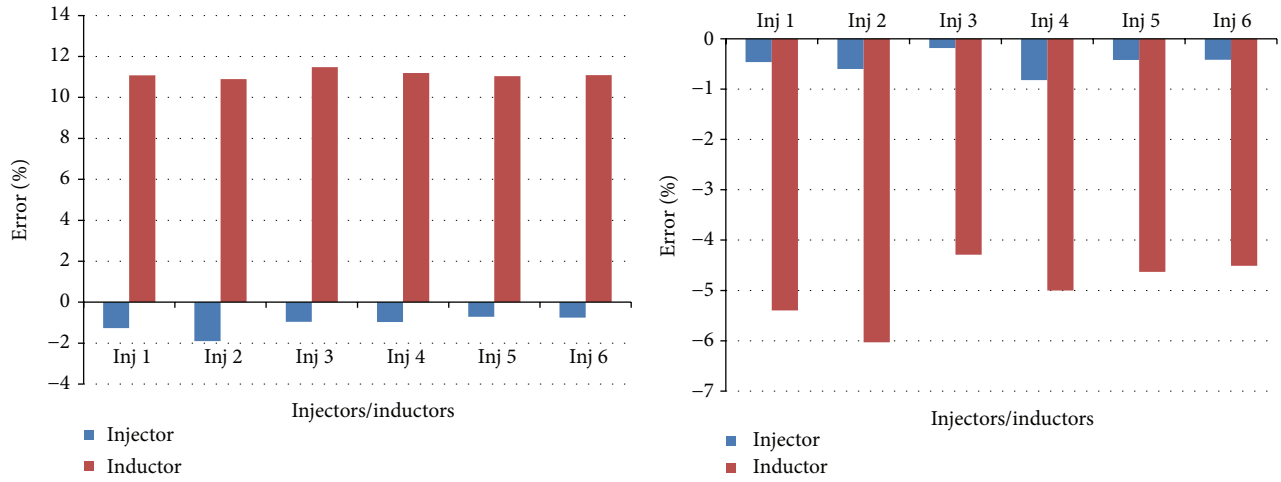


Figure 5: (a) Mean error (%) using single threshold at 0 V. (b) Mean error (%) using single threshold at 3 V.

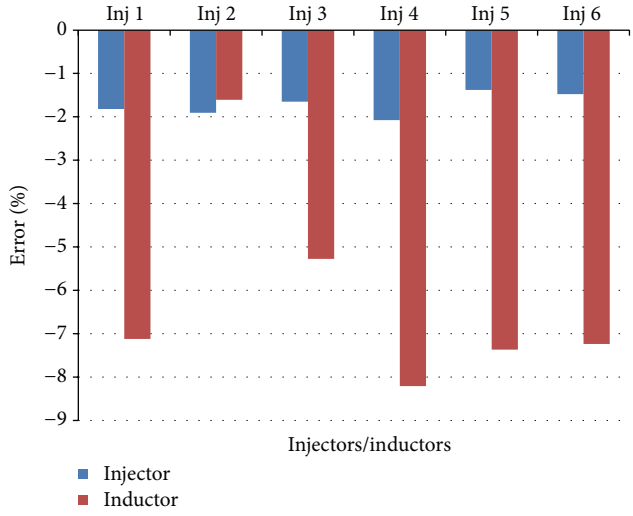


Figure 6: Mean error (%) using double threshold.

Table 1: DOE fixed factors and their levels.

Factors	Levels			
Engine speed (rpm)	750	1500	2250	
Pressure (bar)	600	1200	1800	
Fueling quantity (mg/stk)	10	50	100	150

it is critical to know the variability involved and the factors that contribute to the variability in order to have confidence in the system. And a correction model can be sought in future to make the system as accurate as possible over the entire operating range. Three fixed factors have been identified, that is, engine speed, common-rail pressure and fueling quantity at various levels in Table 1. Fifty replicates, that is, pulses were collected over randomized sequence of factor levels were collected using double threshold with SOI at 0.5 V and EOI at 2 V, with six injectors, as well as, six inductors.

The six injectors/inductors also showed variation in performance, however, the six injectors/inductors have been considered random factor, since they are expected to be identical and only the variability involved in the production process of the injectors contribute to the variability in the accuracy of the fueling quantity estimated by the system.

A full factorial design of experiment (DOE) was carried out with randomized run order of the fixed factors both on injectors and inductors, having the error percentage in the estimation of the fueling quantity being the response variable. The DOE result with 95% confidence interval showed that, all the fixed factors and interactions were contributing to the rejection of the null hypothesis that, the data collected over all the levels of all the factors represent the natural variability of only one process. The mathematical model of this experiment that uses three way analysis of variance (ANOVA) and design is,

$$E = \mu + S_i + P_j + F_k + SP_{ij} + SF_{ik} + PF_{jk} + SPF_{ijk} + \varepsilon_{m(ijk)}, \quad (2)$$

where S is the engine speed, P is the common-rail pressure, F is the fueling quantity, i, j, k are 1 through 50, and $m = 1, 2, 3, 4$.

Six injectors and six inductors were used in this research and it has been found that, the variability of error is similar. However, the mean error percents are much higher with inductors than with real production injectors as load on the HIL bench. Figure 7 shows that, the error percent is distinctly higher with inductors than with the injectors, however, the consistency of the error with both the injectors and inductors points out the fact that, the error can be corrected with regression algorithm.

Figure 8 shows that the difference in mean error percents is significant between injectors and inductors and the error varies over a larger range in response to the common-rail pressure with inductors than the injectors.

Figure 9 shows the large influence of fueling quantity on the mean of error percents with inductors as opposed to

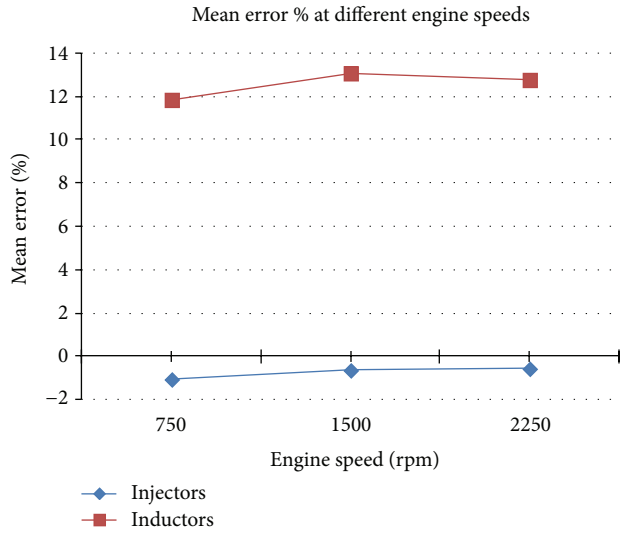


Figure 7: Comparing error % against engine speed for injectors and inductors.

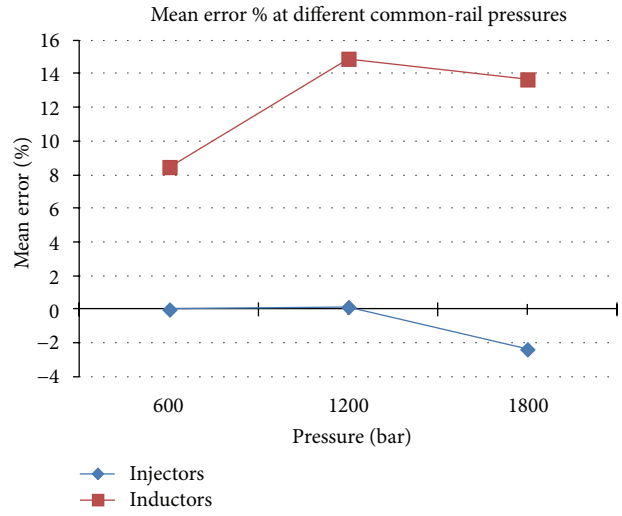


Figure 8: Comparing error % against bar pressure for injectors and inductors.

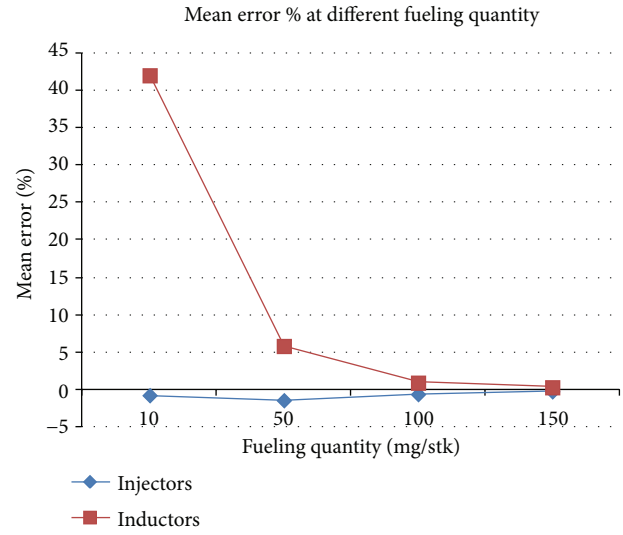


Figure 9: Comparing error % against fueling quantity for injectors and inductors.

injectors. However, the mean error shows a consistent trend that can be corrected with a regression algorithm

Previous analysis of variability showed that the variability of standard deviation of error percent with injectors at different engine speeds is significantly (95% statistical confidence) different from the variability of the whole process of variability, considering the fixed factors. However, incorporating the six injectors and carrying out mixed model ANOVA exposed the fact that the interaction of engine speed and fueling add significantly different variability distribution compared to the normally distributed variability of the whole process of standard deviation. It signifies that, if the engine speed and fueling do not vary, the standard deviation of error maintains the same distribution around the mean standard deviations of error percent. Interaction of engine speed and fueling generates statistically significant different distribution of standard deviation of error percent; that is, varying pressure or using different injectors does not contribute to the variability of the standard deviation of error percent. On the other hand, the inductors with only the fixed factor model shows that all the fixed factors fall under the same normal distribution of standard deviation of error percent. However, when the mixed model of ANOVA was carried out, it pointed out that, the interaction of engine speed with all other three factors, that is, pressure, fueling quantity and inductor number add different distribution that is statistically significant. Therefore, the system's variability depends on all the variable factors, including different inductors. The standard deviation of all the values of standard deviations of error percent with injectors in logarithmic scale is 3.48534, while with inductors, the standard deviations of all the values of standard deviations of error percent in logarithmic scale is 3.14255.

The research involved development of a bench setup that is capable to test injector performance for fueling quantity monitoring, with automated test sequence that goes over all the predefined steady state operating points, using custom steps in NI Teststand to establish communication through CUTY and CAN bus to override parameter values on ECM, as well as monitoring the CAN bus to make sure that the correct values are being registered by the ECM from the emulated sensors. The bench uses FPGA personality to model the engine crank shaft rotation by generating high speed EPS/ESS signal, in addition to pressure sensor signal. The bench is capable to monitor the analog injector pulse, use double thresholds to capture the fueling on-time, and feed the engine model running on real-time computer with the correct values of fueling quantity through high speed DMA transfer using FIFO method. Future research can be carried out on this bench with different types of injectors, such as, injectors using piezoelectric technology, without using the expensive resource, that is, the fully capable closed loop test development bench. The research shows the high amount

of error that persists if single AI module is used for more than one injector monitoring. There are certain regions of operation with low error, therefore, the cheaper solution can be selected for an application that does not operate in high error region or in cases when the tests carried out are not susceptible to this high error. Statistical analysis was carried out on the system that uses one module allowing data acquisition at 125 kHz on each injector with differential input, which is the most expensive solution to implement on the closed loop HIL test bench. The research also compares the performance of the system with the production injector and the inductors load cells that emulates the injectors. The statistical analysis shows that, the expensive injectors can be replaced by inductor load cells if an error correction algorithm is incorporated into the system, since it showed about 40% error at lower fueling quantity, which may potentially cause unstable solution of the idle state of engine simulation on the bench. On the other hand, the injectors perform with very low error percent, that is, −2.38045% to 0.13551% error with less than 4% standard deviations, unless the system is used at an engine speed as high 2250 rpm. The mixed model ANOVA, which is a relatively new technique to carry out multivariate ANOVA exposed the fact that, the variability of the error percent varies with 95% statistical confidence, over different interaction values of engine speed and fueling quantity when the production quality injectors are used, while the value varies with the impact on all four variables, that is, engine speed, fueling quantity, pressure, and different inductors.

4. Conclusion

Based on the experimental data, it can be concluded that the injector loads are a better choice for the HIL bench to realistically emulate the injector waveforms. This analysis also shows that the variability of the error percent is not influenced by the common-rail pressure or different injectors, which is not the case with inductors. However, the variability is not very different with injectors or inductors in terms of standard deviation of error percent; that is, if the variability range is acceptable for a particular HIL test application, the inductors can be used, provided that the error correction algorithm is incorporated into the system. The proposed system was found to be capable of reducing the latency in the delivery of fueling quantity in closed loop HIL tests on the current benches that utilize CAN message instead. Additionally, the double thresholding approach was found to yield better accuracy and lower variability in capturing the correct fueling quantity. Finally, in order to take the inherent offset in the system into account, a test sequence was developed to generate a fueling correction table for a particular platform that shows a better result in estimating the fueling quantity.

Acknowledgment

This work was made possible by a research Grant by Cummins Inc. to IUPUI (Grant no. T119906).

References

[1] Introduction to FPGA technology: top five benefits, http://zone.ni.com/devzone/cda/tut/p/id/6984.

[2] L. M. Reyneri, B. Bbllei, E. Bussolino, F. Gregoretti, L. Mari, and F. Renga, "Simulink-based codesign and cosimulation of a common rail injector test bench," *Journal of Circuits, Systems and Computers*, vol. 12, no. 2, pp. 171–202, 2003.

[3] G. Saldaña-González, H. Salazar-Ibargüen, O. M. Martínez Bravo, and E. Moreno-Barbosa, "2D image reconstruction with a FPGA-based architecture in a gamma camera application," in *Proceedings of the 20th International Conference on Electronics Communications and Computers*, pp. 102–105, 2010.

[4] K. T. Pozniak, "FPGA-based, specialized trigger and data acquisition systems for high-energy physics experiments," *Measurement Science and Technology*, vol. 21, no. 6, Article ID 062002, 2010.

[5] M. Turqueti, J. Saniie, and E. Oruklu, "MEMS acoustic array embedded in an FPGA based data acquisition and signal processing system," in *Proceedings of the 53rd IEEE International Midwest Symposium on Circuits and Systems (MWSCAS '10)*, pp. 1161–1164, August 2010.

[6] N. Guerassi and P. Dupraz, "A common rail injection system for high speed direct injection diesel engines," SAE Technical Paper 980803, 1998.

[7] NI CRIO-9014 controller, http://sine.ni.com/nips/cds/view/p/lang/en/nid/203500.

[8] NI CRIO-9111 chassis, http://sine.ni.com/nips/cds/view/p/lang/en/nid/206762.

[9] NI 9205 analog input module, http://sine.ni.com/nips/cds/view/p/lang/en/nid/208800.

[10] NI 9264 analog output module, http://sine.ni.com/nips/cds/view/p/lang/en/nid/208807.

[11] NI 9401, 8 channel, 5 V/TTL high speed biderictional digital I/O module, http://sine.ni.com/nips/cds/view/p/lang/en/nid/208809.

Hindawi Publishing Corporation
Journal of Control Science and Engineering
Volume 2013, Article ID 657182, 15 pages
http://dx.doi.org/10.1155/2013/657182

Research Article

Design of Attitude Control Systems for CubeSat-Class Nanosatellite

Junquan Li, Mark Post, Thomas Wright, and Regina Lee

Department of Earth & Space Science and Engineering, York University, 4700 Keele Street, Toronto, ON, Canada M3J 1P3

Correspondence should be addressed to Junquan Li; junquanl@yorku.ca

Received 18 December 2012; Accepted 24 April 2013

Academic Editor: Sabri Cetinkunt

We present a satellite attitude control system design using low-cost hardware and software for a 1U CubeSat. The attitude control system architecture is a crucial subsystem for any satellite mission since precise pointing is often required to meet mission objectives. The accuracy and precision requirements are even more challenging for small satellites where limited volume, mass, and power are available for the attitude control system hardware. In this proposed embedded attitude control system design for a 1U CubeSat, pointing is obtained through a two-stage approach involving coarse and fine control modes. Fine control is achieved through the use of three reaction wheels or three magnetorquers and one reaction wheel along the pitch axis. Significant design work has been conducted to realize the proposed architecture. In this paper, we present an overview of the embedded attitude control system design; the verification results from numerical simulation studies to demonstrate the performance of a CubeSat-class nanosatellite; and a series of air-bearing verification tests on nanosatellite attitude control system hardware that compares the performance of the proposed nonlinear controller with a proportional-integral-derivative controller.

1. Introduction

The development of nanosatellites (with a mass of 1–10 kg) is currently a significant trend in the area of space science and engineering research. The development of CubeSat-class nanosatellites started in 1999 as a collaborative effort between California Polytechnic State University and Stanford University and has achieved great success as a way to efficiently construct and orbit small, inexpensive satellites using commercial technology. CubeSat, in general, is described as a class of nanosatellites ranging from 1 kg, $10 \times 10 \times 10\,\text{cm}^3$, and upwards in 10 cm increments of length. Currently more than 50 research groups around the world are developing CubeSat-class nanosatellites for technology demonstration and scientific and student training missions. A solid model of a typical 1U CubeSat with attitude control systems (ACS) is shown in Figure 1.

Full-scale satellite attitude control systems are generally too large or too expensive to be installed in CubeSat-class nanosatellites [1], so passive attitude control systems have usually been used for nanosatellites in the past [2, 3]. More active attitude control subsystems [4] in CubeSat-class nanosatellites have been implemented with the development of suitable actuators like magnetorquers (torque coils or torque rods) and small-sized reaction wheels [5, 6]. An overview of the proposed ACS design adopted for this study is shown in Figure 2. Currently, commercial nanosatellite torque rods and reaction wheels are too expensive for use in many research nanosatellite projects. The contributions of this research are the development of ACS hardware from off-the-shelf components, complete simulation of the ACS system, and validation testing of the ACS system for attitude control in the lab environment.

In this paper, we first briefly describe the ACS hardware proposed for CubeSat-class nanosatellite missions. We outline the hardware development of the ACS actuators in Section 2. In particular, we describe the sizing and design of the magnetorquers. More discussions on the ACS hardware selection, design, and characterization for CubeSat-class nanosatellite missions, currently under development at York University, can be also found in [7–11]. In Sections 3 and 4, we describe the satellite system models and the results from the simulation study based on this design. In Section 5, we show the ground testing results of the hardware and software system. Section 6 includes future work that is planned. Section 7 concludes the paper.

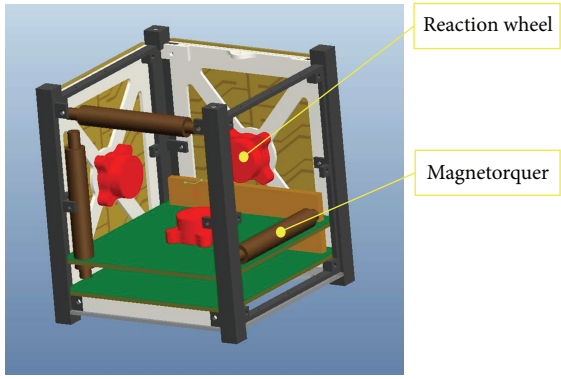


Figure 1: ACS (CubeSat-class nanosatellite with three reaction wheels and three torque rods).

2. CubeSat ACS Hardware

2.1. Attitude Sensors. Attitude magnetic sensor hardware in the current study consists of a Honeywell HMC5883L three-axis MEMS magnetometer for magnetic field measurements. Angular rate information is obtained in three axes from three orthogonally mounted Analog Devices ADXRS614 MEMS gyroscopes. The attitude control system is managed by an AT91SAM9260 32-bit ARM9 microcontroller that runs embedded Linux, with 32 MB SRAM, and 256 MB NAND Flash attached for volatile and nonvolatile storage. All programming of control algorithms is accomplished in the C language using the GNU C compiler for the ARM processor. The system is designed for power-efficient operation because there is typically less than 3 W generated from the photovoltaics on a typical 1U CubeSat in low-earth orbit, and a battery must be used during eclipse periods.

2.2. Magnetorquer Design. Magnetic torque coils, also referred to as magnetorquers, in CubeSat-class nanosatellites provide baseline control in many small satellites. They are commercially available in two typical configurations: in loose coils of flat-wound wire and in tightly wound coils around a permalloy rod. The rod configuration is often preferred because of its compactness and rigidity and the use of high-permeability, μ, materials for the core. To meet the mass and power requirements of a nanosatellite, a maximum mass, M, of 30 g and a conservative maximum power draw, P, of 0.2 W were set, and a typical maximum supply voltage, V, between 3.7 V and 4.2 V was assumed to avoid having to step up voltage on power components. After combining the power and mass equations (1), where R is resistance, W_w is the resistivity of the wire, and ρ is the density, and solving for the core radius, r_c, and length, l_c,

$$\begin{aligned} R &= \frac{V^2}{P}, \\ l_w &= \frac{R}{W_w}, \\ M &= \pi r_w^2 l_w \rho_w + \pi r_c^2 l_c \rho_c. \end{aligned} \tag{1}$$

The power and mass constraints were applied using (1), and a power value was estimated as the lesser of 0.2 W and the power dissipation was achieved at the maximum current for the wire

$$\begin{aligned} r_c^2 l_c &= \frac{M}{\pi \rho_c} - \frac{\rho_w}{\rho_c} \frac{r_w V^2}{P W_w}, \\ N_d &= \frac{4\left[\ln\left(l_c/r_c\right) - 1\right]}{\left(l_c/r_c\right)^2 - 4\ln\left(l_c/r_c\right)}, \\ D &= \frac{r_c V}{2W_w}\left[1 + \frac{\mu_r - 1}{1 + \left(\mu_r - 1\right) N_d}\right]. \end{aligned} \tag{2}$$

Using (2) from the well-known solenoid equation and the relations derived in [12], where N_d is the demagnetizing factor, a parametric analysis of the effect of core length and core radius on magnetic dipole moment, D, was used to determine the optimal length and radius of the core material, given the wire thickness and corresponding length to satisfy the power and dipole moment requirements. Figure 3 illustrates the effect of core sizing on generated magnetic moment. The number of turns is implicitly determined, as the dipole moment of the torque rod is independent of the number of turns of wire, and the core radius is also constrained to sizes that are commercially available.

To maximize the field generated with the dimensions while satisfying the design constraints, a prototype magnetorquer was designed with 70 mm long permalloy core and 36 AWG wire to provide a maximum load power of 200 mW at 4.2 V. The design parameters of the constructed torque rods are shown in Table 1.

In order to precisely wind 36 AWG wires around a core, a coil winding machine was designed and assembled using stepper motors and L298 H-bridge drivers controlled by an ATMega644P microcontroller. A permalloy core is rotated by one motor, while another positions a plastic feeder guide from the wire spool. The winder, shown with a completed torque rod in Figure 4, allows a torque rod to be automatically wound by setting the number of turns required, length of the core, and thickness of the wire, which determines the ratio of winding speed to feeder speed.

2.3. Reaction Wheel Design. Pointing and slew maneuvering of satellites are often accomplished by a motorized rotating mass such as a reaction wheel and momentum wheel, which provide maneuvering torque and momentum storage [8]. Reaction wheels can provide a high degree of attitude control accuracy with the limitation that the wheel may reach saturation after continued use, requiring an additional momentum control method such as magnetorquers to desaturate the wheel in a process known as momentum dumping.

Each of the reaction wheels in the proposed ACS system consists of a steel cylinder that is press-fitted to the shaft of a Faulhaber brushless flat micromotor. Design choices for the motor were limited to inexpensive commercial motors with low-power consumption, and the reaction wheels were sized to provide maximum momentum storage given the mass and volume constraints of a 1U CubeSat. Three reaction wheels

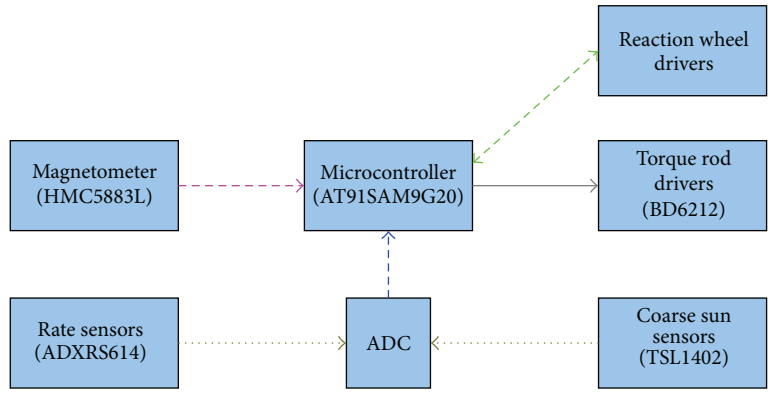


Figure 2: Overview of proposed ACS design for a CubeSat-class nanosatellite.

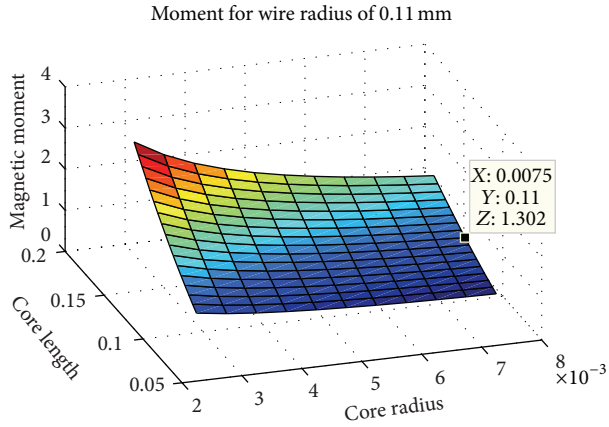


Figure 3: Magnetorquer sizing surface.

Table 1: Magnetorquer parameters.

Parameter	Value	Unit
Maximum dipole moment	0.37	A m^2
Total mass	28	g
Number of turns	6063	
Core diameter	5.7	mm
Wire diameter	0.127	mm
Wire resistance	121	Ω
Maximum current	34.7	mA

can be used in the ACS if maximum control authority is required. Table 2 shows the design parameters of the reaction wheels used on the proposed ACS [13]. A completed reaction wheel assembly is shown in Figure 5.

2.4. Electronic Integration of ACS Components. To control the reaction wheel and magnetorquers hardware and house the attitude sensors and actuator drivers, a printed circuit board (PCB) was fabricated, shown in Figure 6. The board stacks with existing PC/104 sized on-board computer (OBC) hardware and provides both I/O breakout and power supplies for the ACS hardware. It contains 3.3 V and 5 V switching supplies for the ACS sensors, as well as external interfaces for a battery and radio to be used specifically for air-bearing

Table 2: Reaction wheel parameters.

Parameter	Value	Unit
Rotor mass	0.214	kg
Moment of inertia (axial)	9.41×10^{-5}	Kg m^2
Moment of inertia (transverse)	5.02×10^{-5}	kg m^2
Motor shaft Torque	6.0×10^{-4}	N m
Maximum speed	1539	rad/s
Supply voltage	3.7–4.2	V

ACS testing. The board makes a HMC5883 three-axis magnetometer, an ADXL345 3-axis accelerometer, and an ITG-3200 3-axis MEMS rate gyroscope available on the OBC I^{2}C bus. Primary rate sensing for attitude control is accomplished by three independent ADXRS614 rate gyro units oriented on orthogonal axes by means of right-angle IC sockets and connected to the first three ADC channels on the OBC. This allows accurate high-speed sampling of rotation rates for use by the attitude controller. To drive the magnetorquers, three BD6212 integrated H-bridges are used, controlled by three PWM channels from the OBC and three general purpose IO pins for current direction control. To allow one PWM signal and one direction pin to control each H-bridge, the inputs are demultiplexed by a SN74LVC1G8 tristate output demultiplexer and pull-up resistors. In full operation, the board draws up to 100 mW of power on average, though components can be shut down as needed to conserve power if not in use.

3. System Models

3.1. Attitude Equations of Motions. In this section, the satellite is modelled as a rigid body with actuators that provide torques about three mutually perpendicular axes that defines a body-fixed frame. The equations of motion [14, 15] are given by

$$\widehat{J}\dot{\omega}_b = -\omega^{\times}\left(J_S\omega_b + A_i J_w \Omega\right) + A_i\tau_c + \tau_m + \tau_d, \qquad (3)$$

where $\omega_b = (\omega_{b1}\omega_{b2}\omega_{b3})^T$ is the angular velocity of the satellite expressed in the body frame. J_s is the inertia matrix of the satellite. J_w is the inertia matrix of the reaction wheel and $\widehat{J} = J_s - A_i J_w A_i^T$. A_i is the layout matrix of the reaction

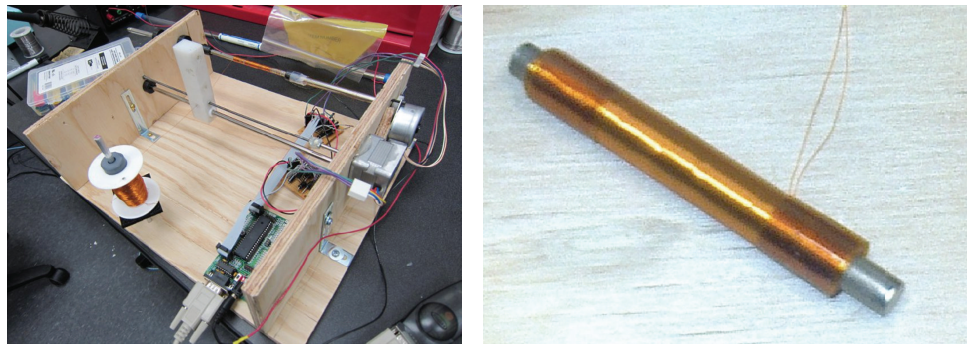


Figure 4: Magnetorquer winding apparatus and completed torque rod.

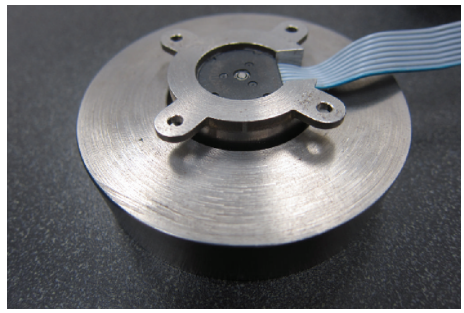


Figure 5: Reaction wheel assembly.

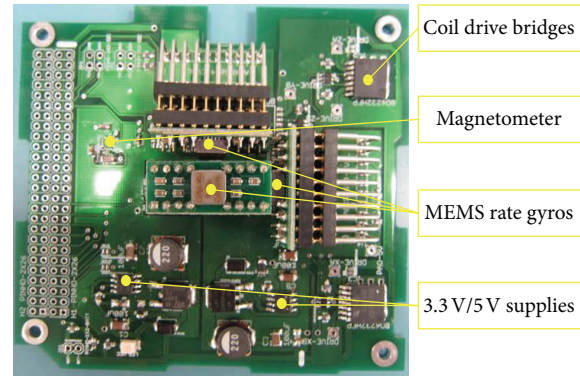


Figure 6: ACS sensor and actuator board.

wheels whose columns represent the influence of each wheel on the angular acceleration of the satellite. Ω is the velocity of a reaction wheel, τ_c is the torque control provided by the reaction wheel, τ_m is the torque control provided by the magnetorquers, and τ_d is the bounded external disturbance, which is a sum of the gravity gradient τ_{gravity}, aerodynamic τ_{aero}, and solar radiation pressure τ_{solar} disturbances.

The gravity gradient disturbance is $\tau_{\text{gravity}} = 3\sqrt{\mu_e/a^3}^2 CJ_sC$, where μ is the gravitational parameter of the Earth, a is the semimajor axis of the orbit, and C_k is the direction cosine matrix in terms of quaternions.

The aerodynamic disturbance is $\tau_{\text{aero}} = C_d/2\rho v^2 AL$, where C_d is the coefficient of drag for a flat plate, A is the cross-sectional area causing aerodynamic drag, v is the satellite velocity, L is the distance between the centre of pressure and the centre of gravity, and ρ is the atmospheric density related to the altitude.

The solar radiation pressure disturbance is $\tau_{\text{solar}} = F_s/cA_s(1+r)L$, where F_s is the solar constant at the Earth's orbital distance from the Sun, c is the speed of light, A_s is the illuminated surface area, and r is the surface reflectance.

3.2. Attitude Kinematics. The satellite attitude kinematics is represented using quaternions

$$\dot{q} = \frac{1}{2}\begin{pmatrix} q_4 I_{3\times 3} + \overline{q}^{\times} \\ -\overline{q}^T \end{pmatrix} \omega_b \equiv \frac{1}{2} A(q)\,\omega_b, \tag{4}$$

where $q = (\overline{q}^T, q_4)^T = (q_1, q_2, q_3, q_4)^T$.

In terms of Euler angles, we can also express the satellite attitude as

$$\begin{bmatrix} \dot{\psi} \\ \dot{\alpha} \\ \dot{\gamma} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\psi)\tan(\alpha) & \cos(\psi)\tan(\alpha) \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin\left(\dfrac{\psi}{\cos(\alpha)}\right) & \dfrac{\cos(\psi)}{\cos(\alpha)} \end{bmatrix} \omega_b, \tag{5}$$

where ψ is the roll angle about the x-axis, α is the pitch angle about the y-axis, and γ is the yaw about the z-axis.

3.3. Sensor Models. Magnetic field vectors are obtained in the orbit reference frame

$$\begin{aligned} B_1 &= \frac{M_e}{r_0^3} \\ &\quad \times \left[\cos(\omega_0 t)\left(\cos(\epsilon)\sin(i) - \sin(\epsilon)\cos(i)\cos(\omega_e t)\right)\right. \\ &\quad \left. - \sin(\omega_0 t)\sin(\epsilon)\sin(\omega_e t)\right], \\ B_2 &= \frac{-M_e}{r_0^3}\left[\left(\cos(\epsilon)\cos(i) + \sin(\epsilon)\sin(i)\cos(\omega_e t)\right)\right], \\ B_3 &= \frac{3M_e}{r_0^3} \\ &\quad \times \left[\sin(\omega_0 t)\left(\cos(\epsilon)\sin(i) - \sin(\epsilon)\cos(i)\cos(\omega_e t)\right)\right. \\ &\quad \left. -2\sin(\omega_0 t)\sin(\epsilon)\sin(\omega_e t)\right], \end{aligned} \tag{6}$$

where ω_0 is the angular velocity of the orbit with respect to the inertial frame, r_0 is the distance from the center of the satellite to the center of the Earth, i is the orbit inclination, ε is the magnetic dipole tilt, ω_e is the spin rate of the Earth, and M_e is the magnetic dipole moment of the Earth. The magnetometer model is

$$H = C_k \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix} + \eta_m + b_m, \tag{7}$$

where C_k is the direction cosine matrix in terms of quaternions, η_m is the zero mean Gaussian white noise of the magnetometer, B is the vector formed with the components of the Earth's magnetic field in the body frame of the reference, and $B = C_k \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix}$. b_m is the magnetometer bias.

The angular velocity is measured from three rate gyroscopes. The model is given by

$$\begin{aligned} \omega_g &= \omega + b_g + \eta_g, \\ \dot{b}_g &= -k_f b_g + \eta_f, \end{aligned} \tag{8}$$

where ω_g is the output of a gyroscope and ω is the real angular rate of the gyro. η_g and η_f are Gaussian white noise. b_g is the random drift of the gyro and k_f is the drift constant.

3.4. Actuator Models. Reaction wheels are widely used to perform precise satellite attitude maneuvers because they allow continuous and smooth control of internal torques. Torques are produced on the satellite by accelerating or decelerating the reaction wheels. Let the torque demanded by the satellite be denoted as τ_c, where $\tau_c = J_w(\dot{\Omega} + A_i \dot{\omega} b)$. The input voltage e_a required to control the actuator dynamics of the reaction wheel can be written as

$$e_a = k_b \Omega - R_b k_t^{-1} \left(A_i' \tau_c \right), \tag{9}$$

where k_t is the motor torque constant, k_b is the back-EMF constant, R_b is the armature resistance, and friction in the reaction wheels is ignored. The maximum voltage of the reaction wheel is 4.2 V, and a dead zone for the reaction wheel is estimated to be below 1 V. k_t is 0.0082, k_b is 0.007, R_b is 0.5, and the moment of inertia of the reaction wheel is 0.0001 kg m^2.

4. Control Law Design and Simulation Results

4.1. Satellite Attitude Control Laws. Magnetic control has been used over many years [16, 17] for small spacecraft attitude control. The main drawback of magnetic control is that magnetic torque is two-dimensional and it is only present in the plane perpendicular to the magnetic field vector [18]. The accuracy of satellite attitude control systems (ACS) using only magnetic actuators is known to be accurate on the order of 0.4–0.5 degree [18]. The satellite cannot be controlled precisely in three-dimensional space using only magnetorquers [18], but the combination of magnetorquers with one reaction wheel expands the two-dimensional control torque possibilities to be three-dimensional. The attitude accuracy of the combined actuators has been compared with three reaction wheels-based attitude control in the references [18, 19]. Classical sliding mode control has also been used for magnetic actuated spacecraft [20, 21]. However, the proposed nonlinear adaptive fuzzy sliding mode control law has never been used in magnetic attitude control.

To address the attitude tracking problem, the attitude tracking error $q_e = (\overline{q}_e^T, q_{4e})^T$ is defined as the relative orientation between the body frame and the desired frame with orientation $q_d = (\overline{q}_d^T, q_{4d})^T$. In order to apply the proposed nonlinear controller, the equations of motion are rewritten as

$$\ddot{q}_e = f\left(\overline{q}_e, \dot{\overline{q}}_e\right) + \hat{\tau}_c + \hat{\tau}_m + \hat{\tau}_d. \tag{10}$$

The adaptive fuzzy sliding mode magnetic control law is given by

$$\tau = -k_1 S - \theta^T \xi - k_2 \tanh\left(\frac{3K_u \varsigma S}{\epsilon}\right), \tag{11}$$

$$\dot{\theta} = \delta S \xi \theta, \tag{12}$$

$$\tau_{\text{ap}} = \frac{\tau S}{\left\| S^2 \right\| S}, \tag{13}$$

$$M = \tau_{\text{ap}} \times \frac{B}{\left\| B^2 \right\|}. \tag{14}$$

Here, $\hat{\tau}_m = M \times B$ are the torques generated by the magnetorquers, M is the vector of magnetic dipoles for the three magnetorquers, and B is the vector formed with the components of the Earth's magnetic field in the body frame of the reference. $S = \dot{\overline{q}}_e + K\overline{q}_e$ is the sliding surface. θ and ξ are the adaptive parameters and fuzzy weight functions generated by the fuzzy logic systems [22], and δ, k_1, k_2, K_u, ς, ϵ are positive constants used for tuning the control response.

Remark 1. AFSMC controller design details can be found in the author's previous papers [11]. The design includes: (1) Sliding surface design [22] and (2) fuzzy logic system design [22].

4.2. Simulation Results. The attitude detumbling and attitude stabilization phases are considered in the ACS simulation. The B-dot algorithm, PD magnetic control law, and adaptive fuzzy sliding mode magnetic control law are used for the two phases, respectively. We note that the orbit used for the present simulation study is a 500 km circular orbit with 45° inclination. At this altitude, the total disturbance torque for 1U CubeSats is estimated to be on the order of 5×10^{-7} Nm. This is intended to be a slight overestimation to include a safety margin.

4.2.1. Detumbling Mode and Stabilization Mode

Scenario 1. In the initial stage of ACS control, the angular velocities of the satellite are assumed to be 0.169 rad/s as a result of separation from the launch vehicle. The ACS damps the angular rate by controlling three magnetorquers. The control logic generally used for detumbling is called B-dot

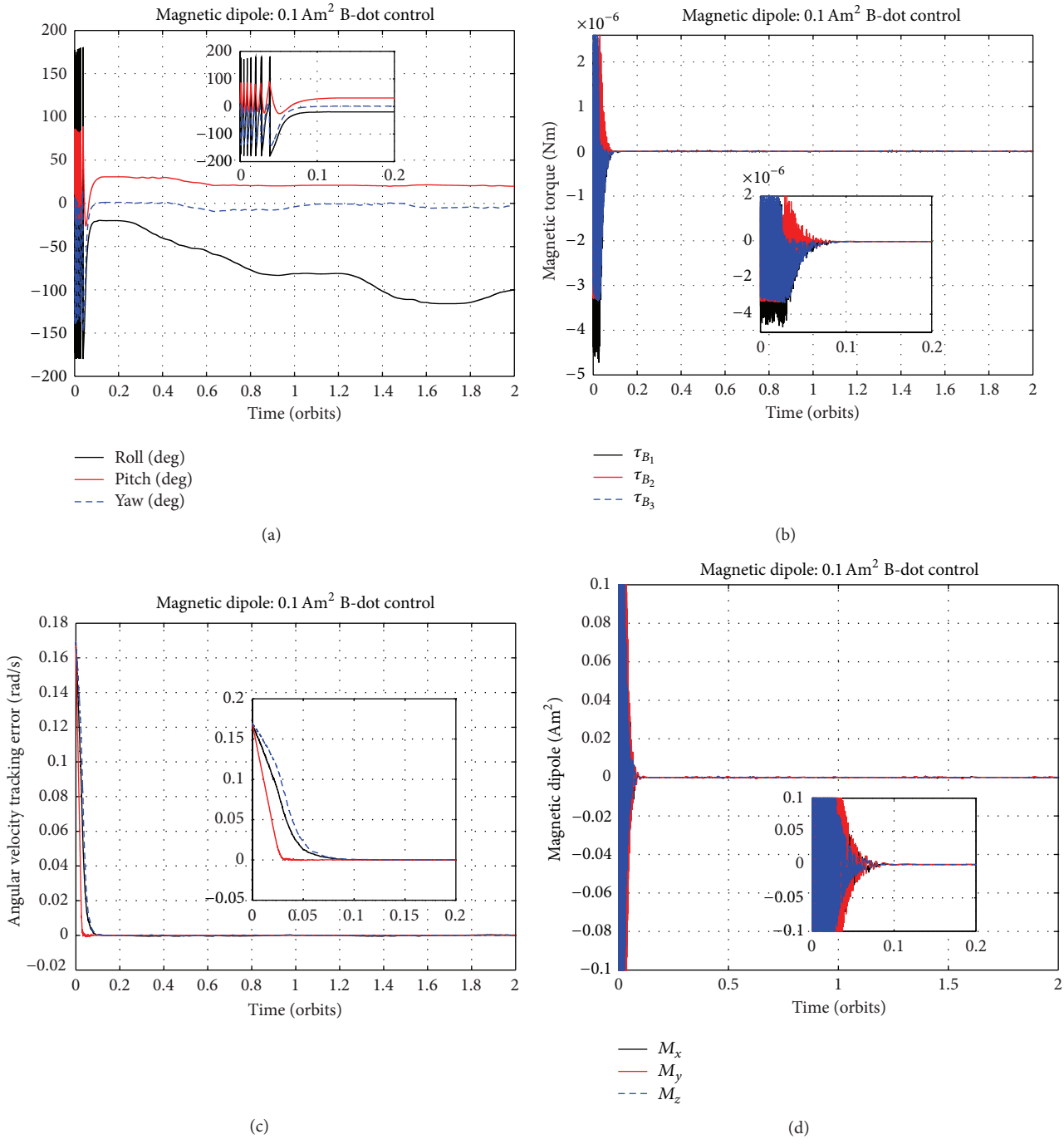


Figure 7: Scenario 1 detumbling control results.

control [1], as it makes use of the derivative of the magnetic field "B". For a CubeSat with moment of inertia $J = \text{diag}(0.002, 0.002, 0.002)\,\text{kg m}^2$, we include the external disturbances (aerodynamic, gravity gradient, and solar pressure), set the desired quaternion to be $(0, 0, 0, 1)$, set the initial quaternion to be $(0.1, -0.1, 0.1, 0.9849)$, and assume the magnetic dipole maximum of the rods to be $0.1\,\text{Am}^2$. The Euler angle tracking errors, angular velocity tracking errors, and magnetic torquers, magnetic dipoles results are shown in Figure 7. The satellite starts at the selected tip-off rate, and, after 1 orbit, the angular velocities are reduced to the required rates before continuing with other ACS tasks.

Scenario 2. Now, we consider a CubeSat with the same moment of inertia and orbit and assume the magnetic dipole maximum of the magnetorquer to be $0.4\,\text{Am}^2$ with a magnetometer sensor bias calculated by $20 * 10^{-4} * \text{rand}(1)$. Proportional-derivative (PD) magnetic control [23] laws (shown in (15)) are used in this simulation and the results over 6 orbits are shown in Figure 8. During the first three orbits, three

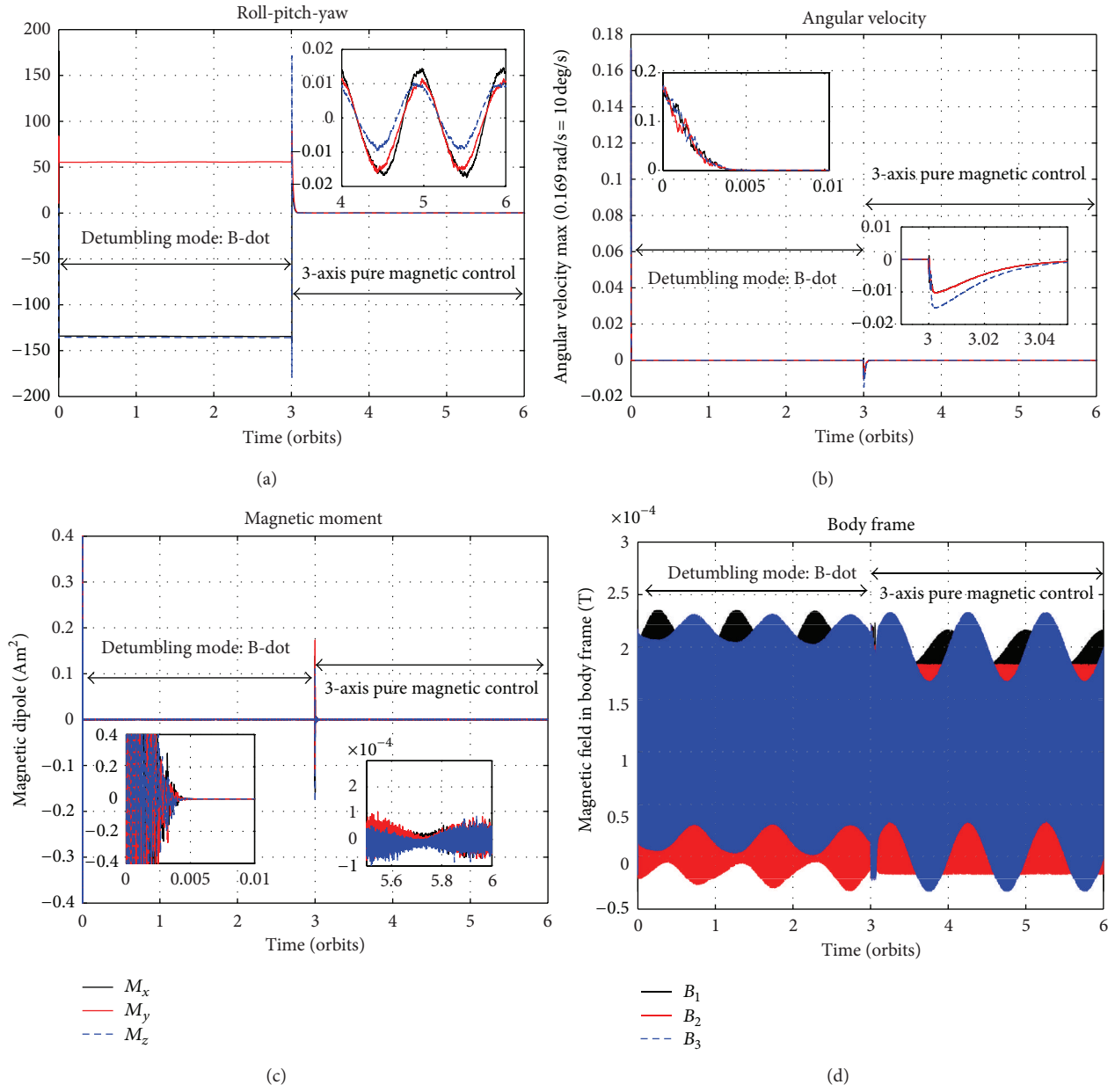


FIGURE 8: Scenario 2 detumbling mode and stable mode control results.

magnetorquers are used for the detumbling mode. In the second three orbits, three magnetorquers and one reaction wheel are used for the stable mode. The attitude control accuracy is less than 0.02 degree while using the PD magnetic control laws

$$\begin{aligned} \widehat{\tau}_m &= M \times B, \\ M &= K_1 \omega_b \times B, \\ M &= K_1 \omega_b \times B + K_2 q \times B. \end{aligned} \tag{15}$$

4.2.2. Attitude Stabilization Mode: Nadir and Limb Pointing with Three Magnetorquers and One Pitch Reaction Wheel. Next, we consider the second stage of nanosatellite control with a low initial angular velocity after detumbling and large slew angle target for limb and nadir pointing. The configuration with three magnetorquers and one flywheel [24] has been used for many years. In a real nanosatellite mission, hardware failures of the reaction wheels are very common [19]. When there are two wheel failures, the ACS can be switched from using three reaction wheels to three magnetorquers and one reaction wheel, and attitude control accuracy maintained using this method. The adaptive fuzzy sliding mode magnetic control laws are shown in (11)–(14). The attitude control accuracy using the nonlinear adaptive fuzzy sliding mode control law will be more robust to the external disturbances than using the PD magnetic control law in (15).

Scenario 3. An ACS using one reaction wheel and three magnetorquers as actuators for limb pointing with the desired

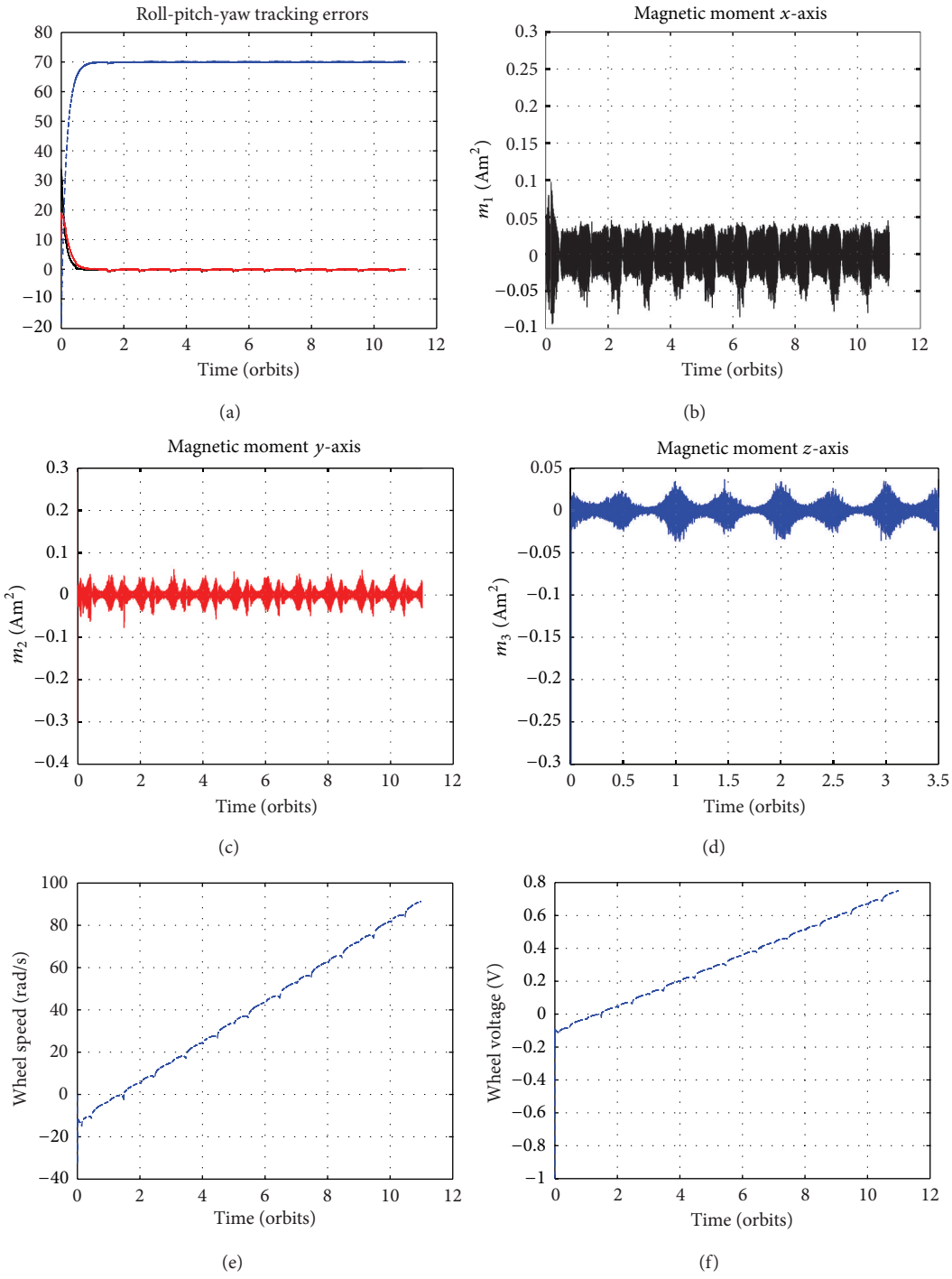


FIGURE 9: Scenario 3 limb pointing control results using one reaction wheel and three magnetorquers.

quaternion set to $(0, 0.5736, 0, 0.8192)$ is examined using AFSMC over 10 orbits. The wheel dead zone is not considered here. The initial quaternion is $(0.1, -0.1, 0.1, 0.9849)$ and initial angular velocity is $(0.0169, 0.0169, 0.0169)$ rad/s. Considering only the pitch reaction wheel is available, the numerical simulations demonstrate that the proposed technique achieves a high pointing accuracy (<0.09 degree) for small satellites. The magnetic dipoles from the three magnetorquers, attitude tracking errors, wheel speed, and wheel voltage are shown in Figure 9.

Scenario 4. An ACS using one reaction wheel and three magnetorquers as actuators for nadir pointing with the desired quaternion that is set to $(0, 0, 0, 1)$ is examined using AFSMC over 10 orbits. The wheel dead zone is assumed to be ± 1.0 V. The initial quaternion is $(0.1, -0.1, 0.1, 0.9849)$ and initial

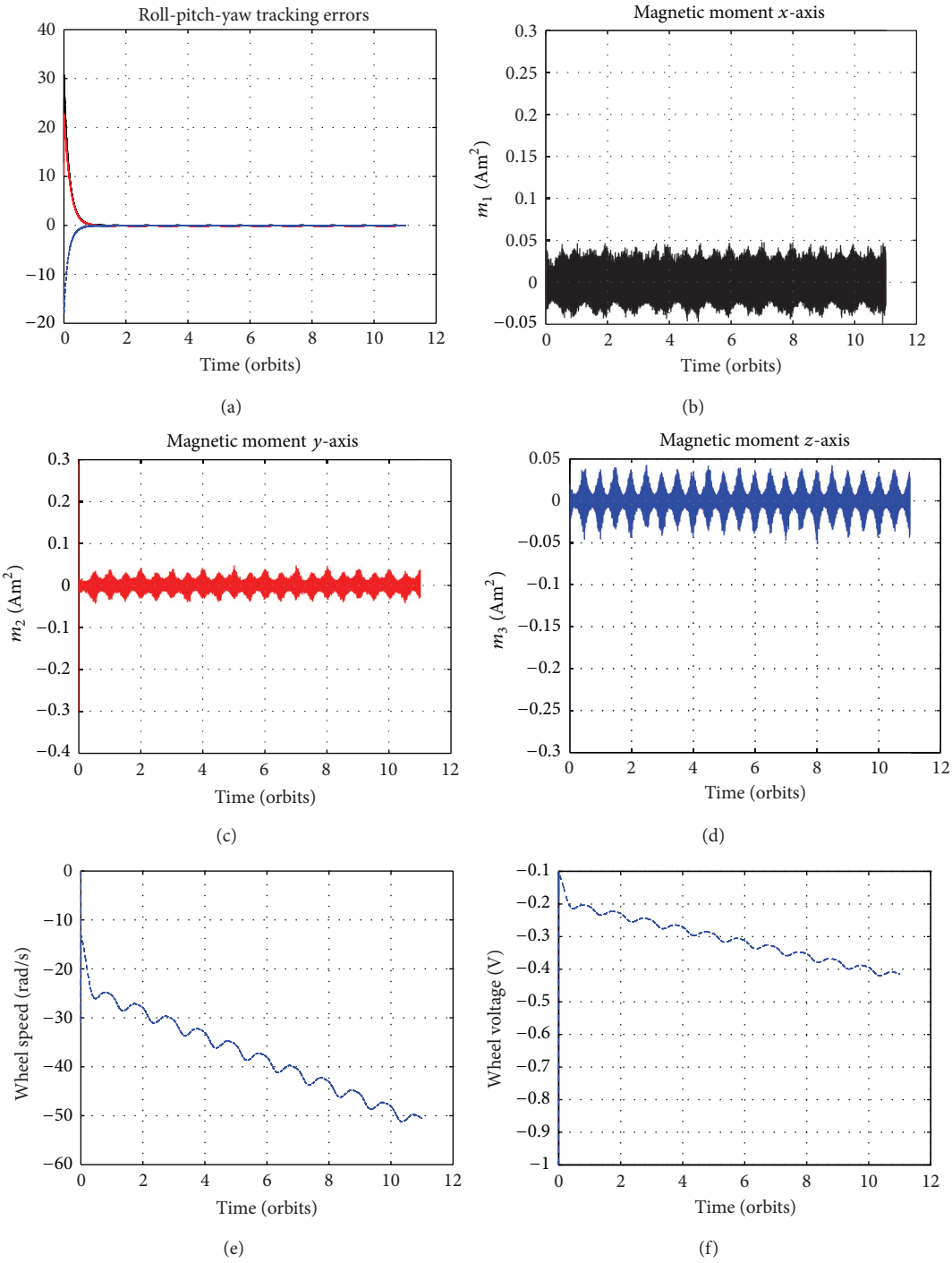


FIGURE 10: Scenario 4 nadir pointing control results using one reaction wheel and three magnetorquers.

angular velocity is (0.0169, 0.0169, 0.0169) rad/s. The pitch reaction wheel and three magnetorquers are used. These simulations demonstrate that the proposed technique can also achieve high accuracy (<0.09 degree with all disturbances) pointing control for small satellites. The magnetic dipoles of the three magnetorquers, attitude tracking errors, wheel speed, and wheel voltage are shown in Figure 10. It takes about 40 orbits for the wheel speed to increase from 0 to 2000 rpm (209 rad/s).

4.2.3. Attitude Stabilization Mode: Nadir Pointing with Three Reaction Wheels

Scenario 5. An ACS using three reaction wheels as actuators for nadir pointing with the desired quaternion set to (0, 0, 0, 1) is examined using AFSMC over 0.1 orbits. The wheel dead zone is assumed to be ±1.0 V. The initial quaternion is (0.1, −0.1, 0.1, 0.9849) and initial angular velocity is (0.0169, 0.0169, 0.0169) rad/s. This configuration can also

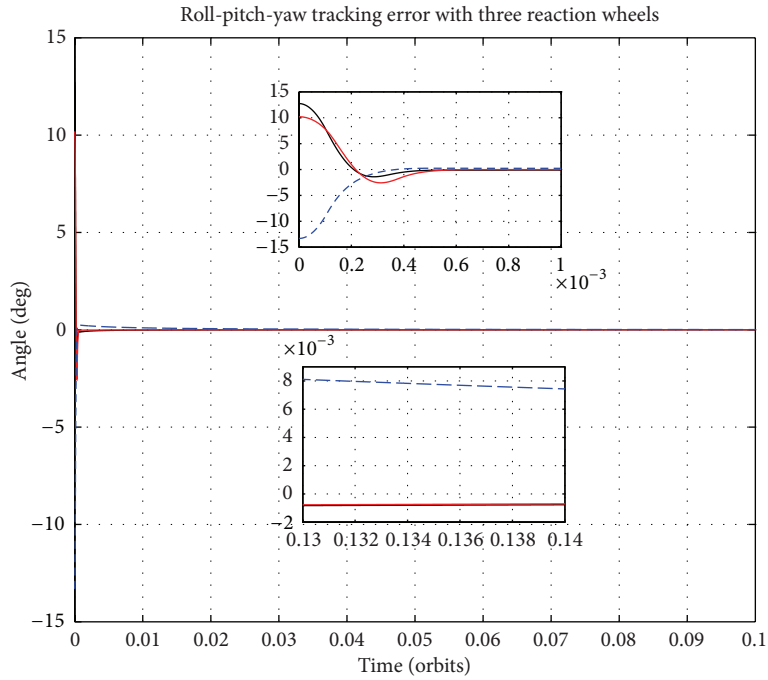


Figure 11: Scenario 5 nadir pointing control results using three wheels: Euler angle tracking errors.

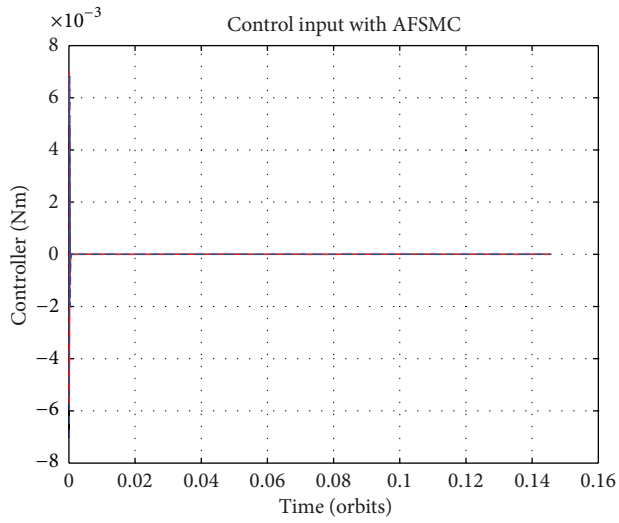


Figure 12: Scenario 5 nadir pointing control results using three wheels: control input.

achieve high attitude control accuracy (<0.008 degree with all disturbances). The reaction wheel attitude control laws use (11) and (12). The settling time is shorter and the pointing accuracy is higher than that achieved using only three magnetorquers and one reaction wheel as actuators. However, the power consumption is higher than using three magnetorquers and one reaction wheel. The Euler angle tracking errors, control inputs, wheel speeds, and wheel voltages are shown in Figures 11, 12, 13, and 14.

5. Satellite Attitude Control System Hardware Testing

5.1. Spherical Air-Bearing Testbed for Satellite Attitude Control Systems. The ground testing of the proposed CubeSat ACS design was performed at York University using a nanosatellite attitude control testbed. This facility consists of a spherical air-bearing platform [25, 26] suspended upon a thin layer of air providing a full three degrees of freedom with negligible friction for ACS testing. The platform includes a manual balancing system and platform electronics that include an on-board computer (OBC), wireless transceiver for telemetry, reference inertial measurement unit (IMU), power distribution board, and batteries. The air-bearing system used for 1U CubeSat testing is shown in Figure 15.

5.2. Ground Test Results: ACS Testing Results with Three Reaction Wheel Actuators. Nonlinear attitude control has been explored widely in theory [14, 27, 28]. In real satellite applications, nonlinear controllers are usually not selected due to their complexity of design. We have tested nonlinear control algorithms [11] with our spherical air-bearing system. We now test the proposed control method (from (11) and (12)) on this system using the sensors and ACS board described above and three-axis reaction wheel actuation. A PID control law is also used for comparison on the same hardware. The control laws are programmed in the C language, and the OBC runs the Linux operating system. More details of the implementation can be also found in [11]. Here, we will show air-bearing system test results with a 1U CubeSat. The design parameters of the control laws used are given in Tables 3 and 4.

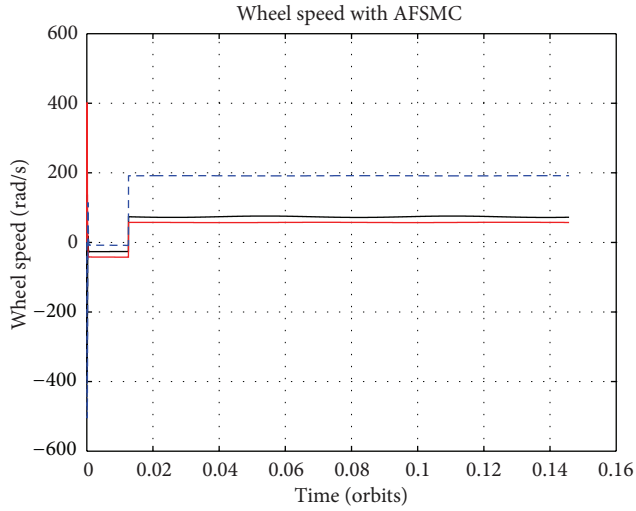


Figure 13: Scenario 5 nadir pointing control results using three wheels: wheel speed.

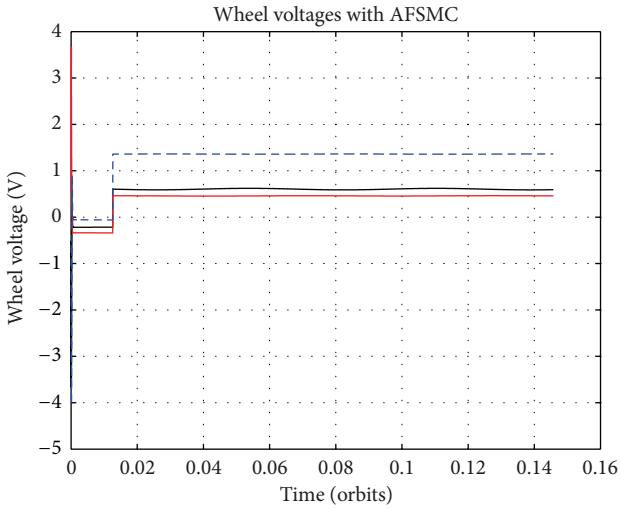


Figure 14: Scenario 5 nadir pointing control results using three wheels: wheel voltage.

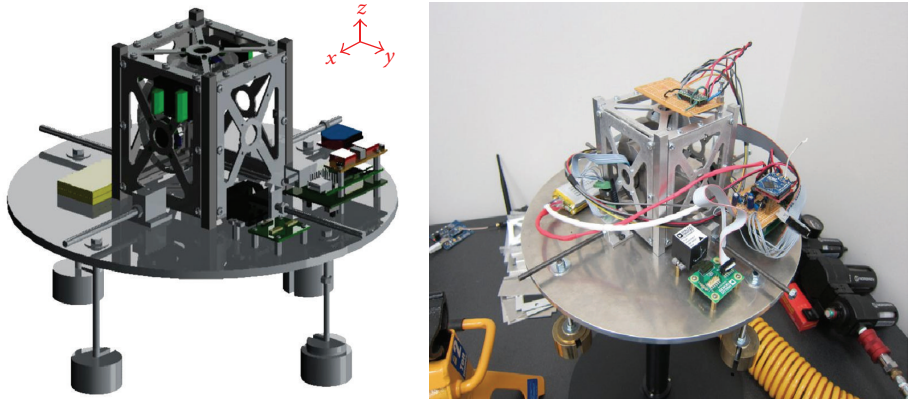


Figure 15: Air-bearing ACS ground testing system.

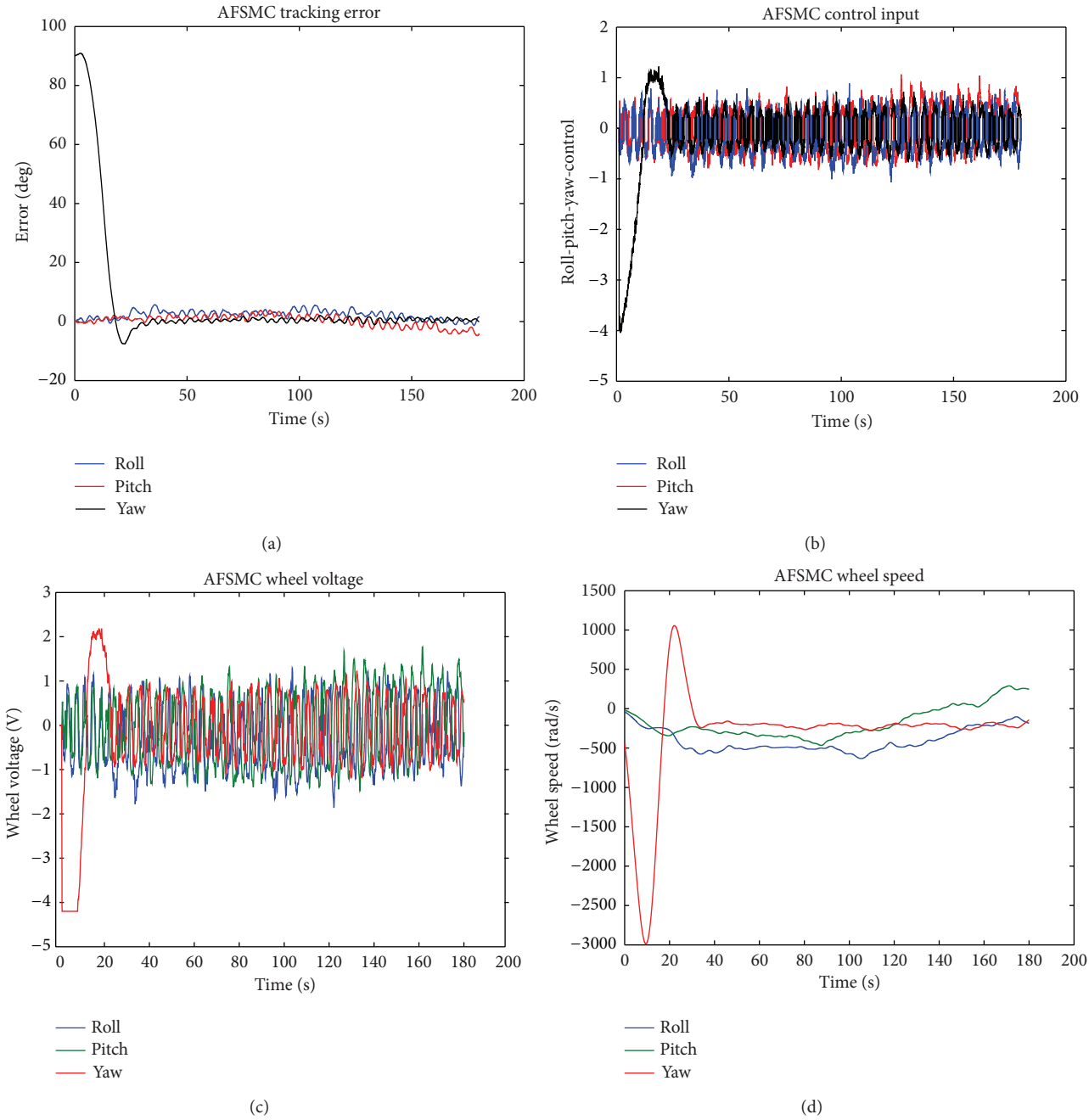


Figure 16: Air-bearing AFSMC controller results.

Table 3: PID controller parameters for air bearing testing.

Name	Values
Proportional parameters	0.03
Integral parameters	0.0001
Derivative parameters	0.11

Table 4: AFSMC controller parameters for air bearing testing.

Name	Values
Sliding surface gain	0.00001, 100
Fuzzy membership function	1, 1
Adaptive gains δ, K_u, ς, ϵ	0.1, 0.36, 0.01, 2
AFSMC parameter k_1	0.004
AFSMC parameter k_2	0.0025

Figures 16 and 17 show the attitude tracking errors, AFSMC/PID control output signals, reaction wheel voltages, and reaction wheel velocities for a 90 degree yaw slew of the system about the z-axis while maintaining 0 degrees of roll and pitch. Due to the difficulty of perfectly balancing the air-bearing system, a gravitational disturbance, larger than

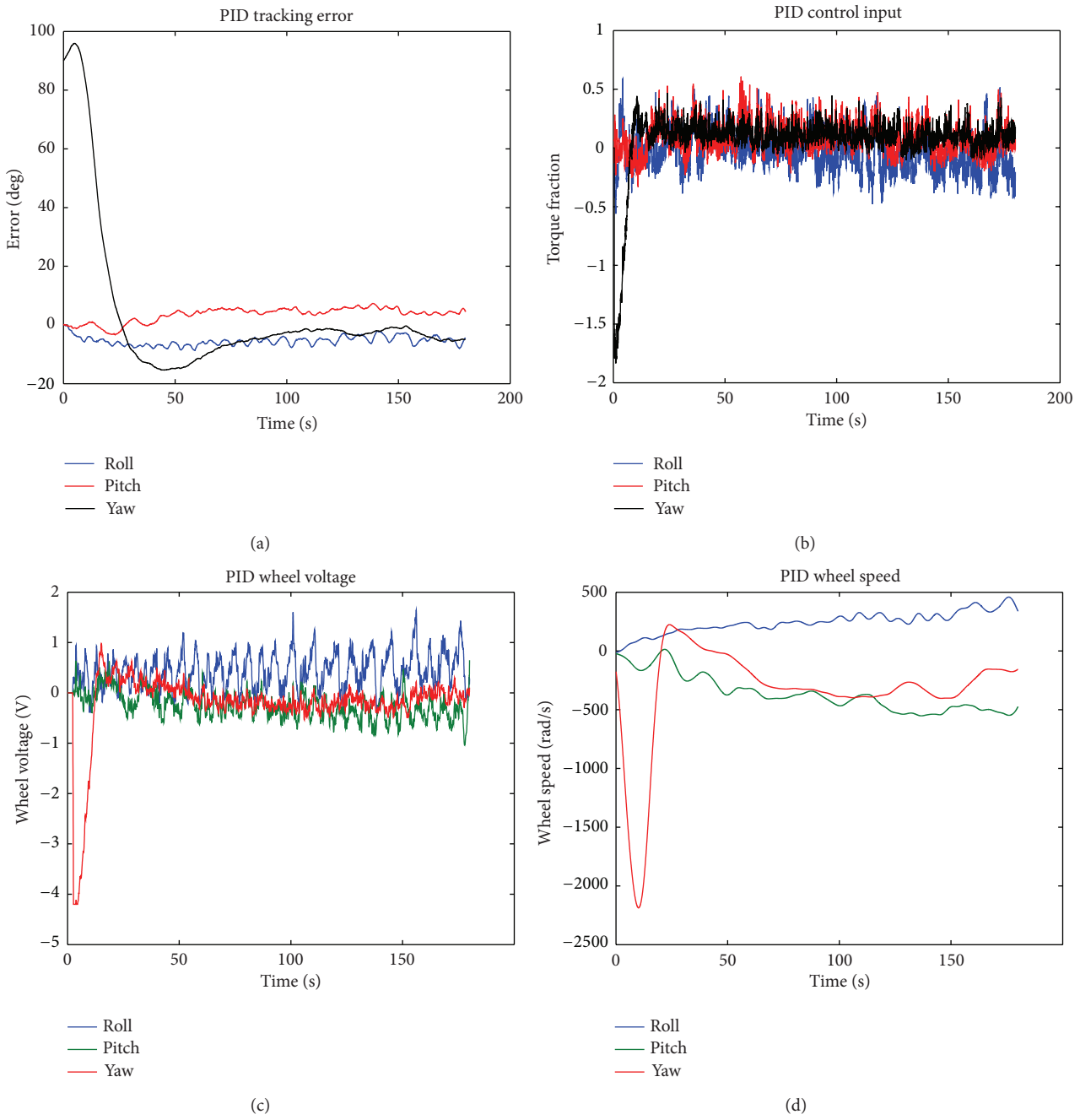


Figure 17: Air-bearing PID controller results.

normal in size, is considered to be present about the x and y rotational axes. Compared to the PID controller, the AFSMC controller uses nearly the same gain and has much better tracking performance under these conditions.

6. Future Work

While the ground test demonstrates the effective and promising control accuracy of the proposed ACS design, it is difficult to predict the performance of pure and hybrid magnetic control as the Earth's magnetic field present in the test facility is constant and invariant with many sources of additional magnetic noise. The next step in the development of a complete ACS test facility is the addition of a Helmholtz cage for magnetic control tests. The magnetic cage and the air-bearing system in the lab are shown in Figure 18 and are presently being prepared for air-bearing magnetic control testing. Future work will demonstrate the effectiveness of magnetic control in further ground testing.

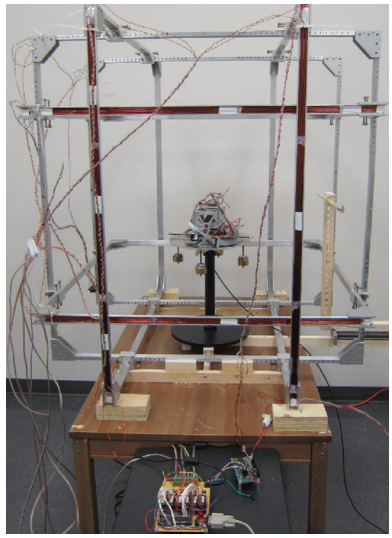


Figure 18: Helmholtz cage for magnetic field generation.

7. Conclusion

In this paper, the proposed attitude control system design will allow inexpensive and capable satellites to be developed for academic use. The proposed attitude control system uses three reaction wheels as actuators or one wheel with three magnetorquers as actuators. The hardware designs of the actuators and embedded attitude control systems are also described for prototype development. Five different scenarios of numerical simulation results for a 1U CubeSat-class nanosatellite show the effectiveness of the proposed attitude control design for detumbling and attitude pointing purposes. The ground testing results in this paper provide a promising comparison of an adaptive fuzzy sliding mode controller with a conventional proportional-integral-derivative controller for a 1U CubeSat-class nanosatellite. The nanosatellite testing hardware can be also extended for use in 2U or 3U CubeSat-class nanosatellites in the future.

Acknowledgments

The authors gratefully acknowledge the support provided by COM DEV Ltd., NSERC, MITACS, and OCE. The authors would also like to acknowledge the work of M. Cannata, I. Proper, T. Ustrzycki, G. Benari, and H. Hakima in this paper.

References

[1] Y. W. Jan and J. C. Chiou, "Attitude control system for ROCSAT-3 microsatellite: a conceptual design," *Acta Astronautica*, vol. 56, no. 4, pp. 439–452, 2005.

[2] M. Ovchinnikov, V. Pen'kov, O. Norberg, and S. Barabash, "Attitude control system for the first Swedish nanosatellite 'MUNIN'," *Acta Astronautica*, vol. 46, no. 2, pp. 319–326, 2000.

[3] M. I. Martinelli and R. S. S. Peña, "Passive 3 axis attitude control of MSU-1 pico-satellite," *Acta Astronautica*, vol. 56, no. 5, pp. 507–517, 2005.

[4] G. P. Candini, F. Piergentili, and F. Santoni, "Miniaturized attitude control system for nanosatellites," *Acta Astronautica*, vol. 81, pp. 325–334, 2005.

[5] T. Xiang, T. Meng, H. Wang, K. Han, and Z. H. Jin, "Design and on-orbit performance of the attitude determination and control system for the ZDPS-1A pico-satellite," *Acta Astronautica*, vol. 77, pp. 82–196, 2012.

[6] M. Abdekrahman and S. Y. Park, "Integrated attitude determination and control system via magnetic measurements and actuation," *Acta Astronautica*, vol. 69, no. 3-4, pp. 168–185, 2011.

[7] M. Cannata, *Development of a sun vector determination algorithm for Cubesat-Class spacecraft [M.S. thesis]*, Dept. of Earth and Space Science and Engineering, York University, Toronto, Canada, 2010.

[8] I. Proper, *Reaction wheel design, construction and qualification testing [M.S. thesis]*, Dept. of Earth and Space Science and Engineering, York University, Toronto, Canada, 2010.

[9] J. Li, M. A. Post, and R. Lee, "Real time fault tolerant nonlinear attitude control system for nanosatellite application," in *AIAA Infotech@Aerospace 2012 Conference*, pp. 19–21, Garden Grove, Calif, USA, 2012.

[10] J. Li, M. A. Post, and R. Lee, "Nanosatellite air bearing tests of fault-tolerant sliding-mode attitude control with Unscented Kalman Filter," in *AIAA Guidance, Navigation, and Control Conference*, Minnesota, Minneapolis, Minn, USA, 2012.

[11] J. Li, M. A. Post, and R. Lee, "Nanosatellite attitude air bearing system using variable structure control," in *IEEE 25th Annual Canadian Conference on Electrical and Computer Engineering*, Montreal, Canada, May 2012.

[12] M. F. Mehrjardi and M. Mirshams, "Design and manufacturing of a research magnetic torquer Rod," *Contemporary Engineering Sciences*, vol. 3, no. 5, pp. 227–236, 2010.

[13] T. Ustrzycki, *Spherical air bearing testbed for nanosatellite attitude control development [M.S. thesis]*, Dept. of Earth and Space Science and Engineering, York University,, Toronto, Canada, 2011.

[14] C. H. Won, "Comparative study of various control methods for attitude control of a LEO satellite," *Aerospace Science and Technology*, vol. 5, pp. 323–333, 1999.

[15] J. Y. Lin, S. Ko, and C. K. Ryoo, "Fault tolerant control of satellites with four reaction wheels," *Control Engineering Practice*, vol. 16, no. 10, pp. 1250–1258, 2008.

[16] E. Silani and M. Lovera, "Magnetic spacecraft attitude control: a survey and some new results," *Control Engineering Practice*, vol. 13, no. 3, pp. 357–371, 2005.

[17] C. J. Dameran, "Hybrid magnetic attitude control gain selection," *Proceedings of the Institution of Mechanical Engineers G*, vol. 223, no. 8, pp. 1041–1047, 2009.

[18] Z. Q. Zhou, "Spacecraft attitude tracking and maneuver using combined magnetic actuators," in *AIAA Guidance, Navigation, and Control Conference*, Toronto, Ontario, Canada, August 2010.

[19] J. R. Forbes and C. J. Damaren, "Geometric approach to spacecraft attitude control using magnetic and mechanical actuation," *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 2, pp. 590–595, 2010.

[20] R. Wisniewski, *Satellite attitude control using magnetic actuation only [Ph.D. thesis]*, Dissertation, Dept. of Control Engineering, Aalborg University, Denmark, 1996.

[21] P. Wang, Y. B. Shtessel, and Y. Q. Wang, "Satellite attitude control using only magnetorquers," in *Proceeding of the 13th Southeastern Symposium on System Theory*, West Virginia University, Morgantown, WVa, USA, March 1998.

[22] J. Li and K. D. Kumar, "Fault tolerant attitude synchronization control during formation flying," *Journal of Aerospace Engineering*, vol. 24, no. 3, pp. 251–263, 2011.

[23] D. V. Guerrant, *Design and analysis of fully magnetic control for picosatellite stabilization [M.S. thesis]*, California Polytechnic State University, 2005.

[24] M. Chen, S. J. Zhang, F. R. Liu, and Y. C. Zhang, "Combined attitude control of small satellite using One flywheel and magnetic torquers," in *Proceedings of the 2nd International Symposium on Systems and Control in Aerospace and Astronautics (ISSCAA '08),* Shenzhen, China, December 2008.

[25] J. Prado, G. Bisiacchi, L. Reyes et al., "Three-axis air-bearing based platform for small satellite attitude determination and control simulation," *Journal of Applied Research and Technology*, vol. 3, no. 3, pp. 222–237, 2005.

[26] J. J. Kim and B. N. Agrawal, "Automatic mass balancing of air-bearing-based three-axis rotational spacecraft simulator," *AIAA Journal of Guidance, Control, and Dynamics*, vol. 32, no. 3, pp. 1005–1017, 2009.

[27] K. S. Kim and Y. Kim, "Robust backstepping control for slew maneuver using nonlinear tracking function," *IEEE Transactions on Control Systems Technology*, vol. 11, no. 6, pp. 822–829, 2003.

[28] S. C. Lo and Y. P. Chen, "Smooth sliding mode control for spacecraft attitude tracking maneuvers," *Journal of Guidance Control and Dynamics*, vol. 18, no. 6, pp. 1345–1349, 1995.

Hindawi Publishing Corporation
Journal of Control Science and Engineering
Volume 2013, Article ID 801237, 10 pages
http://dx.doi.org/10.1155/2013/801237

Research Article

Real-Time Energy Management Control for Hybrid Electric Powertrains

Mohamed Zaher and Sabri Cetinkunt

University of Illinois at Chicago, Chicago, IL 60607, USA

Correspondence should be addressed to Mohamed Zaher; mhzaher@asme.org

Received 19 December 2012; Accepted 9 April 2013

Academic Editor: Salem Haggag

This paper focuses on embedded control of a hybrid powertrain concepts for mobile vehicle applications. Optimal robust control approach is used to develop a real-time energy management strategy. The main idea is to store the normally wasted mechanical regenerative energy in energy storage devices for later usage. The regenerative energy recovery opportunity exists in any condition where the speed of motion is in the opposite direction to the applied force or torque. This is the case when the vehicle is braking, decelerating, the motion is driven by gravitational force, or load driven. There are three main concepts for energy storing devices in hybrid vehicles: electric, hydraulic, and mechanical (flywheel). The real-time control challenge is to balance the system power demands from the engine and the hybrid storage device, without depleting the energy storage device or stalling the engine in any work cycle. In the worst-case scenario, only the engine is used and the hybrid system is completely disabled. A rule-based control algorithm is developed and is tuned for different work cycles and could be linked to a gain scheduling algorithm. A gain scheduling algorithm identifies the cycle being performed by the work machine and its position via GPS and maps both of them to the gains.

1. Introduction

Fuel efficiency and reduced emissions are two of the most important considerations in all combustion-based power generation, including diesel engines [1]. In order to meet these demands, alternative powertrain concepts including all electric, hybrid, and fuel cell are being developed. The concept behind the hybrid devices is to store the otherwise wasted regenerative mechanical energy in energy storage devices and reuse that energy later in the work cycle. The regenerative energy recovery opportunity exists in every condition where the speed of motion is in opposite direction to the applied force or torque (Figure 1 [2]). This condition is satisfied in various conditions such as (Figure 2) the following:

(1) vehicle braking (i.e., slowdown to a lower speed on zero slope or vehicle is moving down a hill and braking must be applied to maintain a desired speed),

(2) load is moved by gravitational (load) force.

A parallel electric-hybrid powertrain concept is discussed in the next section. The real-time control challenge is to balance the machine power demands from both the engine and the hybrid storage device. The constraints faced in developing the control strategy are as follows:

(1) minimize fuel consumption while meeting low emission requirements,

(2) maintain or improve the work-machine productivity,

(3) prevent the depletion of the energy storage device, and maintain an acceptable state of charge (SOC).

A rule-based control strategy with multiple tunable gains for different work cycles is developed. The control algorithm should be robust and have low sensitivity to gain changes to accommodate real working conditions. This paper focuses on controlling hybrid powertrain mobile vehicles in real time via an optimal robust control that will enable continuous operation through repetitive and between different cycles. Yafu and Cheng [3] studied mild hybrid electric vehicles (HEV) with integrated starter generator (ISG). They used a parallel assist control strategy and modeled the system in Simulink and achieved their objective of reducing the fuel consumption.

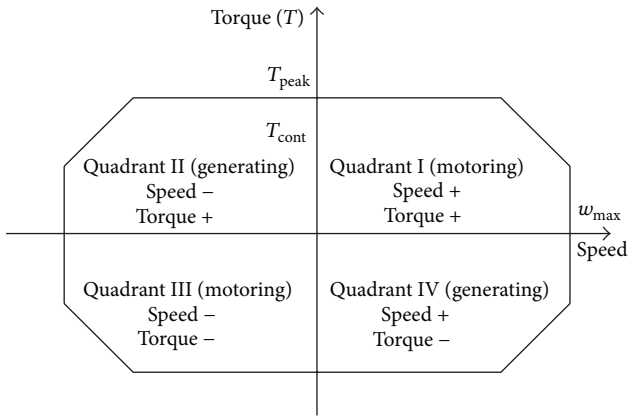


Figure 1: Torque versus speed motoring and generating quadrants (adopted from [2]).

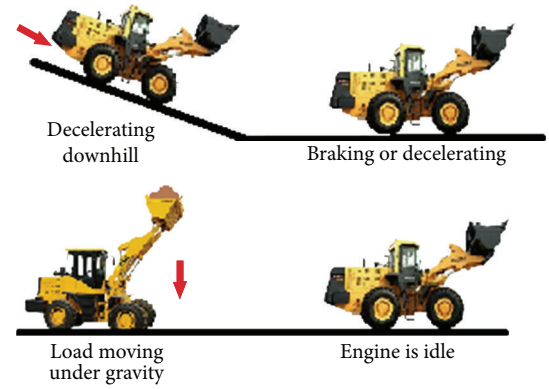


Figure 2: Regeneration opportunities.

Steinmaurer and del Re [4] used an ISG with an HEV with a statistics-based control to achieve task optimization in an attempt to emulate dynamic programming.

Two control algorithms are most commonly investigated in the literature [1]. Kessels et al. [5] surveyed the control algorithms for HEVs up to 2007. Liu and Peng [6] studied the control of Toyota PSHEV using two control algorithms: stochastic dynamic programming and equivalent consumptions minimization strategies (ECMS) [7]. They concluded that, with ECMS, frequent shifting should be avoided by adding extra constraints between gear switching decisions, while with stochastic dynamic programming approach (SDP) an extra input operating gear mode is needed beside the engine speed and the SOC. Syed et al. [8] used a fuzzy logic gain scheduling algorithm with proportional-integral (PI) controllers which are used with power-split HEV (PSHEV). The results of testing the controller on a Ford Escape showed that a minimum of four rules are needed to ensure smooth engine speed. Canova et al. [9] studied the engine start/stop dynamics, which led to an engine model that is used in a linear quadratic regulator control algorithm. Lin et al. [10] studied the dynamics of the Toyota Prius PHEV and developed an optimal control energy management strategy and artificial neural networks, that was modified to a suboptimal controller.

Gokasan et al. [11] used sliding mode control to limit the series HEVs to the optimal efficiency operating range and compared it to PI controllers. They used two simultaneous sliding mode controllers one to control the engine speed, and the other to control the engine torque. Moura et al. [12] used SDP in PHEV power management and explored the potential of charge depletion, overcharging, and the economy of fuel-electric usage in hybrid technology. Their approach allows for energy management control without prior knowledge of the driving cycle. Johannesson et al. [13] developed a GPS-based SDP control algorithm for HEV that utilizes the travel and location information of the HEV to manage its power sources to assess the potential benefit of predictive controls. They compared a position-dependent controller and a position-invariant one to an optimal control algorithm which showed a slight reduction in fuel consumption when the position information is taken into account. However, they also discovered that only the average travel information is needed and the high level of detail of travel conditions is not needed.

Huang et al. [14] used a threshold logic control strategy that is optimized continuously during operation to control a PHEV. Lee et al. [15] developed neuro-fuzzy technique to analyze and learn GPS data and control the hybrid vehicle to improve fuel economy and reduce emissions. Fu [16] developed a methodology for real-time prediction of torque availability in interior permanent magnet motors. When implemented on HEVs, it prevents torque requests at times when there are no availabilities, hence protecting the battery from depletion. According to Borhan et al. [17] dynamic programming requires knowledge of the upcoming cycle while ECMS are less sensitive to computation, short-sighted, and sensitive to their parameters. Borhan et al. [17] linearized the models and simulated over different driving cycles. They modeled the system using the resistance torque, rate of SOC equations, and Willan's line method. For this reason, MPC strategy for PSHEV has been developed to minimize fuel consumption, reduce service brake usage, and prevent overcharge and discharge of the battery. Hybrid powertrain technology has been studied and implemented in passenger vehicles and trucks; yet very little literature can be found about hybrids in construction equipment applications. Grammatico et al. [18] implemented PSHEV technology in small wheel loaders and used a classic proportional-integral-derivative controller (PID) to control the system.

The machine investigated in this work is a medium wheel loader (MWL). However, the procedures used are general such that they are applicable to other types of work machines.

2. Medium Wheel Loaders and Hybrid Technology

The prototype machine selected is a medium wheel loader (MWL). A detailed dynamic machine model is described in Figure 3. For this purpose, accurate virtual models with relatively high fidelity are needed to attempt mimicking the actual machine. However, replicating every aspect of the

machine in virtual reality is impossible due to difficulties in modeling actual physics. A detailed virtual dynamic machine model is developed using the mathematical equation and embedded in S-functions in Simulink. The model is that is validated by comparing it to experimental data. There are four general work cycles for this type of machine: truck-loading, load-and-carry, pile dressing, and roading. Both the truck-loading and the load-and-carry are cycles that involve digging, moving earth from one location to another and dumping it at the new location. The truck-loading is loading a truck or a hopper with earth and is categorized into two major cycles: aggressive truck-loading (ATL) and moderate truck-loading (MTL). ATL is characterized by its speed and the machine is operated with the engine at full throttle at all times.

Moderate truck-loading (MTL) can be as fast as or slower than ATL but the operator varies engine speed command and may never reach full throttle. The load-and-carry cycle is moving dirt to a hopper far away from the pile and is defined by the travel distance into short and long. Pile dressing is moving the earth in the pile around to make it ready for the following operations. Roading is driving the loader from one location to another without being involved in any of load movements. There are five main systems in loaders, namely,

(1) engine,

(2) transmission and lower powertrain,

(3) hydraulics (for work tool and steering),

(4) controllers, and

(5) chassis (frame cabin and linkages).

The engine is the primary source of energy for this machine from which all power requirements are drawn. The torque provided by the engine is split between two, the powertrains which are responsible for the motion of the loader and the hydraulics which are responsible for mainly linkages operations and possibly braking, cooling fan motoring, and steering.

2.1. Engine. The engine is a four-stroke turbocharged after cooled injection approximately 9-liter diesel engine, with average power rating (Figure 4) of 224–261 kW at 1800–2200 rpm and compression ratio of 17 : 1. The output gross power from the engine is not the actual available power to the wheel loader main systems due to the power consumptions from accessories such as the alternator, the muffler, the emission control, and the cooling system [19]. Thus, the power losses in the range of 6–14% must be considered when calculating the net available power to the powertrain and the hydraulics. Unlike automotive engines, the diesel engine speed in construction equipment applications is limited to about 2300 rpm. This is due to the need for higher torque values at lower machine speeds, the desire for longer engine life and reduced fuel consumption. The engine dynamic model allows for the calculation of the torque and speed along the lug curve and calculating the engine fuel consumption via the brake fuel consumption map. With hybrid implementation this engine is downsized to approximately 7 liters of diesel allowing the engine to run in a more efficient zone and making the hybrid system more cost effective.

Table 1: Transmission ratios.

Gear shift	Gear ratio
4F	−0.77
3F	−1.36
2F	−2.44
1F	−4.66
N	0
1R	4.22
2R	2.21
3R	1.23
4R	0.70

2.2. Transmission and Lower Powertrain. The main function of the powertrain is to transfer the power from the engine to the wheels in order to create the necessary rimpull for the motion through a series of speed reductions and torque multiplications. The powertrain of a wheel loader consists of wheels, axle reductions, differentials, axles, a torque converter (TC), and transmission (Figure 5 (http://www.caranddriver.com/photos-09q4/309587/2010-bmw-activehybrid-x6-automatic-transmission-and-electric-motor-diagram-photo-310275)).

A TC is a hydrodynamic coupling device which transfers power between its input and output while absorbing the difference in speed between the two shafts by slipping. The difference in power between input and output is absorbed and dissipated in the oil inside the TC in the form of heat. The two main defining characteristics of the TC are the primary torque T_p which is the value of the rated torque of the impeller at the rated speed N_{rated} (usually 1700 rpm) and the torque ratio T_r which is the ratio between the output turbine torque and the input impeller torque. Also, T_r is function of the speed ratio S_r which is the ratio between the output turbine speed and the input impeller speed (N_i). Through the knowledge of T_p, S_r, and the rated and the input speed N_i and the input torque T_i can be calculated at various speeds using (1) and Figure 6:

$$T_i = T_p\big|_{S_r}\left(\frac{N_i}{N_{\text{rated}}}\right)^2. \tag{1}$$

The TC is followed by a planetary gearbox which reduces the output speed to a desired range for the machine ground velocity. The gearbox is constituted of several planetary gear trains connected together via electrohydraulically controlled brakes and clutches which are used to select the gear. The explanation of how the mechanism works is available in the literature [19]. The regular gear train in wheel loaders has four forward speeds, four reverse, and one neutral (Table 1). The dynamic transmission model is based on multiple look-up tables for the gear combinations and clutches' pressures.

2.3. Hydraulics. The hydraulic system of the prototype MWL machine used in this study is a closed center load sensing

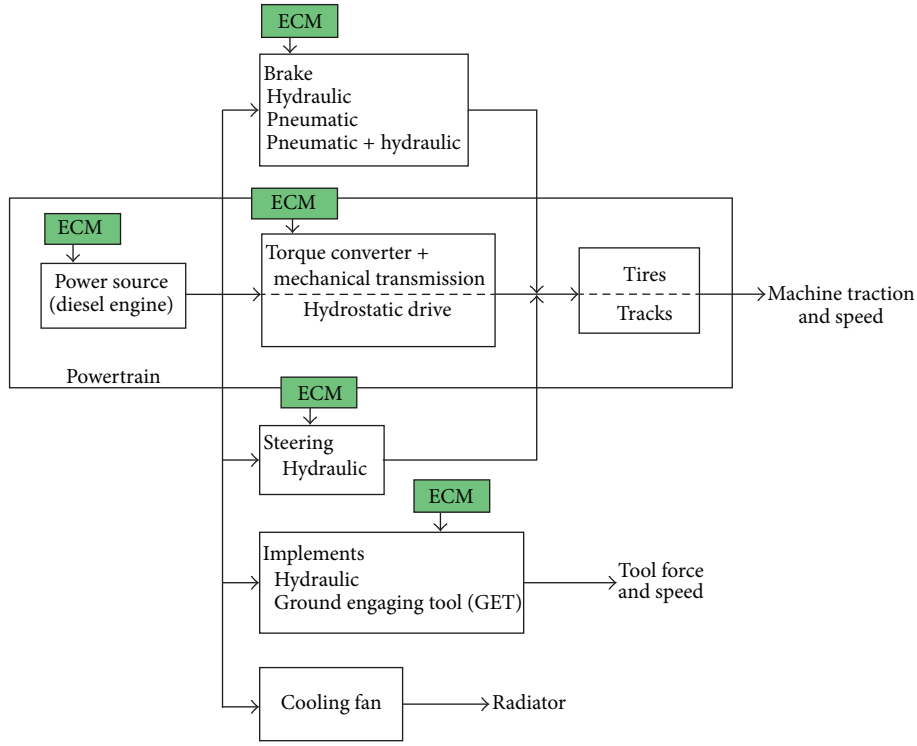


Figure 3: Wheel loader systems.

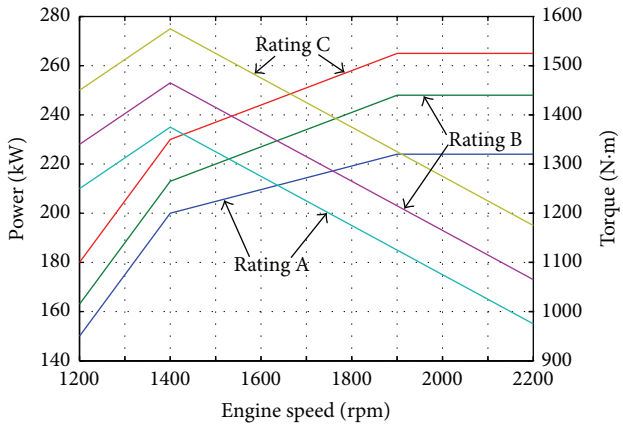


Figure 4: Rated engine lug curve and power rating.

hydraulics system for the work tool (implement) circuit. This system is the common trend in MWL as it avoids dissipation of energy and adapts the amount of flow provided by the pump to meet the needs of the machine, thus minimizing the losses unlike open center systems. The load sensing system compares the pump pressure at the output to the cylinders pressures, and then it adjusts the pump's swash plate based on that feedback in order to maintain a certain pressure differential (Figure 7 (http://www.mobilehydraulictips.com/efficient-mobile-hydraulic-systems/)). A shuttle valve senses the highest pressure between the tilt and lift pressures and sends it through feedback for comparison. Equation (2) governs the angle of the pump's swash plate [2]:

$$\theta_{\text{cmd}} = \theta_{\text{offset}} + K\left(\Delta P_{\text{cmd}} - \left(P_s - P_L\right)\right), \tag{2}$$

where θ_{cmd} is the commanded swash plate angle, θ_{offset} is the swash plate angle offset from the desired position, K is a gain, ΔP_{cmd} is the desired pressure differential, P_s is the pump outlet pressure, and P_L is the load pressure.

A pressure limiting control system in the pump is called the high pressure cut-off or pressure compensator, whose function is to limit the maximum system pressure by reducing pump displacement to zero when the set pressure is reached. The post compensation regulates the amount of flow sent to the tilt and lift based on the pressure feedback from both valves such that the valve with the higher pressure receives more flow than that with the lower pressure; that is, it maintains a constant pressure drop across the valve. The spool movement of a compensator valve is governed by (3):

$$\Delta x_s = \frac{1}{K_{\text{spring}}}\left(p_s A_s - p_l A_l\right), \tag{3}$$

where p_s is the supply pressure, A_s is the area of the piston at the supply pressure side, p_l is the load pressure, A_l is the area of the piston at the load pressure side, K_{spring} is the spring coefficient, and Δx_s is the length of the opening from the supply to the load in the valve. Equations (4) through (7)

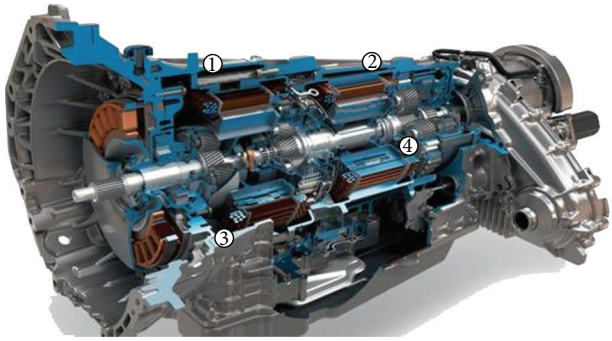


Figure 5: Transmission.

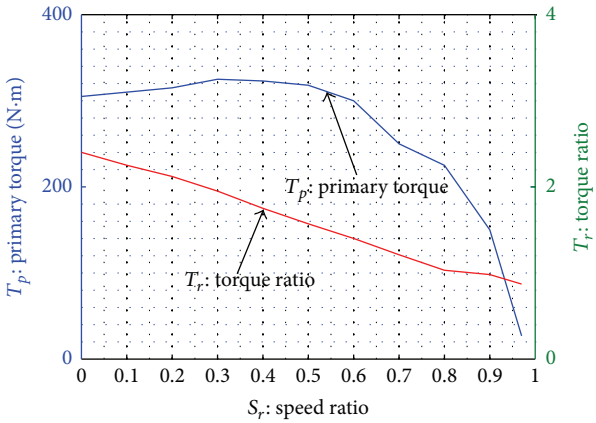


Figure 6: Torque converter characteristics.

govern the dynamics and static hydraulics and the hydromechanical equations,

$$Q = \frac{V\dot{P}}{\beta}, \tag{4}$$

$$Q = C_d A \sqrt{\frac{2\Delta P}{\rho}}, \tag{5}$$

$$Q = A\dot{x}, \tag{6}$$

$$F = PA, \tag{7}$$

where β is the bulk modulus, Q is the volumetric flow rate, P is the pressure, ρ is the oil density, C_d is the valve coefficient, A is the valve opening area, and x is the displacement of the piston.

2.4. Chassis. The chassis is the body and linkages of the machine and it is governed by the mechanical kinematic constraints and dynamics which can be represented using multibody dynamics approaches.

2.5. Electric Hybrid. The electric hybrid concept (Figure 8) is used to convert the regenerative mechanical energy to electrical energy via an electric generator and then store it in electrochemical batteries. The electric hybrid consists of a dual-function motor/generator (ISG) actuator, an inverter that requires a separate cooling system, and a lithium-Ion battery pack. The Li-I battery was selected as it is the most promising rechargeable battery technology available and is widely used in electric hybrid technology [20]. The SOC changes over time interval dt, with discharging or charging current i. The battery SOC can be calculated using (8):

$$\mathrm{SOC} = \mathrm{SOC}_0 - \int \frac{i(t)}{Q(i(t))} dt, \tag{8}$$

where $Q(i(t))$ is the ampere-hour capacity of the battery at current rate $i(t)$. While the electric hybrid is the most expensive, it is receiving the most investment in all mobile industries due to its maturity. With the engine downsizing, the ISG system costs come to neutral. The diesel engine is the primary power plant; electric hybrid is the energy bumper. The ISG adds power to powertrain from the battery when power assist is needed. It also stores the energy to the battery from the powertrain when a power storage opportunity exists. The electric hybrid cooling cycle passes through all the components where the coolant fluid goes through the shunt tank into radiator, through the pump, inverter, battery, and ISG and then back to radiator. Also, the shunt tank is connected to all the hybrid parts to allow air bubbles to escape and to the radiator to allow coolant to expand and supply excess fluid.

3. Control Strategies

In this section, the proposed control strategy is presented. The control strategy under investigation is a rule-based control. Based on the analysis and the results it will be determined if gain scheduling via GPS tracking and cycle recognition algorithm is needed. The rule-based control gains are tuned to different cycles to minimize the fuel consumption, bring the final SOC closest to the initial SOC, minimize the cycle time, and regulate the minimum engine speed to be close to 1000 rpm or higher (Figure 9). The high-level Simulink diagram is shown (Figure 10).

3.1. Cycle Model. The cycle model is built to mimic the commands of the real operator sent to the different machine systems, as well as the work area. In total, there are four different cycles and hence four different operator models used in this investigation: ATL, MTL, short load-and-carry (SLC), and long load-and-carry (LLC). Table 2 shows the input and output signals to the operator model. The cycle model is modeled as "if statements" with multiple possible output scenarios based on time and distance along the cycle. Each if statement output corresponds to a group of time- and distance-based operator commands to the machine. The sequence of these commands will result in the machine operations that will lead to completing the cycle. Table 3 shows the input and output signals to the controller.

3.2. Rule-Based Logic. The rule-based control (Figure 10) is designed based on knowledge of machine functions and system of the machine. This control is torque-based control

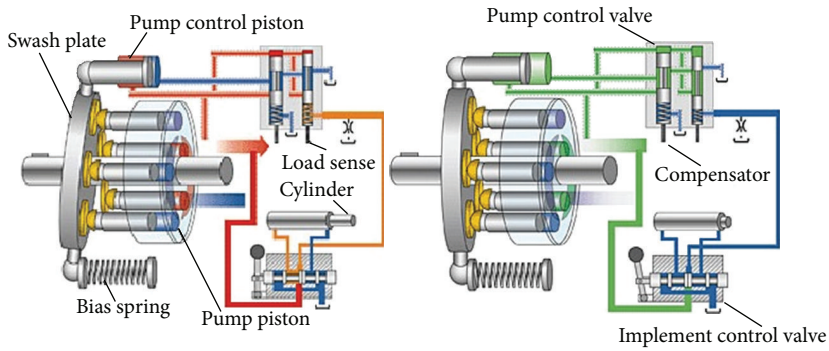


Figure 7: Load sensing axial piston pump with swash plate (http://www.mobilehydraulictips.com/efficient-mobile-hydraulic-systems/).

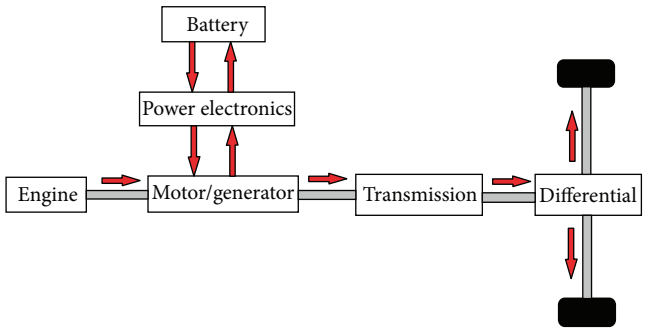


Figure 8: Electric hybrid concept (adopted from [6]).

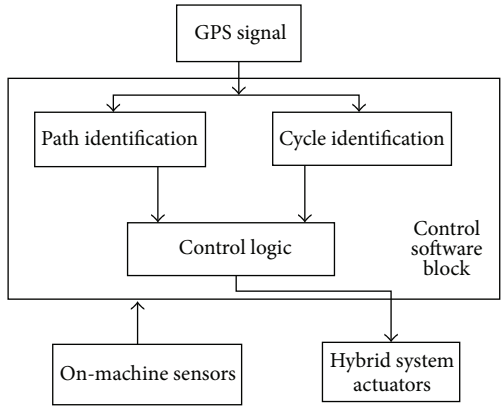


Figure 9: Rule-based control block diagram.

Table 2: Operator model signals.

Cycle model input signals	Cycle model output signals
(1) Engine speed (rpm)	(1) Desired engine speed (rpm)
(2) Desired gear number	(2) Gear command
(3) Lift displacement (m)	(3) Lift command
(4) Tilt displacement (m)	(4) Tilt command
(5) Machine ground velocity (m/s)	(5) Brake command
	(6) Bucket load
	(7) Grade
	(8) Drawbar

Table 3: Rule-based logic signals.

Input signals	Output signals
(1) Engine speed (rpm)	(1) ISG torque command
(2) Desired engine speed (rpm)	
(3) Engine load torque (Nm)	
(4) Engine maximum torque (Nm)	
(5) SOC	
(6) Gear command	
(7) Grade	

that will send out desired torque to the ISG and based on this the generator-motor action will be determined. To decide whether to charge or assist multiple factors are considered. These factors are represented by gains and thresholds that will enable reaching the decision. These gains and thresholds are tuned for different cycles to achieve optimum robust results based on the criteria listed earlier. The gains tuned in this logic are: delay timer, assist threshold, charge threshold, low SOC threshold, idle charge torque, ISG torque-required threshold, engine speed factor, assist/charge threshold offset, and the idle charge SOC threshold.

To reach this decision, the first step is to multiply the engine torque limit by three of the gains and compare the results to the engine load torque. These gains are the assist threshold, charge threshold, and hysteresis factor. Another gain existing to assist in decision making is the low SOC threshold which is biased to charging when the SOC drops below it. For low SOC mode, the more aggressive (charge bias) assist/charge thresholds should be arrived at by taking the standard values and offsetting them by assist/charge threshold offset. Along with comparison of the engine speed and torque demand with the actual, this procedure will determine the torque demand out of the engine. This step determines if the engine is capable of supplying the demanded power on its own, needs assistance from the hybrid, or has excess power to be used for charging. If the engine is idle and the SOC is less than the idle SOC charge

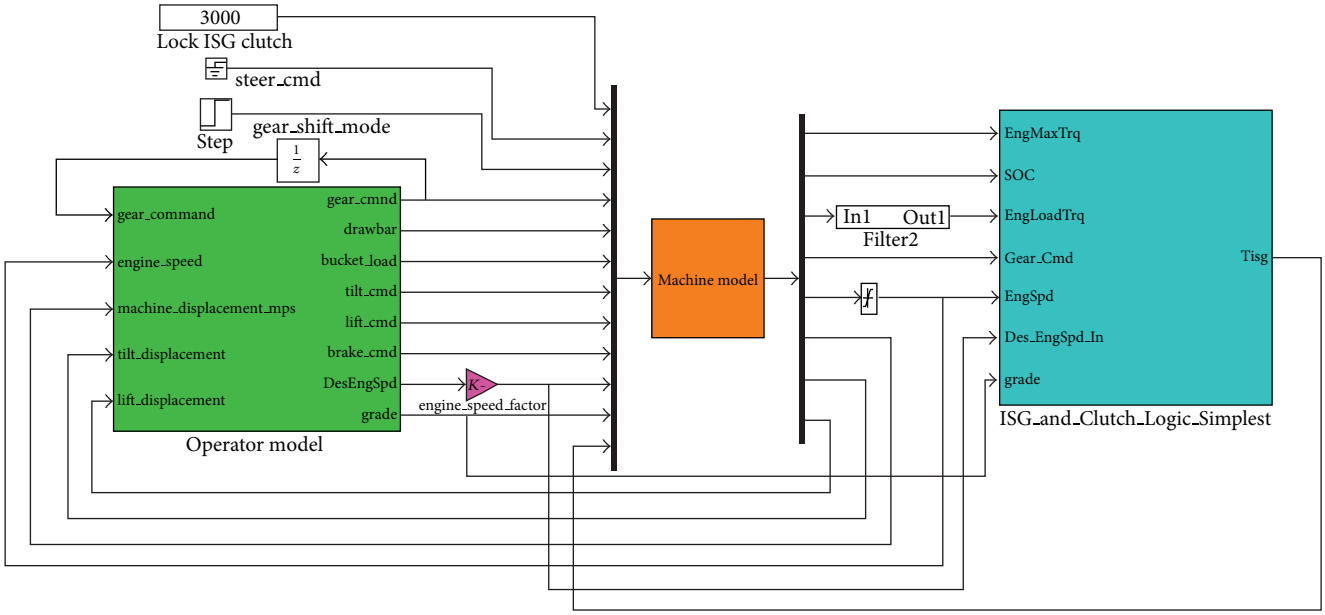


Figure 10: Rule–based control upper level Simulink diagram.

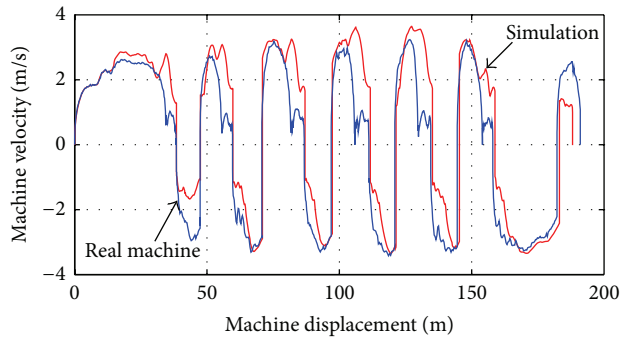


Figure 11: Validation machine velocity.

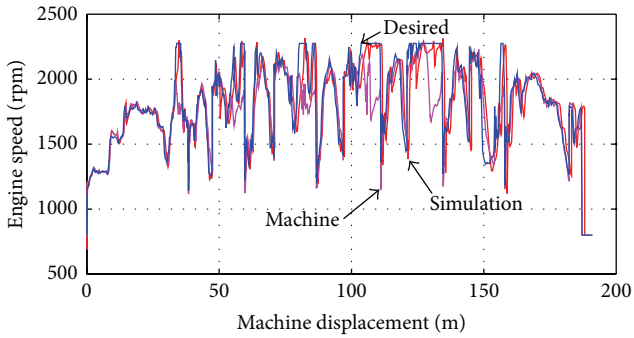


Figure 12: Validation engine speed.

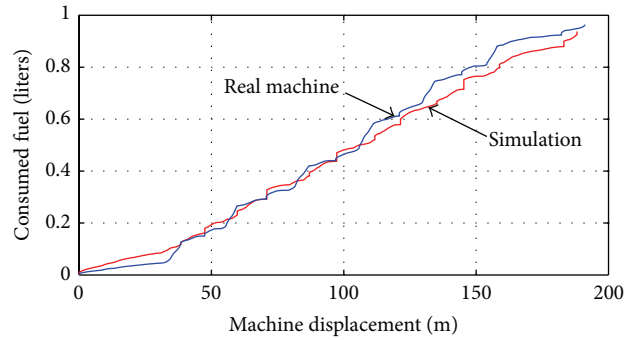


Figure 13: Validation consumed fuel.

threshold, then a charge command is issued. Charging is also enforced when the retard engine speed threshold is exceeded.

With any gear upshift or if the engine deceleration rate exceeds the engine deceleration rate threshold, assist is triggered. The gain tuning process is an iterative and time-consuming process. The objective is to obtain an optimal set of parameters for robust design such that slight changes in the gain values do not affect the performance much. The orthogonal array experiments only require a fraction of the full-factorial combinations and the same result can be achieved. This can be done by using the analysis of the variance (ANOVA) thus making it the most suitable for tuning the nine gains. A predictive model based on the ANOM is then formulated through which an estimate of the optimal set of interactions is determined and tested. By repeating this process over time, the desired optimal robust solution can be obtained for each cycle [21].

4. Results

In this section, the machine virtual dynamic model validation as well as the results obtained during this work is presented. The validation results show that the used machine model is in good correlation with the real-life machine. The results

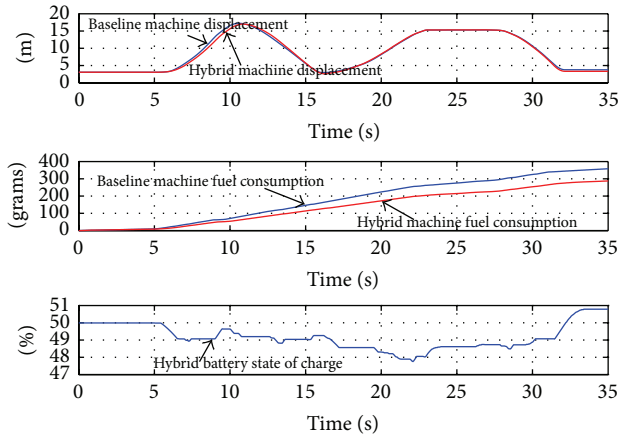


Figure 14: ATL hybrid versus baseline.

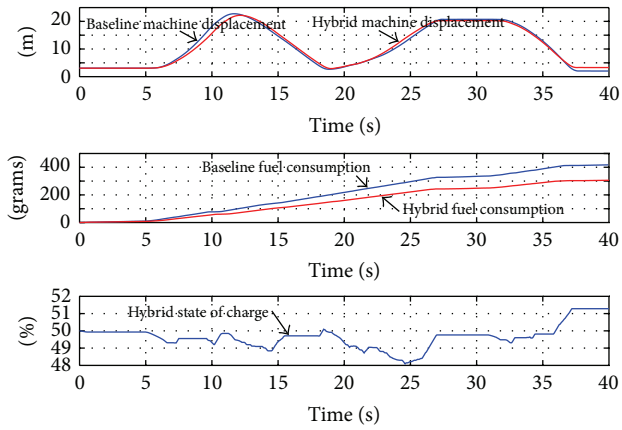


Figure 15: Moderate truck loading cycle.

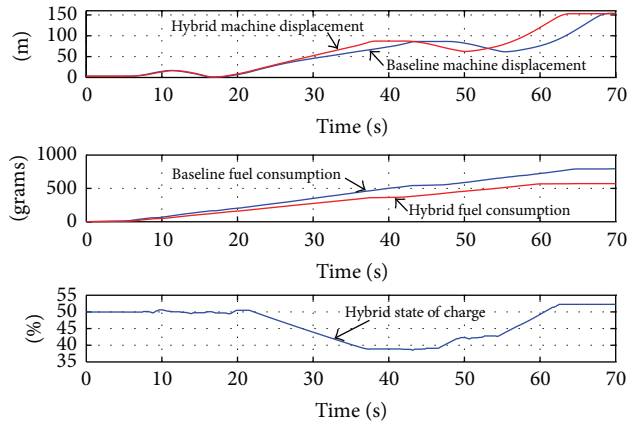


Figure 16: Short load and carry cycle.

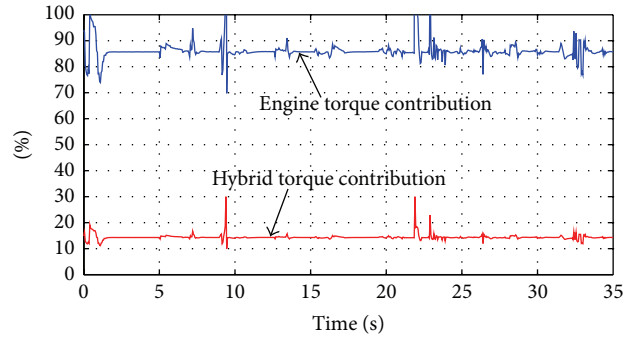


Figure 17: Torque contribution.

obtained show that when the engine is downsized, the hybrid system is implemented, its control logic gains are optimized, and the system performs very well without GPS or cycle recognition algorithm. The model validation is performed by obtaining machine data from the real-life wheel loader and giving the same operator commands to the machine model.

Figure 11 shows both the real and simulated machine velocities. The general speed pattern is the same and is in good agreement. The differences could be attributed to the use of approximate dynamics in the different machine systems. Figure 12 shows the desired engine speed sent by the operator to the machine and implemented in the cycle model, the real machine engine speed, and the simulated engine speed. From Figure 12 it is clear that the modeled engine speed response to the desired engine speed command matches to a great extend the actual engine speed response. The differences could be attributed to the use of approximation and estimated numbers in various points in the model including the load on the machine. Figure 13 shows the amount of fuel consumed by the machine versus that estimated by the machine model. The general trend of the fuel consumption is the same and the difference in the end point is minimal. Thus, it can be concluded that the machine model has a very good correlation with the real machine. Three work cycles were used in the analysis: ATL, MTL, and SLC cycles. Figure 14 compares between the baseline regular and the optimized hybrid wheel loader for the ATL cycle.

It is shown that maintaining productivity while decreasing the fuel consumption by 20% can be achieved by switching to the downsized hybrid machine. Figure 15 compares between the baseline regular and the optimized hybrid wheel loader for the MTL cycle. From the obtained results it is shown that maintaining productivity while decreasing the fuel consumption by 26.5% can be achieved by switching to the downsized hybrid machine.

Figure 16 compares between the baseline regular and the optimized hybrid wheel loader for the SLC cycle. From the obtained results it is shown that increasing productivity by 7.7% and decreasing the fuel consumption by 28% can be achieved by switching to the downsized hybrid machine. From Figure 17, it can be deduced that the hybrid system contributes to about 14–17% of the total torque needed by the machine. After the optimization of the control parameters for each independent cycle, the investigation of the effect of using these parameters in different cycles should be carried out. This investigation will determine if online cycle identification and GPS mapping of the worksite is needed.

The optimized gain of the MTL and SLC will be tested in the ATL and compared to the results obtained by its

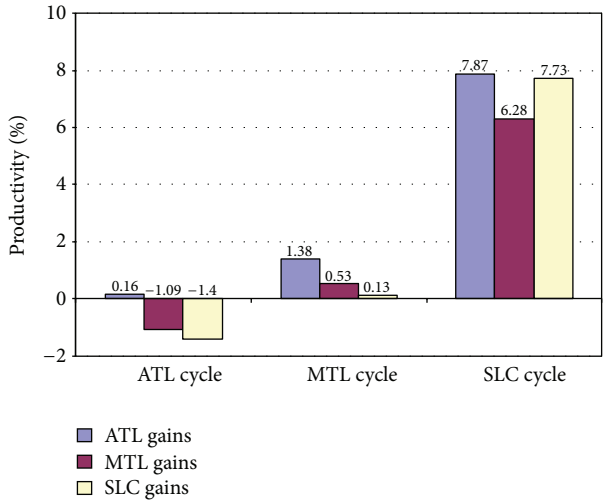


Figure 18: Effect of different gain groups on cycles' productivity.

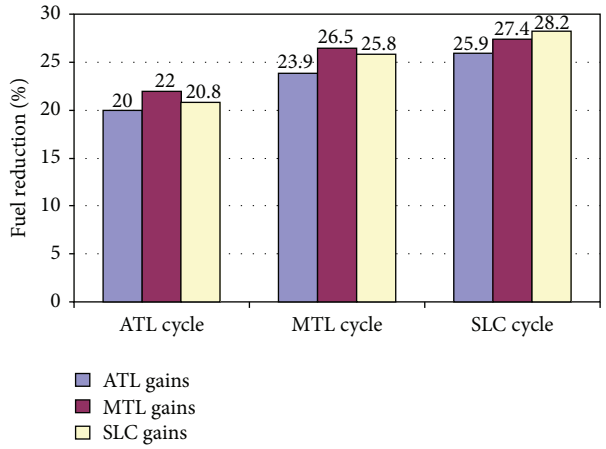


Figure 19: Effect of different gain groups on cycles' fuel reduction.

optimized gains, the gain groups of the ATL and SLC will be tested in the MTL and compared to the results obtained by its optimized gains, and the gain groups of the ATL and MTL will be tested in the SLC and compared to the results obtained by its optimized gains. By examining the results closely (Figures 18 and 19), all the cycles exhibit a change of less than 2% in terms of productivity and less than 3% in terms of fuel reduction from that of the optimized cycle.

From this it can be concluded that the knowledge of the running cycle is not needed as long as the machine is using any optimized set of parameters. Hence, online cycle identification and GPS mapping is not needed to be implemented on the machine but remains useful as an analysis tool for the worksite and operator performance.

5. Conclusions

The implementation of an electric hybrid powertrain on a medium wheel loader is expected to maintain the productivity of the machine within acceptable range. The usage of the hybrid system with a downsized engine is expected to reduce the fuel consumption on the machine by 20–30% at different cycles. The hybrid system supplies 14–17% of the torque required by the machine during its work cycle. By optimizing the hybrid controller gains for any work cycle, the controller can be used with any work cycle. Thus, the GPS tracking and gain scheduling algorithms are not needed.

Abbreviations

ANOVA: Analysis of the variance
ATL: Aggressive truck-loading
BSFC: Brake-specific fuel consumption
CVT: Continuously variable transmission
ECMS: Equivalent consumptions minimization strategies
GPS: Global positioning system
HEV: Hybrid electric vehicle
ISG: Integrated starter generator
LLC: Long load-and-carry
MTL: Moderate truck-loading
MWL: Medium wheel loader
PHEV: Plug-in hybrid electric vehicle
PSHEV: Power-split hybrid electric vehicle
SDP: Stochastic dynamic programming
SLC: Short load-and-carry
SOC: State of charge
TC: Torque converter.

References

[1] C. D. Rakopoulos and E. G. Giakoumis, *Diesel Engine Transient Operation: Principles of Operation and Simulation Analysis*, Springer, 2009.

[2] S. Cetinkunt, *Mechatronics*, John Wiley and Sons, 2007.

[3] Z. Yafu and C. Cheng, "Study on the powertrain for ISG mild hybrid electric vehicle," in *Proceedings of the IEEE Vehicle Power and Propulsion Conference (VPPC '08)*, September 2008.

[4] G. Steinmaurer and L. del Re, "Optimal energy management for mild hybrid operations of vehicle with an integrated starter generator," SAE Technical Paper 2005-01-0280, pp. 73–80, 2005.

[5] J. T. B. Kessels A, J. Sijs, R. M. Hermans, A. A. H. Damen, and P. P. J. van den Bosch, "On-line identification of vehicle fuel consumption for energy and emission management: an LTP system analysis," in *Proceedings of the American Control Conference (ACC '08)*, Seattle, Wash, USA, June 2008.

[6] J. Liu and H. Peng, "Modeling and control of a power-split hybrid vehicle," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 6, pp. 1242–1251, 2008.

[7] G. Paganelli, Y. Guezennec, and G. Rizzoni, "Optimizing control strategy for hybrid fuel cell vehicle," SAE Technical Paper 2002-01-0102, SAE, Warrendale, Pa, USA, 2002.

[8] F. U. Syed, M. L. Kuang, M. Smith, S. Okubo, and H. Ying, "Fuzzy gain-scheduling proportional-integral control for improving engine power and speed behavior in a hybrid electric vehicle," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 1, pp. 69–84, 2009.

[9] M. Canova, Y. Guezennec, and S. Yurkovich, "On the control of engine start/stop dynamics in a hybrid electric vehicle," *Journal*

of Dynamic Systems, Measurement and Control, Transactions of the ASME, vol. 131, no. 6, pp. 1–12, 2009.

[10] X. Lin, H. Banvait, S. Anwar, and C. Yaobin, "Optimal energy management for a plug-in hybrid electric vehicle: real-time controller," in *Proceedings of the American Control Conference (ACC '10)*, pp. 5037–5042, Baltimore, Md, USA, July 2010.

[11] M. Gokasan, S. Bogosyan, and D. J. Goering, "Sliding mode based powertrain control for efficiency improvement in series hybrid-electric vehicles," *IEEE Transactions on Power Electronics*, vol. 21, no. 3, pp. 779–790, 2006.

[12] S. J. Moura, H. K. Fathy, D. S. Callaway, and J. L. Stein, "A stochastic optimal control approach for power management in plug-in hybrid electric vehicles," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 3, pp. 545–555, 2011.

[13] L. Johannesson, M. Åsbogård, and B. Egardt, "Assessing the potential of predictive control for hybrid vehicle powertrains using stochastic dynamic programming," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 1, pp. 71–83, 2007.

[14] Y. J. Huang, C. L. Yin, and J. W. Zhang, "Design of an energy management strategy for parallel hybrid electric vehicles using a logic threshold and instantaneous optimization method," *International Journal of Automotive Technology*, vol. 10, no. 4, pp. 513–521, 2009.

[15] S. H. Lee, S. D. Walters, and R. J. Howlett, "Intelligent GPS-based vehicle control for improved fuel consumption and reduced emissions," in *Proceedings of the 12th International Conference on Knowledge-Based Intelligent Information and Engineering Systems*, pp. 701–708, Zagreb, Croatia, 2008.

[16] Z. X. Fu, "Real-time prediction of torque availability of an ipm synchronous machine drive for hybrid electric vehicles," in *Proceedings of the IEEE International Conference on Electric Machines and Drives*, pp. 199–206, San Antonio, Tex, USA, May 2005.

[17] H. A. Borhan, A. Vahidi, A. M. Phillips, M. L. Kuang, and I. V. Kolmanovsky, "Predictive energy management of a power-split hybrid electric vehicle," in *Proceedings of the American Control Conference (ACC '09)*, pp. 3970–3976, June 2009.

[18] S. Grammatico, A. Balluchi, and E. Cosoli, "A series-parallel hybrid electric powertrain for industrial vehicles," in *Proceedings of the IEEE Vehicle Power and Propulsion Conference (VPPC '10)*, pp. 1–6, September 2010.

[19] F. Croce, *Optimal Design of Powertrain and Hydraulic Implement Systems for Construction Equipment Applications [Ph.D. thesis]*, University of Illinois at Chicago, 2010.

[20] M. Ehsani, Y. Gao, and A. Emadi, *Modern Electric, Hybrid Electric, and Fuel Cell Vehicles: Fundamentals, Theory, and Design Second Edition*, Taylor & Francis, 2010.

[21] Y. W. Fowlkes and C. M. Creveling, *Engineering Methods For Robust Product Design Using Taguchi Methods in Technology and Product Development*, Prentice Hall, 1995.

Hindawi Publishing Corporation
Journal of Control Science and Engineering
Volume 2013, Article ID 678016, 15 pages
http://dx.doi.org/10.1155/2013/678016

Research Article

Model-Based Control Design and Integration of Cyberphysical Systems: An Adaptive Cruise Control Case Study

Emeka Eyisi, Zhenkai Zhang, Xenofon Koutsoukos, Joseph Porter, Gabor Karsai, and Janos Sztipanovits

Institute for Software Integrated Systems (ISIS), Vanderbilt University, Nashville, TN 37212, USA

Correspondence should be addressed to Emeka Eyisi; emeka.eyisi@vanderbilt.edu

Received 7 September 2012; Accepted 18 December 2012

Academic Editor: Sabri Cetinkunt

The systematic design of automotive control applications is a challenging problem due to lack of understanding of the complex and tight interactions that often manifest during the integration of components from the control design phase with the components from software generation and deployment on actual platform/network. In order to address this challenge, we present a systematic methodology and a toolchain using well-defined models to integrate components from various design phases with specific emphasis on restricting the complex interactions that manifest during integration such as timing, deployment, and quantization. We present an experimental platform for the evaluation and testing of the design process. The approach is applied to the development of an adaptive cruise control, and we present experimental results that demonstrate the efficacy of the approach.

1. Introduction

Cyberphysical systems (CPS) represent a class of complex systems characterized by the tight interactions between the physical dynamics, computational dynamics, and communication networks. Automotive systems are classical examples of CPS and have recently been gaining increased attention due to the emerging challenges in their design. Current automotive systems employ up to 100 electronic control units (ECUs) exchanging more than 2500 signals over up to 5 different bus systems [1, 2]. These ECUs control and monitor many subsystems of a vehicle such as chassis control, vehicle stability, and engine control. The development of control software has become one of the greatest challenges in the automotive domain due to the increasing complexity of automotive systems as well as the increasing roles of control, computing, and communication [3–6].

The increased pressure to integrate as much functionality on as few ECUs as possible, the persistent effort for low production costs, and tight time-to-market constraints further complicate the development of automotive software control systems. Due to these challenges, there is a dire need for a reliable and efficient approach for control software development and integration. A major problem in the current state of art is that most issues with deployed control applications are typically discovered in the final phases of the development cycle, and at these later phases, correcting the issues is very expensive as it involves the modification of specifications, requirements, and design. Another issue is the lack of realistic experimental platforms for integrating and testing developed control software prior to deployment.

Systematic design and analysis of automotive control software early in the development cycle is very crucial. The model-driven development approach has been found to be very beneficial in addressing these issues [7]. However, the lack of a sound approach, for the integration of components from the control design phase with the components from software generation and deployment on actual platform/network, makes the model-driven approach very challenging because the tight interactions between the design phases often manifest during integration. The current state-of-the-art often resorts to ad hoc methods with the goal of "making the system work." These ad hoc methods are becoming unpractical as the complexity of the system increases.

In this paper, we present a step towards addressing the challenges in the design and integration of components from control design with components from software generation and deployment on actual platform/network. We present a systematic methodology and a toolchain to integrate control design and scheduling in the development of automotive control applications with specific emphasis on restricting the interactions that manifest during integration such as timing, deployment, and quantization. Our design process utilizes a model-based toolchain, Embedded Systems Modeling Language (ESMoL) [8]. ESMoL, designed in the Generic Modeling Environment (GME) [9], is a single multiaspect embedded software design environment, which streamlines control design with software modeling, code generation, and deployment on platform/network, filling the very important details between the design phases promoting a high-confidence software development process.

In order to evaluate our software development process and toolchain, we employ an experimental platform based on the time-triggered paradigm that enables the deployment and testing of automotive control applications. The time-triggered paradigm is used to address the complexity and composability challenges by precisely defining the interfaces between components in order to provide predictability [10]. Also, our choice of a time-triggered paradigm falls in line with the increased ongoing efforts towards the standardization of in-car communication networks, such as FlexRay and Time-Triggered Ethernet (TTEthernet or TTE), with the overall goal of guaranteeing highly reliable, deterministic, and fault-tolerant system performance [11].

The software development process and toolchain together with the experimental platform efficiently connect all the phases of development of automotive control applications, essentially going from control design using Matlab/Simulink to deployment and hardware-in-the-loop simulations. In order to demonstrate our approach, we apply the proposed process to the development of an adaptive cruise control (ACC). The adaptive cruise control (ACC) system is an active safety and driver-assistance vehicle feature that automatically controls a vehicle's longitudinal velocity in a dynamic traffic environment. ACC enables an ACC-equipped vehicle to follow a leading forward moving vehicle while maintaining a desired distance from the leading vehicle as determined by the vehicle's velocity and a specified time gap or headway. We present experimental results from the hardware-in-the-loop simulations of the designed ACC on the experimental platform.

This paper is organized as follows. The related work is presented in Section 2. In Section 3, we describe our view of CPS design and integration, and we formulate the specific problem considered in this paper. We present the proposed systematic methodology for model-based control design and integration in Section 4. Section 5 presents the system architecture for the experimental platform. Section 6 describes the control design of the adaptive cruise control. Section 7 describes the software design process for the adaptive cruise control. Section 8 presents an experimental evaluation of our proposed approach using the adaptive cruise control case study. Finally, Section 9 provides a brief discussion and concludes the paper.

2. Related Work

Model-based software development approach as well as test-beds for testing automotive control systems architecture is a very active research area. There is an increasing amount of work in this area attempting to address various cross-layer challenges [12]. In [13], an automotive testbed for electronic controller unit testing and verification was presented. The presented platform provides many advantages for testing ECUs and is complementary to our work. The authors in [14] present a software-based implementation and verification scheme for a FlexRay-based automotive network. The main focus of the paper is on verifying timing in control signals and the network and providing a basis for detecting and diagnosing network faults. In [15], the authors used a technique called Instrumentation-Based Verification (IBV) to design automated tests in order to check whether models developed in Matlab/Simulink satisfy specified requirements; while the work addresses an important issue of verifying requirements with control design, it does not go further into the actual software deployment and evaluation on a test platform.

The design of the adaptive cruise control (ACC) has been extensively studied, and there are numerous design techniques for deriving the corresponding control laws. Some of the most common approaches are sliding-mode design techniques [16, 17], optimal control techniques [18, 19], fuzzy logic [20], neural networks, and proportional derivative (PD) type control law [21, 22]. In this work, our main focus in the case study is on the design flow from the high level design of a vehicle control system such as the ACC using model-based tools such as Matlab/Simulink to the actual deployment and testing on an automotive testbed capable of mimicking real world scenarios. In [23], the authors describe a model-based approach for the modeling, design,dere and implementation of an intelligent cruise control. In contrast to their approach, our development approach provides a simpler graphical language that clearly defines the integration of the control software. In addition, we adopt the time-triggered architecture [10] which essentially ensures predictability, determinism, and guaranteed latencies, hence, allowing for a certain level of decoupling in system design. The work in [24] describes a FlexRay-based distributed networked system for automotive applications mainly focusing on the challenges in regards to the paradigm shift from the Controller-Area-Network-(CAN-) based even-triggered communication technologies to the introduction of time-triggered communication scheme while assuming an existing software environment.

Our work differs from the existing works, due to the fact that we consider an end-to-end design flow in the development of control software with well-defined components that are necessary for the efficient and reliable integration of design layers of an automotive CPS. By clearly defining these components which include timing and deployment, we restrict the possible interactions that can potentially make the overall behavior of the system unpredictable.

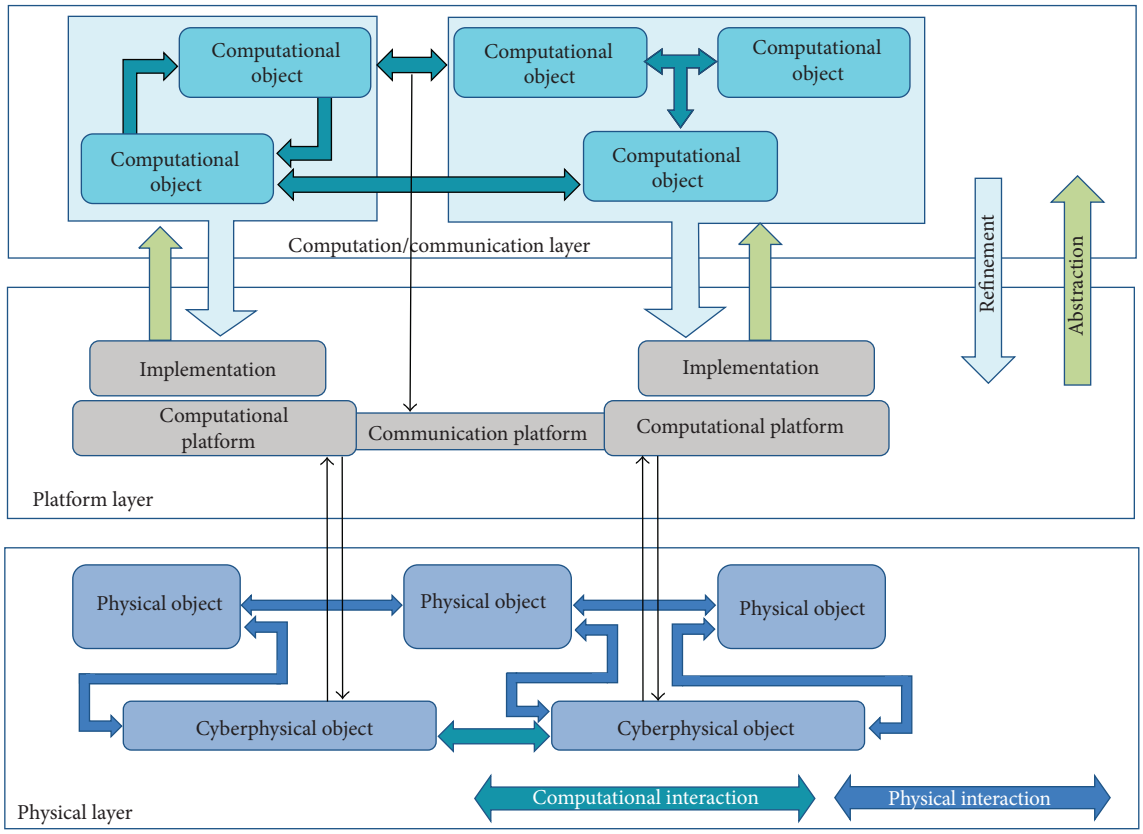


Figure 1: Design flow in CPS design layers [25].

3. Problem Formulation

Figure 1 shows the three fundamental design layers of CPS, such as an automotive vehicle [25].

(1) *The physical layer* represents physical components and their interactions. The behavior of the physical components is governed by physical laws and is typically described in continuous (physical) time using, for example, ordinary differential equations (ODEs). Physical objects are interconnected by physical components (e.g., steering wheel) or cyberphysical objects (e.g., steer by wire).

(2) *The platform layer* represents the hardware side of CPS and includes the network architecture and computation platform that interact with the physical components through sensors and actuators. While executing the software components on processors and transferring data on communication links, their abstract behavior is "translated" into physical behavior.

(3) *The computation/communication layer* represents the software components with behavior expressed in logical time. Interconnections are modeled using various Models of Computations (MoCs) [26]. Software components are connected using an input/output model with an implied notion of causality.

In view of this CPS architecture, for an automotive application, the vehicle chassis together with the engine, transmission, brakes and tires, cyberphysical objects (e.g., steer by wire), and the initial controller design comprise the physical layer. The electronic control units (ECUs) on which the control software applications are deployed, together with the communication network over which the ECUs send and receive data, comprise the platform layer.

In this work, we assume that the components of the physical layer are specified by a given physical vehicle dynamic model. Also, we assume that the platform layer is specified based on a given set of computational nodes and communication network. The main research problem we address is handling the complex interactions in the computation/communication layer of automotive CPS which manifest due to the lack of a clear and well-defined systematic integration of control design and scheduling. This problem involves the need for a "correct-by-construction" end-to-end design methodology for the modeling, designing, analysis, deployment, and testing of automotive control applications. In order for such a development process to be beneficial, it should be systematic and efficient. Additionally, such an

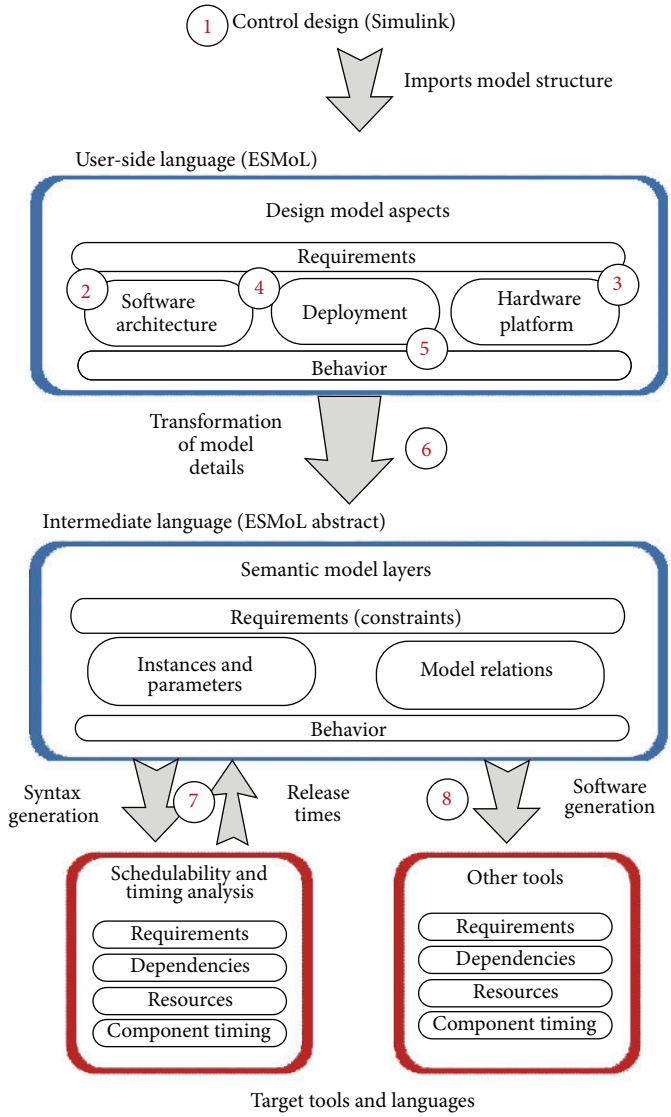


Figure 2: Automotive embedded control software design flow supported by the ESMoL language and modeling tools.

approach will require an experimental platform that is able to model realistic scenarios that can mimic real-world cases.

4. Automotive Control Software Design Process

The proposed approach uses formal models and design concepts integrated in the model-based tool chain ESMoL, to restrict component interactions by specifying attributes in various refinements of a single design model of an application. These refinements, which include component architecture, deployment, and timing/execution models, represent different aspects of the system as the process progresses towards implementation. The combined information expressed in the aspects constitutes a suitable complete model for the actual deployment and implementation of the system. We provide the design flow of the development process.

4.1. Automotive Control Software Design Flow. Figure 2 shows the design flow for the proposed software development process. The design process involves eight main steps numbered in a top-down manner, starting from the first step which involves the importation of a control design model into ESMoL to the eighth and final step of software generation, as shown in Figure 2. The design steps are described as in Figure 2.

4.1.1. Controller Design and Importation. The controller for a specific system functionality is typically designed and validated in Matlab/Simulink using simulations. Real-Time Workshop (RTW) [27], an automatic code generator in Simulink, can subsequently be used to generate the equivalent C code of the designed controller. The controller is typically designed in floating-point math, while the actual control software executes on ECUs with potentially limited number of bits and fixed-point implementation. In order to convert

controller models from floating-point to fixed-point, fixed-point Toolbox [28] is used to aid code generation.

Although Matlab/Simulink provides the environment for modeling and code generation of controllers, it lacks the adequate and important refinement for controllers that provide implementation details, such as real-time operating system (RTOS) environment, timing, hardware platform specifications, and network considerations. Thus, after the controller validation and generation of equivalent C code, in the first step of our design process, the controller's Matlab/Simulink model is automatically imported into the ESMoL environment by using the MDL2MGA tool in ESMoL. The MDL2MGA tool is a model interpreter that creates a structural replica of the Matlab/Simulink model in the ESMoL modeling environment. The replica of the Matlab/Simulink model is represented as a synchronous data flow model (SDF), and each subsystem in the replica becomes an actor in the SDF. The ESMoL model's references to the imported Simulink blocks become the functional specifications for the instances of software components in a logical SDF model. C code fragments may also be used to specify the component functionality. Component ports represent instances of data message types. These types are defined as structures with individual data fields to which Simulink data ports can be mapped. These relations describe the marshaling, demarshaling, and transfer of data between software components.

4.1.2. Logical Software Architecture. The second step in the design process, denoted as ② in Figure 2, involves the specification of the logical software architecture. The logical software architecture model describes the interconnection of component instances representing the functional blocks. The logical software architecture captures the data dependencies between software components independent of their distribution over different processors. The semantics of the logical interconnections are defined by task-local synchronous function invocations as well as message transfers between tasks based on the time-triggered communication paradigm.

4.1.3. Hardware Platform. The third step in the design process, depicted as ③ in Figure 2, involves the definition of the hardware platform on which the controller software is deployed. In this step, by specifying the attributes of the hardware platform, we clearly define the components and interactions of the platform which significantly impacts the behavior of the overall control system. ESMoL's network/platform sublanguage has several components including processing nodes and communication networks for defining the computing nodes as well as the underlying communication networks. Several specific networks for automotive systems are defined in this sublanguage, such as CAN bus and TTEthernet. In this paper, we consider TTEthernet, which is based on the time-triggered paradigm. In ESMoL, hardware platforms are defined hierarchically as hardware units with ports for interconnections. The model attributes for hardware platform also capture timing resolution, overhead parameters for data transfers, task context switching times, and scheduling policies.

4.1.4. Deployment Model. The fourth step in the design process, denoted as ④ in Figure 2, involves the definition of the deployment model. The deployment model represents the mapping of software components to processing nodes and data messages to communication ports. This model captures the assignment of component instances as periodic tasks running on a particular processor. A well-defined specification of the deployment model is important as the overall behavior of the control application depends on the efficient mapping of software components to the designated platforms. In ESMoL, a task executes on a processing node at a single periodic rate, and all components within the task execute synchronously. Message ports on component instances are assigned to hardware interface ports in the model to define the media through which messages are transferred.

4.1.5. Timing Model. The fifth step, denoted as ⑤ in Figure 2, involves the specification of the timing behavior of the system. The timing model allows a designer to specify component execution constraints involving the timing behavior of the component. The specification of the timing model is very important in order to ensure the predictability of the overall system behavior. In ESMoL, the timing model of a control application is established by attaching timing parameter blocks to components and messages. There are three types of timing parameter blocks in ESMoL to represent three different execution modes. Time-triggered execution information (*TTExecInfo*) is used to specify the timing for a task or a message transfer that executes based on a synchronized time base, such as in time-triggered distributed system. If the synchronized time base is not available or used, an event-triggered system needs to be specified. In this case, asynchronous periodic execution information (*AsyncPeriodicExecInfo*) is used for periodic execution, while sporadic execution information (*SporadicExecInfo*) is used for aperiodic execution with a minimum period. The model also indicates which components and messages that will be scheduled independently, and those that should be grouped into a single task or message object. In the case of processor-local message transfers, transfer time is neglected as reads and writes occur in locally shared memory.

4.1.6. Model Transformation. In order to integrate analysis tools and other code generators into ESMoL, rather than directly attaching translators directly to the user language, ESMoL defines a simpler abstract intermediate language whose elements are similar to those of the user language. The sixth step in the design process, depicted as ⑥ in Figure 2, involves this model transformation. An ESMoL interpreter called Stage 1 is used to perform the transformation from the originally defined ESMoL model into an abstract intermediate language that contains explicit relation objects that represent relationships implied by structures in ESMoL. This translation is similar to the way a compiler translates concrete syntax first to an abstract syntax tree and then to intermediate semantic representations suitable for optimization. Stage 1 is implemented using the Universal Data Model (UDM) navigation application interface [29]. The ESMoL_Abstract

target model is a flattened ESMoL model and the source for the transformations for further analysis and software component generations.

4.1.7. Network/Task Scheduling. This step in the design process, depicted as ⑦ in Figure 2, involves network and task scheduling. A scheduler provided by TTTech [30] is used for network scheduling. This scheduler requires a configuration script of the network/hardware platform in order to perform analysis. An ESMoL_Abstract interpreter called Stage 2 is integrated into the ESMoL's abstract intermediate language to generate the TTEthernet configuration script for network scheduling. This interpreter takes the parameters specified in the TTEthernet components of the network/platform model and combines them with message specifications generated for interprocessor message transfers, which can be deduced from the software architecture model and the deployment model. The desired offset fields of the TT messages are obtained from the timing model.

For the task scheduling, we use the bottom-level-based heuristic scheduling algorithm [31]. The algorithm establishes the critical path of the task graph, which needs at least the execution time of any other path in the task graph. In order to distinguish the tasks on the critical path, the notion of bottom-level of a task is used, which is the length of the longest path starting with this task. Because the bottom-level bounds the start time of a task, as-late-as-possible start time of a task can be used to generate the task schedule.

4.1.8. Software Implementation. The final step in the design process, depicted as ⑧ in Figure 2, is the software implementation of the control software. The network schedule from the previous design step is used by a tool called ^{TTE}Build from TTTech to generate the binary configuration files and C code configurations required for the implementation of communication on the platform. An integrated interpreter in ESMoL's abstract intermediate language assembles the C code files generated by RTW and ^{TTE}Build with glue code files and automatically generates a Makefile. After compilation, the executables are deployed onto the respective ECUs.

The proposed software development process and toolchain provides flexibility and convenience in the rapid prototyping of automotive control software while at the same time ensuring that the models are correct at the different stages of design. In the design of automotive control application, it is typical to test multiple configurations and refinements in multiple iteration. Hence, the model-based approach provides the ability to quickly modify models and parameters to reflect changes and subsequently generate deployable software components for testing on the platform.

5. System Architecture

Figure 3 shows the system architecture for the experimental platform used in the evaluation of the automotive software development process.

5.1. Physical Layer. The physical layer modeling the dynamics of the automotive system encompasses two main components as described as follows.

5.1.1. Design/Visualization PC. The design/visualization PC represents the computing platform, running Windows operating system, for the dynamic modeling of a vehicle using CarSim as well as the initial control design and testing using Matlab/Simulink. CarSim is a commercially available parameter-based vehicle dynamics modeling software. It facilitates the efficient simulation and analysis of the behavior of four-wheeled vehicles in response to various inputs such as steering, braking, and acceleration [32]. The design/visualization PC is also used for the visualization and reporting of results from various experiments.

5.1.2. Target PC. The Target PC is a National Instruments LabVIEW Real-Time target running NI's Real-Time Module which provides a complete solution for creating reliable, stand-alone real-time systems [33]. In the experimental platform, the vehicle's physical dynamics modeled in CarSim is deployed on the Target PC during experiments. The Target PC is also integrated with a TTTech PCIe-XMC card [30] which enables the seamless integration and communication with ECUs on the time-triggered network supported by the TTEthernet switch.

5.2. Platform Layer. The platform layer is modeled by the TTEthernet switch and ECUs. These components are described as follows.

5.2.1. TTEthernet Development Switch. The TTEthernet Development Switch is an 8-port 100 Mbps system which supports 100 Base-TX Ethernet and enables hard real-time communication based on the TTEthernet protocol. It supports a star network topology. In Figure 3, the end systems comprised of the ECUs and the XMC card communicate with each other through the switch. The switch operates based on user defined configurations based on an experimental scenario. The configurations are specified in our model-based tool, ESMoL.

5.2.2. Electronic Control Units (ECUs). In Figure 3, the network depicts four ECUs, but there could possibly be more or fewer number of ECUs connected at a time based on a specific configuration. In our framework, an ECU is an IBX-530W-ATOM box with an Intel Atom CPU running a Real-Time Linux (RT-Linux) operating system. Each ECU is integrated with a TTEthernet Linux driver using an implementation of the TTEthernet protocol to enable the communication with other end systems in the TTEthernet network. Controller software components are deployed on the ECUs for the execution of automotive control applications. The controller software components that are deployed on each ECU are generated from the software design process for the controller specified in ESMoL. These software components execute in kernel space of the RT-Linux running on each of the ECUs and utilizes the synchronized time base of TTEthernet.

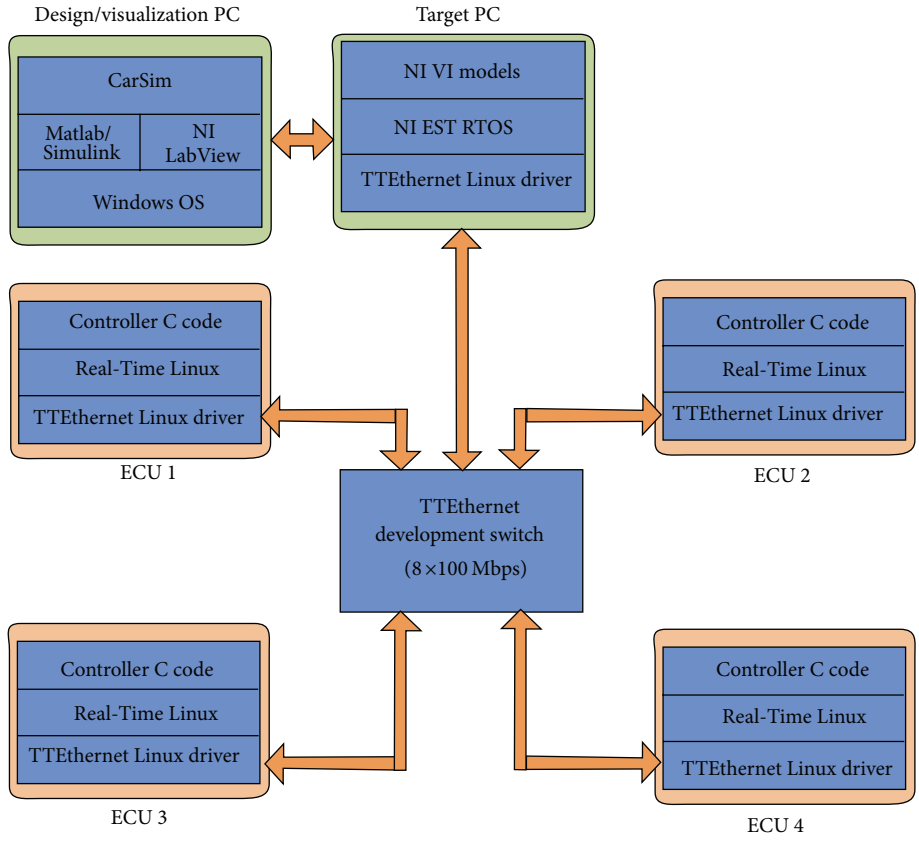


Figure 3: System Architecture for the Experimental Platform.

6. Control Design

In this section, we describe the controller design for the ACC. The operation of an ACC involves the use of a radar system which is attached to the front of the vehicle in order to detect when a vehicle is in the ACC-equipped vehicle's detectable view. When a vehicle is detected by the radar, the ACC system will control the distance between the ACC-enabled vehicle and the leading vehicle. In the absence of a leading vehicle in the ACC-enabled vehicle's path, the ACC system controls the vehicle to maintain a driver set velocity, essentially behaving like the conventional cruise control system.

The vehicle model used for the ACC design only considers the longitudinal motion of the vehicle.

6.1. Longitudinal Vehicle Model. The longitudinal vehicle model is typically based on the following assumptions [17].

A1: The torque converter is locked which implies that the engine speed is algebraically proportional to the vehicle speed via the gear ratios. *A2*: The tire slip is negligible.

The longitudinal dynamics of a vehicle can be described by the following equation provided that assumptions *A1* and *A2* hold:

$$T_e - R_g\left(T_b + M_{\mathrm{rr}} + hF_a + mgh\sin\theta\right) = \beta a, \tag{1}$$

where

$$\beta = \frac{\left[J_e + R_g^2\left(J_{wr} + Jwf + mh^2\right)\right]}{R_g h},$$
$$F_a = C_a v^2. \tag{2}$$

Table 1 provides a summary of the parameter definitions. Figure 4 shows a block diagram of the ACC system. The ACC is hierarchically divided into two levels of control: the upper level controller and the low level controller.

6.1.1. Upper Level Controller. The main functionality of the upper level controller is to compute the desired acceleration for the ACC-equipped vehicle that achieves the desired spacing or velocity. As depicted in Figure 4, the upper level controller, using the driver inputs, the radar measurements, and the current distance and velocity of the ACC-equipped vehicle relative to a leading vehicle, computes the desired acceleration that is required to achieve the desired spacing or velocity. The computed acceleration command is sent to the lower level controller to compute and implement the corresponding actuation commands as needed. The upper level controller can operate in two main control modes.

(a) *Velocity Control*: In this mode, the radar does not detect any vehicle in the path of the ACC-equipped vehicle. In this mode, the ACC essentially acts like the

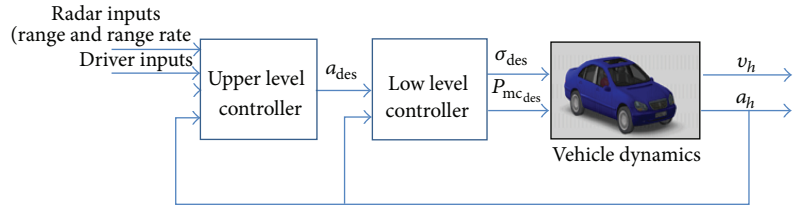


Figure 4: Adaptive cruise control system.

Table 1: Parameters for the longitudinal vehicle dynamics.

Parameter	Definition
β	Lumped inertia
F_a	Aerodynamic drag force
T_e	Net engine torque
T_b	Brake torque
R_g	Gear ratio
M_{rr}	Rolling resistance moments
h	Effective wheel radius
m	Total curb mass of the vehicle
J_e	Inertia of engine
J_{wr}	Inertia of rear axle
J_{wf}	Inertia of front axle
C_a	Aerodynamic drag coefficient
v	Velocity of the vehicle
a	Acceleration of vehicle
θ	Inclination angle of the road
g	Gravitational acceleration

conventional cruise controller. Therefore, the ACC-equipped vehicle's velocity is maintained at the target velocity set by the driver. The control law for computing the acceleration command is a proportional controller defined as.

$$a_{\text{des}} = K_1 * (v_d - v_h), \tag{3}$$

where K_1 is a control gain, v_d is the user-set velocity, and v_h is the velocity of the host or ACC-equipped vehicle.

(b) *Spacing Control*: The spacing control mode is entered when the radar detects a leading vehicle in the ACC-equipped vehicle's path, and the ACC system controls the vehicle to maintain a desired distance based on the velocity of the host vehicle and a user-specified time gap. This desired distance, S_d, can be defined as

$$S_d = \Delta + (v_h * t_{\text{gap}}), \tag{4}$$

where Δ is the desired distance to be maintained in the case where the leading vehicle comes to a complete stop, and t_{gap} is the user-specified time gap with typical values in the range of about 0.7–1.8 seconds.

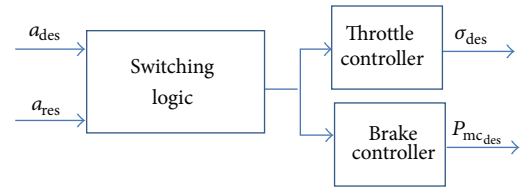


Figure 5: Low level controller.

The control law used in computing the desired acceleration in this mode is

$$a_{\text{des}} = \min(a_1, a_2), \tag{5}$$

where a_1 is computed similar to the desired acceleration in (3), and a_2 is computed as follows

$$a_2 = K_2 * (v_l - v) + K_3 (S_d - S_a), \tag{6}$$

where S_a is the gap distance measured by the radar, v_l is the velocity of the leading vehicle, K_2, and K_3 are control gains.

6.1.2. Low Level Controller. The main objective of the low level controller is twofold. First, using the desired acceleration command from the upper level controller, the lower level controller determines whether to apply braking control or throttle control. Secondly, the required control command is applied to the vehicle in order to achieve the desired acceleration. The applied control command is either throttle angle command, σ_{des}, or master cylinder pressure command, $P_{mc\,\text{des}}$.

(a) *Switching Logic*: The switching logic component shown in Figure 5 determines, based on the desired acceleration from the upper level controller, whether a brake torque or engine torque is required to achieve the desired acceleration. Typically, it is common to assume that a simple logic for choosing between brake and engine control can be based on the sign of the desired acceleration; that is, if the acceleration is greater than or equal to zero, then engine control should be applied, otherwise the brake control should be applied. This approach neglects the fact that with no control inputs, the engine torque is not necessarily zero. Thus, a better alternative is to consider the residual acceleration, a_{res}, due to the presence of engine torque when no control inputs are

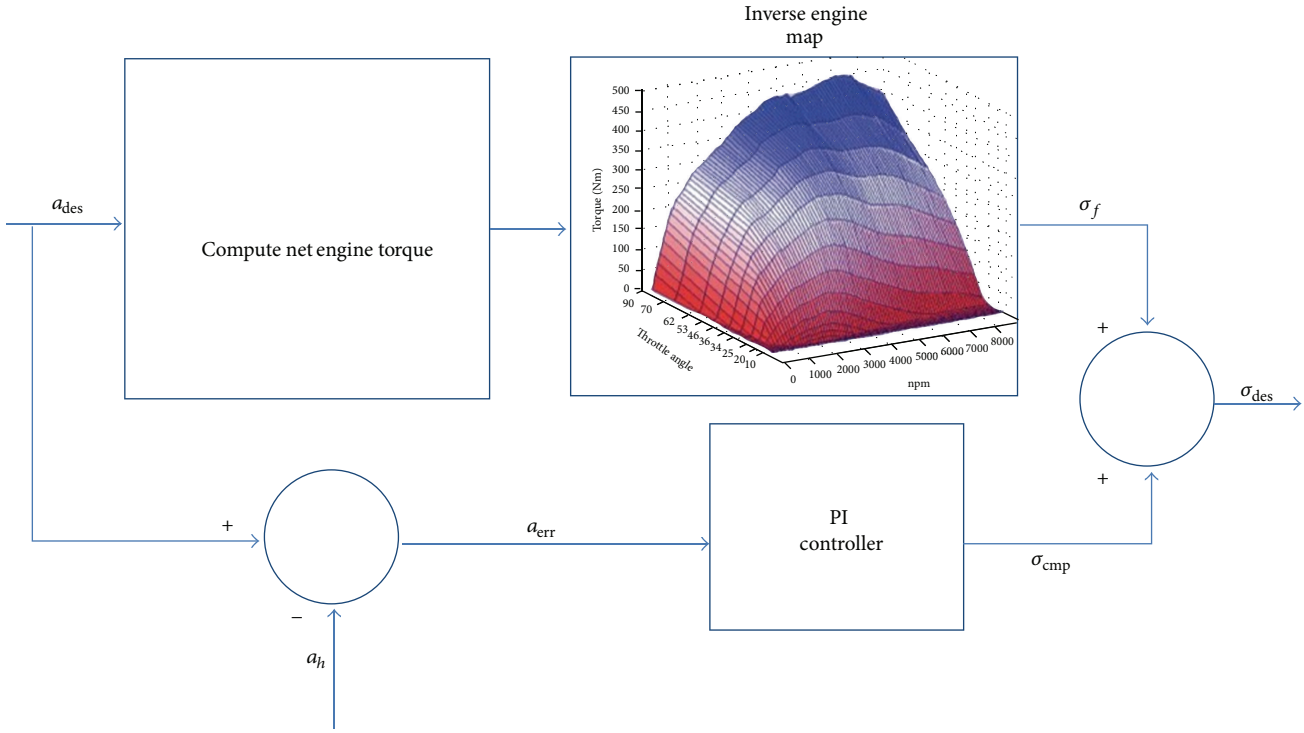


Figure 6: Throttle controller.

introduced [16]. Based on this approach, the engine torque can be subdivided into two parts: minimum or closed throttle torque, T_{ect}, and the portion subject to control, T_{ec}. Substituting these two components for T_e in (1) we have

$$T_{\text{ec}} + T_{\text{ect}} - R_g\left(T_b + M_{\text{rr}} + hF_a + mgh\sin\theta\right) = \beta a. \tag{7}$$

In the absence of control inputs $T_{\text{ec}} = T_b = 0$, the residual acceleration, a_{res}, as a result of closed-throttle torque, can be obtained as

$$a_{\text{res}} = \frac{1}{\beta}\left[T_{\text{ect}} - R_g\left(T_b + M_{\text{rr}} + hF_a + mgh\sin\theta\right)\right]. \tag{8}$$

Once the residual acceleration is calculated, the switching law uses the following criteria to determine whether engine or braking is required:

$$\begin{aligned} a_{\text{des}} \geq a_{\text{res}} &\Longrightarrow \text{throttle control},\\ a_{\text{des}} < a_{\text{res}} &\Longrightarrow \text{brake control}. \end{aligned} \tag{9}$$

In order to prevent rapid chattering between the engine control and brake control models, a small hysteresis, h_{yst}, is introduced. This results in the following switching law:

$$\begin{aligned} a_{\text{des}} \geq a_{\text{res}} + h_{\text{yst}} &\Longrightarrow \text{throttle control},\\ a_{\text{des}} < a_{\text{res}} - h_{\text{yst}} &\Longrightarrow \text{brake control}. \end{aligned} \tag{10}$$

Once the decision of the control mode is determined, the corresponding controller converts the desired acceleration into the appropriate input to the vehicle.

(b) *Throttle Control*: When engine control torque is required, the throttle controller converts the computed desired acceleration into a throttle command that is required to achieve the acceleration. Figure 6 shows the block diagram for the throttle control law. The controller first converts desired acceleration into a desired engine net torque. The desired net torque can be computed based on (1) with $T_b = 0$ as follows:

$$T_{e\,\text{des}} = \beta a_{\text{des}} + R_g\left(T_b + M_{\text{rr}} + hF_a + mgh\sin\theta\right). \tag{11}$$

The computed desired torque is converted into a throttle angle command by using an inverse engine map for the vehicle based on the current engine speed, w_e. This is performed by interpolating the data from an experimentally determined engine map lookup table for the vehicle. Consider

$$T_{e\,\text{des}} \Longrightarrow \text{inverse engine map} \Longrightarrow \sigma_f. \tag{12}$$

Due to the potential inaccuracies from the obtained engine map, an additional proportional-integral (PI) controller is integrated in the throttle controller to ensure that the desired acceleration is achieved.

(c) *Brake Control*: Figure 7 shows the brake controller. When braking control torque is required, the brake controller converts the desired acceleration to an

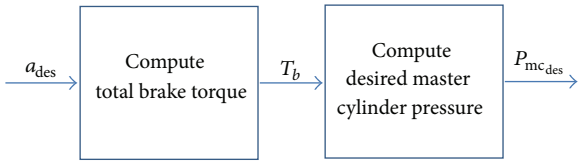


Figure 7: Brake controller.

appropriate brake command. The controller first computes the desired brake torque from the desired acceleration by using (7) and with $T_{\text{ec}} = 0$, which results in the following equation:

$$T_{\text{ect}} - R_g \left(T_{b\,\text{des}} + M_{\text{rr}} + hF_a + mgh \sin\theta\right) = \beta a_{\text{des}}. \tag{13}$$

The computed desired brake torque is then converted to an equivalent master cylinder pressure which is applied as the control input to the vehicle. The master cylinder pressure, P_{mc}, can be related to the brake torque by the following equation:

$$T_b = K_b \left(P_{\text{mc}} - P_{\text{po}}\right), \tag{14}$$

where K_b is a brake gain, and P_{po} is the push-out pressure required to engage brake.

7. ACC Controller Software Design and Implementation

This section describes the ACC controller software design and implementation based on the proposed development process. The initial ACC controller design is performed in Matlab/Simulink. Subsequently, following the automated software development process described in Section 4, the Matlab/Simulink model is imported into the ESMoL environment.

Figure 8 shows the logical software architecture depicting the logical interconnections of ACC controller components. In this model, each component represents a task. The ACC controller has four tasks, Instrument Cluster Sense (*InstrClstrSens*), Instrument Cluster Actuate (*InstrClstrAct*), Upper Level Controller (*UpperLevelController*), and Low Level Controller (*LowLevelController*). The *InstrClstrSens* and *InstrClstrAct* correspond to the tasks for processing the inputs and outputs of the ACC controller, respectively, while *UpperLevelController* and *LowLevelController* tasks implement the functionality of the controller designed in Section 6. Two tasks, *InputHandler* and *OutputHandler*, are used to represent the sensing and actuation for the vehicle dynamics developed in CarSim. In addition to the ACC controller, we also introduced another task, *RearViewMonitor*, in order to emulate the execution of other tasks on the ECUs.

In Figure 9, the network/platform configurations are explicitly modeled in the ESMoL. Three ECUs are specified and are denoted as ECU1, ECU2, and ECU3. The RT-Target node represents the Target-PC or computing node where the CarSim vehicle dynamics is executed. In order to represent the sensors and actuators of a vehicle, two virtual I/O devices are used. For the networked system, specific parameters for TTEthernet need to be defined, such as hyperperiod, bandwidth, time slot size, clock synchronization cycle, and synchronization precision. These specified parameters are used to generate the TTEthernet configuration script in ESMoL.

Figure 10 shows the deployment model of ACC control software. Dashed arrows represent assignment of components to their respective processors, and solid lines represent assignment of message instances (component ports) to communication channels (port objects) on the processor. We manually deploy *InstrClstrSens* and *InstrClstrAct* on ECU1, *UpperLevelController* on ECU2, and *LowLevelController* and *RearViewMonitor* on ECU3.

In Figure 11, the timing and execution model for tasks and message transfers of the ACC control system are shown. The ACC controller runs at a period of 10 ms. Since the TTEthernet provides a synchronized time base for communication, all the message transfers are attached with *TTExecInfo* to indicate time-triggered communication. For example, *ULMExec* is used to specify the timing for the message transfer from *InstrClstrSens* to *UpperLevelController*. We set the following parameters: the execution period which is the hyperperiod of the TTEthernet configuration, the desired offset which is used to specify the *initstart_ns* field of TT message in the generated TTEthernet configuration script, and the worst case duration which is the worst case communication time of the TTEthernet. The TTEthernet driver on each ECU has a scheduler that utilizes the synchronized time base, which can invoke the tasks according to a static schedule. Thus, all the tasks are executed according to the time-triggered paradigm. We specify the *TTExecInfo* for each task. For instance, *ULExec* specifies the execution time of *UpperLevelController* in the 10 ms period. Before scheduling, we only need to provide the execution period and the task's worst case execution time, which is determined empirically.

Using the Stage 1 interpreter as described in Section 4, the ESMoL model is transformed to an ESMoL_Abstract model in the form of XML file. A Stage 2 interpreter is used to generate the TTEthernet network configuration for scheduling and task scheduling. In this case of task scheduling, the critical path is simple as follows: *InputHandler* $\mapsto$ *InstrClstrSens* $\mapsto$ *UpperLevelController* $\mapsto$ *LowLevelController* $\mapsto$ *InstrClstrAct*. After network/task scheduling, the schedule information is updated into the ESMoL and ESMoL_Abstract models automatically. The integrated interpreter then uses the updated ESMoL_Abstract model to assemble all the codes for compilation.

8. Experimental Evaluation

In this section, we present the experimental results from the testing of the ACC system on the experimental platform. The experiments consist of two vehicles, a leading vehicle and an ACC-equipped vehicle which we refer to as the host vehicle. When the ACC feature is enabled and engaged, the host vehicle starts in the velocity control mode and

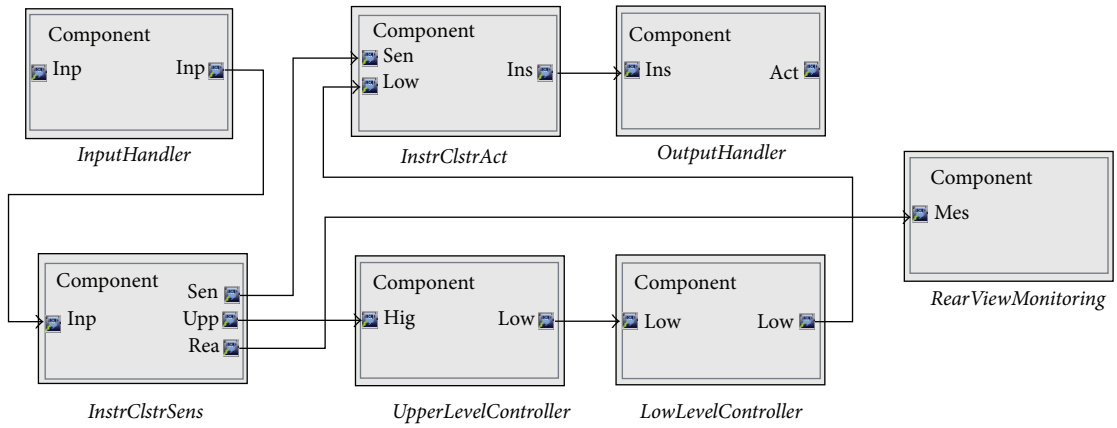


Figure 8: Logical software architecture of ACC controller.

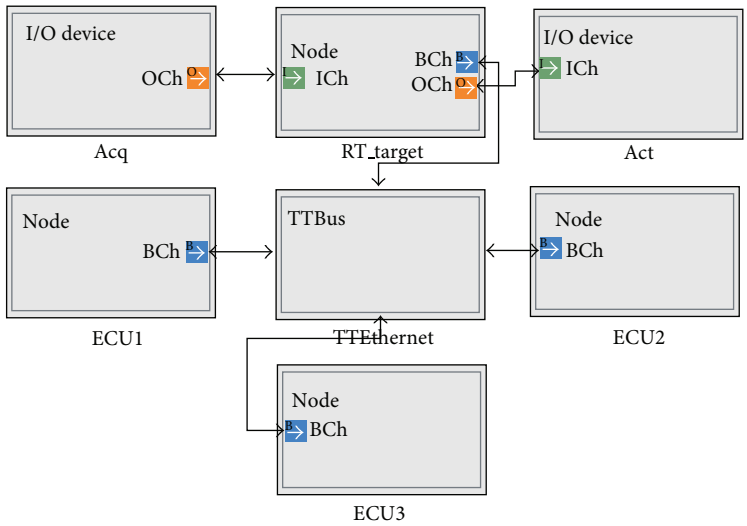


Figure 9: Hardware platform representation of the experimental platform.

Table 2: Parameters for the ACC experiments.

Parameter	Value
K_1	0.5
K_2	2
K_3	0.7
Range of ACC radar	100 m
t_{gap}	1.5 s
Δ	10 m
θ	0 rad
Sampling period	0.01 s
m	1650 kg

maintains a driver-set velocity, and when a leading vehicle is detected, the ACC transitions into the spacing control mode in order to maintain a desired distance based on the driver-set time gap and host velocity as described in Section 6. The parameters for the experiments are provided in Table 2.

The experimental setup is based on the system architecture described in Section 5. The generated software components for the ACC are distributed over three ECUs as described in Section 4. In this experiment, the velocity of the leading vehicle starts at an initial value of 60 km/h. The initial global longitudinal positions of the leading vehicle and the host vehicle are 130 m and 0 m, respectively, which means that the host vehicle radar is initially out of range. The host vehicle initially starts at an initial velocity of 65 km/h with a driver set target velocity of 80 km/h. We present four scenarios based on the driving behavior of the leading vehicle. In addition, we compare the results obtained during the control design stage using Matlab/Simulink with those obtained from the experimental platform. The modeled scenarios are described as follows.

(1) *Scenario 1*: *Velocity Control*. In this scenario, there is no leading vehicle in front of the host vehicle within the range of the radar, and, hence, the host vehicle is under the velocity control mode or the conventional

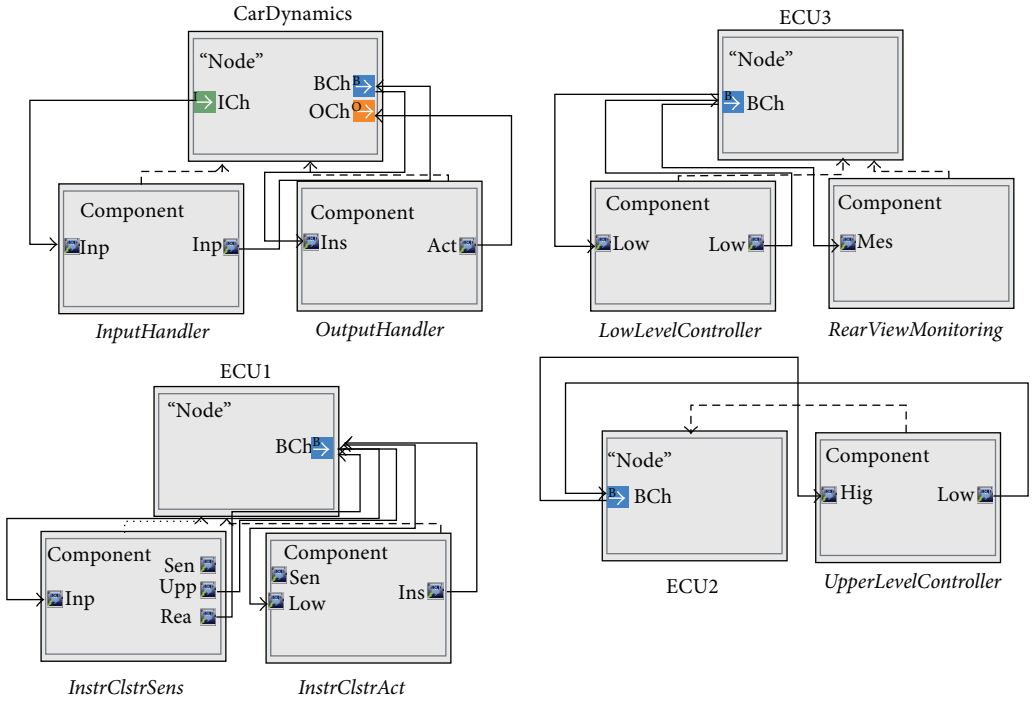


Figure 10: Platform deployment aspect of ACC controller.

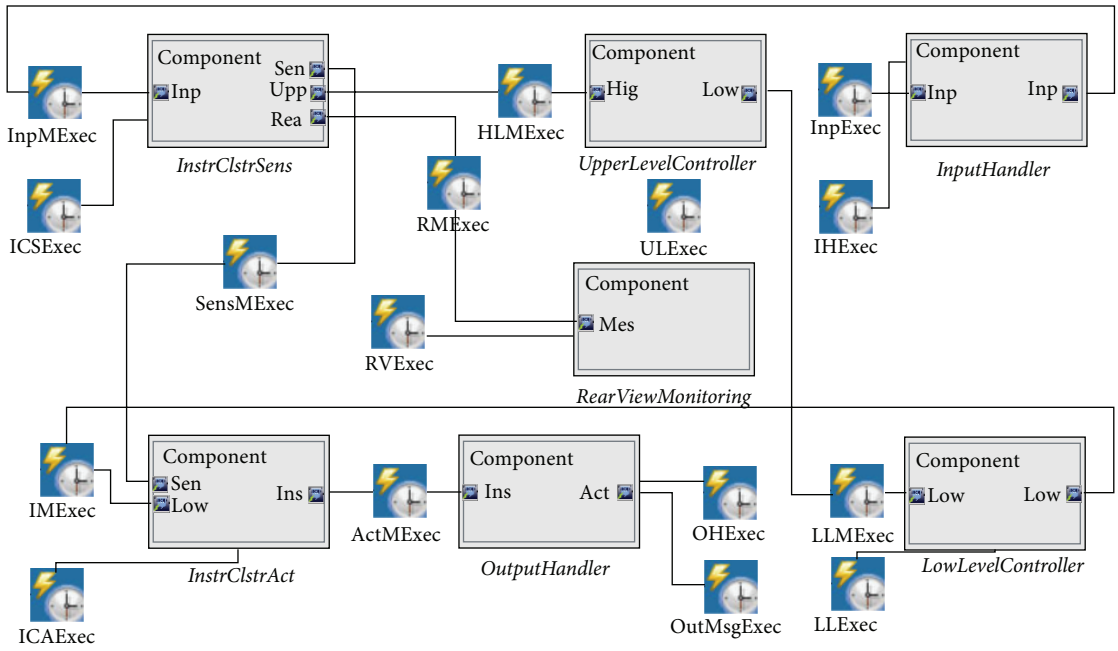
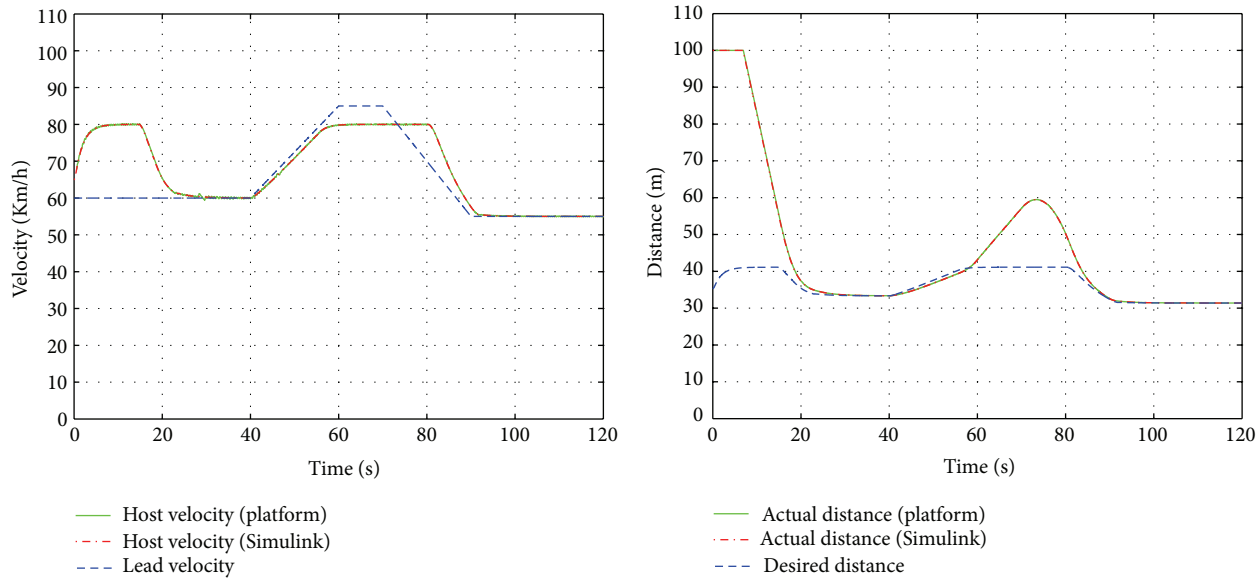


Figure 11: Timing/execution model of ACC controller.

cruise control. Figure 12 shows the results from the control design phase using Matlab/Simulink and the results from the execution on the experimental platform, respectively. This scenario can be observed between the time segment of 0–10 s.

(2) *Scenario 2*: *Spacing Control*. In this scenario, the radar detects a slower leading vehicle and transitions to the spacing control mode to control the distance between the two vehicles to a driver-set time gap. The desired gap distance is attained when the two vehicles travel at the same velocity. This scenario can be observed between the time segment 10–40 s in Figure 12.

(3) *Scenario 3*: *Leading Vehicle Speeds Up*. In this scenario, while in spacing control mode, the leading vehicle



(a) Velocity plots

(b) Gap distance plots

Figure 12: Gap Distance and Velocities.

begins to speed up. As a result, the velocity of the host vehicle also increases in order to maintain a desired velocity. This scenario can be observed between the time segment 40–60 s in Figure 12. From the plots, when the leading vehicle reaches and a maintains a velocity of 85 km/h after 60 s, the host vehicle maintains it is velocity at the driver-set velocity of 80 km/h, since the driver-set velocity is the maximum achievable velocity of the host vehicle based on the ACC algorithm. It can be seen from the distance plots that the distance between the two vehicles increases due to the difference in velocity of the vehicles.

(4) *Scenario 4*: *Leading Vehicle Slows Down*. In this scenario, the leading vehicle slows down, and as a result, the host vehicle also starts to decrease its velocity in order to maintain the desired spacing between the vehicles. This scenario can be observed between the time segment 70–90 s in Figure 12. At approximately 105 s, the two vehicles starts to travel at the same velocity again.

To highlight the importance of the experimental platform for the early assessment for control software before actual deployment on a real vehicle, we compared the results obtained from running the scenarios in Matlab/Simulink to those obtained from running the scenarios on experimental platform. This comparison provides insight to potential impacts of the deployment of the control software on a prototype platform. Figure 13 shows the velocity plots from the simulation in Matlab/Simulink from the control design stage and the results obtained from deploying the resulting software components on the platform. By zooming in on the velocity plots between 25 and 45 s, we notice that compared

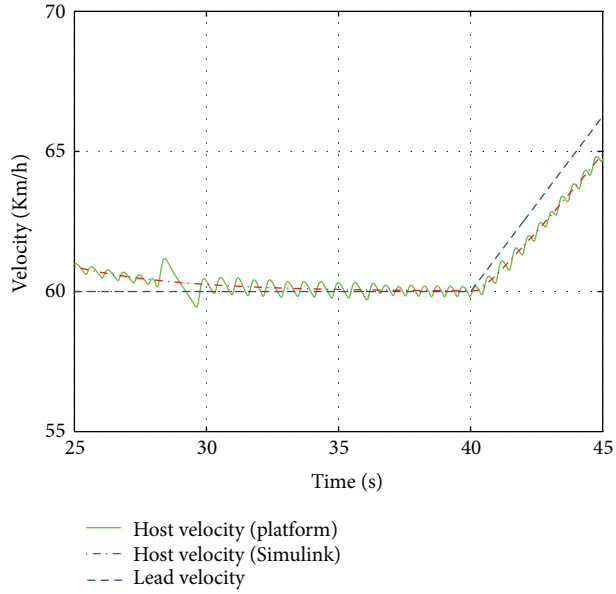


Figure 13: Comparison of Simulink simulation results and results from the experimental platform.

to the Simulink results, the results from the deployed software exhibit some oscillations with an amplitude 0.6 km/h. Although this is barely noticeable, it is important to note the difference in the results which can be attributed to platform effects as a result of deploying the controller on the platform. This implementation limitation is due to the fact that the computation on the RT-Target is not synchronized with the communication.

9. Discussion and Conclusion

Our proposed framework addresses the complex interactions that emerge as a result of integrating the various design layers of CPS. The proposed approach provides an end-to-end methodology and a toolchain for the development of automotive control software. The toolchain is based on simple and visually intuitive formal languages that restrict component interactions in a well-defined manner in order to ensure a "correct-by-construction" design of automotive control software. We employ an experimental platform based on the time-triggered paradigm in order to facilitate the design, deployment, and hardware-in-the-loop simulation of automotive control software. We applied the design methodology to a case study on the adaptive cruise control and presented the experimental results based on realistic scenarios.

In regards to future work, we would like to study the possible interactions between multiple automotive control systems deployed together and formally define their interacting behavior as well as the impact on the overall system. This interacting behavior is important because there are possible cases where these systems can have conflicting objectives, and, hence, a clear understanding of their underlying interaction is crucial.

Conflict of Interests

The authors do not have any conflict of interests with any of the commercial identities mentioned in this paper.

Acknowledgments

The authors would like to thank Shige Wang for his insightful suggestions and discussions. This work is supported in part by the National Science Foundation (CNS-1035655 and CCF-0820088), US Army Research Office (ARO W911NF-10-1-0005), and Lockheed Martin. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the US Government.

References

[1] J. Mossinger, "An insight into the hardware and software complexity of ecus in vehicles," in *Advances in Computing and Information Technology*, vol. 198 of *Communications in Computer and Information Science*, pp. 99–106, 2011.

[2] J. Mossinger, "Software in automotive systems," *IEEE Software*, vol. 27, no. 2, pp. 92–94, 2010.

[3] A. Michailidis, U. Spieth, T. Ringler, B. Hedenetz, and S. Kowalewski, "Test front loading in early stages of automotive software development based on AUTOSAR," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '10)*, pp. 435–440, March 2010.

[4] A. Sangiovanni-Vincentelli, "Electronic-system design in the automobile industry," *IEEE Micro*, vol. 23, no. 3, pp. 8–18, 2003.

[5] J. A. Cook, I. V. Kolmanovsky, D. McNamara, E. C. Nelson, and K. V. Prasad, "Control, computing and communications: technologies for the twenty-first century model T," *Proceedings of the IEEE*, vol. 95, no. 2, pp. 334–355, 2007.

[6] A. Sangiovanni-Vincentelli and M. Di Natale, "Embedded system design for automotive applications," *Computer*, vol. 40, no. 10, pp. 42–51, 2007.

[7] T. Stahl and M. Volter, *Model-Driven Software Development*, John Wiley and Sons, 2006.

[8] J. Porter, G. Hemingway, H. Nine et al., "The esmol language and tools for high-confidence distributed control systems design—part 1: language, framework, and analysis," Tech. Rep. ISIS-10-109, Vanderbilt University, 2010.

[9] A. Ledeczi, M. Maroti, A. Bakay et al., "The generic modeling environment," in *Proceedings of the IEEE Workshop on Intelligent Signal Processing (WISP '01)*, May 2001.

[10] H. Kopetz and G. Bauer, "The time-triggered architecture," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 112–126, 2003.

[11] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert, "Trends in automotive communication systems," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1204–1222, 2005.

[12] M. Broy, S. Chakraborty, D. Goswami et al., "Cross-layer analysis, testing and verification of automotive control software," in *Proceedings of the 9th ACM International Conference on Embedded Software (EMSOFT '11)*, pp. 263–272, 2011.

[13] U. Drolia, Z. Wang, Y. Pant, and R. Mangharam, "AutoPlug: an automotive test-bed for electronic controller unit testing and verification," in *Proceedings of the 14th IEEE International Conference on Intelligent Transportation Systems (ITSC '11)*, pp. 1187–1192, 2011.

[14] W. Hu, M. Wang, and Y. Lin, "On the software-based development and verification of automotive control systems," in *Proceedings of the 33rd IEEE Annual Conference of the Industrial Electronics Society (IECON '07)*, pp. 857–862, 2007.

[15] A. Ray, I. Morschhaeuser, C. Ackermann, R. Cleaveland, C. Shelton, and C. Martin, "Validating automotive control software using instrumentation-based verification," in *Proceedings of the 24th IEEE/ACM International Conference on Automated Software Engineering (ASE '09)*, pp. 15–25, November 2009.

[16] J. C. Gerdes and J. K. Hedrick, "Vehicle speed and spacing control via coordinated throttle and brake actuation," *Control Engineering Practice*, vol. 5, no. 11, pp. 1607–1614, 1997.

[17] J. K. Hedrick and P. P. Yip, "Multiple sliding surface control: theory and application," *Journal of Dynamic Systems, Measurement, and Control*, vol. 122, no. 4, pp. 586–593, 2000.

[18] P. A. Ioannou and C. C. Chien, "Autonomous intelligent cruise control," *IEEE Transactions on Vehicular Technology*, vol. 42, no. 4, pp. 657–672, 1993.

[19] K. Yi, S. Lee, and Y. D. Kwon, "An investigation of intelligent cruise control laws for passenger vehicles," *Journal of Automobile Engineering*, vol. 215, no. 2, pp. 159–169, 2001.

[20] P. S. Fancher, H. Peng, and Z. Bareket, "Comparative analyses of three types of headway control systems for heavy commercial vehicles," *Vehicle System Dynamics*, vol. 25, no. 1, pp. 139–151, 1996.

[21] B. A. Güvenç and E. Kural, "Adaptive cruise control simulator: a low-cost, multiple-driver-in-the-loop simulator," *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 42–55, 2006.

[22] K. Breuer and M. Weilkes, "A versatile test vehicle for acc-systems and components," Tech. Rep., 1999.

[23] A. R. Girard, S. Spry, and J. K. Hedrick, "Intelligent cruise-control applications: real-time, embedded hybrid control software," *IEEE Robotics and Automation Magazine*, vol. 12, no. 1, pp. 22–28, 2005.

[24] E. Armengaud, A. Tengg, M. Driussi, M. Karner, C. Steger, and R. Weiß, "Automotive software architecture: migration challenges from an event-triggered to a time-triggered communication scheme," in *Proceedings of the 7th Workshop on Intelligent Solutions in Embedded Systems (WISES '09)*, pp. 95–103, June 2009.

[25] J. Sztipanovits, X. Koutsoukos, G. Karsai et al., "Toward a science of cyber-physical system integration," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 29–44, 2012.

[26] J. Eker, J. W. Janneck, E. A. Lee et al., "Taming heterogeneity—the ptolemy approach," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 127–143, 2003.

[27] MathWorks—real-time workshop, http://www.mathworks.com/products/rtw/.

[28] B. Chou and T. Erkkinen, "Converting models from floating point to fixed point for production code generation," Matlab Digest, November 2008.

[29] E. Magyari, A. Bakay, A. Lang et al., "Udm: an infrastructure for implementing domain-specific modeling languages," in *Proceedings of the 3rd OOPSLA Workshop on Domain-Specific Modeling*, Anaheim, Calif, USA, 2003.

[30] Ttethernet, http://www.tttech.com/en/products/ttethernet/.

[31] O. Sinnen, *Task Scheduling for Parallel Systems*, Wiley-Interscience, New York, NY, USA, 2007.

[32] Carsim, http://www.carsim.com/.

[33] National Instruments, http://www.ni.com/.

Hindawi Publishing Corporation
Journal of Control Science and Engineering
Volume 2013, Article ID 719683, 12 pages
http://dx.doi.org/10.1155/2013/719683

Research Article

A Real-Time Embedded Control System for Electro-Fused Magnesia Furnace

Fang Zheng, Yang Jie, Tao Shifei, Wu Zhiwei, and Chai Tianyou

State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110819, China

Correspondence should be addressed to Fang Zheng; fangzheng@mail.neu.edu.cn

Received 8 September 2012; Accepted 18 December 2012

Academic Editor: Sabri Cetinkunt

Since smelting process of electro-fused magnesia furnace is a complicated process which has characteristics like complex operation conditions, strong nonlinearities, and strong couplings, traditional linear controller cannot control it very well. Advanced intelligent control strategy is a good solution to this kind of industrial process. However, advanced intelligent control strategy always involves huge programming task and hard debugging and maintaining problems. In this paper, a real-time embedded control system is proposed for the process control of electro-fused magnesia furnace based on intelligent control strategy and model-based design technology. As for hardware, an embedded controller based on an industrial Single Board Computer (SBC) is developed to meet industrial field environment demands. As for software, a Linux based on Real-Time Application Interface (RTAI) is used as the real-time kernel of the controller to improve its real-time performance. The embedded software platform is also modified to support generating embedded code automatically from Simulink/Stateflow models. Based on the proposed embedded control system, the intelligent embedded control software of electro-fused magnesium furnace can be directly generated from Simulink/Stateflow models. To validate the effectiveness of the proposed embedded control system, hardware-in-the-loop (HIL) and industrial field experiments are both implemented. Experiments results show that the embedded control system works very well in both laboratory and industry environments.

1. Introduction

Fused magnesia is an important and widely used refractory and raw material for many industries, which has lots of merits such as high melting point, antioxidation, structural integrity, and strong insulating features [1]. Nowadays, high-purity fused magnesia is produced mainly by three-phase electro-fused furnace [2]. Magnesia is melted by absorbing heat released by the electric arc of three graphite electrodes. Stability of the current of three electrodes is the key factor that influences product quality. Therefore, the most important object of electro-fused magnesia control system is to keep three-phase current stabilizing within a desired range through adjusting position of electrodes, thereby stabilizing the operation of smelting process and achieving corresponding control indices. However, smelting process of electro-fused magnesia furnace is a complicated process that has characteristics like complex operation conditions, strong nonlinearities, and strong couplings, which make it difficult to achieve good control performance using traditional linear control strategy [3]. Consequently, nowadays, the automation level of magnesia smelting process is still low, which is controlled manually in many factories. In order to improve the control performance and enhance the automation level, many researchers [1, 3–6] recently have proposed intelligent control strategies to resolve these problems. However, due to advanced intelligent control methods often involve huge programming task, hard debugging and maintaining problems and are hard to be implemented in programmable logic controller (PLC) and distributed control system (DCS) systems, they are still not widely applied in actual industrial fields.

Recently, with the rapid development of microelectronics, computer technology, and network communication technology, embedded control system has been widely used in the fields of aerospace, automobile manufacturing, industrial process control, intelligent instrument, and robot control because of its strong real-time performance, good customizability, strong communication ability, high reliability, and low cost. However, traditional waterfall development approach which is widely used in traditional control system development procedure cannot meet the demands of nowadays embedded systems [7]. With the rapid development of software engineering technology, embedded software development method based on model design technology, which has been widely used in automobile and aircraft manufacturing fields, provides an effective solution of designing complicated embedded control system. According to [8], model-based design technology can effectively enhance the development efficiency and decrease development cycle, as well as reduce later maintenance costs. Hence, this kind of embedded design method has evoked many researchers' interests. For instance, Karsai et al. [9] used model-based design technology in the control platform of automobiles. Tabbache et al. [10] proposed a hardware-in-the-loop testing platform of city electric car. Wei et al. [11] applied hardware-in-the-loop simulation and model-based control technology to the development and optimization of mathematical model of windscreen wiper. Ferrari et al. [12] introduced model-based design technology to embedded software design of gasoline injection engine through TargetLink tool. Model-based design method was also applied to distributed control system of aircrafts [7, 13]. In the education field, Hercog et al. [14] built a remote control laboratory on the basis of DSP using model-based design technology. In the industrial process control field, Mannori et al. [15] discussed the feasibility of design industrial process control systems based on SciLab and RTAI-Linux. Xu et al. [16] also designed a rapid control prototyping system for temperature control of plastic extruder. It has been widely believed that model-based design technology can improve the design efficiency and reduce the correction time of problem to minimum and decrease development cycle of whole project greatly [8]. However, model-based design technology was rarely introduced to practical industrial process control fields till now, especially in the control of smelting process of electro-fused magnesia furnace.

The main contribution of this paper is that an intelligent control strategy is adopted to achieve good control performance, an embedded real-time control system is implemented using model-based design technology, and the effectiveness of the proposed system is validated through hardware-in-the-loop experiment and practical industrial experiment. The rest of this paper is organized as follows. Section 2 describes the smelting process of arm-type electro-fused magnesia furnace and analyses the control difficulties. Section 3 presents the overall design of the embedded control system and the details of hardware, embedded system software, and intelligent control software. Section 4 analyses the results of hardware-in-the-loop and practical industrial experiments. Section 5 concludes the paper.

2. Description of Smelting Process of Arm-Type Electro-Fused Magnesia Furnace

2.1. Description of Smelting Process. Electro-fused magnesia furnace as shown in Figure 1 is a typical high-energy consumption device, which is actually an electric arc furnace. There are three control subsystems in this equipment, namely, electrode position control system, automatic feeding system, and rotation control system. The main control object is to control the three-phase current to track the setpoint through adjusting the electrode position, which changes the temperate of the furnace indirectly.

The electric arc allows obtaining high temperatures necessary to melt raw ore and realize some chemical reactions. To obtain the electric arc, generally three graphite electrodes are used, which are supplied by a three-phase power transformer. The circuit closes through the metal mass that will be molten. The electric arc appears when the electrodes are near the metal mass. To close the circuit, the electric arc must appear at least between two electrodes and the metal mass. Usually, the distance between the electrode and the metal mass is 5–15 cm. If the length of an electric arc is larger than a certain value, the electric arc will extinguish. In this case, the positioning system must adjust the electrode position so that the electric arc reappears. During the smelting process, feeding machine automatically pours the raw material in the bin into the furnace through the feeding pipe. Therefore, the level of smelting bath rises as more raw materials are melted and filled, and the positions of electrodes have to be adjusted to keep the length of arc within suitable range. In addition, during the smelting process, the furnace is rotated at a certain frequency to ensure the heating surface to be uniform. The smelting process will complete when the level of smelting bath rises up to furnace top.

2.2. Problem Analysis of Controlling Electro-Fused Magnesia Furnace. The main control object of electro-fused magnesia furnace is to increase the product quality and quantity and reduce its energy consumption. However, according to the study of practical industrial equipment, we found at least the following issues increasing the difficulty of accurate and robust control of electro-fused magnesia furnace.

(i) Difficult to establish accurate mathematical model. It is very difficult to obtain accurate mathematical model of smelting process since it is a complicated physical and chemical change process, including electricity, thermodynamics, physics, and chemistry.

(ii) Few controllable variables. As for electro-fused magnesia furnace, available controllable parameters are only the A-phase, B-phase, and C-phase current and three-phase voltages, while the most important variable (inside furnace temperature) which is approximate to 3,000 Celsius cannot be measured directly.

(iii) Complicated disturbances. There exist large range and random disturbances during the smelting process, which come from the inner system rather than the outer.

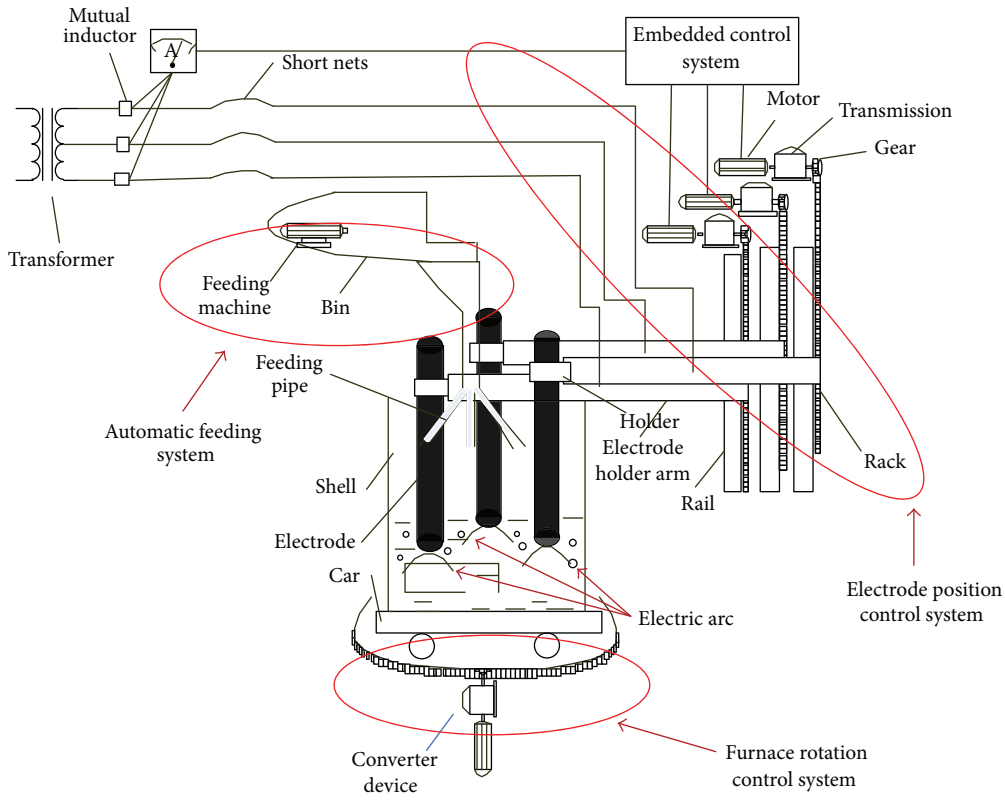


Figure 1: Sketch map of arm-type electro-fused magnesia furnace.

(iv) Strong couplings. There are strong couplings among three-phase current during the smelting process. Therefore, decoupling among three-phase current is a key question.

(v) Harsh operation environments. Tough environment with high temperature, high dust, high power, and high risk poses a higher demand to reliable and stable control system.

Due to the existence of these complicated characteristics, stabilizing the current of three-phases is not easy, and some intelligent control methods should be used to realize the desired control performance. Currently, PLC is the most used controller for control system of electro-fused magnesia furnace. But the small memory and low computation performance of traditional PLC restrict the realization of advanced control algorithms on such platform. In this paper, an embedded control system based on intelligent control method and model-based design technology is designed and developed to control the smelting process of fused magnesium furnace accurately and intelligently.

3. Embedded-Model-Based Control System of Electro-Fused Magnesia Furnace

Figure 2 shows the system architecture of the embedded control system (ECS) for arm-type electro-fused magnesia furnace. The ECS consists of three main components: arm-type electro-fused magnesia furnace, embedded controller, and development PC. The system design is divided into three parts: hardware design, embedded system software design, and intelligent control software design.

3.1. Hardware Design. In this paper, an industrial embedded Single Board Computer (SBC) based on PC104 Bus is chosen as the core control unit to satisfy the computation and industrial environment demands. Figure 3 shows the hardware configuration of the embedded controller.

3.1.1. CPU Board. Since advanced control algorithms always involve large number of calculations, a high CPU frequency is required to ensure the real-time performance of the embedded control system. Besides, the production environment of electro-fused magnesia is very harsh, which belongs to high temperature and high dust operation area. Therefore, the controller should also have high reliability, wide temperature tolerance, low power consumption, and fanless design. According to these demands, a CPU motherboard (Intel PMI2) from SBS Science & Technology Co. is selected in this paper.

3.1.2. Data Acquisition (DAQ) Board. DAQ board is selected according to the input and output demands of practical electro-fused magnesia furnace. Table 1 shows the required inputs and outputs: 11-channel analog input, 4-channel analog

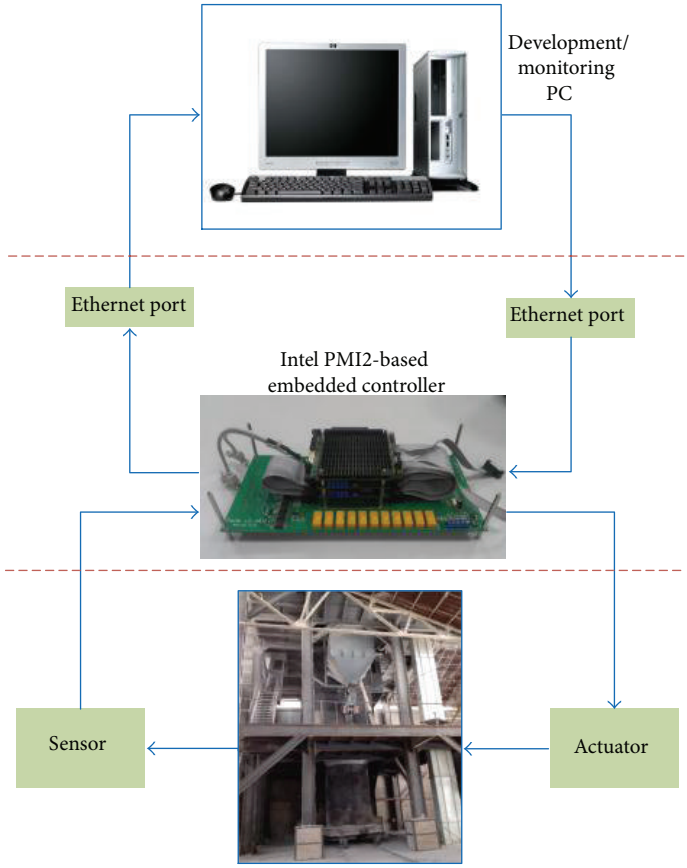


Figure 2: System architecture of arm-type electro-fused magnesia furnace.

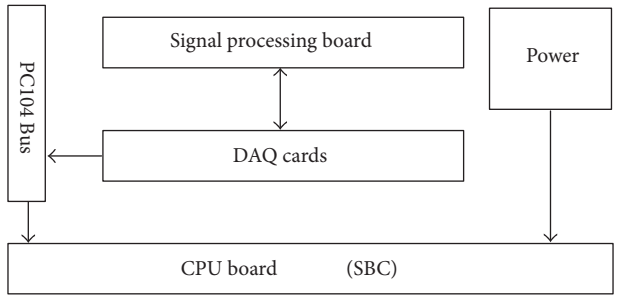


Figure 3: Hardware structure of the controller.

output, 29-channel digital input, and 9-channel digital output. Here, a DAQ board of type ADT652 from SBS Science & Technology Co. is selected. The main ports of this board are as follows: 16-analog input, 4-channel analog output, and 24-channel I/O.

3.1.3. Signal Processing Circuit Design. The signals acquired from electro-fused magnesia furnace often need to be converted into standard signals by the transmitters, and then the standard signals can be acquired by DAQ card. For instance, the electrode current is about 0~15,000 A, which is converted by the current transformer into a current signal of 0~5 A, and then go through the current transmitter and becomes the standard current signal of 4~20 mA. The circuit may failes due to the harsh environment of the industrial field. These failures often lead to large current impact on the DAQ card. Therefore, a signal processing circuit needs to be designed to isolate and filter the signals. The signal processing board for the arm-type electro-fused magnesia furnace is shown in Figure 4. The signal processing board can be divided into four parts: the digital input section, switch output section, analog input section, and analog output section. Both digital input and output signals are isolated by optoisolators. The digital outputs are connected to the plate relays to produce the switch output for industrial process. And the analog inputs are isolated by linear optoisolators.

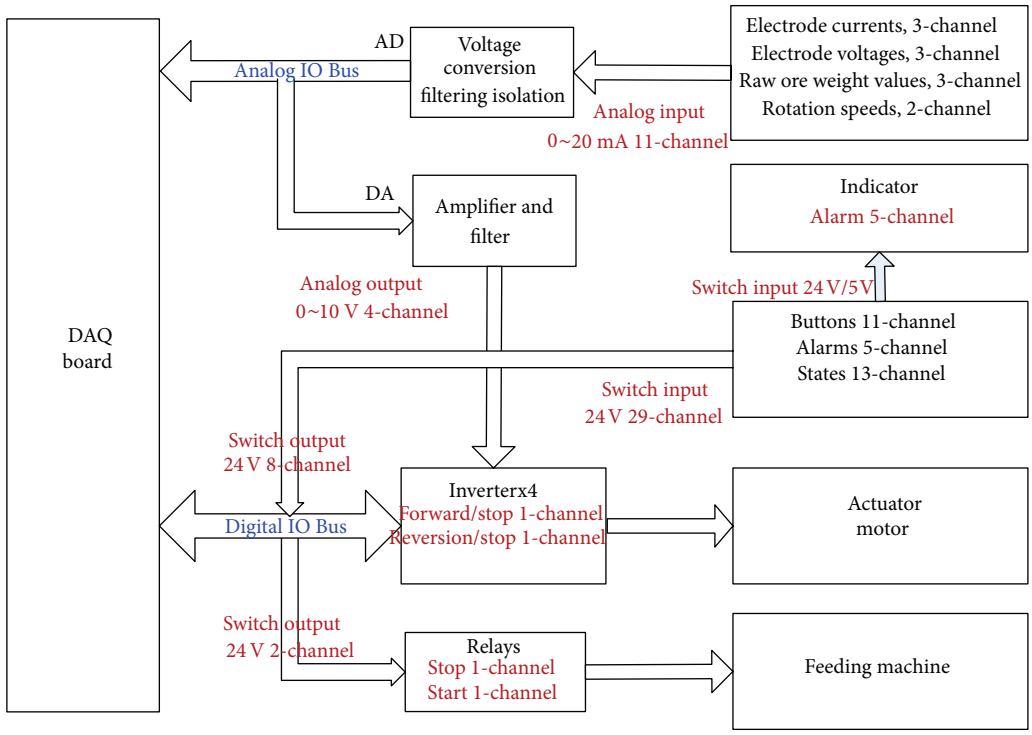


Figure 4: Functional structure of signal processing board.

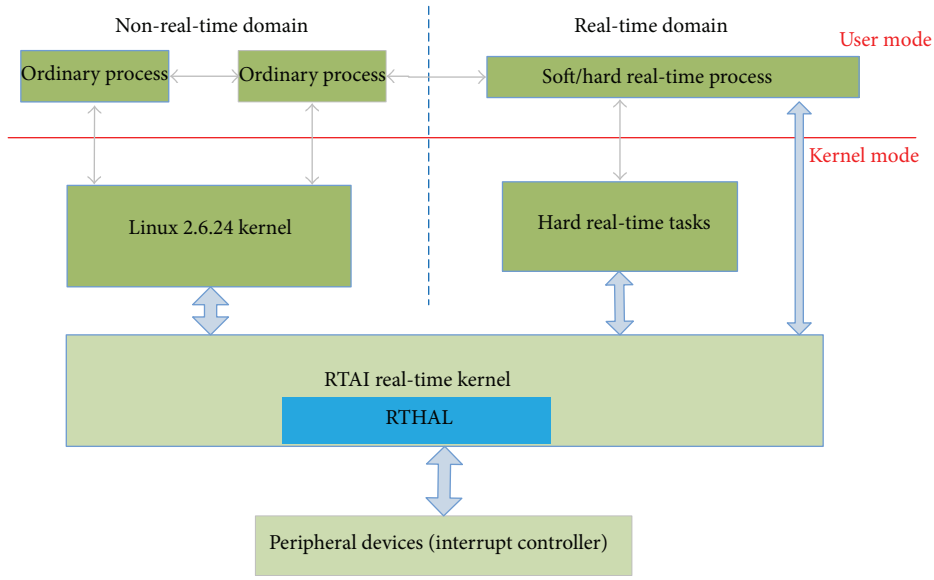


Figure 5: Dual kernel hard real-time system.

3.2. Embedded System Software. In order to make the embedded system become a hard real-time system, dual kernel architecture based on Real-Time Application Interface (RTAI) is used to improve the real-time performance. Besides, Real-Time Workshop (RTW) and Target Language Compiler (TLC) of MATLAB are used to enable automatically generating optimal, real-time embedded code from Simulink/Stateflow models.

3.2.1. Real-Time Kernel. General Linux OS is not a hard real-time operating system. In order to improve its real-time performance to meet the industrial control demands, this paper adopts RTAI-3.7 as the real-time kernel together with Linux 2.6.24 kernel to build a dual kernel hard real-time operating system. The architecture of the real-time OS is shown in Figure 5.

Table 1: Inputs and outputs of the controller.

Signal type	A/D; I/O	Details
3-phase electrode currents	Analog; 3-channel input	0~20 mA
3-phase electrode voltages	Analog; 3-channel input	0~20 mA
3-way raw weight value	Analog; 3-channel input	0~20 mA
Furnace rotation speed	Analog; 1-channel input	0~20 mA
Speed value set by the manual resistor	Analog; 1-channel input	0~20 mA
4-way inverter rotation speed	Analog; 4-channel output	0~20 mA
Upper and lower limit switch of 3-phase electrode	Digital; 6-channel input	Switch signal
Loosen signal of 3-phase electrode holder	Digital; 3-channel input	Switch signal
Holding signal of 3-phase electrode holder	Digital; 3-channel input	Switch signal
High temperature of hydraulic oil	Digital; 1-channel input	Switch signal
Low temperature of hydraulic oil	Digital; 1-channel input	Switch signal
Filter clogging in hydraulic station	Digital; 1-channel input	Switch signal
High pressure of cooling water	Digital; 1-channel input	Switch signal
High temperature of cooling water	Digital; 1-channel input	Switch signal
The furnace reset in place	Digital; 1-channel input	Switch signal
Automatic control mark	Digital; 1-channel input	Switch signal
Manual control mark	Digital; 1-channel input	Switch signal
Manual control electrode lift	digital; 6-channel input	Switch signal
Exhaust button	Digital; 1-channel input	Switch signal
Electric vibrator start button	Digital; 1-channel input	Switch signal
Electric vibrator stop button	Digital; 1-channel input	Switch signal
Inverter forward/stop	Digital; 4-channel output	Switch signal
Inverter reverse/stop	Digital; 4-channel output	Switch signal
Electric feeder machine start/stop	Digital; 1-channel output	Switch signal

The operating system is divided into the real-time domain and non-real-time domain. Real-time processes in the real-time domain are scheduled by the real-time kernel of RTAI, while the ordinary processes in non-real-time domain are still handled by the Linux kernel. Of course, the Linux kernel itself, as a non-real-time process, is managed by RTAI. Therefore, any real-time processes have higher priorities than Linux kernel. Only when there is no real-time process running, the Linux can be scheduled. Secondly, by creating a hardware abstraction layer (called RTHAL) between Linux kernel and hardware, RTAI can get the controllability of hardware interrupt. The RTAI parts that need to be modified in the Linux kernel are defined by RTHAL as a set of API. RTAI can simply use this set of API to communicate with Linux.

3.2.2. Automatic Code Generation. In this paper, Simulink/Stateflow is used as the development tool for developing intelligent control software. Matlab/RTW is used to generate optimized, portable and customized code from Simulink/Stateflow models. Figure 6 shows the automatic generation process.

Since our system uses real-time kernel, some customization steps of automatic code generation mechanism need to be considered.

(i) Customization of entry files of the code generation. In the System Target File (STF) of RTAI, the basic RTW default settings are adopted. And only "codegentry.tlc" file is called to generate five files such as "model.c," "model_data.c," "model.h," "model_private.h," and "model_types.h." In the STF file, Target Language Complier (TLC) variables are configured. Since the code generated is oriented to embedded real-time platform, the "Language" variable is set as "C," the "CodeFormat" is set as "Embedded-C," and the "TargetType" variable is set as "RT." Other variables maintain the original configuration of RTW.

(ii) Customization of code generation process. "STF_make_rtw_hook.m" is responsible for the overall management of the entire code generation process. RTAILab uses the default configuration.

(iii) Customization of code compilation process. Template Makefile (TMF) is modified by RTAILab. In the TMF file, the rt_main.c is firstly imported into the project. Secondly, it also contains some paths of RTW. Paths for the compiling process of RTAILab are listed in Table 2.

3.2.3. DAQ Card Driver Module. As for the DAQ card used in this paper, two steps are required to develop a Simulink device driver. The first step is to use the "set_rt_ext_index()" function provided by RTAI to write the code that controls the DAQ card into the real-time kernel of RTAI. Then, "RTAI_LXRT()"

Table 2: Paths for the compiling process of RTAILab.

Index	Paths/Files
1	/usr/local/matlab/Simulink/src
2	/usr/local/matlab/rtw/c/rtai/devices
3	/usr/local/rtw/c/src
4	/usr/local/rtw/c/libsrc
5	/usr/local/matlab/rtw/c/rtai/lib
6	/usr/src/comedi repectively/include
7	/usr/local/matlab/rtw/c/rtai/devices/sfun-comedi
8	/usr/local/matlab/rtw/c/rtai
9	/usr/real-time/lib/liblxrt.a

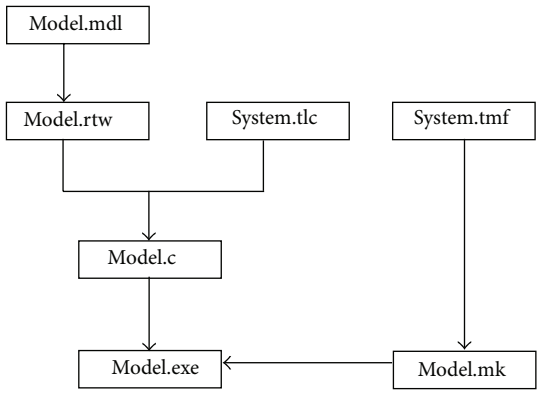


Figure 6: Automatic code generation process.

function is called to map the code into real-time program in the LXRT user space. The second step is writing C MEX S-Function to call the code so that the Simulink model program can control hardware. Figure 7 shows some developed device driver modules.

3.3. Intelligent Control Software. In order to realize robust and accurate control of arm-type electro-fused magnesia furnace, this paper adopts an intelligent control strategy proposed by [3–6, 17]. The intelligent control strategy as shown in Figure 8 is composed of three controllers (current stabilizing controller, exhausting controller, and limit controller) and an operation condition identification module.

According to the previous intelligent control strategy, Simulink and Stateflow are used to develop operating condition identification module, three-phase current stabilizing controller module, exhausting controller module and limit controller module, as shown in Figure 9.

3.3.1. Operating Condition Identification Module. As shown in Figure 9, the inputs of this module contain three-phase current and voltage ("Current A," "Current B," "Current C," and "Voltage") and exhausting and feeding flag ("exh mark" and "pad mark"). The outputs of this module are "Cond," "Param," and "Addition," which represent conditions, parameters and the current fluctuation range, respectively. The internal details are shown in Figure 10, where the "spec cond" submodule is responsible for analyzing the padding and exhausting conditions, and the "cond analysis" submodule is used to identify other special conditions. The two submodules are developed by rule-based reasoning algorithm using Stateflow. Take the "cond analysis" submodule as an example; the padding working condition is conducted regularly, and there exists a timer in the "cond analysis" submodule. When it comes to the fixed time, the "Cond" is outputted as 1, and the "Param" is set up as the relevant parameters of padding working condition. Similarly, when it comes to the exhausting working condition that is conducted regularly, the "Cond" is outputted as 2, and the "Param" is set up as the relevant parameters of exhausting working condition.

3.3.2. Three-Phase Current Stabilizing Controller Module. As shown in Figure 11, three-phase feedback current and the set value of the current are the inputs of this module, and the outputs are the current error "e" and error derivative "ec". Here, a fuzzy control method is adopted to achieve robust control performance [17]. Simulink Toolbox "Fuzzy Logic Toolbox" is used to create Fuzzy Logic Controller. In Matlab, rules editor is used to program the fuzzy rules defined as "fuzzy_fm.fis." In order to prevent large movement of electrode which will cause abnormalities, a saturation module is added.

3.3.3. Exhausting Controller Module. Exhaust controller module based on the RBR algorithm [4] is shown in Figure 12. The inputs of this module are the upper and lower limits and the parameters from the operation condition identification module. The "Chart" submodule is used to adjust three electrodes up and down to exhaust gas.

3.3.4. Limit Controller Module. The state of position limit switch can be determined by the parameters from operation condition identification module. According to the states of the three-phase electrode's position limit switch, the electrodes are slightly lifted up or down to decrease the pressure on the mechanical structure. For example, when the A phase electrode's position limit switch is triggered for upper limit, the A phase electrode should be lifted down slightly. The internal details can be seen in Figure 13.

4. Experiment Results

4.1. HIL Experiment. The HIL simulation platform of arm-type electro-fused magnesia furnace uses one industrial computer based on BP neural network as a virtual arm-type fused magnesia furnace model to simulate the practical operation of arm-type fused magnesia furnace. In the HIL simulation platform, control system considers the values of current and voltage from virtual arm-type electro-fused magnesia furnace model on the industrial computer as its inputs then calculates the action values according to the control algorithm and passes action values to responding actuator (6-group electrical relays). We can verify control effect through watching the relay actions and the output current of the virtual furnace model. The wiring diagram and HIL experiment system are shown in Figures 14 and

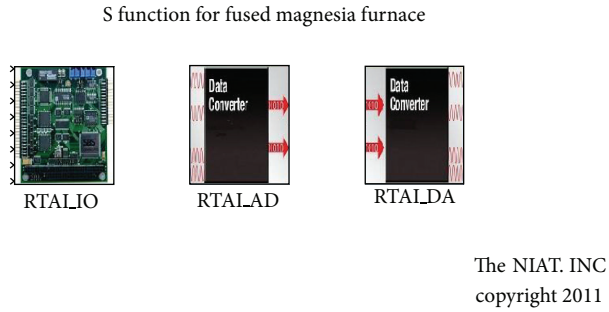


Figure 7: Driver library of the DAQ Card ADT652.

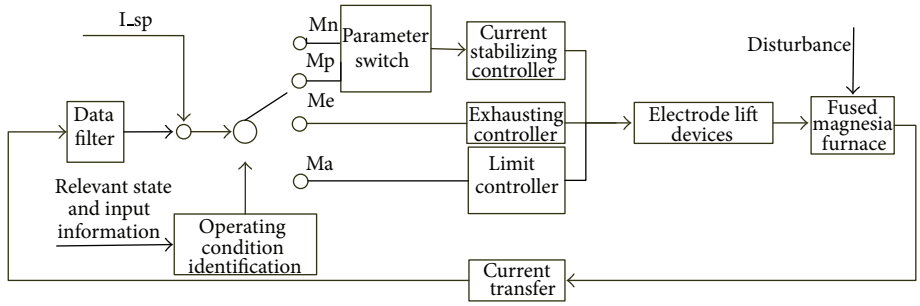


Figure 8: Control strategy of arm-type electro-fused magnesia furnace. I_sp is the current setpoint. Mn means normal condition. Mp means padding condition. Me means exhausting condition, and Ma is limit condition.

Table 3: Parameters of arm-type electro-fused magnesia furnace.

	Parameter	Unit	Value
Shell	Diameter	m	2.2
	Height	m	3.0
Electrode	Diameter	mm	350
Electricity	Capability of transformer	KVA	3500
	Smelted hours	H	12~14
	Rated voltage	V	150

15, respectively. Our control object is to control the three-phase current around 15,000 A. Since it is very difficult to establish an accurate mathematical model of electro-fused magnesia furnace, the HIL experiment is mainly to test the feasibility of the proposed control algorithm and the basic performance of the controller hardware. Figure 16 shows the current control performance. From the result, we can see that, though the control accuracy is not very good in the HIL test, the intelligent controller can realize automatic control of the virtual furnace which is based on BP neural network.

4.2. Industrial Field Experiment. The designed embedded control system was applied to an electro-fused magnesia factory in LiaoNing province, China. The practical furnace and its parameters are shown in Figure 17 and Table 3, respectively.

The proposed embedded control system was applied to practical industrial production to test its performance for one week. During the test, in nearly 85% percent of time, the furnace can be controlled very well without any manually control in one production process. Only at the start-up stage, operators need to control the furnace manually. Our control object is to control the three-phase current around 15,000 A. Figure 18 shows the three-phase currents during the production after the furnace is started up manually by the operator and switched to automatic control. As it is can be seen, three-phase current can be stabilized within the range from 14,000 A to 16,000 A during most time. And, once actual current exceeds the desired range, the controller can adjust the position of corresponding electrode to let current go back to the desired range quickly.

5. Conclusion and Discussion

In this paper, a model-based embedded control system is designed and developed for the process control of the electro-fused magnesia furnace. The embedded controller is based on industrial Single Board Computer and is a hard real-time system based on dual kernel architecture. In the embedded controller, an intelligent control strategy of electro-fused magnesia furnace is developed using model-based design technology. The real-time embedded control software is generated directly from the Simulink/Stateflow models. To validate the performance of the designed embedded controller, HIL experiment and industrial field experiment are both implemented, which demonstrated that our embedded control system works well in both laboratory and industry environments.

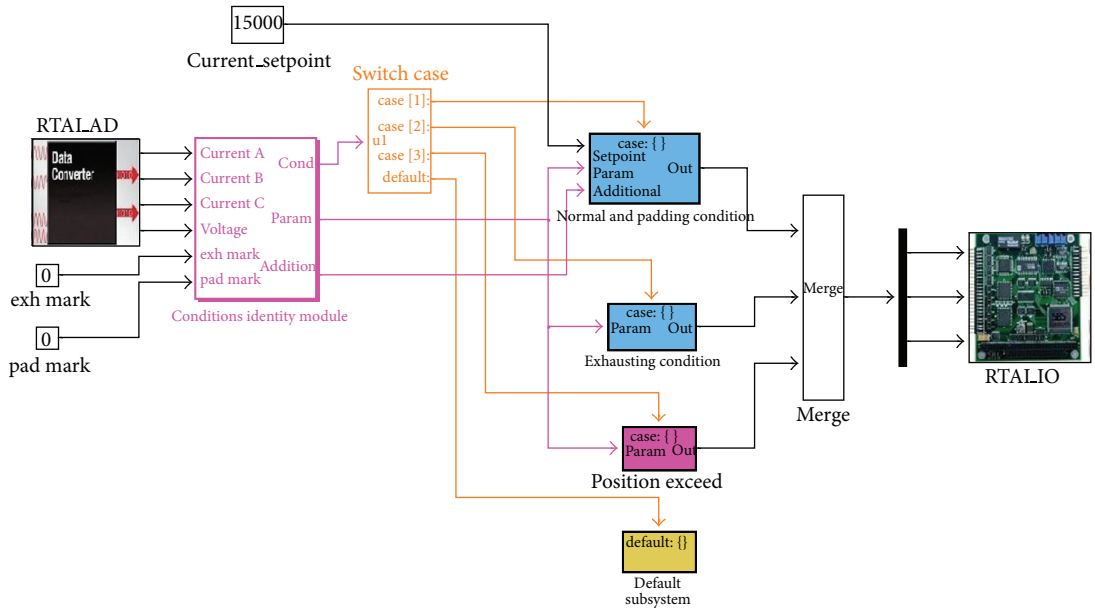


Figure 9: Simulink model of intelligent control strategy.

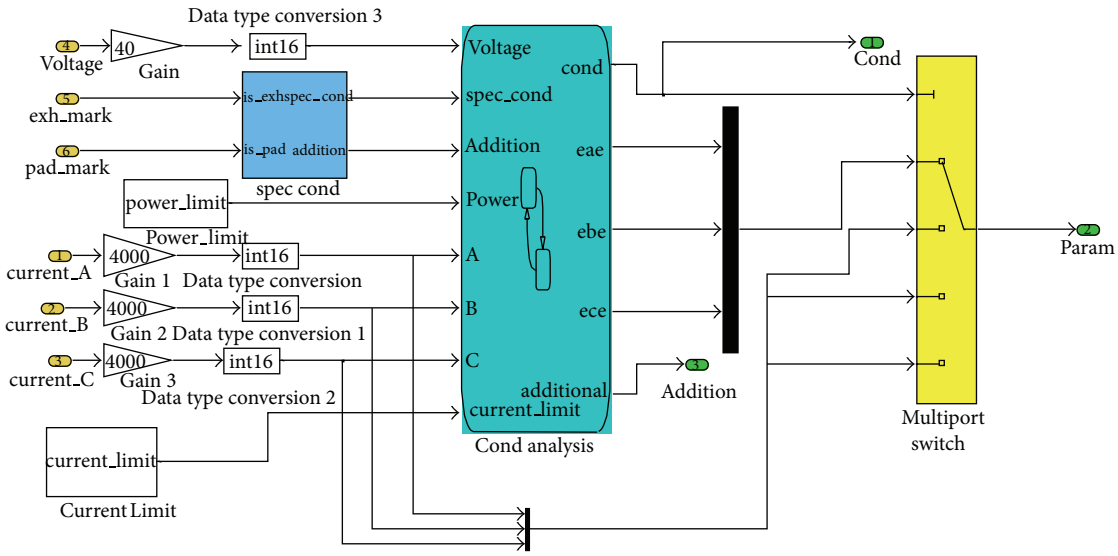


Figure 10: Internal details of operating condition identification module.

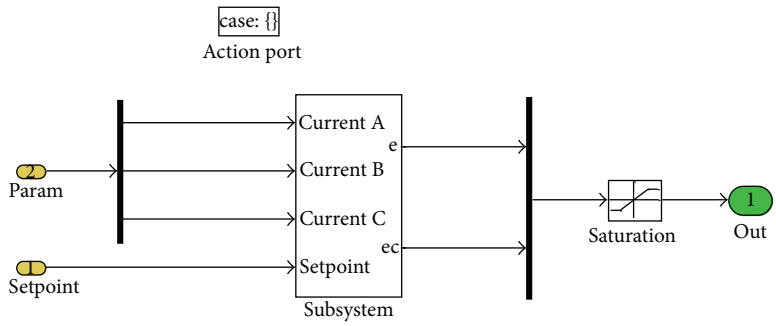


Figure 11: Internal details of three-phase current stabilizing controller module.

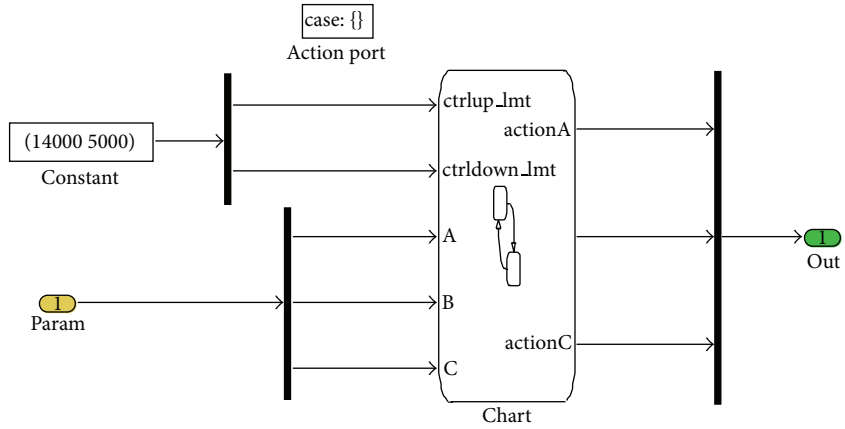


Figure 12: Internal details of exhausting module.

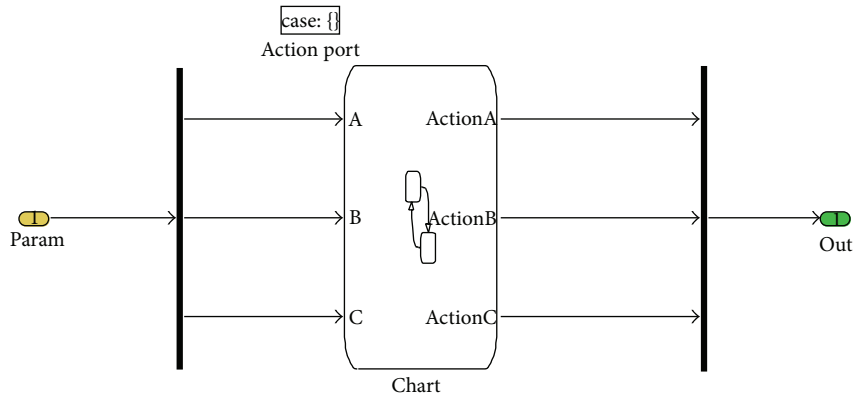


Figure 13: Internal Details of Limit Controller Module.

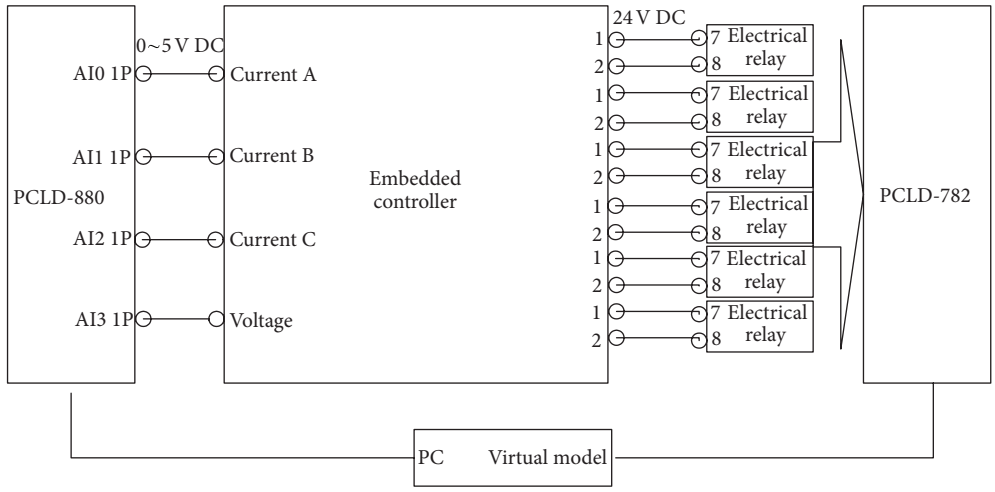


Figure 14: Wiring Diagram of Hardware-in-the-loop Simulation Platform.

Figure 15: Practical HIL Simulation Platform and the Embedded Controller.

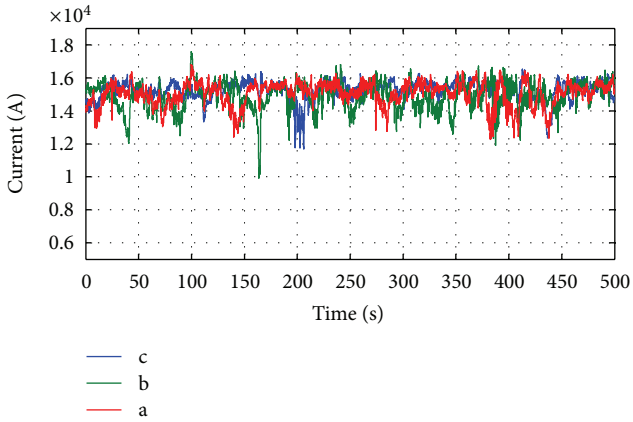


Figure 16: Three-phase current during the HIL test.

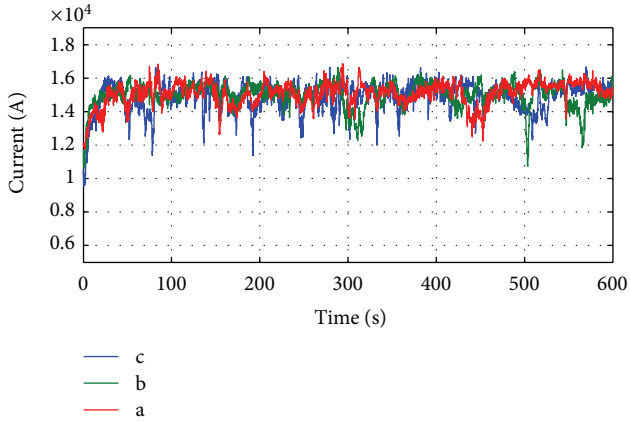


Figure 18: Current curve of intelligent control strategy.

Figure 17: Arm-type electro-fused magnesia furnace.

In the future, the safety and reliability of the controller will be improved to adapt the high dust and strong electromagnetic disturbance environment. Besides, some optimal operation control algorithm, fault diagnosis, and fault-tolerant control algorithms will also be implemented on this embedded control system to further improve the control performance.

Acknowledgments

The authors want to thank the State Key Lab of Synthetical Automation for Process Industries and the Fundamental Research Funds for the Central Universities under Grant no. N100408003, as well as National Science Foundation of China under Grant no. 61040014, supporting on the project.

References

[1] Y. Wu, Z. Wu, B. Dong, L. Zhang, and T. Chai, "The hybrid intelligent control for the fused magnesia production," in *Proceedings of the 48th IEEE Conference on Decision and Control Held Jointly with the 28th Chinese Control Conference*, pp. 3294–3299, Shanghai, China, December 2009.

[2] D. M. Wang and M. X. Guo, "Study on electro-thermal power and structure of electro-fused magnesia furnace," *Energy for Metallurgical Industry*, vol. 16, no. 1, pp. 36–39, 1997.

[3] Z. W. Wu, Z. Fang, T. Y. Chai, X. H. Zhang, and C. Wang, "Research on special embedded controller and its control method for fused magnesium furnace," *Chinese Journal of Scientific Instrument*, vol. 33, no. 6, pp. 1261–1267, 2012.

[4] Z. W. Wu, Y. J. Wu, T. Y. Chai, and L. Zhang, "Intelligent control system of fused magnesia production via rule-based reasoning,"

Journal of Northeastern University (Natural Science), vol. 30, no. 11, pp. 1526–1529, 2009.

[5] Z. W. Wu, T. Y. Chai, J. Fu, and J. Sun, "Hybrid intelligent optimal control of fused magnesium furnaces," in *Proceedings of the 49th IEEE Conference on Decision and Control (CDC '10)*, pp. 3313–3318, December 2010.

[6] Z. W. Wu, Y. J. Wu, and T. Y. Chai, "Intelligent control of fused magnesium furnaces based on SPSA," *Journal of Shanghai Jiaotong University*, vol. 45, no. 8, pp. 1095–1100, 2011.

[7] C. C. Insaurralde, M. A. Seminario, J. F. Jiménez, and J. M. Giron-Sierra, "Model-based development framework for distributed embedded control of aircraft fuel systems," in *Proceedings of the 29th IEEE/AIAA Digital Avionics Systems Conference (DASC '10)*, pp. 6.E.21–6.E.214, October 2010.

[8] B. Selic, "The pragmatics of model-driven development," *IEEE Software*, vol. 20, no. 5, pp. 19–25, 2003.

[9] G. Karsai, A. Ledeczi, S. Neema, and J. Sztipanovits, "The model-integrated computing toolsuite: metaprogrammable tools for embedded control system design," in *Proceedings of IEEE International Symposium on Computer-Aided Control Systems Design*, pp. 50–55, October 2006.

[10] B. Tabbache, Y. Aboub, K. Marouani, A. Kheloui, and M. E. H. Benbouzid, "A simple and effective hardware-in-the-loop simulation platform for urban electric vehicles," in *Proceedings of 1st International Conference on Renewable Energies and Vehicular Technology (REVET '12)*, pp. 251–255, March 2012.

[11] J. L. Wei, A. Mouzakitis, J. H. Wang, and H. Sun, "Vehicle windscreen wiper mathematical model development and optimisation for model based hardware-in-the-loop simulation and control," in *Proceedings of the 17th International Conference on Automation and Computing (ICAC '11)*, pp. 207–212, September 2011.

[12] A. Ferrari, G. Gaviani, G. Gentile, G. Stara, L. Romagnoli, and T. Thomsen, "From conception to implementation: a model based design approach," in *Proceedings of IFAC Symposium on Advances in Automotive Control (IFAC-AAC '04)*, pp. 29–34, 2004.

[13] T. A. Henzinger, C. M. Kirsch, M. A. A. Sanvido, and W. Pree, "From control models to real-time code using Giotto," *IEEE Control Systems Magazine*, vol. 23, no. 1, pp. 50–64, 2003.

[14] D. Hercog, B. Gergič, S. Uran, and K. Jezernik, "A DSP-based remote control laboratory," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 6, pp. 3057–3068, 2007.

[15] S. Mannori, R. Nikoukhah, and S. Steer, "Free and open source software for industrial process control systems," http://www.scicos.org/ScicosHIL/angers2006eng.pdf.

[16] J. L. Xu, G. K. Zuo, J. H. Chen, and M. H. Wan, "A rapid control prototyping system design for temperature control of plastic extruder based on Labview," in *Proceedings of International Conference on Electronics, Communications and Control*, pp. 2471–2474, September 2011.

[17] S. Y. Jiang, *Design and development of control software for arm type fused magnesia furnace [M.S. thesis]*, Northeastern University, 2012.

Hindawi Publishing Corporation
Journal of Control Science and Engineering
Volume 2013, Article ID 136841, 12 pages
http://dx.doi.org/10.1155/2013/136841

Research Article

Position Control of a 3-CPU Spherical Parallel Manipulator

Massimo Callegari,[1] Luca Carbonari,[1] Giacomo Palmieri,[2] Matteo-Claudio Palpacelli,[1] and Donatello Tina[1]

[1] *Department of Industrial Engineering & Mathematical Sciences, Polytechnic University of Marche, 60131 Ancona, Italy*
[2] *e-Campus University, Faculty of Engineering, 22060 Novedrate, Italy*

Correspondence should be addressed to Massimo Callegari; m.callegari@univpm.it

Received 7 September 2012; Accepted 18 December 2012

Academic Editor: Sabri Cetinkunt

The paper presents the first experimental results on the control of a prototypal robot designed for the orientation of parts or tools. The innovative machine is a spherical parallel manipulator actuated by 3 linear motors; several position control schemes have been tested and compared with the final aim of designing an interaction controller. The relative simplicity of machine kinematics allowed to test algorithms requiring the closed-loop evaluation of both inverse and direct kinematics; the compensation of gravitational terms has been experimented as well.

1. Introduction

Parallel kinematics machines, PKMs, are known to be characterized by many advantages like a lightweight construction and a high stiffness but also present some drawbacks, like the limited workspace, the great number of joints of the mechanical structure, and the complex kinematics, especially for 6-dof machines [1]. Therefore the A.'s proposed to decompose full-mobility operations into elemental subtasks, to be performed by separate reduced mobility machines, similarly to what is already done in conventional machining operations. They envisaged the architecture of a mechatronic system where two parallel robots cooperate in order to perform complex assembly tasks. The kinematics of both machines is based upon the same 3-CPU topology but the joints are differently assembled so as to obtain a translating parallel machine (TPM) with one mechanism and a spherical parallel machine (SPM) with the other.

This solution, at the cost of a more sophisticated controller, would lead to the design of simpler machines that could be used also stand-alone for 3-dof tasks and would increase the modularity and reconfigurability of the robotized industrial process. The two robots are now available at the prototypal stage, and the present paper reports the first experiments on the motion control of the orienting device (SPM).

2. Robot's Architecture and Kinematics

2.1. Mechanical Architecture. Since the detailed description of machine's kinematics and prototype design has been provided already in Callegari et al. [2], hereby only the most relevant aspects are recalled.

The spherical parallel machine under study is made of three identical serial chains connecting the moving platform to the fixed base, as shown in Figure 1; each leg is composed by two links: the first one is connected to the frame by a cylindrical joint (C), while the second link is connected to the first one by a prismatic joint (P) and to the end-effector by a universal joint (U); for this reason its mechanical architecture is commonly called 3-CPU. A few *manufacturing conditions*, already investigated for a general pure rotational tripod by Karouia and Hervè [3], must be fulfilled in order to constraint the end-effector to a spherical motion:

(i) the axes of the cylindrical joints ($\mathbf{a}_i$, $i = 1, 2, 3$) are aligned along the x, y, z axes of the base frame and intersect at the center O of the spherical motion;

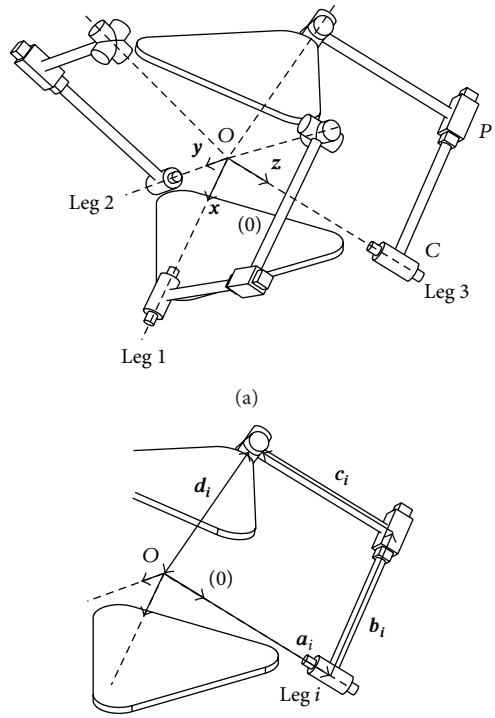


Figure 1: Kinematic schemes of the 3-CPU robot (a) and geometry of the legs (b).

(ii) the axis $\mathbf{b}_i$ of each prismatic pair is perpendicular to the axis of the respective cylindrical joint $\mathbf{a}_i$;

(iii) the first axis of each universal joint is perpendicular to the plane of the corresponding leg (plane identified by the axes $\mathbf{a}_i$ and $\mathbf{b}_i$);

(iv) the second axis of the 3 universal joints (resp., for the leg 1, 2, and 3) are aligned along the y_1, z_1, x_1 axes of a local frame centered in P (coincident with O) and attached to the mobile platform.

For a successful operation of the mechanism, one *mounting condition* must be satisfied too; assembly should be operated in such a way that the two frames $O(\mathbf{x_0}, \mathbf{y_0}, \mathbf{z_0})$ and $P(\mathbf{x_1}, \mathbf{y_1}, \mathbf{z_1})$ come to coincide when the robot is in its homing position. Such configuration is obtained when the three displacements a_i are equal to the length of the second link c and the displacements of the prismatic joints b_i are equal to the constant distance d. If the mounting conditions are verified, the points P and O remain fixed and coincident while the moving platform performs a spherical motion around them.

2.2. Kinematic Relations. The platform is actuated by driving the strokes of the 3 cylindrical joints; therefore joint space displacements are gathered into the following vector $\mathbf{q}$:

$$\mathbf{q} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}. \tag{1}$$

The position kinematics of the robot expresses the relation between the orientation of the mobile platform and the displacements of the actuators; the attitude of the machine in space is fully provided by the rotation matrix ${}^{O}_{P}\mathbf{R}$, that can also be conveniently expressed as a composition of elemental rotations. In the development of robot's kinematics, the following Cardan angles set is used:

$$\begin{aligned} {}^{O}_{P}\mathbf{R}(\alpha, \beta, \gamma) &= \mathbf{R}_x(\alpha)\,\mathbf{R}_y(\beta)\,\mathbf{R}_z(\gamma) \\ &= \begin{bmatrix} c\beta c\gamma & -c\beta s\gamma & s\beta \\ s\alpha s\beta c\gamma + c\alpha s\gamma & -s\alpha s\beta s\gamma + c\alpha c\gamma & -s\alpha c\beta \\ -c\alpha s\beta c\gamma + s\alpha s\gamma & c\alpha s\beta s\gamma + s\alpha c\gamma & c\alpha c\beta \end{bmatrix}. \end{aligned} \tag{2}$$

The position kinematics of the robot is simply expressed by

$$\begin{aligned} r_{12} &= -c\beta s\gamma = \frac{c - a_1}{d}, \\ r_{23} &= -s\alpha c\beta = \frac{c - a_2}{d}, \\ r_{31} &= -c\alpha s\beta c\gamma + s\alpha s\gamma = \frac{c - a_3}{d}, \end{aligned} \tag{3}$$

where r_{ij} is the element at the ith row and jth column of rotation matrix ${}^{O}_{P}\mathbf{R}$. The solution of the *direct position kinematics* (DPK) problem requires the computation of the rotation matrix ${}^{O}_{P}\mathbf{R}$ as a function of internal coordinates $\mathbf{q}$, which has been solved already by Carbonari et al. [4]. According to Innocenti and Parenti-Castelli [5], a maximum number of 8 different configurations can be worked out; however, a single feasible solution is found when the real workspace of the robot is considered; that is, the actual mobility of the joints is taken into consideration. *Inverse position kinematic* (IPK) problem admits just one solution and it is trivially solved by working out joint displacements $\mathbf{q}$ in (3).

Turning to *differential kinematics*, the expression of the analytic Jacobian $\mathbf{J}_A$ is immediately obtained as a function of the Cardan angles and their rates:

$$\begin{bmatrix} \dot{a}_1 \\ \dot{a}_2 \\ \dot{a}_3 \end{bmatrix} = \mathbf{J}_A \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix},$$

$$\mathbf{J}_A = d \begin{bmatrix} 0 & -s\beta s\gamma & c\beta c\gamma \\ c\alpha c\beta & -s\alpha s\beta & 0 \\ -s\alpha s\beta c\gamma - c\alpha s\gamma & c\alpha c\beta c\gamma & -c\alpha s\beta s\gamma - s\alpha c\gamma \end{bmatrix}. \tag{4}$$

By taking into account the relation between the derivatives of the Cardan angles and the angular velocity $\boldsymbol{\omega}$:

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & s\beta \\ 0 & c\alpha & -s\alpha c\beta \\ 0 & s\alpha & c\alpha c\beta \end{bmatrix} \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix} = \mathbf{T} \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix}, \tag{5}$$

the geometric Jacobian $\mathbf{J}_G$ is easily obtained too:

$$\begin{bmatrix} \dot{a}_1 \\ \dot{a}_2 \\ \dot{a}_3 \end{bmatrix} = \mathbf{J}_A \mathbf{T}^{-1} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \mathbf{J}_G \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}, \tag{6}$$

with

$$\mathbf{J}_G = d \begin{bmatrix} 0 & -c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha c\gamma - s\alpha s\beta s\gamma \\ c\alpha c\beta & 0 & -s\beta \\ -s\alpha s\beta c\gamma - c\alpha s\gamma & c\beta c\gamma & 0 \end{bmatrix}. \tag{7}$$

2.3. User Frames. In order to better define the tasks to be commanded and visualize the obtained results, it is useful to choose a different set of reference frames, as shown in Figure 2. The fixed frame $O^*(\mathbf{x_0^*}, \mathbf{y_0^*}, \mathbf{z_0^*})$ is defined as follows:

(i) the origin is located at the center of the moving platform when it assumes its initial configuration;

(ii) the $\mathbf{z_0^*}$ axis is aligned to the vector $\mathbf{g}$ of gravity acceleration;

(iii) the $\mathbf{x_0^*}$ axis lies on the upper plane of the platform and points toward the axis $\mathbf{a}_1$ of the cylindrical joint of the first leg;

(iv) the $\mathbf{y_0^*}$ axis is placed according to the right-hand rule.

The mobile frame $P^*(\mathbf{x_1^*}, \mathbf{y_1^*}, \mathbf{z_1^*})$ is coincident with the fixed frame $O^*(\mathbf{x_0^*}, \mathbf{y_0^*}, \mathbf{z_0^*})$ when the platform is in its initial configuration. Of course, since the frames are not placed at the center of the spherical motion, the two origins O^* and P^* will be coincident only in the home position.

Once the location of the new frame O^* has been defined by means of the ${}^{O}_{O^*}\mathbf{R}$ rotation matrix, the orientation of the mobile platform can be described in the new frames by

$${}^{O^*}_{P^*}\mathbf{R} = {}^{O}_{O^*}\mathbf{R}^T \, {}^{O}_{P}\mathbf{R} \, {}^{O}_{O^*}\mathbf{R}, \tag{8}$$

where it has been used the identity ${}^{O}_{O^*}\mathbf{R} = {}^{P}_{P^*}\mathbf{R}$. Of course, being the mobile and fixed frames modified, also the Cardan angles $\varphi_x, \varphi_y, \varphi_z$ that yield the rotation matrix ${}^{O^*}_{P^*}\mathbf{R}$ are different from the previously described set (α, β, γ):

$${}^{O^*}_{P^*}\mathbf{R}\left(\varphi_x, \varphi_y, \varphi_z\right) = \mathbf{R}_{x^*}\left(\varphi_x\right) \mathbf{R}_{y^*}\left(\varphi_y\right) \mathbf{R}_{z^*}\left(\varphi_z\right). \tag{9}$$

Henceforth these angles are used to describe the orientation of the manipulator and to assign the tasks of the mobile platform; since they are assumed as external coordinates for the computation of the differential kinematics, the analytic and the geometric Jacobians are worked out again as previously described, providing similar but more complex relations.

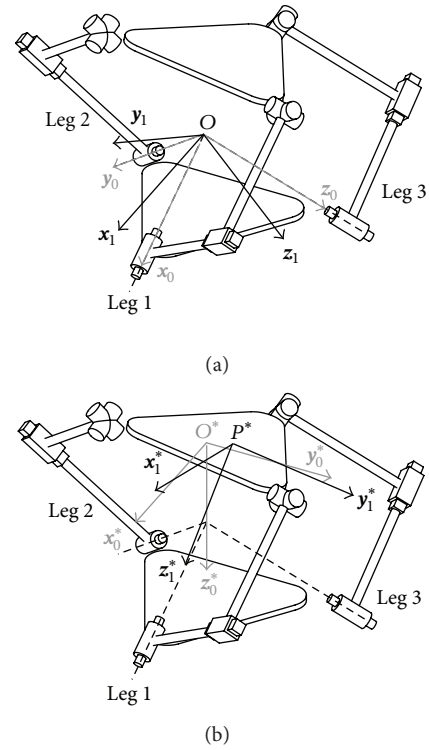


Figure 2: User-defined task frames.

3. Control Algorithms

3.1. Overview. Several kinds of control schemes have been tried on the 3-CPU SPM, with the immediate goal of testing the prototypal robot but aiming at the final design of an efficient co-operative environment for mechanical assembly. In the end, 3 different algorithms have been studied in simulation and then experimentally tested:

(i) a conventional joint resolved PID [6];

(ii) a joint resolved PID with the compensation of gravity forces [7];

(iii) a task-space PID with gravity compensation [8].

In all control schemes, the PID loop has been computed as usual in the following way:

$$\mathbf{u}(t) = \mathbf{K}_P \left(\mathbf{e}(t) + \frac{1}{\mathbf{T}_i} \int_{\mathbf{0}}^{t} \mathbf{e}(\tau)\, d\tau + \mathbf{T}_D \frac{d}{dt} \mathbf{e}(t) \right), \tag{10}$$

with $\mathbf{u}(t)$ control action and $\mathbf{e}(t)$ input position error; $\mathbf{K}_P$, $\mathbf{T}_I$, and $\mathbf{T}_D$ are, respectively, the proportional gain, integral time, and derivative time matrices of the PID regulator.

3.2. Joint Resolved PID. First, a conventional joint resolved PID has been considered; see Figure 3. The error signal $\tilde{\mathbf{a}}$ is computed in the joint space as a difference between the desired position of the sliders $\mathbf{a}_D$ and their actual values $\mathbf{a}$:

$$\tilde{\mathbf{a}} = \mathbf{a}_D - \mathbf{a}. \tag{11}$$

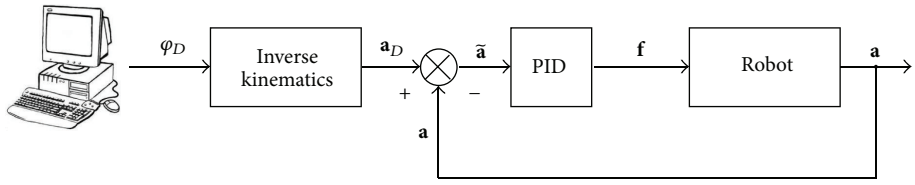


Figure 3: Joint space PID controller.

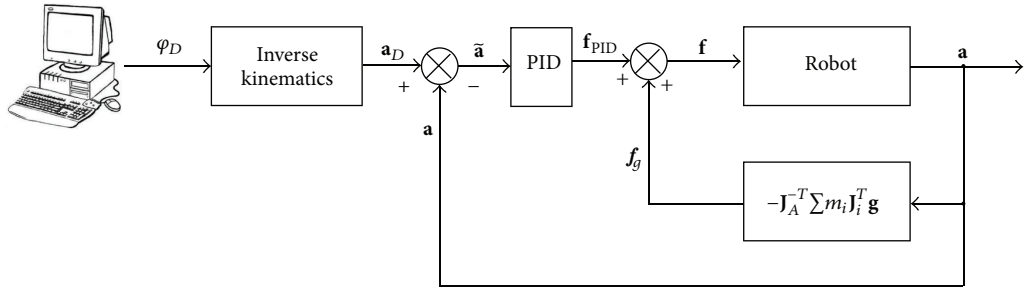


Figure 4: Joint space PID controller with gravity compensation.

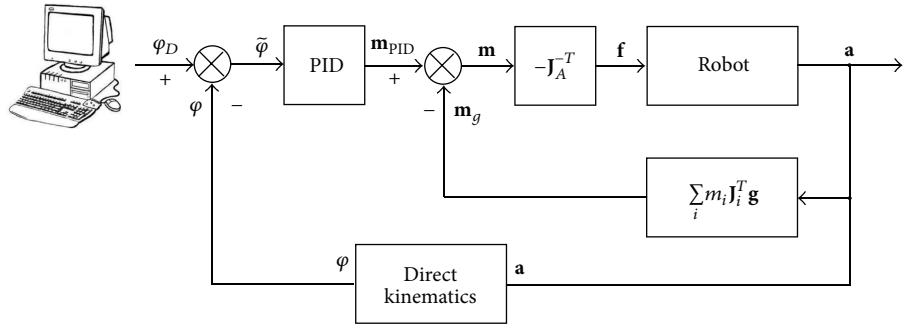


Figure 5: Task space PID controller with gravity compensation.

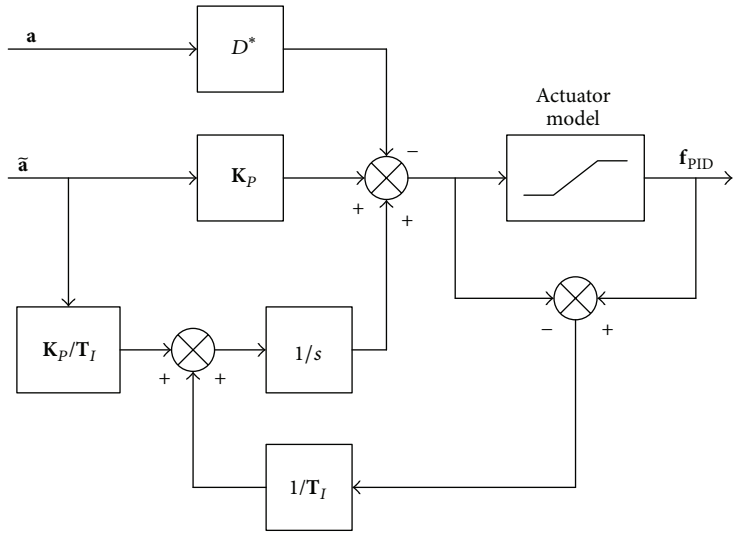


Figure 6: Anti-windup modified PID.

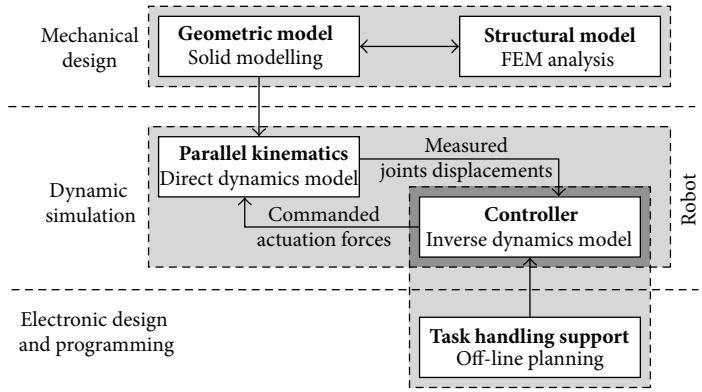


Figure 7: Integrated virtual prototyping environment for PKM's analysis and design.

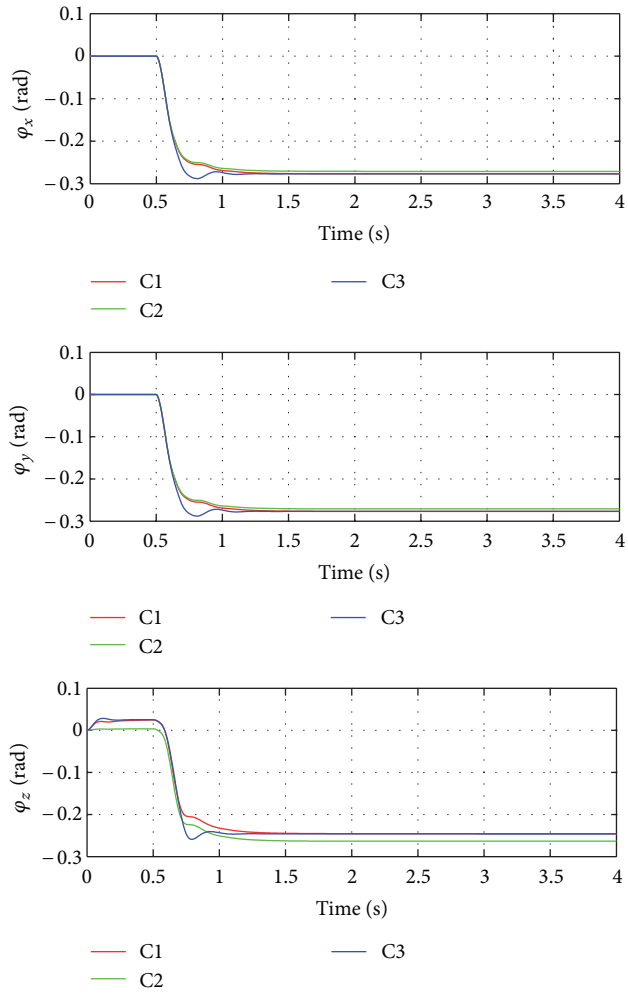


Figure 8: Response to step input: task space trajectories.

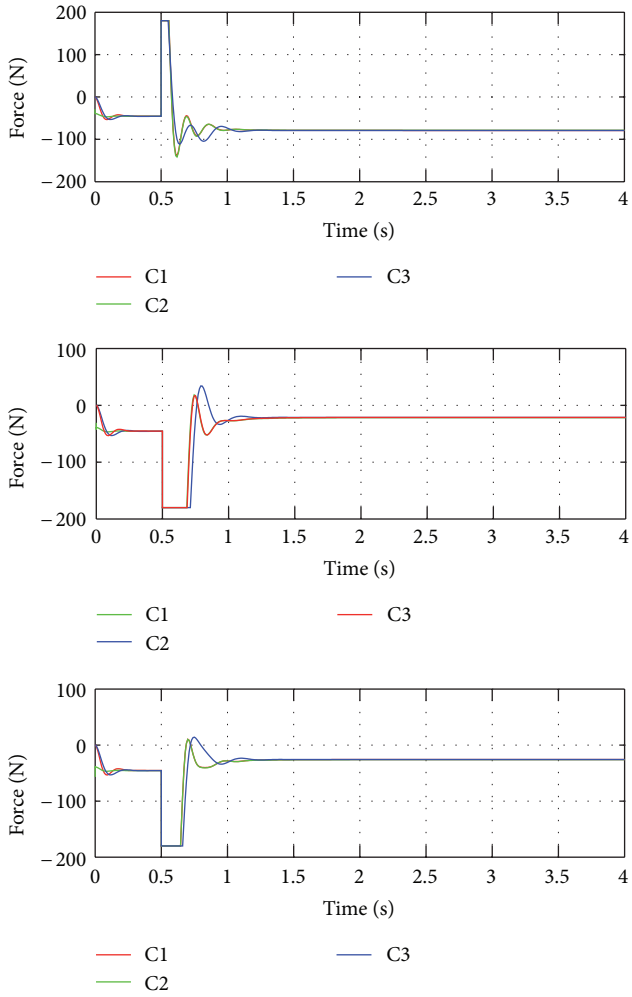


Figure 9: Response to step input: control efforts.

Since planning is programmed in the orientation space by assigning the desired configuration of the robot $\boldsymbol{\varphi}_D$, the corresponding position of the actuated joints is computed by means of inverse kinematics relations. The actuation effort of the motors is computed as

$$\mathbf{f} = \mathbf{K}_P \left[\tilde{\mathbf{a}} + \frac{1}{\mathbf{T}_i} \int \tilde{\mathbf{a}}\, dt + \mathbf{T}_D \frac{d\tilde{\mathbf{a}}}{dt} \right], \tag{12}$$

where the diagonal matrices $\mathbf{K}_P$, $\mathbf{T}_I$, and $\mathbf{T}_D$ have been introduced already in the previous section.

3.3. Joint Resolved PID with Gravity Compensation. In robotics the effects of gravitational field are often much more relevant than the other dynamics terms, at least for the small/moderate velocities attained during assembly tasks; such terms can be easily evaluated by means of the virtual work principle, as worked out in Callegari et al. [2]. Thus, a compensation term can be introduced by adding the force vector:

$$\mathbf{f}_g = -\mathbf{J}_A^{-T} \sum_i m_i \mathbf{J}_i^T \mathbf{g}, \tag{13}$$

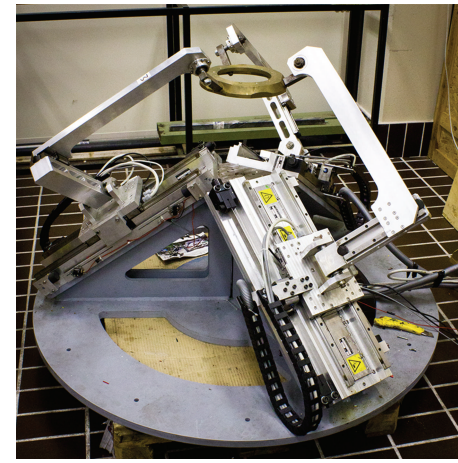


Figure 10: The prototype of the spherical parallel machine.

where $\mathbf{J}_A$ is the analytic Jacobian matrix, m_i is the mass of the ith member, $\mathbf{J}_i$ is the Jacobian that links the velocity of the centre of gravity of the ith member to the vector $\dot{\mathbf{a}}$, and $\mathbf{g}$ is the gravity acceleration. The resulting control scheme is shown in Figure 4.

3.4. Task Space PID with Gravity Compensation. The third control scheme that has been taken into consideration is a task space PID, with the compensation of the gravitational terms; see Figure 5:

$$\mathbf{f} = \mathbf{J}_A^{-T} \left(\mathbf{K}_P^{'} \left[\tilde{\boldsymbol{\varphi}} + \frac{1}{\mathbf{T}_I^{'}} \int \tilde{\boldsymbol{\varphi}}\, dt + \mathbf{T}_D^{'} \frac{d\tilde{\boldsymbol{\varphi}}}{dt} \right] - \sum_i m_i \mathbf{J}_i^T \mathbf{g} \right), \tag{14}$$

where $\tilde{\boldsymbol{\varphi}}$ is the error signal in the task space and the PID gains $\mathbf{K}_P^{'}$, $\mathbf{T}_I^{'}$, and $\mathbf{T}_D^{'}$ are diagonal matrices once again. This algorithm is computationally more expensive than the previous one, since it requires the evaluation of direct kinematics that for PKMs is more complex than inverse kinematics; on the other hand, it could prove useful, for example, in vision assisted assembly tasks with position-based controls, as already experimented on the 3-CPU translating parallel machine by Palmieri et al. [9].

3.5. Implementation in Real-Time Controller. During the implementation of algorithms (12)–(14) on the real-time controller, it was took into consideration the sensitiveness to noise of differentiation. Considering the Laplace transform of (10), the mentioned problem has been numerically mitigated by substituting the classic derivative term $K_P T_D s$ with the following derivative operator:

$$D^*(s) = \frac{K_P T_D s}{1 + s T_D / N}, \tag{15}$$

where N has been chosen equal to 10. Another problem in the implementation of PID controllers over a real-time system is the windup effect. This phenomenon is due to the integral action which saturates the actuators output.

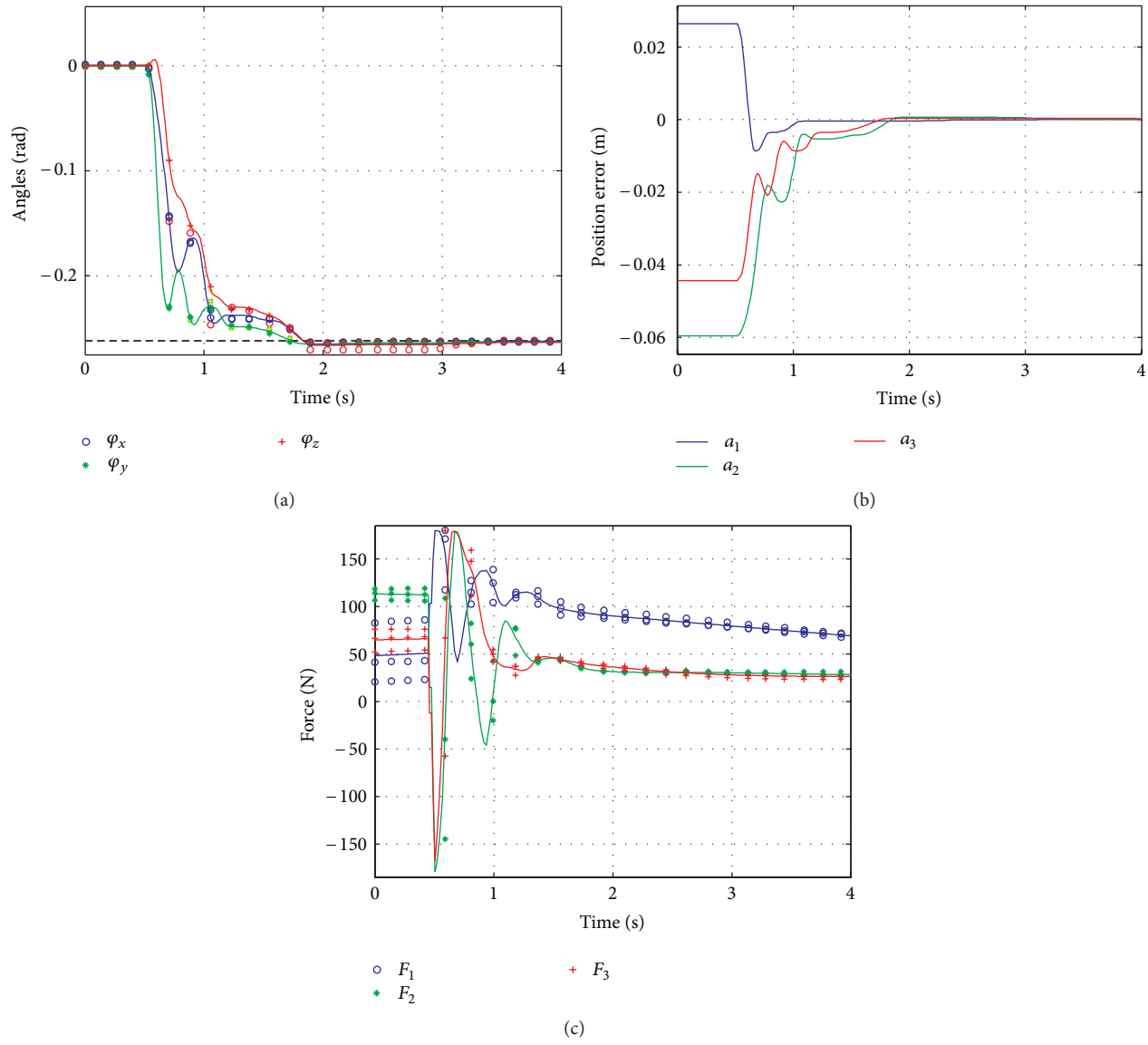


Figure 11: Joint-space PID controller: platform's trajectory in task-space (a), joint space errors (b), and motors thrusts (c).

Figure 6 shows a modified scheme of the PID control which implements a typical anti-windup strategy; the model of the actuator, mentioned in the scheme, was easily obtained after identification of the motor mechanical and electrical parameters summarized in Table 2.

4. Simulation Results

4.1. Simulation Environment. Figure 7 shows the virtual prototyping environment used at the Robotics Laboratory of the Polytechnic University of Marche for the design of automated and robotized systems, in particular for the design and virtual testing of parallel kinematic manipulators. The mechanical design is developed through conventional CAD tools, which allow to easily define even the most complex geometries and also to perform, for example, by means of FEM modules, the needed structural analyses; the interface with a multibody code allows to perform closed-loop dynamic analyses, with different levels of difficulty according to the associativity of the used programs. In this case, the LMS Virtual. Lab Motion package has been used, which is able to handle conveniently also complex situations like, for instance, the occurrence of an impact. The multibody package receives in input from the controller the actuation torques and integrates the equation of direct dynamics, providing in output the state variables assumed to be measured. The control system, which is implemented in the Matlab/Simulink environment, computes the control actions by taking into account the commanded task and sometimes, just like the present case, by also exploiting the complete or partial knowledge of robot's dynamics (inverse dynamics model). If the task is constrained by the contact with the environment, like is usually the case for assembly, the contact forces can be evaluated too, to set up more efficient force control schemes. It is noted that, by

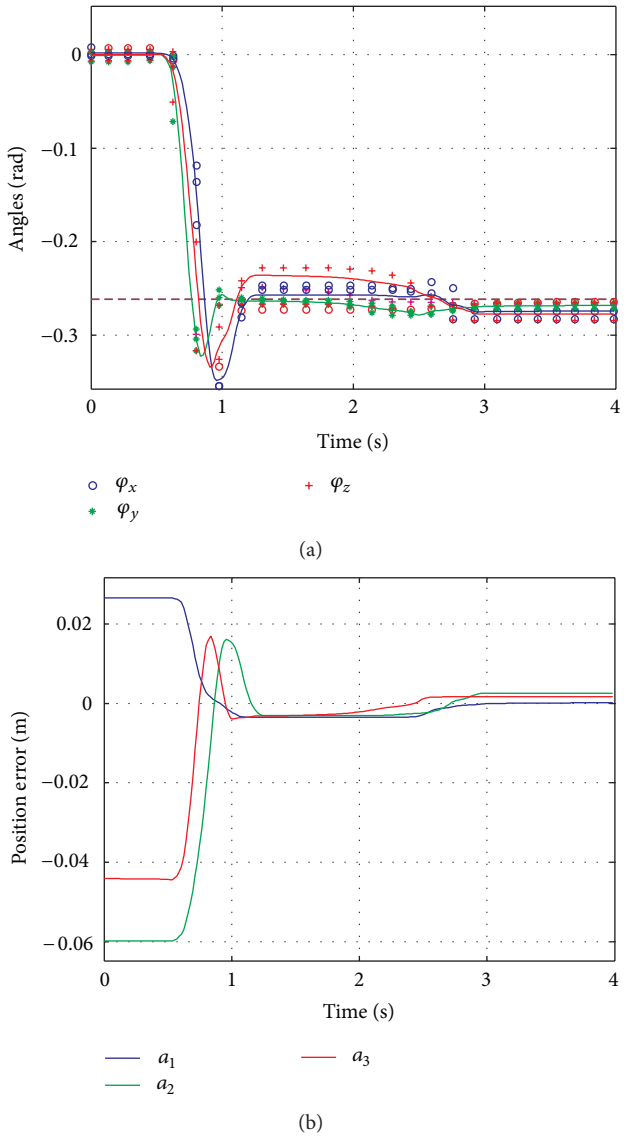


FIGURE 12: Joint-space PID with gravity compensation: platform's trajectory in task-space (a) and joint space errors (b).

TABLE 1: PIDs gains.

	Joint space	Joint space with gravity compensation	Task space with gravity compensation
K_P [N/m]	25000	15000	1000
T_I [s]	200	100	100
T_D [s]	1	0.2	0.25

using the Real-Time Workshop package of the Matlab suite, the same code used during the simulations in the virtual prototyping environment has been directly ported to the real-time control hardware afterwards.

In this way, by means of the mentioned prototyping software, a model of the spherical robot has been made available for the design of the control system and for the

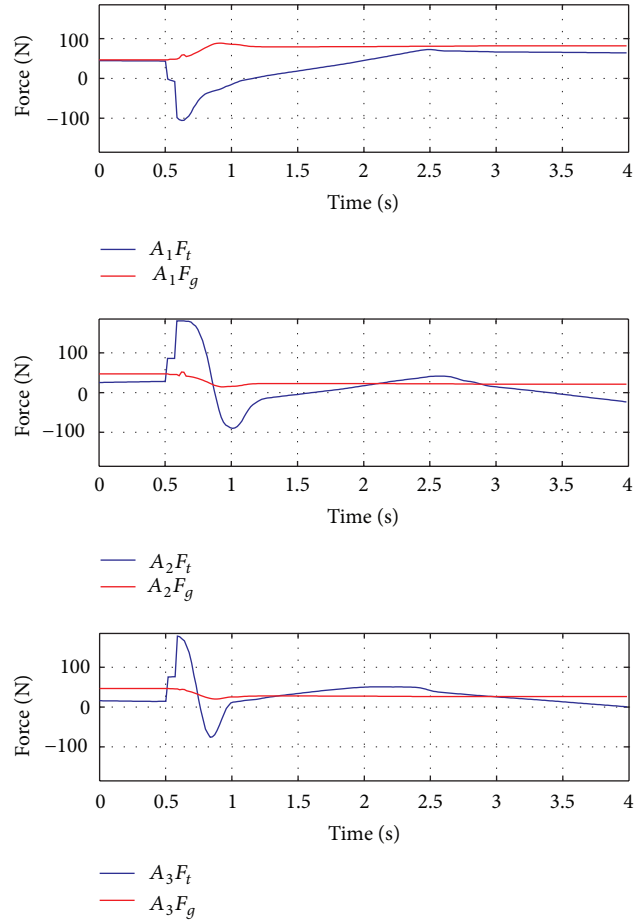


FIGURE 13: Joint-space PID with gravity compensation: F_t total force provided by the motors, F_g gravity component.

TABLE 2: Parameters of the linear drives.

Motors properties			
M_s	2.95	kg	Stator mass
K_t	58	N/A	Torque constant
I_n	3	A	Nominal supply current
T_n	184	N	Nominal thrust
v_n	6	m/s	Nominal speed

tuning of the PID's. Table 1 collects some control gains at the end of the tuning procedure, based on both simulation runs and experimental tests.

4.2. Simulation Analysis. A few test cases have been set up in simulation to evaluate the performances of the 3 PID controllers described in Section 3. The figures show the response of the system when the robot started at rest from

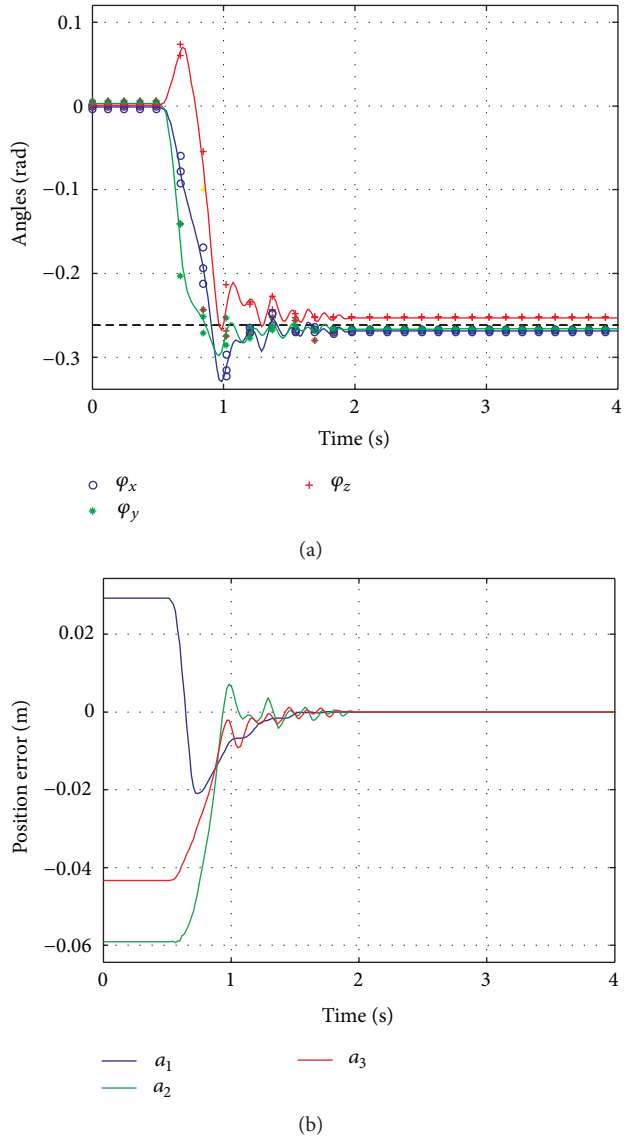


FIGURE 14: Task-space PID with gravity compensation: platform's trajectory in workspace (a) and joint space errors (b).

the home configuration ($\varphi_x = \varphi_y = \varphi_z = 0$) and was required to attain the set point:

$$\boldsymbol{\varphi}_D = \begin{bmatrix} \varphi_{x,D} \\ \varphi_{y,D} \\ \varphi_{z,D} \end{bmatrix} = \begin{bmatrix} -15^\circ \\ -15^\circ \\ -15^\circ \end{bmatrix} = \begin{bmatrix} -0.262 \\ -0.262 \\ -0.262 \end{bmatrix} \text{ rad}, \qquad \dot{\boldsymbol{\varphi}}_D = 0. \tag{16}$$

Such task is very challenging for machine's controller because the set point lies close to a singular configuration of the robot and algorithms (13) and (14) require the inversion of the Jacobian matrix. Figure 8 shows the different performances, in simulation, between the three different controllers: C1, C2, C3 are, respectively, joint resolved PID, joint resolved PID with gravity compensation, and task space PID with gravity compensation. It is noted that the robot is not kept at its home position by means of the brakes but the motors are used to

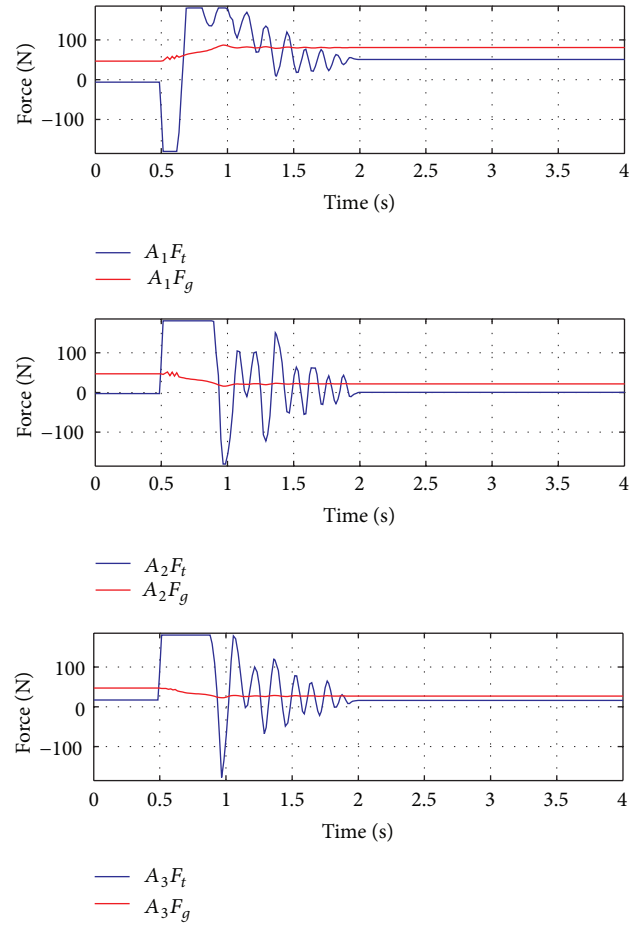


FIGURE 15: Task-space PID with gravity compensation: F_t total force provided by the motors, F_g gravity component.

this aim instead, then the set point has been applied in all trials at the time instant $t = 0.5$ s. The orientation trajectories in the task space show the better behaviour of the closed-loop system when it is equipped with the conventional PID algorithm, due to the mentioned presence of singularities.

The simulations also return useful information for what regards the control effort forces, which are plotted in Figure 9: in all cases the application of the set point causes a peak in the required forces, which saturates the actuators.

In the end, it is noted that the task space PID with gravity compensation is more sensitive to parameter variations. This is due to the intrinsic characteristics of robot prototype, which has no external sensor and many singular configurations. In this way, all the information about the task space is obtained through the direct kinematics and the robot Jacobian. Small errors in the computation may affect heavily control system's performance.

5. Experimental Results

5.1. Experimental Setup. The prototype robot is shown in Figure 10; it is actuated by three brushless linear motors by *Phase* and controlled by a *National Instrument* board based

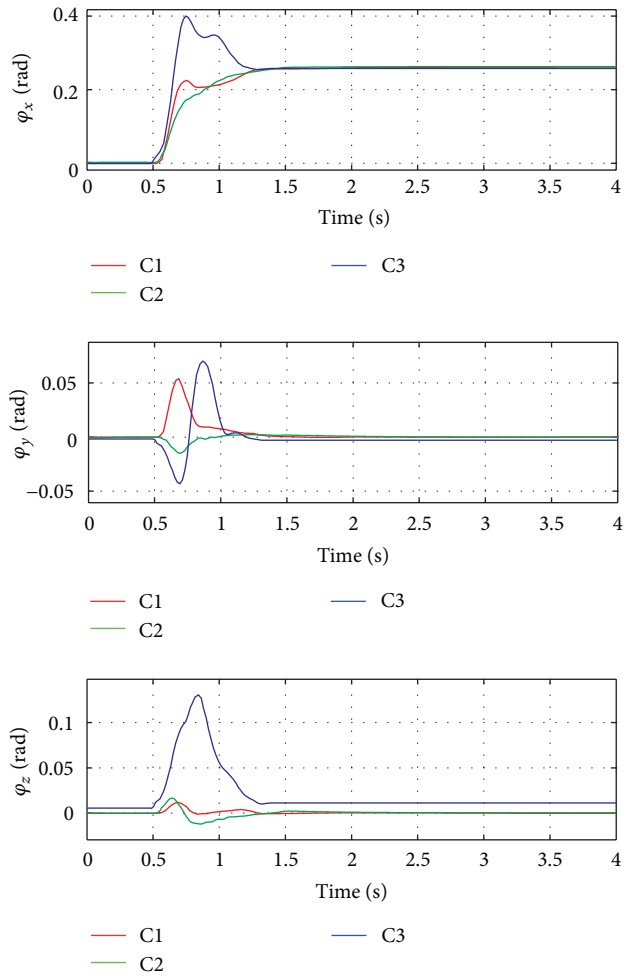


FIGURE 16: Comparative behavior of the three controllers: task-space trajectories.

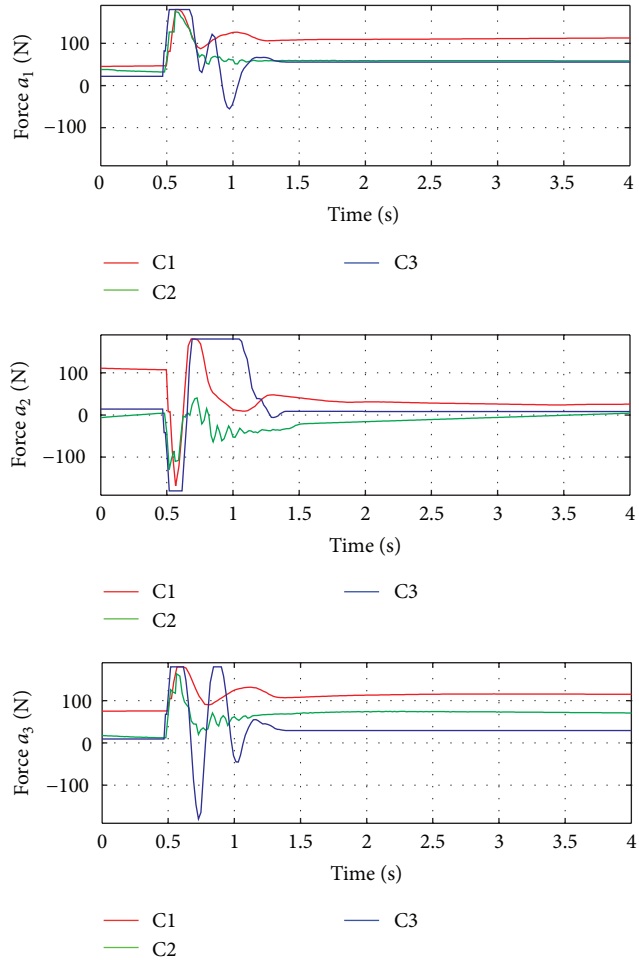


FIGURE 17: Comparative behavior of the three controllers: actuation forces.

on the *PXI/FlexMotion* hardware. The force developed by the sliders is obtained by directly setting the current loop of the drivers, according to the usual relation between the current i and the thrust F:

$$F = K_t i, \tag{17}$$

where the torque constant K_t characterizes the performances of the motor; see Table 2.

With reference to the symbols introduced in Figure 1, the main design data of the prototype are collected in Table 3.

A series of experimental tests have been carried out in order to validate the numerical model described in the previous sections and to experimentally assess the performances of control laws (12)–(14). Results are here reported.

5.2. Case Study A. The first case study was already investigated in simulation, so that numerical and experimental results can now be compared. The platform at the home position has been requested to attain once again the task space set point (16), which corresponds to the following motor strokes:

$$\mathbf{a}_D = \begin{bmatrix} a_{1,D} \\ a_{2,D} \\ a_{3,D} \end{bmatrix} = \begin{bmatrix} 516.4 \\ 430.6 \\ 445.6 \end{bmatrix} \text{[mm]}, \qquad \dot{\mathbf{a}}_D = 0. \tag{18}$$

Many tests have been performed for each one of the three control laws (12)–(14) and in the figures some experimental results are presented; values of one of the experimental trials are represented by circle markers while the averaged quantities are represented by solid lines.

Figure 11 presents some results obtained with the conventional PID loop. It is seen that steady state is achieved in less than one second without significant oscillations, due to the pretty high mechanical damping of the system. The corresponding actuation forces F_i are rather large in the first instants, approaching motors' saturation thrusts, then they settle along the static value required for gravity compensation.

By observing Figures 11, 12, 13, and 14, it results that the introduction of a gravity compensation term into the joint loop brings in system's dynamics overshoots that increase

Table 3: Mechanical data of the robot prototype.

Geometrical data		
c	210	mm
d	490	mm
h	280	mm
$a_{i\min}$	319	mm
$a_{i\max}$	661	mm
$b_{i\min}$	130	mm
$b_{i\max}$	210	mm
Mass data		
Slider	7.15	kg
Link 1	1.90	kg
Link 2	2.21	kg
Platform	11.73	kg

the settle time. The task-space controller, on the other hand, requires much more actuation efforts in the first instants of the trials; see Figure 15.

5.3. Case Study B. The second case study was aimed at comparing the performances of the 3 controllers and therefore an elemental task was chosen; the wrist started in quiet at the home pose and was requested to reach the configuration:

$$\boldsymbol{\varphi}_D = \begin{bmatrix} \varphi_{x,D} \\ \varphi_{y,D} \\ \varphi_{z,D} \end{bmatrix} = \begin{bmatrix} 15^\circ \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.262 \text{ rad} \\ 0 \\ 0 \end{bmatrix}, \qquad \dot{\boldsymbol{\varphi}}_D = 0, \quad (19)$$

which corresponds to the following motor strokes:

$$\mathbf{a}_D = \begin{bmatrix} a_{1,D} \\ a_{2,D} \\ a_{3,D} \end{bmatrix} = \begin{bmatrix} 470.2 \\ 533.2 \\ 470.2 \end{bmatrix} \text{ [mm]}, \qquad \dot{\mathbf{a}}_D = 0, \qquad (20)$$

Figure 16 shows the trend of the workspace variable vector $\varphi = [\varphi_x \ \varphi_y \ \varphi_z]^T$ in the three different cases, while Figure 17 plots the actuation forces developed by the motors.

6. Conclusions

The paper presented the first experiments in driving a prototypal SPM developed at the Polytechnic University of Marche by means of linear controllers. The use of a virtual simulation environment can be very profitable in the design of robots' controllers and even in the draft tuning of their parameters. In the present case, three controllers have been first designed in simulation and then implemented on an embedded system for real-time application by means of rapid prototyping software. The task-space controller with the compensation of the gravity terms provided poorer performances than conventional joint-space loops, but in A.'s opinion it could be due to calibration errors, that would heavily affect the computation of direct kinematics, which is required with the present sensing equipment; the use of ANN controllers could be profitable to overcome unmodeled disturbances due to static friction or calibration errors [10].

As a matter of fact, task-space control schemes would be very useful for the realization of interaction controllers, see Siciliano and Villani [11], which is the objective of A.'s coming researches, since they aim at performing mechanical assembly by means of cooperating PKMs [12].

The simplicity of direct kinematics of this machine (in comparison with the usual complexity of PKMs) allows an efficient implementation of algorithms with loop closures in the task space; the same can be said for the easy compensation of the static unbalance of robot's links.

These features and the possible use of visual servoing suggest a possible implementation of control schemes based on force control, where machine's dynamics has to be computed in task-space coordinates, which is rather "natural" for parallel kinematics machines.

References

[1] J. P. Merlet, *Parallel Robots*, Springer, Dordrecht, The Netherlands, 2nd edition, 2006.

[2] M. Callegari, L. Carbonari, G. Palmieri, and M. C. Palpacelli, "Parallel wrists for enhancing grasping performances," in *Grasping in Robotics*, G. Carbone, Ed., pp. 189–219, Springer, New York, NY, USA, 2013.

[3] M. Karouia and J. M. Hervè, "A three-dof tripod for generating spherical rotation," in *Advances in Robot Kinematics*, J. L. Lenarcic and M. M. Stanisic, Eds., pp. 395–402, Kluwer Academic, Dordrecht, The Netherlands, 2000.

[4] L. Carbonari, L. Bruzzone, and M. Callegari, "Impedance control of a spherical parallel platform," *International Journal of Intelligent Mechatronics and Robotics*, vol. 1, no. 1, pp. 40–60, 2011.

[5] C. Innocenti and V. Parenti-Castelli, "Echelon form solution of direct kinematics for the general fully-parallel spherical wrist," *Mechanism and Machine Theory*, vol. 28, no. 4, pp. 553–561, 1993.

[6] I. Cervantes and J. Alvarez-Ramirez, "On the PID tracking control of robot manipulators," *Systems and Control Letters*, vol. 42, no. 1, pp. 37–46, 2001.

[7] C. Yang, Q. Huang, H. Jiang, O. Ogbobe Peter, and J. Han, "PD control with gravity compensation for hydraulic 6-DOF parallel manipulator," *Mechanism and Machine Theory*, vol. 45, no. 4, pp. 666–677, 2010.

[8] R. Kelly and J. Moreno, "Manipulator motion control in operational space using joint velocity inner loops," *Automatica*, vol. 41, no. 8, pp. 1423–1432, 2005.

[9] G. Palmieri, M. C. Palpacelli, M. Battistelli, and M. Callegari, "A comparison between position-based and image-based dynamic visual servoings in the control of a translating parallel manipulator," *Journal of Robotics*, vol. 2012, Article ID 103954, 11 pages, 2012.

[10] D. Tina, L. Carbonari, and M. Callegari, "Design and experimentation of a neural network controller for a spherical parallel machine," in *Proceedings of the 9th International Conference on Informatics in Control, Automation and Robotics (ICINCO '12)*, vol. 1, pp. 250–255, Rome, Italy, 2012.

[11] B. Siciliano and L. Villani, *Robot Force Control*, Springer, Dordrecht, The Netherlands, 2000.

[12] L. Bruzzone and M. Callegari, "Application of the rotation matrix natural invariants to impedance control of purely rotational parallel robots," *Advances in Mechanical Engineering*, vol. 2010, Article ID 284976, 9 pages, 2010.