Target-based Prototyping System Applied to Fuel Control

Pingan He, Blake R. Suhre, Chris Doyle, and Michael J. Lemancik

***Abstract*—A target-based rapid prototyping system, MotoHawk, is described for controls development, vehicle or engine calibration, fleet testing and production. MotoHawk features the capability of controlling varied types of engines (gasoline and diesel engines from single-cylinder to multi-cylinder), auto-code generation of Simulink/Stateflow models to a family of production Electronic Control Units (ECUs), and a calibration interface incorporated into the models. Finally, an illustrative example of a MotoHawk application, the design and implementation of a fuel control strategy for gasoline engines, is discussed in detail.**

I. Introduction

TODAY'S advanced Engine Management Systems (EMS) are highly sensed, controlled and actuated, and need sophisticated embedded software and require long development periods. On the other hand, software cost pressure and intense competition require continuous reduction of EMS innovation cycle time. One such solution is the concept of the rapid prototyping system [1-6].

Traditionally, the transition from the algorithm and prototype development to the production code generation is realized by dedicated software engineers, which requires significant software engineering overhead. In order to make this transition fast and efficient, MotoTron has introduced the target-based prototyping/production code development system – MotoHawk [7-8]. By providing a common tool for autocode generation, modeling, control system design, and I/O functions, MotoHawk can automatically and efficiently generate development *and* production code simultaneously, which closes the gap between systems engineers and software engineers, reduces software validation capital investment, and improves software quality.

The primary capabilities of MotoHawk relative to engine system modeling, control system design and analysis, and diagnostics include the following [6-8]:

1) Capability of controlling various types of engines, including gasoline and diesel, from single-cylinder to multi-cylinder;
2) A generous, flexible, I/O set library, which is directly accessed and configurable through calibration;
3) An embedded software framework, which provides an RTOS, angular and time-based priority scheduling, and multiple communication protocols (CAN and RS 485),
4) A fault manager module, which standardizes sensor, actuator and other engine subsystem on-board diagnostics;
5) An embedded calibration tool, which allows for the display of all calibrations and measurement parameters, simultaneous connections to multiple ECUs, and multiple applications connected to any single ECU;
6) A system debug tool, which monitors the stack, heap, and CPU margin violation, as well as other critical errors in real time;

In addition, MotoHawk provides knock detection and control, variable cam timing phase detection and control, and is compatible with Motorola's MPC555, MPC565 and HCS12 microcontrollers.

Nomenclature is listed in Section II. In Section III, we introduce MotoHawk basics, and outline the necessary steps to build an engine management system using the MotoHawk environment. Section IV presents a fuel control strategy design and its implementation with MotoHawk. In Section V, detailed experimental steps are listed to test the proposed fuel control strategy performance on a spark ignition engine and Section VI presents the conclusions.

II. Nomenclature

η_v	volumetric efficiency, [*unitless*]
θ	throttle position, [%]
ϕ	equivalence ratio, [*unitless*]
A/F	air fuel ratio, [Stoichiometric = 14.7]
m_a	mass of air inducted into each cylinder, [g]
$\dot{m}_a$	total engine mass air flow rate, [g/s]
N	engine speed, [rad/sec]
n	number of cylinders, [*unitless*]
n_R	number of crank revolutions for each power stroke per cylinder, [*unitless*]
P	intake manifold air pressure, [Pa]
P_{baro}	barometric pressure, [Pa]
R	ideal gas constant, [= 8.314 $J/(mol \cdot K)$]
T_a	intake air temperature, [$Kelvin$]
T_q	torque, [$N \cdot m$]
V_d	total cylinder swept volume, [m^3]

III. MotoHawk

MotoHawk is an ECU-based rapid prototyping system that provides the ability to develop, test and validate engine control applications designed in Simulink/Stateflow running on top of the RTOS framework and production hardware [7]. The primary MotoHawk functional blocks consist of RTOS interface, module digital sequences (injector, spark coil,

P. He, B. R. Suhre, C. Doyle, and M. J. Lemancik are with MotoTron Corp., 505 Marion Rd, Oshkosh, WI 54901 USA (contact author: P. He; e-mail: evertuga@yahoo.com).

variable cam phase, etc.), analog I/O (PWM, wide-band oxygen sensor, etc.), interface to CAN and serial communications devices, interface to the calibration and display, fault manager definition, engine position (encoder), and debug tools.

In the following example, a basic engine management system (EMS) is constructed using MotoHawk. The top level MotoHawk diagram of an EMS is shown in Figure 1. This EMS consists of an operating system, engine constants, vehicle and gauge, controller, and engine. Each of these subsystems will be explained next.

A. *Operating System*

The operating system definition block defines the target hardware, stack size, heap size, and system debug tool. The target hardware could be, for example, an MPC555, MPC565, MPC563, HCS12 or other microcontroller-based system. Foreground, background, idle and interrupt stack sizes need to be defined for the multi-tasking operating system. A system debug tool monitors the stack, heap, and CPU margin violations or other critical errors in real time. In any case, when one of these errors is discovered, certain actuator will be shut down safely, such as fuel injectors and spark coils. The operating system can handle event-based and/or time-based embedded systems and supports both angle- and time-based priority scheduling.

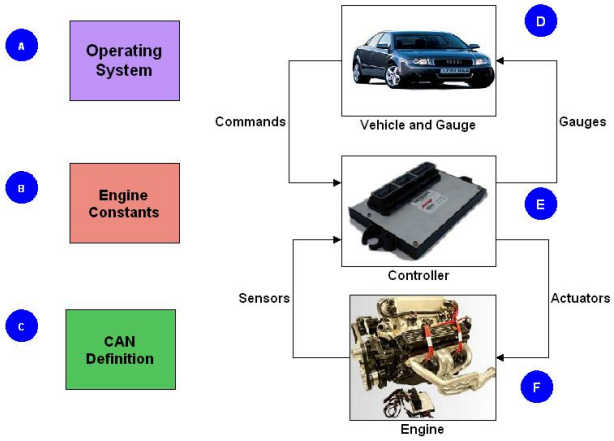

Fig. 1. EMS top level diagram.

B. *Engine Constants*

Within the engine constants block, global data are defined and can be used anywhere in the engine management system model. These constants include but are not limited to cylinder displacement, air-fuel ratio, cylinder firing order, cylinder firing angles, number of crank revolutions for each power stroke per cylinder, engine type (diesel or gasoline), etc. All of these constants can be adjusted or changed during the calibration process.

C. *CAN Definition*

The communications between driver command, gauges, calibration, and ECU can be realized by CAN or serial communications. The CAN definition, transmit blocks, and receive blocks set up all the communication parameters such as baud rate, transmit and receive queue size, CAN message definition, asynchronous or synchronous transmission, etc. This ensures accurate messages transmission among driver inputs, gauges, sensors, ECUs, and calibrations.

D. *Vehicle and Gauge*

The vehicle operator initiates control commands to the ECU, and the ECU calculates suitable control signals based on engine operation conditions and the structure of its control algorithms. Then the control signal is sent to actuators to perform the required task. The gauge gathers and displays, graphically, engine running parameters such as RPM, engine coolant temperature, manifold pressure, etc.

E. *Controller*

The engine controller can be partitioned into several subsystems: Virtual Sensors, Virtual Commands, Engine Control, Actuator Characterization, and Diagnostics. The engine controller system diagram is shown in Figure 2. The structure and function of individual subsystems are outlined below.

1) Virtual Sensors Subsystem: it receives the sensor and command signals, and then calculates the Virtual Sensor signals (such as engine state and normalized effective engine coolant temperature) based on the physical framework, logical framework and engine operating conditions.

The Virtual Sensor signals are used in all other engine controller subsystems.

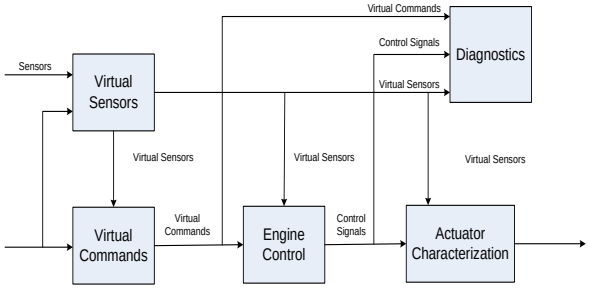

Fig. 2. Engine controller system.

2) Virtual Commands Subsystem: it is used to synthesize the commands from driver based on engine models and sensor signals. It is useful for the Engine Control subsystem and Diagnostics subsystem.

3) Engine Control Subsystem: combining Virtual Commands and Virtual Sensors signals, and based on engine models and control requirements, the Engine Control subsystem calculates suitable torque request, fuel, air, spark advance, spark energy and ignition timing accounting for engine configuration (naturally aspirated or turbocharged engines, engines with or without EGR) and different situations such as warm-up, cruise control, knock control and emission control. It may include [3, 5, and 9]:

- The Engine Torque Coordinator: its function is to simplify and integrate multiple torque requests (for example: driver request, idle control, cruise control and emission control may simultaneously and independently

make torque requests) placed on the engine as well as coordinate how that torque is produced (i.e. via air, fuel or spark). It is only available for engines equipped with an electronic throttle.

- The Air/Fuel Coordinator: similar to the engine torque coordinator, all mixture demands are coordinated in one air/fuel manager. Based on the operating conditions, a set of basic functions controls the air/fuel ratio within the physical limits.
- Cylinder Individual Knock Control: using knock sensors to detect the onset of detonation, retards spark timing on a per cylinder basis.
- Idle Speed Governor: including idle speed control, idle transition (entry and exit) control, idle spark advance and air control;
- Emissions Control: works to optimize emissions during cranking, start and after start. This enables the realization of various catalyst warm-up strategies.
- Additional customer defined functions as required.

The engine control subsystem signals are used in Diagnostics and Actuator Characterization subsystems as follows.

4) Actuator Characterization: including injector characterization, spark coils characterization, electronic throttle characterization, etc. It receives signals from Virtual Sensors and Engine Control subsystems and sends out signals to the engine actuators.

5) Diagnostics: including all the sensors, actuators and catalyst diagnostics. The Diagnostics subsystem receives signals from Virtual Sensors, Virtual Commands and the Engine Control subsystems, and interfaces with calibrations, SCAN-tools, and audio-visual devices.

F. Engine

In terms of engine control, the Engine portion of the model consists of Sensors and Actuators. The Sensors section receives electronic signals from the physical sensors or engine model at the hardware pins and then converts them into suitable engine units for computation in the Controller. It also includes signal conditioning and filtering. The Actuators section links the signal from Actuator Characterization subsystem to the correct hardware pins. The Sensors and Actuators can be in subsystems running at different update rates.

To be capable of controlling different engines with the same engine model, a universal encoder block, injector sequence block and spark sequence block are included in MotoHawk. The injector sequence sets up a sequence of injection pulses, starting with the specified pin and working up. It can accommodate up to 12 either high-impedance or low-impedance injectors. Similar to the injector sequence, the spark sequence sets up a sequence of spark pulses dynamically, starting with the specified pin and working up. The MotoHawk encoder block supports different Crank/Cam sensor combinations and the mechanical vs. electrical offset can also be adjusted at runtime. This means the same, already-developed, engine model and control algorithm can be code-generated once and the binary executable file can be used for different engines with different encoder types, number of cylinders, firing order, sensor inputs, and driver outputs. By doing this, the software development time is greatly reduced.

After introducing the basics of MotoHawk and the construction of an engine management system in this section, a fuel control strategy is proposed in the following section.

IV. Fuel Control Design

To meet stringent engine emissions requirements and increase fuel economy, a model-based adaptive and feedforward fuel control strategy is proposed for gasoline engines. The controller structure is shown in Figure 3.

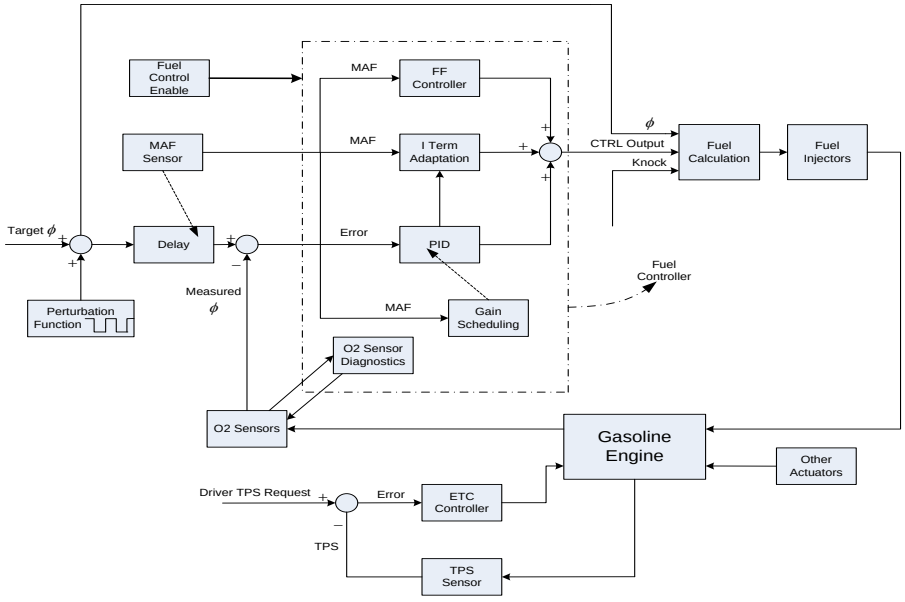

Fig. 3. Fuel controller structure.

The model-based design centers around a physical plant model. Being a feedforward-based design, the model is used to predict engine state based on current sensor inputs. This provides more responsive control. The adaptive portion of the design uses feedback signals from engine sensors and actuators to adapt the engine model thus minimizing the amount of control done in the feedback portion of the loop. These three parts are necessary for a stable and robust engine controller and are widely used in every major MotoHawk project.

The fuel control algorithm outlined below is integrated into the engine management system to apply the appropriate amount of fuel and determines the resulting air/fuel ratio. The fuel controller is comprised of five subsystems as shown in Figure 3:

1) Fuel Control Enable Subsystem: logic to decide whether the strategy is enabled or disabled..
2) Feedforward (FF) Control Subsystem: to deal with difficult engine transients;
3) Gain-Scheduled PID Control Subsystem: is provided to reduce instantaneous A/F errors;
4) Long-Term PID Integration-Term (I-term) Adaptation Subsystem: to adapt to changing engine parameters (such as combustion chamber and inlet tract deposits and oxygen sensor aging);
5) Oxygen Sensor Diagnostics Subsystem: for the indication of oxygen sensor operational status.

The control objective is to track the delayed perturbed target equivalence ratio to realize optimal emissions system performance. The stoichiometric equivalence ratio is first perturbed with specific amplitude and switching frequency. Then the perturbed equivalence ratio passes through a delay block. The error input to the PID is obtained by comparing the delayed, perturbed equivalence ratio with the measured equivalence ratio from oxygen sensors. The final output of the fuel controller is the sum of PID output, the feedforward control output and I-term adaptation. Then the output of the fuel controller goes to the fuel calculation block to compute the correct amount of fuel to inject. The air ingested by the engine is controlled by a separate electronic throttle controller.

Next, the control strategy development is explained in detail.

A. Fuel Controller Enable Logic

Fuel closed-loop control is enabled when all the following conditions are satisfied:

1) Fuel control enable flag is set via calibration;
2) Engine indicated work is above a calibrated value indicating oxygen sensor warm-up;
3) Desired equivalence ratio is within a calibrated range of stoichiometric;
4) There is no oxygen sensor or catalyst fault;
5) There is no other engine sensor or actuator faults;

B. Target Equivalence Ratio

It has been shown that commercially available Three-Way-Catalysts (TWC) show a significant improvement in conversion rates when the equivalence ratio oscillates around the nominal value of 1.0 [10-11]. The ideal equivalence ratio for the fuel controller is the nominal value of 1.0 oscillated with a calibratable switching period of T and amplitude α depending on the engine operating conditions, as shown in Figure 4.

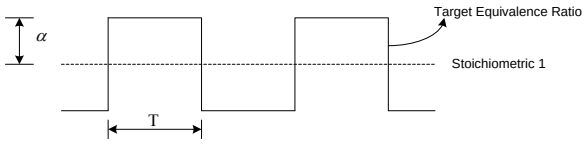

Fig. 4. Target equivalence ratio

Another key factor in successfully implementing the fuel control strategy is to include a pure measurement delay, which is composed of fuel transport delay plus the oxygen sensors' reaction time. The fuel transport delay consists of the time from when the fuel command is issued to the injector to the time the exhaust valve opens, and the time for the gas to move from the exhaust port to the oxygen sensor [12]. The pure measurement delay introduces a limitation on the system bandwidth and limits the transient accuracy of the system response. For this implementation, we not only introduce a feedforward term to make the controller act quickly, but delay the target equivalence ratio to match with the pure measurement delay. This is realized by passing target equivalence ratio through a measurement delay block before comparing it with the equivalence ratio signal from the oxygen sensors, as shown in Figure 3. The measurement delay block consists of a look-up table based on the mass of air flow sensor input. The idea is to use the mass air flow input instead of speed and load to reduce the calibration effort. The mass air flow rate can be obtained from a digital MAF sensor or via the speed-density calculation [13-14].

A lower MAF value usually means a lower speed and load, and larger measurement delay. So the transport delay block is able to take in account different engine operation conditions.

C. Feedforward Controller

The feedforward controller's reacts instantaneously to engine transients and the measurement delay. Based on the knowledge of engine behavior, combined with current engine operating conditions (speed and load), the feedforward controller 'predicts' a response that will allow the engine operate optimally during transient operation and reduce the reliance on the feedback, PID controller loop.

D. Gain Scheduled PID

To eliminate any instantaneous equivalence ratio error, a feedback PID controller is introduced. To adapt to different operating conditions (speed and load), a gain scheduled strategy is proposed as shown in Figure 3. Based on different values of mass air flow rate, the PID controller parameters are adjusted accordingly for better performance. The PID parameters are grouped in a calibrated table and stored in EEPROM.

E. Long Term PID I-term Adaptation

To reduce the PID controller loading and adapt to changes in engine characteristics, such as oxygen sensor aging, a long-term PID Integration-term (I-term) adaptation is included in the fuel controller structure. The principle used in this I-term adaptation is: with a well-calibrated, gain-scheduled, PID controller, plus a feedforward controller, the integration term of the PID controller should be small. If the engine characteristics change over time, the I-term will absorb any error caused by these changes. This "accumulated" error in the integration term will then be used to adapt part of the feedforward model, in this case the injector model. By doing so, the I-term in the PID controller will be continuously driven toward zero.

F. Oxygen Sensor Diagnostics

Oxygen sensor diagnostics consist of: enable logic, sensor voltage monitor, sensor response monitor, and diagnostics interface.

1) Oxygen sensor enable logic: the oxygen sensor voltage and response monitors are primarily enabled during fuel closed-loop control. The enable logic is duplicated for each monitor to provide different speed/load and enable conditions via calibration. The rationale for the different enable conditions is that the oxygen sensor must be cycling in a well controlled manner to perform some of the response tests, whereas this is not the case for the voltage tests.
2) Oxygen sensor voltage monitor: it detects a low or high voltage output from oxygen sensor during fuel closed-loop control;
3) Oxygen sensor response monitor: it detects either a slow sensor output transition or a slow-to-respond sensor;
4) Diagnostics interface: the fault manager contained in MotoHawk defines and manages the oxygen sensor fault and supports the OBD scan tool protocol.

V. EXPERIMENT

This section is to validate the proposed fuel control performance. The control objective is to have the measured equivalence ratio track the target equivalence ratio. In the section IV, the target equivalence ratio is a square wave as shown in Figure 4. Since there is no way to exactly track a step response, the target equivalence ratio of the square wave is passed through a low-pass filter, which results in a more realistic target equivalence ratio. The switching period T and amplitude α were set at 4 seconds and 0.04, respectively.

Two different scenarios are considered: 1) different operation regimes: the engine is operated at different speeds and loads to test the controller performance; 2) with or without delay compensation: the engine is operated at $RPM = 1000$ with measurement delay compensation or without delay compensation (both cases without load). In each scenario, target equivalence ratio and measured equivalence ratio are plotted together to show the tracking ability of the proposed fuel controller.

The engine is a GM V-8 of 5.0 liter displacement with an electronic throttle and a dynamometer to control engine load. Two NGK Universal Exhaust Gas Oxygen (UEGO) sensors are used to measure the equivalence ratio in the exhaust stream. The hardware platform is the MotoTron MPC565-based 128-pin module [7]. ECU565-128 is operated at 56MHz and has dual CAN 2.0B data-links, one of which is reserved for communication to the calibration tool. In addition there is one RS485 serial bus and one ISO 9141 channel. This ECU can accommodate up to 34 analog inputs, 8 low frequency digital inputs, two wide-range oxygen sensor inputs, and two wide-band knock sensors. It also has 12 injector drivers, three H-Bridge drivers, up to 10 low-side PWMs, and one main power relay driver.

In following each plot, the horizontal axis is time and represents a 20 second period, the vertical axis is equivalence ratio and ranges from 0.95 to 1.05. The solid and curved lines are the target and measured equivalence ratio, respectively.

A. Different Engine Operating Regimes

In this scenario the controller performance is tested at different engine operating points (speeds and loads). The measured equivalence ration and target equivalence ratio are recorded for comparison. Figure 5 shows the controller performance at $RPM = 1110$ and a brake torque of $44.5\ N \cdot m$. At this operating point, $\dot{m}_a = 20.39\ g/s$, manifold pressure is $50.8\ KPa$, throttle position (θ) is $8.76\ \%$. The measurement delay compensation is 0.24 seconds.

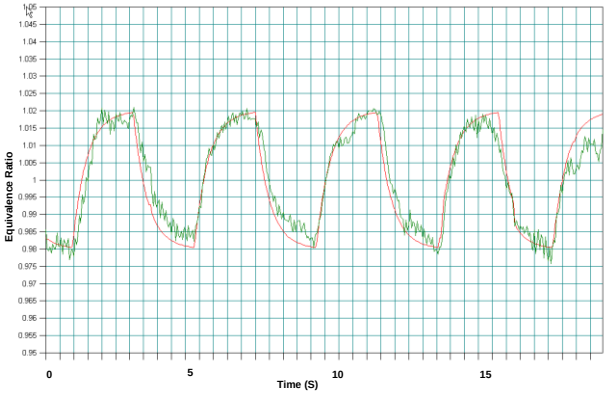

Fig. 5. Controller performance at low speed and load.

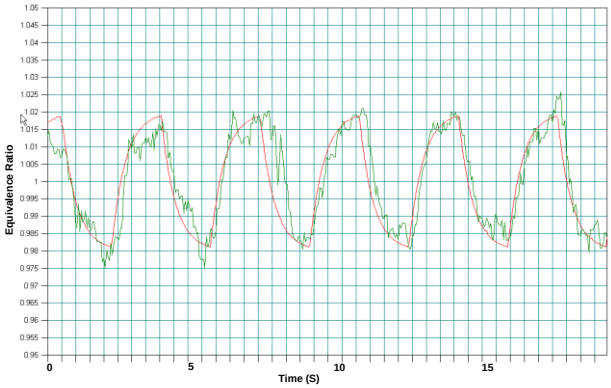

Fig. 6. Controller performance at high speed and load.

Figure 6 shows the controller performance at $RPM = 1220$, a brake torque of 162.2 $N \cdot m$, $\dot{m}_a = 32.3\ g/s$, a manifold pressure of $MAP = 68.6\ KPa$, and throttle position of $\theta = 13.5\,\%$. The measurement delay compensation is 0.14 seconds.

Comparing the measurement delay at two cases, an important conclusion can be made: as expected, the measurement delay at a high speed and load point is less than that at a low speed and load point.

Obviously, both plots show the fuel controller to have good performance under different engine operating conditions.

B. *With or Without Delay Compensation*

The delay compensation block is enabled or disabled in order to compare the controller performance with or without delay compensation. Figure 7 shows the controller performance at $RPM = 1000$, $\dot{m}_a = 11.1\ g/s$, manifold pressure of $MAP = 36.6\ KPa$, throttle position of $\theta = 5.1\,\%$. The measurement delay compensation is 0.34 second. From Figure 7, the measured equivalence ratio tracks the target very well. Figure 8 shows the controller performance under the same engine operating conditions and controller configuration except that the delay compensation subsystem is disabled. Figure 8 confirms that there is a pure delay between the measured signal and target signal.

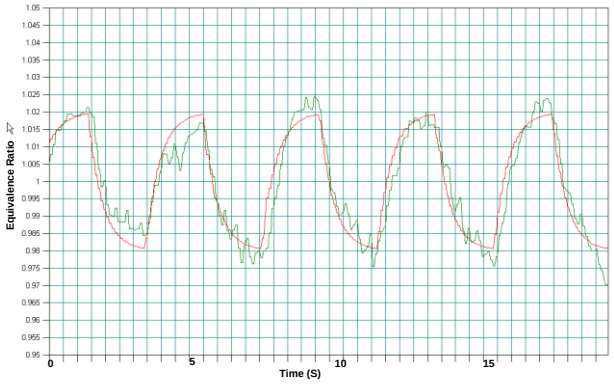

Fig. 7. Controller performance with delay compensation.

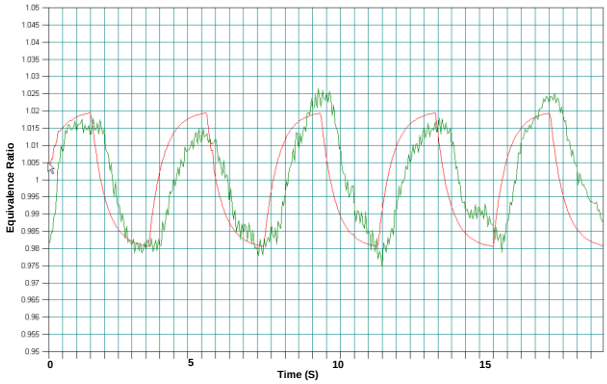

Fig. 8. Controller performance without delay compensation.

VI. Conclusion

This paper presents an ECU-based rapid prototyping system, MotoHawk, in this case used for engine management control system design, calibration, and development. It has the benefits of simpler and faster software development through the use of production ECU hardware in development, a transparent environment for software/system integration which reduces validation time and cost, and extensibility to a flexible family of products. Engine fuel control was discussed as an illustrative example of a MotoHawk application.

Acknowledgement

The authors would like to thank Eric Bradley at MotoTron Corp. for the development of the MotoHawk concept, Aaron J. Ward at MotoTron Corp. for implementing and testing parts of the fuel control strategy and Mark Frank and Patrick McCarthy at FEV Engine Technology Inc. for many helpful discussions and suggestions regarding the development of the fuel control strategy.

References

[1] C. Cao, D. Shull, and E. Himes, "A model-based environment for production engine management system (EMS) development," *SAE* paper No. 2001-01-0554, 2001.

[2] S. Furry, and J. Kainz, "Rapid algorithm development tools applied to engine management systems," *SAE* paper No. 980799, 1998.

[3] J. Gerhardt, H. Honninger and H. Bischof, "A new approach to functional and software structure for engine management systems – Bosch ME7," *SAE* Paper No. 980801, 1998.

[4] R. Lawrie and M. McKenna, "An architecture based design process for deploying control software on production hardware using MotoHawk," *SAE* Paper No. 2003-02-0853, 2003.

[5] B. Mencher, H. Jessen, L. Kaiser, and J. Gerhardt, "Preparing for CARTRONIC – interface and new strategies for torque coordination and conversion in a spark ignition engine-management system," *SAE* Paper No. 2001-01-0268, 2001.

[6] J. D. Naber, E. K. Bradley, and J. E. Szpytman, "Target based rapid prototyping control system for engine research," *SAE* paper No. 2006-01-0267, 2006.

[7] MotoTron Corp., web site, www.mototron.com.

[8] MotoHawk web site, www.motohawk.info.

[9] *Bosch Gasoline Engine Management Handbook*, Robert Bosch GmbH, 2004.

[10] L. Padeste, *Three-Way Catalysts in a Hybrid Drive-System: Experimental Study and Kinetic Modeling*, ETH-Dissertation No. 10515, Swiss Federal Institute of Technology (ETH), 1994.

[11] L. Guzzela, "Models and model-based control of IC-engines - a nonlinear approach," *SAE* Paper No. 950844, 1995.

[12] J. L. Kainz and J. C. Smith, "Individual cylinder fuel control with a switching oxygen sensor," *SAE* Paper No. 1999-01-0546, 1999.

[13] B. R. Suhre, *MotoTron Engine Control and Calibration Basics*, MotoTron technical report, 2005.

[14] J. B. Heywood, *Internal Combustion Engine Fundamentals*, McGraw-Hill Book Company, 1988.