Computational Aspect Analysis of Real-Time Systems: Application to HDD Servo Control

Kenta Morishima^{*a)} Non-member, Shigeru Oho^{**} Member

(Manuscript received May 14, 2018, revised June 27, 2018)

Co-simulation for a hard disk drive (HDD) servo system and its microcontroller control system was developed to analyze computational aspects of the control system such as memory use and calculation load on processors. The microcontroller was modeled with its instruction set simulator that executed the binary codes of control commands. The model also incorporated the peripheral functions of timer units, an interrupt controller, an analog-to-digital converter, and so forth. The plant model of the HDD mechanism and its control algorithms were adopted from a study on HDD servo controls. The microcontroller and HDD models communicated with each other periodically to realize co-simulation. The control algorithms, which were originally written in Matlab®, were converted to C codes and eventually to binary codes by an automated code generation tool to eliminate manual programming and to guarantee fair comparison. The proposed co-simulation method compared different control algorithms, as well as different microcontrollers, for selection. The accuracy of the co-simulation was investigated, and the potential use of co-simulation approach was discussed. A synchronization adaptor was proposed to improve the timing accuracy of the control system simulation.

Keywords: co-simulation, computational aspect, real-time control

1. Introduction

In the development of embedded control systems, the control design to achieve the required performance is often of primary interest. The microcontroller resources required for carrying out the designed control schemes are also of great concern for product-design engineers in industries. Real-time execution of large and complex control operations often becomes a difficult task to achieve due to practical restrictions in available microcontrollers. Thus, control designers also need to pay attention to the computational load of microcontrollers, processing time of the control algorithms, use of computer memories, and so forth. These computational aspects cannot be evaluated quantitatively with the currently available control design tools, such as Matlab® or Modelica®. To analyze the computational aspects of control execution, the microcontroller needs to be specified first, and then the control algorithms must be converted into the implementation code in binary format and run on the chosen microcontroller. Conventionally, real-time codes are analyzed experimentally by externally monitoring microcontroller operations. Because microcontrollers have limited bandwidths at monitoring channels, a full spectrum measurement on computational issues is often difficult.

For example, the motion control of a hard disk drive (HDD)

head is a typical embedded real-time system. There have been numerous studies on control methods to improve performances of HDD controls^{(1)–(5)}. A group of control engineers at IEEJ set the benchmark problems for HDD head seeking and following controls and studied more than a half dozen control strategies, comparing their performances. They published the results of their research and made the source codes of the control simulation open to the public⁽⁶⁾⁽⁷⁾. However, the computational requirements were not the scope of this study. Atsumi⁽⁸⁾ replaced a look-up table with a polynomial in order to reduce memory use in the HDD head control. To curtail numerical operations, Jia⁽⁹⁾ applied a multi-rate control scheme to the HDD control. These studies, however, did not quantitatively evaluate the usage of computational resources.

Recently, microcontroller models have been applied to the analyses of embedded control systems⁽¹⁰⁾. These microcontroller models simulate the behavior of actual microcontrollers. Sugure⁽¹¹⁾, for example, used a microcontroller model for an automotive engine controller and evaluated the computational load of a frequency spectrum analysis to detect engine knocking.

Control plants, such as HDD mechanism or automotive engines, are physical and continuous systems, while microcontrollers are logical and discrete systems. Therefore, it is necessary to realize the co-simulation of continuous and discrete systems in order to utilize the microcontroller models for control system analysis. However, the accuracy of the computational aspect analysis depends on the accuracy of the microcontroller model and the method of co-simulation.

In this paper, we analyze the implementation code of the HDD following control and evaluate quantitatively the computational load, processing time, and memory use. The

a) Correspondence to: Kenta Morishima. E-mail: Kenta.morishima.pv@hitachi.com

* R&D Division, Hitachi, Ltd.
1-280, Higashi-koigakubo, Kokubunji, Tokyo 185-8601, Japan

** Nippon Institute of Technology
4-1, Gakuendai Miyashiro-machi, Minami-Saitama, Saitama 345-8501, Japan

control schemes and the plant model for the HDD system provided by Yamaguchi⁽⁶⁾⁽⁷⁾ are utilized in this study. First, the microcontroller model and HDD servo control are introduced. The HDD control algorithms to be examined are converted into the implementation codes, and the co-simulation of the HDD plant and the microcontroller is then ready. Next, the results of the co-simulation are presented, and the computational aspects of HDD control schemes are summarized. The accuracy of the co-simulation is discussed in terms of numerical operation and timing jitter. A synchronization adapter is proposed to improve the timing accuracy of the co-simulation.

2. Co-simulation of HDD Disk-Head Control

2.1 Computational Aspect Analysis Using Microcontroller Models When control engineers analyze control systems, they often carry out a continuous system simulation with plant and controller models, and they examine how their control algorithms work, as shown in the upper part of Fig. 1. In actual systems, however, the controller is often built with a microcontroller, and control algorithms are executed as their implementation codes. Thus, realistic control systems are often a hybrid system of continuous dynamics and discrete events. In this study, we implement the control algorithms into a microcontroller model and run a co-simulation of continuous and discrete systems, as shown in the lower part of Fig. 1.

The microcontroller model is simulated by an instruction set simulator (ISS). If an ISS represents instruction cycles correctly, the microcontroller model is referred to as cycle-accurate. By using a cycle-accurate microcontroller model, execution time of control commands can be measured by accumulating each cycle of instructions and data transfer to/from a peripheral circuit. Logged data of memory accesses yield the required amount of data storage. Models of a timer circuit and an interrupt controller allow us to evaluate the overhead of real-time operating systems for task switching.

2.2 Microcontroller Model In this study, we used Matlab® for the control plant model and Synopsys CoMET-METeor® for the microcontroller model. As the microcontroller for control execution, the Renesas SH7251, which has integrated a built-in floating point unit (FPU), was chosen as

a representative case. Later, the SH7251 was replaced with an ARM926 and Renesas V850 for comparison.

The configuration of the microcontroller model is shown in Fig. 2. It incorporates the central processing unit (CPU), the read only memory (ROM), the random access memory (RAM), and numerous other peripheral functions, such as digital input/output (DIO), interrupt control unit (INTC), timer unit (TU), and analog-to-digital converter (ADC).

The control program is stored in ROM in binary format. The CPU fetches the implementation code in ROM and executes it step-by-step. The contents of the RAM, where control variables are kept, and peripherals are updated accordingly with the result of the CPU operation. The TU generates an interrupt request periodically at the beginning of each control period, and the INTC initiates a specified control program. Analog signals from the control plant are supplied to the ADC and sent to the CPU through the Bus Bridge. Executing the control instructions, the CPU calculates control output signals, and sends them to the control plant through the DIO.

To realize co-simulation with the plant model, the plant model and the microcontroller model are connected via the bi-directional data transfer path (Matlab® Interface). To synchronize the progress of each simulation of the plant and microcontroller, the data exchange is time-stamped. There is no exact rule for the frequency of the data exchange, or synchronization, in the co-simulation system. We empirically set the synchronization period at an integer fraction of the control period. Obviously, there is a trade-off in the choice of the synchronization period; frequent synchronization decreases the timing error of co-simulation, but increases the simulation time.

The operations of the microcontroller model described above mimic the actual behavior of the microcontroller. The pulse input/output signals, however, are often replaced with their duty cycle values, as exact simulation of pulse waveform is much too time-consuming.

2.3 HDD Disk-Head Plant Model and Its Following Control

The configuration of the HDD disk-head control system is shown in Fig. 3⁽⁶⁾⁽⁷⁾. The magnetic data on the disk are recorded in concentric circles. Besides the recorded data, the concentric sectors formed on the disk also provide the location information so that the disk head can track them. The demodulator circuit extracts the positional data for the servo operation from the preamplifier signal. The servo controller inputs the decoded position signal, calculates

Fig. 1. System simulation and co-simulation with microcontroller mode

Fig. 2. Microcontroller model

Fig. 3. Configuration of disk-head control system⁽⁶⁾

Fig. 4. Disk-head following control simulation

Fig. 5. HDD feedback control simulation using execution codes

positioning errors of the magnetic header, and activates the voice coil motor (VCM) through the driving circuit to achieve accurate tracking of the disk head. The servo control system for the HDD head following is shown in Fig. 4. A disturbance noise source is included in the feedback control system to analyze the dynamic response to a mechanical vibration or shock.

Yamaguchi⁽⁶⁾⁽⁷⁾ compared the performances of such head-following schemes as PID, phase stabilizing, state feedback, and H-infinity controls. These control algorithms were written in Matlab®, and formatted in state-space (state feedback) and transfer-function (PID, phase stabilizing, and H-infinity) models. For the details of modeling, refer to the literature. The control period of these algorithms are commonly set at $18.9\mu\text{s}$. In this study, we use these control algorithms as the test cases.

2.4 Control Code Generation The control algorithms written in Matlab® were converted to C-codes by using Mathworks's automatic code generator, and then to the implementation codes in binary format with the compiler tools (Renesas HEW for SH7251 and GNU Compiler Collection for ARM926). Timer input and output routines and interrupt handling software were created manually for real-time operation. Note that the manual code generation was minimized for fair comparison of the computational aspects.

2.5 Co-simulation of Disk-Head Responses with the Microcontroller Model The controller for the HDD head following control in Fig. 4 was replaced with the microcontroller model (see Sec. 2.3), as shown in Fig. 5. The execution codes generated in the previous section were stored in the ROM of the microcontroller model. As shown in Fig. 5, a disturbance noise is injected into the control output signal to create a case of mechanical shock. Then, we are ready to run

the co-simulation of continuous and discrete systems of the HDD head following control. The results will be reported in the following sections.

3. Comparing Computational Aspects

3.1 Comparing Feedback Control Performance In Fig. 5, a single pulse with a duration of 10 ms was periodically injected as the disturbance noise, and co-simulation was carried out for PID, phase stabilizing, state feedback, and H-infinity control schemes. The results shown in Fig. 6 compare the performance of the control schemes examined here. Note that the results were almost identical to the ones of the original Matlab® simulation. (They are not shown here.) Here, we just confirm that the co-simulation is as accurate as the original one, and do not go into the detailed discussion on control performance issues. Refer to Yamaguchi⁽⁶⁾⁽⁷⁾ for performance comparison.

3.2 Comparing the Use of Microcontroller Resources

In Table 1, the required amount of ROM and RAM sizes and the CPU processing time (cycles) for each of the control schemes are summarized. Note that the numerical data used for the CPU processing was expressed in the floating point format with single precision accuracy. Due to the hardware constraint of microcontrollers, the single precision format was chosen, though Matlab® simulation set the double precision accuracy as the default setting. It is interesting to see that the PID control scheme may not be the best choice in view of calculation cost for this particular application, though it is very popular in many industrial systems.

The CPU cycle data also yield an accurate measure on the microcontroller loading in real-time control. For a given sampling period of $19.6\mu\text{s}$ and a clock frequency of 676 MHz, a cycle number of 3848 steps (PID) translates to a $5.69\mu\text{s}$ net execution time and 29% CPU loading. To ensure reliable controller operation, there is usually an upper limit on the CPU loading. With the co-simulation analysis, one can evaluate precisely how much time margin is left in the controller.

The co-simulation therefore yielded quantitative measures on the computational aspects of control. In practical system developments, control engineers often need to compromise on performance and functions due to limited computer resources. The co-simulation results should help them make the right decisions.

3.3 Comparing Choice of Microcontrollers

In the same co-simulation reported in the previous section, the type of microcontroller was replaced for comparative study. Here, the PID control scheme was used as the representative case. The microcontrollers used here were the SH7251 with and without FPU, ARM926, and V850. Though the same control program and plant model are commonly used in this co-simulation, the C compilers used for binary code generation are different. We chose a typical compiler for each microcontroller, and code efficiency can be dependent on the C compiler chosen. Therefore, the results summarized in Table 2 can vary accordingly with the choice of C compiler. Even so, we believe this is still a reasonably "fair" comparison, for we regard the compilers as a part of microcontroller systems.

The required ROM and RAM sizes shown in Table 2 are useful numbers for control engineers to choose an appropriate microcontroller. Another piece of important information

Fig. 6. Co-simulation results of HDD head response to a disturbance

Table 1. Comparing control schemes with respect to processing time and memory use

Control schemes	ROM [byte]*1	RAM [byte]*2	CPU *3 cycle (step)
PID	17684	1340	3848
Phase stability	16888	1188	2307
State feedback	21936	1436	5017
H-infinity	17516	1347	4326

*1) Exclude the initialization routine

*2) Exclude the stack area for program flow control

*3) CPU clock cycle = 676.1 MHz

is the CPU cycles to execute the PID control. In this case, the ARM926 needed about 40% more CPU cycles than the SH7251. Note that the results included the efficiency of the compilers used for binary code generation and may not accurately reflect the performance difference of the two CPUs.

3.4 Microcontroller Overload

Feedback control can

Table 2. Comparing microcontrollers in terms of processing time and memory use

Microcontroller	ROM [byte]*1	RAM [byte]*2	CPU *3 [cycle]	CPU Clock [MHz]
SH2A-FPU[6]	17684	1340	3848 *7	676.1
SH-2A 32-bit fixed-point	23817	1543	61294 *4 *7	3381
ARM926	48304	2088	5624 *5	676.1
V850E2M	37466	1784	11281 *5 *6	1000

*1) Exclude the initialization routine

*2) Excluding the stack area

*3) CPU clock cycle = 676.1 MHz

*4) CPU clock cycle = 3381 MHz

*5) GCC compiler

*6) CPU clock cycle = 1000 MHz

*7) Renesas compile r (HEW)

collapse due to microcontroller overload. Unless the numerical control operation is completed in time in every innovation step of digital control, feedback control can become erroneous and fail. This is also a very important computational aspect, and the co-simulation method allows us to analyze the collapse of the control operation. In the co-simulation reported in the previous section, the microcontrollers' system clocks were set very high so that the microcontrollers could carry out the required control operation in time and establish stable control.

In the case of the SH7251 co-simulation, a normal response of feedback control on the HDD head was observed, as shown in Fig. 7(a), where an FPU was used to accelerate the numerical operation. When the numerical operation was carried out with a 32 bit fixed-point processor (without FPU), the feedback control failed, as shown in Fig. 7(b), because the processor was too slow to complete the control calculation in time. To recover the control system to normal, the system clock was increased to 3381 MHz, a very unusual number for real microcontrollers.

The abovementioned cases clearly demonstrate a big advantage of microcontroller co-simulation for control system analysis in implementation designs. One can precisely examine the malicious effect of degradation in computational performance in control systems.

4. Accuracy of Co-simulation

4.1 Accuracy issues of Co-simulation As reported in the previous section, the continuous and discrete system co-simulation clarifies the computational aspects of control systems. To utilize the co-simulation for practical product designs, however, we need to verify its accuracy and look into the causes of errors and possible fixes.

Plant simulation errors are largely dependent on the physical modeling of the hardware systems, namely, modeling the engineers' understanding on the physical system. Microcontrollers can be modeled highly accurately, in principle, to include all the digital components. In fact, modern microcontrollers are usually designed with accurate models at semiconductor vendors for their own simulation. These accurate models are, however, usually not available, nor useful for practical control system development; they are essentially

Fig. 7. Collapse of feedback control due to microcontroller overload

overkill and too slow to run for control system simulation. Therefore, the microcontroller models for control applications are often abstracted for simplification. The abstracted microcontrollers can cause certain errors in computational aspects analysis. Nonetheless, a quantitative estimate on the use of microcontrollers in control systems would be valuable for practical designs.

Apart from the modeling accuracy, the co-simulation of microcontroller and plant models has its own issues: the accuracies of numerical operation and timing. In the following sections, we discuss these issues.

4.2 Accuracy Issues of Numerical Operation At first glance, the co-simulation seems almost identical to its original Matlab® simulation in most of the cases reported in Sec. 3.1. A close-up of the response traces, however, shows a subtle difference between them. Figure 8 depicts a close-up view of the steady-state response to a single pulse disturbance in the PID control. There can be observed a slight difference in the steady-state offset. This is due presumably to the difference in the accuracy of numerical operations; in the control system simulation with Matlab®, numerical calculation is often carried out in double precision floating point accuracy for both control algorithms and plants. In the real world, microcontrollers usually have a single precision FPU, and the models of microcontrollers also, of course, have single precision

Fig. 8. Close-up of Fig. 6(a) steady-state error

Fig. 9. Delay in feedback control cycle close-up of Fig. 6(a)

accuracy. Therefore, it is deemed that the accuracy difference in the controller's numerical operation can accumulate to the difference in the steady state.

4.3 Timing Jitter in Co-simulation and Its Correction

In the co-simulation, the microcontroller simulation needs to be periodically synchronized with the plant simulation. The control period is set at $19.6\mu\text{s}$ in this case study of HDD head following control, and the synchronization period is empirically set at one-tenth of the control period.

The discrete part of the co-simulation can cause a timing jitter, due to a one-step delay in synchronization, and result in a significant difference in the time response. As shown in Fig. 9, the response trace of co-simulation is one-step delayed from that of the original Matlab® simulation. The change in the controller output in the current synchronizing session was transmitted to the control plant in the next session. Then, the control plant started to respond to the change after the one-step delay.

The delayed response in the co-simulation is deemed harmless if the synchronization period is much shorter than the time constant of the control feedback loop. However, some may feel that the timing jitter degrades the fidelity of co-simulation and find it less trustable. Therefore, we delve into the generation mechanism of the timing jitter and try to find a fix for this phenomenon.

Referring to Fig. 10, the progress of the feedback control loop simulation in Fig. 5 is illustrated in Fig. 11 and described in detail below.

Fig. 11(a)

In the case of the Matlab® only simulation, (1) the

Fig. 10. Cause of the timing jitter in the co-simulation

Fig. 11. Time chart showing the cause of the timing jitter

controller first detects the change in the error signal, (2) then it calculates the control outputs according to its specified control scheme, and (3) its output is fed to the control plant, i.e., the VCM. The above steps of (1)–(3) are repeated periodically in every sampling period, which is specified to be $19.6\mu\text{s}$ in this particular case.

Fig. 11(b)

In the case of co-simulation, (4) the microcontroller first detects the change in the error signal, (5) then it executes the control program which realizes the specified control scheme, and calculates the control outputs. The control program is triggered at every sampling period of $19.6\mu\text{s}$ by using a timer counter interrupt which is integrated in the microcontroller. (6) Finally, the output is fed to the control plant.

Fig. 11(c)

As the control program execution is triggered by the hardware timer model and is independent from the sampling period, it can start at any time in the sampling period. In the case of Fig. 11(b), the control program execution is completed in the same control step as in Fig. 11(a). However, in Fig. 11(c),

Fig. 12. Synchronization adaptor

Fig. 13. Synchronization adaptor timing chart

the control output is not updated until the next control step, due to the late trigger timing. Therefore, the updated control output is fed to the control plant with a one-step delay, causing a timing jitter as shown in Fig. 9.

To correct the timing jitter in the co-simulation, we proposed the method of synchronization adaptor shown in Fig. 12. The synchronization adaptor is a pair of circuits: a synchronization pulse generator and an external interrupt port. The synchronization pulse is generated by the plant model simulator at the beginning of each sampling period. The pulse is fed to the external interrupt port and initiates the control program. Thus, the two simulators are synchronized at the beginning of the sampling period.

The timing chart with the synchronization adaptor is shown in Fig. 13. Because the synchronization pulse is transferred to the microcontroller in the next synchronization period, the timing jitter is eliminated, as shown in Fig. 8.

5. Conclusion

In this study, we analyzed the computational aspects of an HDD head following control as an example case. Co-simulation of the continuous control plant and discrete microcontroller was introduced. The microcontroller model included the CPU, which ran implementation codes of control programs, and many other peripheral functions, such as timer units, interrupt controllers, and analog interfaces. The co-simulation results detailed the memory use and computational load for each of the control algorithms examined. Different types of microcontrollers were tested by the co-simulation for a comparative study. One can select an appropriate microcontroller for a particular control application. The accuracy of the co-simulation was discussed in terms of numerical operation and timing jitters. To correct any timing issues of the co-simulation, a synchronization adaptor approach was proposed and verified. Overall, the proposed co-simulation method was proved to be useful for

Matlab® a trademark of the MathWorks, Inc.

Modelica® is a trade mark of the Modelica Design Group

CoMET-METeor® is a trade mark of the Synopsys, Inc.

computational aspect analysis of real-time control systems.

The timing accuracy, jitter elimination, and synchronization of multiple simulators in co-simulation are still unresolved issues in the proposed real-time system analysis approach. Establishing a reliable co-simulation environment is still an enormous challenge for control application engineers. The proposed co-simulation technique needs to be further examined by control designers and real-time system integrators to draw the maximum benefit from the continuous and discrete hybrid simulation technology.

References

- (1) M. Hirata, M. Takiguchi, and K. Nonami: "Track-Following Control of Hard Disk Drives Using Multi-Rate Sampled-Data H_∞ Control", Proc. of the IEEE Conference on Decision and Control, Maui, Hawaii USA (2003)
- (2) K. Ohno and S. Hara: "A Temperature-Dependent Multi-Rate Robust Controller for Hard Disk Drives", *SICE Transactions on Industrial Applications*, Vol.5, No.9, pp.57–65 (2006)
- (3) J. Zheng, M. Wang, and C. Du: "Nonlinear Tracking Control for a Hard Disk Drive Dual-Stage Actuator System", *IEEE/ASME Transactions on Mechatronics*, Vol.13, No.5, pp.510–518 (2008)
- (4) T. Atsumi: "Emerging Technology for Head-Positioning System in HDDs", *IEEJ Journal of Industry Applications*, Vol.5, No.2, pp.117–112 (2016)
- (5) T. Atsumi: "Estimation Method of Unobservable Oscillations in Sampled-Data Positioning Systems", *IEEJ Journal of Industry Applications*, Vol.6, No.3, pp.199–204 (2017)
- (6) T. Yamaguchi, M. Hirata, and H. Fujimoto: "Nanoscale Servo Control", *Tokyo Denki University Press* (2007) (in Japanese)
- (7) T. Yamaguchi, M. Hirata, and C.K. Pang: "Advances in High-Performance Motion Control of Mechatronic Systems", CRC Press (2013)
- (8) T. Atsumi: "Shock-Response-Spectrum Analysis of Sampled-Data Polynomial for Track-Seeking Control in Hard Disk Drives", Proc. of the IEEE International Workshop on Advanced Motion Control, pp.240–247 (2008)
- (9) Q.W. Jia: "New Design of Multirate Digital Control Scheme for Computation Saving", *IEEE International Conference on Industrial Technology*, pp.1–4 (2008)
- (10) M. Ishikawa, D.J. McCune, G. Saikalis, and S. Oho: "CPU Model-based Hardware/Software Co-design for Real-Time Embedded Control Systems", *SAE World Congress 2007-01-0776* (2007)
- (11) Y. Sugure, S. Oho, S.S. Phatak, and G. Saikalis: "Virtual Engine System Prototyping with High-Resolution FFT for Digital Knock Detection Using CPU Model-Based Hardware/Software Co-simulation", *SAE World Congress 2009-01-0532* (2009)

Kenta Morishima (Non-member) graduated from Tokyo university of Agriculture and Technology in 1990, received B.E. and received M.E. in 1992. He joined Hitachi, Ltd. in 1992 and then worked at central research laboratories. Now he works at Research & Development Group, Hitachi, Ltd., Tokyo, Japan. Morishima is a member of SAE.

Shigeru Oho (Member) graduated from National Institute of Technology, Sasebo college and received BE from University of Electro-Communications, M.E. and Ph.D. from Tokyo Institute of Technology in 1976, 1978, 1980, 1997, respectively. He joined Hitachi, Ltd. in 1980 and worked at its R&D divisions in Japan and the USA. From 1986 to 1987 he was a visiting scientist at Massachusetts Institute of Technology. Since 2012 he has been a professor at Nippon Institute of Technology. Dr. Oho is a member of IEEJ, IEICE, SICE, ISCIE, IEEE, SAEJ, and SAE.