Simulating Hybrid Dynamic Systems

By C.A. RABBATH, M. ABDOUNE, J. BELANGER, and K. BUTTS

This article presents the problem of inaccuracy involved in the conventional simulations, at fixed time steps, of event-based systems and then proposes the use of a new Simulink toolbox, known as the RT-Events Blockset, to significantly reduce the error between simulated and actual event-based systems. We will present the conventional techniques used in the fixed step-size simulations of the basic integrator block with reset performed in between the iteration steps. Among the issues discussed is the adverse phenomenon known as reset walk. We also propose a new set of tools in the form of the RT-Events Blockset. The important blocks of the RT-Events Blockset are succinctly explained, and three numerical examples illustrating the effectiveness of the new simulation tools are given, including a closed-loop V-6 engine system.

Background

Today's simulation engineers are faced with the task of simulating complex dynamic systems. The high level of complexity may come from the very nature of certain continuous-time and discrete-time systems that have a changing dynamical behavior depending on the occurrences of so-called discrete events. These systems are named event-based systems in this article. They can be considered as a class of hybrid dynamic systems [1]. A practical example of an event-based system is the internal-combustion (IC) engine of an automobile. Such a system is characterized by the crankshaft angle of the engine which determines one of four cycles; namely intake, compression, expansion, and exhaust. Thus, in-cylinder dynamics are determined by the crankshaft event. In the development of a power-train system, simulation engineers usually perform a simulation of the engine dynamics in a closed-loop with a simulated control unit and eventually carry out a hardware-in-the-loop (HIL) simulation of the simulated engine dynamics in a closed-loop with the actual electronic control unit (ECU). However, the events associated with the crankshaft angle can take place between the fixed time steps at which the simulation is executed. This causes a discrepancy between the actual event occurrence and the discrete-time detection of the event by the simulated system. Consequently, there are differences between the simulated and the actual engine dynamics. It was shown in [2] that the error between simulated and actual dynamics can in fact render real-time engine simulations untruthful and misleading.

Several problems are involved with fixed step simulations of event-based systems; for instance, the process of integrating a continuous signal with reset of the integrator done at time instants other than integral multiples of the fixed step size. In this case, there will be errors between the output of the simulated integrator and that of the theoretical integration because the simulated integrator can be reset at integral multiples of the fixed step size, and not at the time instants at which the theoretical integrator is reset. However, keeping in mind that the simulations should approximate as closely as possible the behavior of the actual integrator, the discrepancy between simulated and

1070-9932/02/$17.00©2002IEEE

actual integrators creates a problem of inaccuracy. One approach to solve the problem of inaccuracy is to use steps of variable sizes. A simulation using variable step sizes is based on an algorithm that selects a step size according to signal tolerances, disregarding time-related constraints. This sort of scheme has the disadvantage of possibly resulting in time-consuming simulations. Furthermore, if one is interested in assuring compatibility of the simulations with hard real-time constraints, the variable step-size solution has to be discarded. That is, with an eventual connection to and synchronization of input/output (I/O) boards and to a timer card for HIL simulations, a fixed step size must be selected. Another approach to the simulation of event-based systems that yields a relatively small discrepancy between the simulated system and the actual (theoretical) one is to utilize a relatively short, fixed step size. Yet, the computing power of the target processor could limit the effectiveness of the simulation. In the case of complex event-based systems, it is possible that infinitesimally small step sizes be necessary to achieve successful simulations.

Conventional Simulation Techniques

The basic process taking place in dynamic systems simulation is signal integration. Knowing the limitations associated with the simulation of a reset integrator is helpful in understanding the intricacies involved in the simulation of a large set of event-based dynamic systems. Some of the conventional fixed step reset integrator simulation techniques are presented here, followed by an explanation of the problem known as reset walk. Furthermore, the limitations of the conventional simulation techniques to accurately simulate the latch operation, or the aperiodic sample-and-hold, are discussed.

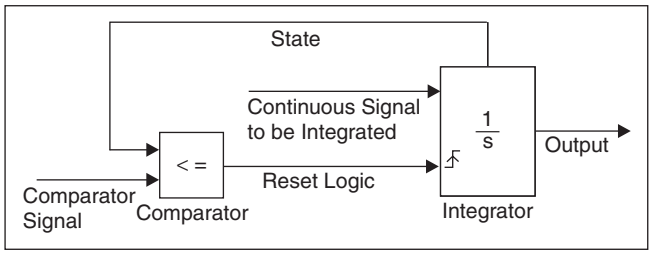

Figure 1. *Model of a reset integrator.*

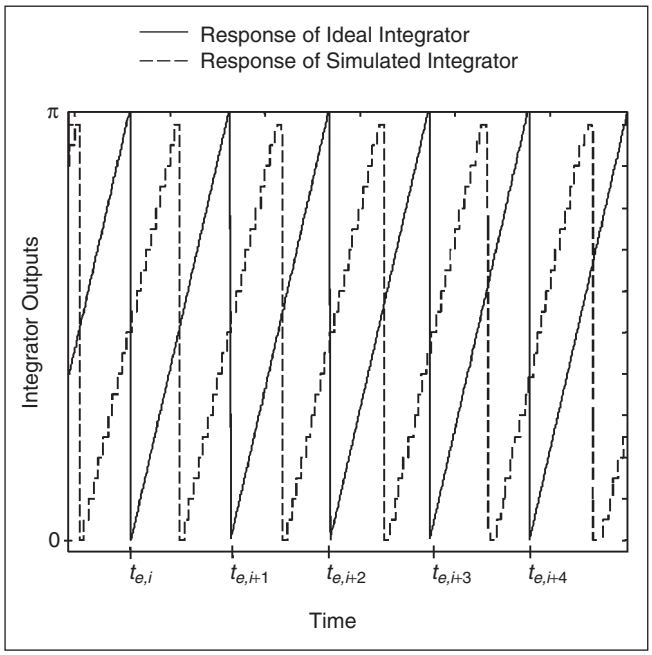

Figure 2. *Integrator outputs.*

Integrator Simulation

The most widely used techniques to simulate the integration of a continuous signal can be divided into two categories: those using a scheme to approximate the integration based on the evaluation of n points at each iteration step, such as the Runge-Kutta methods [3], and those calculating a discrete-time transfer function to approximate the continuous-time transfer function of the integrator, i.e. $1/s$, such as the Forward and Backward Euler methods and the Trapezoidal technique [4], or even the multistep methods [5]. The latter methods are not only used in the field of simulation but also in the digital redesign of continuous-time control systems [6], [7] to convert continuous-time controllers to discrete-time equivalents.

On the one hand, when the reset of the integration takes effect at an integral multiple of the fixed step size, the actual reset event and that obtained with the fixed step simulation of the integrator are occurring at the same time instant. On the other hand, when the reset event of the theoretical model takes place at a nonintegral multiple of the fixed step size, the integrator simulated with any of the aforementioned techniques resets at a step following the occurrence of the event, not necessarily the first iteration step following the time instant at which the reset event took place. Therefore, there are two sources of errors when simulating a reset integrator with fixed steps: the numerical approximation of the integration and the occurrence of reset events at nonintegral multiples of the fixed step size.

Reset Walk Effect

Consider Fig. 1, which presents a model for the simulation of the integration of a continuous signal with reset based on a comparison between the integrator state and an external signal. With a block diagram language such as Simulink, the notion of integrator state is used to avoid algebraic loops [8]. Even for such a simple system, there is an adverse phenomenon that arises in integrator simulations when the reset events occur in between the sampling instants. This phenomenon is called reset walk and is explained as follows. Suppose that several reset events take place over a given time period, assuming that the fixed simulation step size has been selected such that a maximum of one event arises between successive iteration steps. Moreover, assume that the integrator is approximated with any of the aforementioned integrator simulation techniques. Fig. 2 shows the outputs of the actual (or theoretical) and simulated integrators in the case of the integration of a

constant signal with reset occurring every time the state of the integrator reaches a value superior or equal to π. In Fig. 2, the reset events are denoted as $t_{e,i}$, where i represents the ith occurrence of a reset event, and the step size is denoted as T_s in this article. From Fig. 2, it is seen that there is a relative movement (although very subtle) between the time instants at which the resets of the fixed step simulated integrators take place and the instants at which the reset events of the actual integrator occur. To understand the difference in reset event occurrences between the actual and the simulated models and its increase with time, one has to keep in mind that the reset is based on the feedback of the state of the integrator, as shown in Fig. 1. As a consequence of the cumulative effect of integration, reset events of the simulated integrator move (in the time domain) with respect to the events of the actual integrator.

Latch of a Signal

The latch operation can be modeled as a block that samples and holds the value of an input signal at time instants for which a condition is satisfied. In short, the latch acts like a sample-and-hold operator except that, instead of performing the sampling operation periodically at fixed time instants, the sampling operation is carried out at discrete events depending on an algebraic condition. Fig. 3 presents the latch operation, where *HS* samples the input signal at time instants determined by the value of the logic signal and holds it until the next event.

When simulating the latch operation with a fixed step size, it is possible that the simulation omits the time instants at which the actual latch is triggered. Of course, in a fixed step simulation, the input signal is sampled and held at the first step following the triggering of the actual latch. However, the discrepancy between the latch output of the theoretical model and that of the simulated model is always present.

Simulations with RT-Events Blockset

In order to solve the problems inherent to the conventional fixed step simulations of event-based systems, Opal-RT Technologies has created the RT-Events Blockset [9], which is comprised of a set of discrete-time blocks that use a compensated discrete-time simulation algorithm implemented within the Simulink environment and executed with fixed steps. Therefore, the RT-Events blocks are compatible with the Real Time Workshop and RT-LAB [10], [11].

The RT-Events library comprises a comparator block, a discrete-time integrator, a latch, and logical operators (AND, OR, NOR, NAND, XOR, NOT). The comparator block generates a trigger signal whose value depends on the result of a comparison between the two inputs to the block. The output of the

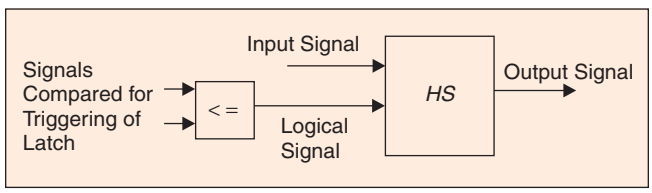

Figure 3. *Model of a latch.*

There are two sources of errors when simulating a reset integrator with fixed steps: the approximation of the integration and the occurrence of reset events at noninteger multiples of the fixed step size.

comparator block carries the information concerning the occurrence of a discrete event. The discrete-time integrator block uses numerical methods to approximate the integration of its input signal and comprises an algorithm that compensates for the occurrence of discrete events in between sampling instants. The key feature of the latch block is that it incorporates a method that compensates for the triggering taking place between sampling instants. Finally, the logical operators perform standard logical operations on signals while compensating for transitions occurring between the sampling times.

The major features of the RT-Events Blockset are summarized as follows:

- It compensates for the errors introduced by events occurring in between the simulation steps. Still, the simulation accuracy depends on the step size selected.
- It allows fast simulations of event-based systems.
- It is suitable for hard real-time applications.
- It is easily adaptable for distributed real-time simulations as obtained within the RT-LAB environment.
- It is compatible with physical counter boards for count reading and writing.

Key Concept

The main idea behind the RT-Events Blockset is to include an algorithm that compensates for events taking place in between simulation steps. Note that in the case of discrete-time

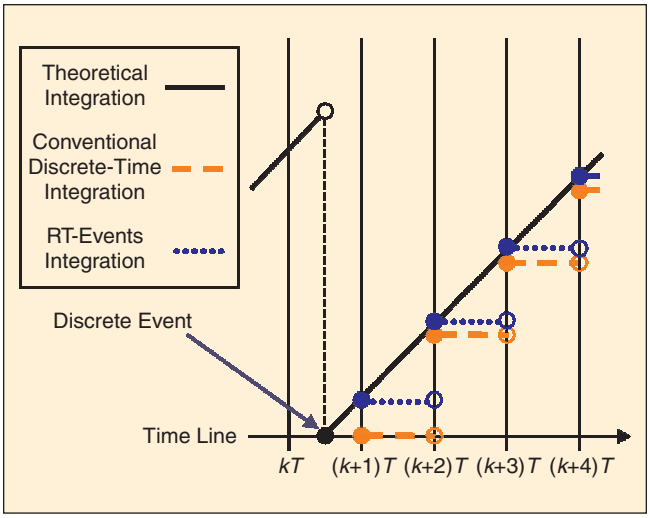

Figure 4. *Output of RT-Events integrator versus others.*

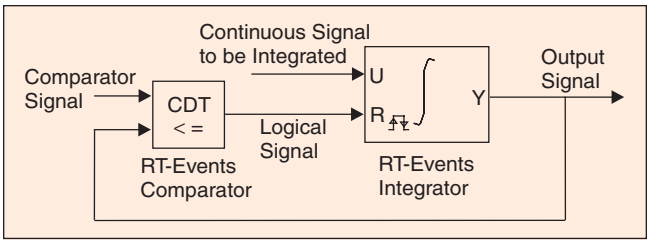

Figure 5. *RT-Events-based model of integration.*

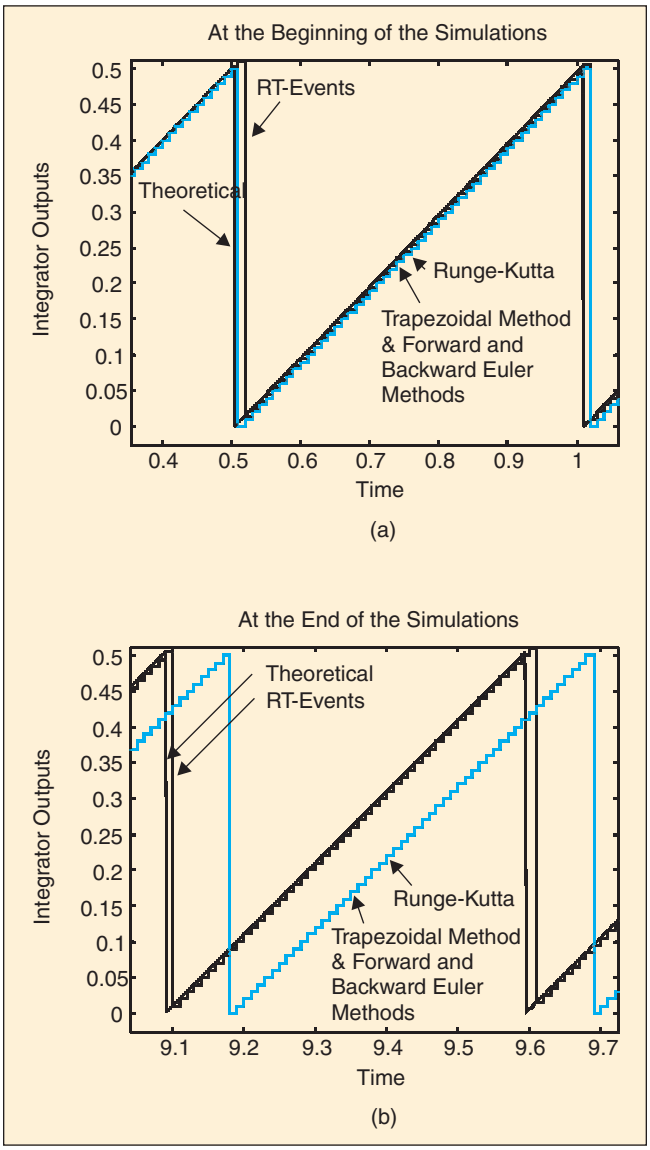

Figure 6. *Integrator outputs (a) at the beginning and (b) at the end of the simulations.*

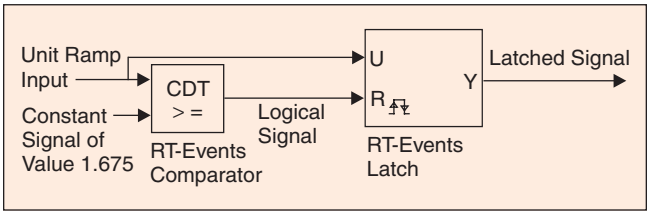

Figure 7. *Schematics of the latch system.*

systems, the simulation step size is assumed to be equal to the sampling period. For instance, in the case of an integrator with reset, at the sampling instants following the occurrence of an event, the discrete-time integrator does not reset its output to zero (as is the case with conventional techniques) but fixes its output to a value very close to that of the actual integrator—a value that is obtained with the compensatory algorithm. This provides a more accurate simulation with respect to the theoretical system. The principle is illustrated in Fig. 4 for the numerical integration of a constant signal with a reset event taking place in between steps kT and $(k+1)T$.

As another example, the output of the latch is not taken to be the value of the input signal at the simulation step following the triggering of the latch, as is done with the current simulation tools. Instead, this value is compensated to account for the event not taking place at the simulation step. Thus, this technique results in more accurate fixed step simulations of the latch operation.

Solution to Reset Walk Effect

Using the integrator and comparator blocks of the RT-Events library solves the reset walk problem mentioned earlier. As opposed to the conventional integrator simulation techniques, the RT-Events integrator guarantees a compensation for the reset event at the sampling instants following the occurrence of an event. Thus, the error between outputs of the RT-Events simulated and the actual reset integrators cannot increase with time as it does with the classical simulation techniques.

Numerical Examples

This section provides three examples illustrating the effectiveness of the new simulation tools. The first two examples are simple systems showing the compensated integrator and latch blocks, respectively, whereas the third example is that of a V-6 engine control system. Preliminary results for the engine simulation were presented in [12]. The reader is reminded that models executing at variable steps are referred to as theoretical models. For such variable-step simulation execution, relatively small tolerances are set.

Integrator with Reset

This example is modeled, as shown in Fig. 1, for the variable-step and conventional fixed step simulations. The continuous signal to be integrated is a unity constant and the comparator signal is chosen to be a constant value of 0.505. The fixed step size is selected as $T_s = 0.01$. With the fixed step size and the comparator signal values selected as such, the reset events take place in between integral multiples of T_s. The setup for the RT-Events-based simulation is shown in Fig. 5. The integrator outputs are shown in Fig. 6 for the beginning and the end of the simulations so that it is easier to visualize the various signals. The reset walk is apparent for the conventional fixed step simulation techniques (five-point Runge-Kutta, Forward and Backward Euler, and Trapezoidal methods), whereas no such effect can be seen for the simulation per-

formed with the RT-Events blocks. In fact, the increase in integration error with time, as obtained with the conventional fixed step simulation techniques, is easily seen by comparing Fig. 6(a) and (b). This example clearly favors the RT-Events toolbox over the so-called native Simulink integrator and comparator blocks since no accumulation of error is present with the RT-Events-based simulations.

Latch of a Continuous Signal

The setup of the simulation for the RT-Events-based model is shown in Fig. 7. The latch obtained with the Simulink library is the triggered feedthrough block and is not shown for brevity. The input to the latch is a unit ramp signal. The latch takes effect at the time the input reaches a value of 1.675. Thus, the theoretical output of the latch is a constant signal of value 1.675 starting at time $t = 1.675$. The fixed step simulations are carried out with $T_s = 0.01$. The outputs of the simulated and theoretical latches are shown in Fig. 8. The compensated latch (RT-Events-based model output) shows a behavior superior to that achieved with the conventional fixed step simulations. The time delay between the actual latch event occurrence and the latch of the fixed step simulations is expected since the actual latch event occurs at 1.675, which is between the fixed simulation steps 1.67 ($167 \cdot T_s$) and 1.68 ($168 \cdot T_s$).

Distributed Simulations of an IC Engine Control System

An engine control system is modeled with the Simulink block diagram language [13]. The block diagram of the closed-loop system is shown in Fig. 9. The engine dynamics (plant), which include the four stroke modes of the cylinders, the electronic control unit (controller), the sensors, and the actuators are modeled.

ENGINE MODELING

As shown in Fig. 10, the V-6 engine model comprises torque generation and intake and engine dynamics. The key modeling issue for simulations executing at fixed time steps occurs within the torque generation subsystem. The torque generation subsystem of the model requires determination of the stroke mode. At each simulation step, the stroke mode of a given cylinder is determined by the crankshaft angle. The algebraic condition that indicates the stroke mode is based on the crossing of each cylinder's shaft position through 0, π, 2π, and 3π.

The comparison between the reference positions (0, π, 2π, and 3π) and the crankshaft positions, which allows detection of the event, is performed with blocks coming from the RT-Events library at the fixed simulation steps. This avoids propagation of the numerical simulation error over time (reset walk) when the simulated system is compared with the actual system.

ELECTRONIC CONTROL UNIT MODELING

The ECU, which is modeled as a discrete-time control system, is a computer that uses input data from the sensors to calculate an output control signal that moves the actuators in such a way as to coerce the plant to follow the desired reference trajectory. In order to do so, there are signal processing (filtering, rate limiting, etc.) controllers, as well as spark, fuel, and exhaust gas recirculation controllers.

One important application of the RT-Events/RT-LAB combination is its capability to simulate automotive systems as real-time, hardware-in-the-loop simulations, or as faster-than-real-time simulations.

ACTUATORS AND SENSORS MODELING

The actuators are modeled as a simple throttle valve (pressure ratio, first-order filtering effect), an injector, exhaust gas recirculation, and a spark plug. The model of the sensors comprises a heated exhaust gas oxygen sensor, noise effects, air-fuel ratio flow measurement, and fault monitoring.

SOFTWARE/HARDWARE INTEGRATION

The RT-LAB environment enables distributed simulation and parallel processing over a PC cluster, as shown in Fig. 11.

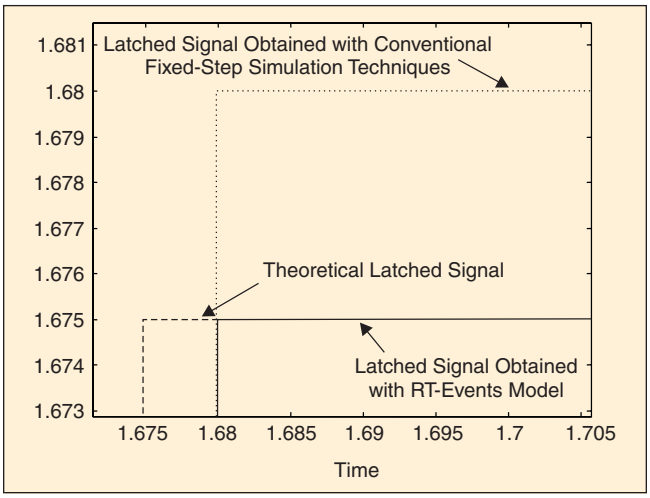

Figure 8. *The various latched signals.*

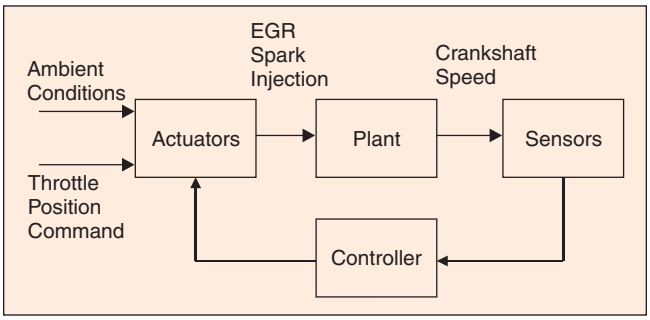

Figure 9. *Feedback control system.*

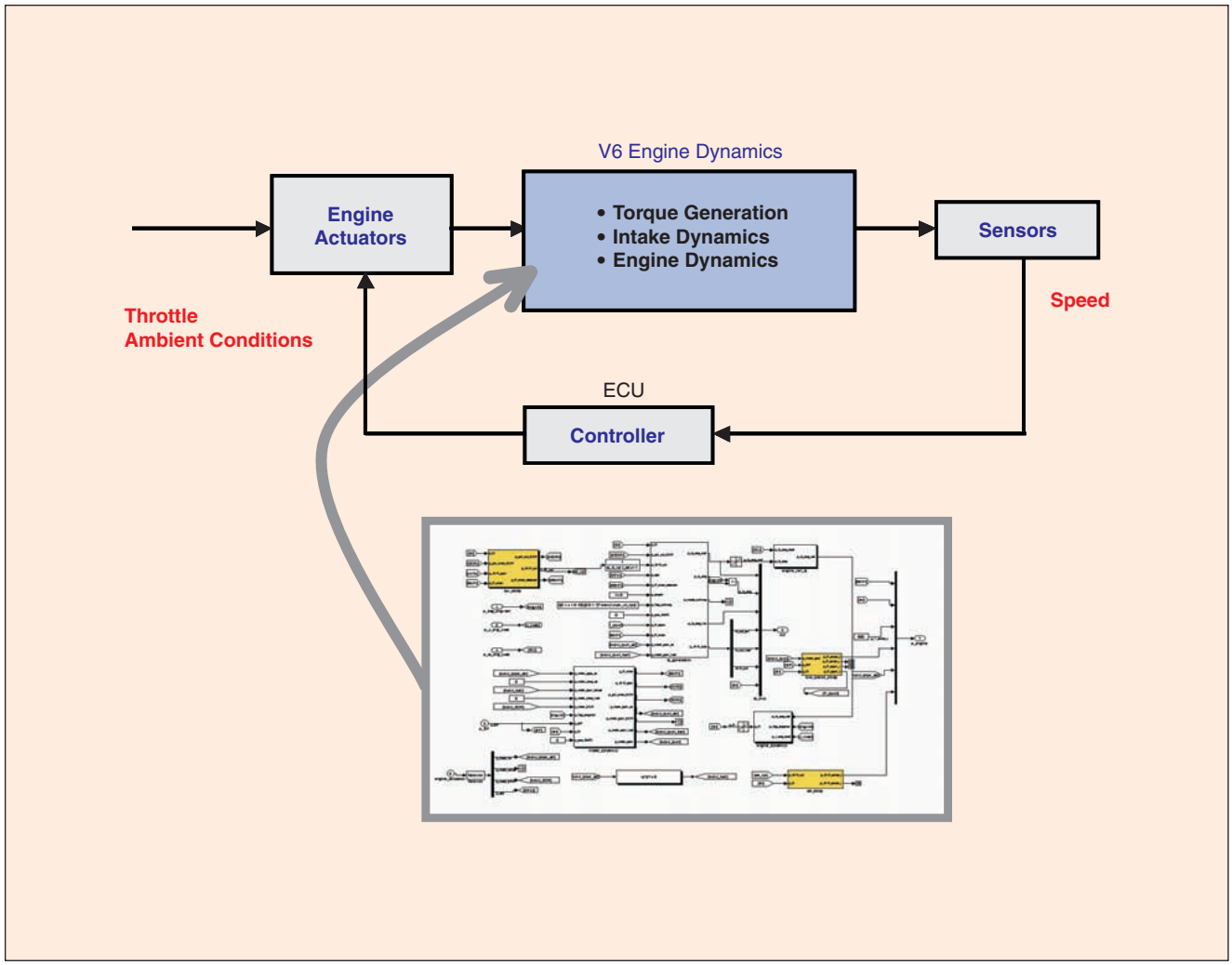

Figure 10. *Block diagram schematics.*

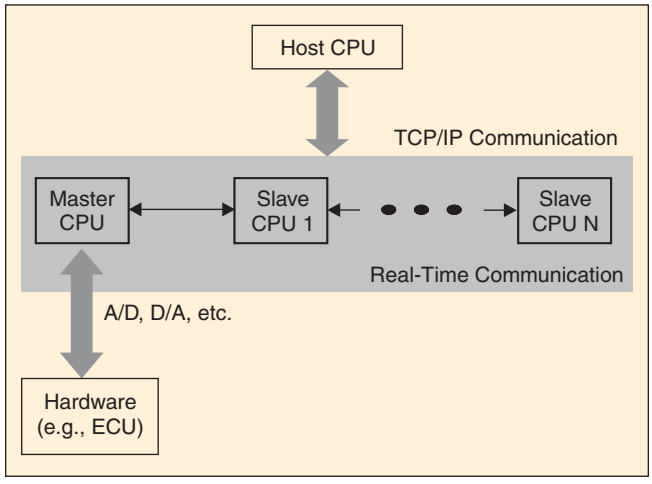

Figure 11. *Distributed simulation environment.*

An application example of distributed simulations can be found in [14]. Furthermore, an interesting discussion on a distributed simulation environment is presented in [15]. Overall, RT-LAB constitutes an environment integrating block diagram languages (for modeling) and multiple processors (for distributed simulation). The physical configuration of the simulation system, as shown in Fig. 11, is divided into the host and the target sides. On the host side, the engineer designs the model and runs offline simulations. On the target side, several microprocessors are connected via a high-speed communication network, such as cLAN, FireWire, or shared memory, and serve in the execution of the distributed simulations. The simulations can either be executed in the fast mode (with a Windows operating system) or in the real-time mode (with a QNX or Neutrino operating system). Control of the simulation is provided by RT-LAB, which sits on the host computer and communicates with the target nodes via TCP/IP. As noticed from Fig. 11, the master computer can be connected to actual hardware for HIL tests and to a set of slave nodes for distributed execution. Furthermore, an embedded target, such as a PC/104, can be connected to an actual engine system, as shown in Fig. 12.

RT-LAB automates the following sequence of tasks

- separation of a model into sub-models
- code generation for sub-models

- compilation of sub-model code
- distribution of executables on desired target nodes.

RT-LAB works with the standard Simulink and SystemBuild block diagram languages and their code generators to enable parallel, real-time execution of simulations and I/O interfacing. RT-LAB handles the setup of all the communications, including real-time data exchanges between target processors. Furthermore, RT-LAB provides the interface to control simulation execution via the main control panel located on the host computer.

The following features are supported by the Application Program Interface (API) of RT-LAB: model editing; model separation, code generation, and compilation; simulation control (run, step-by-step, pause, reset); target operating system and node selection; on-the-fly snapshot of simulation state; on-line parameter tuning; and data probing.

The open architecture of RT-LAB allows third-party products to be connected for visuals; for instance, LABVIEW and ALTIA. Furthermore, complex configurations can be implemented; for instance, drive-by-wire control systems can first be modeled on a simulation cluster and then, as components arrive, the simulated components are replaced on a block-by-block basis, thus reducing time to market.

Table 1. Simulation execution speeds.

Model and Configuration	Average Effective Step Size (μs)	Total Execution Time (s)	Speed Gain
Reference Offline: Simulink	N.A.	2000	N.A.
Fixed-Step Offline: Simulink	3500	700	2.86
Fixed-Step RT-Events: RT-LAB 1 CPU (Pentium III)	445	89	22.47
Fixed-Step RT-Events: RT-LAB 2 CPUs (Pentium III) Shared Memory	255	51	39.22
Fixed-Step RT-Events: RT-LAB 3 CPUs (Xeon Server) Shared Memory	140	28	71.43

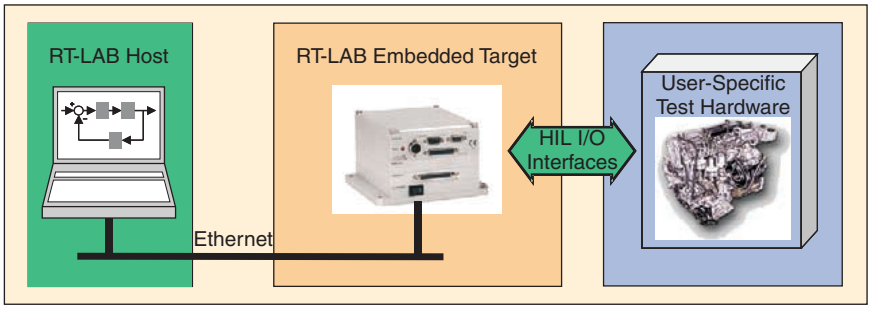

Figure 12. *Hardware-in-the-loop configuration.*

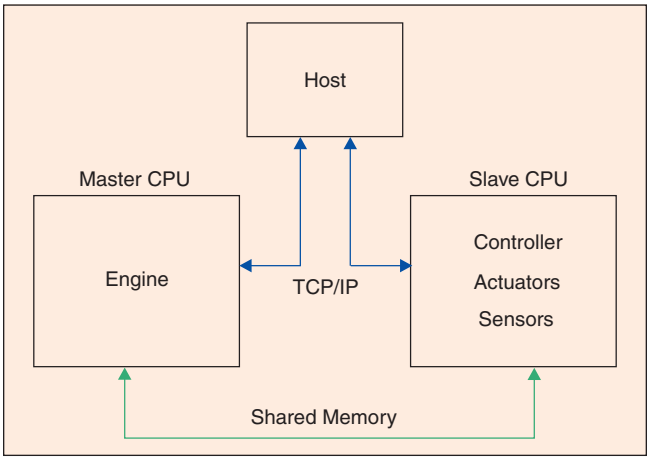

Figure 13. *2-CPU distributed simulation schematics.*

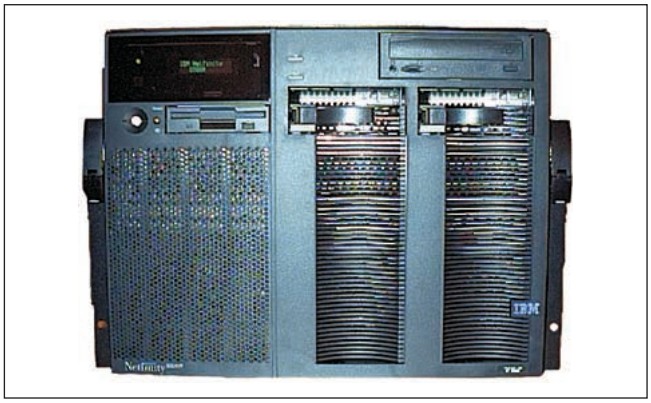

Figure 14. *8-way Xeon server.*

NUMERICAL SIMULATIONS

The Simulink model of the V-6 engine system comprises integrator and latch blocks of the RT-Events library, where appropriate, to model the occurrence of the discrete crankshaft events. The engine model is simulated at fixed time steps on a cluster of microprocessors with the RT-LAB environment. Two aspects are emphasized in the numerical simulations of the engine system: 1) accuracy and 2) speed improvement obtained with the novel modeling and simulation method and the integrated software-hardware solution. It should be emphasized that, knowing the computational and communication loads associated with the various processes involved in a distributed simulation, one can determine the smallest achievable simulation step size for real-time simulations and HIL.

In this experiment, the closed-loop engine model is simulated on: 1) a single Windows-NT-based, Pentium III 550 MHz processor with 128 MB RAM; 2) two Windows-NT-based, Pentium III 550 MHz, 128 MB RAM processors connected via shared memory, where the ECU model and actuator and sensor models are simulated on one machine whereas the plant model is dedicated to the other microprocessor, as shown schematically in Fig. 13; and 3) a 667 MHz, 512 MB RAM 8-way Xeon server comprising Pentium III microprocessors

connected via shared memory, in which case the portioning of the model is done in such a way as to dedicate one target microprocessor to the ECU model, another to the torque generation subsystem, and the other CPU to actuators, sensors, and intake and engine dynamics. Finally, 100-s scenarios are simulated at a step size of 500 μs. Fig. 14 shows the IBM Xeon server on which the simulations are executed in parallel. Both the host (for simulation control) and the targets (for simulation execution) lie within the Xeon machine.

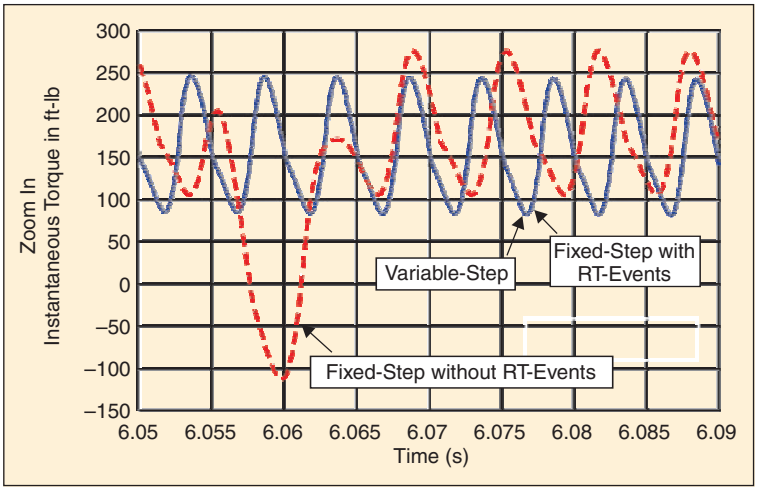

Figure 15. *Instantaneous torque.*

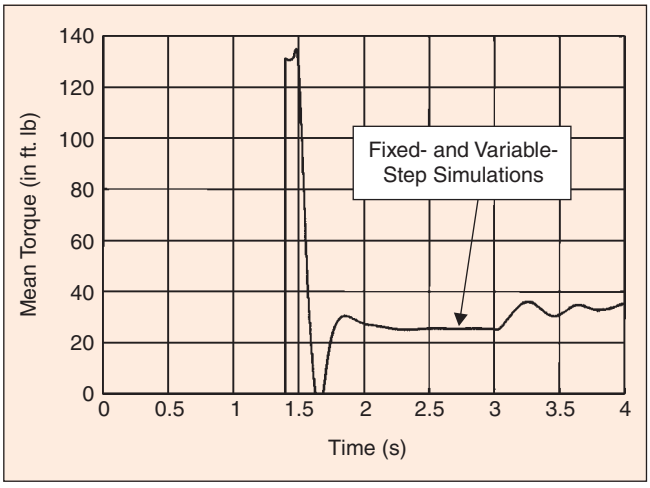

Figure 16. *Mean torque.*

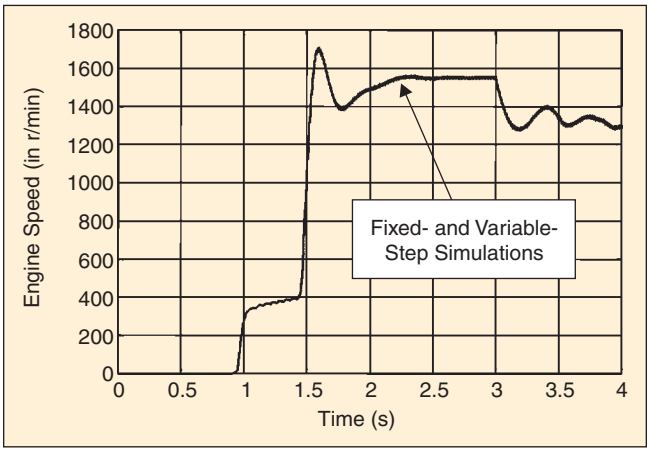

Figure 17. *Engine speed.*

To demonstrate simulation accuracy, engine speed (in rpm), mean torque (in ft lb) and instantaneous torque (in ft lb) are measured. The time trajectories of the signals are obtained via binary files generated during the simulation run and are compared against the signal trajectories of a variable-step, Simulink simulation of the engine model, which is considered the yardstick, or reference, against which all fixed step simulations are measured. Figs. 15-17 present the instantaneous torque, the mean torque, and the engine speed, respectively. It can be seen that models comprising the RT-Events blocks present output signals that are almost indistinguishable from the signals obtained with the reference model.

Intuitively, the engineer expects speed improvement in the execution of simulations by distributing the engine system simulations and by executing the various processes in parallel. In fact, the time devoted for the computations and the communications, which constitutes the so-called effective step size associated with the execution of a distributed simulation for each time step, should be reduced. This is indeed the case, as shown in Table 1. The following can be noticed from the table:

- The slowest simulation execution takes place within the Simulink environment in the offline mode and at variable step sizes.
- The fastest simulation execution occurs for the three-CPU configuration, as obtained with RT-LAB, and is almost 72 times faster than that achieved within Simulink.

Summary

The problems inherent to the techniques currently available to simulate event-based systems, namely the time-consuming variable-step algorithms and the inaccurate conventional fixed step algorithms, can be solved by using the novel RT-Events Blockset for system modeling and the RT-LAB environment for distributed simulation. The key concept behind the RT-Events library of blocks is one of compensation for the events taking place in between the sampling instants. The potential applications of this new toolbox are clear, including the use of the RT-Events Blockset to effectively model and simulate IC engines.

Acknowledgments

The authors would like to acknowledge the work and support of Jon Friedman, Dejan Boskovic, Shiva Sivashankar, Simon Abourida, and Vincent Lapointe for their investigations into event-based engine simulation in commercial modeling environments.

Keywords

Discrete events, dynamic systems, modeling, simulation, block diagram language, internal combustion engine.

References

[1] C. Cassandras, *Discrete Event Systems: Modeling and Performance Analysis*. Burr Ridge, IL: Richard D. Irwin, Inc. and Aksen Assoc., Inc., 1993.

[2] C.A. Rabbath, M. Abdoune, and J. Belanger, "Effective real-time simulations of event-based systems," in *Proc.2000 Winter Simulation Conf.*, Orlando, FL, 1999, pp. 232-238.

[3] J. Pachner, *Handbook of Numerical Analysis Applications*. New York: McGraw-Hill, 1984.

[4] K.J. Astrom and B. Wittenmark, *Computer-Controlled Systems: Theory and Design*. Englewood Cliffs, NJ: Prentice Hall, 1990.

[5] T.T. Hartley, G.O. Beale, and S.P. Chicatelli, *Digital Simulation of Dynamic Systems—A Control Theory Approach*. Englewood Cliffs, NJ: Prentice Hall, 1994.

[6] G.F. Franklin, J.D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*. Reading, MA: Addison-Wesley, 1994.

[7] C.A. Rabbath, N. Hori, P.N. Nikiforuk, and K. Kanai, "Order reduction of PIM-based digital flight control systems," in *Proc. IFAC World Congr.*, 1999, vol. Q, pp. 127-132.

[8] *Simulink User's Guide*. Natick, MA: The Mathworks, Inc., 2000.

[9] *RT-Events User's Guide*. Montreal: Opal-RT Technologies, Inc., 2000.

[10] M. Papini and P. Baracos, "Real-time simulation, control, and HIL with COTS computing clusters," presented at the AIAA Guidance, Navigation, and Control Conf., Denver, CO, 2000.

[11] *RT-LAB User's Guide. Montreal: Opal-RT Technologies, Inc., 2001.*

[12] C.A. Rabbath, H. Desira, and K. Butts, "Effective modeling and simulation of internal combustion engine control systems," in *Proc. American Control Conf. 2001*, Arlington, VA, 2001, pp. 1321-1326.

[13] N. Sivashankar, D. Boskovic, and J. Friedman, "A control-oriented engine benchmark model," presented at the Symp. Advanced Automotive Technologies, Orlando, FL, 2000.

[14] J. Ozard and H. Desira, "Simulink model implementation on multi-processors under Windows-NT," *Military, Government, and Aerospace Simulation*, vol. 32, no. 3, 2000.

[15] L. Pollini and M. Innocenti, "A synthetic environment for dynamic systems control and distributed simulation," *IEEE Control Syst. Mag.*, vol 21, pp. 49-61, Apr. 2000.

C.A. Rabbath received the Ph.D. degree in 1999 from McGill University, Montreal, Canada. He has worked as a control systems specialist and consultant for Opal-RT Technologies from 1999 to 2002, and is now a Defence Scientist for Defence Research and Development Canada and an adjunct professor in the Department of Mechanical Engineering at McGill University. His research interests include sampled-data control of dynamic systems, missile autopilot, control of gas-turbine and internal-combustion engines, and discrete-time modeling of continuous-time systems.

M. Abdoune received the M.Eng. degree from Université de Setif, Algeria, in 1986; the Diploma of Specialization in nuclear engineering from Chinese Atomic Energy Institute, Beijing, China, in 1990; and the M. Eng. Degree from Ecole de Technologie Superieure, Montreal, Canada, in the field of control systems in 1999. Currently, he is a control systems specialist and consultant for Opal-RT Technologies. His research interests include optimal and flight control, and real-time modeling, control, and simulation.

J. Belanger received the B.Eng. degree in electrical engineering from Laval University, Quebec City, Canada, in 1971 and the MScA degree from Montreal University, Montreal, Canada, in 1975. He worked for Hydro-Quebec or one of its subsidiaries for almost 20 years until 1997. In 1997, he cofounded Opal-RT Technologies, Inc. and has served as president since. His research interests include real-time simulations, power systems simulations, and distributed and parallel simulations of complex dynamic systems.

K. R. Butts received the B.E. degree from General Motors Institute in 1980, the M.S. from the University of Illinois at Urbana-Champaign in 1982, and the Ph.D. degree from the University of Michigan, Ann Arbor, in 1993, all in electrical engineering. He worked on various aspects of automotive powertrain control for General Motors Corporation from 1980 to 1994. He joined the Powertrain Control Systems Department of the Ford Research Laboratory, Ford Motor Company, in 1994, where he is currently a staff technical specialist. His research interests continue to be focused on automotive powertrain and vehicle system control with particular emphasis on the application of computer-aided control system design methods and tools to controller development.

Address for Correspondence: C.A. Rabbath, Defence Research & Development Canada—Valcartier, 2459 Pie-XI Blvd. North, Val-Belair, Quebec, G3J 1X5 Canada. E-mail: Camille-Alain.Rabbath@drdc-rddc.gc.ca.