Proceedings of the 2016 Winter Simulation Conference
T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, eds.

ADAPTIVE RESOLUTION CONTROL IN DISTRIBUTED CYBER-PHYSICAL SYSTEM SIMULATION

Dylan Pfeifer

Power Standards Lab
980 Atlantic Blvd
Alameda, CA 94501, U.S.A.

Andreas Gerstlauer
Jonathan Valvano

Electrical & Computer Engineering
1616 Guadalupe Stop C0803
Austin, TX 78701, U.S.A.

ABSTRACT

Cyber-physical systems challenge the field of parallel and distributed simulation due to the plurality of simulators required to model and simulate diverse system components. Additionally, simulation may require fine resolution to simulate events that occur at simulated system software-level frequencies or simulated system electrical circuit-level frequencies. The frequency of events at may prohibit long simulation runs due to high event communication messaging among distributed simulators. Adaptively varying the simulation resolution during the simulation can provide a speedup by reducing messaging traffic. This work offers a general means for adaptive resolution control using the Kahn Process Network and Interpolated Event method of parallel and distributed simulation. The method is applied through the SimConnect and SimTalk simulation tools to the parallel and distributed simulation of a closed-loop motor control system. Results demonstrate reduction in messaging traffic for adaptively controlled resolution cases versus fixed resolution cases with controllable tradeoffs in speedup versus accuracy.

1 INTRODUCTION

Cyber-physical systems (CPS), defined as engineered systems that integrate computation and physical processes (Lee 2008), are an emerging challenge to the disciplines of simulation. Hybrid by construction, CPSs inherit the field of real-time embedded systems, but challenge engineering simulation and electronic design automation (Sangiovanni-Vincentelli 2007) because CPS components continue to proliferate in quantity, increase in computational capability, decrease in area and power, and increase in system-on-chip (SOC) complexity (NSF 2006, Klesh et al. 2012, Rajkumar et al. 2010). Additionally, CPS systems may include hardware, software, mechanical, or even biological components (Lee 2008). While individual simulators may specialize in modeling some of these components, no single simulator yet superlatively models all of them. When modeling and simulation demands eclipse the capacity of any single simulator or simulation solution, it is possible to endeavor the simulation requirements by coordinating multiple different simulators, each specializing in a simulation capability required of the simulated system. For this work, this is termed the *simulator interfacing* approach.

Interfacing and coordinating multiple heterogeneous simulators, which may execute independently over distributed computation resources, introduces challenges that locate the simulator interfacing approach in the field of parallel and distributed simulation (PADS, Fujimoto 2000). Principally, simulators may execute with independent tracking of time and time advancement, introducing a challenge of simulator time synchronization. Next, simulators running with local time and independent state must deliver the same global simulation event causality that they would if executing sequentially with combined state. Solutions to these principal challenges have been offered in the field of PADS over the past four decades, with seminal contributions by Chandy and Misra (1979), Jefferson (1985), and Fujimoto (1989).

Once the simulator coordination challenge is overcome, a remaining challenge is to reduce the simulation time required. In CPS simulation, some simulators may increase in simulation time

exponentially as model complexity increases (Pfeifer and Gerstlauer 2011). Additionally, if CPS simulation observes events at a circuit-level frequency, or digital gate-level frequency, simulators can potentially post events at a minimum simulation time step resolution. This can be on the order of simulation nanoseconds and even picoseconds for modern VLSI clock frequencies (Lee 2008). Where simulation times on the order of seconds are required to observe system software OS-level events (which may occur many orders of magnitude less frequently than gate and circuit-level events), and on the order of milliseconds to observe many simulated CPU instructions, the simulation runtime may be prohibitively long if simulated conservatively (Fujimoto 1995).

Optionally, the system may be simulated optimistically to endeavor simulator speed up, but this may impose interfacing requirements for the optimistic simulation scheme that may not be supported by the simulators desired to be interfaced. While an optimistic communication interface may be constructed for the simulator (particularly since most engineering simulators offer user-level software interfaces), the time to construct and verify the interface may eclipse resources sets for the CPS simulation.

Another means is to dynamically vary the simulation lookahead (Fujimoto 1995) for conservatively coordinated simulations, since simulation runtime performance and lookahead values for conservative models have been shown to be correlated (Fujimoto 1989). That is, distributed models with large lookahead may simulate faster than distributed models with small lookahead due to higher messaging traffic required for small lookahead models. Lookahead variation can be effectively equivalent to dynamic external event resolution for distributed models.

However, the implementation of dynamic lookahead variation mid-simulation can implementation specific, and may not be supported by all simulators desired in a CPS simulation. A general means of adaptive event resolution control, independent of simulator design and internal code visibility, is desired to offer the capability of dynamic simulation resolution without requiring the coordinated simulators to have interfaces for lookahead or lookahead management. This is particularly important for proprietary engineering simulators which may be designed with closed internal architecture.

To offer a simplified simulator interfacing scheme that endeavors to impose as few possible functional burdens as possible on coordinated simulators, the Kahn Process Network (KPN) and Interpolated Event (IE) method of simulator coordination was developed in (Pfeifer 2013). The method attempts to approach the broad range of required CPS simulation by offering simplified means to coordinate as many diverse simulators as possible, with reduced functional requirements for interfacing diverse simulators and reduced functional burdens for the coordinating simulation backplane compared to existing techniques.

This method utilizes the dataflow properties of the Kahn Process Network model of concurrent computing in order to provide simulator synchronization. The tokens of the KPN with the method are restricted to be of a data type called the Interpolated Event (Pfeifer and Valvano 2011). This method enables both conservative and optimistic coordination of distributed simulators that are able to interface to an instantiated Kahn Process Network through model-level Interpolated Event ports. The full properties of the method, its comparative advantages and drawbacks for CPS simulation, and its application through SimConnect and SimTalk software tools are given in (Pfeifer, Valvano, and Gerstlauer 2013). SimConnect implements the KPN dynamics in a simulation backplane, and SimTalk enables the transport of Interpolated Events among simulators according to the rules of a KPN. The tools and method have been shown to be effective for interfacing engineering design simulators that have user-level code-modeling interfaces with OS-level system call support, but that otherwise have closed, proprietary internal architecture.

Adaptive control is achieved by dynamically varying the resolution of Interpolated Events as they are routed through the SimConnect backplane. The KPN-IE method, compared to alternative interfacing solutions, is designed to be a dataflow-based approach to simulator coordination. The benefits of a dataflow approach are covered in (Pfeifer 2013).

2 RELATED WORK

Development of the KPN-IE method for PADS is given in (Pfeifer, Valvano, and Gerstlauer 2013). In (Pfeifer and Valvano 2013), it is applied to CPS simulation without dynamic or adaptive resolution control, and in (Pfeifer et al. 2013) it is applied to CPS simulation with pre-programmed dynamic control.

Another important simulator interfacing solution is the IEEE 1516 standard (IEEE 2010), formerly the U.S. Department of Defense High Level Architecture (HLA) for distributed simulation (Kuhl et al. 1999). HLA has proven successful for many PADS applications, but as argued in (Pfeifer 2013), its preeminence in the work of coordinating commercial engineering design simulators has yet to be established with the strength for which it has shown in other applications. HLA can offer dynamic lookahead variation during the simulation when a federated simulator changes is registered lookahead value with the Run Time Infrastructure (RTI) backplane through an "RTI ambassador" software interface.

The potential speedup benefit of dynamic time management in distributed simulation is discussed in (Zhiwu and Yanfeng 2009), with application specific solutions offered in (Lungeanu and Shi 1999, Lee et al. 2002, and Chung and Kyung 2006). However, for cyber-physical systems, the HLA-based techniques discussed in (Zhiwu 2009) require HLA support among cyber-physical system simulators, which may require internal simulator kernel code modification (Pfeifer et al. 2013), particularly since time regulated federates in the HLA taxonomy must surrender all time advancement control to the RTI backplane. The SimConnect/SimTalk based solution to dynamic time management, by contrast, endeavors to be code interfacing-friendly through the Interpolated Event data type and SimTalk connector. Implementation of the dynamic resolution compliance, for example, added less than 500 lines of C code to the tools from their code size given in (Pfeifer 2013).

3 ADAPTIVE RESOLUTION CONTROL WITH THE KAHN PROCESS NETWORK AND INTERPOLATED EVENT METHOD

The Kahn Process Network and Interpolated Event (KPN-IE) method of simulator coordination for parallel and distributed simulation (PADS) is thoroughly covered in works (Pfeifer, Valvano and Gerstlauer 2013). Properties of the method are re-summarized for the discussion of adaptive control.

3.1 Kahn Process Networks

The Kahn Process Network (KPN) is a dataflow formalism for concurrent computing introduced by Gilles Kahn in (Kahn 1974). The KPN formalism consists of independent compute nodes connected by simplex FIFOs that carry tokens from compute node producer to compute node consumer. The KPN is completely characterized by the dataflow graph consisting of the set of nodes (the independent computation processes), and the set of arcs connected them (the simplex FIFO channels from producer to consumer). A KPN has the following rules:

- The KPN is a directed graph with arcs, representing point-to-point simplex FIFOs, and nodes, representing concurrent compute elements without interdependent side-effects.
- Nodes may read from input FIFOs and write to output FIFOs, but reads are blocking (the node stalls) if the FIFO is empty, while writes always succeed (the node does not stall).
- Nodes may not conditionally execute by FIFO sniffing, and FIFOs are unbounded (infinite depth).

Kahn showed that if these dataflow rules are followed, the execution of the KPN and token processing is deterministic and independent of node execution scheduling. The node execution rather is completely determined by the FIFO contents in time and does not require a centralized controller to coordinate when nodes are to execute. This is a benefit to the problem of simulator coordination, where we do not want to

program or control when simulators are to execute, particularly if many hundreds or more simulators comprise the system simulation (Fujimoto 2000).

3.2 Interpolated Events

The KPN-IE method requires tokens of the KPN to be a data type called the Interpolated Event (IE). Interpolated Events are thoroughly described in (Pfeifer 2013).

Definition 1 *Interpolated Event. An Interpolated Event (IE) is a 3-tuple set element (v, t_m, t_n) from the product set $V \times T \times T$, where V is a set of values, and T is a set of tags. For a given Interpolated Event (v, t_m, t_n), the value v is defined to be constant on the interval [t_m, t_n) specified in the IE, such that the tag set T is ordered.*

An Interpolated Event (v, t_m, t_n) assigns a "stable" time to the signal value v for producers and consumers over the tag range [t_m, t_n). If a simulator consumes an interpolated event (v, t_m, t_n), it may assume the value v is constant on the tag range [t_m, t_n), and not need to sample the value again until expiration time tn. Therefore, an Interpolated Event encapsulates both communication (the signal value) and synchronization (the start and end time). Interpolated Events collapse to conventional time stamped events (Lee and Sangiovanni-Vincentelli 1996), or events of the form (v, t), if $t_m = t_n$. IEs collapse to untimed FIFO events if $t_m = t_n = NULL$, where *NULL* is defined to be the absence of specification of t_m and t_n. The value ($t_n - t_m$) is called the IE duration, or ΔIE. It may be fixed or varied over the course of a simulation.

Interpolated Events enable simulator synchronization if simulators are restricted to communicate signals through streams of IEs distributed through a KPN. In terms of conservative PADS synchronization, the IE expiration t_n value can be seen as a look ahead value (Fujimoto 1989) for the signal and lower bound on the t_n value on all future IEs posted on a FIFO. It can also be considered the conservative lower-bound time stamp (LBTS) value (Fujimoto 2000) for all input signal IEs if the minimum last received IE t_n value is evaluated across all simulator IE input ports at any moment in simulation time (Pfeifer 2013). An IE (v, t_m, t_n) can also be considered as the union of event (v, t_m) and the set of all (Chandy, Misra and Bryant 1979) styled NULL messages { (t_i , *NULL*) } such that $t_m < t_i < t_n$.

When a simulator local clock reaches an IE expiration time t_m on one of its input KPN FIFO ports, the simulator must query the KPN FIFO for a new IE. If the KPN FIFO is empty, the simulator must block according to the rules of the KPN. Because the simulator cannot advance local time beyond the last received t_n value on an input port, and because the simulator must block if no new FIFO IEs are available, the simulator is unable to advance beyond the t_m values of any not yet received IEs. In this way a simulator cannot advance beyond the start time of future IEs, and equivalently cannot receive an IE in its local time's past. These two properties enable the simulator to observe the local causality constraint (Fujimoto 2000), a restriction in conservatively coordinated simulation that simulators process all events (internally produced or externally received) in time stamp order. Obeying the local causality constraint enables simulators to guarantee that the parallel and distributed simulation has the same event causality as a sequential one for the same set of simulation events (Fujimoto 2000).

4 ADAPTIVE SIGNAL RESOLUTION CONTROL WITH INTERPOLATED EVENTS AND KPN-IE

Because all IE traffic in a KPN-IE method simulation must go through the KPN-implementing simulation backplane, the KPN backplane has the ability to globally observe IE traffic or IE properties in signal FIFOs. The backplane may issue resolution change request messages to signal producing simulators based on properties observed in an IE stream that it monitors.

For example, the backplane may choose to increase resolution for IE streams showing frequently changing IE v values. Alternatively, it may decrease resolution for IE streams showing unchanging or slowly changing IE v values. Measurement of IE value change rates can be accomplished through finite differences-based estimators of discretized signals (Golub and Ortega 1993, Turner 2001). For a continuous signal discretized to a fixed time step h, finite difference estimators can offer flexibly accurate and flexibly

stabile measurements of the signal n^{th} order derivatives. For fixed duration, sampling based IEs, h is simply replaced by ΔIE for the IE stream representing the signal. Values of the signal for finite difference approximations may be taken on the IE t_m or t_n values, but for this study it is taken on the t_m value.

Equation 1 provides the definition of a 2^{nd} order central-difference estimator of a sampled signal second derivative with step size h (Turner 2001).

$$f''(x) \approx \frac{f(x+h) - 2 \times f(x) + f(x-h)}{h^2} \quad (1)$$

Equation 2 provides an equivalent expression for an IE stream with three IEs (IE_2, IE_1, and IE_0) with fixed ΔIE duration, and IE_0 (t_m) > IE_1 (t_m) > IE_2 (t_m). The IE stream 2^{nd} order central difference can be measured by:

$$f''(x) \approx \frac{IE_2(v) - 2 \times IE_1(v) + IE_0(v)}{(\Delta IE)^2} \quad (2)$$

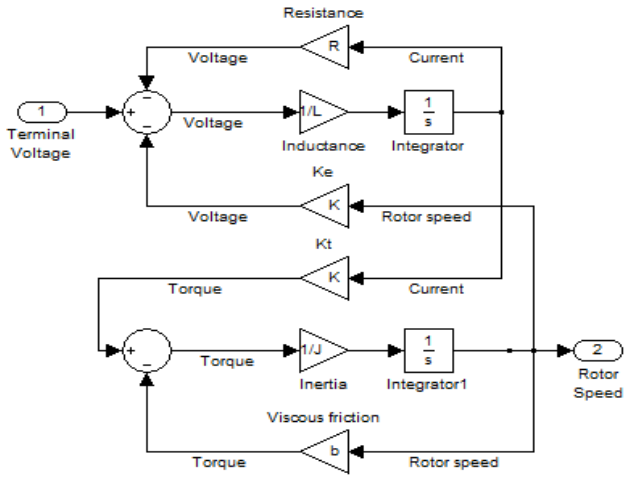
The advantage of using backplane-based IE resolution trackers and adaptive resolution adjustors is that the simulation can proceed without operator pre-assignment of resolution change points (which are few in number and pre-programmed in the experiments in Pfeifer et al. 2013). With adaptive resolution control, the resolution change points are generated by the backplane and can be numerous (thousands, for example, in experiments results in Table 1).

4.1 Experiment

The CPS under simulation is a software-based, Proportional-Integral-Derivative (PID) based control of a pulse-width modulated (PWM) DC motor. The PID and PWM control is implemented by a simulated microcontroller, and the DC motor is represented by a 6-parameter, 2nd order ordinary differential equation (ODE) electro-mechanical model taken from (Franklin et al. 2002).

4.1.1 Experiment Setup: 1-Simulator Configuration

The experiment setup is adapted from (Pfeifer, Valvano and Gerstlauer 2013). To establish a truth condition, a 1-simulator, Matlab/Simulink-only simulation of the PID/PWM control system serves as a single-simulator, sequential simulation to evaluate the accuracy of the more refined and distributed 2-simulator based simulation of Section 4.1.2. The DC motor is modeled in Simulink block-diagram form with summing, integrating, and gain blocks in Figure 1.



R	motor terminal electrical resistance	1.0 Ohm
L	motor terminal inductance	0.001 H
Kt	torque constant	0.1 Nm/A
Ke	electrical constant	0.1 Nm/A
b	rotor viscous friction coefficient	0.001 Nm · s
J	rotor moment of inertia	0.01 kg · (m/s)^2

Figure 1: Simulink DC motor electro-mechanical model.

The model is driven with the Simulink PID continuous controller block in Figure 2 configured to a set point of half-speed 23.44 radians/s, 5 Volt output ceiling, with K_p, K_i, and K_d coefficients of 8, 2, and 1 respectively, with clamping anti-integrator windup. Figure 5 plots the closed-loop transient response.

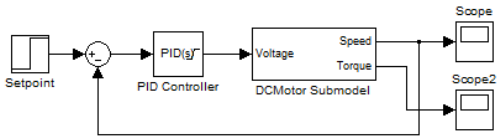


Figure 2: Simulink continuous PID controller.

4.1.2 Experiment Setup: 2-Simulator Configuration

The controller is refined to a software-based PID difference equation algorithm with PWM actuators hosted on the 9S12 microcontroller, simulated with TExaS (Valvano 2011) at the cycle-estimating, instruction-set architecture (ISA) level. The SimConnect/SimTalk signal structure is given in Figure 3.

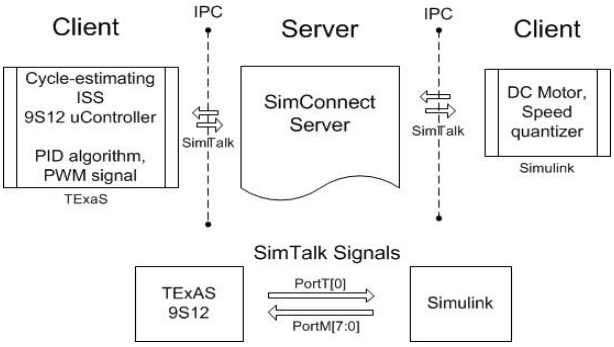


Figure 3: 2-Simulator configuration.

The software-based PID algorithm in 9S12 assembly is adapted from (Valvano 2011). Conversion of a continuous-time frequency-domain specified controller to a digital controller is covered in (Franklin et al. 2002). The refinement uses a free-running 1 kHz sampling rate PID main loop of sixty-three 9S12 assembly instructions, and a total code length of 158 instructions. The algorithm also incorporates anti-integrator windup and output limit checking. The refined Simulink model is given in Figure 4. The "socket_input" and "socket_output" S-Functions register SimTalk signals PortT[0] and PortM[7:0] for exchange with the SimConnect server. The PortT[0] digital signal is the PWM wave generated by TExaS. The 0/1 signal is amplified to 5 Volts for application to the motor terminals. The PWM wave and voltage in this configuration is modeled as ideal (zero rise/fall time).

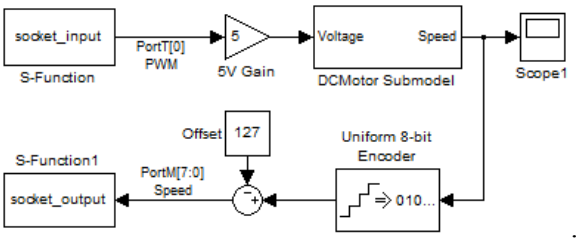


Figure 4: Simulink DC motor model with SimTalk I/O interface.

4.2 Static Simulation Resolution Results

The 2-simulator model is conducted first at a static 100 µs IE ($t_n - t_m$) resolution on signals PortT[0] and PortM[7:0]. The transient response is plotted against the Simulink-only classical continuous case in Figure 5 to verify the functionality of the distributed modeling of the digital PID/PWM microcontroller-based simulation versus the Simulink continuous-time 1-simulator controller.

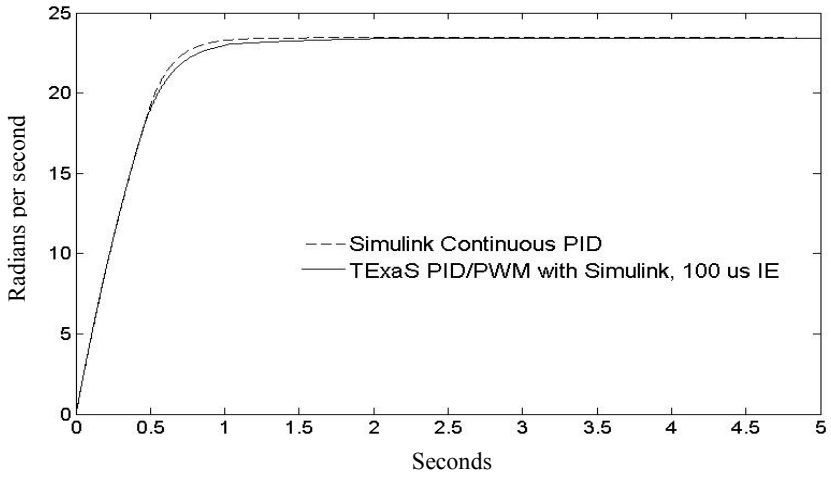


Figure 5: Model output speed versus time with Simulink-only and 2-simulator PID control.

The simulators and SimConnect backplane were hosted over two Microsoft Windows™ 8 OS, Intel dual-core Celeron™ processor machines, each with 4 GB of DDR3 RAM and connected over a wired Ethernet-based LAN. In six experiments, Cases A through F, the static IE resolution was increased from coarse resolution for finer resolution (10 ms to 10 µs) in order to determine at what IE resolution simulation speed became limited due to increasing messaging costs for these particular host machines. As IE resolution decreases, however, signal accuracy may decrease because models share events less often. The maximum percent error of measurement of the motor rotor speed against the set point within the last two seconds of the simulation was used as a metric for the rotor run-up time simulation accuracy. Simulation runtime versus IE duration and simulation accuracy is plotted in Figure 6.

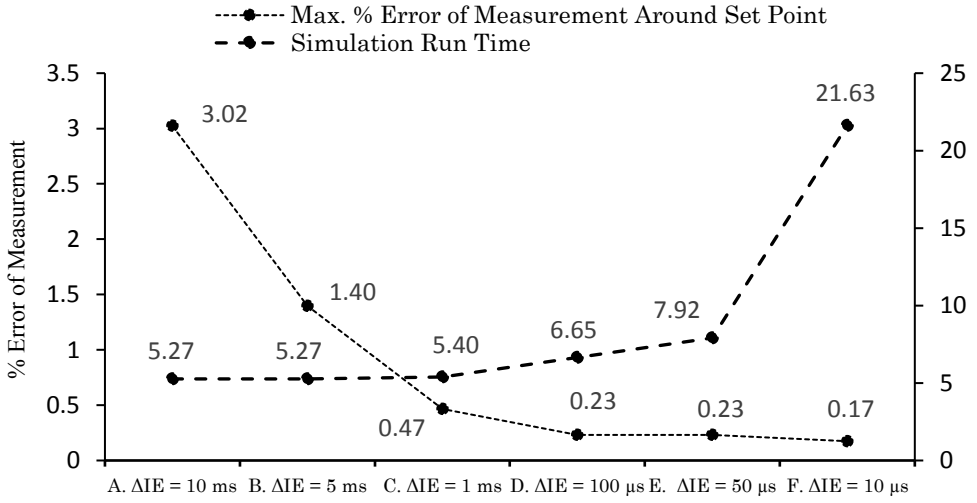


Figure 6: Simulation runtimes and accuracy by simulation case ΔIE configuration.

In Figure 6, as the static ΔIE duration is gradually decreased from 10 ms to 10 µs over simulation Cases A through F, the simulation run time increases more than four times over the lowest resolution Case A due to increasing IE traffic and SimTalk messages. As ΔIE becomes less than or equal to 1 ms after Case C, the simulation latency becomes message bound, increasing to 21 minutes in simulation time for Case F. Each case simulated 5 seconds of model time, but Case F exchanges four million SimTalk messages at highest resolution compared to Case A with only four thousand SimTalk messages. Until Case D resolution (100 µs), the simulation latency is bounded by waiting on computation cycles in the simulators, with TExaS having the highest computation latency of the simulator and backplane mix.

4.3 Adaptive Simulation Resolution Results

In (Pfeifer et al. 2013), IE resolution change points were pre-programmed manually in the SimConnect backplane before the simulation based on some prior knowledge of the rotor speed signal profile. This has a drawback of requiring prior knowledge of the curve properties.

It is desired to have the SimConnect backplane adaptively adjust the simulation IE resolution independently. In the adaptive experiment, a monitor that captures the central differences-based 2^{nd} derivative estimator was attached to the IE FIFO in the SimConnect backplane that forwards the motor rotor speed IE stream. A resolution decision criteria was chosen as follows: where the 2^{nd} derivative was non-zero, IE resolution was increased to capture potentially changing signal detail. Where the 2^{nd} derivative was zero (stable rate of change of the signal), the IE resolution was gradually decreased to relax detail and messaging traffic.

The monitor was parameterized in experiments in terms of how the resolution is decreased (by a multiplying factor or additive constant), how the resolution is increased (by multiplying factor, additive constant, or immediate return to minimum resolution), and how often (sample interval) the 2^{nd} derivative is evaluated on the IE FIFO. The monitor kept minimum 3-deep IE history of the stream in order to evaluate a 2^{nd} order central difference approximation.

In Figure 7, with times and counters given in Table 1, experiment Cases G to P vary the monitor resolution parameters. In Case G, a multiplicative constant of 5 was chosen to increase or decrease resolution, with the 2^{nd} derivative measured every 5 IE samples, with an initial IE duration of 10 µs. When the 2^{nd} derivative was zero, the resolution was decreased by multiplying the current IE resolution by 5. When the 2^{nd} derivative was non-zero, the resolution increased by dividing the current IE resolution by 5.

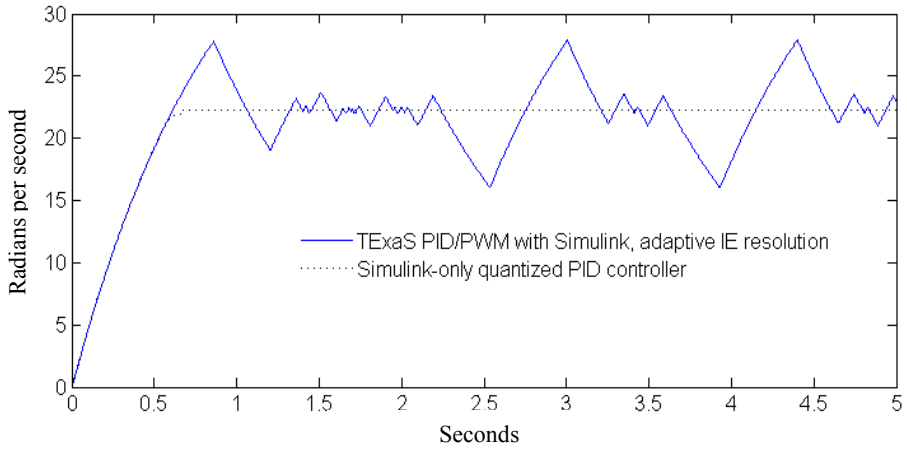


Figure 7: Case G motor speed versus time.

Although the message count for Case G was reduced to 1830 SimTalk messages, with IE variation ranging from 10 µs to 156.25 ms, the simulation accuracy is far too afield. The rotor speed oscillates widely around the set point as the resolution is varied, with up to 25.19 % error of measurement. With the message

count lowered, the simulation runtime was compute-bound, taking 5m 28s to execute, which is comparable to the run time static 5 ms IE duration experiment Cases A and B, but with lower accuracy. 78 SimTalk resolution change messages were sent independently by the SimConnect backplane during the simulation based on the 2^{nd} derivative monitor tuning. These are more change resolution points than would be desired to enter manually as in previous work (Pfeifer 2013).

To improve, the monitor parameters were changed to additive constants for decreasing resolution. In Table 1 Cases H through P, resolution was increased or decreased by an additive constant each time the 2^{nd} derivative was evaluated. In Case J, accuracy was improved by increasing the IE duration by 1 ms each resolution relaxation change or decreasing it by 1 ms each resolution increase change.

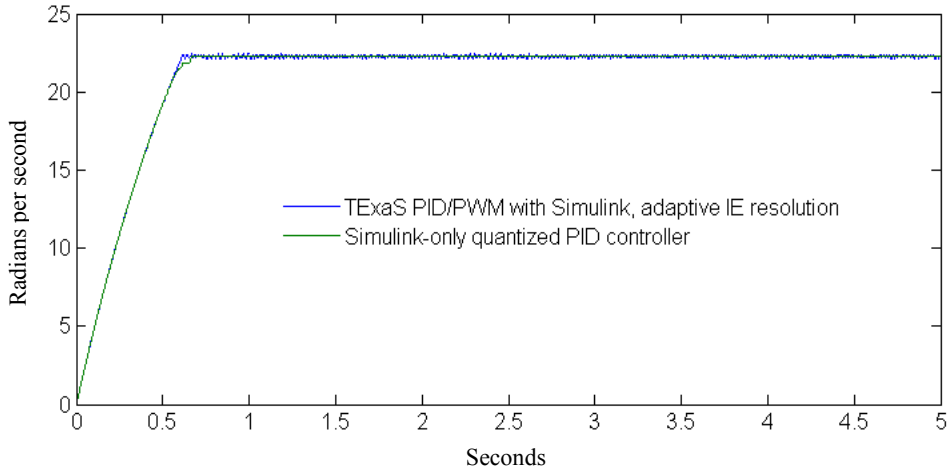


Figure 8: Case H motor speed versus time.

In the Figure 8 profile, the percent error of measurement around the set point was reduced to 0.76 % (down from 25.19 % in Case G) while maintaining a comparable runtime (5m 30s). More SimTalk messages were delivered (15,958), and more resolution change messages were sent (668), ranging from IE duration of 3.61 ms to 10 µs. However, the message traffic was still in the range for which the system was compute bound, so simulation run time did not significantly increase. This makes Case H similar in accuracy to static IE Cases B and C, but with less than half of the SimTalk messages for static IE duration Case C (ΔIE = 1 ms).

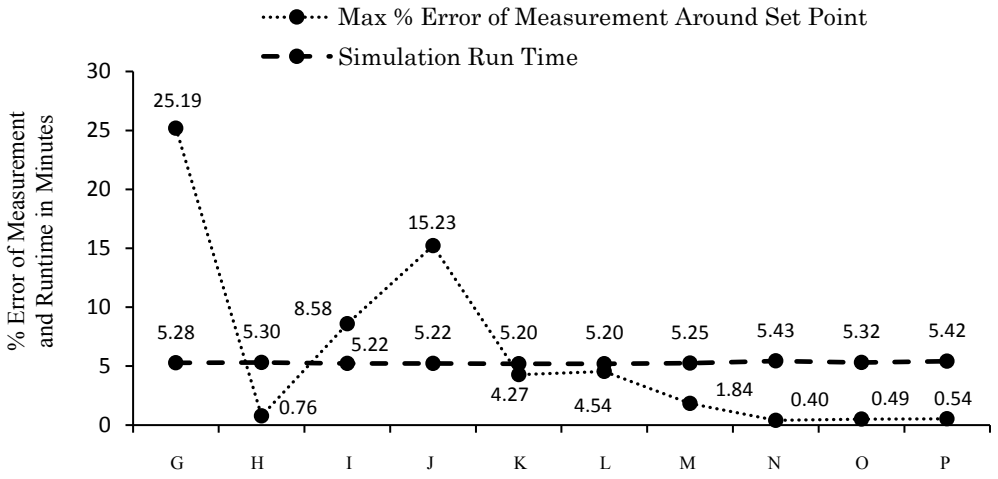


Figure 9: Simulation cases G to P runtimes and accuracy with adaptive IE resolution.

With cases G to P metrics counted in Table 1, Figure 9 shows the simulation run times under adaptive resolution control with expression of accuracy in the maximum percent error of measurement of the rotor speed around the set point in the steady state of the simulation. In the best case, Case N delivered 0.40 percent error of measurement in the set point with 49,710 SimTalk messages, 2,072 resolution change requests, and an adaptive IE resolution range from 3.1 ms to 10 µs.

5 EVALUATION OF RESULTS

Adaptive resolution cases G to P show modest reductions in messaging traffic (five to ten percent message reduction) against static resolution cases for similar simulation times and accuracy. The accuracy of the adaptive resolution monitor used is highly dependent on its run time parameters, particularly in the manner it increases or decreases resolution (whether by a multiplicative or additive factor).

The primary advantage of the adaptive resolution case is that the simulation resolution does not have to be chosen a priori; the simulation can execute without prior knowledge of the controlled motor speed run up profile. In static and pre-programmed dynamic experiments cases A through F, some knowledge of the controlled motor speed profile was required before hand to choose a static resolution or pre-program resolution change points.

Additionally, the adaptive resolution monitor is able to autonomously register many more resolution change requests during the simulation (up to 2,072 in Case N) than a simulation operator might chose to program manually. Reduction of message count, even though modest with these experimental tunings, is also significant for systems where network latency and messaging cost has a fixed lower bound compared to simulator node compute performance.

6 CONCLUSION

Adaptive simulation resolution control with the Kahn Process Network and Interpolated Event (KPN-IE) method of parallel and distributed simulation is shown in this work to reduce simulation messaging traffic while retaining simulation run time and accuracy compared to static resolution cases for the simulated system of a software-based PID/PWM controller of a DC motor. Adaptive resolution control is implemented in the SimConnect KPN backplane, freeing the simulation operator from having to modify potentially closed architecture internal simulator code of engineering simulators. In this work a simple *2nd* order central-differences approximation of an IE stream second derivative was applied to adaptively adjust IE stream resolution in the SimConnect backplane. Where the second derivative was zero, resolution was decreased. Where the second derivative was non-zero, resolution was increased.

The primary advantage of adaptive resolution control with the KPN-IE method and SimConnect/SimTalk tools is that participating simulators are not required to have any notion of lookahead, or register any dynamically changing lookahead values with the coordinating backplane. Rather, the properties of dynamic Interpolated Event duration provide a means of signal resolution relaxation or restriction, and the KPN dataflow properties provide simulation synchronization that meets the local causality constraint for conservative simulation. This enables commercial engineering simulators to be coordinated if they present a user-level coding interface sufficient to host a SimTalk software connector. Because the adaptive control scheme is performed by the backplane, it can reduce the functional interfacing requirements for participating simulators, hoping to reduce the burden of interfacing the many diverse engineering simulators that can be required for cyber-physical system simulation.

Table 1: Adaptive ΔIE simulation configuration, times, counters and traffic.

A	B	C	D	E	F	G	H	I	J	K
G	78	3	1,830	25.19	ΔIE / 5	ΔIE * 5	156 ms	10 µs	5:28	2
H	668	3	15,958	0.76	ΔIE - 100 µs	ΔIE + 100 µs	3.61 ms	10 µs	5:30	2

I	126	3	3,014	8.58	ΔIE - 10 ms	ΔIE + 10 ms	40 ms	10 µs	5:22	2
J	40	3	942	15.23	ΔIE - 1 ms	ΔIE + 10 ms	56 ms	10 µs	5:22	2
K	174	3	4,166	4.27	ΔIE - 500 µs	ΔIE + 1 ms	22 ms	10 µs	5:20	2
L	314	3	7,502	4.54	ΔIE – (ΔIE – 1 ms)	ΔIE + 5 ms	11 ms	10 µs	5:20	2
M	842	3	20,198	1.84	ΔIE – (ΔIE – 0.1 ms)	ΔIE + 1 ms	10.1 ms	10 µs	5:25	2
N	2,072	3	49,710	0.40	ΔIE – (ΔIE – 0.5 ms)	ΔIE + 0.5 ms	3.1 ms	10 µs	5:43	2
O	1,468	3	35,190	0.49	ΔIE – (ΔIE – 10 µs)	ΔIE + 0.5 ms	4 ms	10 µs	5:32	2
P	1,932	3	46,334	0.54	ΔIE – (ΔIE – 1 ms)	ΔIE + 0.5 ms	3 ms	10 µs	5:42	2

Column Legend

A Experiment case with distributed simulators: Simulink(S), TExaS(T), SimConnect(I)
B Number of resolution change messages
C Number of instantiated SimTalk connectors
D Number of SimTalk messages
E Max. % error of measurement of set point in steady state
F Resolution increase method
G Resolution decrease method
H Maximum resolution
I Minimum resolution
J Simulation execution time (minutes:seconds)
K Number of host machines

REFERENCES

Chandy, K. M. and J. Misra. 1979. "Distributed Simulation: A Case Study in Design and Verification of Distributed Programs." In *IEEE Transactions on Software Engineering* SE-5, 5 Sep 1979.

Chung, M.-K. and C.-M. Kyung. 2006. "Improving Lookahead in Parallel Multiprocessor Simulation Using Dynamic Execution Path Prediction." In *The 20th Workshop on Principles of Advanced and Distributed Simulation*, 11-18.

Franklin, G., J.D. Powell, and A. Emami-Naeini. 2002. *Feedback Control of Dynamic Systems*. 4th ed. Prentice Hall, Upper Saddle River, NJ.

Fujimoto, R.M. 1989. "Parallel Discrete Event Simulation." In *Proceedings of the 1989 Winter Simulation Conference,* edited by K. J. Musselman, P. Heidelberger, and E. A. MacNair, 19-28. Washington, D.C.

Fujimoto, R. M. 1995. "Parallel and Distributed Simulation." In *Proceedings of the 1995 Winter Simulation Conference*, edited by W. R. Lilegdon, D. Goldsman, C. Alexopoulos, and K. Kang, 118-125. Arlington, VA.

Fujimoto, R. M. 2000. *Parallel and Distributed Simulation Systems*. Wiley, New York, NY.

Golub, G. and J. Ortega. 1993. *Scientific Computing: An Introduction With Parallel Computing*. Academic Press, San Diego, CA.

IEEE. 2010. "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)--Framework and Rules." *IEEE Standard* 1516-2010. 1-38.

Jefferson, D. R. 1985. "Virtual Time." In *ACM Transactions on Programming Languages and Systems* 7, 3. July 1985.

Kahn, G. 1974. "The Semantics of a Simple Language for Parallel Programming." *Information Processing*, Stockholm, Sweden, 471-475.

Klesh, A. T., J. W. Cutler and E. M. Atkins. 2012. "Cyber-Physical Challenges for Space Systems." In *The 2012 IEEE/ACM Third International Conference Cyber-Physical Systems (ICCPS)*, 45-52.

Kuhl,F., R. Weatherly and J. Dahmann. 1999. *Creating Computer Simulation Systems: An Introduction to the High Level Architecture.* Prentice Hall, NJ.

Lee, C., E. Coe, J.M. Clark, J. Stepanek, C. Raghavendra, S. Bhatia, and R. Puri. 2002. "Scalable Time Management Algorithms Using Active Networks for Distributed Simulation." In *Proceedings of the DARPA Active Networks Conference and Exposition,* 366- 378.

Lee, E. A. 2008. "Cyber-Physical Systems: Design Challenges. The University of California at Berkeley Center for Hybrid and Embedded Software Systems." Technical Report No. UCB/EECS-2008-8.

Lee, E. A. and A. Sangiovanni-Vincentelli. 1996. "Comparing Models of Computation." In *The IEEE/ACM International Conference on Computer-Aided Design Digest of Technical Papers*, 234-241.

Lungeanu, D. and C. Shi. 1999. "Distributed Simulation of VLSI Systems via Lookahead-free Self-adaptive Optimistic and Conservative Synchronization." In *The 1999 IEEE/ACM International Conference on Computer-Aided Design*, 500-504.

National Science Foundation. 2006. CPS. http://www.nsf.gov/pubs/2012/nsf12520/nsf12520.htm.

Pfeifer, D. and A. Gerstlauer. 2011. "Expression-level Parallelism for Distributed Spice Circuit Simulation." In *The 15th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications*, 4 Sep 2011.

Pfeifer, D. 2013. "Parallel and Distributed Cyber-Physical System Simulation." Ph.D. thesis, Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX.

Pfeifer, D. and J. Valvano. 2011. "Kahn Process Networks Applied to Distributed Heterogeneous HW/SW Cosimulation." In *The 2011 Electronic System Level Synthesis Conference*. 5-6 June 2011.

Pfeifer, D., J. Valvano and A. Gerstlauer. 2013. "SimConnect and SimTalk for Distributed Cyber-Physical System Simulation." In *Simulation: Transactions of the Society for Modeling and Simulation International* 89, 10 (2013), 1254-1271.

Pfeifer, D., J. Valvano and A. Gerstlauer. 2013. "Dynamic Resolution in Distributed Cyber-Physical System Simulation." In *Proceedings of the 2013 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, 277-284.

Rajkumar, R., I. Lee, L. Sha and J. Stankovic. 2010. "Cyber-Physical Systems: The Next Computing Revolution." In *The 2010 ACM/IEEE 47th Design Automation Conference*, 731-736.

Sangiovanni-Vincentelli, A. 2007. "Quo Vadis, SLD? Reasoning about the Trends and Challenges of System Level Design." In *Proceedings of the IEEE* 95, 3 March 2007, 467-506.

Turner, P. 2001. *Guide to Scientific Computing*. 2nd ed. CRC Press, FL.

Valvano, J. 2011. *Embedded Microcomputer Systems: Real Time Interfacing*. 3rd ed. Cengage Learning, Stamford, CT.

Zhiwu, D. and L. Yanfeng. 2009. "Dynamic Time Management Algorithms Research in Simulation System HLA-Based." In *The Second International Workshop on Computer Science and Engineering*, Vol. 1, 580-583.

AUTHOR BIOGRAPHIES

DYLAN PFEIFER is a Senior Firmware Engineer at Power Standards Lab. He received his Ph.D. in Electrical and Computer Engineering from The University of Texas at Austin in 2013 specializing in cyber-physical system simulation. His email address is dcpfeifer@gmail.com.

ANDREAS GERSTLAUER is Assistant Professor in Electrical and Computer Engineering and at The University of Texas at Austin. Prior to joining UT Austin in 2008, he was an Assistant Researcher in the Center for Embedded Computer Systems (CECS) at UC Irvine. His email address is gerstl@ece.utexas.edu.

JONATHAN VALVANO is Professor in Electrical and Computer Engineering at The University of Texas at Austin, performing research in the fields of embedded systems, simulation, and medical instrumentation. His email address is valvano@mail.utexas.edu.