Hierarchical Modeling of Mode-Switching Systems

James E. Weimer and Bruce H. Krogh
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
{jweimer | krogh}@ece.cmu.edu

Keywords: mode-switching, hybrid systems, block diagrams, hierarchical models

Abstract

This paper introduces a set of input-output blocks to build intuitive models of dynamic systems with mode switching due to changes in operating conditions, sensor failure, model reconfiguration, and model-order reduction. These blocks make it possible to construct hierarchical component models of mode-switching systems. An application to an automotive steer-by-wire example and Simulink simulation results are presented.

1. INTRODUCTION

Models of dynamic systems often exhibit multiple modes of behavior, reflecting different physical operating regimes or controller switching. Simulation languages typically include primitives from which multi-mode models can be developed, but they often lack structures for creating these models in a systematic way. This paper presents a set of input-output blocks designed to support the modular and hierarchical construction of models of dynamic systems with mode switching.

There are several types of mode switching in dynamic models. Basic mode switching includes discontinuous changes in the dynamic model parameters due to signals saturating or crossing specified thresholds. Mode transitions with memory, such as in systems with hysteresis or controllers with finite-state switching logic, lead to hybrid dynamic systems (models with both continuous and discrete states) [6]. More complex mode switching could include changes in the order of the dynamic model or complete reconfiguration of the interconnections between subsystems [8]. Significant changes in the dynamic model typically require a reinitialization of the continuous state when the mode changes [5]. The block set proposed in this paper provides structures for supporting all of these types of mode-switching behaviors while retaining flexibility for defining the discrete and continuous dynamics for each mode.

Charon [1], HyVisual [3], and Simulink (with the Stateflow toolbox) [7] are three frameworks capable of modeling mode-switching systems. Charon can specify a hybrid system in a hierarchical way, but does not use the input-output block structure familiar to control system engineers. HyVisual contains everything necessary to model mode-switching systems, but does not provide a user-friendly way to build mode-switching systems from component models. Simulink and Stateflow do not naturally lead to a hierarchical model (although hierarchical models are possible), since alternating discrete-state and continuous-state hierarchy is not allowed.

The following section presents the five proposed input-output blocks with brief descriptions of their structure and purpose. Section 3 describes the use of the block set to model an automotive steer-by-wire system that includes mode switching due to torque sensor failure, driver modes, and model-order reduction in certain operating regimes. Simulation results in Sect. 4 illustrate mode-switching behaviors for the steer-by-wire system using realizations of the blocks in Simulink. The concluding section summarizes the contributions of this paper and directions for further research.

2. INPUT-OUTPUT BLOCKS FOR MODE SWITCHING SYSTEMS

To build hierarchical models of mode switching systems, the following blocks are introduced: *basic system block*, *composite system block*, *mode block*, *supervisor block*, and *hybrid system block*.

2.1. Basic System Block (BSB)

The BSB shown in Figure 1 contains the continuous-state dynamics and generates its own state initial conditions. The continuous-state dynamics evolve only when the BSB is *active*. When a transition from *passive* to *active* occurs, the continuous states are initialized and evolve according to the state dynamics. In hierarchical models of mode-switching systems, BSBs are the fundamental building blocks containing all of the equations defining the continuous-state dynamics (continuous-time or discrete-time) in the system.

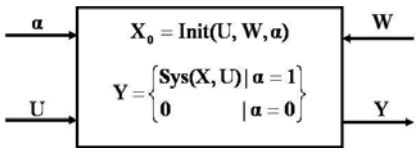

Figure 1. Basic system block (BSB).

A BSB has signals U, Y, W, and α corresponding to the

 ISBN # 1-56555-316-0

input, output, initial condition input, and activate input respectively. A BSB contains the system states, X, and the state dynamics, Sys. When the block is active ($\alpha = 1$), the system states evolve and $Y = Sys(X,U)$. When the block is passive ($\alpha = 0$), the system states do not evolve and $Y = 0$. The system states are initialized when a block transitions from passive to active according to the initial condition generator function $Init(U,W,\alpha)$.

2.2. Composite System Block (CSB)

The CSB provides the capability to create dynamic system components from the interconnection of a set of subsystem components. It can contain any number of dynamic system blocks (BSBs, CSBs, and HSBs (defined below)). Figure 2 shows an example CSB, where the solid black circle represents a multiplexing operation. (In later figures, hollow black circles represent a demultiplexing operation.) A CSB has the same inputs and outputs as a BSB. A signal routing function defines the activate input for each system block inside a CSB to be the active input of the CSB. This results in all the contained system blocks being *active* or *passive* simultaneously. Other signal routing functions define the connection of system blocks contained in a CSB to the inputs and output of the CSB.

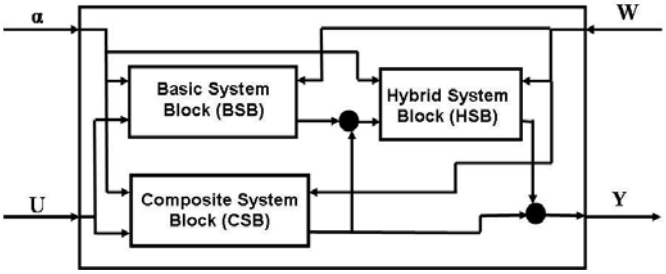

Figure 2. Composite system block (CSB).

2.3. Mode Block (MB)

The MB is a parallel grouping of system blocks representing different models for a mode-switching component. Since all system blocks have the same input-output structure, their placement within a MB is arbitrary, as shown in Figure 3, where the stripped box represents the switching between which system block drives the output of the MB. Only one system block is ever active at an instant. Additionally the output of the MB is connected to the initial condition input of every system block within the MB. MBs contain sets of modes that can be selected by an associated supervisor block SB (defined below).

A MB has an input and output the same as a BSB. A mode select input, Q, determines which system block is active by setting the corresponding active state, α_n, to 1. Signal routing functions define the interconnection of signals between the

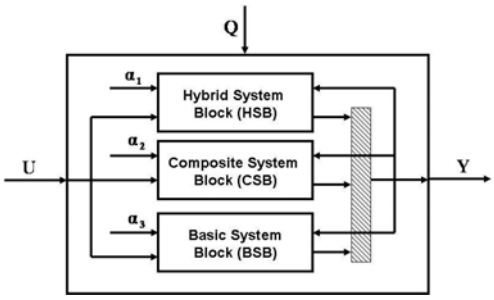

Figure 3. Mode block (MB).

system blocks contained by a MB with the inputs and output of the MB.

2.4. Supervisor Block

A SB consists of two user defined functions, Sup^R and Sup^Q, which choose the operating mode, as shown in Figure 4. A SB monitors all input and output signals for the MBs being supervised in a hybrid system block (defined below), and chooses the operating region for each MB accordingly.

The SB continuously monitors the signals, updates its state, and chooses the operating mode, while the actual mode switching and system dynamics are performed inside the MBs. When passive, the SB and all the system blocks within all the MBs being supervised are also passive.

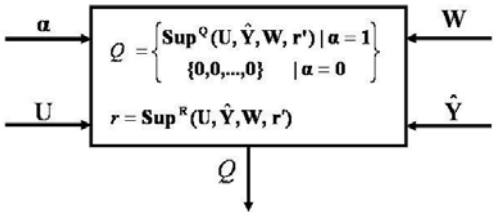

Figure 4. Supervisor block (SB).

A SB has inputs of U,α,W, and $\hat{Y}$ representing the input, activate input, initial condition input, and outputs respectively. A SB has a supervisor output, Q.

2.5. Hybrid System Block

An HSB contains exactly one SB and a system of MBs, as shown in Figure 5. Through the use of hybrid system blocks, mode switching and model order switching systems are implemented. The HSB allows modeling flexibility since it can be interchanged with BSBs and CSBs. In a complete model, MBs and SBs exist only within HSBs. An HSB has the same inputs and outputs as the BSB and CSB. Signal routing functions define the interconnection of signals within an HSB.

3. MODELING MULTI-MODE SYSTEMS

This section discusses how to model multi-mode systems, applying the blocks presented in the previous section. We

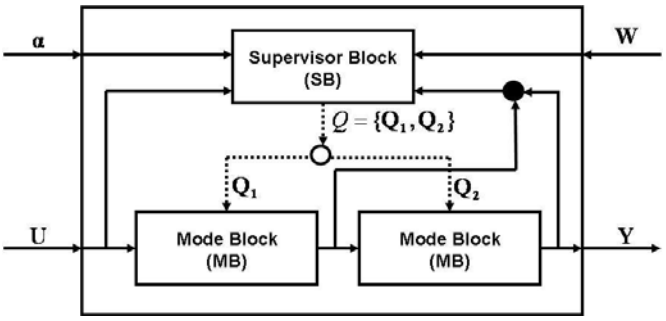

Figure 5. Hybrid system block (HSB).

use the steer-by-wire system shown in Figure 6 to illustrate the features of the proposed methodology. In comparison to conventional steering mechanisms, steer-by-wire systems can improve steering performance and vehicle handling by varying the response to driver inputs to accommodate variations in the vehicle dynamics [11].

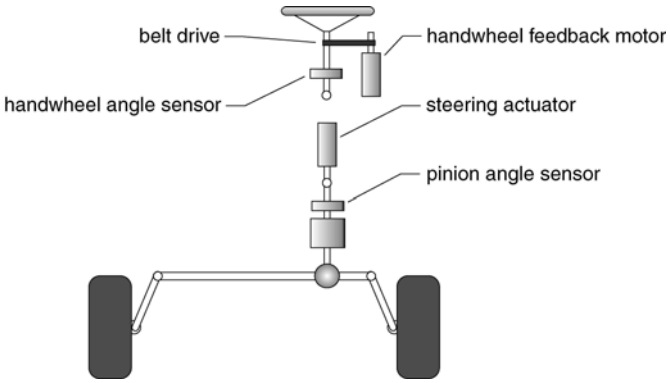

Figure 6. Steer-by-wire system (from [10]).

Figure 7 (on the following page) shows the steer-by-wire system model. Components of each block are labeled (HSB, CSB, SB), which describes the block. The driver and steering controller are HSBs since they contain mode switching, while BSBs model the labeled components. By modeling systems in this fashion, individual component models can be updated without changing the structure of the entire system.

3.1. Basic Dynamic Components

In the steer-by-wire example, there are three basic dynamic components: digital communication, steering wheel dynamics, and vehicle dynamics. The steering wheel dynamics receives a torque input from the driver and a signal from the digital communication representing the feedback torque. The digital communication is a bidirectional network. The steering controller receives the steering wheel angle encoder measurement from the steering wheel dynamics and the steering wheel dynamics receives the digital feedback torque signal from the steering controller.

The vehicle dynamics are modeled using the bicycle model approximation presented in [4]. The bicycle model assumes only two wheels, and computes the tire sideslip angle, vehicle yaw rate, and steering aligning torque from the vehicle velocity and steering angle. The vehicle dynamics produce the steering angle.

3.2. Hybrid System Components

To model the steer-by-wire system, the driver's response must be considered. We assume the driver operates in two different modes: force-input mode and position-servo mode. In the force-input mode, the driver applies an uncontrolled torque force (a disturbance) to the steering wheel, such as releasing or jerking the steering wheel. In the position-servo mode, the human uses sensory feedback information and a desired trajectory to steer the vehicle by applying a controlled torque to the steering wheel. The position-servo mode models normal steering.

Figure 8 shows a model of the driver's modes of operation. The left MB has a single mode, representing the desired trajectory, while the right MB contains the control strategies. Both control strategies use the same subsystem to generate the desired trajectory; however, by modeling the system as in Figure 8, a change in the desired trajectory model only affects one block. This allows easy model updating through the modeling flexibility inherent with the proposed block structures.

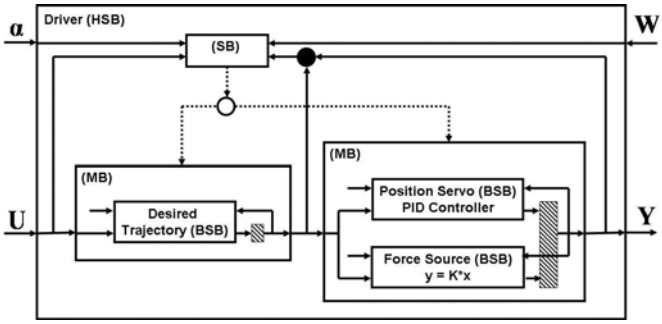

Figure 8. Driver modes.

The steering controller in a steer-by-wire system can contain many modes. A candidate controller involves a torque sensor to measure accurately the aligning torque generated by the vehicle dynamics [2]. If the torque sensor were to fail, the operating mode would change, whereby the controller would operate without a torque sensor. The steering controller uses the desired steering angle from the digital communication and all available sensory data from the vehicle dynamics to generate a steering actuator control signal.

Figure 9 shows the steering controller model. In the model, the control strategy changes with the availability of a torque sensor measurement. Similar to the driver model, the steering

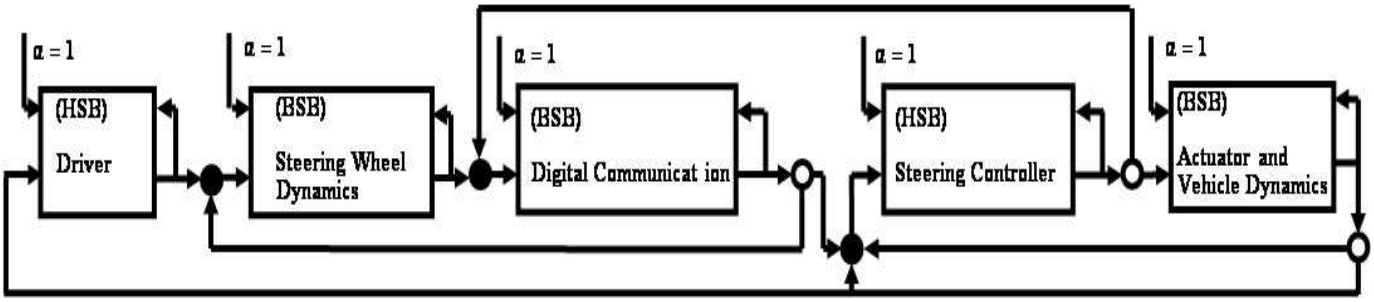

Figure 7. Steer-by-wire system block diagram.

controller design allows for changes to occur without updating multiple blocks. For instance, if the state feedback controller were to change, only the state feedback block would need to be updated.

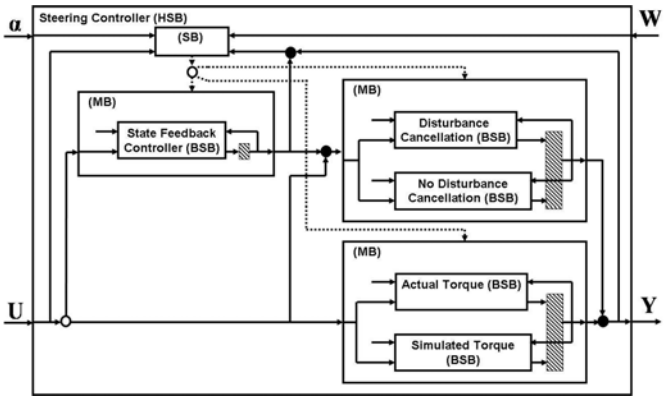

Figure 9. Steering controller block diagram (In all Figures, dotted lines represent the SB block output).

In certain operating modes, the driver and steering wheel dynamics may be modeled with a reduced-order system, where the actual steering wheel angle can be modeled as the desired steering angle. This leads to a reduced-order model where the human and steering wheel dynamics can be neglected. Figure 10 (on the following page) shows the reduced-order block diagram. The reduced-order model will improve computation time at a cost of accuracy. The loss in accuracy varies in different operating modes; thus the reduced-order model should only be used in operating regions where the accuracy loss is acceptable. In more critical regions, the full-order model is desired to more realistically approximate the system's response. The ability to expand the system to incorporate reduced-order models is an attribute of the proposed modeling framework.

To build a model order switching system, we begin with the full order system (shown in Figure 10). With a CSB, we create a subsystem containing the portion of the full order system being reduced. This CSB and a system block representing the reduced-order system are placed inside a MB. A HSB contains the MB and a SB controls to use the full or reduced-order system as shown in Figure 11. Since the HSB has the same input-output structure as the CSB representing the original full order system, the overall system structure does not change. The ability to change model order within a single system can drastically reduce computation time.

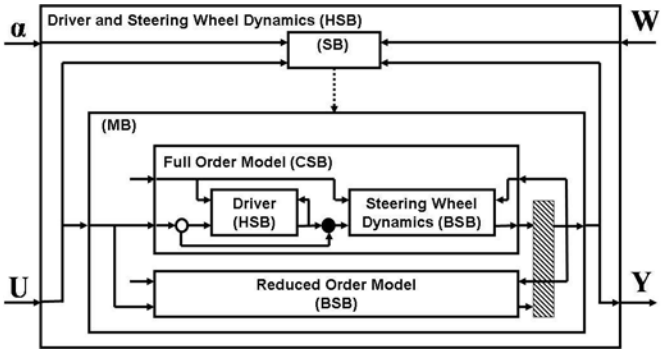

Figure 11. HSB containing both full-order and reduced-order models for driver and steering wheel.

4. SIMULATION RESULTS

4.1. Simulink Implementation

Simulink contains all the necessary components to model mode-switching systems, but lacks the structure to model mode-switching systems both hierarchically and conveniently. Implementing the proposed blocks in Simulink as subsystem blocks enhance Simulink by providing the needed structure effectively model mode-switching systems.

The BSB shown in Figure 1 can be implemented in Simulink as in Figure 12. The state equations subsystem contains the continuous dynamic equations, while the initial condition generator determines the initial conditions.

The CSB shown in Figure 2 can be implemented in Simulink as in Figure 13. Since the CSB is a subsystem block, it is implemented by using subsystem blocks. The multiplexers and demultiplexers are used to route signals accordingly.

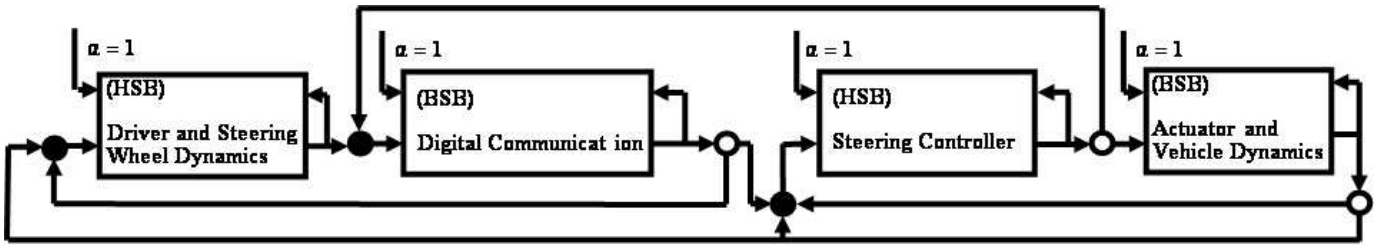

Figure 10. Reduced order Steer-by-wire system block diagram (top level).

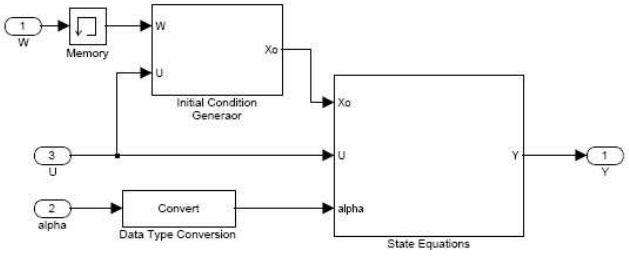

Figure 12. Simulink implementation of a BSB.

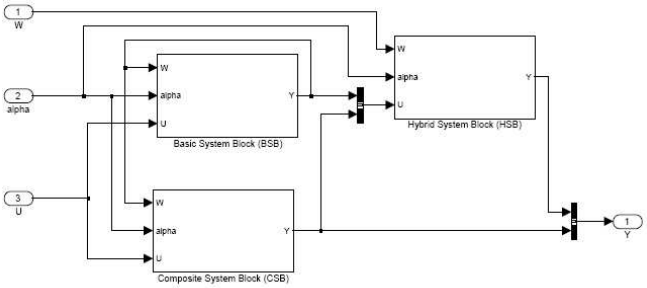

Figure 13. Simulink implementation of a CSB.

The MB shown in Figure 3 can be implemented in Simulink as in Figure 14. Similar to the CSB, the modes are implemented using subsystem blocks, and switches with comparative logic are used to select the mode.

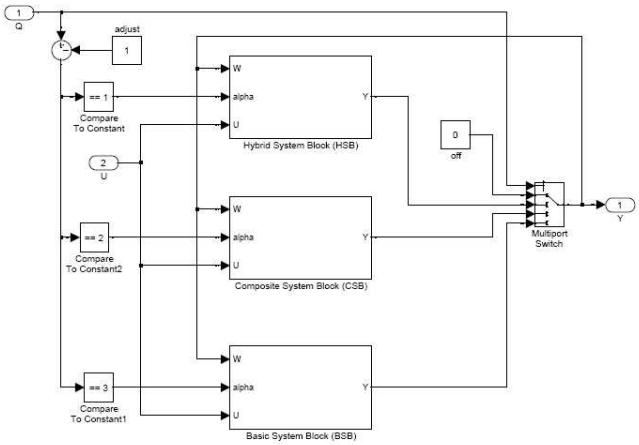

Figure 14. Simulink implementation of a MB.

The SB shown in Figure 4 can be implemented in Simulink as shown in Figure 15. The SB consists of a finite-state machine implemented with a stateflow block and an event generator, which monitors the input signals and creates events for the stateflow block.

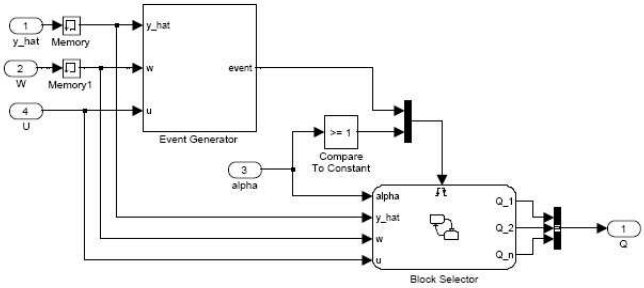

Figure 15. Simulink implementation of a SB.

Finally, the HSB shown in Figure 5 can be implemented in Simulink as shown in Figure 16. The HSB uses subsystem

blocks to represent the MBs and SB. As in the other blocks, the multiplexer and demultiplexer are used to route the signals.

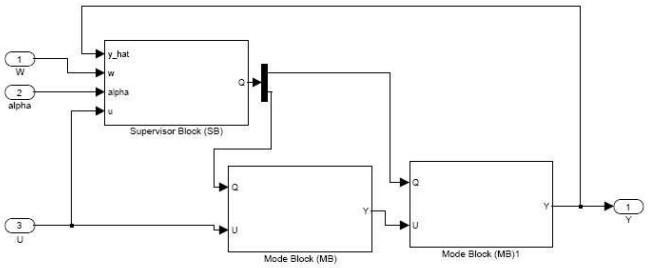

Figure 16. Simulink implementation of a HSB.

As part of the Simulink implementation, additional specifications are needed to complete the block translations. In the BSBs, the initial condition generator and dynamic-state equations must be defined. In the MBs, additional comparative logic and switch inputs are added according to the number of system blocks in the MBs. The SBs need the event generator and stateflow block to be specified. Finally, all the blocks $(\mathrm{BSB}, \mathrm{CSB}, \mathrm{MB}, \mathrm{SB}, \mathrm{HSB})$ need multiplexers, demultiplexers, and signal routing performed according to their respective block definitions.

These blocks have been implemented as a Simulink library.

4.2. Simulation Results

To demonstrate the proposed modeling framework, the steer-by-wire system is implemented in Simulink by using the block translations above. The first simulation assumes no mode-switching and uses the full order system. The desired trajectory is a sine wave of $amplitude = 0.526$ and $frequency = 6.2 rads/sec$ to represent a driver continuously turning the steering wheel. Figure 17 shows that the actual steering angle closely follows the desired steering angle with phase lag of 0.15 seconds. This phase lag is caused by the dynamics inherent in the steer-by-wire system.

In the steer-by-wire example, a torque sensor measures the aligning torque generated by the vehicle dynamics for the steering controller. In a system with sensory information for control, the effect of a sensor failure must be addressed. In the event of a torque sensor failure, the steering controller will no longer have the measured aligning torque, and will use a different control strategy. For the trajectory in Figure 17, a simulation with a torque sensor failure at time $t = 6$ seconds is shown in Figure 18. When the torque sensor fails, there is a noticeable disturbance in the steering angle of 0.51 radians. The effect of this disturbance must be analyzed to ensure safety of the system.

Modeling driver behavior is complex [9]. In this example, the driver is assumed to have two operating modes. In the first

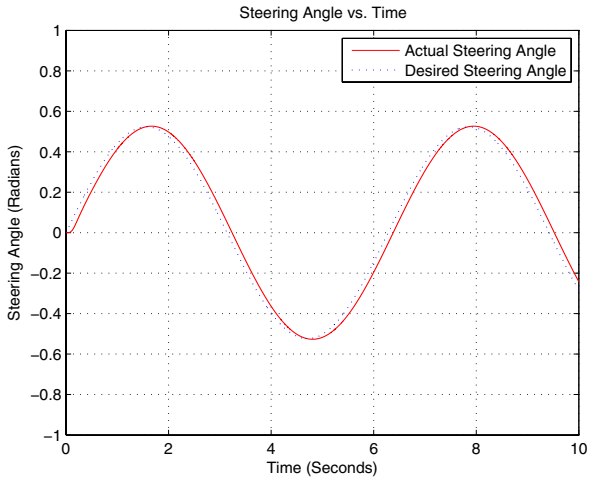

Figure 17. Steer-by-wire system simulation (no mode switching).

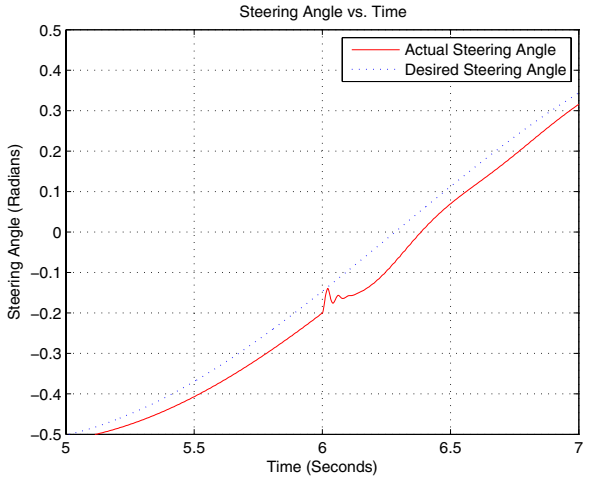

Figure 18. Steer-by-wire system simulation (torque sensor failure at $t = 6$ seconds).

mode, the driver applies an uncontrolled torque and in the second applies a controlled torque based on the desired steering angle. To show how the steer-by-wire system behaves, a simulation where the driver releases the steering wheel, effectively applying a zero torque force to the steering wheel, is shown in Figure 19. When the driver lets go of the steering wheel, mode switching occurs and the driver is modeled as a zero torque source. In this mode, the feedback torque drives the steering wheel until the steering angle generated by the vehicle dynamics becomes zero.

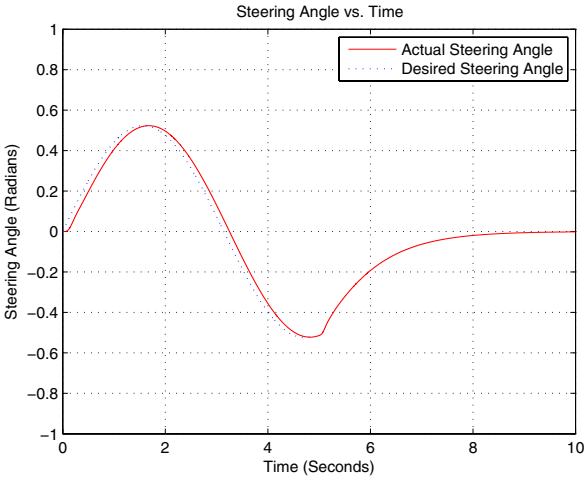

Figure 19. Steer-by-wire simulation (human mode switching at $t = 5$ seconds).

To reduce computation time, reduced-order models are used at a cost of accuracy. A reduced-order model of the steer-by-wire system can be used for the driver and steering wheel dynamics. In the reduced-order mode, the driver and steering wheel dynamics can be modeled as a small delay. Figure 20 shows the simulation results for model order switching. This plot closely follows Figure 17 which did not use model order switching.

When using a variable integration step solver to simulate a mode-switching system, the step size can be analyzed as an indicator of performance. In Simulink, the integration step size is computed dynamically based on system complexity. The integration step size inversely relates to the time a simulation run requires. The larger the step size, the faster the simulation run, and vice versa. Figure 21 shows the integration step size computed by Simulink during the simulation in Figure 20. These results show that the integration step size in mode 2 (reduced-order model) is about 50 times the integration step size in mode 1 (full-order model). Using reduced-order models, when available, can significantly reduce the time required to perform a simulation. For more complex systems, the computation time saved can be significant.

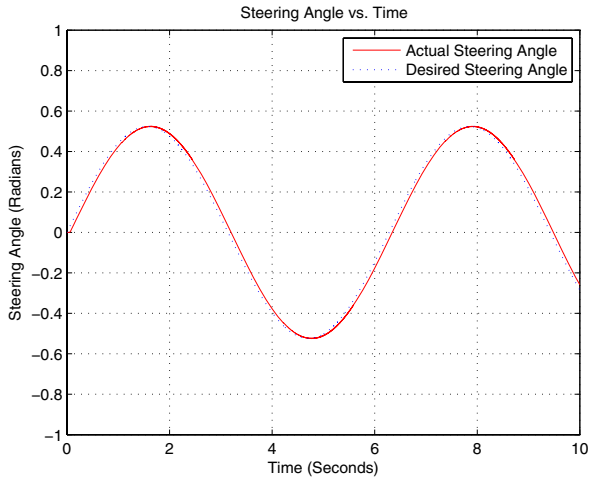

Figure 20. Steer-by-wire simulation (model order switching).

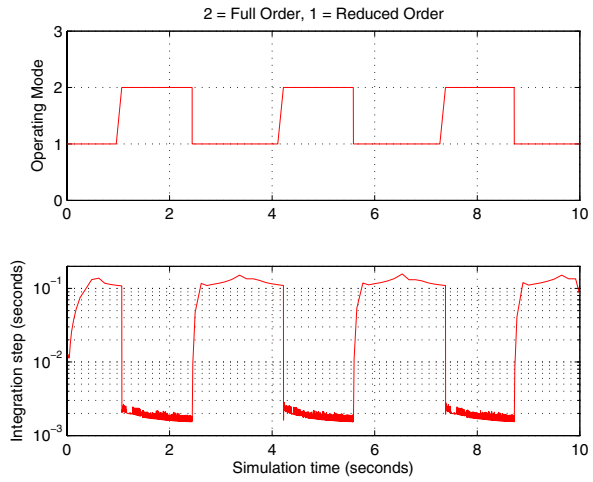

Figure 21. Model order switching and integration step vs. simulation time (step size ratio = 1:50).

5. DISCUSSION

This paper presents a set of input-output blocks for modeling dynamic systems with multiple modes, including modes with varying numbers of continuous state variables and modes defined by interconnections of subsystems. Consequently, hierarchical complex mode-switching behaviors, including changes in model-order and subsystem reconfiguration can be modeled. Dynamic state re-initialization is also supported. Simulation results using Simulink demonstrate the features of the block set. The steer-by-wire example illustrates how complex mode-switching can arise in applications, and how the block set supports the construction of modular models that are easy to develop and modify.

Directions for future research include a thorough analysis of methods for verifying properties of models using the proposed block set. The order of the continuous dynamics in hybrid system models is a principal barrier to effective verification. This provides a further motivation for introducing order-reducing mode switching in models of dynamic systems. Introducing the possibility of switching to lower-order models whenever possible (e.g., by eliminating fast dynamics when transients have died out) may aid in the verification of models of complex hybrid systems.

REFERENCES

[1] Rajeev Alur, Radu Grosu, Yerang Hur, Vijay Kumar, and Insup Lee. Modular specification of hybrid systems in charon. In *HSCC*, pages 6–19, 2000.

[2] Sanket Amberkar, Farhad Bolourchi, Jon Demerly, and Scott Millsap. A control system methodology for steer-by-wire systems. In *Steering and Suspension Technology Symposium*, 2004.

[3] C. Brooks, A. Cataldo, E.A. Lee, J. Liu, X.Liu, S. Neuendorffer, and H. Zheng. Hyvisual: A hybrid system visual modeler. Technical report, EECS Dept., UC Berkeley, July 2005.

[4] C. Gadda, P. Yih, and J. Gerdes. Incorporating a model of vehicle dynamics in a diagnostic system for steer-by-wire vehicles. In *Proceedings of AVEC*, pages 779–784, 2004.

[5] Hans-Michael Hanisch. Closed-loop modeling and related problems of embedded control systems in engineering. In *Abstract State Machines*, pages 6–19, 2004.

[6] Thomas A. Henzinger and Shankar Sastry, editors. *Hybrid Systems: Computation and Control, First International Workshop, HSCC'98, Berkeley, California, USA, April 13-15, 1998, Proceedings*, volume 1386 of *Lecture Notes in Computer Science*. Springer, 1998.

[7] Matlab and Simulnk. www.mathworks.com.

[8] Mohan Ravindranathan and Roy Leitch. Model switching in intelligent control systems. *AI in Engineering*, 13(2):175–187, 1999.

[9] K. Suzuki and H. Jansson. An abalysis of driver's steering behavious during auditory or haptic warnings for designing of lane departure warning system. In *JSAE Review*, pages 65–70, 2003.

[10] P. Yih. Steer-by-wire: implications for vehicle handling and safety, 1995.

[11] P. Yih and J.C. Gerdes. Steer-by-wire for vehicle state estimation and control. In *Proceedings of AVEC*, pages 785–790, 2004.