Building Hybrid Observers for Complex Dynamic Systems Using Model Abstractions

Pieter J. Mosterman[1]* and Gautam Biswas[2]**

[1] Institute of Robotics and System Dynamics DLR Oberpfaffenhofen, P.O. Box 1116, D-82230 Wessling, Germany
Pieter.J.Mosterman@dlr.de
[2] Knowledge Systems Laboratory, Department of Computer Science, Stanford University, Stanford, CA 94305, U.S.A.
biswas@ksl.stanford.edu

Abstract. Controllers for embedded dynamic systems require models with continuous behavior evolution and discrete configuration changes. These changes may cause fast continuous transients in state variables. *Time scale* and *parameter* abstractions simplify the analysis of these transients, causing discontinuities in the state variables. The two abstraction types have a very different impact on the analysis of system behavior. We have developed a systematic modeling approach that introduces formal semantics for behavior generation. This paper discusses the implementation of this scheme in a hybrid observer designed to track embedded system behavior. The resultant observer is based on piecewise simpler continuous models with mode transitions defined between them. Actual mode transitions in the system are provided by a digital controller and directly obtained from measuring physical variables.

1 Introduction

The drive to achieve more optimal and reliable performance on complex systems such as aircraft and nuclear plants while meeting rigorous safety constraints is necessitating detailed modeling and analysis of the embedded controllers for these systems. In embedded systems, the continuous physical process interaction with digital control signals requires modeling schemes that facilitate the analysis of mixed continuous and discrete, i.e., *hybrid* behavior. Discrete phenomena may also occur when modeling abstractions are applied to simplify fast nonlinear continuous process behavior.

Consider the primary aerodynamic control surfaces of an airplane in Fig. 1 [19]. Modern avionics systems employ electronic signals generated by a digital computer, which are transformed into the power domain by electro-hydraulic actuators. The primary flight control system exemplifies the need for hybrid modeling

* Pieter J. Mosterman is supported by a grant from the DFG Schwerpunktprogramm KONDISK.
** Gautam Biswas is on leave from the Department of Computer Science, Vanderbilt University, Nashville, TN.

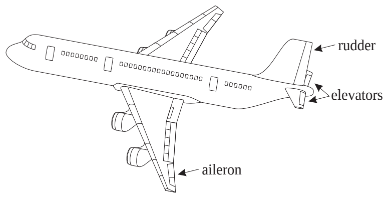

Fig. 1. Aerodynamic control surfaces.

in embedded control systems. At the lowest level in the control hierarchy, positioning of the rudder, elevators, and ailerons is achieved by continuous PID control. Desired set point values are generated directly by the pilot or by a supervising control algorithm implemented on a digital processor. Digital control may mandate *mode* changes at different stages of a flight plan (e.g., *take-off*, *cruise*, and *go-around*). Detection of component failures may lead to discrete changes in system configuration. Model simplification by discretizing fast nonlinear transients also results in discontinuous variable changes.

We have developed a hybrid modeling paradigm that encompasses analysis of embedded systems and modeling abstractions in physical systems [13, 16, 17]. The methodology for abstracting complex transients has been developed into compostional hybrid automata models with formal semantics for computing the discontinuous changes in the system state vector [15].

Observer schemes form a key component in the design and implementation of controller and diagnosis schemes for dynamic systems [10, 18]. In this paper, our focus is on applying our hybrid modeling methodology to develop effective observers for complex dynamic systems. We illustrate the approach by building a hybrid observer, i.e., an observer that includes mode change effects, for the elevator positioning subsystem of the primary flight control system of aircraft. The resultant observer simplifies complex nonlinear models to simpler piecewise linearized models with discrete mode transitions. There exist well-defined robust schemes for constructing observer models for linear systems [4, 10]. Formal model semantics enable us to compute the discontinuous changes in the state vector across mode transitions.

2 Hybrid Modeling of Physical Systems

Hybrid modeling paradigms [1, 7, 17] supplement continuous system description by mechanisms that model discrete state changes resulting in discontinuities in the field description and the continuous state variables. In previous work, we [12, 16] have formulated a systematic approach to hybrid modeling of dynamic physical systems based on a local switching mechanism implemented as finite state automata. The dynamically generated topology in a mode is used to translate the switching specifications to conditions based on state variables. Switching conditions may be expressed in terms of the values immediately before switching occurred, (*a priori* values), or in terms of the values computed by solving the initial value problem for the newly activated mode (*a posteriori* values).

2.1 Definitions

Differential equations form a common representation of continuous system behavior. The system is described by a state vector, x. Behavior over time is specified by field f. Discrete systems, modeled by a state machine representation, consist of a set of discrete modes, α. Mode changes caused by events, σ, are specified by the *state transition function* ϕ, i.e., $\alpha_{i+1} = \phi(\alpha_i)$. A transition may produce additional discrete events, causing further transitions. A mode change from α_i to α_{i+1}, may result in a field definition change from f_{α_i} to $f_{\alpha_{i+1}}$. Discontinuous changes in the state vector are governed by an algebraic function g, $x^+ = g_{\alpha_i}^{\alpha_{i+1}}(x)$. Discrete mode changes are caused by an *event generation function,* γ, associated with the current active mode, α_i, $\gamma_{\alpha_i}(x) \leq 0 \rightarrow \sigma_j$.

2.2 Abstractions in Physical System Models

On a macroscopic level, physical systems are continuous but phenomena may occur at multiple temporal and spatial scales. To simplify system models, small, parasitic, dissipation and storage parameters are abstracted away to simplify the system model causing discontinuous changes in system behavior. Time scale abstractions collapse the end effect of phenomena associated with very fast time constants to a point in time. Parameter abstractions remove small and large parameter values (parasitic dissipative and storage elements) from the model [14, 16].

Time Scale Abstraction. Consider filling a cylinder with oil by pulling a piston (Fig. 2). Fig. 2(a) shows a loose object in the oil moving towards the connection between the cylinder and the pipe that provides the oil supply. If the object is rigid, it obstructs oil flow when it reaches the orifice (Fig. 2(b)). Therefore, the oil flow rate becomes 0, but at this point if the external force is insufficient to keep the object in place, it drops down, and the orifice opens partially. As a result flow resumes, and the force exerted by the oil on the object causes it to topple over and move through the orifice. If all these phenomena occur at time constants much smaller than the normal oil flow rate, the oil flow rate seems to change at a very fast rate from a nonzero value to a zero value, before oil flow resumes and builds up a new velocity, v_{new}. Therefore, the new configuration, α_3, is reached immediately after the state vector is updated.

The hybrid model imposes an algebraic constraint on the oil flow velocity in the *obstruct* mode, α_2, and the switching specifications are such that this mode is departed immediately after the state vector is updated, $x = x^+$. The intermediate mode α_2 is called a *pinnacle.*

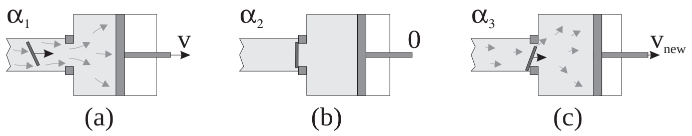

Fig. 2. An object may block an orifice.

Parameter Abstraction. If the object is flexible, it bends by the force of the pulling piston when it blocks the orifice. Almost immediately the object pops through the connection (Fig. 3(c)). A hybrid modeling approach may be adopted to avoid specifying this detailed bending behavior. When reasoning with this model, one has to analyze whether the force on the stuck object is sufficient to pull it through the orifice. This is done by computing the force generated by the instantaneous change of oil velocity from a finite value to 0. If the computed force is large enough, the model switches to a new mode where the orifice is not blocked, and the oil continues to flow with an initial value that equals the value of oil flow velocity just before the object got stuck in the orifice.

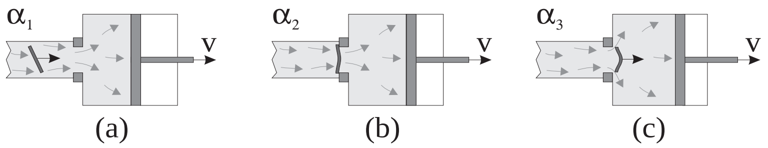

Fig. 3. An object may be pulled through an orifice.

Formally, the consecutive mode switch from α_2 to α_3 has to occur before the state vector is updated to its *a posteriori* value, $x = x^+$. otherwise the velocity would be 0. The intermediate mode, α_2, between α_1 and α_3 is a so-called *mythical* mode [16].

3 The Elevator System

Attitude control in an aircraft is achieved by the elevator control subsystem [6, 19]. This system may consist of two mechanical elevators (Fig. 1) which are positioned by electro-hydraulic actuators. When a failure occurs, redundancy management may switch actuator systems to ensure maximum control. Continuous feedback control drives the elevator to its desired set point, while higher level redundancy management selects the active actuator.

Fig. 4 shows the operation of an actuator. The continuous PID control mechanism for elevator positioning is implemented by a servo valve. The output of the servo valve controls the direction and speed of travel of the piston in the cylinder by means of a spool valve mechanism, illustrated in Fig. 5. The piston and connected elevator flap constitute the load. In the servo mechanism, the feedback signal may be provided by the fluid pressure, mechanical linkage, electrical signals, or a combination of the three.

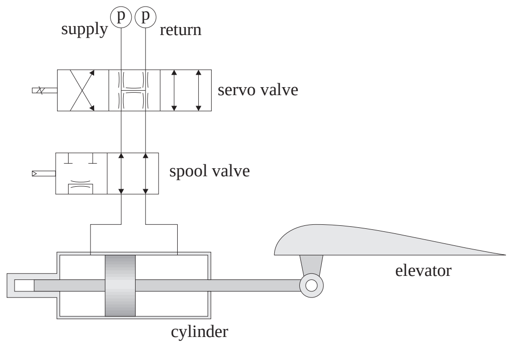

Fig. 4. Hydraulics of one actuator.

A typical spool valve (Fig. 5) consists of a piston that moves in a cylinder. A number of cylinder ports connect the supply and return part of the hydraulic system with the load. Cylindrical blocks called lands, connected to the piston, can be placed at different positions to render the servo mechanism and actuator *active* or *passive*. Fig. 5(a) and (c) show two possible oil flow configurations of the actuator. In Fig. 5(a) the control signal passes through the spool valve to the load, i.e., the actuator is *active*. In Fig. 5(c) the spool valve causes damping behavior, i.e., the actuator is *passive*. When the actuator is active, the spool valve is in its *supply* mode and the control signal generated by the servo is transferred to the cylinder that positions the elevator. The direction of the transmitted signal depends on which port is connected to the supply. When the actuator is *passive*, the spool valve is in its *loading* mode, and control signals cannot be transferred to the cylinder. However, oil flow between the chambers is possible through a loading passageway, as shown in Fig. 5(c), otherwise the cylinder would block movement of the elevator, canceling control signals from the redundant *active* actuator. When moving between *supply* and *loading*, the spool valve passes through the *closed* configuration where oil flow is blocked, as shown in Fig. 5(b).

Consider a scenario where a sudden pressure drop is detected in the left elevator actuator. Redundancy control moves the spool valve of this actuator from *supply* to *loading* and the spool valve of the other actuator from *loading* to *supply*. This causes transients that are studied in greater detail below.

4 Modeling the Elevator System

We employ *parameter* and *time scale* abstractions to design a simpler but adequate model of the elevator subsystem for control purposes. We show how these different abstraction types relate back to physical parameters in the real system.

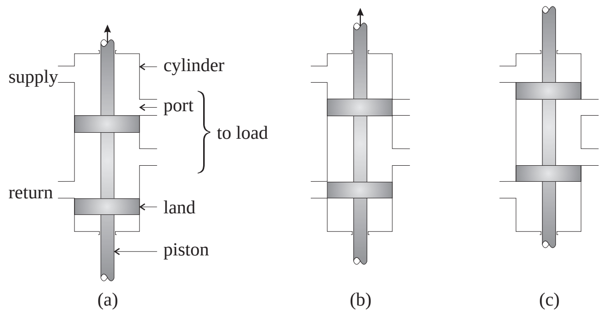

Fig. 5. A typical spool valve.

4.1 Mode Changes in the Spool Valve

To facilitate analysis of transient behavior, four modes of operation are modeled for the spool valve:

α_0 *loading*: The valve operates as a load. Pressure changes generated by the servo valve are blocked. Oil flow between chambers of the elevator positioning cylinder occurs through the loading passageway (Fig. 5(c)).

α_1 *closed*: The spool valve is closed. Pressure changes generated by the servo valve are blocked. Oil flow between the chambers of the elevator positioning cylinder is not possible as shown in Fig. 5(b).

α_2 *opening*: The valve is opening. While its lands move past the ports, fluid inertia effects may become active. Depending on the physical construction of the valve, these may have significant effects on transient behavior.

α_3 *supply*: The spool valve is opened and supplies control power. Pressure changes generated by the servo valve are transferred to the cylinder that positions the elevator. Flow of oil into and out of this cylinder is possible. This corresponds to the configurations in Fig. 5(a).

The modes α_1 and α_2 are transitional modes between α_0 and α_3. Mode changes of the spool valve are controlled by the redundancy management module which monitors a number of critical system variables. In the fault scenario, a sensor reading in actuator1 generates the failure event, σ_f. In response, the redundancy management reconfigures control by generating a sequence of discrete control signals that cause a switch of actuators. The combined state α_{ij} indicates the state of actuator2, α_i, and actuator1, α_j.

1. A control event is generated that causes the piston in the spool valve of actuator1 to move from its *supply* to *loading* position at a constant rate of change. Along the trajectory, a number of physical events occur when the displacement of the spool valve cylinder with respect to its center point, Δx, reaches valve specific values:
   (a) $\Delta x > -\lambda \rightarrow \sigma_{close} \Rightarrow \alpha_1$, the overall system mode becomes α_{01} (actuator2 is loading, and actuator1 is closed).

(b) $\Delta x > \lambda \rightarrow \sigma_{open} \Rightarrow \alpha_2$, the overall system mode becomes α_{02} (actuator2 is loading, and actuator1 is opening).
(c) $\Delta x > x_{th} \rightarrow \sigma_{load} \Rightarrow \alpha_0$, the overall system mode becomes α_{00} (actuator2 is loading, and actuator1 is loading).

2. A second control event is generated that moves the piston in the spool valve of actuator2 from its *loading* to *supply* position with a constant rate of change, causing the following physical events:
(a) $\Delta x < \lambda \rightarrow \sigma_{close} \Rightarrow \alpha_1$, the overall system mode becomes α_{10}
(b) $\Delta x < -\lambda \rightarrow \sigma_{open} \Rightarrow \alpha_2$, the overall system mode becomes α_{20}
(c) $\Delta x < -x_{th} \rightarrow \sigma_{supply} \Rightarrow \alpha_3$, the overall system mode becomes α_{30}.

Actuator1 is completely deactivated by switching to its *loading* mode α_0 before actuator2 is activated. The values of λ and x_{th} are based on physical parameters of the valve, e.g., the shape of ports and lands [6,11]. An *overlapped* or *closed center* valve has a small nonzero λ value, whereas a a *zero lap* or *critical center* type valve has $\lambda = 0$ [6].

4.2 Model Assumptions

When an actuator moves to its *closed* mode, oil flow into and out of the positioning cylinder is blocked. This implies that the cylinder piston that controls elevator position cannot move, and the elevator stops moving as well. In more detail, internal dissipation and small elasticity parameters of the oil cause the elevator velocity to change continuously during the transition. The continuous transient behavior between *supply* and *closed* is shown in Fig. 6. How quickly the system reaches 0 velocity in the *closed* mode depends on the elasticity and internal dissipation parameters of the oil.

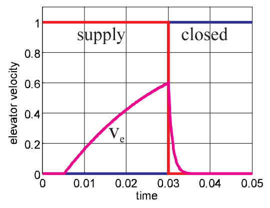

Fig. 6. Continuous transients: *closed* mode.

After a short time in the *closed* mode, the actuator moves to the *opening* mode, and inertial effects become active. Fig. 7 illustrates the continuous transients in this transition. The fluid inertia parameter of the clearance determines the final elevator velocity, v_e. In the *opening* mode, the inertial effect decreases as the clearance between port and land increases eventually becoming negligible, and the actuator operates as a simple load (*loading* mode). This is shown in Fig. 8 for two different values of the fluid inertia parameter. This also shows that the redundant actuator takes over elevator positioning control (mode α_{30}).

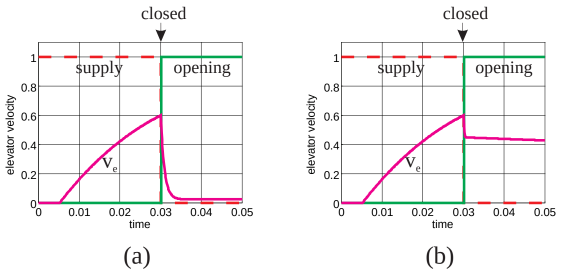

Fig. 7. Continuous transients: *opening* mode. Fluid inertia (a) $I = 1$ and (b) $I = 100$.

The continuous transients described above are not of much interest to the modeler for analysis and control (see Fig. 8 where the transients in the opening mode are still clearly visible but the continuous transients in the closed mode are not). Model simplification results in removal of small elasticity and inertia effects, but Fig. 8 illustrates that depending on their magnitude, they may have a distinct impact on the overall system behavior.

4.3 Applying Abstractions to Achieve Model Simplification

We apply model simplification by abstraction to analyze the elevator system.

Time Scale Abstraction. In the *opening* mode, fluid inertia and dissipative effects in the clearance between land and port cause a second order build-up of fluid flow. Though the fluid flow velocity and its time derivative are 0 initially, the velocity of the elevator and the driving piston are not. This results in a pressure build-up in the cylinder governed by the elasticity coefficient of the oil which causes a rapid increase of fluid flow through the land/port clearance. The pressure also causes the elevator velocity to decrease rapidly, resulting in the transient in Fig. 7(a). The initial transient from moving into the *closed* mode is replaced by the transient moving into the *opening* mode. The difference is best seen by comparing Fig. 6 with Fig. 7(a). The final value of the velocity after this transient depends on the dissipative effects and starting point and duration of the *opening* mode.

If the elastic and inertial effects are abstracted away, the *closed* and *opening* modes are traversed instantaneously in sequence into the *loading* mode. However, the inertia element has a distinct effect on system behavior, and the influence occurs over a small time interval. This is an example of time scale abstraction, where mode change phenomena are expressed at *a point* in time. An important implication is that the state vector has to be modified through the sequence of mode changes. An algebraic relation is derived to compute the elevator velocity to correspond to the fast transient behavior in the mode transitions (Fig. 6 and Fig. 7(a)).

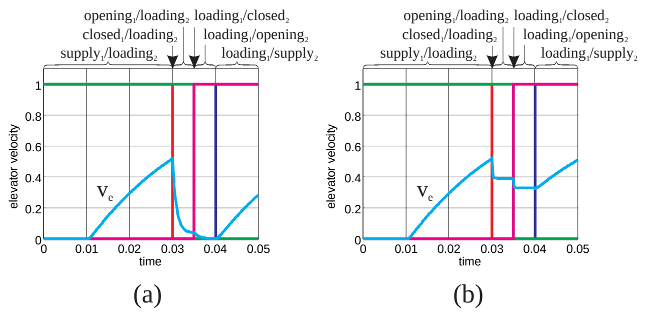

Fig. 8. Continuous transients: *loading* mode. Fluid inertia (a) $I = 1$ and (b) $I = 100$.

Parameter Abstraction. When dissipation in the land/port clearance dominates the inertial effect, a much faster response in fluid flow velocity occurs because dissipation does not introduce a time derivative effect. The flow of oil into and out of the cylinder is fast, and the pressure build-up in the cylinder is small. As a result, elevator velocity remains almost unchanged as the model switches from *closed* to *opening* (Fig. 7(b)). Small parameter values are abstracted away, and the transitions through the *closed* and *opening* modes are instantaneous (no time derivative effects are present). For small parameter values (Fig. 6 and Fig. 7(b)), the transients to *opening* (Fig. 7) may result in very different behavior from transients into *closed* (Fig. 6). When a discontinuous jump occurs, the eventual elevator velocity is not computed by first executing the jump to *closed* and then to *opening*, but immediately to *opening*. Otherwise, *closed* would have set the velocity to 0, which would also be the value in the *opening* mode. For parameter abstractions these intermediate steps are completely abstracted away.

5 A Hybrid Observer

Modern controllers and fault isolation systems are based on accurate estimation of the state vector of the system under consideration. Typically, a limited number of physical measurements are made on the system, and functional redundancy methods are utilized to derive values of other system variables [8]. These approaches rely on accurate numerical models that can be used to reconstruct the internal system state from the observed variables by means of an observer [10]. Modeling uncertainties often require that temporal sequences of a number of variable values be available to make accurate estimates of the system state.

For linear continuous processes this form of system identification is well understood, and some techniques exist for nonlinear continuous systems [5]. However, complex embedded systems, such as the elevator control subsystem, operate in multiple modes, and the continuity constraint is often violated in the *models* that define the transitions between the modes of operation. As a result, mode transitions are often accompanied by discontinuities in the system state vector. Conventional observer schemes cannot be applied in these situations. Because a mode change may cause a large deviation in the estimated state vector (e.g., Fig. 7), convergence of the observer may require much more time. Incorporating the semantics of time scale and parameter abstractions into the observer scheme may mitigate a number of problems introduced by the large state changes when mode changes occur.

5.1 The Observer for Continuous Behavior

We investigate this conjecture by building a hybrid observer for the elevator velocity of the elevator control system using a standard observer scheme depicted in Fig. 9. The input to the elevator actuators includes the PID control pressure for actuator1 and actuator2:

$$u = [p_{in,1}\ p_{in,2}]. \tag{1}$$

The measured variables associated with the actuator hydraulics are:

$$y = [q_{oil,1}\ q_{oil,2}\ p_{oil,1}\ p_{oil,2}], \tag{2}$$

where q_{oil} represents the oil flow rate into the servo valve, and p_{oil} represents the internal pressure in the positioning cylinder. The state vector of the system is

$$x = [p_{oil,1}\ p_{oil,2}\ v_e], \tag{3}$$

where v_e is the elevator velocity. We focus on reconstructing v_e from the measured variables in the observer system.

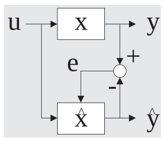

Fig. 9. A general observer setup.

A traditional Luenberger observer scheme for computing the estimated velocity, $\hat{v}_e$, during normal operation uses an observer model that is identical to the process model. We present a simplified observer structure, without accounting for scaling factors, derived for actuator1 in its *active* mode and actuator2 in its *passive* mode. This corresponds to the overall system mode α_{03}.

The estimate of v_e is computed from the estimated oil flow into the active hydraulics system, which is $\hat{q}_{in,1}$ when actuator1 is active. The value of $\hat{q}_{in,1}$ is a function of the pressure difference across the servo and spool valve, $p_{in,1} - \hat{p}_{oil,1}$, and the fluid resistance in this path, R_{hy} (equal in value for both actuator1 and actuator2):

$$\hat{q}_{in,1} = \frac{p_{in,1} - \hat{p}_{oil,1}}{R_{hy}}. \tag{4}$$

The pressure $p_{in,1}$ is a known control signal. Using $\hat{q}_{in,1}$ the oil flow into the positioning cylinder capacity, C_{hy}, can be calculated by subtracting the amount of oil required to move the piston in the cylinder at the estimated velocity v_e. Consequently, the estimated pressure change in the cylinder is given by

$$\dot{\hat{p}}_{oil,1} = \frac{1}{C_{hy}}(\hat{q}_{in,1} - \hat{v}_e + K_p(p_{oil,1} - \hat{p}_{oil,1})), \tag{5}$$

where K_p is the gain factor used to determine observer convergence.

The change of pressure in the inactive cylinder is computed similarly, the difference being that there is no external oil flow into the cylinder. Instead, there is an oil flow through its loading passageway. This yields

$$\dot{\hat{p}}_{oil,2} = \frac{1}{C_{hy}}(-\frac{\hat{p}_{oil,2}}{R_{leak}} - \hat{v}_e + K_p(p_{oil,2} - \hat{p}_{oil,2})), \tag{6}$$

where K_p is the gain factor for observer convergence (same as in Eq. (5)).

Finally, the change of elevator velocity is a function of the oil pressures generated by both cylinders, $\hat{p}_{oil,1}$ and $\hat{p}_{oil,2}$, and a convergence term based on the difference in actual oil flow into the system and the estimated oil flow, i.e.,

$$\dot{\hat{v}}_e = \frac{1}{m_e}(\hat{p}_{oil,1} + \hat{p}_{oil,2} + K_v(q_{in,1} - \hat{q}_{in,1})). \tag{7}$$

Convergence is a function of the gain factor, K_v. The dynamics for mode α_{30} can be computed similarly. Since the system is well behaved and linear, and given that the input values and output measurements listed in Eq. (1) and (2) are known for the above observer, $\hat{v}_e$ easily converges to v_e.

5.2 The Hybrid Observer Scheme

As a next step, mode changes as a result of system failure are included. The resulting elevator control system model now includes a number of modes α_{ij} described earlier, where fast continuous transients occur. These behaviors contain complex nonlinearities and are hard to model in detail. This makes it difficult to build a robust and efficient observer.

Instead of modeling complex nonlinearities directly, we apply our model abstraction schemes to transform the behavior into a set of piecewise simpler (possibly linear) behaviors with a set of discontinuous mode transitions incorporated into the observer model. Fig. 10 shows the discrete event switching structure of the hybrid observer. Because the complex continuous transitions between modes are abstracted into discontinuous changes, the complex ODEs can be replaced by piecewise simpler ODE models (sODE) [15]. The states are indexed as sODE$_{ijk}$, where i determines whether the observer feedback is active ($i = 1$) or not ($i = 0$), j corresponds to α_j of actuator2, and k corresponds to α_k of actuator1. When a failure occurs and the redundancy management controller starts to switch states of actuator1, it generates event $\sigma_{close,1}$. This event changes the observer mode to the one representing both actuator1 and actuator2 in the *loading* mode. In this setup, there can be significant discrepancies between the actuator system model and the observer model. Since the fast continuous transients are not modeled in the observer, the error feedback is deactivated when these transitions occur.

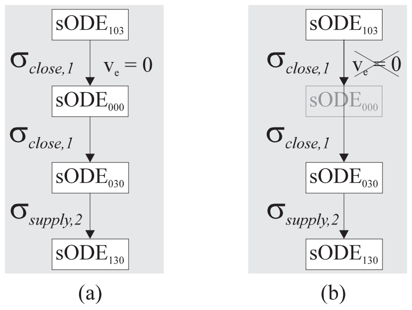

Fig. 10. Hybrid automata specifying the observer discrete event structure.

Since the detailed continuous transients are abstracted into discontinuous changes in the observer model, an immediate further change occurs, with the spool valve of actuator2 going into its *supply* mode. To achieve this consecutive change, the same event $\sigma_{close,1}$ is modeled to trigger the transition. Because the fast continuous transients are still active, the observer feedback is still disabled. Given the valve specifications, the fast continuous transients settle when $\Delta x < -x_{th}$, and this generates the physical event $\sigma_{supply,2}$ that causes the observer to restart the estimation process.[1] Note that the transition through sODE_{000} is crucial for time scale abstraction because it brings about the corresponding discontinuous jump in v_e (see Fig. 10(a)). However, removing states based on this global knowledge destroys the compositional characteristics of the hybrid automata [15].

The resulting observer model contains the two continuous modes α_{03} and α_{30} described above, where it may or may not have its error feedback activated. The complete observer configuration is illustrated in Fig. 11.

After mode changes are completed, the continuous observer is reactivated, and this includes setting the estimated value of p_{oil} to its measured value. However, this information is not available for $\hat{v}_e$ and its initial value has to be computed using the applicable semantics derived from the mode transition definitions. To ensure a short convergence time after the continuous observer is reactivated, it is important that $\hat{v}_e$ estimated from discontinuous transition semantics be close to the actual value of v_e after the fast continuous transients.

[1] In general, estimating mode transitions, and the actual mode of the system in itself can be a very complex task.

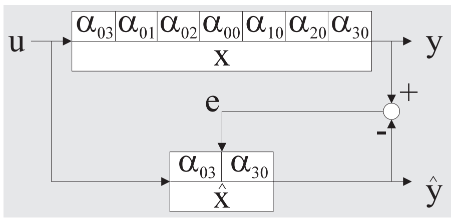
u
α_{03} α_{01} α_{02} α_{00} α_{10} α_{20} α_{30}
x
y
e
+
-
α_{03} α_{30}
$\hat{x}$
$\hat{y}$

Fig. 11. A hybrid observer configuration.

5.3 Applying Abstraction Semantics

Depending on the type of spool valve used to execute the mode change, the elevator velocity may quickly be forced to zero, or it may change little from its current value (see Fig. 7). Fig. 12 shows by simulation the estimated v_e for two types of spool valves. The grayed bars indicate the time interval during which the observer feedback mechanism is disabled because discrete mode changes occur. In Fig. 12(a), time scale abstraction explains the fast continuous opening/closing behavior of the valve (note the discontinuous change in $\hat{v}_e$ at 0.01). Because of the time derivative behavior, this response takes more time than a parameter abstraction which has semantics that explain the behavior shown in Fig. 12(b). Both results show quick convergence of $\hat{v}_e$ after mode changes have occurred. A small initial difference between v_e and $\hat{v}_e$ demonstrates the convergence process of the observer during continuous behavior.

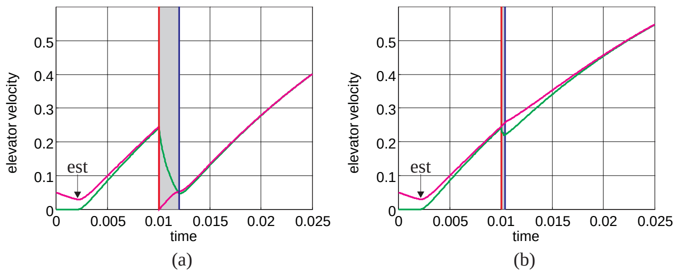

Fig. 12. Elevator velocity estimate for (a) time scale and (b) parameter abstraction.

To make the need for applying the correct modeling abstractions more explicit, we implemented an observer where time scale and parameter abstraction semantics were intentionally switched. Fig. 13 shows the results in terms of estimate convergence. In both cases the initial estimate of $\hat{v}_e$ after mode changes deviates much more from the actual value, and, therefore, convergence takes much longer in the continuous mode. This may result in inefficient or even erroneous behavior by the system.

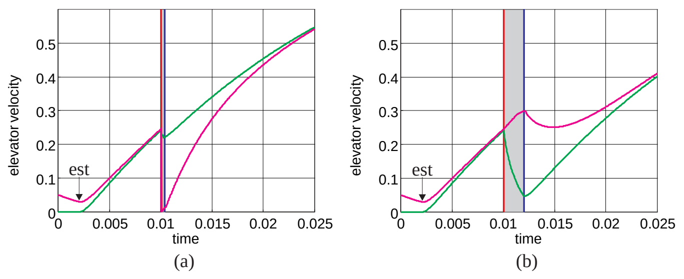

Fig. 13. Elevator velocity estimate for incorrect (a) time scale and (b) parameter abstraction.

6 Conclusions

The hybrid observer for the elevator system demonstrates how observer design can be made computationally simpler without sacrificing accuracy and convergence characteristics. In other work [15], we have implemented combined discrete transitions with continuous behavior evolution as hybrid automata, where discrete transitions cause changes in the system behavior model, but discontinuous changes in the state vector values may also occur. Time scale abstraction and parameter abstraction along with associated semantics that govern discontinuous changes in behavior specification are incorporated into the hybrid automata framework. An important feature of our work is that these abstractions relate back to physical parameters of the physical system that cause fast continuous transients. The hybrid automata scheme can form the basis for designing observer schemes in the manner illustrated in the last section.

Further extensions to this approach will involve extending our modeling schemes to represent complex nonlinear behaviors as piecewise simpler behaviors with discrete transitions between modes, e.g., the transition from *closing* to *supply* may involve regions of turbulent and laminar flow. The system model can then be decomposed into piecewise components each corresponding to a different regime of operation, and controllers synthesized for the resulting hybrid models. There has been work in this area by [2, 3], but it would be very interesting to combine such approaches with modeling abstractions discussed in this paper, and apply them to controller design tasks. In other work we have started looking at the use of singular perturbation techniques [9] to automate the generation of simpler models and the mode switching conditions with the correct abstraction semantics.

References

1. R. Alur, C. Courcoubetis, T.A. Henzinger, and P.H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. *Lecture Notes in Computer Science*, vol. 736, pp. 209–229. Springer-Verlag, 1993.

2. A. Balluchi, M.D̃i Benedetto, C. Pinello, C. Rossi, and A. Sangiovanni-Vincentelli. Hybrid control for automotive engine management: The cut-off case. *Lecture Notes in Computer Science, Hybrid Systems: Computation and Control*, pp. 13–32, Springer-Verlag, Berlin, 1998.
3. A. Beydoun, L.Y. Wang, J. Sun, and S. Sivasankar. Hybrid control for automotive powertrain systems: A case study. *Lecture Notes in Computer Science, Hybrid Systems: Computation and Control*, pp. 33–48, Springer-Verlag, Berlin, 1998.
4. K. Brammer and G. Siffling. *Kalman-Bucy Filters.* Artech House, Norwood, MA, 1989.
5. R.S. Bucy and J.M.F. Moura. *Nonlinear Stochastic Problems.* Reidel, Dordrecht, 1983.
6. W.L. Green. *Aircraft Hydraulic Systems.* John Wiley & Sons, Chichester, UK, 1985.
7. J. Guckenheimer and S. Johnson. Planar hybrid systems. In P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, eds., *Hybrid Systems II*, vol. 999, pp. 202–225. Lecture Notes in Computer Science, Springer-Verlag, 1995.
8. R. Isermann. A review on detection and diagnosis illustrate that process faults can be detected when based on the estimation of unmeasurable process parameters and state variables. *Automatica: IFAC Journal*, 20(4):387–404, 1989.
9. P. Kokotovic, H.K. Khalil, and J. O'Reilly. *Singular Perturbation Methods in Control.* Academic Press, London, UK, 1986.
10. D.G. Luenberger. *Introduction to Dynamic Systems: Theory, Models, & Applications.* John Wiley, New York, 1979.
11. H.E. Merritt. *Hydraulic Control Systems.* John Wiley, New York, 1967.
12. P.J. Mosterman. *Hybrid Dynamic Systems: A hybrid bond graph modeling paradigm and its application in diagnosis.* PhD dissertation, Vanderbilt University, 1997.
13. P.J. Mosterman and G. Biswas. Formal Specifications for Hybrid Dynamical Systems. *IJCAI-97*, pp. 568–573, Nagoya, Japan, Aug. 1997.
14. P.J. Mosterman and G. Biswas. Principles for Modeling, Verification, and Simulation of Hybrid Dynamic Systems. *5th Intl. Conf. on Hybrid Systems*, pp. 21–27, Notre Dame, IN, Sep. 1997.
15. P.J. Mosterman and G. Biswas. Hybrid Automata for Modeling Discrete Transitions in Complex Dynamic Systems. *IFAC Intl. Symp. on AI in Real-Time Control*, Grand Canyon National Park, AZ, Oct. 1998.
16. P.J. Mosterman and G. Biswas. A theory of discontinuities in dynamic physical systems. *J. of the Franklin Institute*, 335B(3):401–439, Jan. 1998.
17. P.J. Mosterman, G. Biswas, and J. Sztipanovits. A hybrid modeling and verification paradigm for embedded control systems. *Control Engg. Practice*, 6:511–521, 1998.
18. H. Schneider and P.M. Frank. Observer-based supervision and fault detection in robots using nonlinear and fuzzy logic residual evaluation. *IEEE Trans. on Control Systems Technology*, 4(3):274–282, May 1996.
19. J. Seebeck. *Modellierung der Redundanzverwaltung von Flugzeugen am Beispiel des ATD durch Petrinetze und Umsetzung der Schaltlogik in C-Code zur Simulationssteuerung.* Diplomarbeit, TU Hamburg-Harburg, 1998.