# Deriving Discontinuous State Changes for Reduced Order Systems and the Effect on Compositionality

**Pieter J. Mosterman***
Institute of Robotics and System Dynamics
DLR Oberpfaffenhofen
D-82230 Wessling
Pieter.J.Mosterman@dlr.de

**Gautam Biswas**†
Knowledge Systems Lab
Stanford University
Stanford, CA 94305
biswas@ksl.stanford.edu

## Abstract

Dynamic behavior of complex physical systems is often nonlinear and includes multiple temporal scales. For efficient model analysis, *singular perturbation* methods can be employed to decouple and analyze the fast and slow behavior in two steps: (i) by assuming the fast behavior quickly reaches a quasi steady state, and (ii) by analyzing the slow behavior of the system. The decoupling achieved by applying the quasi steady state solution reduces the complex system of ordinary differential equations (ODEs) to simpler ODEs. This process of abstracting fast continuous behavior into algebraic constraints may cause discontinuous jumps in variable values when configuration changes occur, requiring the system variables to be reinitialized correctly. The application of traditional singular perturbation approach correspond to discontinuous changes resulting from *parameter* abstraction. This paper extends this notion to analysis of discontinuous changes caused by *time scale* abstraction. Deriving the explicit discontinuous jumps caused requires analysis of the interactions between model components, therefore, they are configuration dependent. Therefore, reduced order model components (or fragments) may not be valid in other configurations, and, therefore, may not be directly usable in a compositional modeling framework.

## 1 Introduction

The pressure to achieve optimal performance and meet rigorous safety standards in industrial processes, aircraft, and nuclear plants, is necessitating more detailed modeling and analysis of these systems. In embedded systems, the inherently continuous physical process interacts with digital control signals that have very fast time constants. In general, complex systems exhibit nonlinearities attributed to small parameters that manifest as behaviors on very fast time scales. These fast transients make it hard to simulate and analyze system behavior. Sophisticated numerical simulation algorithms that vary their time step to accommodate multi time scale behaviors have been developed, but the variable step size makes it hard to bound their runtime computational complexity. This makes them unsuitable for real-time analysis. As an alternative, modeling methodologies have been developed recently that combine continuous and discrete, i.e., *hybrid* models into an integrated framework. The resultant systems have piecewise continuous modes of behavior evolution with discrete transitions between the modes. Our previous work includes hybrid modeling and analysis of both embedded systems and abstractions of complex nonlinear behavior in physical systems [11; 13; 15].

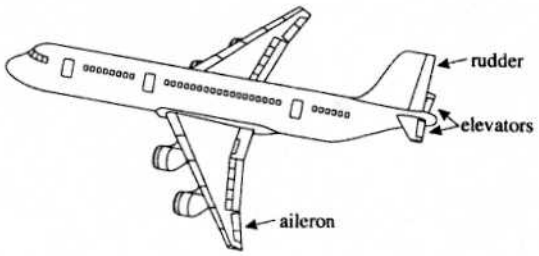


Figure 1: **Primary aerodynamic control surfaces.**

As an example, consider the primary aerodynamic control surfaces of the airplane in Fig. 1 [19]. Modern avionics systems employ electronic *fly-by-wire* control, where electronic signals generated by a digital processor are transformed into the power domain by electro-hydraulic actuators. The primary flight control system demonstrates the paradigm for hybrid modeling of embedded control systems. At the lowest level in the control hierarchy, continuous PID control moves the rudder, elevators, and ailerons to set positions. Desired set-

* Pieter J. Mosterman is supported by a grant from the DFG Schwerpunktprogramm KONDISK.

† Gautam Biswas is supported by grants from Hewlett-Packard, Co. at Vanderbilt Univ. and NIST grant number 70NANB6H0075-05 at Stanford Univ. He is on leave from Vanderbilt Univ., Nashville, TN for AY 1998-99.

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*, *go-around*). Detection of failures may lead to discrete changes in system configuration. Model simplifications created by discretizing fast, nonlinear transients produce discontinuous variable changes.

To accommodate these scenarios, hybrid dynamic modeling paradigms [1; 5; 9; 15] abstract the detailed continuous behavior represented as a system of complex ordinary differential equations, cODE, into piecewise simpler sODEs. In the singular perturbation approach [7], the sODEs are derived by decoupling fast and slow behavior in the cODE and assuming the fast behavior has reached its steady state. In the qualitative reasoning domain [20], QSIM [8] uses the fast and slow decompositions to create a hierarchy of constraint networks to simulate complex physical system behavior that occurs at different temporal and spatial scales across multiple time scales. Iwasaki and Bhandari [6] have used relative magnitudes of coefficients in an influence matrix (i.e., the $A$ matrix) of a linear system to determine "nearly decomposable" substructures. Ignoring the weak interactions (i.e., the small parameters) between the substructures results in simpler aggregated systems that ignore insignificant small time constant dynamic effects on overall system behavior.

Our goal is to extend and generalize these approaches to linear and nonlinear systems. We have shown that small time constant effects cannot always be ignored in analyzing dynamic system behavior. Abstracting fast transients may lead to jumps in the system state vector variable values when configuration changes occur. To address this, we have developed systematic modeling methodologies where the task at hand is employed to derive abstractions that simplify the system model and abstract fast behaviors to occur at a point in time [11; 13; 15]. The resultant system model exhibits multiple modes of operation [18], each with simpler *piecewise* continuous behavior, but transitions between the modes may introduce discontinuous changes in the system variables.

In this paper we demonstrate the effects of abstracting fast continuous transients exhibited by complex systems, into discontinuous changes of the continuous state vector and its effect on compositionality of models. In previous work [11; 15], we have established the different semantics involved with the discontinuous state vector changes corresponding to two kinds of behavioral abstraction: *parameter* abstraction and *time scale* abstraction. This paper demonstrates a systematic methodology for generating the simpler ODE models from the more complex ODE models of system behavior. The simpler piecewise ODE models are then compiled into *hybrid automata* to facilitate efficient run time analysis of hybrid behavior.

Hybrid automata [1] extend traditional finite state automata with a continuous dimension. Each discrete state (i.e., mode) has an associated ODE that describes continuous behavior evolution of the system in time. Changes in values of continuous variables may result in discrete events that cause state (mode) changes. Mode changes may also cause abrupt changes in the continuous state vector, and these are explicitly specified in the state transitions of the hybrid automata.

## 2 Hybrid Dynamic Systems

Hybrid dynamic systems combine discrete state changes with continuous behavior evolution [1; 5; 9; 15]. Building hybrid dynamic models of physical systems requires the specification of three component parts [9; 11; 15].

**The Continuous Part**

Differential equations form a common representation of continuous system behavior. The system is described by a *state vector*, $x$, and other variables called *signals*, $s$, are derived algebraically, $s = h(x)$. Behavior over time is specified by a field $f$. Interaction with the environment is specified by *input* and *output* signals, $u$ and $y$. The dynamics of system behavior is expressed as a set of ODEs, $\dot{x} = f(x, u)$.

**The Discrete Part**

Discrete systems, modeled by a state machine representation, consist of a set of discrete modes, $\alpha$. Mode changes caused by events, $\sigma$, are specified by the *state transition function* $\phi$, i.e., $\alpha_{i+1} = \phi(\alpha_i)$. A transition may produce additional discrete events, causing further transitions.

**Interaction**

In hybrid dynamic systems, a mode change from $\alpha_i$ to $\alpha_{i+1}$, may result in a field definition change from $f_{\alpha_i}$ to $f_{\alpha_{i+1}}$, and a discontinuous change in the state vector governed by an algebraic function $g$, $x^+ = g_{\alpha_i}^{\alpha_{i+1}}(x)$. Discrete mode changes are caused by an *event generation function* $\gamma$ associated with the current active mode, $\alpha_i$, $\gamma_{\alpha_i}(x) \leq 0 \rightarrow \sigma_j$.

## 3 Abstracting Fast Transients

Continuous behavior in physical systems can occur on a hierarchy of temporal and spatial scales. To simplify system models, parasitic dissipation and storage effects are abstracted away but they may cause discontinuous changes in system behavior. *Parameter* abstractions remove the corresponding small and large parameter values from the model. This has no immediate effect on the system state vector, but may cause configurational changes in the model that implicitly cause discontinuous state changes. *Time scale* abstractions collapse the end effect of phenomena associated with very fast time constants to a point in time causing discontinuous changes in state vector values. In previous work [11; 12; 13], we have developed formal semantics for mode transitions. For parameter abstractions, mode switching is governed by the *a posteriori* state vector value, whereas for time scale abstractions they are governed by *a priori* state vector values. In this section, we formalize the derivation of simplified models generated by parameter and time scale abstractions.

### 3.1 Parameter Abstraction

Parameter abstractions eliminate small[1] parameters in a system model to achieve a reduced model that is simpler to analyze. This is the basis of the *singular perturbation method* [7]. A singular perturbation representation formulates the system behavior model into a complex system of ordinary differential equations (cODE) with two time scales:

$$\begin{cases} \dot{x} = f(x, z, \epsilon, t), \ x(t_0) = x^0, \ x \in \Re^n, \\ \epsilon \dot{z} = g(x, z, \epsilon, t), \ z(t_0) = z^0, \ x \in \Re^m, \end{cases} \quad (1)$$

where $\epsilon$ embodies small and large parameter values that cause fast transients. The function $f$ models the slower dominant system behavior. Setting $\epsilon$ to 0 reduces the second equation to an algebraic form. Assuming that $g(x, z, 0, t) = 0$ has distinct real roots, the fast behaviors corresponding to $z$ can be solved for algebraically, and substituted in $f$. This results in a reduced-order *quasi steady-state* model that embodies an sODE,

$$\begin{cases} \dot{\bar{x}} = f(\bar{x}, \bar{\phi}(\bar{x}, t), 0, t), \ \bar{x}(t_0) = x^0, \\ \bar{z} = \bar{\phi}(\bar{x}, t). \end{cases} \quad (2)$$

We apply this approach to the collision between two bodies shown in Fig 2. A first order approximation of the collision process includes two parameters: (i) $C$, that models the elastic interaction between the bodies, and (ii) $R$, that models the dissipative effects. If the momentum of the bodies, $p_i$, and the displacement, $q$, of the spring which models the elasticity parameter $C$, are chosen as state variables, the dynamic behavior of the system is described by the following ODE:

$$\begin{cases} \dot{p}_1 = -\frac{q}{C} - R(\frac{p_1}{m_1} - \frac{p_2}{m_2}) \\ \dot{p}_2 = \frac{q}{C} + R(\frac{p_1}{m_1} - \frac{p_2}{m_2}) \\ \dot{q} = \frac{p_1}{m_1} - \frac{p_2}{m_2}. \end{cases} \quad (3)$$

This singular system of equations can be reduced to a second order system by applying the transformation

$$v = \frac{p_1}{m_1} - \frac{p_2}{m_2}, \quad (4)$$

resulting in a second order ODE:

$$\begin{bmatrix} \dot{v} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} -R(\frac{1}{m_1} + \frac{1}{m_2}) & -\frac{1}{C}(\frac{1}{m_1} + \frac{1}{m_2}) \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v \\ q \end{bmatrix}. \quad (5)$$

In many cases, the detailed continuous transients caused by the $R$ and $C$ parameters are not of interest to the modeler. If these parameters are removed from the model, a simpler system of equations would result, but the state variables may exhibit explicit discontinuous jumps. Therefore, simplification requires computation of the discontinuous jumps from the detailed continuous transients. To apply singular perturbations, we assume $C$ to be small and $R$ to be large and take $\frac{1}{R}$ to be the small $\epsilon$ parameter. Therefore,

$$\begin{bmatrix} \frac{1}{R}\dot{v} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} -(\frac{1}{m_1} + \frac{1}{m_2}) & -\frac{1}{RC}(\frac{1}{m_1} + \frac{1}{m_2}) \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v \\ q \end{bmatrix}, \quad (6)$$

[1] Also, large, because the reciprocal of a large parameter value is small.

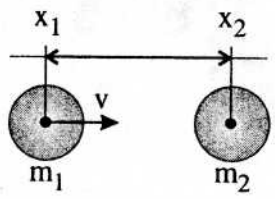


Figure 2: **Collision of a body $m_1$ with velocity $v$ and a body $m_2$.**

where $v$ contains the fast behavior. Substituting $\frac{1}{R} = 0$ results in $v = 0$. Transforming this back to the original state variables, yields $\frac{p_1}{m_1} - \frac{p_2}{m_2} = 0$, i.e., $v_1 - v_2 = 0$. This is the equivalent of a perfect non elastic collision [13].

### 3.2 Time Scale Abstraction

Instead of eliminating the fast transient due to dissipative effects, if we were to reduce the effect of elasticity to occur at a point in time, we get a time scale abstraction. Consider the system of colliding bodies again (Fig. 2) with detailed behavior given by Eq. (5). If $C$ is taken to be the small $\epsilon$ parameter, this gives

$$\begin{bmatrix} C\dot{v} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} -RC(\frac{1}{m_1} + \frac{1}{m_2}) & (\frac{1}{m_1} + \frac{1}{m_2}) \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v \\ q \end{bmatrix}, \quad (7)$$

For $\epsilon = 0$, this yields $q = 0$, and, therefore, $\dot{q} = 0$ which requires $v = 0$. When $C$ becomes small but not 0, the solution of the system in Eq. (5) has eigenvalues with imaginary components and the resultant dynamic behavior for the transient is:

$$v(t) = v(0)e^{-\frac{R}{2}(\frac{1}{m_1} + \frac{1}{m_2})t} cos((\sqrt{\frac{4}{C} - R^2(\frac{1}{m_1} + \frac{1}{m_2})^2})t). \quad (8)$$

This shows that $v = 0$ is the steady state solution. However, in case of colliding bodies this behavior transient is aborted long before steady state is attained, because the $v$ and $q$ values generated by the transient cause the two bodies to disconnect.

To analyze this in detail, consider the case of two point masses. The collision process becomes active when $x_1 \geq x_2$, where $x_1$ and $x_2$ are the positions of body $m_1$ and $m_2$, respectively. The bodies disconnect when the force between them becomes negative, i.e., $F_{12} < 0$. At this point, the state variable values (i.e., the two body velocities) constitute the final, *a posteriori*, values around the discontinuous jump corresponding to the collision. Since $F_{12} = \frac{q}{C} < 0$ at the disconnect point, this implies $q < 0$ since $C > 0$. The time point at which the disconnect occurs is computed to be

$$t_d = \frac{\pi}{\sqrt{\frac{4}{C} - R^2(\frac{1}{m_1} + \frac{1}{m_2})}}. \quad (9)$$

At $t_d$, $v$ has changed from $v(0)$ to $v(t_d) = \lambda v(0)$ with $(cos(\pi) = -1)$, therefore,

$$\lambda = -e^{-\frac{R}{2}(\frac{1}{m_1} + \frac{1}{m_2})t_d} \quad (10)$$

As the $C$ parameter becomes very small, $t_d$ does too, and in the limit, $v(t_d) \rightarrow v(0)^+$. The discontinuous change in $v$ can then be represented by an algebraic equation

$$v(0)^+ = \lambda v(0) \tag{11}$$

Transforming this back to the original state variables, yields

$$\frac{p_1}{m_1}^+ - \frac{p_2}{m_2}^+ = \lambda(\frac{p_1}{m_1} - \frac{p_2}{m_2}). \tag{12}$$

Written in terms of the body velocities,

$$v_1^+ - v_2^+ = \lambda(v_1 - v_2). \tag{13}$$

This form is the well known Newton's collision rule [2], where $\lambda$ is called the coefficient of restitution that describes the amount of kinetic energy loss in the collision. If $R = 0$ in Eq. (10), $\lambda = -1$ and this describes a perfect elastic collision with no loss of energy. Note that $C$ cannot be taken to equal 0, as this would remove all elasticity and the corresponding ideal rigid body collision has no mechanism for storing kinetic energy as potential energy and returning it as kinetic energy. Therefore, this immediately causes $v = 0$. Consequently, behavior does not converge uniformly as $C \rightarrow 0$.

### 3.3 Summary

The previous two abstraction types demonstrate that singular perturbation methods apply well in case of parameter abstraction, where small parameters are abstracted away by setting their corresponding $\epsilon$ in Eq. (1) to 0.

When eigenvalues that have imaginary parts are abstracted away, reversible behavior of the fast variables around steady state is collapsed to a point in time. This reversible behavior often corresponds to energy restitution during fast transients, and switching conditions may abort these transients. Such energy restitution corresponds to a time scale abstraction and requires a more extensive analysis of the detailed fast behavior. If the transient for the elastic collision was not aborted when $F_{12} < 0$, then the fast behavior would show a damped oscillation (corresponding to a spring-mass-damper model) that also achieves $x = 0$, i.e., the same gross behavior as that of a nonelastic collision.

The difference between a parameter and a time scale abstraction in this case depends on the presence of imaginary parts in the eigenvalues that are abstracted away. Therefore, the criterion for applying a parameter abstraction corresponds to

$$\frac{4}{C} - R^2(\frac{1}{m_1} + \frac{1}{m_2})^2 \leq 0.$$

Otherwise, a time scale abstraction is applied.

A corresponding physical interpretation is that parameter abstractions relate to abstractions of behavior dominated by dissipative (or resistive) effects, and time scale abstraction relates to abstraction of behavior dominated by capacitive and inductive effects.

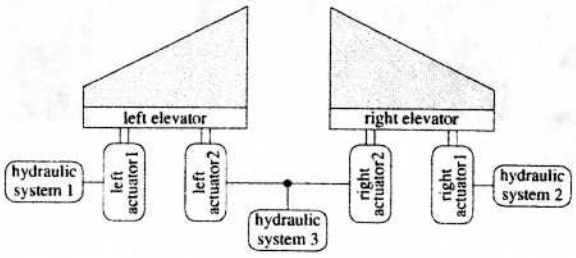


Figure 3: **Elevator system.**

## 4 The Elevator System

Aircraft are safety critical systems and their control systems incorporate several forms of redundancy. Attitude control in an aircraft is achieved by the elevator control subsystem [4; 19]. This system may consist of two mechanical elevators (Fig. 3) that 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.

Figure 4 shows the operation of one 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. When the actuator is *active* the spool valve is in its *supply* mode, and the control signal generated by the servo valve is transferred to the cylinder that positions the elevator. When the actuator is *passive*, the spool valve is in its *loading* mode that disallows control signals to be transferred to the cylinder. In this mode, flow of oil between the chambers is allowed through a loading passageway, otherwise the cylinder would block movement of the elevator, canceling control signals from the redundant *active* actuator. The piston in the positioning cylinder and connected elevator flap constitute the load. In the servo valve mechanism, the feedback signal may be provided by the fluid pressure, mechanical linkage, electrical signals, and a combination of the three.

### 4.1 The Servo Valve

The servo valve consists of a cylinder that connects its supply side with its loading side. A piston inside the cylinder can be adjusted to change the size of the orifices between supply and loading, and, therefore, controls the amount of oil flow from supply to loading. The amount of oil flowing in, $q_s$, has to equal the amount of oil flowing out $q_l$. This oil flow is determined by the pressure drop, $p_s - p_l$, across the orifice that is opened by an amount $x$,

$$\begin{cases} q_s = (p_s - p_l)x \\ q_s = q_l \end{cases} \tag{14}$$

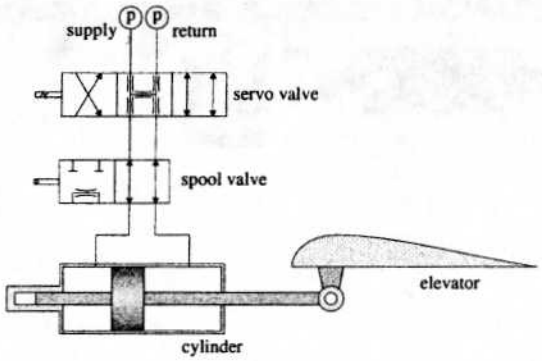


Figure 4: **Hydraulics of one actuator.**

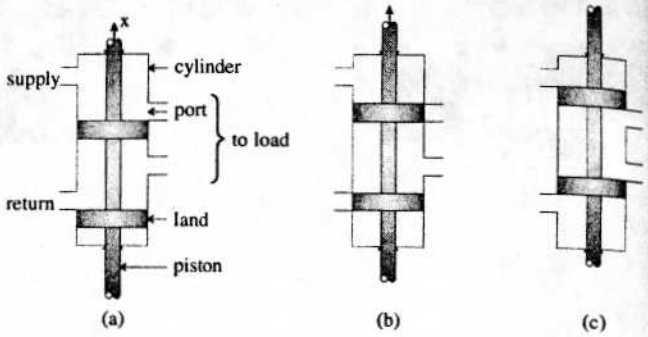


Figure 5: **A typical spool valve.**

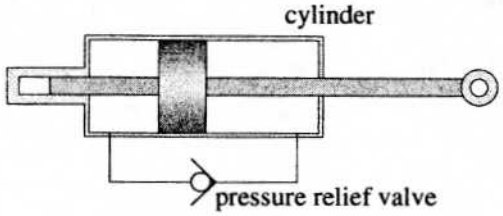


Figure 6: **A pressure relief valve may prevent high pressure.**

## 4.2 The Spool Valve

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 valve mechanism and thus the actuator *active* or *passive*. Figures 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, $\alpha_2$, and the control signal generated by the servo valve is transferred to the cylinder that positions the elevator. In this mode, the pressure on the supply side of the valve, $p_s$, equals the pressure on load side, $p_l$. Also, the oil flow from the supply, $q_s$, equals the oil flow to the load, $q_l$. When the actuator is *passive*, the spool valve is in its *loading* mode, $\alpha_0$, and control signals cannot be transferred to the cylinder. However, oil flow between the chambers is possible through a loading passageway with fluid flow resistance $R_l$, as shown in Fig. 5(c). When moving between *supply* and *loading*, the spool valve passes through the *closed* configuration, $\alpha_1$, where oil flow is blocked, as shown in Fig. 5(b). This is captured by the following equations:

$$\alpha_2 : \begin{cases} p_s = p_l \\ q_s = q_l \end{cases} \quad \alpha_1 : \begin{cases} q_l = 0 \\ q_s = 0 \end{cases} \quad \alpha_0 : \begin{cases} p_l = q_l R_l \\ q_s = 0 \end{cases} \tag{15}$$

## 4.3 The Pressure Relief Valve

In addition to the servo-spool valve configuration of Fig. 4, consider a pressure relief valve (Fig. 6) as a safety device connected to the positioning cylinder. This valve is normally closed (mode $\alpha_0$), but it may open (mode $\alpha_1$) when the pressure in the elevator positioning cylinder, i.e., the input pressure to the relief valve, $p_r$, exceeds a threshold value, $p_{th}$. This may happen because of a rapid buildup in pressure in the positioning cylinder, caused by changes in the elevator velocity, $v_e$. The pressure and flow relations in the two modes are

$$\alpha_0 : \ \{ \ q_r = 0 \qquad \alpha_1 : \ \{ \ p_r = q_r R_l \tag{16}$$

When the relief valve is open, it allows an oil flow, $q_r$, through a fluid path with resistance $R_l$.

## 4.4 Modeling the Elevator Dynamics

The dynamics of the elevator are studied in terms of the movement of the piston in the positioning cylinder, expressed as the velocity, $v_e$. The behavior can be derived by composing models of the servo valve, spool valve, relief valve, and the positioning cylinder. We express this as a second order system with two state variables: (i) $p_c$, the pressure of the oil in the cylinder, and (ii) $v_e$, the elevator velocity.

$$\begin{cases} C_c \dot{p}_c = q_{in} + q_r - q_e \\ q_e = A_p v_e \\ A_p F_e = p_c + R_c(q_{in} + q_r - q_e) \\ m_e \dot{v}_e = F_e \end{cases} \tag{17}$$

$C_c$ models the elasticity effects and $R_c$ models the dissipative effects of the oil in the positioning cylinder. The variables $q_{in}$ and $q_r$ represent the inflow of oil into the cylinder from the servo and relief valves, respectively, and $q_e$ represents the oil flow due to movement of the piston. The value of $q_e$ is a function of $A_p$, the area of the piston and $v_e$, the elevator velocity. The force exerted on the piston is a function of $p_c$, and the product of internal dissipation of the oil, $R_c$, and the overall flow rate. Newton's Second Law relates the elevator velocity to the force exerted on the piston. In state equation

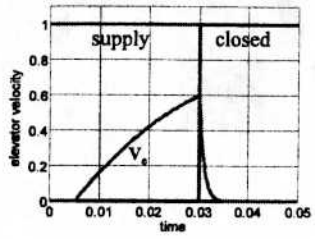


Figure 7: **Continuous transients when switching to the *closed* mode.**

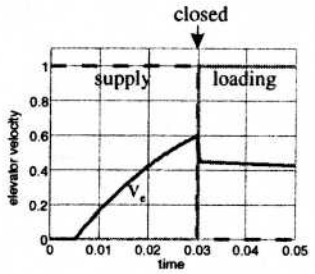


Figure 8: **Continuous transients when switching to the *loading* mode.**

form, Eq. (17) is:

$$\begin{bmatrix} \dot{p_c} \\ \dot{v_e} \end{bmatrix} = \begin{bmatrix} 0 & -\frac{A_p}{C_c} \\ \frac{1}{m_e A_p} & -\frac{R_c}{m_e} \end{bmatrix} \begin{bmatrix} p_c \\ v_e \end{bmatrix} + \begin{bmatrix} \frac{1}{C_c} & \frac{1}{C_c} \\ \frac{R_c}{m_e A_p} & \frac{R_c}{m_e A_p} \end{bmatrix} \begin{bmatrix} q_{in} \\ q_r \end{bmatrix}. \quad (18)$$

Consider a scenario where a sudden pressure drop is detected in the hydraulics supply system of an elevator actuator. Redundancy control moves the spool valve of this actuator from *supply* to *loading* and the spool valve of another actuator from *loading* to *supply* to take over the control actions. When the spool valve of 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, the 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. 7. How quickly the system reaches 0 velocity in the *closed* mode depends on the elasticity and internal dissipation parameters of the oil. Typically, soon after the *closed* mode, the spool valve starts opening and goes into the *loading* mode. The effect on elevator velocity for the detailed continuous behavior when switching from *supply* to *loading* is shown in Fig. 8.

The elasticity and dissipative effects of the oil define the transient and the final elevator velocity, before the second actuator becomes active. The details of the continuous transients are not of much interest for analysis of the control behavior. Model simplification by parameter and time scale abstractions results in removal of small elasticity and large dissipative effects. At the same time, configuration changes in the system (e.g., the spool valve moving into the *closed* mode) may cause discontinuous changes in the oil inflow into the cylinder. The resulting fast transient affects the elevator velocity, $v_e$, and these effects need to preserved across the configuration changes. A detailed analysis of the transient behavior, its simplification by parameter and time scale abstraction, and the resultant hybrid automata that describes overall system behavior is presented in [14].

We systematically derive the simpler models for the hybrid automata and the transition conditions using the methods based on singular perturbation described in Section 3 and replace the detailed continuous transients defined by Eq. (18) by an equation that captures the fast continuous change as an instantaneous discontinuous jump. We analyze the transient about the point where the spool valve closes, and the relief valve is also closed, i.e., $q_{in} = q_r = 0$. The determinant of the eigenvalue equation corresponding to this behavior is given by

$$\frac{R_c^2}{m_e^2} - \frac{4}{m_e C_c}, \quad (19)$$

indicating that there are two types of transients. The first can be attributed to the large oil dissipation parameter, $R_c$, which results in the determinant being positive with real eigenvalues. The second can be linked to the small oil elasticity coefficient, $C_c$, which results in a negative determinant and complex eigenvalues.

In case of real eigenvalues, the elevator dynamics can be computed to be

$$v_e(t) = e^{-\frac{R_c}{2m_e}t}(k_1 e^{\frac{1}{2}(\sqrt{\frac{R_c^2}{m_e^2} - \frac{4}{m_e C_c}})t} + k_2 e^{-\frac{1}{2}(\sqrt{\frac{R_c^2}{m_e^2} - \frac{4}{m_e C_c}})t}), \quad (20)$$

where $k_1$ and $k_2$ are constants that depend on $v_e(0)$ and $p_c(0)$. Like before, the restitution coefficient for the oil, affected by the spool valve closing, i.e., $\lambda_s$, can be computed by determining the value of $t_d$ at the point when the ports are opened again. If $x$ is the displacement of the piston in the spool valve, the piston may first block the ports when $x = 0$ and open them again when $x > x_{th}$, where $x_{th}$ is a parameter depending on the particular type of spool valve. The value of $t_d$ is then determined by $x_{th}$ and the speed with which the piston is moved by an external control signal. The corresponding time interval during which the oil flow into the cylinder is 0 results in an elevator velocity change as a function of $v_e(0)$ and $p_c(0)$.

In case of complex eigenvalues, the elevator dynamic behavior is governed by

$$v_e(t) = e^{-\frac{R_c}{2m_e}t}(k_1 cos(\frac{1}{2}(\sqrt{\frac{4}{m_e C_c} - \frac{{R_c}^2}{{m_e}^2}})t) \quad (21)$$

$$+k_2 sin(\frac{1}{2}(\sqrt{\frac{4}{m_e C_c} - \frac{{R_c}^2}{{m_e}^2}})t)) \quad (22)$$

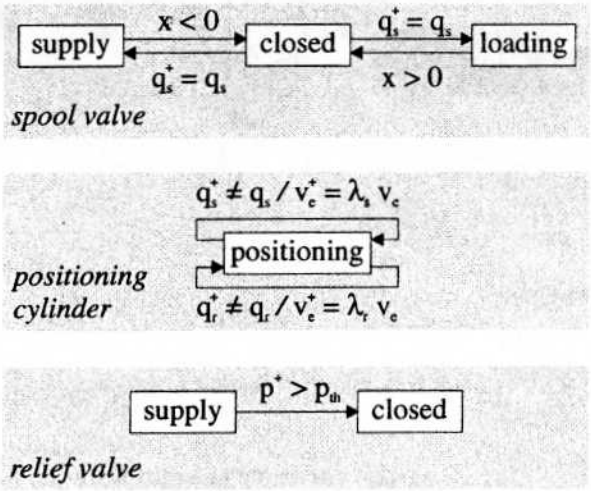


Figure 9: **Individual hybrid automata for the spool valve, positioning cylinder, and relief valve.**

where $k_1$ and $k_2$ are constants depending on $v_e(0)$ and $p_c(0)$. Again, the change of elevator velocity at $t_d$ can be computed as a function of $v_e(0)$ and $p_c(0)$. In this case, the elevator velocity may reverse much like the velocity of a bouncing ball reverses.

### 4.5 A Scenario

Fig. 9 explains the phenomena. When the spool valve goes from *supply* mode ($\alpha_2$) to *closed* mode ($\alpha_1$), causing $q_s$, and, therefore, $q_{in}$, in the positioning cylinder to change discontinuously, the fast transient that affects $v_e$ can be simplified by parameter and time scale abstraction, and $v_e$ goes through an instantaneous change in velocity given by ${v_e}^+ = \lambda_s v_e$. Because the behavior of the spool valve around $x = 0$ is abstracted away, the spool valve switches into its *closed* mode when the piston in the valve reaches 0 from the right, $x < 0$, or from the left, $x > 0$. Immediately after the discontinuous changes due to this mode are effected, $q_s^+ = q_s$, the spool valve switches out of the *closing* mode.

If the oil is assumed to be incompressible, the corresponding simplified ODE for elevator velocity in the positioning cylinder is calculated by setting $C_c = 0$:

$$\begin{cases} 0 = q_{in} + q_r - q_e \\ q_e = A_p v_e \\ A_p F_e = p_c \\ m_e \dot{v}_e = F_e \end{cases} \tag{23}$$

The number of equations and unknowns are still the same, though the sODE is first order, whereas the cODE was second order.

To compute variable values for this system, the equations of all components in their active mode are gathered and solved with respect to the unknown variables, i.e., exogenous and state variables. If the actuator is active, the servo valve equations, the spool valve equations in mode $\alpha_2$, the pressure relief valve equation in mode $\alpha_0$, and the simplified equations for the cylinder are gathered, and sorted to establish computational causality.

Now, consider the scenario with the relief valve. Note that the abrupt change in velocity from $v_e$ to ${v_e}^+$, as the spool valve goes from its supply mode, $\alpha_2$, to the loading mode, $\alpha_0$, through the intermediate closed mode, $\alpha_1$, will cause a fast pressure buildup. In the reduced order model, this buildup is governed by a discontinuous change of $v_e$, and, therefore, $v_e^+ \neq v_e$. The $m_e \dot{v}_e = F_e$ equation causes an impulse force, $F_e$, and corresponding pressure $p_e$.

In a component oriented modeling approach, this pressure impulse will always cause the relief valve to open because of its infinite magnitude, no matter how small the $v_e^+ - v_e$ difference. The more detailed model of the cylinder includes small elasticity and dissipation parameters, and they are employed to compute a more realistic value of the maximum pressure generated. This can be included in the reduced order model, by replacing the $m_e \dot{v}_e = F_e$ equation with the algebraic constraint $K_c(v_e^+ - v_e)$ providing the value for $F_e$. $K_c$ is a damping coefficient that captures the $(R_c C_c)$ effect. Using this first order approximation, the pressure buildup can be described as

$$p_c^+ = A_p K_c({v_e}^+ - v_e),$$

If the value of $p_c^+$ exceeds the critical value, $p_{th}$, this causes a further discontinuous mode change in the relief valve, which goes from closed ($\alpha_0$) to open ($\alpha_1$). In this case, the abrupt change in elevator velocity is governed by a restitution coefficient defined by the complex ODE model of the relief valve. This coefficient of restitution, $\lambda_r$, can be derived in a manner similar to the derivation for the spool valve, but the final elevator velocity, after the mode transitions, is now given by ${v_e}^+ = \lambda_r v_e$. The simplified ODE model for $v_e$ in the supply mode with relief valve *open* can also be derived similarly. Figure 9 defines the individual hybrid automata for the spool valve, the positioning cylinder, and the relief valve. In the next section, we compose the individual automata into an integrated hybrid automata for real time simulation and analysis of system behavior.

## 5 The Hybrid Automata for the Elevator System

Consider the scenario described in the previous section, where the supervisory controller switches from the current active actuator to a redundant one. We construct the hybrid automata that models the behavior of the actuator that goes from its active to passive mode by switching the spool valve from *supply* ($\alpha_2$) to *loading* ($\alpha_0$). The goal is to replace the cODEs that describe the system behavior including its transients by sODEs and a discrete event generation function, $\gamma$, and state mapping, $g$. Applying parameter and time scale abstractions results in piecewise continuous models with discrete transitions between the models. The effect of the fast transients are reduced to occur at a point in time, resulting in discontinuous changes in the elevator

velocity, $v_e$. The resultant sODEs, and the corresponding discrete transition functions, $\phi$, $\gamma$, and $g$, (Section 2) were derived systematically in the previous section.

### 5.1 Generating the Hybrid Automata

The complete hybrid automata is shown in Fig. 10. The modes are $\alpha_{ij}$, where the subscript $i$, represents the mode of the spool valve (2 - open, 1 - closed, and 0 - loading), and subscript $j$ represents the mode of the relief valve (1 - open, and 0 - closed). The corresponding sODEs are also subscripted accordingly. Initially, the actuator is in mode $\alpha_{20}$. In the simplified hybrid automata, the detailed continuous behavior around $x = 0$ is abstracted away, and the corresponding discrete events, $\{\sigma_{close}, \sigma_{spool}, \sigma_{load}, \sigma_{relief}\}$ are generated by monitoring physical variables. Figure 10 shows the relevant $g$ functions for updating the state variable value, $v_e$, along with the event generation functions, $\gamma$.

It is interesting to observe the role of the relief valve. Normally, closing the spool valve causes an instantaneous change in the oil flow rate to 0. Therefore, $q_s^+ \neq q_s$ and a rapid drop in the elevator velocity, $v_e$, occurs before the valve opens again and goes into the loading mode. The change in velocity is computed as, ${v_e}^+ = \lambda_s v_e$. However, the change in velocity causes a pressure transient, $p^+ = K_c({v_e}^+ - v_e)$, and if $p^+ > p_{th}$, $\sigma_{relief}$ is generated causing the relief valve to open, and the system goes into mode $\alpha_{11}$, with ${v_e}^+ = \lambda_r v_e$. Therefore, ${v_e}^+ = \lambda_s v_e$ is not executed and $v_e^+$ not affected by mode $\alpha_{10}$. Once the state vector is updated, ${q_s}^+ = q_s$ (i.e., the *a posteriori* and *a priori* values are the same), and $\sigma_{load}$ is generated causing the spool valve to go into *loading* (mode $\alpha_{01}$). If $\sigma_{relief}$ did not occur, ${v_e}^+ = \lambda_s v_e$ remains valid, and after the state vector is updated ${q_s}^+ = q_s$ and the mode transition to $\alpha_{00}$ occurs based on the event $\sigma_{load}$. The stroked transitions in Fig. 10 represent transitions where the $\gamma$ function is applied after the state vector has been updated.

### 5.2 Composability of Models

In the process of building the simpler ODEs and the $\gamma$ and $g$ functions from the cODE models, one has to take into account the interactions between the different component subsystems. Therefore, the traditional notion of composing the system model from individual component models [3; 10] is restricted to model components that contain detailed continuous transients instead of explicit discontinuous jumps. When new components are added to the system, one has to re-evaluate the detailed continuous interaction between the different states (modes) of the overall hybrid automata based on the complex ODEs to derive the discontinuous jumps. If the interactions are analyzed systematically at compile time, as was done for the actuator system, one can build efficient hybrid automata that can be used for real time applications. We have applied this methodology to analyze computationally complex sliding mode simulations [17], and to construct hybrid observers that track real time

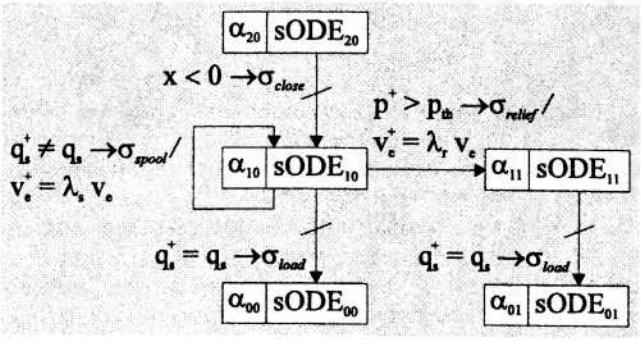


Figure 10: **Hybrid automata for the operation of an actuator.**

behavior of the elevator system [14] with promising results.

To clarify this further, note that $\lambda$ is a parameter that describes the elevator velocity change because of damping parameters in the cylinder, but its value is determined by the time $t_d$ during which the oil flow into the cylinder is blocked. This blockage occurs in a configuration where the spool valve and relief valve are closed. However, these are separate model components, and, therefore, utilizing knowledge about their individual behavior to simplify the cylinder model results in a model component that is configuration specific. Consequently, the cylinder model is only valid in this specific configuration and new values for $\lambda$ have to be derived when it is applied in a different configuration. For example, if another spool valve is cascaded with the existing one, the $t_d$ may change, and, therefore, $\lambda_s$ in the cylinder model differs.

This shows that composability of model fragments is limited by the abstraction level of the fragments themselves. If the model fragments do not include explicit discontinuous state vector value changes, composability is preserved. This requires including small and large parameter values to achieve complex ODEs that incorporate fast transient behavior when mode switches occur. These fast transients are governed by *contact behavior* [16] that can be abstracted away to achieve simpler models. However, the contact behavior is the result of interactions between connected model fragments, and, therefore, the abstraction only holds for the specific configuration.

## 6 Conclusions

In this paper we have developed a systematic methodology derived from singular perturbations for generating simpler ODE models by applying time scale and parameter abstractions to complex nonlinear system models that exhibit fast transient behavior. The key to this methodology is the ability to decouple the fast transients from the slower behaviors, and solve for the fast transients to obtain a quasi steady state solution. This solution introduced to the original set of ODEs generates a lower order set of ODEs, and this simplified behavior

analysis. Enforcing the different semantics associated with the two types of abstraction, allows the derivation of discrete transition conditions between the modes of continuous behavior. Compiling the sODEs and transition conditions into hybrid automata generates run time models that can be used for real time simulation and analysis of system behavior.

The apparent drawback of creating piecewise simpler hybrid models is that the compositionality property is lost when interactions between the component subsystems have to be analyzed in advance to build the hybrid automata. We will investigate this issue in greater detail in future work.

## References


[1] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Lecture Notes in Computer Science*, volume 736, pages 209–229. Springer-Verlag, 1993.

[2] Raymond M. Brach. *Mechanical Impact Dynamics.* John Wiley and Sons, New York, 1991.

[3] B. Falkenhainer and K. Forbus. Compositional modeling: Finding the right model for the job. *Artificial Intelligence*, 51:95–143, 1991.

[4] W. L. Green. *Aircraft Hydraulic Systems.* John Wiley, Chichester, UK, 1985.

[5] John Guckenheimer and Stewart Johnson. Planar hybrid systems. In Panos Antsaklis, Wolf Kohn, Anil Nerode, and Shankar Sastry, editors, *Hybrid Systems II*, volume 999, pages 202–225. Springer-Verlag, 1995. Lecture Notes in Computer Science.

[6] Yumi Iwasaki and Inderpal Bhandari. Formal basis for commonsense abstraction of dynamic systems. In *Proc. AAAI*, pages 307–312, 1988.

[7] Petar V. Kokotović, Hassan K. Khalil, and John O'Reilly. *Singular Perturbation Methods in Control: Analysis and Design.* Academic Press, London, 1986. ISBN 0-12-417635-6.

[8] Benjamin Kuipers. Qualitative simulation using time-scale abstraction. *International Journal of Artificial Intelligence in Engineering*, 3:185–191, 1988.

[9] Bengt Lennartson, Michael Tittus, Bo Egardt, and Stefan Pettersson. Hybrid systems in process control. *IEEE Control Systems*, pages 45–56, October 1996.

[10] Alon Levy, Yumi Iwasaki, and Richard Fikes. Automated model selection for simulation based on relevance reasoning. *Artificial Intelligence*, 96:3–32, 1997.

[11] Pieter J. Mosterman and Gautam Biswas. Formal Specifications for Hybrid Dynamical Systems. In *IJCAI-97*, pages 568–573, Nagoya, Japan, August 1997.

[12] Pieter J. Mosterman and Gautam Biswas. Principles for Modeling, Verification, and Simulation of Hybrid Dynamic Systems. In *Fifth International Conference on Hybrid Systems*, pages 21–27, Notre Dame, Indiana, September 1997.

[13] Pieter J. Mosterman and Gautam Biswas. A theory of discontinuities in dynamic physical systems. *Journal of the Franklin Institute*, 335B(3):401–439, January 1998.

[14] Pieter J. Mosterman and Gautam Biswas. Building Hybrid Observers for Complex Dynamic Systems using Model Abstractions. In Frits W. Vaandrager and Jan H. van Schuppen, editors, *Hybrid Systems: Computation and Control*, pages 178–192, 1999. Lecture Notes in Computer Science; Vol. 1569.

[15] Pieter J. Mosterman, Gautam Biswas, and Janos Sztipanovits. A hybrid modeling and verification paradigm for embedded control systems. *Control Engineering Practice*, 6:511–521, 1998.

[16] Pieter J. Mosterman and Peter Breedveld. Some Guidelines for Stiff Model Implementation with the Use of Discontinuities. In *ICBGM99*, pages 175–180, San Francisco, January 1999.

[17] Pieter J. Mosterman, Feng Zhao, and Gautam Biswas. Sliding mode model semantics and simulation for hybrid systems. In *Hybrid Systems V.* Springer-Verlag, 1998. Lecture Notes in Computer Science.

[18] T. Nishida and S. Doshita. Reasoning about discontinuous change. In *Proceedings AAAI-87*, pages 643–648, Seattle, Washington, 1987.

[19] J. Seebeck. *Modellierung der Redundanzverwaltung von Flugzeugen am Beispiel des ATD durch Petrinetze und Umsetzung der Schaltlogik in C-Code zur Simulationssteuerung.* Diplomarbeit, Arbeitsbereich Flugzeugsystemtechnik, Technische Universität Hamburg-Harburg, 1998.

[20] Dan Weld and Johan de Kleer. *Readings in Qualitative Reasoning about Physical Systems.* Morgan Kaufmann, San Mateo, CA, 1990.