Efficient Tracking of Behavior in Complex Hybrid Systems via Hybrid Bond Graphs

Benjamin Podgursky [1], **Gautam Biswas** [1], **and Xenofon Koutsoukos** [1]

[1] *Dept of EECE/ISIS, Vanderbilt University, Nashville, TN, 37235, United States*
benjamin.t.podgursky@vanderbilt.edu
gautam.biswas@vanderbilt.edu
xenofon.koutsoukos@vanderbilt.edu

ABSTRACT

For many real-world systems, which exhibit complex, nonlinear, and hybrid behavior, it is important to accurately track and monitor the state and health of these systems. The continuous state estimation problem has been well studied, and a number of extensions of the Kalman filter to nonlinear systems have been proposed. Hybrid state estimation poses an additional challenge, because the model must be quickly updated during a mode change to facilitate accurate, real time tracking. This paper discusses an approach to minimize the amount of equation regeneration necessary when the system undergoes hybrid mode changes. These equations are used as the state update equation for an Unscented Kalman Filter that tracks the system's state. We demonstrate the effectiveness of our approach by tracking the hybrid behaviors of NASA's ADAPT test bench. Results show that our algorithm scales well for tracking large nonlinear and hybrid systems.

1. INTRODUCTION

Many mission critical systems, such as aircraft and power generation systems, exhibit complex nonlinear and hybrid behaviors. Hybrid systems are characterized by intervals of continuous behavior interspersed with discrete changes. With increased needs for safety and reliability, it is becoming important to monitor system behavior online, and couple the monitoring system with accurate fault detection and isolation mechanisms. In another example, accurate and robust tracking is the primary functionality of aircraft avionics systems.

Accurate tracking of complex, nonlinear hybrid behaviors is in itself a major challenge. In realistic situations, this challenge is further compounded by noisy sensors and inaccuracies in the system models. Kalman filters and their extensions (Lefebvre, Bruyninckx & Schutter, 2004) have been used extensively for tracking online behaviors. We briefly discuss two of the formalisms: (1) the Extended Kalman Filter (EKF) (Welsh & Bishop, 2006), and (2) the Unscented Kalman Filter (UKF) (Julier & Uhlmann, 1997) in the next section.

Tracking the hybrid system mode and state adds additional challenges to the nonlinear continuous state estimation problem since the modes, i.e., the configurations of the system, and therefore, the dynamic system model change during system behavior evolution. For example, aircraft operate in different modes that include taxi, takeoff, cruise, descend, and land. Traditionally hybrid systems have been modeled as Hybrid Automata (Cuijpers & Reniers, 2005), (Henzinger, 1996), where each mode of operation is described as a state of the automaton, and the automata specifies the transitions between different modes of operation. Within each mode, the continuous state of the system is modeled by a set of differential equations. This representation works well for modeling small systems; however, modeling of large, complex systems requires complete enumeration of all of the system modes. Pre-enumerating all of these modes is wasteful in both space and time, because most modes may not occur at runtime. On some large systems, complete mode enumeration may be completely infeasible.

A more efficient method for tracking hybrid behavior is to generate models for each mode only when that mode is visited. However, model generation after a mode change must occur quickly, before the state estimate diverges from the true state.

In this paper, we adopt the Hybrid Bond Graph (HBG) approach to model nonlinear hybrid systems (Mosterman & Biswas, 1998). Real world systems are large, nonlinear, and may have lots of switching components. HBGs represent a paradigm for representing these large complex systems in a compact form. This is because in the HBG representation, system modes do not have to be pre-enumerated, instead they are generated at runtime when a transition occurs to that mode. We use the equations generated from the HBG model in a mode to formulate the equations for the Unscented Kalman Filter, which tracks the behavior of the system.

Section 2 defines the filtering problem and discusses the strengths and weaknesses of the Extended Kalman

Filter and Unscented Kalman Filter approaches for tracking system state. Section 3 describes an efficient incremental regeneration process to derive the system state equations when mode changes occur. Section 4 discusses an implementation of our modeling framework in the Generic Modeling Environment (GME), and the use of interpreters within the environment to generate the state equations and the filter to estimate the system state. In section 5 we run comparative analysis experiments with the generated UKFs and EKFs on several nonlinear systems at varying noise levels. We also show how the structure of our equation representation allows us to improve runtime performance by caching intermediate results. Last, we show how automated equation generation and state tracking via the UKF allows us to easily model large hybrid systems.

2. STATE ESTIMATION USING KALMAN FILTERS

The Kalman filter (Welsh, 2006) is known to be an optimal filter for linear time invariant dynamical systems where modeling errors and measurement noise are Gaussian. Unfortunately, solutions to the corresponding optimal nonlinear filtering problem are infinite-dimensional in the general case, and nonlinear filters must approximate the optimal solution using linearization or sampling methods. EKFs and UKFs are extensions of the linear Kalman filter to nonlinear systems.

2.1 The Extended Kalman Filter

The Extended Kalman Filter (EKF) was for a long time the standard algorithm for nonlinear state estimation; an excellent description of the EKF can be found in (Welsh, 2006), which forms the basis for the description below. The general idea behind the EKF is that by linearizing the nonlinear state and measurement equations about the current state estimate, the prediction and update equations from the linear Kalman Filter can be applied to a nonlinear system. The EKF assumes that the system is updated by the discrete-time non-linear stochastic difference equation f, with measurements generated by h:

$$\begin{aligned} x_k =& f(x_{k-1}, u_{k-1}, w_{k-1}) \\ z_k =& h(x_{k-1}, v_{k-1}) \end{aligned}$$

Since the process and measurement noise are not known, x and z are approximated:

$$\begin{aligned} \overline{x}_k =& f\left(\widehat{x}_{k-1}, u_{k-1}, 0\right) \\ \overline{z}_k =& h\left(\overline{x}_k, 0\right) \end{aligned}$$

An estimation step is broken into two parts: predict and update. In the predict step, the system model is used to propagate the state estimate forward:

$$\begin{aligned} \widehat{x}_k^- =& f(\widehat{x}_{k-1}, u_{k-1}, 0) \\ P_k^- =& A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T \end{aligned}$$

In the update step, the measurement is combined with the state estimate to get a new state and covariance estimate:

$$\begin{aligned} K_k =& f\left(\widehat{x}_{k-1}, u_{k-1}, 0\right) \\ \widehat{x}_k =& x_k^- + K_k\left(z_k - h\left(\widehat{x}_k^-, 0\right)\right) \\ P_k =& \left(I - K_k H_k\right) P_k^- \end{aligned}$$

The EKF linearizes f and h about x to generate the A, W, H, and V matrices.

$$\begin{aligned} A_{[i,j]} =& \frac{df_{[i]}}{dx_{[j]}}\left(\widehat{x}_{k-1}, u_{k-1}, 0\right) \\ W_{[i,j]} =& \frac{df_{[i]}}{dw_{[j]}}\left(\overline{x}_{k-1}, u_{k-1}, 0\right) \\ H_{[i,j]} =& \frac{dh_{[i]}}{dx_{[j]}}\left(\overline{x}_k, 0\right) \\ V_{[i,j]} =& \frac{dh_{[i]}}{dv_{[j]}}\left(\overline{x}_k, 0\right) \end{aligned}$$

There are two main drawbacks to the first-order linearization approach used by the EKF. First, it is well known that, because linearization is only accurate to the first-order, the EKF does not perform well when tracking highly nonlinear systems. Second, generating symbolic solutions for A, W, H, and V can be significantly more difficult than generating the system state equations (Julier, 1997). These Jacobian matrices can be computed numerically, but on nonlinear systems this can lead to numerically unstable behavior.

2.2 The Unscented Kalman Filter

The Unscented Kalman Filter (Julier, 1997) is an alternate approach for nonlinear tracking. The UKF, an extension to the Kalman filter, uses a deterministic sampling approach instead of an explicit linearization. Like the EKF, the UKF assumes a discrete time nonlinear state transition function:

$$\begin{aligned} x_{k+1} =& F\left(x_k, u_k\right]) + v_k \\ y_k =& H\left(x_k\right) + n_k \end{aligned}$$

Like the EKF, the UKF operates with alternating time and measurement update steps (Merwe & Wan, 2001), outlined below: Initialize:

$$\begin{aligned} \widehat{x}_0 =& E\left[x_0\right] \\ P_0 =& E\left[\left(P_0 - \widehat{x}_0\right)\left(x_0 - \widehat{x}_0\right)^T\right] \end{aligned}$$

For $k \in \{1, \cdots, \infty\}$

$$X_{k|k-1}^* = \left[\widehat{x}_{k-1}\widehat{x}_{k-1} + \gamma\sqrt{P_{k-1}}\widehat{x}_{k-1} - \gamma\sqrt{P_{k-1}}\right]$$

Time update:

$$\begin{aligned} X_{k|k-1}^* =& F\left(X_{k-1}, u_{k-1}\right) \\ \widehat{x}_k^- =& \sum_{i=0}^{2L} W_i^{(m)} X_{i,k|k-1}^* \\ P_k^- =& \sum_{i=0}^{2L} W_i^{(c)}\left[X_{i,k|k-1}^* - \widehat{x}_k^-\right]\left[X_{i,k|k-1}^* - \widehat{x}_k^-\right]^T \\ & + R^v \\ X_{k|k-1} =& \left[\widehat{x}_k^-\widehat{x}_k^- + \gamma\sqrt{P_k^-}\widehat{x}_k^- - \gamma\sqrt{P_k^-}\right] \\ Y_{k|k-1} =& H\left[X_{k|k-1}\right] \\ \widehat{y_k^-} =& \sum_{i=0}^{2L} W_i^{(m)} Y_{i,k|k-1} \end{aligned}$$

Measurement update:

$$
\begin{aligned}
P_{\widetilde{y}_k\widetilde{x}_k} =& \sum_{i=0}^{2L} W_i^{(c)} \left[Y_{i,k|k-1} - \widehat{y_k^-}\right] \left[Y_{i,k|k-1} - \widehat{y_k^-}\right]^T \\
&+ R^n \\
P_{x_k y_k} =& \sum_{i=0}^{2L} W_i^{(c)} \left[X_{i,k|k-1} - \widehat{x_k^-}\right] \left[Y_{i,k|k-1} - \widehat{y_k^-}\right]^T \\
K_k =& P_{x_k y_k} P_{\widetilde{y}_k\widetilde{y}_k}^{-1} \\
\widehat{x}_k =& \widehat{x_k^-} + K_k\left(y_k - \widehat{y_k^-}\right) \\
P_k =& P_k^- - K_k P_{\overline{y}_k\overline{y}_k} K_k^T
\end{aligned}
$$

$\{W_i\}$ is the set of weights:

$$
\begin{aligned}
W_0^{(m)} =& \frac{\lambda}{(L+\lambda)} \\
W_0^{(c)} =& \frac{\lambda}{(L+\lambda)} + (1 - \alpha^2 + B) \\
W_i^{(m)} =& W_i^{(c)} = \frac{1}{(2\,(L+\lambda))}, i = 1, \cdots, 2L \\
\lambda =& \alpha^2\,(L+\kappa) - L \\
\gamma =& \sqrt{(L+\lambda)}
\end{aligned}
$$

λ, α, and β are scaling parameters. R^v is the process noise covariance. R^n is the measurement noise covariance.

By not linearizing the state equations about the state estimate, the UKF addresses two of the main weaknesses of the EKF. First, when using the UKF, f and h do not need to be differentiated, making automated equation generation significantly more feasible. Second, the UKFs deterministic sampling captures both the mean and covariance of a Gaussian Random Variable, which when propagated through the nonlinear system captures the estimate mean and covariance to the 3rd order for any nonlinearity, as opposed to the first order approximation from the EKF. Many papers have compared the performance of the UKF and EKF, and many have found that in both theoretical results (Lefebvre, 2004) and practical applications (Romanenko & Castro, 2003) the UKF matches or outperforms the EKF for nonlinear systems.

2.3 Extending the tracking problem to hybrid systems

Hybrid system behavior introduces new challenges to system tracking, because hybrid behaviors have to include mode changes. To track the state of a hybrid system, an observer must now do three things: (1) check for and detect mode changes, (2) update the system model when mode changes occur, and (3) perform continuous estimation in each mode of operation.

Hybrid mode estimation is a well researched area, and a number of techniques have been developed for simultaneous state and mode estimation. Many techniques maintain a bank of state estimate trajectories, and use hidden Markov models (HMMs) to estimate the most likely current mode. Mode changes triggered by the continuous system state update the HMM beliefs, and each of the state trajectories is updated by the modes the HMM finds most probable (Hofbaur & Williams, 2002). For this paper, we do not attempt to estimate the system mode; in all of the models in this paper, mode changes are interpreted as clearly observed system inputs. The techniques described here, though, could be used in conjunction with mode estimation techniques.

After the new mode is determined, it is important to quickly update the system model so that the filters state estimate does not diverge from the correct estimate while the equations are being updated. The linearization performed by the EKF is even more of a computational burden in a hybrid system because every mode change requires the re-generation of the symbolic Jacobian matrices used to linearize the state equations. To use the EKF on a hybrid system, the matrices A, H, W, and V, as well as f and h, need to be recomputed at each mode change. The UKF requires updated f and h functions, but has no matrices to update. This makes the UKF significantly more desirable as a state estimator for hybrid and nonlinear systems. Last, within each system mode, the filter must still estimate the continuous state of the nonlinear system, given noisy measurements and an imperfect system model.

Figure 1 illustrates this process. The techniques presented in this paper speed up the model update step by efficiently regenerating the state equations incrementally, rather than regenerating the entire set of state equations on a mode change.

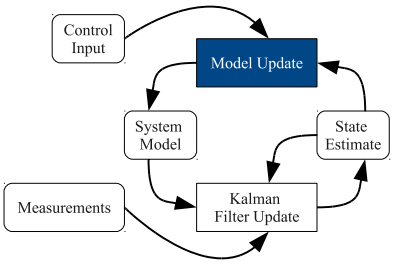

Figure 1: State estimation process in a hybrid system

3. HYBRID BOND GRAPH MODELING OF HYBRID SYSTEMS

Bond Graphs are a topological domain-independent energy-based paradigm for modeling physical systems and processes (Karnopp, Margolis & Rosenberg, 2006). One of the strengths of this modeling paradigm is that it has a strong correspondence with free-body diagrams in mechanics, circuit diagrams in electricity, and similar representations in other physical domains. A Bond Graph is built of primitive elements which include storage elements (capacitors, C and inductors, I), elements which dissipate energy (resistors, R), and those which transform energy from one form to another (transformers, TF, and gyrators, GY.) Sources of effort (Se) and sources of flow (Sf) define interactions of the system with its environment. These primitive components are connected to each other via junctions (0 and 1 junctions), which represent parallel and series connections between components. Bonds represent energy connections, and have associated effort and flow variables with $effort \times$

$flow$ defining the rate of energy flow (Roychoudhury, Daigle, Biswas & Koutsoukos, 2010).

Hybrid Bond Graphs (HBGs) extend Bond Graphs by introducing discontinuous behavior via *switching* junctions, which turn on or off as a function of system state and/or control input. (Mosterman, 2002) presents a more in-depth discussion of the HBG modeling paradigm and the semantics of the switching junctions. HBGs are a powerful paradigm for representing large hybrid systems because they can compactly represent a hybrid system; the equations for each state do not need to be enumerated prior to execution, but instead can be generated as needed. This is an advantage over modeling paradigms like Hybrid Automata, which generally require an enumeration of all system modes prior to execution (Alur, Courcoubetis & Ho, 1993).

Modulated components allow the modeler to introduce nonlinearities into a Hybrid Bond Graph model. When a component is modulated, its parameter is a function of other effort or flow variables. Modulated functions introduce potentially un-resolvable algebraic loops into the state equations, so during simulation, only the previous time steps value of the modulated functions input variables are used. This is the same approach to modulated functions taken by HyBrSim (Mosterman, 2002) and block diagrams (Daigle, Roychoudhury, Biswas, Koutsoukos & Daigle, 2007).

3.1 Behavior Generation for Hybrid Bond Graphs

As a hybrid system, the behavior of a HBG is defined by continuously differentiable behavior within a mode, and discrete mode changes when the system state equations change. To simulate a HBG, state equations must be generated for each mode, and then numerically solved via methods such as Euler or Runge-Kutta for the duration of the mode. When the state of a switching junction changes, triggered either by external input or from internal switching, new state equations must be generated.

Continuous Behavior Generation

Generating the continuous behavior of a system is straightforward when the state equations are a set of ODEs. When the causality of a HBG is fixed, generating ODEs modeling a HBG is a straightforward process. The equations for each bond graph component can guide effort and flow variable substitutions, until the derivative of each state variable is a function of the system state, parameters, and inputs.

The equation generation process is guided by the causal configuration of a HBG. The rules for causality assignment are given in detail in (Roychoudhury, 2010). The algorithms discussed here generate equations only for systems where every component can be assigned integral causality. When the causality assignment rules do not assign a causal stroke to every bond, the remaining bonds can be given any valid causality assignment, and the model is said to have a zero-order causal path. When this happens, equation generation is more complicated (Dijk & Breedveld, 1991), and the resulting equations are said to contain an algebraic loop.

Work has already been done to improve the efficiency of HBGs when modeling large hybrid systems. Block diagrams represent the equations for each junction as connections between junctions which can be easily reconfigured. The Hybrid SCAP algorithm describes how to efficiently reconfigure a block diagram on hybrid mode changes (Daigle, 2007). The algorithms presented below extend the Block Diagram representation by performing symbolic manipulation to derive the state equations.

Building State Equations From HBGs

The equation representation strategy presented here represents the equations of a Bond Graph as a directed acyclic graph, which we call a *Hybrid Equation*. In the graph, a child node ofa variable represents an expression it is equivalent to when particular structural constraints are satisfied. The children of an operator node are its operands. As the system transitions between modes, the structural constraints of the HBG change, and variables acquire new child nodes when new modes are visited. The previous child nodes are cached to allow easy retrieval if a previously generated mode is revisited.

When the mode of a HBG changes, in most cases the majority of the state equations will not change. If the equations are cached properly, only the parts of the equations which have changed will need to be regenerated. While generating the equations for a HBG, we note the *constraints* imposed by the current system mode during the equation generation process, when the causality and mode guide the variable substitutions. To see which constraints it is important to record, consider what changes to the HBG occur during a hybrid mode change: 1) the determining bond of junctions change, and 2), non determining bonds turn on or off. (1) will only affect the equation generation in two cases; when following the flow into a 1-junction or the effort into a 0-junction. Therefore, each of these variable substitutions is augmented with a constraint of the form $< B_x$ determines $J_y >$, indicating that the substitution is only valid while bond x is the determining bond of junction y. (2) will change only the effort at a 1-junction or the flow at a 0-junction. Therefore, each of these substitutions is augmented with the constraint $< x_1 \cdots x_n$ active for $J_z >$, where $x_1 \cdots x_n$ are booleans which indicate the state of each of the bonds attached to junction z.

From these observations, we get algorithm 1 which builds the initial state equations for the HBG. Algebraic loop resolution is discussed in more detail later.

Algorithm 1 Build initial equations

1: Run hybrid SCAP to assign causality to the current bond graph configuration
2: Run $generate\ equations$
3: Run $resolve\ loops$

When generating equations which are valid in multiple system modes, the effort and flow variables must be augmented with the direction from which the substitution occurs. Consider f_2 in Model 2; when junction o is on, the component equations contain: $f_2 = \frac{e_2}{R}$; however, when o is off, bond 2 determines z, and the component equations contain $e_2 = f_2 R$. If there is only a single f_2 node in the graph, the graph will be cyclic, as the equalities $f_2 = \frac{e_2}{R}$ and $e_2 = f_2 R$ are both valid in both modes. This can be prevented by noting from which direction a variable substitution is made, and maintaining distinct nodes for a variable found by following a bond (labeled in) and going against the bond (labeled out.) So in Model 2, the equations for

Algorithm 2 Generate equations

Require: stack $unexplored$ containing equations to generate
while $unexplored$ not empty **do**
  pop $node$ from $unexplored$
  if $node$ is a non-state variable and does not have a valid substitution **then**
    let sub be the direct substitution for $node$
    add subsitution sub to $node$ with its constraints
    set sub as $node$'s current substitution
    if sub is a variable **then**
      push sub onto $unexplored$
    else if sub is an operator **then**
      push the children of sub onto $unexplored$
    end if
  end if
end while

bond 2 are instead $f_{2out} = \frac{e_{2in}}{R}$, and $e_{2out} = f_{2in}R$.

Example 1: Derivation of state equations for a simple model (model 3):

The equations for each component and the derived state equations are shown below.

$$\begin{aligned} e_1 &= C:q \\ e_3 &= C1:q \\ f_5 &= Sf \\ f_2 &= \frac{e_2}{R} \\ f_1 &= f_4 = f_2 \\ e_5 &= e_4 = e_3 \\ e_2 &= e_4 - e_1 \\ f_3 &= f_5 - f_4 \end{aligned}$$

$$C:q' = \frac{1}{C} f_1 = \frac{1}{CR}\left(C1:q - C:q\right)$$

$$C1:q' = \frac{1}{C1} f_3 = \frac{1}{C1}\left(Sf - \frac{1}{R}\left(C1:q - C:q\right)\right)$$

The hybrid equation structure captures the variable substitutions that are necessary to derive state equations for $C:q'$ and $C1:q'$, using the component equations above. Figure 4 shows the full Hybrid Equation for figure 2. The first variable substitution $C:q' = \frac{1}{C} f_1$ is represented by an arc from $C:State'$ to $\frac{f_1}{C}$ in figure 4. f_1 is not a state variable, so we can continue by using the equality $f_1 = f_2$, adding another arc. We contine this process until the full state equations are generated.

Algebraic Loop Resolution

When the causality of a Bond Graph is not completely fixed by storage or source elements, there exists a zero-order causal path between two or more resistive elements (Dijk, 1991), and the state equations of the Bond Graph implicitly represent a set of Differential Algebraic Equations (DAEs) instead of a set of ODEs. There are three general approaches to dealing with algebraic loops in Bond Graphs: structural augmentation, numerical methods, and symbolic resolution. Structural augmentation

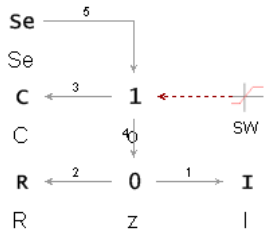

Figure 2: Bond 2 will determine z in only one mode

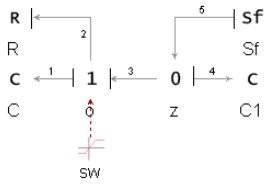

Figure 3: A simple bond graph

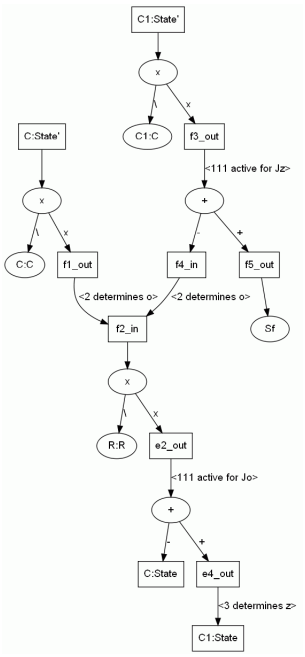

Figure 4: Hybrid equation for figure 3

removes zero-order causal paths by inserting small state elements which fix the causality of the system (Gawthrop & Smith, 1992). Unfortunately, augmented models include additional state variables, and the resulting system of equations may be stiff, causing numerical issues during simulation. We would like to avoid adding unnecessary state variables, as this software is intended to assist in state and parameter estimation. Numerical methods leave the equations in DAE form, and use a numerical DAE solver to solve for the future state of the system. However, numerically solving a DAE system can be very computationally expensive (Borutzky & Cellier, 1996).

Several groups have used symbolic manipulation on the systems DAEs to derive a system of ODEs. 20-sim, a Bond Graph modeling and simulation tool, first attempts algebraic manipulation before resorting to numerical methods (20-sim, 2009). In (Borutzky, 1996), techniques are developed to intelligently choose variables to use to tear the system when converting it to an ODE. While ODE solutions to DAE systems are not possible in the general nonlinear case, the use of timestep-delayed modulated functions to model nonlinearities means that our DAE system will contain only terms linear with respect to the variables being solved for. This indicates that it is possible to convert our original DAEs into a system of ODEs, albeit with a potentially large amount of symbolic manipulation.

If the equations representing a Bond Graph contain an algebraic loop, it will be clearly identifiable as a back-edge in the Hybrid Equation (figure 6). Loops are resolved by finding a variable contained in an algebraic loop and algebraically compiling a solution for the variable. By noting the constraints which define the structure of the algebraic loop, once a loop is resolved in one mode, the solution will remain valid for other modes, and will not need to be re-solved in many future modes. The algorithm for resolving algebraic loops is shown below.

Algorithm 3 Resolve loops

1: **while** a variable v can be found in an algebraic loop **do**
2: let $originalSub$ = v's current substitution
3: call $copy\ and\ compile$ on v to generate sub
4: replace v's original substitution with sub
5: solve the loop for v
6: find a new variable v in an algebraic loop
7: **end while**

In $solve\ equation$, x and y could potentially still contain references to v, if there is an intermediate variable still in an algebraic loop; otherwise function $copy\ and\ compile$ would not terminate. The variable used to break the algebraic loop is found via a DFS of the graph which returns the first variable found in a cycle. This variable will not necessarily be the optimal choice to solve; choosing the minimal number of variables to solve is in general NP-complete (Borutzky, 1996). In the simple case with one zero-order causal path, the algorithm will always terminate in a single iteration.

Example 2: Derivation of state equations for a model with an algebraic loop (figure 5)

The component equations for figure 5 are shown.

Algorithm 4 Copy and compile equation

Require: node v
1: **if** v is a state variable, v is a parameter, or v has already been seen **then**
2: **return** v
3: **else if** a substitution s for v has already been generated **then**
4: **return** the s
5: **else if** v is a non-state variable **then**
6: note the constraints on v's substitution
7: **return** CopyAndCompile on v's current substitution
8: **else if** v is is an addition (multiplication) operator **then**
9: let $newOp$ = a new addition (multiplication) operator
10: recursively copy each of v's children and make them children of $newOp$
11: **return** $newOp$
12: **end if**

Algorithm 5 Solve equation

Require: node v
1: simplify the substitution for v into the form $vx + y$
2: resolve the loop by setting $v = \frac{y}{1-x}$

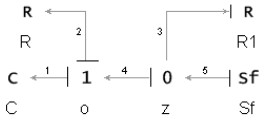

Figure 5: Simple model with an algebraic loop

$$
\begin{aligned}
e_2 &= Rf_2 \\
e_1 &= C:q \\
f_5 &= Sf \\
f3 &= \frac{1}{R1} e_3 \\
e_5 &= e_3 = e_4 \\
f_1 &= f_2 = f_4 \\
f_4 &= f_5 - f_3 \\
e_4 &= e_1 + e_2 \\
C:q' &= \frac{1}{C} f_1 = \frac{1}{C} \left(\frac{-\frac{1}{R1} C:q + Sf}{1 + \frac{R}{R1}} \right)
\end{aligned}
$$

Figure 7 illustrates this procedure as represented by a Hybrid Equation. The initial equations for figure 5 contain an algebraic loop: $f_4 = f_5 - f_3 = Sf - \frac{1}{R1} C : q - \frac{R}{R1} f4$. This problem manifests itself in the Hybrid Equation as a cycle, as seen in figure 6. By algebraically compiling a solution, f_4 can be solved for explicitly:

$$f_4 = \frac{Sf - \frac{1}{R1} C:q}{1 + \frac{R}{R1}}$$

Figure 7 shows the hybrid equation after resolving the algebraic loop and generating explicit state equations. f_{4out} was the variable used to break the cycle. The new symbolic substitution for f_{4out} is equivalent to the one computed above. The constraints on this substitution correspond to the constraints that defined the structure of the algebraic loop in Figure . The loop will not need to be re-solved for any subsequent mode in which these constraints hold.

One situation this system is not able to handle is the presence of causal loops (Dijk, 1991), where no explicit set of ODEs can be generated. We would need to implement numerical methods to simulate such a system. In the systems our research focuses on, we have not encountered a system with a causal loop which could not be reduced into an equivalent system lacking causal loops.

Evaluation

Once algebraic loops have been removed from the equations, the value of any node can be computed recursively, as shown in algorithm 6.

Algorithm 6 Evaluate

Require: node v
if the value of v has already been cached **then**
  return the cached value
else if v is a state variable or parameter **then**
  $value =$ the value of $evaluate$ on v's substitution
else if v is a non-state variable **then**
  $value =$ the value of $evaluate$ on v's substitution
else if v is an addition (multiplication) operator **then**
  $value =$ the sum(product) of all children
  cache $value$ for node v
end if
return $value$

Mode Changes

When the mode of a HBG changes, some constraints will be invalidated, and the parts of the state equations which depend on them will need to be regenerated. We can tell which constraints have changed: 1) constraints which rely on the determining bond of a junction, which can be found while tracing causality changes through the system during the Hybrid SCAP procedure, and 2) constraints for junctions which neighbor a bond which has changed state. The algorithm for equation generation has already been defined, so the equation update algorithm is simple.

Algorithm 7 Update equation on mode change

update all junction state to reflect structural changes
perform Hybrid SCAP
for all variables v with an invalid substitution **do**
  if v has a different substitution s which satisfies the active constraints **then**
    set s as the current substitution of v
  else
    generate the updated equation for v
  end if
end for
resolve algebraic loops

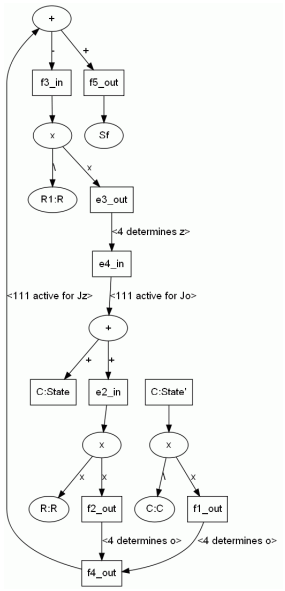

Figure 6: Hybrid equation for figure 5 with algebraic loop

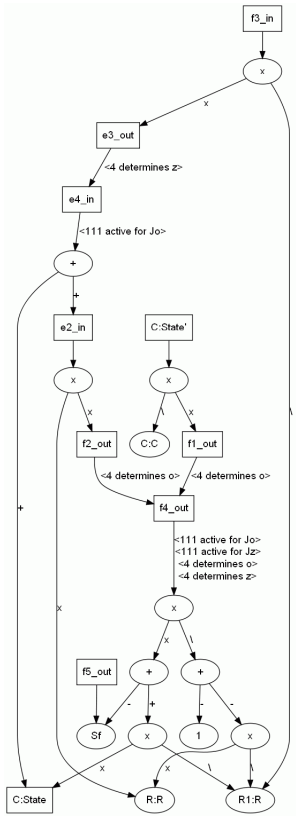

Figure 7: Hybrid equation for figure 5 after algebraic loop resolution

Only if a variable has a substitution which a mode change has made invalid, will the substitution need to be recomputed. If the system ever re-enters a previously visited mode, every necessary variable will have at least one valid substitution. As more modes are reached, it becomes increasingly likely that a new substitution will not need to be generated with a mode change.

Example 3: In model 3, junction o is a switching junction. When it turns off, the state equations change:

$$C : q' = 0$$
$$C1 : q' = \frac{1}{C1} Sf$$

Figure 8 shows the hybrid equation for figure 3 after the mode change. The bonds active for junction z have changed, so f_3 has a new substitution in the new mode. C is disconnected from the bond graph when bond 1 turns off, so the new ODE $C : State = 0$ does not need to be explicitly stored.

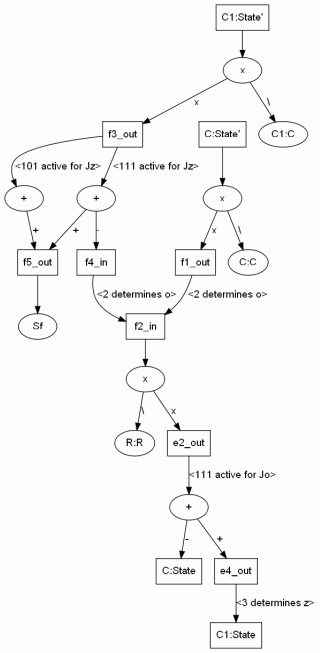

Figure 8: The equations for figure 3 after the 1-junction switches off

3.2 Performance Improvements

A Hybrid Equation is sufficient for numerical simulation, but this equation structure facilitates two strategies which improve runtime performance: caching and compilation.

Caching

In $evaluate$, the computed value of each node is cached for later reuse. Because nodes in a Hybrid Equation can have multiple parents, and in most HBG models, a large number of intermediate variables are shared between equations, this often saves a considerable amount of processing time. The ODEs generated could be solved by an external ODE solver, but it is difficult to ensure that a third-party ODE solver is able to take advantage of this redundancy. Other model-based approaches such as Block Diagrams also implicitly cache intermediate values.

Compilation

To improve simulation performance when the system is in a frequently occurring mode, it is possible to compile a state specific *state equation*, which is a compressed version of the Hybrid Equation valid only that single mode. The generation of the structure may or may not be worthwhile, depending on how long the system is expected to remain in that mode. State Equations can be stored or discarded and regenerated as necessary. To handle memory constraints, it is possible to set up a LRU cache of State Equations, regenerating state equations which are cache misses. A state equation is generated by calling $copy\ and\ compile$ on each derivative and observed variable. This procedure preserves the intermediate caching in $evaluate$. Figure 9 shows state equations for two models.

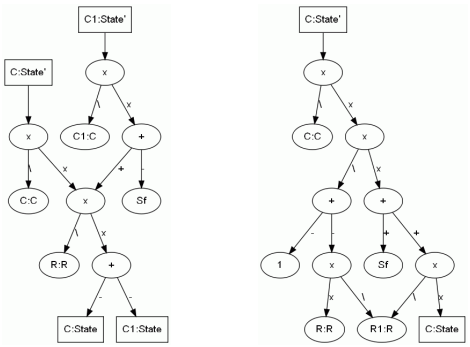

Figure 9: Compressed equations for figures 3 and 5

3.3 Summary of HBG representation and equation generation

HBGs provide a compact representation of a hybrid system which does not require a full enumeration of all the reachable states in a hybrid model. By taking note of structural constraints in the bond graph while generating state equations, we can efficiently update the state equations when the system mode changes. Having an explicit system of equations allows us to resolve algebraic loops via symbolic manipulation. The loop will not need to be resolved again in later modes as long as the structural constraints which define it are not violated. To improve performance at runtime, we can compile the state equations and cache intermediate results when evaluating the value of a variable.

4. IMPLEMENTATION

The Generic Modeling Environment is a meta-modeling environment which facilitates the construction of domain specific modeling languages (Agrawal, Karsai & Ledeczi, 2003). The HBG modeling paradigm has been implemented as a meta-model in GME. This allows for a graphical, component-oriented approach for building system models using a drag and drop interface. Because component interfaces are well-defined, it is easy to build

large system models that are syntactically and semantically correct by composition. Interpreters can then be applied to these models to convert them into the different forms needed for building the observers for tracking system behavior.

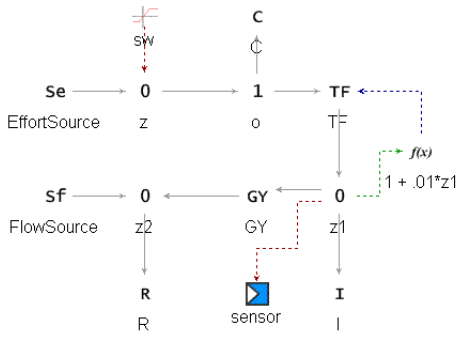

Figure 10: An example of a HBG built using GME. Sw is a decision function which specifies the state of junction z. TF is a modulated transformer, whose value is a function of the effort at z_1. A sensor measures the effort across z_1.

The incremental equation generation algorithms described in section 3. were implemented as a stand-alone C++ application, using textual representations of bond graphs produced by GME. We use the extensible open source Bayes++ library of filter classes (Stevens, 2006) to implement both the EKF and UKF. The state and observation equations for the models evaluated in section 5. were derived automatically. On models where we tested the performance of the EKF, the symbolic solutions for the matrices A, H, W, and V for each model were derived by hand.

5. RESULTS

To evaluate the performance of the integrated modeling system presented here, we have run 3 sets of experiments. We first plot the size of our state equation representation as hybrid modes are exhaustively visited, and test the processing time saved through caching. Second, we compare the performance of the EKF and UKF on two nonlinear systems. Last, we show that we can track the behavior of large hybrid systems by automatically generating equations and using the UKF to estimate the system state.

5.1 Equation Representation Performance

By only selectively re-generating parts of the Hybrid Equation, we achieve substantial space savings over a re-generation of the equations during each mode change; which is demonstrated for two models. Figure 12 is a 4 state system with an algebraic loop. Figure 13 is a simplified section of NASAs ADAPT test bench which contains one battery and two loads (1) one NEMO, and (2) one resistive. Figure 18 shows a flat representation. The battery is augmented with internal switching junctions.

Figure 14 shows the size of the hybrid equation for each model as a function of the number of modes visited. The structure size is the sum of all terms (variables, operators, and parameters) and substitutions in the hybrid

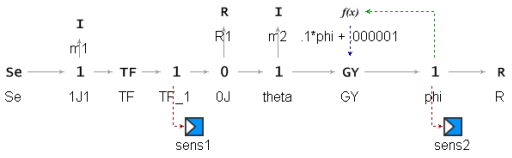

Figure 11: A nonlinear two-state pump model

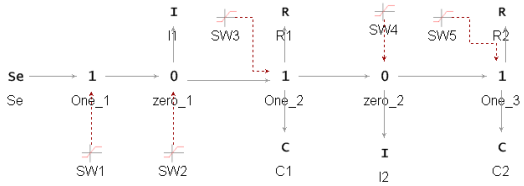

Figure 12: A HBG with 4 state variables

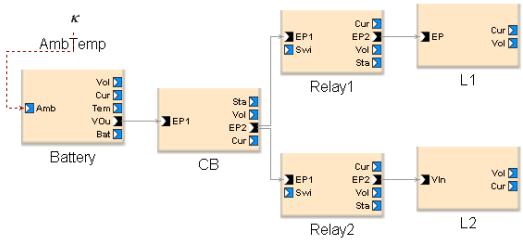

Figure 13: Simplified ADAPT battery

equation. Figure 14 shows that usually during a mode change, the majority of the equation components can be re-used, keeping the equation footprint light.

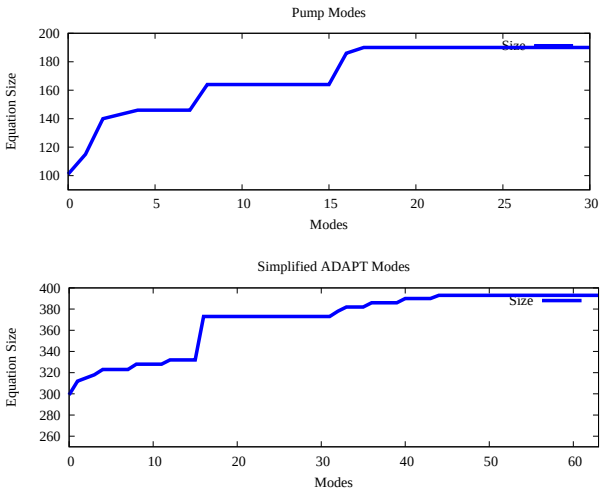

Figure 14: The size of the hybrid equations as the modes in figures 12 and 13 are enumerated

The caching performed during function evaluation saves a significant amount of processing time. Simulation used a fixed step forward Euler method for nu-

merical integration. Results are shown in table 1 for 3 models.

5.2 EKF and UKF Comparisons

To compare the performance of the EKF and UKF on the systems we are studying, we tracked the performance of each on several systems. The update equations for each were automatically derived from the shown models, except for the symbolic derivatives used by the EKF, which were generated by hand. All systems were simulated with additive process and measurement noise, and the filters were given an incorrect initial state estimate.

2-State Pump

Figure 11 shows a nonlinear 2-state pump model. The modulating function regulating GY adds nonlinearity to the system.

The UKF and EKF estimates the state of this model are compared in figure 15 . Both filters were initialized with incorrect m1 and m2 state estimates. As expected for a nonlinear model, the UFK converges on the correct state estimates faster than the EKF.

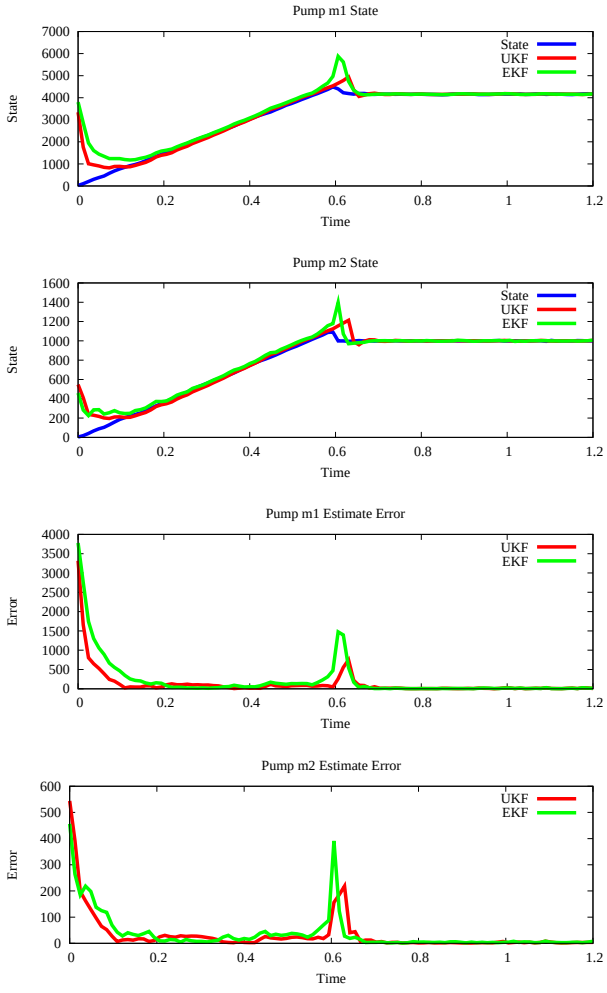

Figure 15: Pump state m1 and m2 estimates and errors

ADAPT Battery

Figure 13 shows a full schematic of the simplified nonlinear ADAPT battery model. The filters were run on the systems with 5 levels of Gaussian distributed measurement noise (.04, .0012, .004, .00012, and .0004.) The initial filter estimates of the state of the Load 2 charge (L2C) and the battery charge (CIs1) were incorrect.

The performance of the UKF and EKF in estimating each of these states at each measurement noise level is shown in Figure . The state estimates of the filter at each of the 5 levels of noise on each of these states are shown below.

CIs1s state behaves close to linearly over this time span, so we expect the UKF and EKF to perform very similarly, which is supported by the results seen in figure 15. The state of L2C does not behave linearly, so we expect the UKF to outperform the EKF, which is supported by the results shown in 17.

In all of the models evaluated, the UKF matches or outperforms the EKF, and we expect this to hold on other models derived from HBGs.

5.3 Scaling

Using the integrated system for modeling, equation generation, and state tracking described above, it is easy to model and track large hybrid systems. The system was run on a more complete version of the ADAPT DC testbench, shown in figure 19, again on with 5 different observation noise levels (.01, .003, .001, .0003, and .0001). The filter had an incorrect initial estimate of the charge on battery 1. At time 30, two loads on the system disconnect, and at time 60 reconnect. Figure 20 show the UKF state estimate at each of the 5 observation noise levels.

6. RELATED WORK AND CONCLUSION

The Advanced Diagnostics and Prognostics Testbed testbench used in section 5. was built to evaluate and compare diagnostic and fault detection techniques, and was the basis for the 2009 diagnostic competition. The Pro-Diagnose diagnosis system performed diagnosis very succesfully by compiling a bayesian system model to an arithmetic circuit, which operated very efficiently (Ricks & Mengshoel, 2009). (Grastien & John, 2009) also competed, but relied on a complete model of the test bench, including faulty behavior. The specifications of the ADAPT testbench were built into each of these systems. RODON is a commercial diagnostics reasoner which showed good results on the ADAPT testbench (BunusIsaksson Frey & Munker, 2009). RODON uses Rodelica, a modification to the general-purpose Modelica modeling language.

A number of Bond Graph modeling systems have been built. VirtualDynamics has developed a Bond Graph toolkit for Mathematica (Venuti, 2001); Matlab has a Bond Graph block library as well (Geitner, 2008). 20-sim is a software system dedicated to Bond Graph modeling and simulation (Broenink & Kleijn, 1999). However, there are very few systems designed to efficiently model Hybrid Bond Graphs (Daigle, 2007).

HyBrSim is an experimental Java modeling and simulation environment developed for modeling Hybrid Bond Graphs (Mosterman, 2002). HyBrSim can simulate a system by propagating state values forward through the

Table 1: Performance gains from caching

-	Structure size	Without caching (ms)	With caching (ms)	Gain
ADAPT Battery (figure 18)	299	23141	5859	74.7%
Pump (figure 11)	99	56078	7391	86.8%
4-state model (figure 12)	101	3812	1594	58.2%

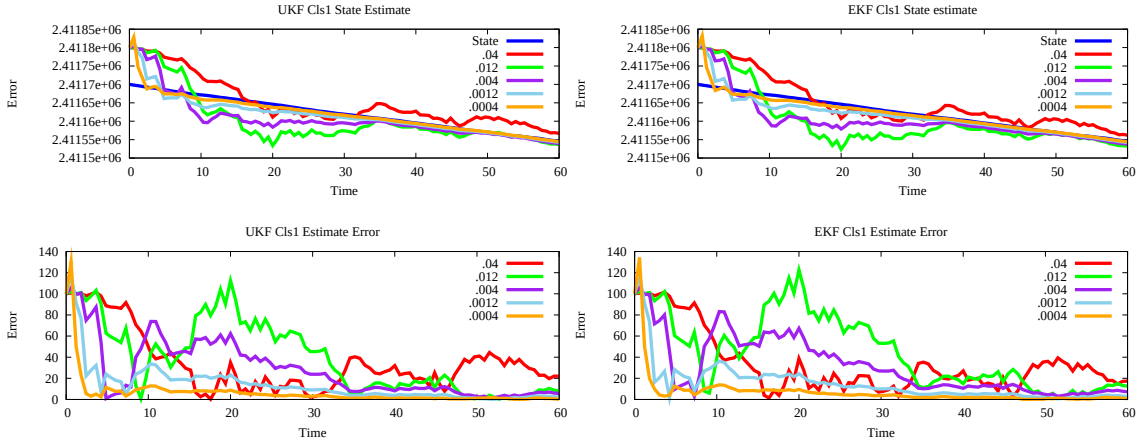

Figure 16: CIs1 state estimates and errors

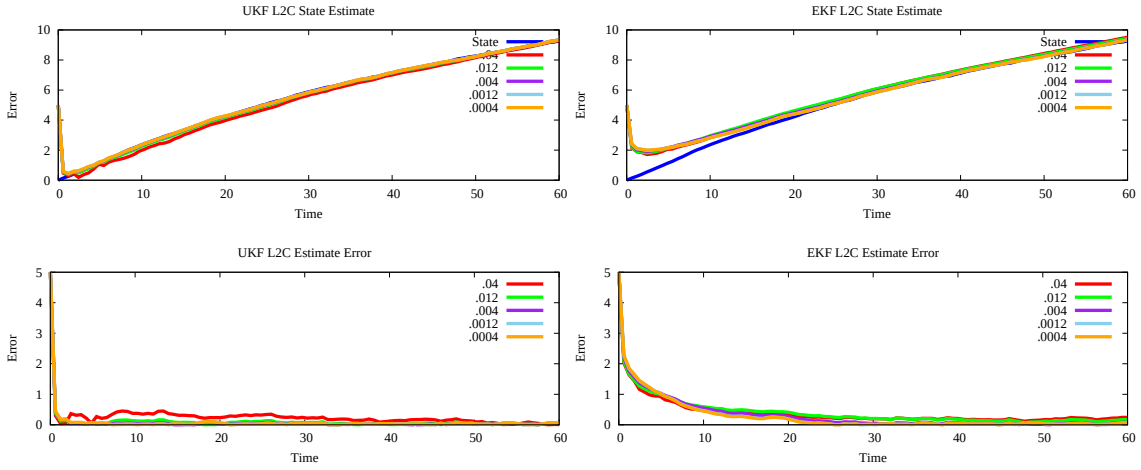

Figure 17: L2C state estimates and errors

all Bond Graph junctions, without explicit equation generation. HyBrSim can also export state equations as a set of DAEs solvable by a Differential Algebraic Equation (DAE) solver. Last, it can export equations for a mode as C++ code; however, modes exported to C++ must be pre-enumerated, limiting its use to small systems.

This work is an extension of the Block Diagram approach to simulation of Hybrid Bond Graphs (Daigle, 2007), whose system is simulated via Simulink. However, Block Diagrams do not attempt to resolve algebraic loops, and do not generate an explicit system of equations. The Block Diagram approach also does not explicitly cache the equations for commonly reached states as done here.

The system described here system stores HBG state and measurement equations in a fashion that minimizes the reconfiguration necessary on mode changes. We have shown that the size of the hybrid equation structure scales very well with the number of hybrid modes generated. By generating explicit equations, we are able to resolve many algebraic loops, and are able to simplify the equations to improve runtime performance.

We then showed that on the systems in which we are interested, the Unscented Kalman Filter matches or outperforms the Extended Kalman Filter. This is ideal for our purposes, as the UKF can be easily integrated into

our system; the UKF does not require the manual symbolic Jacobian generation required by the EKF. Combining these components, we have developed an automated system for modeling a Hybrid Bond Graph, generating its state equations, resolving many algebraic loops, efficiently regenerating equations on mode changes, and using these equations to estimate the state of the system given noisy states and observations.

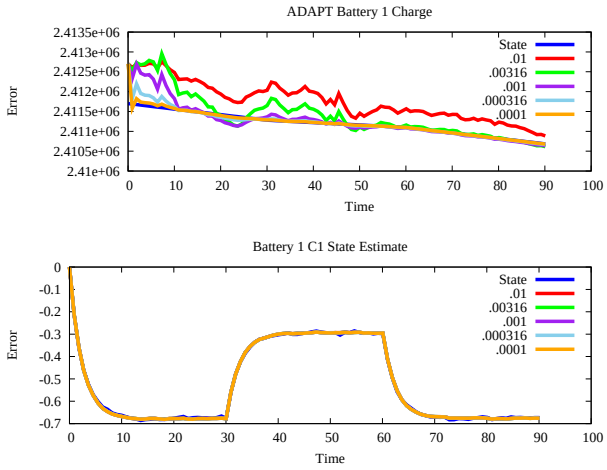

Figure 20: ADAPT battery and load charge estimates

REFERENCES

20-sim. 'Algebraic loops,' http://www.20sim.com/webhelp/editor/compiling/algebraicloops.htm; accessed April 19, 2010.

Agrawal, A., Karsai, G. & Ledeczi, A. (2003). An end-to-end domain-driven software development framework In *Conference on Object Oriented Programming Systems Languages and Applications.*

Alur, R., Courcoubetis, C. H. & Ho, P. H. (1993). Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems In *In Hybrid Systems* Vol 736.

Borutzky, W. & Cellier, F. (1996). Tearing in Bond Graphs with Dependent Storage Elements In *IMACS Multiconference*

Broenink, J. F. & Kleijn, C. (1999). Computer Aided Design of Mechatronic Systems Using 20-sim 3.0 In *Workshop on European Scientific and Industrial Collaboration*

Bunus, P., Isaksson, O., Frey, Beate. & Mnker, Burkhard. (2009). RODON - A Model-Based Diagnosis Approach for the DX Diagnostic Competition In *Proceedings of the 20th International Workshop on Principles of Diagnosis*

Cuijpers, P. & Reniers, M. (2005). Hybrid Process Algebra In *The Journal of Logic and Algebraic Programming*

Daigle, M.Roychoudhury, I.Biswas, G. & Koutsoukos, X. (2007). Efficient Simulation of Component-Based Hybrid Models Represented as Hybrid Bond Graphs In *Lecture Notes in Computer Science*

Dijk, J. V. & Breedveld, P. (1991). Simulation of System Models Containing Zero-Order Causal Paths–II Numerical Implications of Class 1 Zero-Order Causal Paths In *Journal of the Franklin Institute*

Gawthrop, P. J. & Smith, L. (1992). Causal Augmentation of Bond Graphs with Algebraic Loops In *Journal of the Franklin Institute*

Geitner, G. H. 'Bond graph add-on block library BG V.2.1.' http://www.mathworks.com/matlabcentral/fileexchange/11092-bond-graph-add-on-block-library-bg-v-2-1; accessed April 19,2010.

Grastien, A. & John, P. K. (2009). Wizards of Oz, description of the 2009 DXC entry In *International Workshop on Principles of Diagnosis (DX-09)*

Henzinger, T. A. (1996). The Theory of Hybrid Automata In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*

Hofbaur, M. W. & Williams, B. C (2002). Mode Estimation of Probabilistic Hybrid Systems In *Hybrid Systems: Computation and Control* Vol. 2289

Julier, S. J. & Uhlmann, J. K. (1997). A new extension of the Kalman filter to nonlinear systems

Karnopp, D. C. Margolis, D. L. & Rosenberg, R. C. (2006). System Dynamics: Modeling and Simulation of Mechatronic System

Lefebvre, T.Bruyninckx, H. & Schutter, J. D. (2004). Kalman filters for non-linear systems: a comparison of performance In *International Journal of Control*

Merwe, R. V. & Wan, E. A. (2001). The Square-Root Unscented Kalman Filter For State And Parameter-Estimation In *International Conference on Acoustics, Speech, and Signal Processing*

Mosterman, P. J. (2002). HYBRSIMa modelling and simulation environment for hybrid bond graphs In *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*

Mosterman, P. J. & Biswas, G. (1998). A theory of discontinuities in physical system models In *Journal of the Franklin Institute*

Ricks, B. W. & Mengshoel, O. J. (2009). Methods for Probabilistic Fault Diagnosis: An Electrical Power System Case Study In *Proceedings of the Annual Conference of the Prognostics and Health Management Society*

Romanenko, A. & Castro, J. A. (2003). The unscented filter as an alternative to the EKF for nonlinear state estimation: a simulation case study In *Computers and Chemical Engineering*

Roychoudhury, I., Daigle, M. J. , Biswas, G. & Koutsoukos, X. (2010). Efficient simulation of hybrid systems: A hybrid bond graph approach In *Simulation: Transactions of the Society of Modeling and Simulation International*

Stevens, M. 'Bayes++ Open Source Bayesian Filtering Classes' http://bayesclasses.sourceforge.net/Bayes++.html; accessed April 26,2010.

Venuti, N. (2001). Bond Graph Empowered by Mathematica In *Simulation series* Vol 33.

Welsh, G. & Bishop, G. (2006). An Introduction to the Kalman Filter.

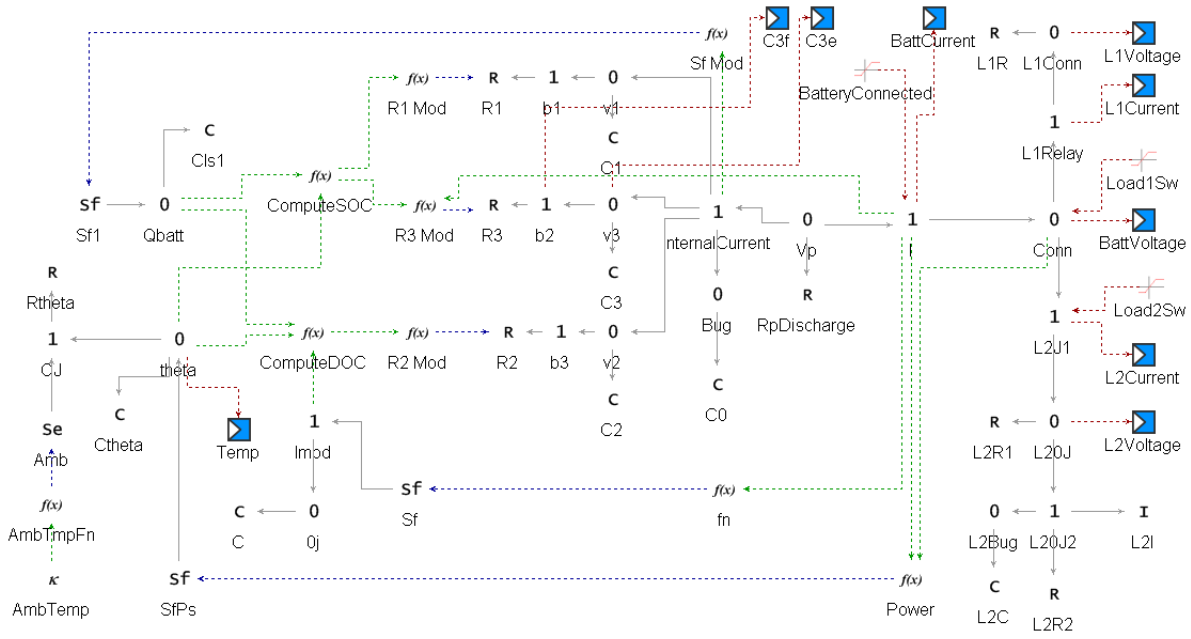

Figure 18: Flat schematic of the simplified ADAPT battery (figure 13)

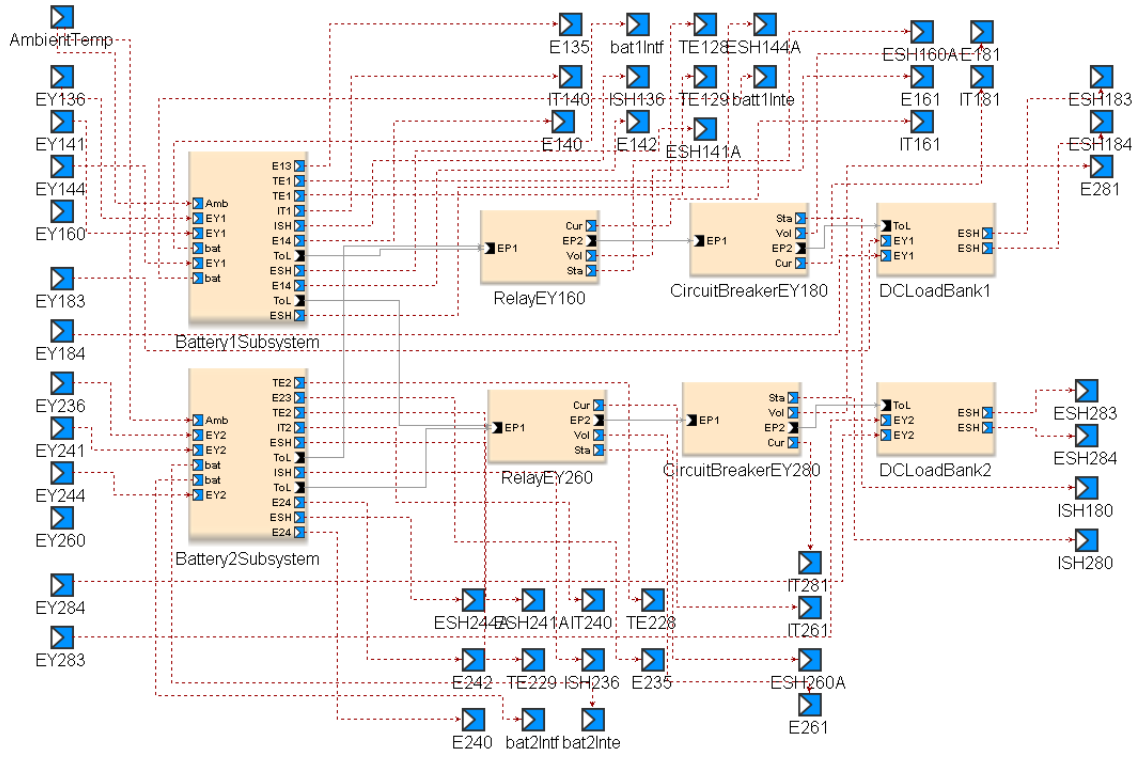

Figure 19: A high level view of the ADAPT DC configuration