Efficient diagnosis of hybrid systems using models of the supervisory controller

Sriram Narasimhan and Gautam Biswas
Dept. of EECS, Box 1679 Station B
Vanderbilt University, Nashville, TN 37235. USA.
nsriram,biswas@vuse.vanderbilt.edu

Abstract

This paper presents a model-based approach to diagnosis of hybrid systems. We have developed a combined qualitative-quantitative diagnosis scheme that uses hybrid models of the system and a model of the supervisory controller. By applying the supervisory controller model to diagnostic analysis we significantly cut down on the complexity in tracking behaviors, and in generating and refining hypotheses across discrete mode changes in the system behavior. We present the algorithms for hybrid diagnosis: hypotheses generation by back propagation, and hypotheses refinement by forward propagation and parameter estimation. Example scenarios demonstrate the effectiveness of this approach.

1. Introduction

Modern systems, such as aircraft and manufacturing plants, are complex and include supervisory control that switches

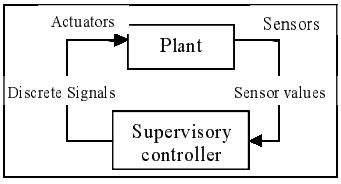

Fig. 1: Plant + Supervisory Controller

modes of behavior of the system to increase reliability and improve performance. Consider a plant with a supervisory controller in Fig. 1. Actuators directly controlled by the supervisory controller govern the system input, and sensors measure system variables that are used to estimate the system state.

The supervisory controller is a software program running on a digital processor. Unlike lower-level regulators in feedback loops, this controller is not tightly meshed with the continuous plant dynamics. It maintains pre-defined system functionality by generating discrete actions at predetermined points in time, and when pre-defined events occur in the plant dynamics. Variables values are directly sensed or computed from the measurements made on the system. The discrete actions of the controller change the input to the plant, or cause a reconfiguration of the plant. This changes the models that govern the continuous dynamics of the plant. We call such a system (plant + supervisory controller) a *hybrid* system, where the continuous behavior of the plant is interspersed by discrete changes in the plant models and the plant variable values.

The continuous dynamics of the plant are defined by differential and algebraic equations

$$\dot{x}(t) = f(x(t),u(t),q(t))$$
$$y(t) = g(x(t),u(t),q(t)),\ t >= 0,$$

where $x(t)$ is the continuous state vector, $u(t)$ is the input, $y(t)$ is the output vector, and $q(t)$ is the discrete *mode*. Mode changes in the plant are attributed to *controlled* and *autonomous* events [2]. The supervisory controller may change the discrete mode resulting in changes to the $u(t)$, $f(.)$, and $g(.)$ functions. This is called a *controlled event*. The discrete mode, $q(t)$, may also change when the state variables, $x(t)$, cross boundary values, which brings about a change in $f(.)$ and $g(.)$. These are called *autonomous events*, typically attributed to modeling abstractions [9].

We study the fault detection and isolation (FDI) problem in hybrid systems with supervisory controllers. System faults may be component, actuator, sensor, and controller faults. When the controller issues commands that generate behavior in conflict with the desired functionality, the controller may be said to be faulty. We do not deal with these kinds of faults and make the assumption that the commands issued by the controller are consistent with the desired functionality. This paper develops a model-based methodology that combines qualitative and quantitative reasoning techniques to perform parameterized fault isolation of plant component faults.

2. Modeling for Diagnosis

Model-based approaches to FDI in hybrid systems with supervisory control uses explicit models of the plant and controller to track system behavior and detect and analyze faults.

2.1 Controller Model

The primary model of the controller is implemented as a finite state machine (FSM). States of the FSM correspond to the states of the controller, which in turn define modes of the physical plant. The transitions determine the conditions for switching states and specify the discrete actuation signal

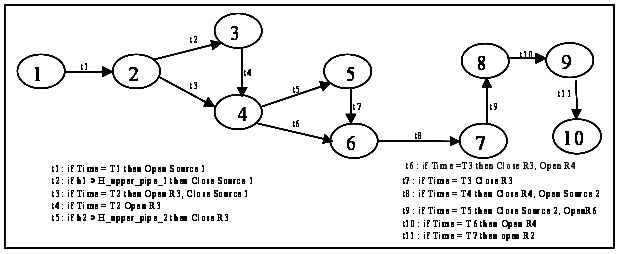

Fig. 2: Controller model for three-tank system

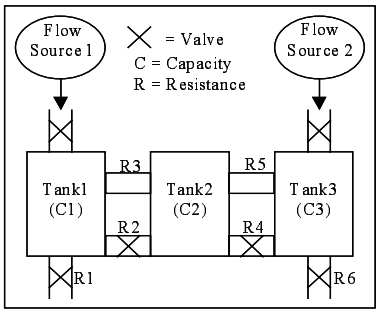

Fig. 3: Three-tank system

generated when the transition is executed. The transitions also describe the initialization of variables in the new state based on values from the previous state. They define a partial order of discrete events that may occur in the system. Fig. 2 illustrates a supervisory controller model for a connected three-tank system shown in Fig. 3. Tanks 1 and 3 can be filled and emptied independently. The supervisory controller directly controls the actuators that open and close the valves on the pipes. Autonomous events occur when liquid levels attain particular heights, which cause the intermediate connecting pipes to become active or inactive.

2.2 Plant Model

Our approach to modeling the plant involves building hybrid automata that model the continuous and discrete parts in a unified framework [1]. We use a *hybrid bond graph* modeling paradigm for the hybrid automata. Hybrid bond graphs include *controlled junctions* that facilitate the modeling of discrete mode transitions in system behavior [7]. This provides a compact representation of the system model across all its nominal modes of operation. Instead of pre-enumerating the bond graph for each mode, the hybrid bond graph uses individual junctions to model local mode transitions. The controlled 0- and 1- junctions represent idealized discrete switching elements that can turn the corresponding energy connection on and off. A finite state machine determines the ON/OFF physical state of the junctions. The transitions in this automaton depend on both control signals and internal variable values.

Fig. 4 illustrates the hybrid bond graph model of the three-tank system. The two flow sources into tanks 1 and 3 are indicated by Sf1 and Sf2, respectively, the tank capacities are shown as C1, C2, and C3, and the pipes are modeled by simple resistances. Valves are modeled by controlled junctions, which are shown in the figure as junctions with subscripts. The control signals for turning these junctions on and off are generated by the finite state automata shown in Fig. 4. The toggling signal for the automata comes directly from the supervisory controller. For autonomous transitions in the system, also modeled by controlled junctions, the transition conditions computed from system variables (e.g., see the transition conditions for junctions 4 and 6). A mode in the system is defined by the state of the eight controlled junctions in the hybrid bond graph model. Therefore, theoretically the system can be in 256 different modes.

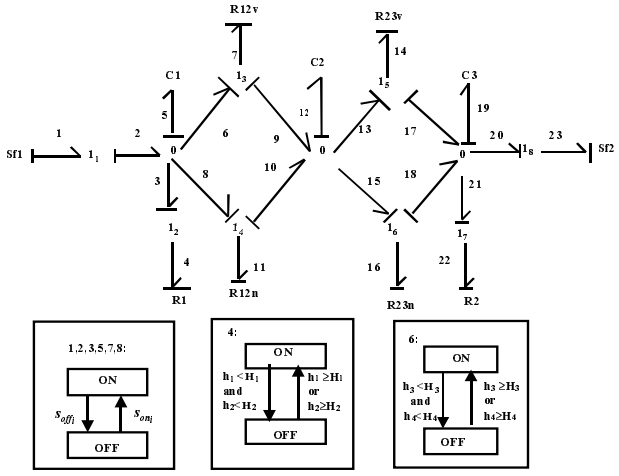

Figure 4: Hybrid Bond Graph for Three-tank system

State equations and temporal causal graphs (TCGs) can be systematically derived from the bond graph representation of the system [8,14]. When mode changes occur, the appropriate controlled junctions are toggled, a new bond graph model is derived corresponding to the current system configuration, and a new state equation model and TCG can be derived for this mode. The state equations simulate system behavior, and they along with the temporal causal graphs constitute our diagnosis models.

3. Our Methodology for Hybrid Diagnosis

Our model-based approach (Fig.5) requires the use of a hybrid observer to track normal system behavior, a fault detection mechanism, and a fault isolation unit. The observer uses a quantitative hybrid model of the plant to follow the continuous dynamics of the plant in a continuous operating region, and identify discrete mode changes to make the switch from the current to the new mode of system operation by updating the plant model and its continuous system vector. We have adopted a combination of a hybrid automata and Kalman filtering approach to design our hybrid observer [12]. Small differences, attributed to minor imperfections in the model and noise in the measurements, are compensated for in the observer mechanism. Significant differences that the observer cannot compensate for cause the fault detection

unit to signal the presence of a fault in the system. In most cases, noise and the complexity of the signals, and the imperfections in the sensors and the system model, require the use of sophisticated signal analysis techniques to detect discrepancies in the observed measurements [5]. The fault isolation unit generates candidate faults and refines them with the hybrid model and measurements from the system.

Algorithm 1 gives a high level overview of the computational mechanism we employ for model-based diagnosis of hybrid systems. The following information is assumed to be available to all modules of the diagnosis system: (i) HBG - Hybrid Bond Graph model of the system, (ii) FSA_{system} - Hybrid Automaton model of the system, (iii) $FSM_{controller}$ - FSM model of controller, (iv) Σ_A - all possible autonomous events in the system, (v) U - inputs to the system, (vi) Y – measurements from the system, and (vii) $Parameters_{nominal}$ - nominal values of all parameters in the system. M and X refer to the discrete mode and, the continuous system state,

```
MODULE DIAGNOSE(M_initial, X_initial)
// Observe the system until a fault is detected
<Stack_M, Y_estimated> = OBSERVER(M_initial, X_initial);
// Convert the quantitative residuals to qualitative values
QualResidual_current = SIGNAL_TO_SYMBOL(Y, Y_estimated);
// Back propagate across modes to identify fault candidates
BackHorizon = 2;
List_candidates = HYBRID_BACK_PROP(Stack_M, QualResidual_current, BackHo
rizon);
// Forward propagate across modes to isolate the fault
List_candidates = HYBRID_FAULT_OBSERVER(List_candidates, Y_estimated);
END DIAGNOSE
```

Algorithm 1: Diagnosis Module

respectively. Our focus in this paper is on the fault isolation algorithms for hybrid diagnosis.

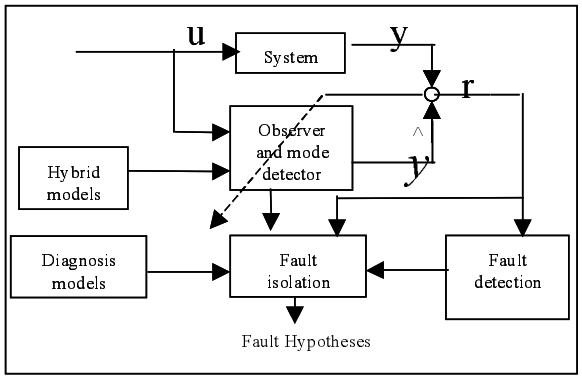

Fig. 5: Diagnosis System Architecture

4. Fault Isolation

The type of plant model employed determines the scheme to be employed for fault isolation. Discrete event approaches pre-compile fault models and fault trajectories into finite state automata for tracking nominal and faulty system behavior [4,15]. Traditional fault observer schemes in the continuous domain use structured and directional residual approaches. An algebraic function transforms the raw residual generated by the observer scheme into a form where there is a one-to-one mapping between hypothesized faults and the observed residual vectors [13]. These techniques mainly apply to linear systems, though recently there have been some applications to non-linear dynamic systems [3]. Extending these continuous methodologies to hybrid systems becomes intractable because the residual transformation functions have to be pre-computed for all modes of operation. Further, when faults occur, predicting the true system mode in itself becomes a challenging task. The fault isolation problem becomes even more complex, when the fault is detected in a later mode of operation. The predicted mode sequence may no longer be the true mode sequence the system goes through after the occurrence of the fault. Additional methods have to be introduced for detecting mode transitions, switching the system model when such transitions occur, and correctly initializing the system state so that the fault observers perform correctly. Mode changes introduce discrete effects and transients making it difficult to analyze fault transients across mode changes. Therefore, extending continuous FDI schemes to hybrid systems is a non-trivial task.

Our approach to fault isolation involves *hypotheses generation* and *hypotheses refinement*. We use a qualitative framework for hypotheses generation, and a combined qualitative-quantitative approach for hypothesis refinement. Qualitative analysis requires a symbol generation methodology (SIGNAL_TO_SYMBOL) presented in [5]. The use of qualitative methods to initiate hypotheses generation and refinement overcomes a number of the limitations of quantitative schemes, such as convergence and accuracy problems in dealing with complex non-linearities and lack of precision of parameter values in system models. We use a combination of extended versions of the backward and forward propagation algorithms [8] that operate on the TCGs representing the system dynamics to generate fault hypotheses and predict their consequences over time as fault signatures. When used in conjunction with the supervisory controller model, this can provide a computationally simpler mechanism to track faulty behavior across discrete mode changes. The qualitative reasoning scheme is fast and effective, but it has limited discriminatory ability. To overcome this, we use parameter estimation schemes that involve the initiation of a fault observer for every hypothesized fault [6]. The use of a reduced set of individual observers mitigates computational convergence and complexity problems of quantitative analysis methods. In addition, the quantitative estimate provides detailed information about the extent of degradation in faulty components.

4.1.1 Hypotheses Generation

Back propagation for initial hypotheses generation has to be

performed across modes the system has traversed through because the fault may have occurred in a previous mode but the manifestations are not seen until a later mode. For example, this happens when none of the observed variables are

```
MODULE HYBRID_BACK_PROP(Stack_M, QualR_i, BackHorizon)
 // Generate candidates in each mode in the mode trajectory
   <M_current,Time_current> = Pop(Stack_M);
   TCG_current = GET_TCG(HBG, M_current);
 // Back propagate in selected mode for candidates in the mode
   F_current = CONTINUOUS_BACK_PROP(TCG_current, QualR_i);
   Add(List_candidates,<M_current,Time_current,F_current,1>);
   Count = 0;
 // Go back in the mode horizon upto BackHorizon number of nodes
   While (Count < BackHorizon)
 // Select next mode in mode trajectory and calculate TCG
     <M_next,Time_next> = Pop(Stack_M);
     TCG_next = GET_TCG(HBG, M_next);
 // Propagate qualitative deviations across modes
     QualR_next=
      BACK_PROP_ACROSS_MODES(M_current , M_next , QualR_i)
 // Back propagate in selected mode for candidates in the mode
     F_next = CONTINUOUS_BACK_PROP (TCG_next, QualR_next);
     Add(List_candidates,<M_next,Time_next,F_next,1>);
     M_current = M_next;
   End While
   Return(List_candidates);

END MODULE
```

Algorithm 2: Hybrid Back Propagation

affected in the mode in which the fault occurs. The problem is that once a fault occurs the predicted mode sequence of the observer may no longer be correct, and a worst-case analysis may require considering all possible modes in generating fault hypotheses. However, the assumption that the controller model is correct implies that *the observer predicted the correct mode sequence till the fault occurred. Therefore, the mode in which the fault occurred must be in the predicted trajectory of the observer*. Back propagation is applied to each of the modes in the mode trajectory predicted by the observer. This ensures that the true fault hypothesis, which includes the fault and the mode in which the fault occurred (<mode, fault>) will be included in our initial hypothesis set. As a reasonable heuristic, we further limit our search by looking back only *k* modes (BackHorizon). This is based on the assumption that the effects of a fault must manifest within *k* mode changes.

Algorithm 2 presents the back propagation algorithm extended to handle hybrid systems including the model of the controller. In the algorithm, **Pop** removes and returns the top element of a stack. **GET_TCG** returns the TCG of the system in the current mode. **BACK_PROP_ACROSS_MODES** specifies qualitative changes in variables across modes. **CONTINUOUS_BACK_PROP** back propagates the values of a discrepancy through the current TCG [8]. **Add** adds an element to a list.

4.1.2 Hypotheses Refinement

Hypotheses refinement first applies a qualitative forward propagation and then quantitative parameter estimation. In the qualitative step, for each hypothesized fault candidate we forward propagate the qualitative effects of the fault through the TCG. This is done in the mode that the fault occurred. If the qualitative predictions do not match the observations in the mode, then we have one of two possible situations. Either the fault candidate is invalid or a mode change has occurred in the system. To take into account both possibilities, *all possible mode changes from the current mode are hypothesized. This involves using the model of the controller plus additional information from the system to limit the number of possible mode transitions*. For each of the hypothesized modes, forward propagation is continued to come up with qualitative predictions in the new mode. If the predictions still do not match the observations, then the mode candidate is dropped. For a fault hypothesis, if all mode candidates are dropped, then the fault candidate is dropped since none of the possible mode change sequences were sufficient to make the predictions for the fault candidate and the observations consistent.

Algorithm 3 presents the hybrid forward propagation algo-

```
MODULE HYBRID_FAULT_OBSERVER(List_candidates,Y_estimated)
// Keep isolating till the fault set size falls to the desired size
While (COUNT(List_Candidates) > θ_desired)
 // If the number of candidates is a manageable number,
start parameter estimation
 If (COUNT(List_candidates) < θ_quant)
  Fork
    PARAMETER_ESTIMATION(List_candidates);
  End Fork
 End If
 // Check for consistency for each of the candidates
 For All <M_i,Time_i,F_i,Level_i> ε List_candidates
 // Generate the TCG, predict signatures and perform
progressive monitoring
 TCG_i = GET_TCG(HBG, M_i);
 Signature_i = CONTINUOUS_FOR_PROP(TCG_i,F_i);
 QualResidual_i = SIGNAL_TO_SYMBOL(Y(Time_i),Y_estimated);
 Consistent=
PROGRESSIVE_MONITORING(Signature_i,QualResidual_i,Ti
me_i+Time_increment);
 // If inconsistency has persisted across few mode drop
candidate, else hypothesize mode change
 If (Consistent = FALSE)
  If (Level_i >θ_consistency)
   Remove(List_candidates, <M_i,Time_i,F_i,Level_i>);
  Else
    <List_M> = CALCULATE_NEXT_MODES(FSA_controller,M_i,
Time_i+Time_increment,Σ_A);
     For All M_j ε List_M
      Add(List_candidates,<M_j,Time_i+Time_increment,F_i,Level_i+1>);
     End For
   End If
 End If
End For
// Increment time
Time_increment = Time_increment + Time_step
End While
Return List_candidates;
END MODULE
```

Algorithm 3:Hybrid Fault Observer

```
MODULE PARAMETER_ESTIMATION(List_candidates)
LevelOfConfidence = 99;
// Start an estimator for each candidate
For All <M_i,Time_i,F_i,Level_i> ε List_candidates
  // Generate the symbolic state equations
 StateEquations_Mi =
GET_STATE_EQUATIONS_WITH_FAULT(HBG, M_i,F_i);
  // Calculate error while estimating parameter
  PredictionError =
SYSTEM_ID_ESTIMATION(M_i,StateEquations_Mi,Time_i,U,Y);
  // Check if Prediction error is statistically close to 0
  If (STAT_TEST(PredictionError,LevelOfConfidence) = TRUE)
    Remove(List_candidates, <M_i,Time_i,F_i,Level_i>);
  End If
End For
END MODULE
```

Algorithm 4: Parameter Estimation

rithm. In the algorithm, **COUNT** counts the number of elements in a list. $\theta_{desired}$ is the desired size of candidate set. If the number of candidates falls below θ_{quant}, we start the parameter estimation procedure. **Fork** starts another task in parallel. **GET_TCG** was described in the last section. **CONTINUOUS_FOR_PROP** generates the signatures of a fault in any single mode [8]. **PROGRESSIVE_MONITORING** checks if the signatures match the actual observations over time [8]. $\theta_{consistency}$ is the number of modes we continue progressive monitoring in case of a discrepancy. **Remove** removes an element from a list. **CALCULATE_NEXT_MODES** identifies all modes that the system may transition using the controller model and the possible autonomous events in the current mode. Discrepancies in progressive monitoring are justified by mode changes, and discrepancies in the new mode are justified by further mode changes. This nesting is done for fixed number of steps.

The limited discriminative ability of the qualitative analysis [6] requires us to switch to quantitative parameter estimation to uniquely isolate the fault. This approach works within a single continuous mode. If there is a mode switch during the estimation process, we face two problems. (i) How we do know that a mode change has occurred, and if a mode change has occurred what is the new mode? It depends on the estimated parameter value. (ii) Even if we can identify the new mode of the system, how do we continue parameter estimation since the state space model for the new mode is different? In our work we perform parameter estimation in a single mode. When a mode change occurs we switch the mode and start the estimation afresh.

Algorithm 4 presents the parameter estimation procedure. **GET_STATE_EQUATIONS_WITH_FAULT** returns the state equations in the current mode with all but the faulty parameter substituted with nominal parameter values [6]. **SYSTEM_ID_ESTIMATION** procedure tries to estimate the faulty parameter value from the state equations, U, and Y using system id techniques [6]. **STAT_TEST** is a statistical test to check if the PredictionError is statistically close to 0. The **Remove** procedure removes an element from a list.

5. Experiments

We have studied the effectiveness of our hybrid algorithms by running extensive experiments on a three-tank system test-bed in our Modeling and Analysis of Complex Systems (MACS) laboratory. The controller model for the three-tank system (Fig. 3) is illustrated in Fig 2. Four fault scenarios are illustrated below.

In all our experiments, system behavior is tracked with a sampling rate of 0.1 second. In the first experiment, our measurements are the height of liquid in tank 3, i.e., h_3 and

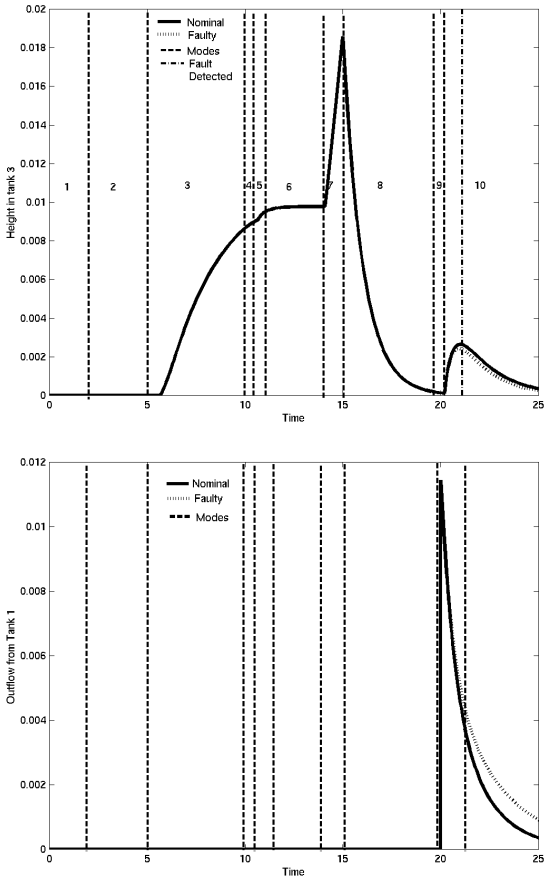

Figure 6: Experiment 1 – fault C2- introduced in mode 9, but detected in mode 10. h_3 and f_{R1} are measured. Mode changes in the system are shown by vertical lines.

out flow from pipe 1, i.e., f_{R1}. Fig. 6 displays the results of tracking the level of fluid in tank 3 using the observer algorithm. We introduced a fault in the system at time 20 (mode 9) but the fault detection unit detects the fault only at time 21 (mode 10). At this point the mode trajectory seen so far (1,2,3,4,5,6,7,8,9,10) is saved and the fault isolation algorithms are invoked. The hybrid back propagation algorithm is executed to identify the initial candidates. Hypothesis generation using the hybrid backprop algorithm in mode 10

generation using the backprop algorithm in mode 10 on discrepancy h3+ yields the candidates <10, C3- R6+ R4+ C2- R2- C1- R1+ > (10 indicates the mode number). However, since the fault could have occurred earlier, the extended backprop algorithm generates faults from past modes in the mode trajectory. In order to reduce the complexity we restrict ourselves to only two previous modes (8 and 9). Performing back propagation in each of these two modes yields more candidates <9, C3- R6+ R4+ C2- > and <8, C2- R4+>.

After initial candidate generation, the next step is hypothesis refinement by generating fault signatures and then tracking individual fault candidates by progressive monitoring. For this experiment, the 2nd order signatures for the fault hypotheses are listed in Table 1.

Fault	Tank 3 Height	Tank 1 Outflow
C1-	00-	+-+
C2-	0-+	0+-
C3-	-+-	00+
R1+	000	+-+
R2-	00-	0-+
R4-	0-+	00-
R6+	-+-	000

For faults C3- and R6+, the signatures imply a discontinuous change for the height in tank 3. Since this is not observed, these candidates are dropped. Similarly, C1- and R1- are dropped, since they predict a discontinuous change for the outflow from tank 1. The outflow measurement signature for R2- (0 - +) does not match the actual outflow. For a purely continuous system, this fault hypothesis would be eliminated at this stage. However, when tracking hybrid behavior this cannot be done. Instead the algorithm assumes a mode change could have occurred, and it generates all possible transitions that are feasible, given the direction of change of variables, the autonomous transition definitions, and the predictions of the supervisory controller model Progressive monitoring is continued, but in each of the new hypothesized modes the outflow retains the same signature for fault R2-. Hypothesizing more autonomous changes would result in the same signature and hence R2- is dropped when we reach the predefined level of recursion for hypothesizing additional mode changes. The remaining candidates <10, C2- R4+>, <9, C2- R4+> and <8, C2- R4+>, cannot be distinguished further by qualitative analysis, therefore, the quantitative parameter estimation algorithm is invoked. A fault observer is initiated for each of the remaining candidates. Estimation and a statistical check for convergence shows that that <8,C2-> is the true fault (see Fig. 6). For all other fault candidates the parameter estimation diverges.

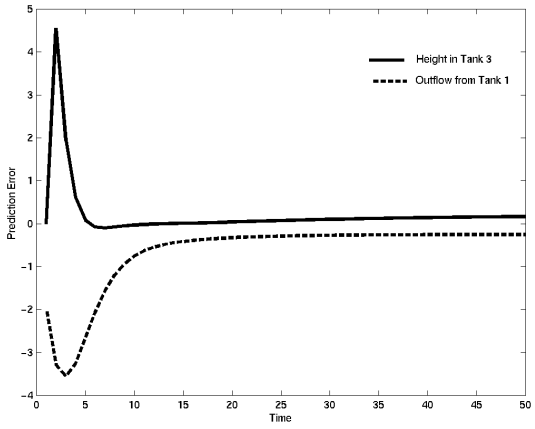

Figure 7: <8, C2-> identified as the true candidate by parameter estimation and statistical convergence

The second experiment illustrates the importance of parameter estimation. The measure variable is the height of tank 3. In this case, the hypothesis generation procedure generates the same candidate, C2- in mode 5 and mode 6. The fault is detected in mode 6 (11.2s). The qualitative signatures for the two faults are identical (similar to table 1), and Fig. 7 illustrates the measured height in tank 3 for the two faults. The solid curve represents the behavior of the system under nominal conditions. The dotted and dashed curves represents the two faulty behaviors Parameter estimation, however, establishes <5, C2-> as the true fault.

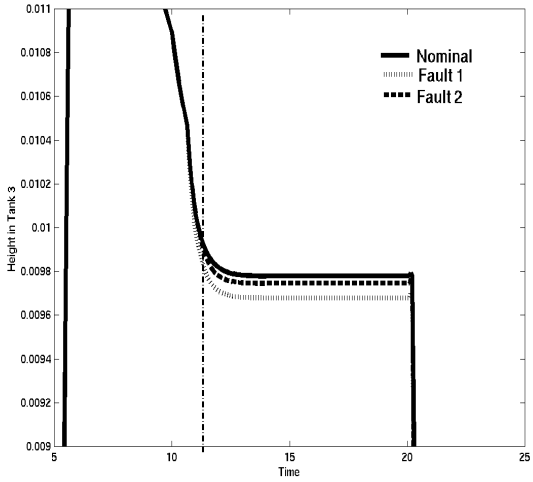

Figure 8: Experiment 2: Measurement is the height of liquid in tank 3

The third experiment shows that the fault isolation procedure works even if the observer predicts an incorrect mode

trajectory after the fault occurs. For the case in which the abrupt fault, C1- occurs in mode 2 (t=4s), and the outflow from tank 1, h_1 is the only measured variable, the fault is not detected until mode 8 (t=20.5s), when the deviation in out flow from tank 3 becomes significant (see Fig. 9). Fig. 9 also shows the height in tank 1 (not measured) to illustrate that the fault occurred earlier. An autonomous event corresponding to flow in the upper connecting pipe occurs after t=4s, but the observer is unable to predict this event since the height is estimated assuming nominal conditions. The observer predicts the opening of the pipe connecting tanks 1 and 2 as the next controlled event. Therefore, the actual mode sequence to the point when the failure is detected is 1, 2, 3, 4, … 8 (see Fig. 2), whereas the predicted mode sequence is 1, 2, 4, …, 8, i.e., the observed mode sequence is different from the actual mode sequence. Since the mode in which the fault occurs is part of the predicted observer trajectory, BACK_PROP adds the <2,C1-> fault candidate to the hypothesis list. Fault candidates from modes 4 through 8 are also generated, but they are all eliminated very early in the hypothesis refinement process described in the first experiment When the parameter estimation procedure is invoked, the system succeeds in isolating the true fault.

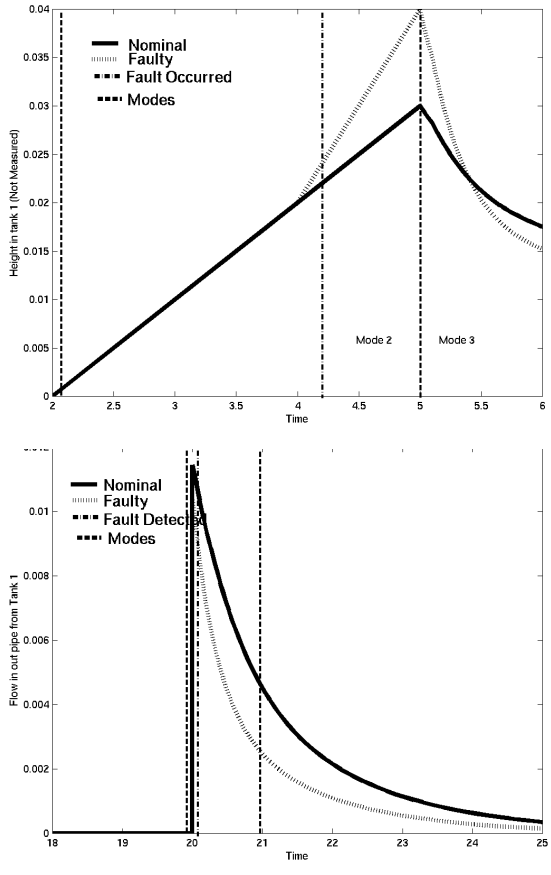

Figure 9: Experiment 3 – The true fault is isolated even though the observer misses a mode transition

In the fourth experiment (Fig. 10), we illustrate that even if the fault cause our observer to go through modes that system does not actually go through, the hypothesized faults in these modes are eventually dropped in the fault isolation phase. We introduce a fault (C1 +) in the system at time 4 (mode 2). Again we measure only the outflow from tank 1. Fig.10 also shows the height in tank 1 to indicate that trajectory is affected significantly even though the difference between actual and observed mode sequences is one mode (3).

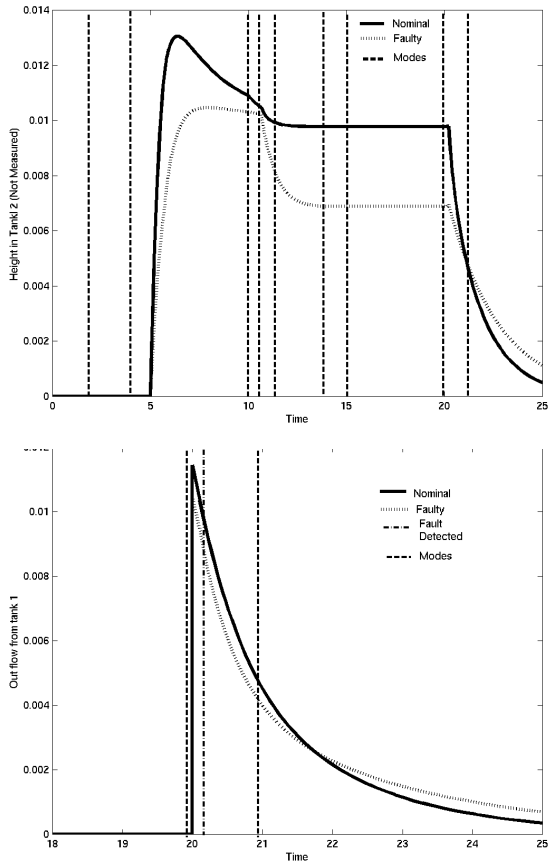

Figure 10: Experiment 4 -- Correct fault isolation occurs even though the observer predicts additional modes after the fault occurrence

The actual system goes through the mode sequence (1,2,4,5,6,7,8,9,10) but the observer predicts the mode sequence (1,2,3,4,5,6,7,8,9,10). The sudden increase in the capacity of the tank 1 causes the height in tank 1 to drop and due to this mode 3 (corresponding to the height in tank 1 crossing the upper connecting pipe) never occurs. Since the observer is not aware of the fault, it uses the nominal value of the height and predicts the occurrence of the autonomous events and hence goes through mode 3. At the time of fault detection, back propagation selects candidates from mode 3 also but these candidates get eliminated either in the hybrid forward propagation or parameter estimation steps.

6. Conclusions

In this paper, we extend previous work in FDI of hybrid systems [10,11], which assume the mode sequence in behavior evolution is known even after faults occur in the system. Relaxing this assumption could create an exponential blow up in the number of mode transitions that need to be considered after fault occurrences (Number of trajectories that have to be considered is the number of modes in the system x the number of potential faults). In previous work (e.g., [15,16]) this is avoided by assuming the fault is detected in the mode in which it occurs, or it is known when exactly the effect of a fault will be observed. In either case the controller model is then used to pre-compile the fault trajectories of the system. In this work, since we are dealing with a mostly continuous system, pre-enumerating fault trajectories is computationally intractable for the reasons mentioned above. To avoid the intractability problem, like [15,16] and other work we assume that have a correct model of the supervisory controller. We do not pre-enumerate fault trajectories, but by incorporating the supervisory controller model into our approach, we significantly cuts down on the search space that we explore during the back propagation (hypothesis generation) step of our FDI algorithm. This is based on the observation that the mode in which the fault occurred must lie in the trajectory hypothesized by the hybrid observer. Similarly, the controller model helps cut down search during forward propagation to generate predicted behaviors.

A significant component of our hybrid diagnosis system involved the design of the hybrid observer for tracking continuous behaviors across mode changes. An efficient scheme that compiles our hybrid bond graphs [7] into hybrid automata [1] was described in [12]. Future work will involve building observers that can perform mode identification based only on measurements, under nominal and faulty conditions. This will permit us to identify actuator, sensor and plant faults. We also need to extend our work to derive more robust online parameter estimation techniques. The observer will then be integrated with our qualitative and quantitative diagnosis algorithms for fault detection and isolation in hybrid systems, and also for fault-adaptive control of complex systems.

References

1. Alur, R., et al., 1993. *Hybrid Automata: an algorithmic approach to the specification and verification of hybrid systems,* in R.L. Grossman, et al, eds., Lecture Notes in Computer Science, Springer, Berlin, 736, 209-229.
2. Branicky, M.S., V. Borkar, S. Mitter, 1994. *A Unified Framework for Hybrid Control: Background, Model, and Theory*, Proc. 33rd IEEE Conf. on Decision and Control, FL, Paper No. LIDS-P-2239.
3. Garcia, E.A., Frank, P.M. 1997. *Deterministic nonlinear observer based approaches to fault diagnosis—A survey*, Control Engg. Practice, Vol. 5 (5), 663-670.
4. Lunze J., 1999. *A timed discrete-event abstraction of continuous-variable systems*. Intl. Jour. of Control, 72, 1147-1164.
5. Manders E.J., P.J. Mosterman, G. Biswas, 1999. *Signal to symbol transformation techniques for robust diagnosis in TRANSCEND,* Tenth Intl. Workshop on Principles of Diagnosis, Loch Awe, Scotland, 155-165.
6. Manders E.J., et al., 2000. *A combined qualitative quantitative approach for efficient fault isolation in complex dynamic systems*, SAFEPROCESS, 512-517.
7. Mosterman P.J., G. Biswas, 1998. *A theory of discontinuities in physical system models*, Jour. of the Franklin Inst., 335B, 401-439.
8. Mosterman P.J., Biswas G., 1999. *Diagnosis of Continuous Valued Systems in Transient Operating Regions*. IEEE Trans. on Systems, Man and Cybernetics: 29, 554-565.
9. Mosterman P.J., Biswas G., 2000. *Towards Procedures for Systematically Deriving Hybrid Models of Complex Systems*. Third Intl. Wkshp. on Hybrid Systems: (HSCC'00), 324-337.
10. McIlraith S., et al., 2000. *Towards Diagnosing Hybrid Systems*. Third Intl. Wkshp. on Hybrid Systems: (HSCC'00), 282-295.
11. Narasimhan, S., et al., 2000. *Fault Isolation in Hybrid Systems Combining Model-based Diagnosis and Signal Processing*, SAFEPROCESS, 1074-1079.
12. Narasimhan, S., et al., 2000. *Building Observers to Handle Fault Isolation and Control Problems in Hybrid Systems*, Proc. IEEE Intl. Conf. on SMC, Nashville, TN, 2393-2398.
13. Patton, R.J., Frank, P.M., and Clark, R.N. (eds.), 2000. *Issues of Fault Diagnosis for Dynamic Systems,* Springer-Verlag, London, U.K.
14. Rosenberg, R.C., Karnopp, D. 1983. *Introduction to Physical System Dynamics,* McGraw Hill, New York.
15. Sampath M., et al., 1996. *Failure Diagnosis using Discrete Event Models*. IEEE Trans. On Control Systems Tech., 4(2), 105-124.
16. Sampath, M, et al, 2000. *Combining Qualitative and Quantitative Reasoning: A Hybrid Approach to failure Diagnosis of Industrial Systems*. SAFEPROCESS, 494-501.

Acknowledgements. The research is supported by grants from DARPA SEC under contract number #F33615-99-C-3611 and Xerox PARC. The authors benefited a lot from discussions with Prof. Gabor Karsai at Vanderbilt University and Dr. Feng Zhao at Xerox PARC.