# Diagnosing Hybrid Cyber-Physical Systems using State-Space Models and Satisfiability Modulo Theory

**Alexander Diedrich**[1] and **Oliver Niggemann**[2]
[1]Fraunhofer IOSB-INA, Lemgo, Germany
[2]Institute Industrial IT, Lemgo, Germany
e-mail: alexander.diedrich@iosb-ina.fraunhofer.de
oliver.niggemann@hs-owl.de

## Abstract

Diagnosing faults in large cyber-physical production systems is hard and often done manually. In this paper we present an approach to leverage methods from the fault detection and isolation community as well as model-based diagnosis to diagnose faults. Given a model of the production system we capture its dynamic behaviour with a state-space model. Then we translate the state-space model into satisfiability theory modulo linear arithmetic over reals. This translation converts numerical information in symbolic logic. These symbols can be used to diagnose faults with Reiter's diagnosis algorithm.
We use a four-tank model as a demonstration use case. Under the assumption that the use-case is fully-observable (i.e. all components except the water tanks can be observed) our methodology detects all injected faults.



## 1 Introduction

For operators of large industrial cyber-physical production systems (CPPS) it is often a hard task to precisely detect, identify, and isolate technical faults [1]. This is especially the case in large production plants in the process industry or in pharmacological processes which often extend over significant physical distances, consist of highly interdependent components, and involve many parallel paths in the form of pipe and valve networks. The correct behaviour of a biological reactor, for example, depends on the exact amount of different ingredients, their pressure, their temperature, their viscosity, the ambient temperature and pressure, or other important process values. Further, subsequent processes heavily depend on the correct amount, quality, and time of discharge from the biological reactor. Since these systems are interdependent it is hard for human operators to physically locate the root-cause of a fault. For example, a valve might break and block the flow of some liquid into the reactor. Due to this blocked flow, the pressure and temperature within the reactor might change and degrade the material by changing its viscosity. The degraded material might then go on into a stamping and forming process. Consequently, due to the changed viscosity, the presses will register changed pressures within their control systems. For a human operator, all these components will at some point sound an alarm indicating that parameters are outside their normal operating conditions. In modern industrial plants, the amount of these alarms can quickly overwhelm an operator and thus keep him from finding the faulty component that caused the fault [2].
Identifying and isolating faults in large industrial plants can take precious time which can quickly lead to a significant deterioration of the produced material or even physical destruction of involved components. For operators of these plants this can lead to high costs. These high costs occur even in smaller scale enterprises, due to low quality output, costs to locate and repair the broken component, and costs to ramp up production again after the fault.
In the presented approach we attempt to perform fault detection and isolation (FDI) through model-based diagnosis over satisfiability modulo theory (SMT). For this, a model of the physical process is created manually. Industrial cyber-physical production systems are dynamic because their parameters change over time and contain hybrid signals which can be either binary or continuous. Here we use state-space models to capture the dynamic behaviour of hybrid and cyber-physical production systems. We limit our use case to a multiple tanks model. The behaviour of the tanks is modelled using differential equations, while the connection between components is modelled using predicate logic. A set of piecewise functions translates the state space model into satisfiability theory modulo linear arithmetic over reals ($\mathcal{LRA}$). Through the translations it is possible to leverage standard model-based diagnosis algorithms, once developed to diagnose binary circuits, to diagnose hybrid cyber-physical production systems. The outcome of this translation is a set of tuples which states for each component whether or not it is currently faulty $< comp, status >$. This tuple is converted into predicate logic and combined with the connection model of the plant. We employ Reiter's diagnosis lattice [3] to find the minimum cardinality diagnosis and thus isolate the fault that caused the production plant to fail.
In this work we demonstrate how our described approach can be used with a multiple tanks model, how we translate the numerical state-space model into predicate logic, and how we can perform diagnosis using Reiter's well-known algorithm [3]. We show that in case of a fully-observable system our approach is able to find all faults as part of the minimum cardinality diagnosis. By fully-observable we mean that we can observe the behaviour of all components except the water level within the tank. The water level is inferred through calculation in the state-space model. We also give ideas how the approach can be extended to semi-observable and non-observable systems.

This paper makes the following contributions:

1. We show how to capture timing behaviour for model-based diagnosis on the basis of hybrid and dynamic cyber-physical systems.
2. We demonstrate how diagnosis methods (i.e. Reiter's algorithm) from the model-based community can be successfully combined with approaches from the fault detection and isolation (FDI) community.
3. We show how satisfiability modulo theory can help to perform diagnosis in hybrid and dynamic systems by keeping the amount of computations low.

## 2 Demonstration Use Case

For this work we will use the four tank system depicted in Figure 1 as a running example. The system consists of four

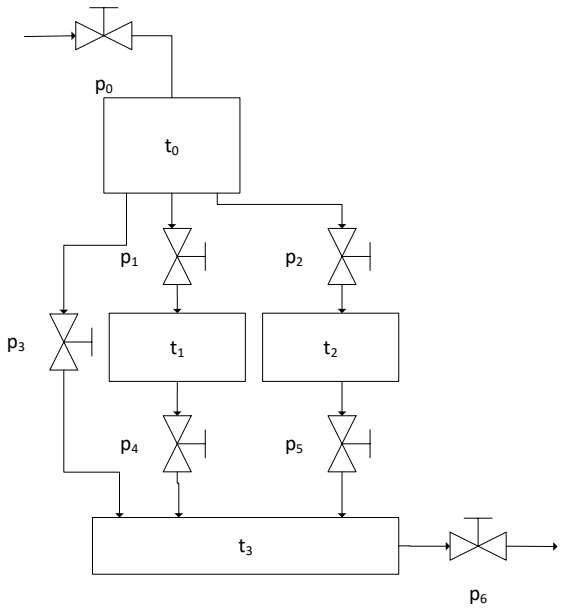


Figure 1: The demonstration use case showing a four-tanks model

water tanks $t$, seven electric valves $p$ with integrated flow sensors, an unlimited water source and an unlimited water sink (not shown). Valve $p_0$ controls water from the unlimited water source, for example the public water mains, into tank $t_0$. From there three pipes with an equal diameter lead to valves $p_1$, $p_2$, and $p_3$. Valve $p_1$ leads into tank $t_1$ and valve $p_2$ leads into tank $t_2$. Valve $p_3$ bypasses both tanks and is directly connected to tank $t_3$. Over valves $p_4$ and $p_5$ the two tanks can be drained into tank $t_3$. Finally, valve $p_6$ drains tank $t_3$ into the unlimited water sink, for example a river or a processing facility.
Each tank has two binary sensors which indicate overflow and underflow, respectively. There are no provisions to directly measure the water level. Each valve has a switch which indicates whether or not the valve is enabled. In addition, each valve has an associated flow sensor. For the present system the following assumptions are made:

**Assumption 1** (Pipes)**.** *Pipes are invisible to the system, have ideal physical properties and cannot break*

**Assumption 2** (Measurement errors)**.** *Measurements from the flow sensors and over- and underflow switches are always perfect without measurement error. If necessary, it is stated when this assumption is relaxed.*

Justification for assumption 1 is that it suffices to simulate faults within the valves and tanks. Modelling the pipes with physical properties would dramatically increase the model size and thus reduce the clarity. Assumption 2 is taken to simplify the used equations. When necessary this assumption is relaxed.
This demonstration use case can be imagined as a preprocessing stage in a larger industrial plant within the process industry. A reliable external water supply is provided by the facilities of the industrial park. The water flows from the supply line into a buffer tank $t_0$. From there it can go into one or both or the two intermediate tanks or bypass both tanks to go directly into tank $t_3$. The two intermediate tanks can be thought of as a mixing stage (not modelled) where ingredients are added to the water until it reaches the holding tank $t_3$. From the holding tank the water flows to subsequent process steps.

The water level in tanks can be described by well-known differential equations. Laubwald [4] provided a comprehensive overview about modelling multiple water-tank systems. A single tank can be described with the differential equation

$$Q_i(t) - Q_o(t) = A\frac{dh}{dt} \tag{1}$$

which describes the time derivative of the height $h$ given the tank area $A$. $Q_i$ is the inflow to the tank and $Q_o$ is the outflow. However, in the real world tanks are subjected to gravity and properties of their materials. Therefore the outflow of a tank is calculated by

$$Q_o = C_d a\sqrt{2gh} \tag{2}$$

where $C_d$ is the discharge coefficient taking into account all fluid characteristics, losses, and irregularities and $g$ is the gravitational constant. $a$ is the cross sectional area of the orifice within the tank. All tanks have a perfectly circular bottom and a cylindrical shape. Combining equations 1 and 2 one can create

$$Q_i(t) = C_d a\sqrt{2gh} + A\frac{dh}{dt} \tag{3}$$

and reformulating this to bring $\frac{dh}{dt}$ on the left-hand side

$$\frac{dh}{dt} = \frac{1}{A}(Q_i - C_d a\sqrt{2gh}) \tag{4}$$

To calculate a new tank height $h$ at time $t$, given the previous height $h_0$ one can use

$$h(t) = h_0 + \Delta h \tag{5}$$

Through substitution into equation 4 this results in

$$h(t) = h(t-1) + \frac{1}{A}(Q_i(t) - C_d a\sqrt{2gh(t-1)}) \tag{6}$$

which describes that, given the discharge coefficient, the gravitation, and the diameter of the orifice it is possible to calculate a new height from a given input with only the knowledge of the previous water level.
These differential equations can be used to create a simulation of the four-tank system depicted in 1. The following

| Parameter | Element | Value | Unit |
|---|---|---|---|
| Area | | | |
| $A$ | $t_0$ | 4.0 | $\mathrm{m}^2$ |
| | $t_1$ | 2.0 | $\mathrm{m}^2$ |
| | $t_2$ | 2.0 | $\mathrm{m}^2$ |
| | $t_3$ | 6.0 | $\mathrm{m}^2$ |
| Height | | | |
| $H$ | $t_0$ | 20.0 | $\mathrm{m}$ |
| | $t_1$ | 10.0 | $\mathrm{m}$ |
| | $t_2$ | 10.0 | $\mathrm{m}$ |
| | $t_3$ | 20.0 | $\mathrm{m}$ |
| Discharge Coefficient | | | |
| $C_d$ | $t_0$ | 1.0 | None |
| | $t_1$ | 1.0 | None |
| | $t_2$ | 1.0 | None |
| | $t_3$ | 1.0 | None |
| Orifice | | | |
| $a$ | $p_0$ | 0.3 | $\mathrm{m}^2$ |
| | $p_1$ | 0.1 | $\mathrm{m}^2$ |
| | $p_2$ | 0.1 | $\mathrm{m}^2$ |
| | $p_3$ | 0.1 | $\mathrm{m}^2$ |
| | $p_4$ | 0.1 | $\mathrm{m}^2$ |
| | $p_5$ | 0.1 | $\mathrm{m}^2$ |
| | $p_6$ | 0.3 | $\mathrm{m}^2$ |
| Gravitational Constant | g | 9.81 | $\mathrm{m/s}^2$ |

Table 1: Parameters of the four-tank system for the demonstration use case

parameters were used to run the system simulation: The parameters in table 1 have been chosen to represent a typical industrial use case. At the same time, noise parameters such as the discharge coefficient are kept neutral to further the argument. The parameters area and height for each tank have been chosen such that they are big enough to hold and store some water while the experiments are running. This way occurring errors will also have a longer time to propagate. Further, the orifices for each tank are quite small. This prolongs the time it takes to drain the tanks in case the water supply stop through the occurrence of a fault.

## 3 Related Work

Struss [5] published a paper on the fundamentals of MBD of dynamic systems. In this he described how hybrid systems can be modelled without resorting to a complete simulation of the system under investigation. He proposed to capture the temporal and dynamic behaviour of a hybrid system in a set of modes which model the system. Each mode has distinct state and temporal constraints in addition to so called Continuity, Integration, and Derivatives (CID) constraints that affect all modes. For one mode, all variables have a domain which captures the permissible states (i.e. values) for this variable. Diagnosis is performed by checking whether the set of constraints together with the observations from sensors is consistent. He demonstrates his approach on a car's anti-braking system and claims to find all usually occurring faults.

When dealing with hybrid systems there always exists the problem of discretization. Provan used a composite automaton for this. Struss divides his system into modes by discretizing the underlying sensor values. Lin [6] already showed in 1994 that online and offline diagnosis for discrete event systems (DES) can be realised by using simple Moore and Mealy automata.

Daigle et al. [7] have adapted a discrete event approach to diagnose continuous systems. They claim that each fault that occurs in a continuous system has a unique fault signature. A fault signature denotes a qualitative effect that a fault occurs in an observation. They also claim that there exists a measurement ordering that describes which sequence measurements deviate until a fault occurs. To capture fault ordering they manually construct a temporal causal graph. Under the assumption that all fault signatures and measurement orderings are known, they employ a diagnoser that traces the states through the temporal causal graph based on measurements. The output of the diagnoser is a fault trace. A second diagnosis algorithm takes this fault trace and determines which components must be faulty to explain this trace. This second diagnosis step is similar to the diagnosis lattice introduced by Reiter.

Grastien et al. [8] have developed an approach to extend Reiter's diagnosis algorithms which was described for binary circuits to include discrete event systems and hybrid systems. Their approach is similar to Daigle et al, Struss, and Provan in that they transform the continuous parts of a model into qualitative states. Following this, their preferred-first algorithm goes through Reiter's diagnosis lattice and computes valid hypothesis with the goal of finding a minimum cardinality diagnosis. An improvement over previous work is that they implement their hypotheses tests with a SAT solver.

Roychoudhury et al. [9] [10] have shown how to use hybrid bond graphs (HBG) to diagnose hybrid systems. HBGs abstractly model the system by describing causal, continuous relationships between components. In Daigle et al. [7] they have shown how to employ the developed HBGs to diagnose a spacecraft power distribution system. Prakash et al. [11] have used an extended framework with HBGs to make improvements in diagnosing two-tank systems.

Grastien [12] used SMT for the diagnosis of hybrid systems. He discretizes values in a hybrid system into a set of distinct states. Each observation $< \tau, A >$is understood as a behaviour $A$ at time $\tau$, where $A$ is a partial assignment of the variables in a state. Measurement errors are included by including constraints which state that the observed voltage must be between two tolerance thresholds. Each variable is augmented with an indicator stating at which time-step the variable expression is valid.

Fraenzle et al. [13] have augmented SMT with stochastic in order to analyse stochastic hybrid systems. By using bounded-model checking together with probabilistic hybrid automata, piecewise deterministic Markov processes, and stochastic differential equations they are able to create an analysis system without the need to formulate intermediate finite-state abstractions as the methods mentioned above do.

In another work Khorasgani and Biswas [14] describe a hybrid system model through hybrid minimal structurally overdetermined sets (HMSOs). These are sets of differential equations and (in-) equations which model the behaviour of a hybrid system. Their FDI algorithm works as follows: The algorithm detects the current system mode and generates an appropriate model. From this it generates a minimal set of HMSOs for this mode. The residuals are computed for each HMSO and can then be combined with fault signature to perform diagnosis.

In contrast to Struss, Provan, and Lin we do not use au-

tomatons and mode estimation to partition the system into different states. Instead, we only sample the system at some suitable interval and use the obtained information directly to model the states in the state-space representation. Unlike to space-craft in the case of Daigle in industrial systems fault signatures and measurement orderings are unknown, which requires us to pursue a more uninformed approach. Our approach can be more seen as an alternative to hybrid bond graphs used by Roychoudhury et al., while they are at the same time an extension to the work of Grastien and Khorasgani and Biswas. In comparison to Grastien we do not singly use satisfiability modulo theory, but instead capture system behaviour in a state-space representation. We expect this to reduce the required computational effort. We also make use of (in-) equations and differential equations as were used by Khorasgani and Biswas, but augment these with the diagnostic reasoning of traditional model-based diagnosis. Compared to Fraenzle, we do not make use of stochastic SMT at this point to keep the system more explainable for users.

## 4 Modelling a Hybrid System

This section first shows the requirements for developing and evaluating a FDI method for hybrid, cyber-physical systems. Then it shows the concept to realise these requirements. The developed approach makes use of MBD by modelling the hybrid system with a state-space representation. This model is augmented with an observer, which determines boolean residuals. These residuals indicate whether or not a component is faulty. The information about which components indicate faults are merged with sensor observations and make up the diagnostic part of the approach. Diagnosis is done through Reiter's diagnosis lattice.
The formal form of a state-space representation is:

$$\begin{aligned} x(t+1) &= (A+\Delta A)x(t) + (B+\Delta B)u(t) + \\ &\quad B_d d_k(t) + Ba f_a(t) + B_c f_c(t) \\ y(t) &= (C+\Delta C)x(t) + D_s f_s(k) + D_\omega \omega(k) \end{aligned} \tag{7}$$

$x(t)$ is the systems's state, $u(t)$ the control input, $y(t)$ the observed output, $f_a(t)$ the unexpected actuator fault, $f_c(t)$ a component fault, $f_s(t)$ a sensor fault $d_k(t)$ a process disturbance, and $\omega(t)$ measurement noises. $A, B, C, B_d, B_a, B_c, D, s, D_\omega$ are known parameter matrices and $\Delta A, \Delta B, \Delta C$ modelling parameter errors.
For observer-based methods and to calculate residuals these equations can reformulated to:

$$\begin{aligned} \hat{x}(t+1) &= A\hat{x}(t) + Bu(t) + Kr(t) \\ r(t) &= y(t) - \hat{y}(t) \\ \hat{y}(t) &= C\hat{x}(t) \end{aligned} \tag{8}$$

Here, $\hat{x}(t)$ and $\hat{y}(t)$ are estimates of the state and output values. $r(t)$ is the calculated residual signal, and $K$ is a gain factor.
According to assumption 2 we can safely neglect the factors the $\Delta$ and $\omega$ terms in equation 7. Further, in this approach we will only model component faults. Therefore, we can remove $f_a(t)$, $f_s(t)$, and through assuming that there are no disturbances within the process we can also remove $d_k(t)$. This leaves only the system's observable input, observable output, state, and component fault in equation 7. Through these simplifications equation 7 becomes closer to equation 8. To perform model-based diagnosis according to the principles proposed by Reiter [3] it is necessary to separate the diagnostics part from the state propagation. Therefore, we will not calculate classical residuals as in equation 8 and can further remove the gain factor and residuals $Kr(t)$.
For this work the state-space model needs to be described more abstractly. First the top-level information flow is described. This shows how the state is propagated within the system. Here, the model is general enough to be extended and adapted to many use cases. After that the state-space model is described on the demonstration use case introduced in section 2. In a third step the diagnosis part is described which fusions the calculation of binary residuals with an expression in predicate and SMT logic. The state is propagated through

$$\begin{aligned} \mathbf{x}(t+1) &= f(\mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{y}(t) &= g(\mathbf{x}(t), \mathbf{u}(t), \tau) \end{aligned} \tag{9}$$

where $\mathbf{x}(t+1)$ is a vector of the state in the next time step, $\mathbf{x}(t)$ is the current state vector, $\mathbf{u}(t)$ is the observable input vector, $\mathbf{y}(t)$ is the observable output vector, and $\tau$ is a vector of threshold values.
According to the demonstration use case the water level cannot be measured directly. Therefore, each tank's water level needs to be calculated through its inflow and outflow. The inflow and outflow can be measured through the associated valves in each in- and outflow pipe. Since each tank has sensors to indicate under- and overflow, these are used for the target (output). For the state, input, and output vectors we thus have:

$$\mathbf{x} = \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} flow_0 \\ flow_1 \\ \vdots \\ flow_6 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} overflow_0 \\ \vdots \\ overflow_3 \\ underflow_0 \\ \vdots \\ underflow_3 \end{bmatrix}$$

The function $f(\mathbf{x}, \mathbf{u})$ models the current state and its current input and from this computes the next state. Therefore we can write:

$$f(\mathbf{x}(t), \mathbf{u}(t)) = \mathcal{A}\boldsymbol{\Delta}(\mathbf{x}, \mathbf{u}, t) + \mathcal{B}\mathbf{u}(t) \tag{10}$$

with the connection matrices being

$$\mathcal{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and

$$\mathcal{B} = \begin{bmatrix} 1 & -1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & -1 \end{bmatrix}$$

$\mathcal{A}$ shows that each current state only influences the exact next state. For the demonstration use case this means each differential equation which models a tank will only affect the state of this single tank. Matrix $\mathcal{B}$ shows the connections between the system's components, which in this case are the pipes between the tanks. The first row describes how the system's primary input is connected as an input (indicated by the number 1 in the first column) to tank 1. The

three values of $-1$ in the first row show there are three pipes that are used as the output of tank 1.
To model the water level in each tank it is possible to use differential equations. Each differential equation has the form introduced in section 2:

$$h(t) = h(t-1) + \frac{1}{A}(Q_i(t) - C_d a\sqrt{2gh(t-1)}) \quad (11)$$

Using the parameters from the state-space system this is written as

$$x(t+1) = \frac{1}{A}(u(t) - C_d a\sqrt{2gx(t)}) \quad (12)$$

A vector $\mathbf{\Delta}(\mathbf{x}, \mathbf{u}, t)$ can be created with the right-hand side of these equations:

$$\mathbf{\Delta}(\mathbf{x}, \mathbf{u}, t) = \begin{bmatrix} x_0(t+1) = \frac{1}{A_0}(u_0(t) - C_{d,0}a_0\sqrt{2gx_0(t)}) \\ x_1(t+1) = \frac{1}{A_1}(u_1(t) - C_{d,1}a_1\sqrt{2gx_1(t)}) \\ x_2(t+1) = \frac{1}{A_2}(u_2(t) - C_{d,2}a_2\sqrt{2gx_2(t)}) \\ x_3(t+1) = \frac{1}{A_3}(u_3(t) - C_{d,3}a_3\sqrt{2gx_3(t)}) \end{bmatrix}$$

With this model it is possible to propagate the state of the system as it evolves through time. Differential equations calculate the water level in the tank for the next state, given the current water level and the inflow obtained by reading the valve flow sensors. However, given this information a control system cannot yet determine the full behaviour of the system. For this, the output vector $\mathbf{y}(t) = g(\mathbf{x}(t), \mathbf{u}(t), \tau)$ needs to be calculated.

$$g(\mathbf{x}(t), \mathbf{u}(t), \tau) = \mathcal{C} \begin{bmatrix} o(h_0, \tau_0^o) \\ \vdots \\ o(h_3, \tau_3^o) \\ l(h_0, \tau_0^l) \\ \vdots \\ l(h_3, \tau_3^l) \end{bmatrix} \quad (13)$$

and

$$\mathcal{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$\tau$ is a vector of threshold values which indicate at what height the tank is overfull or underfull. For notation we use $\tau_i^o$ to denote the threshold for the upper limit of tank $i$ and $\tau_i^l$ to denote the lower limit of tank $i$. The diagonal matrix $\mathcal{C}$ maps the results of the functions $o(h, \tau)$ and $l(h, \tau)$ into the output vector $y$. The function $o(h, \tau)$ indicates when the water level within the tank has approached the upper limit. This is calculated by

$$o(h_i, \tau_i^o) = \begin{cases} 0 & \text{if } h_i \leq \tau_i^o \\ 1 & \text{else} \end{cases} \quad (14)$$

Likewise, the lower limit of the water level can be calculated

$$l(h_i, \tau_i^l) = \begin{cases} 0 & \text{if } h_i \geq \tau_i^l \\ 1 & \text{else} \end{cases} \quad (15)$$

To diagnose faults within the described state-space system it is necessary to obtain health information about single components. In the presented demonstration use case two fault models for tanks and valves exist, respectively. Tanks fail, when the water level within the tank reaches either the upper limit (overflow) or the lower limit (underflow). Pumps fail when the measured flow deviates more than a certain amount from the expected flow.
Classical MBD uses observations(OBS), a system description(SD), and a component description(COMPS) for describing a system. After having described the actual behaviour of the hybrid system with state-space equations, it is now important to translate this into diagnostic information. OBS are given by the input vector $\mathbf{u}(t)$. The component behaviour COMPS is described by the differential equations in the case of tanks and by assuming no further properties for the valves, resulting in $input(valve_i) = output(valve_i)$. SD is given in two parts. For normal operation this is the incidence matrix $\mathcal{B}$, a predicate logic description of the inputs and outputs of the system, and a fault model. For the given demonstration use case it suffices to specify a weak-fault model (WFM). A WFM to model the fault modes of the tanks can be specified as

$$\sigma_{T,i} : H_{T,i} \rightarrow \neg o(i) \wedge \neg l(i) \quad (16)$$

For valves the statement is specified as

$$\sigma_{P,i} : H_{P,i} \rightarrow (flow_i^l \leq flow_i) \wedge (flow_i^u \geq flow_i) \quad (17)$$

In this case the health variables $H$ do not describe a probability for the component being faulty, but are instead binary. The terms $\sigma_{t,i}$ and $\sigma_{p,i}$ can be written as a vector

$$\mathbf{C} = \begin{bmatrix} \sigma_{T,0} & \dots & \sigma_{T,3} & \sigma_{P,0} & \dots & \sigma_{P,6} \end{bmatrix}^T$$

If $\mathbf{C}$ is semantically interpreted through an SMT solver $\mathcal{C}' = \mathcal{I}(\mathcal{C})$, we obtain the diagnosis vector

$$\mathbf{C}' = \begin{bmatrix} c_0 & c_1 & \dots & c_{10} \end{bmatrix}^T$$

with $c_i \in \{\top, \bot\}$. This vector shows for each component whether it is faulty or not, given the current observations from the sensors.
The numerical information for the statements $\sigma_{T,i}$ and $\sigma_{P,i}$ is obtained from the state space model. Within the statements the state vector $\mathbf{x}(t)$ represents the water level $h_i$ and the input vector $\mathbf{u}(t)$ represents the flow values $flow_i$. By interpreting the statements it is possible to translate the subsymbolic, numerical data within the state-space model into symbolic information used for diagnosis through the vector $\mathbf{C}'$
Equation 9 shows the propagations of the state vector through time. For each new time step the statements 16 and 17 have to be reformulated. To capture this time-related behaviour we adopt the notation of Grastien [12] and state that $varname@t$ stands for the variable $varname$ at time $t$, where $t \in \mathbb{N}^0$. From this, we can state the value for each variable at each observed time step. When the observations are only carried out while the system is in normal operation it is possible to create a logical representation of all observations so far:

$$\bigwedge_t \sigma_{T,i}@t \cup \bigwedge_t \sigma_{P,i}@t$$

which describes the logical conjunction of all $\sigma_{T,i}$ and $\sigma_{P,i}$ over all time steps. Adding the statements in each time step to the knowledge base as described by Grastien will

increase the required space linearly and still take exponential time to check the consistency. Especially in large industrial plants where observations run for months with individual observations being performed at second intervals, a linearly growing knowledge base is infeasible. For example, when observing 200 signals with a sampling rate of one second a knowledge base would grow by 17,280,000 data points per day. Therefore, in this work we will focus only on the observations in the current time step. This keeps the knowledge base size constant and adds no additional computational complexity. More observations can bring a higher precision in locating a fault. In this case, the number of observations can be increased by some constant factor, taking for example always the last three observations into account.
A hybrid system can be represented through a directed-acyclic graph (DAG) showing the connections and causal relationships between components. Depending on the location of the component within the graph a fault in one component may cause several other components to fail as well. In the demonstration use case, for example, if valve $p_0$ fails, all the other valves and the tanks will also exhibit anomalous behaviour. The goal in diagnosis is therefore to find the smallest amount of components which would explain a fault. This search for minimum cardinality diagnoses can be done with Reiter's diagnosis lattice. First, a power set $P = \mathbb{P}(\mathrm{COMPS})$ is constructed. This contains all sets of sets of components. From this the diagnosis lattice can be created. On the bottom is the empty set which denotes no faulty components. In the row above are all sets that contain exactly one component. In the row above that are sets that contain exactly two components and so forth. Each observation of the sensors within the system leads to a recomputation of the set of possible faulty components $\mathcal{C}'$. By computing the hitting sets of all these observations it is possible to close in on the faulty components. In the diagnosis lattice this is done by searching the lattice bottom-up and refuting all branches which include a component that can be proven to be healthy through observations. Once the lattice has been searched the solutions with the minimum number of components are the minimum cardinality diagnoses $\omega'$.
Once the diagnosis framework has been set up three possible usages can be identified: fully-observable, semi-observable, and non-observable. In the first type of usage the diagnoser can observe every property of the physical system, except the water level described by the state vector $\mathbf{x}(t)$. In the second type of usage, only a subset of sensors are accessible to the diagnoser. Thus, some values need to be approximated. In the third type of usage only the primary inputs and primary outputs can be observed, while all other values need to be inferred. The following three sub-sections describe these types of usage in detail.

## 5 Observability

In this paper we will focus on diagnosis of fully-observable systems. This simplifying use case makes it easier to describe the methods, while the extension to semi-observable systems, and non-observable systems is reserved for future work. However, this section gives some ideas on how to extend the developed methodology to include semi- and non-observable systems.

**Fully-observable system**
In full-observable systems we assume that each component can be observed. For the demonstration use case this means that we can measure the water flow through each valve at each point in time. This is also a realistic assumption for many smaller scale application and most demonstration plants which are built with observability in mind. Older and more complex industrial plants, however, contain more often component whose parameters can not be observed.

**Semi-observable system**
In the case that the system is semi-observable, not all components' behaviour can be observed with sensors. This is the case in most real-world industrial plants where its either too expensive to add sensors for every machine parameter or it is infeasible due to physical constraints or historical reasons. A diagnostic system which cannot observe every parameter has to work with partial information. If necessary the missing values need to be estimated. In the case boolean circuits this can be done straightforward. For every component in a boolean circuit the behaviour model is known. Further, the system description SD is known. If only parts of the components can be observed a diagnostic reasoning system can infer the missing values.
In hybrid, physical systems inferring values is more difficult. Some real-world components may behave non-linearly, stochastically, or very unpredictable. Further, signal propagation may not be instantaneous as in boolean circuits, but a change in one parameter may only be noticeable some time later. This is the case, for example, in bioreactors. If the temperature in a reactor changes, the substance may only exhibit a change in an observable property some time later.
In the demonstration use case the tanks are modelled through differential equations and the valves have the throughput that is maximally allowed by the outflow of a tank. Thus, even if not all sensors can be observed it is still possible to infer missing values.

**Non-observable system**
In non-observable systems only the primary inputs and outputs are observable. A diagnostic system needs to measure the primary input signals, the primary output signals and combine these with the model SD and COMPS of the hybrid system. To perform diagnosis, every intermediate value must be assumed by propagating the primary input values through the system. This approach is the most computing intensive, since assumable values need to be computed in sequence. Diagnosis is performed by comparing the expected primary output values with the observed primary output values and then going through the circuit back-to-front to find diagnosis candidates.

## 6 Experiments

To show that the developed diagnostic methodology works as intended 16 experiments with the simulation of the demonstration use case were carried out. These are divided into two sets. In the first set the primary input to the demonstration use case was a constant stream of water. We expect in this case that during normal operation the water level in the tanks will reach a constant height and remain there until a fault occurs. In the second set of experiments the primary input was changed to a sinusoidal water stream. For this we used the function

$$in(t) = \begin{cases} O + \alpha \sin(2t) & \text{if } t \leq T\pi \\ 0 & \text{else} \end{cases} \tag{18}$$

Equation 18 shows the form of the sinusoidal wave with period $T$. We use an offset $C$ to ensure a constant basic input stream into tank 1. The gain factor $A$ adjusts the period such that we achieve variability within the tank water levels, but without triggering and under- or overflow. Further, we use a piecewise function to cut off the negative half-wave of the sinusoidal input wave for convenience and ease of interpretation.

For both sets of experiments the normal operating condition, five cases of single-faults, and two cases of double-faults were simulated. For each experiment 300 time steps were carried out, with the respective faults being injected at time step 100 and being removed at time step 200.

We split the developed method into two parts. Part one is the quantitative simulation of the demonstration use case described in section 2. Part two is the diagnosis algorithm consisting of the state-space model, the SMT logic, and the diagnosis lattice. Both parts have been written in Python 3.4.5. The quantitative simulation provides the user with functionalities to inject faults and generate normal process data. The location and number faults can be specified as well as the type of input (for example, if the water inflow is constant or sinusoidal). The output of the simulation is a .csv file which contains all process data as well as the diagnostic information. This method was chosen to be close to real industrial use cases.

SD is modelled through predicate logic. In the fully-observable case for the demonstration use case it suffices to explicitly model the connections between components, inputs, and outputs. Therefore, only three functions are used for the predicate logic: The function $component(c)$ with arity 1, and the function $input(i, c)$ and $output(o, c)$ with arity 2. These model the names of components in the system and the number and names of their inputs and outputs. In addition the relation $connects(c_i, c_j)$ specifies which input is connected to which output. For the present use case the constants $source$ and $sink$ are used to denote primary inputs and primary outputs, respectively. With this logic it is possible to describe the connections between components in the form:

$$
\begin{aligned}
&component(t0) \wedge \\
&\dots \\
&component(p6) \wedge \\
&input(t0.i0, t0) \\
&\dots \\
&output(p6.o0, p6) \wedge \\
&connects(source, p0.i0) \\
&\dots \\
&connects(t3.o0, p6.i0) \\
&connects(p6.o0, sink)
\end{aligned}
$$

## 7 Results

Table 2 shows the experiments for constant and sinusoidal input streams, the injected fault and whether or not the fault was detected. An $x$ in the column *detected* denotes that the injected fault was among the result set of the diagnosis algorithm. This means the algorithm is complete. An $x^*$ denotes that only the injected fault was detected, which corresponds to soundness of the algorithm. It must be noted here, however, that finding only the injected faults depends heavily on the granularity of the underlying data source. For example, if valve 5 stops working its flow would almost immediately go to 0. The sampling frequency is high enough to detect this decrease in the flow rate early enough that the water level in the tanks is not yet significantly affected. However, in large industrial plants sampling rates are often far lower. A faulty component might then only be recognised once its effects have propagated into other observations from other components. Further, in the semi- and non-observable cases not every status of every component is known. In this case, too, the set of possible faulty components will grow in size. As the criterion in table 2 we evaluated the output of the diagnosis algorithm in the time step 101 which was directly after the fault had been injected. It is evident that due to the SMT logic statement in equation 17 every unexpected change in the throughput of a valve would be immediately detected.

| | Constant | | Sinusoidal | |
|---|---|---|---|---|
| Index | # Faults | Detected | # Faults | Detected |
| 0 | $p_0$ | x* | $p_0$ | x* |
| 1 | $p_3$ | x* | $p_3$ | x* |
| 2 | $p_5$ | x* | $p_5$ | x* |
| 3 | $p_6$ | x* | $p_6$ | x* |
| 4 | $p_1, p_3$ | x* | $p_1, p_3$ | x* |
| 5 | $p_4, p_5$ | x* | $p_4, p_5$ | x* |

Table 2: Recognized faults for experiments with constant input stream or sinosoidal input stream at time-step 100

| | Constant | | Sinusoidal | |
|---|---|---|---|---|
| Index | # Faults | Detected | # Faults | Detected |
| 0 | $p_0$ | x (11) | $p_0$ | x (11) |
| 1 | $p_3$ | x (3) | $p_3$ | x (3) |
| 2 | $p_5$ | x (3) | $p_5$ | x (3) |
| 3 | $p_6$ | x (3) | $p_6$ | x (3) |
| 4 | $p_1, p_3$ | x (5) | $p_1, p_3$ | x (7) |
| 5 | $p_4, p_5$ | x (6) | $p_4, p_5$ | x (6) |

Table 3: Recognized faults for experiments with constant input stream or sinosoidal input stream at time-step 199

However, table 3 shows the results when the output of the diagnosis algorithm was evaluated directly before removing a fault at time-step 199. The number in brackets denotes the size of the minimum-cardinality set, while $x$ still denotes whether or not the fault was within the result set. As was the case in table 2, all faults were detected, though the results set also contained components which were not faulty.

Figure 2 shows the development of the water level in tank 3 for the experiment with a constant water input and the fault being injected at valve $p_5$. In this case, the flow from tank 2 into tank 3 is blocked. In the figure the grey line represents normal working behaviour and the black line abnormal working behaviour. The shaded area indicates the time during which $p_5$ is simulated to be faulty. From time-step 0 until 100 it is evident that the water level in tank 3 is the same in both conditions. The tank was initialised with a water level of 7m, which first drains as not enough water is flowing into the tank. Then, once tanks 0, 1, and 2 have reached their normal operating conditions tank 3 stabilises. Once the fault is injected the water level in tank 3 becomes unsteady. This results from the implementation of the model which only models tank levels until they reach their upper limit. Thus, with $p_5$ being disabled tank 2 begins to over-

flow. This in turn redistributes the water pressures in all the other tanks. Once the fault is removed, however, all flows reach their maximum level and the water level in tank 3 increases.

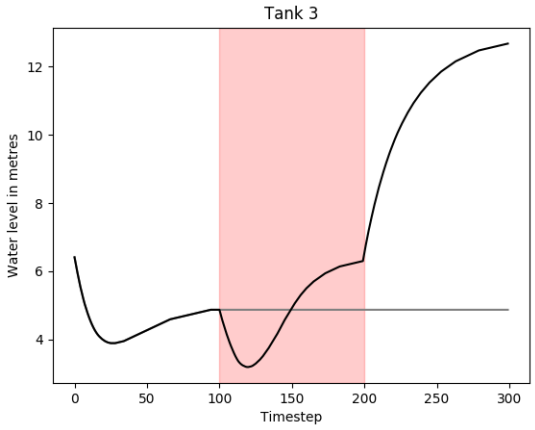


Figure 2: Development of the water level in tank 3 over time in normal conditions (grey) and with a fault in valve 5 (black)

## 8 Conclusion

This paper shows for the limited domain of a fully-observable, hybrid, dynamic industrial production system that we can model its behaviour with state-space equations and then translate it into satisfiability modulo theory and perform diagnosis. So far, operators in the process industry rely only on fault identification techniques such as support-vector machines, artificial neural networks, Bayesian approaches etc. In this work we present an approach which can be used to also bring fault isolation into cyber-physical production systems. Given a suitable model of the production system the presented method is able to capture behaviour over time while preserving the ability to directly react to faults. With a suitably chosen data sampling frequency the potentially huge knowledge-base proposed by Grastien [8] can be avoided.

For future work we will show how to extend this approach to deal with semi- and non-observable systems. For these we need better models of single components by, for example, creating assumables with the help of differential equations specified in SMT logic.

Another direction for further research is the automatic generation and learning of system models. Nowadays, models of systems need to be created manually which is not generalizable and time consuming. The state-space equations and their translations into logic are simple and can be specified algorithmically. Therefore, an attempt should be made to at least semi-automatically learn parts of these models from descriptions of meta-data.

## References


[1] Rolf Isermann and Peter Balle. Trends in the application of model-based fault detection and diagnosis of technical processes. *Control engineering practice*, 5(5):709–719, 1997.

[2] Marta Fullen, Peter Schüller, and Oliver Niggemann. Defining and validating similarity measures for industrial alarm flood analysis. In *Industrial Informatics (INDIN), 2017 IEEE 15th International Conference on*, pages 781–786. IEEE, 2017.

[3] Raymond Reiter. A theory of diagnosis from first principles. *Artificial intelligence*, 32(1):57–95, 1987.

[4] Elke Laubwald. Coupled tanks systems 1. *control-systems-principles. co. uk*, 2015.

[5] Peter Struss. Fundamentals of model-based diagnosis of dynamic systems. In *IJCAI (1)*, pages 480–485, 1997.

[6] Feng Lin. Diagnosability of discrete event systems and its applications. *Discrete Event Dynamic Systems*, 4(2):197–212, 1994.

[7] Matthew J Daigle, Indranil Roychoudhury, Gautam Biswas, Xenofon D Koutsoukos, Ann Patterson-Hine, and Scott Poll. A comprehensive diagnosis methodology for complex hybrid systems: A case study on spacecraft power distribution systems. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 40(5):917–931, 2010.

[8] Alban Grastien, Patrik Haslum, Sylvie Thiébaux, et al. Conflict-based diagnosis of discrete event systems: Theory and practice. In *KR*, 2012.

[9] Indranil Roychoudhury, Matthew J Daigle, Gautam Biswas, and Xenofon Koutsoukos. Efficient simulation of hybrid systems: A hybrid bond graph approach. *Simulation*, 87(6):467–498, 2011.

[10] Sriram Narasimhan and Gautam Biswas. Model-based diagnosis of hybrid systems. *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and humans*, 37(3):348–361, 2007.

[11] Om Prakash and AK Samantaray. Model-based diagnosis and prognosis of hybrid dynamical systems with dynamically updated parameters. In *Bond Graphs for Modelling, Control and Fault Diagnosis of Engineering Systems*, pages 195–232. Springer, 2017.

[12] Alban Grastien. Diagnosis of hybrid systems by consistency testing. In *24th International Workshop on Principles of Diagnosis (DX-13)*, pages 9–14. Citeseer, 2013.

[13] Martin Fränzle, Holger Hermanns, and Tino Teige. Stochastic satisfiability modulo theory: A novel technique for the analysis of probabilistic hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 172–186. Springer, 2008.

[14] Hamed Khorasgani and Gautam Biswas. Structural fault detection and isolation in hybrid systems. *IEEE Transactions on Automation Science and Engineering*, 2017.