A Structural Model Decomposition Framework for Systems Health Management

Indranil Roychoudhury
SGT Inc., NASA Ames Research Center
Moffett Field, CA 94035, USA
indranil.roychoudhury@nasa.gov

Matthew Daigle
NASA Ames Research Center
Moffett Field, CA 94035, USA
matthew.j.daigle@nasa.gov

Anibal Bregon, Belarmino Pulido
University of Valladolid
Valladolid, 47011, Spain
{anibal, belar}@infor.uva.es

Abstract— **Systems health management (SHM) is an important set of technologies aimed at increasing system safety and reliability by detecting, isolating, and identifying faults; and predicting when the system reaches end of life (EOL), so that appropriate fault mitigation and recovery actions can be taken. Model-based SHM approaches typically make use of global, monolithic system models for online analysis, which results in a loss of scalability and efficiency for large-scale systems. Improvement in scalability and efficiency can be achieved by decomposing the system model into smaller local submodels and operating on these submodels instead. In this paper, the global system model is analyzed offline and structurally decomposed into local submodels. We define a common model decomposition framework for extracting submodels from the global model. This framework is then used to develop algorithms for solving model decomposition problems for the design of three separate SHM technologies, namely, estimation (which is useful for fault detection and identification), fault isolation, and EOL prediction. We solve these model decomposition problems using a three-tank system as a case study.**

TABLE OF CONTENTS

1. INTRODUCTION

Systems health management (SHM) is a critical set of technologies that focus on increasing system safety and reliability by detecting, isolating, and identifying faults; and predicting when the system reaches end of life (EOL), so that fault mitigation and recovery actions can be taken. Model-based SHM approaches [1–4] are usually favored over purely data-driven methods [5] when knowledge about the system and its behaviors is available. Model-based approaches typically make use of global, monolithic system models for online analysis, which results in a loss of scalability and efficiency for large-scale systems.

These issues can be addressed by decomposing the global SHM problems into local, independent subproblems, in which the local results can be merged into a global result. For a model-based approach, this corresponds to a decomposition of the underlying models [6–9]. Such decompositions are performed as offline design activities based on structural analysis of the global system model.

In this paper, we propose a common model decomposition framework and an algorithm for structural model decomposition. We show how, through offline analysis of the global system model, we can solve model decomposition problems for three separate SHM tasks, namely, estimation (used for residual generation, fault detection, and fault identification), fault isolation, and prediction (used for fault prognostics). For each SHM task, we select a specific approach – [10] for estimation, [7] for isolation, and [9] for prognostics – and demonstrate how the system can be decomposed using the common framework proposed in this paper to fulfill the needs of each approach.

Our decomposition framework, utilizing the notion of computational causality, decomposes models into submodels by considering some of the variables in a model as local inputs. As a result, some of the unknown variables in the model can be computed from these local inputs, thereby resulting in a smaller set of equations that are decoupled from the rest of the system equations given these local inputs. Thus each submodel can be used to compute the values of its unknown variables independently from all other submodels. These independent submodels allow reasoning algorithms to be implemented on each submodel independently, thereby distributing the computation and improving efficiency.

Specifically, this paper makes the following contributions: (*i*) we propose a formal model decomposition framework for defining causal models and submodels; (*ii*) we develop an algorithm for designing submodels based on structural model decomposition; and (*iii*) we demonstrate the usefulness of the model decomposition framework by applying it to model decomposition problems for three specific SHM functions: estimation, fault isolation, and prediction. Throughout this paper, we illustrate our model decomposition framework and algorithms using a three-tank system.

This paper is organized as follows. Section 2 presents the model decomposition framework. Sections 3-5 demonstrate how this structural model decomposition framework can be leveraged to design submodels that are used in estimation, fault isolation, and prediction, respectively. Section 6 presents related work, and Section 7 concludes the paper.

2. MODEL DECOMPOSITION FRAMEWORK

This section presents our framework for defining models and submodels and a structural model decomposition algorithm

978-1-4673-1813-6/13/$31.00 ©2013 IEEE.
[1] IEEEAC Paper #2343, Version 2, Updated 05/01/2013.

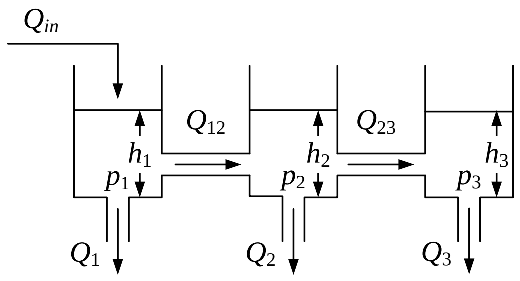


Figure 1. Schematic of three-tank system.

that can be leveraged for efficient and scalable implementation of different functions of SHM. We begin with the definition of a *model*.

Definition 1 (Model) A *model* $\mathcal{M}$ is a tuple $\mathcal{M} = (V, C)$, where V is a set of variables, and C is set of constraints. V consists of five disjoint sets, namely, the set of state variables, X; the set of parameters, Θ; the set of inputs, U; the set of outputs, Y; and the set of auxiliary variables, A. Each constraint $c = (\varepsilon_c, V_c) \in C$ consists of an equation ε_c involving variables $V_c \in V$.

Input variables $u \in U$ are known or measured; and the output variables $y \in Y$ correspond to (measured) sensor signals. Parameters $\theta \in \Theta$ include explicit model parameters that are used in the model constraints. Θ need not consist of all parameters in the equations, only those that must be included explicitly (e.g., for joint state-parameter estimation, as shown in Section 3, or fault isolation, as shown in Section 4). These parameters, by definition, are not computed in terms of any other variables, and, in this way, appear as inputs. Since the state variables X are, by definition, enough to describe the future behavior of the system, the auxiliary variables $a \in A$ are not strictly needed, however, they make the model easier to parse, develop, and implement.

As shown in Defn. 1, a constraint $c = (\varepsilon_c, V_c)$ includes an equation ε_c over the set of variables V_c. Note that c does not impose any computational causality on the variables V_c, i.e., although ε_c captures the information about how to compute a variable $v \in V_c$ in terms of all other variables in V_c, the constraint does not specify which $v \in V_c$ is the dependent variable in equation ε_c. We write a constraint $c_1 = (\varepsilon_{c_1}, V_{c_1})$ by its equation, e.g., as follows:

$$a + b = c + d \qquad (c_1)$$

where $V_{c_1} = \{a, b, c, d\}$.

Throughout the paper, we will use a three-tank model as a common example. The three-tank model is representative of several real-world systems, such as spacecraft propellant loading systems [11] and fuel transfer systems of fighter aircrafts [12], among others. Fig. 1 shows the schematic of a three-tank system. For tank $i \in \{1, 2, 3\}$, p_i denotes the pressure at the bottom of the tank, h_i denotes the fluid height in the tank, and Q_i denotes the volumetric flow rate out of the outflow pipe. For adjacent tanks i and j, Q_{ij} denotes the flow rate in the connecting pipe, and Q_{in} is the inflow into tank 1.

Example 1: We model the three-tank system with the following constraints:

$$p_1 = \int_{t_0}^{t} \dot{p}_1 dt \qquad (c_2)$$

$$p_2 = \int_{t_0}^{t} \dot{p}_2 dt \qquad (c_3)$$

$$p_3 = \int_{t_0}^{t} \dot{p}_3 dt \qquad (c_4)$$

$$\dot{p}_1 = \frac{1}{K_1}\left(Q_{in} - \frac{p_1}{R_1} - \frac{p_1 - p_2}{R_{12}}\right) \qquad (c_5)$$

$$\dot{p}_2 = \frac{1}{K_2}\left(\frac{p_1 - p_2}{R_{12}} - \frac{p_2}{R_2} - \frac{p_2 - p_3}{R_{23}}\right) \qquad (c_6)$$

$$\dot{p}_3 = \frac{1}{K_3}\left(\frac{p_2 - p_3}{R_{23}} - \frac{p_3}{R_3}\right) \qquad (c_7)$$

$$h_1^* = \frac{p_1 \cdot K_1}{A_1} \qquad (c_8)$$

$$Q_{12}^* = \frac{p_1 - p_2}{R_{12}} \qquad (c_9)$$

$$Q_3^* = \frac{p_3}{R_3} \qquad (c_{10})$$

where for tank i, A_i denotes the tank cross-sectional area, K_i denotes the capacitance, R_i denotes the resistance of the outflow pipe, and for tanks i and j, R_{ij} denotes the flow resistance of the pipe between the tanks. Here $X = \{p_1, p_2, p_3\}$, $\Theta = \varnothing$, $U = \{Q_{in}\}$, $Y = \{h_1^*, Q_{12}^*, Q_3^*\}$[2], and $A = \{\dot{p}_1, \dot{p}_2, \dot{p}_3\}$.

In order to define for a constraint c which variable $v \in V_c$ is the dependent variable that is computed by the others using the constraint, we require the notion of a *causal assignment*.

Definition 2 (Causal Assignment) A *causal assignment* α to a constraint $c = (\varepsilon_c, V_c)$ is a tuple $\alpha = (c, v_c^{out})$, where $v_c^{out} \in V_c$ is assigned as the dependent variable in equation ε_c.

Unlike a constraint, a causal assignment defines a computational causality (or computational direction) to a particular variable $v_c^{out} \in V_c$ in the constraint in which it can be computed in terms of all other variables in V_c. We write a causal assignment of a constraint using the constraint's equation in a causal form. For example, for constraint c_1 above choosing $v_{c_1}^{out} = d$:

$$d := a + b - c \qquad (\alpha_1)$$

where Constraint c_1 is rewritten with a := symbol to explicitly denote that the direction of computation is from variables a, b, and c to d.

We say that a set of causal assignments $\mathcal{A}$, for a model $\mathcal{M}$ is *valid* if

- For all $v \in U \cup \Theta$, $\mathcal{A}$ does not contain any α such that $\alpha = (c, v)$, i.e., U and Θ are not computed in terms of any other variables.

[2]We name output variables with an asterisk so as to not confuse the measured variables from unmeasured versions of them that may be used as state or auxiliary variables.

- For all $v \in Y$, $\mathcal{A}$ does not contain any $\alpha = (c, v_c^{out})$ where $v \in V_c - \{v_c^{out}\}$, i.e., no variable is computed in terms of any $y \in Y$.
- For all $v \in V - U - \Theta$, $\mathcal{A}$ contains exactly one $\alpha = (c, v)$, i.e., other than the variables in U and Θ, every variable must have exactly one constraint to compute it.

A *causal model* is a model extended with a valid set of causal assignments.

Definition 3 (Causal Model) Given a model $\mathcal{M}^* = (V, C)$, a *causal model* for $\mathcal{M}^*$ is a tuple $\mathcal{M} = (V, C, \mathcal{A})$, where $\mathcal{A}$ is a set of valid causal assignments.

Example 2: The causal assignments for the three-tank model introduced in Example 1 are as follows:

$$p_1 := \int_{t_0}^{t} \dot{p}_1 dt \qquad (\alpha_2)$$

$$p_2 := \int_{t_0}^{t} \dot{p}_2 dt \qquad (\alpha_3)$$

$$p_3 := \int_{t_0}^{t} \dot{p}_3 dt \qquad (\alpha_4)$$

$$\dot{p}_1 := \frac{1}{K_1}\left(Q_{in} - \frac{p_1}{R_1} - \frac{p_1 - p_2}{R_{12}}\right) \qquad (\alpha_5)$$

$$\dot{p}_2 := \frac{1}{K_2}\left(\frac{p_1 - p_2}{R_{12}} - \frac{p_2}{R_2} - \frac{p_2 - p_3}{R_{23}}\right) \qquad (\alpha_6)$$

$$\dot{p}_3 := \frac{1}{K_3}\left(\frac{p_2 - p_3}{R_{23}} - \frac{p_3}{R_3}\right) \qquad (\alpha_7)$$

$$h_1^* := \frac{p_1 \cdot K_1}{A_1} \qquad (\alpha_8)$$

$$Q_{12}^* := \frac{p_1 - p_2}{R_{12}} \qquad (\alpha_9)$$

$$Q_3^* := \frac{p_3}{R_3} \qquad (\alpha_{10})$$

Here, we assume integral causality, i.e., state variables are computed via integration.

For the purposes of visualizing a causal model, we represent $\mathcal{M}$ using a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices corresponding directly to the variables V in $\mathcal{M}$, and $\mathcal{E}$ is the set of edges, where for every $(c, v_c^{out}) \in \mathcal{A}$, we include an edge (v', v_c^{out}) for each $v' \in V_c - \{v_c^{out}\}$.

Example 3: Fig. 2 shows the causal graph for the three-tank system of Example 1 with $Y = \{h_1^*, Q_{12}^*, Q_3^*\}$. State variables are denoted using dashed boxes, output variables are denoted using solid-lined boxes, and input variables are denoted using dashed circles.

Given a model, we are interested in generating submodels that allow for the computation of a given set of variables using only local inputs. Given a definition of the local inputs (in general, selected from V) and the set of variables we wish to be computed by the submodel (selected from $V - U - \Theta$), we create from a causal model $\mathcal{M}$ a causal submodel $\mathcal{M}_i$. We obtain a submodel in which only a subset of the variables in V are computed using only a subset of the constraints in C. In this way, each submodel computes its variable values independently from all other submodels. A submodel can be defined as follows.

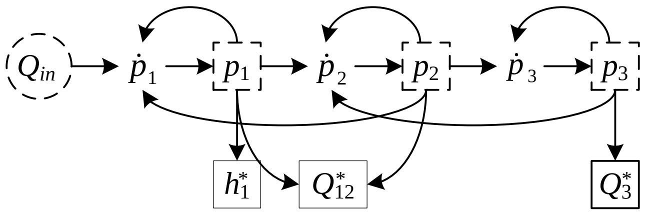


Figure 2. Causal graph of three tank system with $Y = \{h_1^*, Q_{12}^*, Q_3^*\}$.

Definition 4 (Causal Submodel) A *causal submodel* $\mathcal{M}_i$ of a causal model $\mathcal{M} = (V, C, \mathcal{A})$ is a tuple $\mathcal{M}_i = (V_i, C_i, \mathcal{A}_i)$, where $V_i \subseteq V$, $C_i \subseteq C$, and $\mathcal{A}_i$ is a set of (valid) causal assignments for $\mathcal{M}_i$.

Note that, in general, $\mathcal{A}_i$ is not a subset of $\mathcal{A}$, because since we allow to select local inputs from Y, these variables become local inputs, i.e., appear in U_i, and the causal assignment in $\mathcal{A}$ that computes these variables is changed to a form where some other variable in the corresponding constraint is selected as the dependent variable. As a result, these causal assignments will be different, but the rest of the causal assignments in $\mathcal{A}_i$ will still be found in $\mathcal{A}$.

The procedure for generating a submodel from a causal model is given as Algorithm 1. Given a causal model $\mathcal{M}$, a set of variables that are considered as local inputs $U^* \supseteq U$, and a set of variables to be computed V^*, and a preferences list, P (explained below), the `GenerateSubmodel` algorithm derives a causal submodel $\mathcal{M}_i$ that computes V^* using U^*.

In Algorithm 1, the queue, $variables$, represents the set of variables that have been added to the submodel but have not yet been resolved, i.e., they cannot yet be computed by the submodel. This queue is initialized to V^*, the set of variables that must be computed by the submodel. The algorithm then loops until this queue has been emptied, i.e., the submodel can compute all variables in V^* using only variables in U^*. Within the loop, the next variable v is popped off the queue. We then determine the best constraint to use to resolve this variable with the `GetBestConstraint` subroutine (Subroutine 2). We add the constraint to the submodel and the causal assignment for the constraint in the form that computes v. We then need to resolve all the variables being used to compute v, i.e., all its predecessors in the causal graph. Each of these variables that have not already been visited (not already in V_i), are not parameters (not in Θ), and are not local inputs (not in U^*) must be resolved and so are added to the queue. Then the variables are added to the submodel and the loop continues until the queue is emptied.

The goal of the `GetBestConstraint` subroutine is to find the best constraint to resolve v. The subroutine constructs a set C that is the set of constraints that can completely resolve the variable, i.e., resolves v without further backward propagation (all other variables involved in the constraint are in $V_i \cup \Theta \cup U^*$), and then chooses the best according to a preferences list P. If no such constraint exists, then the constraint that computes v in the current causal assignment is chosen, and further backward propagation will be necessary. Here, we are preferring minimal resolutions of v, i.e., those that do not require backward propagation, because then the submodel will be minimal in the number of variables and constraints needed to compute V^*.

Algorithm 1 $\mathcal{M}_i = \texttt{GenerateSubmodel}(\mathcal{M}, U^*, V^*, P)$

```
1:  $V_i \leftarrow V^*$
2:  $C_i \leftarrow \varnothing$
3:  $\mathcal{A}_i \leftarrow \varnothing$
4:  $variables \leftarrow V^*$
5:  while $variables \neq \varnothing$ do
6:      $v \leftarrow \texttt{pop}(variables)$
7:      $c \leftarrow \texttt{GetBestConstraint}(v, V_i, U^*, \mathcal{A}, P)$
8:      $C_i \leftarrow C_i \cup \{c\}$
9:      $\mathcal{A}_i \leftarrow \mathcal{A}_i \cup \{(c, v)\}$
10:     for all $v' \in V_c$ do
11:         if $v' \notin V_i$ and $v' \notin \Theta$ and $v' \notin U^*$ then
12:             $variables \leftarrow variables \cup \{v'\}$
13:         end if
14:         $V_i \leftarrow V_i \cup \{v'\}$
15:     end for
16: end while
17: $\mathcal{M}_i \leftarrow (V_i, C_i, \mathcal{A}_i)$
```

Subroutine 2 $c = \texttt{GetBestConstraint}(v, V_i, U^*, \mathcal{A}, P)$

```
1:  $C \leftarrow \varnothing$
2:  $c_v \leftarrow$ find $c$ where $(c, v) \in \mathcal{A}$
3:  if $V_{c_v} \subseteq V_i \cup U^*$ then
4:      $C \leftarrow C \cup \{c_v\}$
5:  end if
6:  for all $y \in Y \cap U^*$ do
7:      $c_y \leftarrow$ find $c$ where $(c, y) \in \mathcal{A}$
8:      if $v \in V_{c_y}$ and $V_{c_y} \subseteq V_i \cup U^*$ then
9:          $C \leftarrow C \cup \{c_y\}$
10:     end if
11: end for
12: if $C = \varnothing$ then
13:     $c \leftarrow c_v$
14: else if $c_v \in C$ then
15:     $c \leftarrow c_v$
16: else
17:     $C' \leftarrow C$
18:     for all $c_1, c_2 \in C$ where $c_1 \neq c_2$ do
19:         $y_1 \leftarrow$ find $y$ where $(c_1, y_1) \in \mathcal{A}$
20:         $y_2 \leftarrow$ find $y$ where $(c_2, y_2) \in \mathcal{A}$
21:         if $(y_1 \lhd y_2) \in P$ then
22:             $C' \leftarrow C' - \{c_1\}$
23:         end if
24:     end for
25:     $c \leftarrow \texttt{first}(C')$
26: end if
```

In general, a variable v is involved in many constraints, however, exactly one of these constraints, in the given causal assignment, computes v. If this constraint does not completely resolve v, we find the constraints in which v is used to compute some output variable $y \in Y \cap U^*$. We consider modifying the causal assignment so that such a y (used now as an input) is used to compute v, instead of v being used to compute y. This can only be performed if, for the causal assignment in which y is being used to compute v, all other variables involved in the constraint are in $V_i \cup \Theta \cup U^*$, in which case this constraint in this new causal assignment can completely resolve v. If no constraint can be found that completely resolves v, then the constraint that in the current causal assignment computes v will have to be used, and backward propagation will be necessary. Otherwise, we select the most preferable constraint that completely resolves v. Preference among constraints (in which an output would be transformed to an input) is computed using a preferences list P, that contains a partial ordering of all the outputs in the model of the form $y_i \lhd y_j$, meaning that y_j is preferred over y_i. The subroutine goes through every pair of constraints and removes from the list of most preferable constraints, C', any constraint that uses a measured variable that is less preferable to one involved in another constraint. Of those remaining, an arbitrary choice is made. The preferences list can be used to prefer measured variables with less noise over those with more noise.

Example 4: For the three-tank model (Fig. 2), say that we select $U^* = \{Q_{in}, h_1^*, Q_{12}^*\}$ and $V^* = \{Q_3^*\}$, and $P = \{(Q_{12}^* \lhd h_1^*)\}$. Algorithm 1 starts with $V_i = Q_3^*$, and propagates back to p_3, and adds it to V_i. From p_3 it propagates back to $\dot{p}_3$, adding it to V_i. Of the predecessors of $\dot{p}_3$, p_3 is already in V_i, so is not added to the *variables* queue, and p_2 is not, so the algorithm propagates back to p_2, adding it to V_i. At this point, there are two constraints to consider to possibly compute p_2: (*i*) the constraint c_3 with causal assignment α_3 that computes p_2 from $\dot{p}_2$, or (*ii*) the constraint c_9 with causal assignment α_9, set to have the new causal assignment

$$p_2 := p_1 - Q_{12}^* \cdot R_{12}, \qquad (\alpha_{11})$$

that computes p_2 from p_1 and Q_{12}^*. In α_{11}, p_1 is required but is not yet included in V_i, so this constraint cannot completely resolve p_2 and we default to using causal assignment α_3, propagating back to $\dot{p}_2$ and from there to p_1 (p_2 and $\dot{p}_3$ are already in V_i). Now, at p_1, we have three constraints to consider that may resolve p_1: (*i*) the constraint c_2 with causal assignment α_2 that computes p_1 from $\dot{p}_1$, (*ii*) the constraint c_9 with causal assignment α_9, set to have the new causal assignment

$$p_1 := p_2 + Q_{12}^* \cdot R_{12} \qquad (\alpha_{12})$$

that computes p_1 from Q_{12}^* and p_2, and (*iii*) the constraint c_8 with causal assignment α_8, set to have the new causal assignment

$$p_1 := \frac{h_1^* \cdot A_1}{K_1} \qquad (\alpha_{13})$$

that computes p_1 from h_1^*. Since the preferences list P prefers h_1^* over Q_{12}^*, the algorithm chooses to compute p_1 using causal assignment α_{13}. The graph for the resultant submodel is shown in Fig. 3.

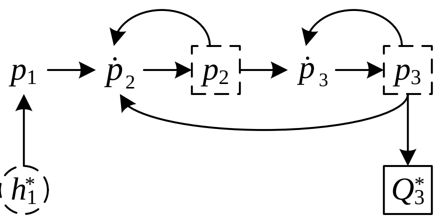


Figure 3. Causal graph for the minimal submodel of the three-tank system computed when $U^* = \{Q_{in}, h_1^*, Q_{12}^*\}$, $V^* = \{Q_3^*\}$ and $P = \{(Q_{12}^* \lhd h_1^*)\}$.

The algorithm generates *complete* submodels, i.e., the submodels contain at least the variables needed to compute its V^*. This is guaranteed because the algorithm only stops propagation at variables included in $V_i \cup \Theta \cup U^*$. Since $U^* \supseteq U$, given a particular U^* and V^*, there may be multiple submodels that can compute V^* using U^*, with the global system model always being an option. The goal of the decomposition algorithm is to generate *minimal* submodels, i.e., submodels with the minimal number of constraints (and hence, state variables) needed to compute V^* using U^*. It is possible for nonminimality to be obtained when the

GetBestConstraint subroutine could have resolved a constraint by searching in the forward causality direction more than one constraint ahead. For example, a variable v can be resolved if an output is computed from a single auxiliary variable that is computed only from v, however GetBestConstraint would ignore such a possibility. The algorithm can be extended to handle such a situation, although at a loss of efficiency, but that is beyond the scope of this work. However, the simpler solution is to structure the auxiliary variables to avoid such circumstances, e.g, when minimality is a requirement the auxiliary variables can be discarded to ensure minimality.

In the worst case, the algorithm must visit all variables and constraints. On each variable, Subroutine 2 is called, which in the worst case considers all variables in $Y \cap U^*$. So, the overall worst-case time complexity is $O((|V| + |E|) \cdot |Y \cap U^*|)$. Since $(Y \cap U^*) \subset V$, the algorithm is polynomial in the model size. On average, some amount of decomposition will be possible so the complexity will be much lower in practice.

In the following sections, we show how this model decomposition approach can be applied to the problems of estimation, fault isolation, and prediction.

3. Model Decomposition for Estimation

Estimation in the SHM context can be used for three main purposes, namely, for fault detection (through residual generation), fault identification, and systems health state estimation. The three different purposes require the estimation problem to be posed somewhat differently. For example, for fault detection, the estimation problem involves estimating the value of measured variables (based on estimated values of state variables) and comparing these estimates to actual measured values to a generate residual signal. For fault identification, a set of parameters associated with faults are estimated. Finally, for system health state estimation, joint estimations of hidden state and unknown parameter variables are computed. In general, the estimation problem is to generate a probability distribution $p(X, \Theta|Y)$.

The advantage of model decomposition for estimation is that the global system model is partitioned into minimal submodels, which can be implemented in a distributed fashion, obtaining robust and efficient online estimation solutions as compared to estimation using the global system model. Each local estimator i computes $p(X_i, \Theta_i|Y_i)$. Due to the decoupled nature of the submodels, the local estimation problems can be defined independently of each other, and the local estimators do not need to communicate. This is in contrast to other distributed estimation schemes in which local estimators communicate state estimates among one another.

The model decomposition framework proposed in this paper can be used to derive minimal estimator submodels from the global system model. Each minimal estimator will take a subset of the system inputs, U, and the system outputs, Y, to compute an estimate of measured variables, Y_i. So, the U^* and V^* sets for this decomposition problem are formed as $U^* = U \cup (Y - \{Y_i\})$ and $V^* = Y_i$. To define a set of local estimators, we typically create one for each $y \in Y$, where $V^* = Y_i = \{y\}$. In previous works, we have used minimal estimation submodels for tracking and fault detection as in [10], for fault identification as in [10, 13], and for systems health state estimation [8].

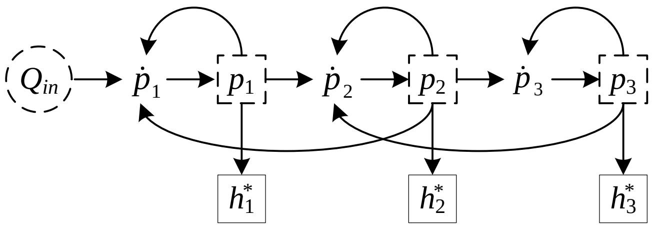


Figure 4. Causal graph of three tank system with $Y = \{h_1^*, h_2^*, h_3^*\}$.

Example 5: As an example, consider the three-tank model with $Y = \{h_1^*, h_2^*, h_3^*\}$ that is given by constraints c_2–c_8 and the new constraints

$$h_2^* = \frac{p_2 \cdot K_2}{A_2} \qquad (c_{11})$$

$$h_3^* = \frac{p_3 \cdot K_3}{A_3} \qquad (c_{12})$$

with the causal assignments α_2-α_8 and the additional causal assignments:

$$h_2^* := \frac{p_2 \cdot K_2}{A_2} \qquad (\alpha_{14})$$

$$h_3^* := \frac{p_3 \cdot K_3}{A_3}. \qquad (\alpha_{15})$$

Fig. 4 shows the causal graph for the three-tank system with the selected outputs. Here, $X = \{p_1, p_2, p_3\}$, $\Theta = \varnothing$, $Y = \{h_1^*, h_2^*, h_3^*\}$, $U = \{Q_{in}\}$, and $A = \{\dot{p}_1, \dot{p}_2, \dot{p}_3\}$. We generate three minimal submodels, one for each measurement. For the first minimal submodel, we select $U^* = \{Q_{in}, h_2^*, h_3^*\}$ and $V^* = \{h_1^*\}$, for the second one $U^* = \{Q_{in}, h_1^*, h_3^*\}$ and $V^* = \{h_2^*\}$, and for the third one $U^* = \{Q_{in}, h_1^*, h_2^*\}$ and $V^* = \{h_3^*\}$. The minimal submodels computed by Algorithm 1 for this example include: for $V^* = \{h_1^*\}$, constraints with causal assignments α_2, α_5, α_8, and

$$p_2 := \frac{h_2^* \cdot A_2}{K_2} \qquad (\alpha_{16})$$

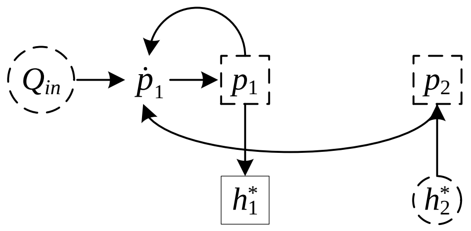
derived from the constraint c_{11} with the causal assignment α_{14} (see Fig. 5a); for $V^* = \{h_2^*\}$, constraints with causal assignments α_3, α_6, α_{13}, α_{14}, and

$$p_3 := \frac{h_3^* \cdot A_3}{K_3} \qquad (\alpha_{17})$$

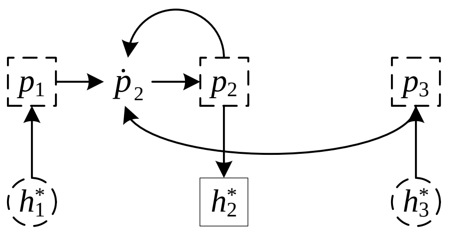
derived from the constraint c_{12} with causal assignment α_{15} (see Fig. 5b); and for $V^* = \{h_3^*\}$, constraints with causal assignments α_4, α_7, α_{16}, and α_{17} (see Fig. 5c). Note that constraints and variables can be included in several submodels.

Example 6: As a second example, consider the model shown in Fig. 2 ($Y = \{h_1^*, Q_{12}^*, Q_3^*\}$), using causal assignments α_5-α_{10} and $P = \{(h_1^* \lhd Q_{12}^*)\}$. For this example, we also compute three minimal submodels, one for each measurement. The minimal submodels computed for this example are: for $V^* = \{h_1^*\}$, constraints with causal assignments α_2, α_5, α_8, and α_{11} (see Figure 6a); for $V^* = \{Q_{12}^*\}$, constraints with causal assignments α_3, α_6, α_9, α_{13}, and

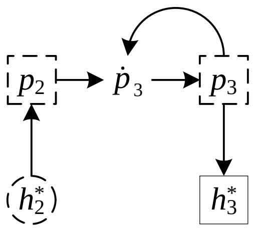
$$p_3 := Q_3^* \cdot R_3 \qquad (\alpha_{18})$$



(a) Causal graph for h_1^* submodel.



(b) Causal graph for h_2^* submodel.



(c) Causal graph for h_3^* submodel.

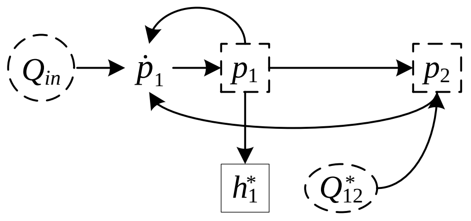
Figure 5. Causal graphs for minimal submodels for estimation of the three-tank system when $Y = \{h_1^*, h_2^*, h_3^*\}$.

derived from constraint with causal assignment α_{10} (see Figure 6b); and for $V^* = \{Q_3^*\}$, constraints with causal assignments α_3, α_6, α_4, α_7, α_{10}, and α_{12} (see Figure 6c). Note that using the preferences P from Example 4 we would have obtained the submodel whose causal graph is shown in Fig. 3. Compared to the previous example, where every submodel has only one state variable, in this example, the submodel for $V^* = \{Q_3^*\}$ includes state variables p_2 and p_3.

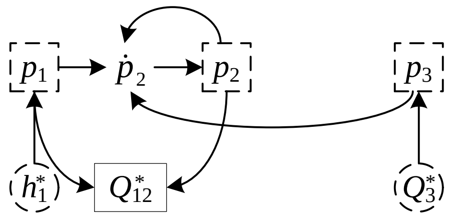
4. Model Decomposition for Fault Isolation

Model decomposition can also be used for the purposes of fault isolation [1, 6]. Here, residual generators are formed from the submodels, and the advantage of model decomposition is that each residual generator will be sensitive to only the faults in the corresponding submodel. As a result, diagnosability is improved [14]. In our approach, we attempt to design local diagnosers based on the submodels that can independently, without communication, generate globally correct fault isolation results [7].

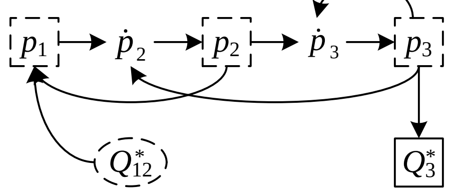
For a given system model $\mathcal{M}$, we define a set of faults F. In this work, we assume that this set of faults can be represented as abrupt, persistent changes in the parameters Θ of the system model. For a given fault isolation algorithm and output set Y, distinguishability of a fault f_i from another fault f_j refers to the notion that the isolation algorithm can, if f_i occurs, eliminate f_j from consideration. We denote this as



(a) Causal graph for h_1^* submodel.



(b) Causal graph for Q_{12}^* submodel.



(c) Causal graph for Q_3^* submodel.

Figure 6. Causal graphs for minimal submodels for estimation of the three-tank system when $Y = \{h_1^*, Q_{12}^*, Q_3^*\}$ and $P = \{(h_1^* \lhd Q_{12}^*)\}$.

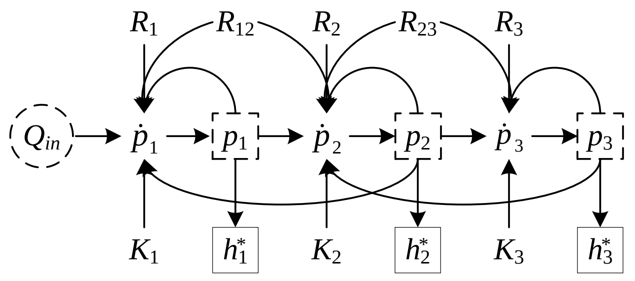


Figure 7. Causal graph of the three tank system including fault parameters.

$f_i \not\sim_Y f_j$. Diagnosability is generally defined as follows.

Definition 5 (Diagnosability) A model $\mathcal{M}$ with faults F and outputs Y is *diagnosable* with a given fault isolation algorithm if and only if $(\forall f_i, f_j \in F) f_i \neq f_j \implies f_i \not\sim_Y f_j$.

Example 7: As an example, consider the three-tank model where tank heights are measured (see Example 5 in Section 3). In this case, we make the system parameters explicit, so $\Theta = \{K_1, R_1, R_{12}, K_2, R_2, R_{23}, K_3, R_3\}$, and consider faults that are represented as increases in the resistances and decreases in the capacitances (denoted using K_i^-, R_i^+, and R_{ij}^+). The causal graph for this model is shown in Fig. 7. As the fault isolation algorithm we use the QED (Qualitative Event-based Diagnosis) approach [15, 16]. In this approach, for each fault we generate a list of symbolic fault signatures, representing magnitude (first symbol) and slope (second symbol) changes in the residuals associated with measurements [2] (where + indicates an increase, − indicates a decrease, and 0 indicates no change), and relative measurement orderings, representing the sequential order of residual deviations within a submodel in response to a fault [15]. Table 1 shows the signatures and orderings for the three-tank system. Here, r_y refers to the residual for output y, and an ordering $r_{y_1} \prec r_{y_2}$ means that r_{y_1} will deviate before r_{y_2}. Also, a $^+$ superscript indicates an increase and a $^-$ superscript indicates a decrease in the parameters. This model is diagnosable because for each pair of faults, the combination of signatures and orderings is different.

Fault	$r_{h_1^*}$	$r_{h_2^*}$	$r_{h_3^*}$	Measurement Orderings
K_1^-	+−	0+	0+	$r_{h_1} \prec r_{h_2}, r_{h_1} \prec r_{h_3}, r_{h_2} \prec r_{h_3}$
R_1^+	0+	0+	0+	$r_{h_1} \prec r_{h_2}, r_{h_1} \prec r_{h_3}, r_{h_2} \prec r_{h_3}$
R_{12}^+	0+	0−	0−	$r_{h_2} \prec r_{h_3}$
K_2^-	0+	+−	0+	$r_{h_2} \prec r_{h_1}, r_{h_2} \prec r_{h_3}$
R_2^+	0+	0+	0+	$r_{h_2} \prec r_{h_1}, r_{h_2} \prec r_{h_3}$
R_{23}^+	0+	0+	0−	$r_{h_2} \prec r_{h_1}$
K_3^-	0+	0+	+−	$r_{h_2} \prec r_{h_1}, r_{h_3} \prec r_{h_1}, r_{h_3} \prec r_{h_2}$
R_3^+	0+	0+	0+	$r_{h_2} \prec r_{h_1}, r_{h_3} \prec r_{h_1}, r_{h_3} \prec r_{h_2}$

Table 1. Fault signatures and relative measurement orderings for the global model of the three-tank system.

We split $\mathcal{M}$ into submodels $\mathcal{M}_i$. We partition the fault set F into subsets F_i, one for each submodel. We are interested in the property of global diagnosability [17].

Definition 6 (Global Diagnosability) A submodel $\mathcal{M}_i$ of model $\mathcal{M}$, with faults $F_i \subseteq F$, is *globally diagnosable* with a given fault isolation algorithm if $(\forall f_i \in F_i, f_j \in F) f_i \neq f_j \implies f_i \nsim_{Y_i} f_j$.

In other words, a submodel is globally diagnosable if the fault isolation algorithm can distinguish all local faults (in F_i) from all other possible faults (both local and nonlocal, i.e., in F) using only its local outputs. If a local fault occurs, then the fault isolation algorithm will know exactly the fault (because it is distinguishable from all other local faults). If a nonlocal fault occurs, the algorithm will know only that the fault is not local (since all local faults are distinguishable from all nonlocal faults). If we design the submodels such that they are all globally diagnosable for their given F_i and Y_i, then we guarantee that the local fault isolation results will be globally correct. When a fault occurs, exactly one local diagnoser will know exactly the fault and all other diagnosers will know the fault is nonlocal.

Example 8: Consider still the three-tank system where heights h_1^*, h_2^*, and h_3^* are measured. We partition the fault sets into $F_1 = \{K_1^-, R_1^+, R_{12}^+\}$, $F_2 = \{K_2^-, R_2^+, R_{23}^+\}$, and $F_3 = \{K_3^-, R_3^+\}$. Consider $\mathcal{M}_1$ created using $U^* = U \cup \{h_2^*, h_3^*\}$ and $V^* = \{h_1^*\}$, so the only residual will be h_1^*. As observed in Table 1, the submodel is not globally diagnosable, because R_1^+ and R_{12}^+ both produce 0+ (smooth increase) on $r_{h_1^*}$. Consider now $\mathcal{M}_1$ created using $U^* = U \cup \{h_3^*\}$ and $V^* = \{h_1^*, h_2^*\}$. Now, faults not in F_1 are included in the submodel (namely, K_2^-, R_2^+, R_{23}^+), so they in addition to the local faults (in F_1) may affect the residuals. If we do not consider measurement orderings, the submodel is not globally diagnosable, because a local fault, R_1^+, and a nonlocal fault, R_2^+, produce the same effects on the two residuals, $r_{h_1^*}$ and $r_{h_2^*}$. If we do consider measurement orderings, however, the submodel is globally diagnosable, because although we observe the same effects on the residuals for those two faults, the effects appear in a different temporal sequence (R_1^+ affects $r_{h_1^*}$ first whereas R_2^+ affects $r_{h_2^*}$ first).

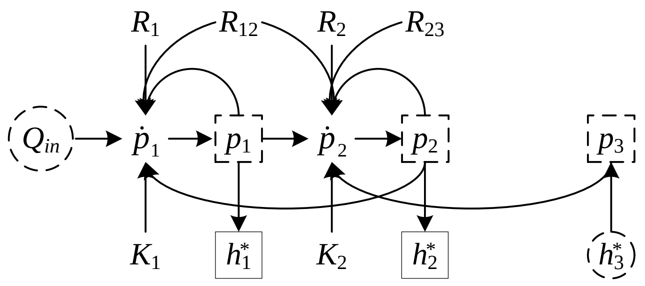
Algorithm 3 Globally Diagnosable Submodel Design

```
**Input:** $\mathbb{S} = \{\mathcal{S}_i = (F_i, Y_i, \mathcal{M}_i) : i = 1, \ldots, n\}, \mathcal{M}, P$
**for all** $\mathcal{S}_i \in \mathbb{S}$ **do**
    **while** $\mathcal{S}_i$ not globally diagnosable **do**
        $\mathbb{S}_y \leftarrow \varnothing$
        **for all** $y \in Y - Y_i$ **do**
            $Y_i' \leftarrow Y_i \cup \{y\}$
            $\mathcal{M}_i' \leftarrow$ GenerateSubmodel$(\mathcal{M}, U \cup (Y - Y_i'), Y_i', P)$
            $S_i' \leftarrow (F_i, Y_i', \mathcal{M}_i')$
        **end for**
        $S_i \leftarrow \arg\min_{S_i' \in \mathbb{S}} GD(F, S_i')$
    **end while**
**end for**
```

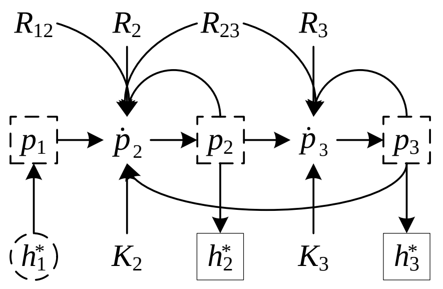
An algorithm for designing globally diagnosable submodels is shown as Algorithm 3 (generalized from [7]). An initial submodel design is given, in which for Y_i, the submodel is defined as $\mathcal{M}_i =$ GenerateSubmodel$(\mathcal{M}, U \cup (Y - Y_i), Y_i, P)$, where P is the (given) set of preferences. The fault set partition and initial measurement sets are provided by the user. We denote the tuple $(F_i, Y_i, \mathcal{M}_i)$ using $\mathcal{S}_i$. Here we define a function $GD(F, \mathcal{S}_i)$ that returns the number of faults in F_i that are not globally distinguishable from those in F.

The algorithm implements a greedy solution that works as follows. For each submodel, we try adding one new output at a time until the submodel becomes globally diagnosable. For each new output that we try to add, we determine how much it improves the global diagnosability (using GD), and whichever one improves it the most (by minimizing GD) is added. We then continue by trying to find the next best output to add. This loop completes when global diagnosability is attained, which in the worst case will occur when all outputs are added. This assumes that the global model is diagnosable; if it is not, then we can group indistinguishable faults as a single *aggregate fault* [17] to make the system diagnosable (a diagnosis of the aggregate fault is interpreted as an ambiguity between the indistinguishable faults). If the worst case is achieved, this essentially means that the fault isolation problem is not decomposable for the given model, fault set, and fault isolation algorithm.

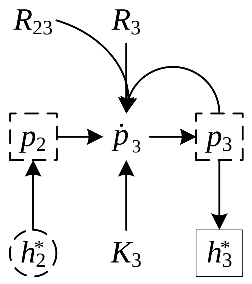
Example 9: Consider again the scenario given in Example 8, where the tank heights are measured and the fault set is partitioned into $F_1 = \{K_1^-, R_1^+, R_{12}^+\}$, $F_2 = \{K_2^-, R_2^+, R_{23}^+\}$, and $F_3 = \{K_3^-, R_3^+\}$. We create $\mathcal{M}_1$ using $Y_1 = \{h_1^*\}$, $\mathcal{M}_2$ using $Y_2 = \{h_2^*\}$, and $\mathcal{M}_3$ using $Y_3 = \{h_3^*\}$. Applying the design algorithm, we find that the first submodel is extended to include h_2, the second is extended to include h_3, and the third does not need to be extended. The corresponding causal graphs are shown in Fig. 8. As discussed in Example 8, $\mathcal{M}_1$ is not diagnosable using only $Y_1 = \{h_1^*\}$. The case is similar for $\mathcal{M}_2$, which is not diagnosable using only $Y_2 = \{h_2^*\}$. However, $\mathcal{M}_3$ using $Y_3 = \{h_3^*\}$ is globally diagnosable, because K_3^- and R_3^+ produce different effects on $r_{h_3^*}$, and there are no nonlocal faults included in the submodel (see



(a) Causal graph for isolation submodel for $\{K_1^-, R_1^+, R_{12}^+\}$.



(b) Causal graph for isolation submodel for $\{K_2^-, R_2^+, R_{23}^+\}$.



(c) Causal graph for isolation submodel for $\{K_3^-, R_3^+\}$.

Figure 8. Causal graphs for isolation submodels for the three-tank system.

Fig. 8c). For $\mathcal{M}_1$, if we add h_2^* (Fig. 8a), in which R_1^+, R_{12}^+ produce different effects, the submodel now becomes globally diagnosable (as discussed in Example 8). For $\mathcal{M}_2$, if h_3^* is added the case is similar and the submodel becomes globally diagnosable.

5. Model Decomposition for Prediction

In the context of SHM, prognostics deals with predicting the time at which a component, subsystem, or system will no longer satisfy desired performance constraints. For example, we may want to predict when a battery will discharge [18], when a valve will no longer open within required time limits [19], or when a crack grows to an unacceptable size [20]. The time at which this event occurs is known as end of life (EOL) and the time until that point is known as remaining useful life (RUL).

The desired performance is expressed through a set of n_c constraints, $C_{EOL} = \{c_i\}_{i=1}^{n_c}$, where c_i describes the mapping of a given point in the joint state-parameter space given the current inputs to a variable $e_i \in \{0,1\}$, where 1 means the performance requirement is satisfied and 0 means it is not satisfied, and we say the system has no *useful* life remaining. In model-based prognostics [3], the system model is augmented to include these constraints, i.e., for $M = (V, C)$, $C_{EOL} \subset C$ and the e_i variables are included in V (as auxiliary variables in A). The causality of these constraints is always fixed such that for constraint c_i, the causal assignment takes the following form:

$$e_i := f_{c_i}(v_1, v_2, \ldots, v_m), \qquad (\alpha_{19})$$

where $\{v_1, v_2, \ldots, v_m\} = V_{c_1}$, and f_{c_i} is a function evaluating the performance requirement.

These individual constraints may be combined into a single system-level *threshold function* $T_{EOL} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_\theta} \times \mathbb{R}^{n_u} \to \mathbb{B}$, defined as

$$T_{EOL}(\mathbf{x}(t), \boldsymbol{\theta}(t), \mathbf{u}(t)) = \begin{cases} 1, & 0 \in \{f_{c_i}(\mathbf{x}(t), \boldsymbol{\theta}(t), \mathbf{u}(t))\}_{i=1}^{n_c} \\ 0, & \text{otherwise.} \end{cases}$$

That is, T_{EOL} evaluates to 1, i.e., the system has reached an unacceptable region of behavior, when any one of the constraints are violated. EOL is then defined as

$$EOL(t_P) \triangleq \inf\{t \in \mathbb{R} : t \geq t_P \wedge T_{EOL}(\mathbf{x}(t), \boldsymbol{\theta}(t), \mathbf{u}(t)) = 1\},$$

i.e., EOL is the earliest time point at which T_{EOL} is met (evaluates to 1). RUL is expressed using EOL as

$$RUL(t_P) \triangleq EOL(t_P) - t_P.$$

Example 10: Consider the three-tank system given by causal assignments α_2-α_4 as well as the following causal assignments:

$$\dot{p}_1 := \frac{1}{K_1}(Q_{in} - Q_1 - Q_{12}) \qquad (\alpha_{20})$$

$$\dot{p}_2 := \frac{1}{K_2}(Q_{12} - Q_2 - Q_{23}) \qquad (\alpha_{21})$$

$$\dot{p}_3 := \frac{1}{K_3}(Q_{23} - Q_3) \qquad (\alpha_{22})$$

$$Q_1 := \frac{p_1}{R_1} \qquad (\alpha_{23})$$

$$Q_2 := \frac{p_2}{R_2} \qquad (\alpha_{24})$$

$$Q_3 := \frac{p_3}{R_3} \qquad (\alpha_{25})$$

$$Q_{12} := u_{12}\frac{p_1 - p_2}{R_{12}} \qquad (\alpha_{26})$$

$$Q_{23} := u_{23}\frac{p_2 - p_3}{R_{23}} \qquad (\alpha_{27})$$

Here, u_{12} and u_{23} are the positions of valves on the connecting pipes. Note also that we include auxiliary variables for the flows out of and between the tanks. The model is augmented with the EOL constraints and the associated variables:

$$e_1 := (p_1 > p^-) \qquad (\alpha_{28})$$

$$e_2 := (p_2 > p^-) \qquad (\alpha_{29})$$

$$e_3 := (p_3 > p^-) \qquad (\alpha_{30})$$

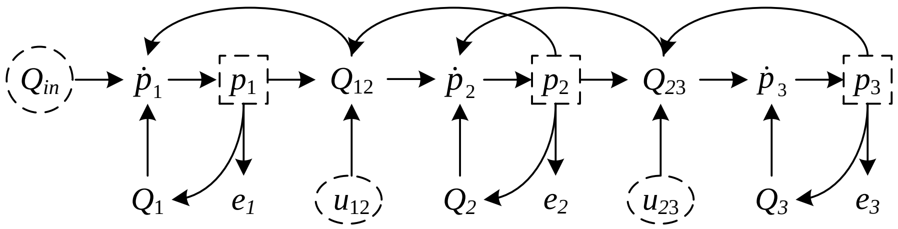


Figure 9. Causal graph for global three-tank prediction model.

where p^- is the minimum allowable tank pressure. Here, $X = \{p_1, p_2, p_3\}$, $\Theta = \varnothing$, $U = \{Q_{in}, u_{12}, u_{23}\}$, $Y = \varnothing$, and $A = \{\dot{p}_1, \dot{p}_2, \dot{p}_3, e_1, e_2, e_3\}$. The corresponding causal graph is shown in Fig. 9.

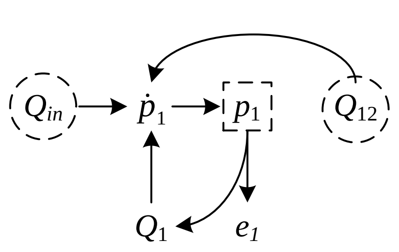
By decomposing the global model into local submodels, we can decompose the prediction problem [9]. For each $c \in C_{EOL}$, we create a local submodel that can evaluate that constraint. Since T_{EOL} is reached when any single $c \in C_{EOL}$ is violated, this time point can be computed by taking the minimum of the local EOLs corresponding to the local performance constraints. For decomposition, we must define a set of local inputs. For prediction, the set of available local inputs is denoted as $U_P \subset V$. Here, U_P is a set of variables whose future values (or a distribution of future values) can be hypothesized. Note that since we do not have access to future sensor readings, V does not include any output variables.

The submodels are generated as follows. For each $c \in C_{EOL}$, we set $\mathcal{M}_c \leftarrow \texttt{GenerateSubmodel}(\mathcal{M}, U_P, v_c, \varnothing)$. (Here, the preferences list P is not needed since there are no outputs.)

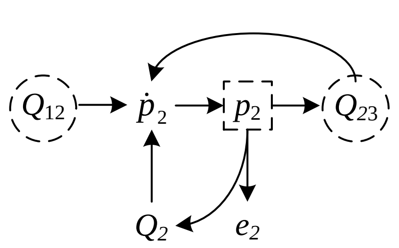
Example 11: To perform prediction in the three-tank system, we need to assign some variables to be in U_P, the set of variables that can be predicted a priori. First, assume that U_P consists of only U, which in this case is $\{Q_{in}, u_{12}, u_{23}\}$. We generate three submodels, one for each EOL constraint. Using the decomposition algorithm for the prediction problem, we find we cannot decompose the global model.[3] To compute any single pressure we need to compute also the other pressures. That is, each submodel has constraints with causal assignments α_2-α_4 and α_{20}-α_{27} with the EOL constraint for that submodel. In this case, it makes sense to simply use the global model (shown in Fig. 9) to compute all three constraints.

Example 12: Now assume that the valves on the connecting pipes are controlled so as to always ensure a constant desired flow through them. In this case, because Q_{12} and Q_{23} are controlled to a known value, their values are known a priori and they can be used as local inputs. So, here, $U_P = U \cup \{Q_{12}, Q_{23}\}$. Now, the decomposition algorithm generates three submodels without overlap. The first submodel uses Q_{in} and Q_{12} as local inputs, and includes α_2, α_{20}, α_{23} and α_{28}. Similarly, the second submodel has only the constraints for the second tank, and the third submodel has only those for the third tank. The corresponding causal graphs are shown in Fig. 10.

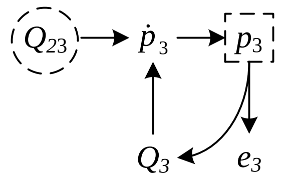
[3]When $U^* \cap Y = \varnothing$, Algorithm 1 simplifies significantly, because in Subroutine 2, there is only ever one constraint to consider, which is the one that in the current causal assignment computes the variable v.



(a) Causal graph for prediction submodel for e_1.



(b) Causal graph for prediction submodel for e_2.



(c) Causal graph for prediction submodel for e_3.

Figure 10. Causal graphs for prediction submodels for the three tank system.

6. Related Work

Several techniques have been proposed for off-line system model decomposition in model-based diagnosis: Analytical Redundancy Relations (ARRs) [21], Possible Conflicts (PCs) [6], Potential Conflicts [22], Minimal Structurally Overdetermined sets of equations (MSOs) [23], among others. All of them are capable of finding the whole set of overdetermined equations in the system model, thus containing enough redundancy to perform fault detection and isolation. The main difference between these approaches comes in the usage of causality information. ARRs and MSOs consider only structural information, i.e., relations between equations and variables in the model. Possible Conflicts introduce causal information and thus identify those submodels that could be computed. The four approaches have been compared in [24], finding out that they are equivalent from a structural point of view. Once causality is introduced, the equivalence can be extended to minimal conflicts [25], which is an on-line technique for computing the source of actual discrepancies in a system, which also induces a model decomposition.

The approach presented in this paper is more similar to the Possible Conflicts approach than the ARRs and MSOs approaches, since we perform the decomposition by assuming a causal assignment in the model. However, important differences arise between the proposed framework and the PCs approach. The most important is related with the applicability of our generalized framework to solve different SHM problems. While PCs only decompose the system model into minimal submodels for estimation, isolation, and identification purposes, our generalized framework also allows to generate minimal submodels for prediction. Also, since we base our decomposition process on a causal model, the algorithm presented here is much more efficient to decompose the problem than the PCs algorithm, which computes evaluable models for all potential causality assignments. Another important difference is the existance and the usage of P, which can be used to drive the decomposition process and does not exist in the PCs approach. Also, our approach generates multi-output submodels, while the PC-based algorithm generates only single-output submodels.

Other approaches have been proposed in the literature for structural model decomposition, but all of them have been developed to provide submodels for a particular SHM function. For example, Williams and Millar presented Moriarty [26], an approach for decomposing a system model into smaller hierarchically organized subsystems. Their decomposition approach generates minimal subsets of equations for estimation from a system mode, and has been applied for parameter estimation, but cannot be used to generate isolation and prediction submodels. Lunze and Schiller [27] presented an approach that, similar to our proposal, uses causal graphs associated to over-constrained systems, however, their approach has only been applied to perform fault diagnosis. More recently, Nyberg [28] proposed a general framework for model-based diagnosis based on structural hypothesis tests, which could be considered as a generalization of structural analysis in the decision theory field.

7. Conclusions

In this paper, we presented a general model decomposition framework, and showed how the concept of model decomposition is pervasive to different SHM methodologies, such as estimation, fault isolation, and prediction. We described the role that model decomposition plays in three specific approaches to estimation, isolation, and prediction, and presented how the different model decomposition problems for these different approaches can be solved within our common framework.

Our model decomposition framework assumes that a single causal assignment exists to the global model that leads to a model that can be implemented and simulated, assuming integral causality. Future work will address derivative causality and algebraic loops. In hybrid systems, the causal assignment can change depending on the system mode, and future work will deal with this aspect as well. Another important consideration for model decomposition when applied to estimation and fault isolation, as shown in those sections, is that the quality of the decomposition depends on the available sensors. Therefore another design aspect is choosing the right set of sensors to get the best model decomposition. We will consider also the extension to the decomposition algorithm to handle auxiliary variables more generally as discussed at the end of Section 2.

Acknowledgments

I. Roychoudhury and M. Daigle's funding for this work was provided by the NASA System-wide Safety and Assurance Technologies (SSAT) Project. A. Bregon and B. Pulido's funding for this work was provided by the Spanish MCI TIN2009-11326 grant.

References

[1] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki, *Diagnosis and Fault-Tolerant Control*. Springer, 2006.

[2] P. J. Mosterman and G. Biswas, "Diagnosis of continuous valued systems in transient operating regions," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 29, no. 6, pp. 554–565, 1999.

[3] M. Daigle and K. Goebel, "Multiple damage progression paths in model-based prognostics," in *Proceedings of the 2011 IEEE Aerospace Conference*, Mar. 2011.

[4] J. Luo, K. R. Pattipati, L. Qiao, and S. Chigusa, "Model-based prognostic techniques applied to a suspension system," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 38, no. 5, pp. 1156 –1168, Sep. 2008.

[5] M. Schwabacher, "A survey of data-driven prognostics," in *Proceedings of the AIAA Infotech@Aerospace Conference*, 2005.

[6] B. Pulido and C. Alonso-González, "Possible conflicts: a compilation technique for consistency-based diagnosis," *IEEE Trans. on Systems, Man, and Cybernetics, Part B, Special Issue on Diagnosis of Complex Systems*, vol. 34, no. 5, pp. 2192–2206, 2004.

[7] A. Bregon, M. Daigle, I. Roychoudhury, G. Biswas, X. Koutsoukos, and B. Pulido, "Improving distributed diagnosis through structural model decomposition," in *Proceedings of the 22nd International Workshop on Principles of Diagnosis*, Oct. 2011, pp. 195–202.

[8] M. Daigle, A. Bregon, and I. Roychoudhury, "Distributed damage estimation for prognostics based on structural model decomposition," in *Proceedings of the Annual Conference of the Prognostics and Health Management Society 2011*, Sep. 2011, pp. 198–208.

[9] ——, "A distributed approach to system-level prognostics," in *Annual Conference of the Prognostics and Health Management Society 2012*, Sep. 2012, pp. 71–82.

[10] A. Bregon, M. Daigle, and I. Roychoudhury, "An integrated framework for model-based distributed diagnosis and prognosis," in *Annual Conference of the Prognostics and Health Management Society 2012*, Sep. 2012, pp. 416–426.

[11] C. Goodrich, S. Narasimhan, M. Daigle, W. Hatfield, R. Johnson, and B. Brown, "Applying model-based diagnosis to a rapid propellant loading system," in *Proceedings of the 20th International Workshop on Principles of Diagnosis*, Jun. 2009, pp. 147–154.

[12] S. Narasimhan and G. Biswas, "Model-based diagnosis

of hybrid systems," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 37, no. 3, pp. 348 –361, may 2007.

[13] A. Bregon, G. Biswas, and B. Pulido, "A decomposition method for nonlinear parameter estimation in TRANSCEND," *IEEE Trans. on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 42, no. 3, pp. 751–763, May 2012.

[14] M. Daigle, A. Bregon, G. Biswas, X. Koutsoukos, and B. Pulido, "Improving multiple fault diagnosability using possible conflicts," in *Proceedings of the 8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, Aug. 2012, pp. 144–149.

[15] M. J. Daigle, X. Koutsoukos, and G. Biswas, "A qualitative event-based approach to continuous systems diagnosis," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 4, pp. 780–793, Jul. 2009.

[16] M. Daigle, A. Bregon, and I. Roychoudhury, "Qualitative Event-based Diagnosis with Possible Conflicts: Case Study on the Third International Diagnostic Competition," in *Proceedings of the 22nd International Workshop on Principles of Diagnosis*, Murnau, Germany, Oct 2011, pp. 285–292.

[17] I. Roychoudhury, G. Biswas, and X. Koutsoukos, "Designing distributed diagnosers for complex continuous systems," *IEEE Transactions on Automation Science and Engineering*, vol. 6, no. 2, pp. 277–290, Apr. 2009.

[18] B. Saha and K. Goebel, "Modeling Li-ion battery capacity depletion in a particle filtering framework," in *Proceedings of the Annual Conference of the Prognostics and Health Management Society 2009*, Sep. 2009.

[19] M. Daigle and K. Goebel, "A model-based prognostics approach applied to pneumatic valves," *International Journal of Prognostics and Health Management*, vol. 2, no. 2, Aug. 2011.

[20] M. Orchard and G. Vachtsevanos, "A particle filtering approach for on-line fault diagnosis and failure prognosis," *Transactions of the Institute of Measurement and Control*, no. 3-4, pp. 221–246, Jun. 2009.

[21] M. Staroswiecki, *Structural analysis for fault detection and isolation and for fault tolerant control*, ser. Fault Diagnosis and Fault Tolerant Control, Oxford, UK, 2002, ch. Encyclopedia of Life Support Systems.

[22] M.-O. Cordier, P. Dague, F. Lévy, J. Montmain, M. Staroswiecki, and L. Travé-Massuyès, "Conflicts versus analytical redundancy relations : A comparative analysis of the model-based diagnostic approach from the artificial intelligence and automatic control perspectives," *IEEE Trans. Syst. Man Cy. B.*, vol. 34, no. 5, pp. 2163–2177, 2004.

[23] M. Krysander, J. Åslund, and M. Nyberg, "An efficient algorithm for finding minimal over-constrained subsystems for model-based diagnosis," *IEEE Trans. on Systems, Man, and Cybernetics, Part A*, vol. 38, no. 1, 2008.

[24] J. Armengol, A. Bregon, T. Escobet, E. Gelso, M. Krysander, M. Nyberg, X. Olive, B. Pulido, and L. Travé-Massuyès, "Minimal Structurally Overdetermined sets for residual generation: A comparison of alternative approaches," in *Proceedings of the 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, SAFEPROCESS09*, Barcelona, Spain, 2009, pp. 1480–1485.

[25] R. Reiter, "A theory of diagnosis from first principles," *Artificial Intelligence*, vol. 32, pp. 57–95, 1987.

[26] B. Williams and B. Millar, "Decompositional model-based learning and its analogy to diagnosis," in *Proc. of the Fifteenth National Conference on Artificial Intelligence*, 1998, pp. 197–204.

[27] J. Lunze and F. Schiller, "Logic-based process diagnosis utilising the causal structure of dynamical systems," *Annual Review in Automatic Programming*, vol. 17, no. 0, pp. 279 – 285, 1992.

[28] M. Nyberg, "A general framework for fault diagnosis based on statistical hypothesis testing," in *Proceedings of the Twelfth International Workshop on Principles of Diagnosis (DX-01)*, Sansicario, Italy, 2001, pp. 135–142.

BIOGRAPHY

Indranil Roychoudhury *received the B.E. (Hons.) degree in Electrical and Electronics Engineering from Birla Institute of Technology and Science, Pilani, Rajasthan, India in 2004, and the M.S. and Ph.D. degrees in Computer Science from Vanderbilt University, Nashville, Tennessee, USA, in 2006 and 2009, respectively. Since August 2009, he has been with SGT, Inc., at NASA Ames Research Center as a Computer Scientist. Dr. Roychoudhury is a member of the Prognostics and Health Management Society and the IEEE. His research interests include hybrid systems modeling, model-based diagnostics and prognostics, distributed diagnostics and prognostics, and Bayesian diagnostics of complex physical systems.*

Matthew Daigle *received the B.S. degree in Computer Science and Computer and Systems Engineering from Rensselaer Polytechnic Institute, Troy, NY, in 2004, and the M.S. and Ph.D. degrees in Computer Science from Vanderbilt University, Nashville, TN, in 2006 and 2008, respectively. From September 2004 to May 2008, he was a Graduate Research Assistant with the Institute for Software Integrated Systems and Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, TN. During the summers of 2006 and 2007, he was an intern with Mission Critical Technologies, Inc., at NASA Ames Research Center. From June 2008 to December 2011, he was an Associate Scientist with the University of California, Santa Cruz, at NASA Ames Research Center. Since January 2012, he has been with NASA Ames Research Center as a Research Computer Scientist. His current research interests include physics-based modeling, model-based diagnosis and prognosis, simulation, and hybrid systems. Dr. Daigle is a member of the Prognostics and Health Management Society and the IEEE.*

***Anibal Bregon** received his B.Sc., M.Sc., and Ph.D. degrees in Computer Science from the University of Valladolid, Spain, in 2005, 2007, and 2010, respectively. From September 2005 to June 2010, he was Graduate Research Assistant with the Intelligent Systems Group at the University of Valladolid, Spain. He has been visiting researcher at the Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, USA; the Dept. of Electrical Engineering, Linkoping University, Linkoping, Sweden; and the Diagnostics and Prognostics Group, NASA Ames Research Center, Mountain View, CA, USA. Since September 2010, he has been Assistant Professor and Research Scientist at the Department of Computer Science from the University of Valladolid. Dr. Bregon is a member of the Prognostics and Health Management Society and the IEEE. His current research interests include model-based reasoning for diagnosis, prognostics, health-management, and distributed diagnosis of complex physical systems.*

***Belarmino Pulido** received his degree, MsC degree, and PhD degree in Computer Science from the University of Valladolid, Valladolid, Spain, in 1992, 1995, and 2001 respectively. In 1994 he joined the Departmento de Informática in the University of Valladolid, where he is Associate Professor since 2002. His main research interests are Model-based reasoning and Knowledge-based reasoning, and their application to Supervision and Diagnosis. He has worked in different national and European funded projects related to Supervision and Diagnosis of complex industrial systems. He is member of different professional associations (IEEE since 2000, ACM since 2003, CAEPIA -Spanish section of ECCAI- since 1997). Moreover, he is the coordinator of the Spanish Network on Supervision and Diagnosis of Complex Systems since 2005.*