Linköping University Post Print

**Residual Generators for Fault Diagnosis Using
Computation Sequences With Mixed Causality
Applied to Automotive Systems**

Carl Svärd and Mattias Nyberg

N.B.: When citing this work, cite the original article.

©2010 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Carl Svärd and Mattias Nyberg, Residual Generators for Fault Diagnosis Using Computation Sequences With Mixed Causality Applied to Automotive Systems, 2010, IEEE TRANSACTIONS ON SYSTEMS MAN AND CYBERNETICS PART A-SYSTEMS AND HUMANS, (40), 6, 1310-1328.

<http://dx.doi.org/10.1109/TSMCA.2010.2049993>

Postprint available at: Linköping University Electronic Press

<http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-61313>

Residual Generators for Fault Diagnosis Using Computation Sequences With Mixed Causality Applied to Automotive Systems

Carl Svärd and Mattias Nyberg

Abstract—An essential step in the design of a model-based diagnosis system is to find a set of residual generators fulfilling stated fault detection and isolation requirements. To be able to find a good set, it is desirable that the method used for residual generation gives as many candidate residual generators as possible, given a model. This paper presents a novel residual generation method that enables simultaneous use of integral and derivative causality, i.e., mixed causality, and also handles equation sets corresponding to algebraic and differential loops in a systematic manner. The method relies on a formal framework for computing unknown variables according to a computation sequence. In this framework, mixed causality is utilized, and the analytical properties of the equations in the model, as well as the available tools for algebraic equation solving, are taken into account. The proposed method is applied to two models of automotive systems, a Scania diesel engine, and a hydraulic braking system. Significantly more residual generators are found with the proposed method in comparison with methods using solely integral or derivative causality.

Index Terms—Fault diagnosis, model-based diagnosis, nonlinear systems, residual generation.

I. INTRODUCTION

FAULT diagnosis of technical systems has become increasingly important with the rising demand for reliability and safety, driven by environmental and economical incentives. One example is automotive engines that are by regulations required to have high-precision onboard diagnosis of failures that are harmful to the environment [1].

To obtain good detection and isolation of faults, model-based fault diagnosis is necessary. In the fault detection and isolation approach to model-based fault diagnosis, *residuals* are used to detect and isolate faults present in the system; e.g., see [2]. Residuals are signals that are ideally zero in the nonfaulty case and nonzero otherwise and are typically generated by utilizing a mathematical model of the system and measurements.

In this paper, we have the view that the design of diagnosis systems is a two-step approach, as elaborated in [3] and [4]. In

the first step, a large number of candidate residual generators are found, and in the second step, the residual generators most suitable to be included in the final diagnosis system are picked out. Since different residual generators have different properties regarding fault and noise sensitivities, it is important for the second step that there is a large selection of different residual generator candidates to choose from. Thus, the initial set of candidate residual generators should be as large as possible.

A residual generator design approach [5] that has shown to be successful in real applications [6]–[10] is to compute unknown variables in the model by solving equation sets one at a time in a sequence, i.e., according to a computation sequence, and then evaluate a redundant equation to obtain a residual. To determine from which equations and in which order the unknown variables should be computed, *structural analysis* is utilized. In addition to [5], similar approaches have been described and exploited in, for example, [2] and [11]–[15].

In the works mentioned above, the approach is to apply either integral or derivative causality [2] for differential equations. However, as will be illustrated in this paper through application studies, it is advantageous to allow the simultaneous use of integral and derivative causality, i.e., mixed causality. Furthermore, real-world applications involve complex models that give rise to algebraic and differential loops or cycles [2], [16], corresponding to sets of equations that have to be simultaneously treated. Thus, it is desirable that a method for residual generation is able to handle mixed causality and equation sets corresponding to algebraic and differential loops. The intention with the following simple example is to illustrate these issues. Consider the following set of differential–algebraic equations (DAEs):

$$\begin{aligned} e_1 : \quad & \dot{x}_1 - x_2 = 0 \\ e_2 : \quad & \dot{x}_3 - x_4 = 0 \\ e_3 : \quad & \dot{x}_4 x_1 + 2x_2 x_4 - y_1 = 0 \\ e_4 : \quad & x_3 - y_3 = 0 \\ e_5 : \quad & x_2 - y_2 = 0 \end{aligned} \quad (1)$$

which is a subsystem of a model describing the planar motion of a point-mass satellite [17], [18] and where x_1 , x_2 , x_3 , and x_4 are unknown variables, and y_1 , y_2 , and y_3 are known variables. Assume that we want to use e_5 as residual. This implies that the unknown variables x_1 , x_2 , x_3 , and x_4 must be computed from e_1 , e_2 , e_3 , and e_4 . A *structure*, i.e., which unknown

Manuscript received April 8, 2009. Date of publication June 21, 2010; date of current version October 15, 2010. This work was supported in part by Scania CV AB and in part by the Swedish Governmental Agency for Innovation Systems (VINNOVA). This paper was recommended by Associate Editor M. Jeng.

C. Svärd is with the Department of Electrical Engineering, Linköping University, 58 183 Linköping, Sweden (e-mail: carl@isy.liu.se).

M. Nyberg is with the Department of Electrical Engineering, Linköping University, 58 183 Linköping, Sweden, and also with Scania CV AB, 151 87 Södertälje, Sweden (e-mail: matny@isy.liu.se).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TSMCA.2010.2049993

variables are contained in which equations, of the equation set $\{e_1, e_2, e_3, e_4\}$ with respect to $\{x_1, x_2, x_3, x_4\}$, in permuted form, is depicted in the following:

	x_3	x_4	x_2	x_1
e_4	1			
e_2	1	1		
e_3		1	1	1
e_1			1	1

(2)

This structure reveals the order and from which equations, marked with bold, the unknown variables should be computed. It is clear that the computation of the variables will involve handling of the differential loop arising in the equation set $\{e_1, e_3\}$, since to compute x_2 the value of x_1 is needed and vice versa. Furthermore, the computation of the variables according to (2) will require use of mixed causality: derivative causality when solving for x_4 in e_2 and integral causality when solving for x_1 in e_1 .

The main contribution of this paper is a novel method for residual generation that enables simultaneous use of integral and derivative causality, and it is able to handle equation sets corresponding to algebraic and differential loops in a systematic manner. In this sense, the proposed method also generalizes previous methods for residual generation, e.g., [2], [5]–[8], and [11]–[15]. To achieve this, a formal framework for sequential computation of variables is presented. In this framework, tools for equation solving and approximate differentiation, as well as analytical and structural properties of the equations in the model, are essential.

In Section II, some preliminaries, basic theories, and references regarding structural analysis and DAE systems are given. Section III presents the framework for sequential computation of variables, in which the concepts *block-lower-triangular (BLT) semiexplicit DAE form*, *tools*, and *computation sequence* are important. Tools, or more precisely algebraic-equation-solving tools (AE tools), are crucial for the ability to handle loops. In Section IV, it is shown how a computation sequence is utilized for residual generation. The resulting residual generator is referred to as a *sequential residual generator*. Motivated by implementation aspects, the concept of a *proper sequential residual generator* is introduced as a sequential residual generator in which no unnecessary variables are computed and in which computations are performed from as small equation sets as possible. A necessary condition for the existence of a proper sequential residual generator is derived, connecting proper sequential residual generators with *minimal structurally overdetermined* (MSO) equation sets [19]. An algorithm able to find proper sequential residual generators, given a model and a set of tools, is outlined. A key step in the algorithm is to find minimal and irreducible computation sequences, which is considered in Section V. In Section VI, the proposed method for residual generation is applied to models of an automotive diesel engine and an auxiliary hydraulic braking system. The application studies clearly show the benefits of using a mixed-causality approach and handling algebraic and differential loops. Finally, Section VII concludes the paper. For readability, proofs to all lemmas and theorems are collected in Appendix A.

II. PRELIMINARIES AND BACKGROUND THEORY

Consider a *model* $M(E, X, Y)$, or M for short, consisting of a set of equations $E = \{e_1, e_2, \dots, e_m\}$ relating a set of unknown variables $X = \{x_1, x_2, \dots, x_n\}$ and a set of known, i.e., measured, variables $Y = \{y_1, y_2, \dots, y_r\}$. Introduce a third variable set $D = \{\dot{x}_1, \dot{x}_2, \dots, \dot{x}_n\}$, containing the (time) derivatives of the variables in X . Without loss of generality, it is assumed that the equations in E are in the following form:

$$e_i : f_i(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{y}) = 0, \quad i = 1, 2, \dots, m \quad (3)$$

where $\dot{\mathbf{x}} = (\dot{x}_1, \dot{x}_2, \dots, \dot{x}_n)$ is a vector of the variables in D , $\mathbf{x} = (x_1, x_2, \dots, x_n)$ is a vector of the variables in X , and $\mathbf{y} = (y_1, y_2, \dots, y_r)$ is a vector of the variables in Y . Furthermore, without loss of generality, it is assumed that each equation $e_i \in E$ contains, at most, one differentiated variable $\dot{x}_j \in D$ and that $\dot{x}_j$ is contained only in one equation. This assumption can be made without loss of generality, since an equation containing more than one differentiated variable can always be written as an equation with only one differentiated variable by introducing new algebraic variables and adding trivial differential equations. For an example, consider the equation $\dot{x}_1 + \dot{x}_2 + x_1 = 0$ containing two differentiated variables. By introducing the algebraic variable x_3 , substituting $\dot{x}_2$ with x_3 , and then adding the equation $x_3 = \dot{x}_2$, the equation can be written as $\dot{x}_1 + x_3 + x_1 = 0$. This equation now contains only one differentiated variable.

Define the set of trajectories of the variables in Y that are *consistent* with the model $M(E, X, Y)$ as

$$\mathcal{O}(M) = \{\mathbf{y} : \exists \mathbf{x} f_i(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{y}) = 0, i = 1, 2, \dots, m\}. \quad (4)$$

The set $\mathcal{O}(M)$ is the *observation set* of the model M . We formally define a residual generator as follows.

Definition 1 (Residual Generator): A system with input $\mathbf{y}$ and output r is a *residual generator* for the model $M(E, X, Y)$, and r is a residual if $\mathbf{y} \in \mathcal{O}(M) \Rightarrow \lim_{t \rightarrow \infty} r \rightarrow 0$. $\square$

A. Integral and Derivative Causality

In the context of the methods for residual generation mentioned in Section I, there are two approaches for handling differential equations, referred to as *integral* and *derivative* causality; e.g., see [2]. When adopting integral causality, the differentiated variables or states of a differential equation can be computed. The use of integral causality hence relies on the assumption that ordinary differential equations (ODEs) can be solved, i.e., integrated, which, in general, requires that initial conditions of the states are known. Integral causality is used in, for example, [13] and [20].

If, instead, derivative causality is applied, a differential equation is interpreted as an algebraic equation, and only undifferentiated, i.e., algebraic, variables can be computed. Usage of derivative causality thus relies on the assumption that values of the differentiated variables in a differential equation are available. This requires, in general, that derivatives of known or previously computed variables can be computed or estimated. Derivative causality is used in [12] and also adopted in, for

example, [6]. The difference between integral and derivative causality is discussed in [21] and from a simulation point of view in [22]. Causality also plays a central role when using a bond-graph modeling framework; e.g., see [23].

The chosen causality approach naturally influences which variables can be computed from an equation set. For instance, consider the differential equation $e_1 : \dot{x}_1 - x_2 = 0$ from (1), where both x_1 and x_2 are unknown variables. If integral causality is used, x_1 can be computed from e_1 , but if, instead, derivative causality is used, x_2 can be computed from e_1 .

B. Structure of Equation Sets

To study which unknown variables are contained in a set of equations, a structural representation of the equation set will be used. Let $E' \subseteq E$ and introduce the following notations:

$$\text{var}_X(E') = \left\{ x_j \in X : \exists e_i \in E', \frac{\partial f_i}{\partial x_j} \neq 0 \vee \frac{\partial f_i}{\partial \dot{x}_j} \neq 0 \right\}$$

$$\text{var}_D(E') = \left\{ \dot{x}_j \in D : \exists e_i \in E', \frac{\partial f_i}{\partial \dot{x}_j} \neq 0 \right\}.$$

Consider the model (1) and let $X = \{x_1, x_2, x_3, x_4\}$ and $D = \{\dot{x}_1, \dot{x}_2, \dot{x}_3, \dot{x}_4\}$. For instance, it holds that

$$\text{var}_X(\{e_3\}) = \{x_1, x_2, x_4\}. \quad (5)$$

Let $G = (E, X, A)$ be a bipartite graph, where E and X are the (disjoint) sets of vertices, and

$$A = \{(e_i, x_j) \in E \times X : x_j \in \text{var}_X(\{e_i\})\}$$

is the set of arcs. We will call the bipartite graph G the *structure* of the equation set E with respect to X . Note that, with this representation, there is no structural difference between the variable x_j and the differentiated variable $\dot{x}_j$. An equivalent representation of G is the $m \times n$ biadjacency matrix B defined as

$$B_{ij} = \begin{cases} 1, & \text{if } (e_i, x_j) \in A \\ 0, & \text{otherwise.} \end{cases}$$

Return to the model (1). The structure of the equation set $\{e_1, e_2, e_3, e_4\}$ with respect to $\{x_1, x_2, x_3, x_4\}$ is given by the biadjacency matrix (2). The result in (5) corresponds to the third row of (2).

We will also consider the structure of E with respect to D , which refers to the bipartite graph $\bar{G} = (E, D, \bar{A})$, where

$$\bar{A} = \{(e_i, \dot{x}_j) \in E \times D : \dot{x}_j \in \text{var}_D(\{e_i\})\}.$$

C. Structural Decomposition

A *matching* on the bipartite graph $G = (E, X, A)$ is a subset of A such that no two arcs have common vertices. A matching with maximum cardinality is a *maximum matching*. A matching is a *complete matching* with respect to E (or X) if the matching covers every vertex in E (or X). By directing the arcs contained in a matching on the bipartite graph G in one direction and the remaining arcs in the opposite direction, a *directed graph* can be obtained from G ; e.g., see [24]. A directed graph is said to be *strongly connected* if, for every pair of vertices x_i

Fig. 1. DM decomposition of the bipartite graph $G = (E, X, A)$. The DM components $G_i^0 = (E_i^0, X_i^0, A_i^0)$ correspond to the SCCs of the structure of E^0 with respect to X^0 .

and x_j , there is a directed path from x_i to x_j . The maximal strongly connected subgraphs of a directed graph are called its *strongly connected components* (SCCs).

There exists a unique structural decomposition of the bipartite graph $G = (E, X, A)$, referred to as the Dulmage–Mendelsohn (DM) decomposition; see [25] and [26]. It decomposes G into irreducible bipartite subgraphs $G^+ = (E^+, X^+, A^+)$, $G_i^0 = (E_i^0, X_i^0, A_i^0)$, $i = 1, 2, \dots, s$, and $G^- = (E^-, X^-, A^-)$, called DM components; see Fig. 1. The component G^+ is the overdetermined part of G , $G^0 = \bigcup_{i=1}^s G_i^0$ is the just-determined part, and G^- is the underdetermined part. The DM components $G_i^0 = (E_i^0, X_i^0, A_i^0)$ correspond to the SCCs of the directed graph induced by any complete matching on the bipartite graph G^0 [26]. The equation set $E^0 = \bigcup_{i=1}^s E_i^0$ is said to be a just-determined equation set with respect to the variables $X^0 = \bigcup_{i=1}^s X_i^0$. For an application of the DM decomposition, see, for example, [27].

1) *Algebraic and Differential Loops*: If the structure of an equation set, with respect to a set of unknown variables, contains SCCs of larger size than one, the equation set contains *loops* or *cycles*; e.g., see [2], [16], and [21]. If the equation set contains cyclic dependence including unknown differentiated variables, the loop is said to be *differential*; else, it is *algebraic*.

In the example outlined in Section I, the structure (2), which, in fact, is the result of a DM decomposition, revealed three SCCs that are bold marked. The SCCs are $(\{e_4\}, \{x_3\})$, $(\{e_2\}, \{x_4\})$, and $(\{e_1, e_3\}, \{x_1, x_2\})$ of size one, one, and two, respectively. The latter corresponds to a differential loop.

D. DAE Systems

Due to its general form, it is assumed that the model (3) contains both differential and algebraic equations, i.e., it is a DAE system or descriptor system [28]–[30]. The most general form of a DAE is $f(\dot{x}, x, y) = 0$, where f is some vector-valued function [cf. (3)]. DAEs appear in large classes of technical systems like mechanical, electrical, and chemical systems. Furthermore, DAEs are also the result when using physically based object-oriented modeling tools, e.g., Modelica [31].

1) *Differential Index*: A common approach when analyzing and solving general DAE systems is to seek a reformulation of the original DAE into a simpler and well-structured description with the same set of solutions [28], [29]. To classify how difficult such a reformulation is, the concept of index has been introduced. There are different index concepts, depending on the kind of reformulation that is sought. In this paper, we will use the *differential index*, which is defined as the minimum number of times that all or parts of the DAE must be differentiated with respect to time in order to write the DAE as an *explicit* ODE $\dot{\mathbf{x}} = \mathbf{g}(\mathbf{x}, \mathbf{y})$; see, for example, [29].

2) *Semiexplicit DAEs*: An important class of DAEs are *semiexplicit* DAEs, i.e.,

$$\dot{\mathbf{z}} = \mathbf{g}(\mathbf{z}, \mathbf{w}, \mathbf{y}) \quad (6a)$$

$$0 = \mathbf{h}(\mathbf{z}, \mathbf{w}, \mathbf{y}) \quad (6b)$$

where $\mathbf{z}$ and $\mathbf{w}$ are vectors of unknown variables, and $\mathbf{y}$ is a vector of known variables. A semiexplicit DAE is of index one if and only if (6b) can (locally) be solved for $\mathbf{w}$ so that $\mathbf{w} = \tilde{\mathbf{h}}(\mathbf{z}, \mathbf{y})$; e.g., see [29]. An explicit ODE can easily be obtained from a semiexplicit DAE of index one by substituting $\mathbf{w} = \tilde{\mathbf{h}}(\mathbf{z}, \mathbf{y})$ into (6a).

III. SEQUENTIAL COMPUTATION OF VARIABLES

In this section, a framework for sequential computation of variables is presented. The framework is built upon the concepts BLT semiexplicit DAE form, tools, and computation sequence. The small model (1) introduced in Section I, will be used as a running example to illustrate and exemplify the theory.

Large sets of equations often have a sparse structure, i.e., only a few unknown variables in each equation. This makes it possible to partition the set of equations into subsets that can be solved, in a sequence, for only a subset of the unknowns. The main argument for computing variables in this way is efficiency, and in some cases, this may be the only feasible way to compute the unknowns. This approach has been used in the context of equation solving (see [32]–[34]) and is also utilized in methods for noncausal simulation [35].

A. BLT Semiexplicit DAE Form

One property that the partitioning must fulfill is that the computation of variables from a certain subset of equations must only use variables that are known, that is, measured or have been computed from another subset in a previous step of the sequence.

Furthermore, with the efficiency argument in mind, it is most desirable to partition the set of equations into as small blocks, i.e., subsets, as possible. However, even if the equation set has a sparse structure, there could be algebraic or differential loops, which makes it impossible to consider subsets of solely one equation.

In addition, it is desirable that the equations are partitioned into blocks or subsets from which variables can be computed in a straightforward manner. Since the considered set of equations (3) contains both differential and algebraic equations, subsets

will correspond to DAEs. The computation of variables from semiexplicit DAEs of index one, referred to as simulation of the DAE, is a well-studied problem, and several methods exist; e.g., see [30] and [36]. Furthermore, as said in Section II-D-2, a semiexplicit DAE of index one can trivially be transformed to an explicit ODE. Explicit ODEs are suitable for real-time simulation in embedded systems, for example, engine control units, because real-time simulation often require the use of an explicit integration method, e.g., forward Euler [30], which assumes an explicit ODE. For a detailed discussion regarding real-time simulation, see [37].

Motivated by these arguments, we consider a partitioning of the equation set so that a BLT form is achieved, where each block corresponds to a semiexplicit DAE of index one.

Definition 2 (BLT Semiexplicit DAE Form): The system

$$\begin{aligned} \dot{\mathbf{z}}_1 &= \mathbf{g}_1(\mathbf{z}_1, \mathbf{w}_1, \mathbf{y}) \\ \dot{\mathbf{z}}_2 &= \mathbf{g}_2(\mathbf{z}_1, \mathbf{z}_2, \mathbf{w}_1, \mathbf{w}_2, \mathbf{y}) \\ &\vdots \\ \dot{\mathbf{z}}_s &= \mathbf{g}_s(\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_s, \mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_s, \mathbf{y}) \end{aligned} \quad (7)$$

where $\mathbf{w}_i = (\mathbf{w}_i^1, \mathbf{w}_i^2, \dots, \mathbf{w}_i^{p_i})$, and

$$\begin{aligned} \mathbf{w}_i^1 &= \mathbf{h}_i^1(\Psi_i, \mathbf{y}) \\ \mathbf{w}_i^2 &= \mathbf{h}_i^2(\Psi_i, \mathbf{w}_i^1, \mathbf{y}) \\ &\vdots \\ \mathbf{w}_i^{p_i} &= \mathbf{h}_i^{p_i}(\Psi_i, \mathbf{w}_i^1, \dots, \mathbf{w}_i^{p_i-1}, \mathbf{y}) \end{aligned}$$

where

$$\Psi_i = (\dot{\mathbf{w}}_1, \dot{\mathbf{w}}_2, \dots, \dot{\mathbf{w}}_{i-1}, \mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_i, \mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{i-1})$$

for $i = 1, 2, \dots, s$, and where $\mathbf{z}_i$ and $\mathbf{w}_i$ are vectors of unknown variables, all pairwise disjoint, and $\mathbf{y}$ is a vector of known variables, is in BLT semiexplicit DAE form. $\square$

Note that it is not necessary that both $\mathbf{z}_i$ and $\mathbf{w}_i$ are present in (7) for every $i = 1, 2, \dots, s$. In particular, the system

$$\begin{aligned} \mathbf{w}_1 &= \mathbf{h}_1(\mathbf{y}) \\ \mathbf{w}_2 &= \mathbf{h}_2(\mathbf{w}_1, \mathbf{y}) \\ &\vdots \\ \mathbf{w}_s &= \mathbf{h}_s(\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{s-1}, \mathbf{y}) \end{aligned}$$

containing no differentiated variables at all, is also in BLT semiexplicit DAE form.

1) *Some Properties of the BLT Semiexplicit DAE Form*: Consider the following system:

$$\dot{\mathbf{z}}_1 = \mathbf{g}_1(\mathbf{z}_1, \mathbf{w}_1, \mathbf{y}) \quad (8a)$$

$$\mathbf{w}_1^1 = \mathbf{h}_1^1(\mathbf{z}_1, \mathbf{y}) \quad (8b)$$

$$\mathbf{w}_1^2 = \mathbf{h}_1^2(\mathbf{z}_1, \mathbf{w}_1^1, \mathbf{y}) \quad (8c)$$

$$\dot{\mathbf{z}}_2 = \mathbf{g}_2(\mathbf{z}_1, \mathbf{z}_2, \mathbf{w}_1, \mathbf{w}_2, \mathbf{y}) \quad (8d)$$

$$\mathbf{w}_2^1 = \mathbf{h}_2^1(\dot{\mathbf{w}}_1, \mathbf{z}_1, \mathbf{z}_2, \mathbf{w}_1, \mathbf{y}) \quad (8e)$$

$$\mathbf{w}_2^2 = \mathbf{h}_2^2(\dot{\mathbf{w}}_1, \mathbf{z}_1, \mathbf{z}_2, \mathbf{w}_1, \mathbf{w}_2^1, \mathbf{y}) \quad (8f)$$

where $\mathbf{w}_1 = (\mathbf{w}_1^1, \mathbf{w}_1^2)$, and $\mathbf{w}_2 = (\mathbf{w}_2^1, \mathbf{w}_2^2)$, which is in BLT semiexplicit DAE form with $s = 2$ and $p_1 = p_2 = 2$. By studying the system (8), we can deduce some properties of the BLT semiexplicit DAE form:

a) *Mixed causality*: The form generalizes the use of integral and derivative causality, since, for example, integral causality is used in (8a) and derivative causality in (8e).

b) *Blocks are DAEs of index one or zero*: Each block, e.g., (8a)–(8c), corresponds to a semiexplicit DAE of, at most, index one with respect to the unknown variables in each block, i.e., $\mathbf{z}_1$ and $\mathbf{w}_1$ in the first block and $\mathbf{z}_2$ and $\mathbf{w}_2$ in the second block. Note that, in accordance with the note above, vectors $\mathbf{z}_1$, $\mathbf{z}_2$, $\mathbf{w}_1$, and $\mathbf{w}_2$ must not all be present in (8). If, for instance, $\mathbf{w}_1$ is missing and, hence, also (8b) and (8c), the first block is an explicit ODE, i.e., a DAE of index zero. If both $\mathbf{z}_1$ and $\mathbf{w}_1$ are present, the first block corresponds to a semiexplicit DAE of index one.

c) *Transformation to ODE*: Due to the previous property, a system in BLT semiexplicit DAE form can trivially be transformed to a variant of an explicit ODE. In (8), we may substitute (8b) into (8c) and then substitute the result along with (8b) into (8a) so that we obtain

$$\begin{aligned}\dot{\mathbf{z}}_1 &= \mathbf{g}_1(\mathbf{z}_1, \mathbf{w}_1, \mathbf{y}) \\ &= \mathbf{g}_1(\mathbf{z}_1, [\mathbf{h}_1^1(\mathbf{z}_1, \mathbf{y}), \mathbf{h}_1^2(\mathbf{z}_1, \mathbf{h}_1^1(\mathbf{z}_1, \mathbf{y}))], \mathbf{y}) \\ &= \tilde{\mathbf{g}}_1(\mathbf{z}_1, \mathbf{y})\end{aligned}$$

and then repeat the procedure for the second block to obtain

$$\begin{aligned}\dot{\mathbf{z}}_1 &= \tilde{\mathbf{g}}_1(\mathbf{z}_1, \mathbf{y}) \\ \dot{\mathbf{z}}_2 &= \tilde{\mathbf{g}}_2(\mathbf{z}_1, \mathbf{z}_2, \dot{\mathbf{y}}, \mathbf{y}).\end{aligned}$$

As said above, ODEs may be preferable in real-time applications.

d) *Blocks are SCCs*: Each block in the BLT semiexplicit DAE form is an SCC of the structure of the corresponding equations with respect to the unknown variables in that block. This can be seen by studying the structure¹ of the equations in (8) with respect to the variables $\{\mathbf{z}_1, \mathbf{w}_1^1, \mathbf{w}_1^2, \mathbf{z}_2, \mathbf{w}_2^1, \mathbf{w}_2^2\}$, which is shown in (9). In this structure, the equation in (8a) has been named e_1 , the equation in (8b) has been named e_2 , and so forth

	$\mathbf{z}_1$	$\mathbf{w}_1^1$	$\mathbf{w}_1^2$	$\mathbf{z}_2$	$\mathbf{w}_2^1$	$\mathbf{w}_2^2$
e_1	1	1	1			
e_2	1	1				
e_3	1	1	1			
e_4	1	1	1	1	1	1
e_5	1	1	1	1	1	
e_6	1	1	1	1	1	1

(9)

e) *Efficiency*: Recall the discussion regarding efficiency in the beginning of Section III-A. As a consequence of the previous property, the original set of equations is partitioned

in as small blocks as possible, in the sense that there is no dependence between blocks, i.e., no loops occur.

f) *Sequential computation of variables*: The BLT structure makes it possible to sequentially compute variables by considering the blocks one at a time, starting from the first block. Since the structure guarantees that a certain block only contains unknown variables from the present and previous blocks.

B. Computational Tools

Whether a system in BLT semiexplicit DAE form can be obtained from a given set of equations and whether trajectories of the unknown variables can be computed from the resulting system naturally depend on the properties of the equations in the model. Equally important is also the set of *tools* that are available for use.

Consider the BLT semiexplicit DAE form (8). To obtain, for example, the function $\mathbf{h}_1^1$ in (8b) from a subset of equations given in the model, some kind of tool for algebraic equation solving is needed. To compute a trajectory of the variable $\mathbf{z}_1$ from (8a), a differential equation must be solved, and hence, a tool for this is needed. Furthermore, to obtain the derivative $\dot{\mathbf{w}}_1$, which is present in (8e), from the trajectory of $\mathbf{w}_1$ computed in (8b) and (8c), a tool for differentiation is needed.

Motivated by this discussion, we consider three types of tools: AE tools, differential-equation-solving tools, and differentiation tools.

1) *AE Tools*: A tool for algebraic equation solving is typically some software package for symbolic or numeric solving of linear or nonlinear algebraic equations. AE tools are essential for handling models containing algebraic loops. If, for example, the available AE tool can only solve scalar equations, loops cannot be handled.

More precisely, an AE tool is a function taking a set of variables $V_i \subseteq X \cup D$ and a set of equations $E_i \subseteq E$ as arguments and returning a function $\mathbf{g}_i$, which can be a symbolic expression or numeric algorithm, taking variables from $\{X \cup D\} \setminus V_i$ and Y as arguments and returning a vector corresponding to the elements in V_i . Now, assume that $\mathbf{g}_i$ is the function returned by an AE tool when V_i and E_i are used as arguments and that the equation set $\bar{E}_i$ corresponds to $\mathbf{v}_i = \mathbf{g}_i(\mathbf{u}_i, \mathbf{y})$, where $\mathbf{v}_i$ is a vector of the elements in V_i , $\mathbf{u}_i$ is a vector of the elements in $U_i \subseteq \{X \cup D\} \setminus V_i$, and $\mathbf{y}$ is a vector of known variables. A natural assumption regarding an AE tool, whatever algorithm or method it corresponds to, is that the AE tool should not introduce new solutions. That is, a solution to $\bar{E}_i$ should also be a solution to the original equation set E_i . Moreover, an AE tool should neither remove solutions, i.e., solutions to E_i must also be solutions to $\bar{E}_i$. Furthermore, motivated by the idea of using sequential computation of variables for residual equations, we are interested in unique solutions. This discussion justifies the following assumption.

Assumption 1: Given U_i and $\mathbf{y}$, the solution sets of $\bar{E}_i$, obtained from the AE tool, and E_i , with respect to V_i , are equal and unique. $\square$

AE tools giving unique solutions generally assume that the given set of equations contains as many equations as unknown

¹It is assumed here that $f(x)$ implies $\partial f / \partial x \neq 0$.

variables. One example is Newton iteration, which is a common numerical method for solving nonlinear equations; e.g., see [38]. In addition, underdetermined and overdetermined sets of equations for which a unique analytical solution exists are rare. This motivates the following assumption.

Assumption 2: An AE tool requires that its arguments V_i and E_i correspond to a just-determined equation set. $\square$

In this paper, we assume that tools for algebraic equation solving are available through existing standard software packages like Maple or Mathematica, and the design and implementation of such tools will not be considered. For solving algebraic loops, *tearing* [33], [34] can also be a successful approach. In the following, we also assume that AE tools fulfill the properties stated in Assumptions 1 and 2.

2) *Differential-Equation-Solving Tools:* A differential-equation-solving tool is typically a method or software for the numerical integration of an (explicit or implicit) ODE, i.e., a DAE of index zero. Numerical integration is a well-studied area, and there are several efficient approaches and methods; e.g., see [29] and [30]. Implementations are available in, for example, MATLAB and SIMULINK.

Independent of which differential-equation-solving tool is used, initial conditions for the state variables are, in general, required. The availability of initial conditions depends on the knowledge about the underlying system represented by the model. For complex physical systems, object-oriented modeling tools, e.g., Modelica [31], are frequently used to build models. Often, this leads to models in which state variables correspond to physical quantities such as pressures and temperatures, and then, initial conditions may have clear physical interpretations. For example, in an engine model, a variable corresponding to the intake manifold pressure should be equal to the ambient pressure when the engine starts.

If all equilibrium points of the considered ODE are (globally) asymptotically stable or, by using, e.g., state-feedback [39], can be made so, the effect of the initial conditions is neglectable. However, the computed trajectory will, in this case, differ from the true trajectory for some time due to transients.

Recall from Section III-A that each block in a BLT semiexplicit DAE system can be transformed to an explicit ODE. In the following, we assume that differential-equation-solving tools are always available and that an explicit ODE can be solved, i.e., that trajectories of the state variables in the ODE can be computed, if the initial conditions of the state variables are known and consistent. Of course, this assumption is not always valid, and numerical solving of ODEs involves difficulties and problems such as stability and stiffness, but this is not in the scope of this paper.

3) *Differentiation Tools:* A differentiation tool is, for example, an implementation of a method for approximate differentiating of known variables. There are several approaches, e.g., low-pass filtering or smoothing spline approximation [40]. An extensive survey of methods can be found in [41]. Methods for approximate differentiation are not in the scope of this paper and will not further be considered.

In the following, we assume that the differentiation of a set of known variables is either possible or not possible. That is, if a tool for approximate differentiation is available, we assume

that the quality of the measurements of the involved variables is good enough to support the tool.

One alternative to directly differentiate variables is to propagate unknown differentiated variables through a set of equations so that these can be expressed as derivatives of measured variables only. Assume, for example, that we want to compute the derivative $\dot{x}_1$, and we also have that $x_1 = y_1$. To compute $\dot{x}_1$, we use a differentiation tool to compute $\dot{y}_1$ and then use $\dot{x}_1 = \dot{y}_1$.

C. Computation Sequence

To describe the way and order in which a set of variables is computed from a set of equations, we will introduce the concept *computation sequence*. Before going into details, we need some additional notation. Let $V \subseteq X \cup D$ and define

$$\text{Diff}(V) = \{\dot{x}_j \in D : \dot{x}_j \in V \vee x_j \in V\} \quad (10)$$

$$\text{unDiff}(V) = \{x_j \in X : x_j \in V \vee \dot{x}_j \in V\}. \quad (11)$$

For instance, we have that $\text{Diff}(\{\dot{x}_1, x_2\}) = \{\dot{x}_1, \dot{x}_2\}$ and $\text{unDiff}(\{\dot{x}_1, x_2\}) = \{x_1, x_2\}$.

Now, consider the model $M(E, X, Y)$, where E is the set of equations specified in (3), X is the set of unknown variables, and Y is the set of known variables.

Definition 3 (Computation Sequence): Given a set of variables $X' \subseteq X$, an AE tool $\mathcal{T}$, and an ordered set

$$\mathcal{C} = ((V_1, E_1), (V_2, E_2), \dots, (V_k, E_k))$$

where $V_i \subseteq \text{var}_X(E_i) \cup \text{var}_D(E_i)$, and $\{E_i\}$ is pairwise disjoint. The ordered set $\mathcal{C}$ is a *computation sequence* for X' with $\mathcal{T}$ if

- 1) $X' \subseteq \text{unDiff}(V_1 \cup V_2 \cup \dots \cup V_k)$;
- 2) a system in BLT semiexplicit DAE form is obtained by sequentially calling the tool $\mathcal{T}$, with arguments V_i and E_i , for each element $(V_i, E_i) \in \mathcal{C}$. $\square$

For an example, recall the model (1), where $E = \{e_1, e_2, e_3, e_4, e_5\}$, $X = \{x_1, x_2, x_3, x_4\}$, and $Y = \{y_1, y_2, y_3\}$. Assume that the given AE tool $\mathcal{T}$ is ideal, in the sense that it can solve all solvable linear and nonlinear equations. Then, the ordered set

$$\mathcal{C} = ((\{x_3\}, \{e_4\}), (\{x_4\}, \{e_2\}), (\{\dot{x}_1\}, \{e_1\}), (\{x_2\}, \{e_3\})) \quad (12)$$

is a computation sequence for $\{x_1, x_2, x_3, x_4\}$ with $\mathcal{T}$ according to Definition 3, since

$$\text{unDiff}(\{x_3\} \cup \{x_4\} \cup \{\dot{x}_1\} \cup \{x_2\}) = \{x_1, x_2, x_3, x_4\}$$

and the BLT semiexplicit DAE system

$$x_3 = y_3 \quad (13a)$$

$$x_4 = \dot{x}_3 \quad (13b)$$

$$\dot{x}_1 = x_2 \quad (13c)$$

$$x_2 = \frac{-\dot{x}_4 x_1 + y_1}{2x_4} \quad (13d)$$

is obtained by sequentially calling $\mathcal{T}$ with elements from $\mathcal{C}$ as arguments.

Note that the obtained BLT semiexplicit DAE system (13) has three blocks: the first block corresponds to (13a), the second corresponds to (13b), and the third corresponds to (13c) and (13d). Furthermore, note that the equation set $\{e_1, e_3\}$, containing a differential loop, corresponds to a semiexplicit DAE of index one given by (13c) and (13d). Furthermore, derivative causality is used in (13b) and (13d), and integral causality is used in (13c).

IV. SEQUENTIAL RESIDUAL GENERATION

In this section, it is shown how a computation sequence can be utilized for residual generation. A residual generator based on a computation sequence will be defined as a sequential residual generator. In a sequential residual generator, the generation of a residual will consist of a finite sequence of variable computations ending with the evaluation of an unused equation. The concepts of minimal and irreducible computation sequence and proper sequential residual generator will then be introduced. A necessary condition for the existence of a proper sequential residual generator is given. The section ends with an algorithm able to find proper sequential residual generators, given a model and an AE tool.

An important property of a computation sequence is given by the following lemma.

Lemma 1: Let the ordered set

$$\mathcal{C} = ((V_1, E_1), (V_2, E_2), \dots, (V_k, E_k))$$

be a computation sequence for the variables $X' \subseteq X$ with the AE tool $\mathcal{T}$, and let $\bar{E}'$ be the set of equations in BLT semiexplicit DAE form obtained from $\mathcal{C}$ with the AE tool $\mathcal{T}$. Then, the solution sets of $\bar{E}'$ and $E_1 \cup E_2 \cup \dots \cup E_k$, with respect to $V_1 \cup V_2 \cup \dots \cup V_k$, are equal and unique. $\square$

With this lemma, the following important result can be proved.

Theorem 1: Let $M(E, X, Y)$ be a model, $\mathcal{T}$ be an AE tool, and

$$\mathcal{C} = ((V_1, E_1), (V_2, E_2), \dots, (V_k, E_k))$$

be a computation sequence for $X' \subseteq X$ with $\mathcal{T}$, where $E_i \subseteq E$. Furthermore, let $e_i \in E \setminus E_1 \cup E_2 \cup \dots \cup E_k$, where $\text{var}_X(e_i) \subseteq X'$, and it is assumed that e_i is written as $f_i(\dot{x}, x, y) = 0$. Then, the BLT semiexplicit DAE system obtained from $\mathcal{C}$ with $\mathcal{T}$ and $r = f_i(\dot{x}, x, y)$ is a residual generator for $M(E, X, Y)$ if

- 1) consistent initial conditions of all states are available;
- 2) all needed derivatives can be computed with the available differentiation tools. $\square$

Motivated by this theorem, we define a sequential residual generator as follows.

Definition 4 (Sequential Residual Generator): A residual generator for $M(E, X, Y)$ obtained from a computation sequence $\mathcal{C}$ and an equation $e_i \in E$, in accordance with the description in Theorem 1, is a *sequential residual generator*

for $M(E, X, Y)$, denoted $\mathcal{S} = (\mathcal{T}(\mathcal{C}), e_i)$, and e_i is a *residual equation*. $\square$

A. Proper Sequential Residual Generator

Regarding implementation aspects, e.g., complexity or numerical issues, smaller computation sequences are generally better. In particular, it is unnecessary to compute variables that are not contained in the residual equation or not used to compute any of the variables contained in the residual equation. Motivated by this discussion, we make the following definition.

Definition 5 (Minimal Computation Sequence): Given a set of variables $X' \subseteq X$ and an AE tool $\mathcal{T}$, a computation sequence $\mathcal{C}$ for X' with $\mathcal{T}$ is *minimal* if there is no other computation sequence $\mathcal{C}'$ for X' with $\mathcal{T}$ such that $\mathcal{C}' \subset \mathcal{C}$. $\square$

Return to the model (1) in Section I. Consider the last two equations in the model, i.e.,

$$e_4 : x_3 - y_3 = 0$$

$$e_5 : x_2 - y_2 = 0$$

and let $\mathcal{T}$ be an ideal AE tool. The computation sequence

$$\mathcal{C}_1 = ((\{x_3\}, \{e_4\}), (\{x_2\}, \{e_5\})) \quad (14)$$

for $\{x_2, x_3\}$ with $\mathcal{T}$ is minimal. The resulting BLT semiexplicit DAE form is given by

$$x_3 = y_3 \quad (15a)$$

$$x_2 = y_2. \quad (15b)$$

However, $\mathcal{C}_1$ is not minimal for $\{x_3\}$ since $\mathcal{C}_2 = (\{x_3\}, \{e_4\})$ is a (minimal) computation sequence for $\{x_3\}$ with $\mathcal{T}$, and $\mathcal{C}_2 \subset \mathcal{C}_1$.

The computation of variables according to a minimal computation sequence thus implies that no unnecessary variables are computed. However, with the complexity and numerical aspects in mind, it is also most desirable that the computation of variables in each step is performed from as small equation sets as possible. This leads to the following definition.

Definition 6 (Irreducible Computation Sequence): Given a set of variables $X' \subseteq X$ and an AE tool $\mathcal{T}$, a computation sequence

$$\mathcal{C} = ((V_1, E_1), (V_2, E_2), \dots, (V_k, E_k))$$

for X' with $\mathcal{T}$ is *irreducible* if no element $(V_i, E_i) \in \mathcal{C}$ can be partitioned as $V_i = V_{i1} \cup V_{i2}$ and $E_i = E_{i1} \cup E_{i2}$, such that

$$\mathcal{C}' = ((V_1, E_1), \dots, (V_{i1}, E_{i1}), (V_{i2}, E_{i2}), \dots, (V_k, E_k))$$

is a computation sequence for X' with $\mathcal{T}$. $\square$

Return to the equation set $\{e_4, e_5\}$ considered above. Clearly, the ordered set $\mathcal{C}_3 = (\{x_2, x_3\}, \{e_4, e_5\})$ is a minimal computation sequence for $\{x_2, x_3\}$ with the ideal AE tool $\mathcal{T}$. The corresponding BLT semiexplicit DAE system is given by (15). However, $\mathcal{C}_3$ is not irreducible since $\mathcal{C}_1$ given by (14) is also a computation sequence for $\{x_2, x_3\}$.

From now on, we will only consider AE tools fulfilling the following quite nonlimiting property.

Assumption 3: Let $E_i = E_{i1} \cup E_{i2}$ and $V_i = V_{i1} \cup V_{i2}$, in accordance with Definition 6. If an AE tool can solve E_i for V_i , it can also solve E_{i1} for V_{i1} and E_{i2} for V_{i2} . $\square$

Sequential residual generators based on minimal and irreducible computation sequences are of particular interest.

Definition 7 (Proper Sequential Residual Generator): Given an equation $e_i \in E$, an AE tool $\mathcal{T}$ and a computation sequence $\mathcal{C}$ for $\text{var}_X(e_i)$ with $\mathcal{T}$. A sequential residual generator $\mathcal{S} = (\mathcal{T}(\mathcal{C}), e_i)$ is *proper* if $\mathcal{C}$ is a minimal and irreducible computation sequence for $\text{var}_X(e_i)$ with $\mathcal{T}$. $\square$

For the construction of a sequential residual generator, a computation sequence and a residual equation are needed. Due to Assumption 2, the equation set contained in a computation sequence is a just-determined set of equations. Since the residual equation is redundant (see Theorem 1), it follows that the equations in a computation sequence and the residual equation constitute an overdetermined equation set. Hence, an overdetermined set of equations is needed to construct a sequential residual generator. For the construction of a proper sequential residual generator, an MSO set [19] is needed.

Theorem 2: Let $\mathcal{S} = (\mathcal{T}(\mathcal{C}), e_i)$ be a proper sequential residual generator, where

$$\mathcal{C} = ((V_1, E_1), (V_2, E_2), \dots, (V_k, E_k))$$

then the equation set $E_1 \cup E_2 \cup \dots \cup E_k \cup e_i$ is an MSO set with respect to $\text{var}_X(E_1 \cup E_2 \cup \dots \cup E_k \cup e_i)$. $\square$

Note that Theorem 2 establishes a link between structural and analytical methods. This is done without the use of any assumptions of generic equations as in, for example, [19], instead assumptions have been placed on the tools.

Recall again the model (1) and consider the computation sequence $\mathcal{C}$, given by (12), with the corresponding BLT semi-explicit DAE form (13). The computation sequence $\mathcal{C}$ together with the equation e_5 is a sequential residual generator for the model (1) if we assume that the initial condition of x_1 is known and consistent and the derivatives $\dot{x}_3$ and $\dot{x}_4$ can be computed with the available differentiation tools. As a matter of fact, the residual generator is a *proper* sequential residual generator since the computation sequence $\mathcal{C}$ for $\text{var}_X(e_5) = \{x_2\}$ with the ideal AE tool $\mathcal{T}$ is minimal and irreducible. Hence, we can, by Theorem 2, conclude that the equation set $E = \{e_1, e_2, e_3, e_4, e_5\}$ is an MSO set.

B. Finding Proper Sequential Residual Generators

Theorem 2 states a necessary condition for the existence of a proper sequential residual generator. Hence, a first step when searching for all proper sequential residual generators may be to find all MSO sets. There are efficient algorithms for finding all MSO sets in large equation sets; e.g., see [19].

Motivated by this, we propose the following algorithm for finding proper sequential residual generators, given a model $M(E, X, Y)$ and an AE tool $\mathcal{T}$. The function `FINDALLMSOs` is assumed to find all MSO sets in the equation set E . The function `FINDCOMPUTATIONSEQUENCE`, taking an equation

set E' , a variable set X' , and an AE tool $\mathcal{T}$, is assumed to return a minimal and proper computation sequence for X' with $\mathcal{T}$.

```

1: function FINDRESIDUALGENERATORS( $E, X, \mathcal{T}$ )
2:    $R := \emptyset$ 
3:    $\text{MSOs} := \text{FINDALLMSOs}(E, X)$ 
4:   for all  $\bar{E} \in \text{MSOs}$  do
5:      $X' := \text{var}_X(\bar{E})$ 
6:     for all  $e_i \in \bar{E}$  do
7:        $E' := \bar{E} \setminus e_i$ 
8:        $\mathcal{C} := \text{FINDCOMPUTATIONSEQUENCE}(E', X', \mathcal{T})$ 
9:       if  $\mathcal{C} \neq \emptyset$  then
10:         $R = R \cup \{(\mathcal{T}(\mathcal{C}), e_i)\}$ 
11:       end if
12:     end for
13:   end for
14:   return  $R$ 
15: end function

```

The algorithm is justified by the following theorem.

Theorem 3: Let $M(E, X, Y)$ be a model and $\mathcal{T}$ be an AE tool. Furthermore, let R be the set returned by `FINDRESIDUALGENERATORS` when E, X , and $\mathcal{T}$ are used as input. Then, all elements $(\mathcal{T}(\mathcal{C}), e_i) \in R$ are proper sequential residual generators for $M(E, X, Y)$ if, in accordance with Theorem 1, consistent initial conditions of all states are available and all needed derivatives can be computed with the available differentiation tools. $\square$

The most important step in `FINDRESIDUALGENERATORS` is thus to find a minimal and irreducible computation sequence, i.e., the function `FINDCOMPUTATIONSEQUENCE`. This is the topic of the next section.

V. METHOD FOR FINDING A COMPUTATION SEQUENCE

A proper sequential residual generator consists of a BLT semiexplicit DAE system, obtained from a minimal and irreducible computation sequence, and a residual equation. Essential for the construction of a proper sequential residual generator is thus to find a minimal and irreducible computation sequence. The method that we propose for finding a computation sequence is presented in this section. First, the different steps of the method are illustrated by studying an example.

A. Illustrative Example

Consider the following set of equations:

$$\begin{aligned}
e_1 : \quad & \dot{x}_1 + x_1 x_6 - x_3 - x_5^2 x_7 = 0 \\
e_2 : \quad & \dot{x}_2 + x_2 x_3 + y_1 = 0 \\
e_3 : \quad & \dot{x}_3 + x_3 - x_2 x_4 + y_2 = 0 \\
e_4 : \quad & \dot{x}_4 + x_2 - x_5 - y_3 = 0 \\
e_5 : \quad & x_1 - x_2 x_3 - x_4 + x_6 - 2x_7 - y_4 = 0 \\
e_6 : \quad & x_3^2 - x_6 - x_7 + y_5 = 0 \\
e_7 : \quad & x_4 - y_6 = 0
\end{aligned}$$

where $X = \{x_1, x_2, \dots, x_7\}$ are unknown variables, and $Y = \{y_1, y_2, \dots, y_6\}$ are known variables. Assume that we want to find a computation sequence for X with a given AE tool.

First, identify the SCCs (recall Section II-C) of the structure of $E = \{e_1, e_2, \dots, e_7\}$ with respect to X and accordingly order the corresponding partitions of the equation and variable sets, i.e.,

$$\begin{array}{c|ccccccc}
 & x_4 & x_3 & x_2 & x_5 & x_6 & x_7 & x_1 \\
\hline
e_7 & \mathbf{1} & & & & & & \\
e_3 & 1 & \mathbf{1} & \mathbf{1} & & & & \\
e_2 & & \mathbf{1} & \mathbf{1} & & & & \\
e_4 & 1 & & 1 & \mathbf{1} & & & \\
e_6 & & 1 & & & \mathbf{1} & \mathbf{1} & \\
e_1 & & 1 & & 1 & \mathbf{1} & \mathbf{1} & \mathbf{1} \\
e_5 & 1 & 1 & 1 & & \mathbf{1} & \mathbf{1} & \mathbf{1}
\end{array} \quad (16)$$

The ordered partitions are

$$\begin{aligned}
 \mathcal{E} &= (\{e_7\}, \{e_2, e_3\}, \{e_4\}, \{e_1, e_5, e_6\}) \\
 \mathcal{X} &= (\{x_4\}, \{x_2, x_3\}, \{x_5\}, \{x_1, x_6, x_7\})
 \end{aligned}$$

where each element in $\mathcal{E}$ is an SCC with respect to the corresponding element in $\mathcal{X}$, e.g., $(\{e_2, e_3\}, \{x_2, x_3\})$. The SCCs are marked with bold in (16).

The first SCC $(\{e_4\}, \{x_7\})$ contains one linear algebraic equation. Under the assumption that our AE tool can handle such equations, e_7 is solved for x_4 , and we obtain

$$x_4 = y_6. \quad (17)$$

Then, consider the next SCC $(\{e_2, e_3\}, \{x_2, x_3\})$, which contains two differential equations. The permuted structure of $\{e_2, e_3\}$ with respect to the differentiated variables $\{\dot{x}_2, \dot{x}_3\}$ is

$$\begin{array}{c|cc}
 & \dot{x}_3 & \dot{x}_2 \\
\hline
e_3 & 1 & \\
e_2 & & 1
\end{array} \quad (18)$$

As seen, the structure (18) contains two SCCs of size one, i.e., $(\{e_3\}, \{\dot{x}_3\})$ and $(\{e_2\}, \{\dot{x}_2\})$. Assuming that our AE tool admits it, we then solve e_3 for $\dot{x}_3$ and e_2 for $\dot{x}_2$ and obtain

$$\begin{aligned}
 \dot{x}_3 &= -x_3 + x_2x_4 - y_2 \\
 \dot{x}_2 &= -x_2x_3 - y_1.
\end{aligned} \quad (19)$$

The next SCC $(\{e_4\}, \{x_5\})$ contains a differential equation. However, since x_5 is the variable intended to compute from the equation, we can handle e_6 as an algebraic equation and solve it for x_5 , i.e.,

$$x_5 = x_2 + \dot{x}_4 - y_3. \quad (20)$$

The SCC $(\{e_1, e_5, e_6\}, \{x_1, x_6, x_7\})$ contains the differential equation e_1 and the two algebraic equations e_5 and e_6 . By analyzing the equations, we see that x_6 and x_7 are algebraic variables contained in both e_5 and e_6 and that x_1 is a differentiated variable present in e_1 . We then solve e_1 for $\dot{x}_1$ and obtain

$$\dot{x}_1 = -x_1x_6 + x_5^2x_7 + x_3. \quad (21)$$

The structure of $\{e_5, e_6\}$ with respect to $\{x_6, x_7\}$ reveals an SCC of size two, i.e.,

$$\begin{array}{c|cc}
 & x_6 & x_7 \\
\hline
e_5 & 1 & 1 \\
e_6 & 1 & 1
\end{array} \quad (22)$$

Under the assumption that our AE tool can handle it, we solve the equation system $\{e_5, e_6\}$ for $\{x_6, x_7\}$ and obtain

$$\begin{aligned}
 x_6 &= 2x_3^2 + x_1 - x_2x_3 - x_4 + 2y_5 - y_4 \\
 x_7 &= x_1 - x_2x_3 - x_4 + x_3^2 + y_5 - y_4.
\end{aligned} \quad (23)$$

Collecting (17), (19)–(21), and (23) gives

$$x_4 = y_6 \quad (24a)$$

$$\dot{x}_3 = -x_3 + x_2x_4 - y_2 \quad (24b)$$

$$\dot{x}_2 = -x_2x_3 - y_1 \quad (24c)$$

$$x_5 = x_2 + \dot{x}_4 - y_3 \quad (24d)$$

$$\dot{x}_1 = -x_1x_6 + x_5^2x_7 + x_3 \quad (24e)$$

$$x_6 = 2x_3^2 + x_1 - x_2x_3 - x_4 + 2y_5 - y_4 \quad (24f)$$

$$x_7 = x_1 - x_2x_3 - x_4 + x_3^2 + y_5 - y_4 \quad (24g)$$

which is a system in BLT semiexplicit DAE form with four blocks. Equation (24a) corresponds to the first block, which only contains an algebraic equation. The second block is given by (24b) and (24c) and corresponds to an explicit ODE with respect to the variables $\{x_2, x_3\}$. Hence, integral causality is used in this block. The third block contains (24d), which is a differential equation in which derivative causality is used. Equations (24e)–(24g) constitute the fourth and last block. This block corresponds to a semiexplicit DAE of index one, with respect to the variables $\{x_1, x_6, x_7\}$.

The resulting computation sequence for $\{x_1, x_2, \dots, x_7\}$ with the given AE tool is

$$\begin{aligned}
 \mathcal{C} &= ((\{x_4\}, \{e_7\}), (\{\dot{x}_3\}, \{e_3\}), (\{\dot{x}_2\}, \{e_2\}), \\
 &\quad (\{x_5\}, \{e_4\}), (\{\dot{x}_1\}, \{e_1\}), (\{x_6, x_7\}, \{e_6, e_5\})).
\end{aligned}$$

B. Summary of the Method

Given an AE tool and a just-determined set of equations, the proposed method for finding a computation sequence can be outlined as follows.

- 1) Find the SCCs of the structure of the equation set with respect to the unknown variables. No distinction is made between a variable and its derivative.
- 2) For each SCC, split the equations into one set of differential equations and one set of algebraic equations and the

variables into one set of differentiated variables and one set of algebraic variables.

- 3) For the differential equations, find the SCCs of the structure of the differential equations with respect to the differentiated variables. For each SCC, try to solve the differential equations for the intended differentiated variables with the AE tool. Note that, due to the assumption that each differential equation only contains one differentiated variable, all SCCs are of size one.
- 4) For the algebraic equations, find the SCCs of the structure of the algebraic equations with respect to the algebraic variables. For each SCC, try to solve the algebraic equations for the intended algebraic variables with the AE tool.

C. Algorithm

The method is formally described in the function `FINDCOMPUTATIONSEQUENCE` below. The function takes a just-determined equation set $E' \subseteq E$, a set of unknown variables $X' \subseteq X$, and an AE tool $\mathcal{T}$ as input and returns an ordered set $\mathcal{C}$ as output. The function `FINDALLSCCs` is assumed to return an ordered set of equation and variable pairs, where each pair corresponds to an SCC of the structure of the equation set with respect to the variable set. The order of the SCCs returned by `FINDALLSCCs` is assumed to be the one depicted in Fig. 1; for more information regarding ordering of SCCs, please refer to [26]. There are efficient algorithms for finding SCCs in directed graphs; e.g., see [42]. The DM decomposition [25] can also be utilized. In MATLAB, the DM decomposition is implemented in the function `dmperm`, from which the order of the SCCs, according to Fig. 1, can also easily be obtained. Other functions used in `FINDCOMPUTATIONSEQUENCE` are the following.

- 1) `DIFF` and `UNDIFF` take a variable set as input and return its differentiated and undifferentiated correspondence; see (10) and (11).
- 2) `ISINITCONDKNOWN` determines if the initial conditions of the given variables are known and consistent, and the function `ISDIFFERENTIABLE` determines if the given variables can be differentiated with the available differentiation tool.
- 3) `ISJUSTDETERMINED` is used to determine if the structure of the given equation set, with respect to the given variable set, is a just-determined equation set. This is essential, since, otherwise, the computation of SCCs makes no sense.
- 4) `GETDIFFERENTIALEQUATIONS` takes a set of equations and a set of differentiated variables as input and returns the differential equations in which the given differentiated variables are contained.
- 5) `ISTOOLSOLVABLE` determines if the given AE tool can solve the given equations for the given set of variables.
- 6) `APPEND` takes an ordered set and an element as input and simply appends the element to the end of the set.
- 7) The operator $|\cdot|$, taking a set as input, is assumed to return the number of elements in the set, and the notion

$A(i)$ is used to refer to the i th element of the ordered set A .

```

1: function FINDCOMPUTATIONSEQUENCE( $E', X', \mathcal{T}$ )
2:    $\mathcal{C} := \emptyset$ 
3:    $S := \text{FINDALLSCCs}(E', X')$ 
4:   for ( $i = 1, 2, \dots, |S|$ ) do
5:      $(E_i, X_i) := S(i)$ 
6:      $D_i := \text{DIFF}(X_i)$ 
7:      $Z_i := \text{var}_D(E_i) \cap D_i$ 
8:      $W_i := X_i \setminus \text{UNDIFF}(Z_i)$ 
9:     if not ISINITCONDKNOWN( $Z_i$ ) then
10:      return  $\emptyset$ 
11:     end if
12:      $E_{Z_i} := \text{GETDIFFERENTIALEQUATIONS}(E_i, Z_i)$ 
13:      $E_{W_i} := E_i \setminus E_{Z_i}$ 
14:      $S_{Z_i} := \text{FINDALLSCCs}(E_{Z_i}, Z_i)$ 
15:     for  $j = 1, 2, \dots, |S_{Z_i}|$  do
16:        $(E_{Z_i}^j, Z_i^j) := S_{Z_i}(j)$ 
17:       if ISTOOLSOLVABLE( $Z_i^j, E_{Z_i}^j, \mathcal{T}$ ) then
18:         APPEND( $\mathcal{C}, (Z_i^j, E_{Z_i}^j)$ )
19:       else
20:         return  $\emptyset$ 
21:       end if
22:     end for
23:     if ISJUSTDETERMINED( $E_{W_i}, W_i$ ) then
24:        $S_{W_i} := \text{FINDALLSCCs}(E_{W_i}, W_i)$ 
25:       for  $j = 1, 2, \dots, |S_{W_i}|$  then
26:          $(E_{W_i}^j, W_i^j) := S_{W_i}(j)$ 
27:         if ISTOOLSOLVABLE( $W_i^j, E_{W_i}^j, \mathcal{T}$ ) then
28:           APPEND( $\mathcal{C}, (W_i^j, E_{W_i}^j)$ )
29:         else
30:           return  $\emptyset$ 
31:         end if
32:       end for
33:     else
34:       return  $\emptyset$ 
35:     end if
36:   end for
37:   return  $\mathcal{C}$ 
38: end function

```

That the ordered set $\mathcal{C}$ returned by `FINDCOMPUTATIONSEQUENCE` is indeed a minimal and irreducible computation sequence is verified in the following theorem.

Theorem 4: Let $E' \subseteq E$ be a just-determined set of equations with respect to the variables $X' \subseteq X$ and $\mathcal{T}$ be an AE tool. If E', X' , and $\mathcal{T}$ are used as arguments to `FINDCOMPUTATIONSEQUENCE` and a nonempty $\mathcal{C}$ is returned, then $\mathcal{C}$ is a minimal and irreducible computation sequence for X' with $\mathcal{T}$. $\square$

VI. APPLICATION STUDIES

The objective of this section is to empirically show the benefits of the method for finding the sequential residual generators proposed in Sections IV-B and V-C. This is done by applying

TABLE I
SIX CONFIGURATIONS OF THE METHOD USED IN THE STUDIES

	D	I	DI	SD	SI	SDI
SCC				x	x	x
IC		x	x		x	x
DC	x		x	x		x

the method to models of an automotive diesel engine and an auxiliary hydraulic braking system. In addition, we illustrate how a sequential residual generator for the diesel engine, found with the proposed method, can be realized. The realized residual generator is then evaluated using real measurements from a truck.

A. Implementation and Configuration of the Method

The analytical models of the two systems were obtained from SIMULINK models by using the toolbox described in [43]. The resulting models are complex DAEs containing nonlinearities like min and max functions, lookup tables, saturations, and polynomials.

The functions FINDRESIDUALGENERATORS and FINDCOMPUTATIONSEQUENCE, described in Sections IV-B and V-C, were implemented in MATLAB. In the implementation of FINDCOMPUTATIONSEQUENCE, the symbolic equation solver in MAPLE was used as an AE tool. To find all MSO sets, the algorithm described in [19] was used. The MSO sets were arranged in classes, so that MSOs containing the same set of known variables belongs to the same MSO class.

For comparison, different configurations of FINDCOMPUTATIONSEQUENCE were applied to the models. The following parameters, which naturally influence the possibility to find computation sequences, were used for configuration.

- SCC Ability to handle SCCs of larger size than one, i.e., equation sets containing algebraic or differential loops.
- IC Ability to use integral causality.
- DC Ability to use derivative causality.

Note that if a configuration uses integral causality, it is assumed that all initial conditions are available. Moreover, it is assumed that all needed derivatives can be computed when a configuration uses derivative causality.

The six possible different configurations are shown in Table I. For example, configuration SI is able to handle equation sets containing loops and use integral causality, but it cannot use derivative causality. The configuration corresponding to the novel approach for finding sequential residual generators proposed in this paper is SDI.

B. Performance Measures

A sequential residual generator is sensitive to those faults that influence its residual equation and the equations contained in its computation sequence. Different MSO sets correspond to different subsets of the equations in the model. Sequential residual generators obtained from computation sequences and residual equations originating from different MSO sets will thus naturally be sensitive to different subsets of faults. To

Fig. 2. Cutaway of a Scania 13-L six-cylinder diesel engine equipped with EGR and VGT. Illustration by Semcon Informatic Graphics Solutions.

achieve good fault isolation, it is hence important that residual generators can be constructed from as many MSO sets as possible.

In the automotive applications studied here, it is particularly important to detect and isolate faults present in sensors and actuators, that is, faults affecting measurements of known variables. Hence, it is also important that residual generators can be constructed from as many MSO classes as possible.

Additionally, different residual generators constructed from the same MSO set or MSO class may have different properties regarding, for example, numerical aspects, sensitivity to faults, and sensitivity to disturbances such as measurement noise or modeling errors. Hence, it is most desirable to be able to evaluate as many residual generators as possible, with real measurement data, to decide which set of residual generators to use in the final diagnosis system.

Motivated by this discussion, we will use the following performance measures to compare the different configurations of the method.

- MSO sets In how many of the total number of MSO sets at least one residual generator could be found.
- MSO classes In how many of the total number of MSO classes at least one residual generator could be found.
- Residual generators The total number of residual generators found.

C. Automotive Diesel Engine

The studied engine is a 13-L six-cylinder Scania diesel engine equipped with exhaust gas recirculation (EGR) and a variable geometry turbocharger (VGT). A cutaway of the engine can be found in Fig. 2.

The model describes the gas flow in the engine; see [44] for more details. The analytical model extracted from the SIMULINK model is a nonlinear DAE system and contains 282 equations, 272 unknown variables, and 11 known variables. Of the equations, eight are differential, and the rest are

TABLE II
RESULTS FOR THE DIESEL ENGINE

	MSO Sets	MSO Classes	Residual Generators
D	4	4	46
I	1	1	5
DI	4	4	46
SD	4	4	46
SI	23	20	58
SDI	120	72	1636
Potential	598	210	135772

Fig. 3. Bars to the left and in the middle show the fractions of the total number of MSO sets and MSO classes in which a residual generator could be found with each configuration of the method. The bars to the right show the fractions of the number of potential residual generators that could be found with each configuration of the method.

algebraic. The differentiated variables represent physical quantities such as pressures, temperatures, and rotational speeds.

In total, 598 MSO sets could be found in the engine model. The MSO sets could be arranged into 210 MSO classes. Theoretically, the total number of potential residual generators that can be constructed from an MSO set is equal to the total number of equations in the MSO set. In this case, 135 772 different residual generators could theoretically be constructed from the 598 MSO sets.

The total number of residual generators found and how many of the MSO sets and MSO classes that could be used, for each configuration of the method, are shown in Table II and Fig. 3. The columns to the left and in the middle of Table II shows in how many of the MSO sets and MSO classes at least one residual generator could be found. The column to the right shows the total number of residual generators that could be found for each configuration of the method.

It is obvious that a very small fraction of the potential residual generators were found (about 1.2%) and that only a small fraction of the MSO sets and MSO classes could be used, independent of configuration. The main reason for this is the complexity of the engine model. The model contains large algebraic and differential loops, including complex nonlinear equations, which are impossible to analytically solve. Nevertheless, many more residual generators were found, and more MSO sets could be used with configuration SDI, i.e., with mixed causality and the ability to handle loops, in comparison with any other configuration of FINDCOMPUTATIONSEQUENCE.

TABLE III
RESULTS FOR THE HYDRAULIC BRAKING SYSTEM

	MSO Sets	MSO Classes	Residual Generators
D	21	14	145
I	6	6	18
DI	21	14	147
SD	33	22	288
SI	29	29	71
SDI	65	44	1293
Potential	125	83	4607

Fig. 4. Bars to the left and in the middle show the fractions of the total number of MSO sets and MSO classes in which a residual generator could be found with each configuration of the method. The bars to the right show the fractions of the number of potential residual generators that could be found with each configuration of the method.

D. Hydraulic Braking System

The Scania auxiliary hydraulic braking system, called *retarder*, is used on heavy-duty trucks for long continuous braking, for example, to maintain constant speed down a slope. By using the retarder, braking discs can be saved for short-time braking.

The model of the hydraulic braking system contains 49 equations, 44 unknown variables, and nine known variables. It is a nonlinear DAE system and contains four differential equations and 45 algebraic equations.

The model contains 125 MSO sets, which can be arranged into 83 MSO classes. The total number of possible residual generators for the model of the hydraulic braking system is theoretically 4607.

Table III and Fig. 4 show, for each configuration of the method, how many of the MSO sets and MSO classes could be used and the total number of residual generators found for the model of the hydraulic braking system. As seen, a significantly larger fraction of the MSO sets and MSO classes could be used, and more residual generators could be found with configuration SDI, in comparison with any other configuration.

E. Realization of a Residual Generator for the Diesel Engine

The purpose of this section is to briefly show how a residual generator for the diesel engine is constructed from a computation sequence obtained with the proposed method.

Fig. 5. Structure of the 203 equations in the considered computation sequence, with respect to the 203 unknown variables. The SCCs of the structure, corresponding to the elements in the computation sequence, are marked with squares. The large SCC contains 102 equations.

1) *Properties of the Computation Sequence:* The considered computation sequence originates from an MSO set containing in total 204 equations, 203 unknown variables, and eight known variables. Thus, the computation sequence contains 203 equations and 203 unknown variables. In total, 33 residual generators were found in the MSO class to which the MSO set belongs. All 33 residual generators were found with configuration SDI of FINDCOMPUTATIONSEQUENCE.

The computation sequence contains 102 elements. All elements, except the last one, contain one equation and one variable. The last element contains 102 equations and 102 variables and corresponds to an SCC of size 102. The structure of the 203 equations contained in the computation sequence, with respect to the 203 unknown variables, is shown in Fig. 5. The SCCs of the structure, corresponding to the elements in the computation sequence, are marked with squares in Fig. 5.

The residual equation used in the residual generator, i.e., the equation removed from the MSO set when the corresponding computation sequence was found, compares the measured and computed pressure in the intake manifold of the diesel engine.

2) *Properties of the BLT Semiexplicit DAE System:* The BLT semiexplicit DAE system obtained from the computation sequence contains 102 blocks and has the following form:

$$\mathbf{w}_1 = \mathbf{h}_1(\mathbf{y})$$

$$\mathbf{w}_2 = \mathbf{h}_2(\mathbf{w}_1, \mathbf{y})$$

$$\vdots$$

$$\mathbf{w}_{64} = \mathbf{h}_{64}(\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{63}, \mathbf{y})$$

$$\mathbf{w}_{65} = \mathbf{h}_{65}(\dot{\mathbf{w}}_{64}, \mathbf{w}_1, \dots, \mathbf{w}_{64}, \mathbf{y})$$

$$\mathbf{w}_{66} = \mathbf{h}_{66}(\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{65}, \mathbf{y})$$

$$\vdots$$

$$\mathbf{w}_{76} = \mathbf{h}_{76}(\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{75}, \mathbf{y})$$

$$\mathbf{w}_{77} = \mathbf{h}_{77}(\dot{\mathbf{w}}_{76}, \mathbf{w}_1, \dots, \mathbf{w}_{76}, \mathbf{y})$$

$$\mathbf{w}_{78} = \mathbf{h}_{78}(\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{77}, \mathbf{y})$$

$$\vdots$$

$$\mathbf{w}_{100} = \mathbf{h}_{100}(\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{99}, \mathbf{y})$$

$$\dot{\mathbf{z}}_{101} = \mathbf{g}_{101}(\mathbf{w}_1, \dots, \mathbf{w}_{101}, \mathbf{y})$$

$$\mathbf{w}_{101}^1 = \mathbf{h}_{101}^1(\mathbf{z}_{101}, \mathbf{w}_1, \dots, \mathbf{w}_{100}, \mathbf{y})$$

$$\mathbf{w}_{101}^2 = \mathbf{h}_{101}^2(\mathbf{z}_{101}, \mathbf{w}_1, \dots, \mathbf{w}_{100}, \mathbf{w}_{101}^1, \mathbf{y})$$

$$\vdots$$

$$\mathbf{w}_{101}^{99} = \mathbf{h}_{101}^{99}(\mathbf{z}_{101}, \mathbf{w}_1, \dots, \mathbf{w}_{100}, \mathbf{w}_{101}^1, \dots, \mathbf{w}_{101}^{98}, \mathbf{y}) \quad (25)$$

where $\mathbf{w}_{101} = (\mathbf{w}_{101}^1, \mathbf{w}_{101}^2, \dots, \mathbf{w}_{101}^{99})$, $\mathbf{z}_{101}$ is of dimension three, and all $\mathbf{w}_i$ and $\mathbf{w}_i^j$ are of dimension one. The largest block, denoted 101 in (25), is a semiexplicit DAE of index one with three differential equations with variables $\mathbf{z}_{101}$ and 99 algebraic equations with variables $\mathbf{w}_{101}^1, \dots, \mathbf{w}_{101}^{99}$, corresponding to a differential loop and an SCC of size 102. Since the block is a semiexplicit DAE of index one, integral causality is used in this block. In two of the blocks, denoted 66 and 77 in (25), derivative causality is used. The remaining blocks, denoted 1–65, 67–76, and 78–100 correspond to algebraic equations. In total, the BLT semiexplicit DAE system contains five differential equations and 198 algebraic equations.

3) *Implementation Issues:* The residual generator, i.e., the obtained BLT semiexplicit DAE system and the residual equation, was implemented in MATLAB. To compute the values of the unknown variables, the approach described in Section III-A2c was used. To solve the resulting explicit ODE, Euler forward with a fixed step size was utilized. All state variables in the residual generators represent physical quantities; hence, the initial conditions were easy to obtain from the available measurements.

Approximate differentiation: In the two blocks where derivative causality is used, i.e., 66 and 77 in (25), the derivatives of variables computed in previous blocks had to be computed. By propagating the two differentiated variables through equations in earlier blocks of the obtained BLT semiexplicit DAE system, the differentiated variables could be expressed as derivatives of known variables only; see Section III-B3. The known variables that had to be differentiated were measurements of the pressure in the exhaust manifold and the rotational speed of the turbo turbine.

The differentiation tool, i.e., the method for differentiation of known variables, used in this case study was a sliding-window least-square polynomial fit approach. By finding a linear approximation, in a least-square sense, to a set of consecutive measurements, referred to as a window, an approximation of the first-order derivative of the measured signal in the window can be obtained as the slope of the linear approximation; e.g., see [41]. This approach was used since it is simple and straightforward to implement and gave good results. An implementation

Fig. 6. Residual obtained from the constructed residual generator. No fault is present the first 100 s. During the remaining 100 s, there is a bias fault in the sensor measuring the pressure in the intake manifold. The dashed lines suggest how thresholds could be chosen in order to detect the fault.

was done in MATLAB; a window size of 40 measurements (20 past and 20 future) was used.

4) *Results:* Real measurements of the known variables in the engine model were collected by driving a truck on the road. Two sets of measurements were collected: one with a fault-free engine and one with an implemented fault. The implemented fault was a constant bias in the sensor measuring the pressure in the intake manifold of the diesel engine.

The residual generator was run off-board by using the collected measurements. The residual was then low-pass filtered to remove some measurement noise and finally scaled. In Fig. 6, the resulting residual is shown. During the first 100 s, the measurements are fault-free. In the remaining time, the measurements contain the implemented bias fault. It is obvious that the residual can be used to detect the injected fault.

VII. CONCLUSION

We have concluded in Section I that it is important that there is a large selection of different candidate residual generators to choose from when designing diagnosis systems. In this spirit, we have presented in this paper a method for deriving residual generators with the key property that it is able to find a large number of different residual generators. This property is first due to the fact that the method belongs to a class of methods that we refer to as sequential residual generation. This class of methods has, in earlier works, been shown to be powerful for real nonlinear systems [6]–[10]. Second, which is the key contribution of the paper, we have extended these earlier methods by handling mixed causality and also, in a systematic manner, equation sets containing differential and algebraic loops.

The method has been presented as an algorithm utilizing an assumed given toolbox of, e.g., algebraic equation solvers. We have proven, in Theorem 1, that the algorithm really finds residual generators and, in Theorems 3 and 4, that the residual generators, or rather the sequential residual generators, found are *proper*. Properness guarantees that the residual generator is not containing unnecessary computations and that computa-

tions are performed from as small equation sets as possible. We have also proven, in Theorem 2, that proper sequential residual generators are always found within MSO sets. This fact has been utilized in the algorithm since there is no need to look for sequential residual generators in equation sets other than MSO sets. Furthermore, this theorem provides a link between structural and analytical methods without the use of any assumptions of generic equations, such as in, for example, [19].

In the empirical study in Section VI, we have evaluated our method on models of two real automotive systems. The results obtained are compared with results from the special cases of using solely differential or integral causality or only handling scalar equations. It is evident that our more general method outperforms the other alternatives. Since the two systems have quite different characteristics, e.g., in the number of redundant sensors, we believe that these results are also representative for a larger class of systems.

APPENDIX

PROOFS OF THEOREMS AND LEMMAS

Proof of Lemma 1: Consider an element $(V_i, E_i) \in \mathcal{C}$ and let $\bar{E}'_i$ denote the set of equations obtained when $\mathcal{T}$ is called with arguments V_i and E_i . It then holds that $\bar{E}' = \bar{E}'_1 \cup \bar{E}'_2 \cup \dots \cup \bar{E}'_k$. Given $\mathbf{y}$, let $\tilde{\mathbf{x}}$ be an arbitrary solution to $\bar{E}'$, i.e., a trajectory fulfilling every equation $e_i \in \bar{E}'$. Trivially, $\tilde{\mathbf{x}}$ is also a solution to the equations in every $\bar{E}'_i$, since $\bar{E}'_i \subseteq \bar{E}'$. Assumption 1 then implies that $\tilde{\mathbf{x}}$ is a unique solution and is also a solution to every E_i and, hence, to $E_1 \cup E_2 \cup \dots \cup E_k$. By taking an arbitrary solution to $E_1 \cup E_2 \cup \dots \cup E_k$ and applying the same arguments as above, it can be shown that this solution is unique and also satisfies $\bar{E}'$, which completes the proof. ■

Proof of Theorem 1: Consider the model $M(E, X, Y)$ and assume that $\tilde{\mathbf{y}} \in \mathcal{O}(M)$. Due to the definition of $\mathcal{O}(M)$ in (4), we know that, given $\tilde{\mathbf{y}}$, there exists at least one trajectory of the variables in X that satisfies the equations in E . Since describing $E_1 \cup E_2 \cup \dots \cup E_k \subseteq E$, it holds that the trajectory $\tilde{\mathbf{y}}$ also belongs to the observation set of the submodel of $M(E, X, Y)$ given by $E_1 \cup E_2 \cup \dots \cup E_k$, i.e., the equation set contained in the computation sequence $\mathcal{C}$. Hence, given $\tilde{\mathbf{y}}$, there exists a trajectory $\tilde{\mathbf{x}}$ of the variables in $\text{var}_X(E_1 \cup E_2 \cup \dots \cup E_k)$ that satisfies $E_1 \cup E_2 \cup \dots \cup E_k$. By Lemma 1, we know that $\tilde{\mathbf{x}}$ is a unique solution that also satisfies the equations of the BLT semiexplicit DAE system obtained by sequentially applying the tool $\mathcal{T}$ to the computation sequence $\mathcal{C}$.

As said in Section III-A1c, a BLT semiexplicit DAE system can be transformed to an explicit ODE, with the exception that the ODE will contain derivatives of known variables. Furthermore, after the discussion in Section III-B2, an explicit ODE always can be solved if the initial conditions are available. From this, it follows that, given $\tilde{\mathbf{y}}$, consistent initial conditions of the states in the BLT semiexplicit DAE system, i.e., $\mathbf{z}_i$ in (7), and the ability to compute all needed derivatives, the trajectory $\tilde{\mathbf{x}}$ can be computed from the BLT semiexplicit DAE system. Since $e_i \in E \setminus E_1 \cup E_2 \cup \dots \cup E_k$ and $\text{var}_X(e_i) \subseteq X' \subseteq \text{var}_X(E_1 \cup E_2 \cup \dots \cup E_k)$, the trajectory $\tilde{\mathbf{x}}$ will also satisfy e_i . We then have that $f_i(\tilde{\mathbf{x}}, \dot{\tilde{\mathbf{x}}}, \tilde{\mathbf{y}}) = 0$. Hence, with

$r = f_i(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{y})$, $\tilde{\mathbf{y}} \in \mathcal{O}(M)$ implies $r = 0$, and we can use r as residual. Thus, the BLT semiexplicit DAE system obtained from the computation sequence $\mathcal{C}$ with $\mathcal{T}$, together with e_i , is a residual generator for $M(E, X, Y)$. ■

Some important properties of a computation sequence, used in subsequential proofs, is given by the following lemma.

Lemma 2: Let $\mathcal{C} = ((V_1, E_1), (V_2, E_2), \dots, (V_k, E_k))$ be a computation sequence for the variables X' with the AE tool $\mathcal{T}$; then, $\{\text{unDiff}(V_i)\}$ is pairwise disjoint and $\text{unDiff}(V_1 \cup V_2 \cup \dots \cup V_k) = \text{var}_X(E_1 \cup E_2 \cup \dots \cup E_k)$. □

Proof: From Definition 3, we have that a system in BLT semiexplicit DAE form can be obtained by sequentially calling $\mathcal{T}$ with arguments V_i and E_i for every $(V_i, E_i) \in \mathcal{C}$. From this fact, it follows that each variable $x_j \in \text{unDiff}(V_i)$ is present in some vector $\mathbf{z}_k$ or $\mathbf{w}_l$ in the obtained BLT semiexplicit DAE system. Since the set of all vectors of known variables in a BLT semiexplicit DAE system by Definition 2 is pairwise disjoint, it follows that $\{\text{unDiff}(V_i)\}$ is pairwise disjoint, and we have shown the first claim. For the second claim, we start by noting that $V_i \subseteq \text{var}_X(E_i) \cup \text{var}_D(E_i)$ due to Definition 3. Since a system in BLT semiexplicit DAE form can be obtained from $\mathcal{C}$ and, according to Lemma 1, the solution sets of $E_1 \cup E_2 \cup \dots \cup E_k$ and the BLT semiexplicit DAE system, with respect to $V_1 \cup V_2 \cup \dots \cup V_k$, are equal and unique, it holds that each unknown variable in $E_1 \cup E_2 \cup \dots \cup E_k$, whether differentiated or undifferentiated, must be present in some V_i . From this fact and by the definitions of the operators $\text{unDiff}()$ and $\text{var}_X()$, it must also hold that $\text{unDiff}(V_1 \cup V_2 \cup \dots \cup V_k) = \text{var}_X(E_1 \cup E_2 \cup \dots \cup E_k)$. ■

For the next proof, we need some additional graph-theoretical concepts (e.g., see [24] and [26]); therefore, consider the bipartite graph $G = (E, X, A)$ describing the structure of E with respect to X (see Section II-B). A *path* on the graph G is a sequence of distinct vertices $v_1, v_2, \dots, v_n$ such that $(v_i, v_{i+1}) \in A$ and $v_i \in E \cup X$. An *alternating path* is a path in which the edges alternatively belong to a matching and not to the matching. A vertex is said to be *free* if it is not an endpoint of an edge in a matching.

Proof of Theorem 2: In this proof, we will use a characterization of an MSO set given in [19], saying that an equation set E is an MSO set if and only if E is a *proper structurally overdetermined* (PSO) set and E contains one redundant equation. Furthermore, an equation set E is a PSO set if $E = E^+$, where E^+ is the structurally overdetermined part obtained from the DM decomposition (recall Section II-C) or equivalently the equations $e \in E$ such that, for any maximal matching, there exists an alternating path between at least one free equation and e .

Returning to our case, we must show that $E_1 \cup E_2 \cup \dots \cup E_k \cup e_i$ is a PSO set and contains one redundant equation, with respect to the variables $\text{var}_X(E_1 \cup E_2 \cup \dots \cup E_k)$. We begin with the second property, i.e., that $E_1 \cup E_2 \cup \dots \cup E_k \cup e_i$ contains a redundant equation. Since $\mathcal{S} = (\mathcal{T}(\mathcal{C}), e_i)$ is a proper sequential residual generator, it follows from Definition 7 that $\mathcal{C}$ is a minimal and irreducible computation sequence for $\text{var}_X(e_i)$ with $\mathcal{T}$. If we let

$$\mathcal{C} = ((V_1, E_1), (V_2, E_2), \dots, (V_k, E_k)) \quad (26)$$

we have from Definition 3 that a system in BLT semiexplicit DAE form is obtained by sequentially calling the AE tool $\mathcal{T}$ with arguments V_i and E_i for every $(V_i, E_i) \in \mathcal{C}$. This and Assumption 2 imply that $|V_i| = |E_i|$ for every $(V_i, E_i) \in \mathcal{C}$, and hence, $\sum_{i=1}^k |V_i| = \sum_{i=1}^k |E_i|$. By the definition of the operator $\text{unDiff}()$ in (11), we can conclude that $|V_i| = |\text{unDiff}(V_i)|$, and therefore, it also holds that $\sum_{i=1}^k |\text{unDiff}(V_i)| = \sum_{i=1}^k |E_i|$. By Lemma 2, we have that $\{\text{unDiff}(V_i)\}$ is pairwise disjoint, which implies that $\sum_{i=1}^k |\text{unDiff}(V_i)| = |\text{unDiff}(V_1 \cup V_2 \cup \dots \cup V_k)|$. Definition 3 states that $\{E_i\}$ is also pairwise disjoint, and therefore, $|E_1 \cup E_2 \cup \dots \cup E_k| = \sum_{i=1}^k |E_i|$. Thus, it holds that $|\text{unDiff}(V_1 \cup V_2 \cup \dots \cup V_k)| = |E_1 \cup E_2 \cup \dots \cup E_k|$. By Lemma 2, we have that $\text{unDiff}(V_1 \cup V_2 \cup \dots \cup V_k) = \text{var}_X(E_1 \cup E_2 \cup \dots \cup E_k)$, and therefore, it also holds that $|E_1 \cup E_2 \cup \dots \cup E_k| = |\text{var}_X(E_1 \cup E_2 \cup \dots \cup E_k)|$, i.e., $E_1 \cup E_2 \cup \dots \cup E_k$ contains as many equations as unknowns. Since $\mathcal{C}$ is a computation sequence for $\text{var}_X(e_i)$ with $\mathcal{T}$, we have, from Definition 3, that $\text{var}_X(e_i) \subseteq \text{unDiff}(V_1 \cup V_2 \cup \dots \cup V_k) = \text{var}_X(E_1 \cup E_2 \cup \dots \cup E_k)$, where the last equality follows from Lemma 2, implying that adding e_i to $E_1 \cup E_2 \cup \dots \cup E_k$ will not introduce any new unknown variables, i.e., e_i is redundant. Hence, the equation set $E_1 \cup E_2 \cup \dots \cup E_k \cup e_i$ contains one more equation than unknown variables, since $|E_1 \cup E_2 \cup \dots \cup E_k \cup e_i| = |E_1 \cup E_2 \cup \dots \cup E_k| + |e_i| = |\text{var}_X(E_1 \cup E_2 \cup \dots \cup E_k)| + 1$.

We will now show that $E_1 \cup E_2 \cup \dots \cup E_k \cup e_i$ is a PSO set with respect to $\text{var}_X(E_1 \cup E_2 \cup \dots \cup E_k \cup e_i)$. To show this, we must show that, for any maximum matching on the bipartite graph describing the structure of $E_1 \cup E_2 \cup \dots \cup E_k \cup e_i$, with respect to $\text{var}_X(E_1 \cup E_2 \cup \dots \cup E_k \cup e_i)$, there exists an alternating path between a free equation and every equation in $E_1 \cup E_2 \cup \dots \cup E_k \cup e_i$. We start by constructing a maximum matching and finding a free equation. Consider the computation sequence $\mathcal{C}$ described by (26) and recall that $\mathcal{C}$, given by (26), is a minimal and irreducible computation sequence for $\text{var}_X(e_i)$ with $\mathcal{T}$. The irreducibility of $\mathcal{C}$ implies that, for each element $(V_i, E_i) \in \mathcal{C}$, it holds that the structure of E_i with respect to $\text{unDiff}(V_i)$ corresponds to an SCC. To see this, assume that (V_i, E_i) not corresponds to an SCC. This implies that it is possible to partition V_i and E_i into $V_i = V_{i1} \cup V_{i2} \cup \dots \cup V_{is}$ and $E_i = E_{i1} \cup E_{i2} \cup \dots \cup E_{is}$ so that

$$\mathcal{C}' = ((V_1, E_1), \dots, (V_{i1}, E_{i1}), \dots, (V_{is}, E_{is}), \dots, (V_k, E_k))$$

is also a computation sequence for $\text{var}_X(e_i)$ with $\mathcal{T}$, due to Assumption 3. This contradicts the irreducibility of $\mathcal{C}$, and hence, (V_i, E_i) must be an SCC. From this property, it follows, by the definition of an SCC, that there exists a maximum matching Γ_i on the bipartite graph, the structure of E_i with respect to $\text{unDiff}(V_i)$. This implies that a maximum matching (let it be denoted Γ) in the structure of $E_1 \cup E_2 \cup \dots \cup E_k$ with respect to $\text{unDiff}(V_1 \cup V_2 \cup \dots \cup V_k)$ can be constructed as $\Gamma = \bigcup_i \Gamma_i$; e.g., see [26]. By Lemma 2, we have that $\text{unDiff}(V_1 \cup V_2 \cup \dots \cup V_k) = \text{var}_X(E_1 \cup E_2 \cup \dots \cup E_k)$, and therefore, Γ is also a maximum matching in the structure of $E_1 \cup E_2 \cup \dots \cup$

E_k with respect to $\text{var}_X(E_1 \cup E_2 \cup \dots \cup E_k)$. In the first part of this proof, we concluded that the equation e_i is redundant, and therefore, Γ is also a maximum matching on the structure of $E_1 \cup E_2 \cup \dots \cup E_k \cup e_i$ with respect to $\text{var}_X(E_1 \cup E_2 \cup \dots \cup E_k \cup e_i)$, and e_i is a free equation, since it is not contained in Γ .

Since there trivially exists a path between e_i and e_i , it is sufficient to show that there exists an alternating path between the free equation e_i and every equation in $E_1 \cup E_2 \cup \dots \cup E_k$. Due to the fact that each $(V_i, E_i) \in \mathcal{C}$ corresponds to an SCC, there exists an alternating path between any two vertices, i.e., equations or variables, in the bipartite graph describing the structure of E_i with respect to $\text{unDiff}(V_i)$; e.g., see [24]. Moreover, the minimality of $\mathcal{C}$ implies that, for $(V_k, E_k) \in \mathcal{C}$, there exists at least one variable $x_m \in \text{unDiff}(V_k)$ such that $x_m \in \text{var}_X(e_i)$, since, otherwise, $\mathcal{C}' = \mathcal{C} \setminus (V_k, E_k)$ is a computation sequence for $\text{var}_X(e_i)$ and $\mathcal{C}$ is not minimal. With the same argument, we have that, for $(V_i, E_i) \in \mathcal{C}$, $i = 1, 2, \dots, k-1$, there exists at least one variable $x_m \in \text{unDiff}(V_i)$ such that either $x_m \in \text{var}_X(e_i)$ or $x_m \in \text{var}_X(E_j)$, where $(V_j, E_j) \in \mathcal{C}$, and $j \in \{i+1, i+2, \dots, k\}$. This means that there exists an alternating path between at least one variable in each $(V_i, E_i) \in \mathcal{C}$ to e_i , either directly or via one or several other $(V_j, E_j) \in \mathcal{C}$. Thus, there exists an alternating path between e_i and every equation in $E_1 \cup E_2 \cup \dots \cup E_k$. We have by this shown that $E_1 \cup E_2 \cup \dots \cup E_k \cup e_i$ is a PSO set. ■

The proof of Theorem 3 is based on the following lemma.

Lemma 3: Let $\bar{E} \subseteq E$ be an MSO set, $\mathcal{T}$ be an AE tool, $X' = \text{var}_X(\bar{E})$, and $E' = \bar{E} \setminus e_i$, where $e_i \in \bar{E}$. A minimal and irreducible computation sequence

$$\mathcal{C} = ((V_1, E_1), (V_2, E_2), \dots, (V_k, E_k))$$

for X' with $\mathcal{T}$, where $E_i \subseteq \bar{E}$, is also a minimal and irreducible computation sequence for $\text{var}_X(e_i)$ with $\mathcal{T}$. □

Proof: Assume that $\mathcal{C}$ is a minimal and irreducible computation sequence for X' with $\mathcal{T}$. First, since $e_i \in \bar{E}$ and $X' = \text{var}_X(\bar{E})$, it trivially holds that $\text{var}_X(e_i) \subseteq X'$, and hence, $\mathcal{C}$ is a computation sequence for $\text{var}_X(e_i)$ with $\mathcal{T}$. Furthermore, it directly follows from Definition 6 that $\mathcal{C}$ is an irreducible computation sequence for any subset of X' , particularly $\text{var}_X(e_i)$. To show that $\mathcal{C}$ is also a minimal computation sequence for $\text{var}_X(e_i)$, assume that there exists a computation sequence $\mathcal{C}' \subset \mathcal{C}$ for $\text{var}_X(e_i)$ with $\mathcal{T}$. Let $\bar{E}'$ and $\bar{X}' = \text{var}_X(\bar{E}')$ denote the equations and variables, contained in the elements of $\mathcal{C}'$, and note that, since $\mathcal{C}' \subset \mathcal{C}$, it holds that $\bar{E}' \subset \bar{E}$. By the argumentation in the proof of Theorem 2, we can conclude that $|\bar{E}'| = |\bar{X}'|$, i.e., $\bar{E}'$ contains as many equations as unknowns. Since $\mathcal{C}'$ is a computation sequence for $\text{var}_X(e_i)$, it must hold that $\text{var}_X(e_i) \subseteq \bar{X}'$. This means that $\bar{E}' \cup e_i$ is a structurally overdetermined set of equations with respect to $\bar{X}'$, which shows that there exists a PSO subset of $\bar{E}$. This contradicts the fact that $\bar{E}$ is an MSO set, and hence, there cannot exist a computation sequence $\mathcal{C}' \subset \mathcal{C}$ for $\text{var}_X(e_i)$ with $\mathcal{T}$. Thus, $\mathcal{C}$ is a minimal computation sequence for $\text{var}_X(e_i)$ with $\mathcal{T}$. ■

Proof of Theorem 3: Consider the model $M(E, X, Y)$ and let $(\mathcal{T}(\mathcal{C}), e_i) \in R$. Due to line 9 in FINDRESIDUALGENERA-

TORS, we can conclude that $\mathcal{C}$ is nonempty. Let

$$\mathcal{C} = ((V_1, E_1), (V_2, E_2), \dots, (V_k, E_k))$$

where $E_i \subseteq E'$, be the minimal and irreducible computation sequence for X' with $\mathcal{T}$, returned by the function FINDCOMPUTATIONSEQUENCE on line 8. Due to lines 3–7, we have that $E' = \bar{E} \setminus e_i$ and $X' = \text{var}_X(\bar{E})$, where $\bar{E} \subseteq E$ is an MSO set, and $e_i \in \bar{E} \setminus E'$. Lemma 3 then implies that $\mathcal{C}$ is also a minimal and irreducible computation sequence for $\text{var}_X(e_i)$ with $\mathcal{T}$. Now, note that, since $e_i \in \bar{E} \setminus E'$ and it holds that $\bar{E} \subseteq E$, we have that $e_i \in E \setminus E'$. Trivially, since $X' = \text{var}_X(\bar{E})$ and $X' \subseteq X$, it also holds that $\text{var}_X(e_i) \subseteq X' \subseteq X$. Thus, the computation sequence $\mathcal{C}$ for $\text{var}_X(e_i)$ with $\mathcal{T}$ and the equation e_i fulfill the prerequisites of Theorem 1. Hence, since all initial conditions are known and all needed derivatives can be computed, we can, by Theorem 1, conclude that the BLT semiexplicit DAE system obtained from $\mathcal{C}$ with $\mathcal{T}$ and e_i is a residual generator for $M(E, X, Y)$. Thus, $(\mathcal{T}(\mathcal{C}), e_i)$ is a sequential residual generator. Since, in fact, $\mathcal{C}$ is a minimal and irreducible computation sequence for $\text{var}_X(e_i)$ with $\mathcal{T}$, $(\mathcal{T}(\mathcal{C}), e_i)$ is a proper sequential residual generator. ■

Proof of Theorem 4: On line 3 in FINDCOMPUTATIONSEQUENCE, the SCCs of the structure of E' with respect to X' are computed. If we assume that the structure contains s SCCs, the ordered set returned by the function FINDALLSCC can be written as

$$S = ((E_1, X_1), (E_2, X_2), \dots, (E_s, X_s)) \quad (27)$$

where each element $(E_i, X_i) \in S$ corresponds to an SCC of the structure of E' with respect to X' . Note that, since E' is just determined with respect to X' , the SCCs of the structure of E' with respect to X' are unique; see Section II-C. As said in Section V-C, we assume that the SCCs in S are ordered according to Fig. 1. Note that this ordering implies the following important property:

$$\text{var}_X E_i \cap \{X_{i+1} \cup X_{i+2} \cup \dots \cup X_s\} = \emptyset \quad (28)$$

for $i = 1, 2, \dots, s-1$. On lines 6–8, the variables in X_i are partitioned into differentiated variables Z_i and undifferentiated variables W_i , i.e., $X_i = \text{unDiff}(Z_i) \cup W_i$, where Z_i contains variables that appear as differentiated in some equation in E_i . On lines 12–14, a corresponding partitioning of the equations in E_i into $E_i = E_{Z_i} \cup E_{W_i}$ is done, where E_{Z_i} are equations that contain any of the differentiated variables Z_i , and E_{W_i} are equations that do not contain any of the differentiated variables Z_i but may contain variables from $\text{unDiff}(Z_i)$. Now, note that, due to the assumptions regarding the model in Section II, each equation in E_{Z_i} contains only one differentiated, which, furthermore, is only present in that equation. This means, first, that E_{Z_i} is just determined with respect to the variables in Z_i and, second, that the structure of E_{Z_i} with respect to Z_i only contains SCCs of size one. On line 14, these SCCs are computed. Assuming that the structure contains s_i SCCs,

the ordered set returned by FINDALLSCC on line 14 can be written as

$$S_{Z_i} = ((Z_i^1, E_{Z_i}^1), (Z_i^2, E_{Z_i}^2), \dots, (Z_i^{s_i}, E_{Z_i}^{s_i})). \quad (29)$$

Due to line 23, we know that the equation set E_{W_i} is just determined with respect to W_i , and hence, the structure of E_{W_i} with respect to W_i can uniquely be partitioned into SCCs. On line 24, these SCCs are computed, and as above, the ordered set of SCCs can be written as

$$S_{W_i} = ((W_i^1, E_{W_i}^1), (W_i^2, E_{W_i}^2), \dots, (W_i^{p_i}, E_{W_i}^{p_i})). \quad (30)$$

Furthermore, as in the case with the set S in (27), the ordering of the SCCs in S_{W_i} implies that

$$\text{var}_X(E_{W_i}^j) \cap \{W_i^{j+1} \cup W_i^{j+2} \cup \dots \cup W_i^{p_i}\} = \emptyset \quad (31)$$

for $j = 1, 2, \dots, p_i - 1$. From the discussion above, we have that a nonempty $\mathcal{C}$ returned by FINDCOMPUTATIONSEQUENCE has the form

$$\begin{aligned} \mathcal{C} = & ((Z_1^1, E_{Z_1}^1), (Z_1^2, E_{Z_1}^2), \dots, (Z_1^{s_1}, E_{Z_1}^{s_1}), \\ & (W_1^1, E_{W_1}^1), (W_1^2, E_{W_1}^2), \dots, (W_1^{p_1}, E_{W_1}^{p_1}), \dots, \\ & (Z_2^1, E_{Z_2}^1), (Z_2^2, E_{Z_2}^2), \dots, (Z_2^{s_2}, E_{Z_2}^{s_2}), \\ & (W_2^1, E_{W_2}^1), (W_2^2, E_{W_2}^2), \dots, (W_2^{p_2}, E_{W_2}^{p_2}), \dots, \\ & (Z_s^1, E_{Z_s}^1), (Z_s^2, E_{Z_s}^2), \dots, (Z_s^{s_s}, E_{Z_s}^{s_s}), \\ & (W_s^1, E_{W_s}^1), (W_s^2, E_{W_s}^2), \dots, (W_s^{p_s}, E_{W_s}^{p_s})) \end{aligned} \quad (32)$$

where every $(Z_i^j, E_{Z_i}^j) \in \mathcal{C}$, and $(W_i^j, E_{W_i}^j) \in \mathcal{C}$ corresponds to an SCC.

We will now utilize Definition 3 to show that the ordered set $\mathcal{C}$ in (32) is a computation sequence for X' with $\mathcal{T}$. First, note that $Z_i^j \subseteq \text{var}_D(E_{Z_i}^j)$ and $W_i^j \subseteq \text{var}_X(E_{W_i}^j)$. When the structure of a just-determined equation set with respect to a set of variables is decomposed into its SCCs, unique partitions of the equation and variable sets are also obtained; see, for example, [25] and Fig. 1 for an illustration. From this fact, it follows that every equation in E' is present in some E_i in (27) only once. When the equations in E_i are split into differential equations E_{Z_i} and algebraic equations E_{W_i} on line 13, it is guaranteed that $E_{Z_i} \cap E_{W_i} = \emptyset$. Moreover, again, due to the fact that a decomposition into SCCs gives a unique partition of the equation and variable sets, we have that every equation in E_{Z_i} is present in some equation set $E_{Z_i}^j$ in (29) only once and that every equation in E_{W_i} is present in some $E_{W_i}^j$ in (30) only once. Thus, we can conclude that each equation in E' is contained in only one equation set in $\mathcal{C}$, that is, all equation sets in $\mathcal{C}$ are disjoint. Hence, the ordered set $\mathcal{C}$ fulfills the prerequisites in Definition 3. According to conditions 1 and 2 in Definition 3, $\mathcal{C}$ is a computation sequence for X' with $\mathcal{T}$ if

$$X' \subseteq \bigcup_{i=1}^s \left(\bigcup_{j=1}^{s_i} \text{unDiff}(Z_i^j) \cup \bigcup_{j=1}^{p_i} W_i^j \right) \quad (33)$$

and a system in BLT semiexplicit DAE form is obtained by sequentially calling the tool $\mathcal{T}$, with arguments Z_i^j and $E_{Z_i}^j$ for every element $(Z_i^j, E_{Z_i}^j) \in \mathcal{C}$ and with arguments W_i^j and $E_{W_i}^j$ for every element $(W_i^j, E_{W_i}^j) \in \mathcal{C}$.

We start by showing condition 1, i.e., (33). From the fact mentioned above that a decomposition of a structure into its SCCs also induces a partitioning of the corresponding equation and variable sets, it follows that every variable in X' is present in some X_i in (27). That is, we have that $X' = \bigcup_i^s X_i$. When the variables in X_i are split into differentiated variables Z_i and undifferentiated variables W_i , it holds that $X_i = \text{unDiff}(Z_i) \cup W_i$. In addition, it holds that every variable in Z_i is present in some variable set Z_i^j in (29) and that every variable in W_i is present in some W_i^j in (30), so that $Z_i = \bigcup_{j=1}^{s_i} Z_i^j$ and $W_i = \bigcup_{j=1}^{p_i} W_i^j$. Hence

$$\begin{aligned} X' &= \bigcup_i^s X_i = \bigcup_i^s (\text{unDiff}(Z_i) \cup W_i) \\ &= \bigcup_i^s \left(\text{unDiff} \left(\bigcup_{j=1}^{s_i} Z_i^j \right) \cup \bigcup_{j=1}^{p_i} W_i^j \right) \\ &= \bigcup_i^s \left(\bigcup_{j=1}^{s_i} \text{unDiff}(Z_i^j) \cup \bigcup_{j=1}^{p_i} W_i^j \right) \end{aligned} \quad (34)$$

where the last equality trivially follows from the definition of $\text{unDiff}()$ in (11). The property (33) and, thus, condition 1 have then been verified.

Condition 2 of Definition 3 will now be verified, that is, that $\mathcal{C}$ can be used to obtain a system in BLT semiexplicit DAE form. Consider an element $(Z_i^j, E_{Z_i}^j) \in \mathcal{C}$. Since $E_{Z_i}^j \subseteq E_{Z_i} \subseteq E_i$ and we have that $(X_i, E_i) \in S$, (28) implies that

$$\text{var}_X(E_{Z_i}^j) \cap \{X_{i+1} \cup X_{i+2} \cup \dots \cup X_s\} = \emptyset \quad (35)$$

for $i = 1, 2, \dots, s - 1$. From lines 17–21 in the algorithm, it follows that the AE tool $\mathcal{T}$ can be used to solve the equations in $E_{Z_i}^j$ for the variables in Z_i^j . Since we have assumed that each differential equation contains at most one differentiated variable and (35) holds, we can use $(Z_i^j, E_{Z_i}^j) \in \mathcal{C}$ and the AE tool $\mathcal{T}$ to obtain

$$\dot{\mathbf{z}}_i^j = \mathbf{g}_i^j(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_i, \mathbf{y}) \quad (36)$$

where $\mathbf{z}_i^j$ is a vector of the variables in Z_i^j , $\mathbf{x}_k$ is a vector of the variables in X_k , $\mathbf{y}$ is a vector of the known variables in E' , and $\mathbf{g}_i^j$ is a function returned by $\mathcal{T}$ when the arguments are Z_i^j and $E_{Z_i}^j$. From the elements $(Z_i^j, E_{Z_i}^j) \in \mathcal{C}$, $j = 1, 2, \dots, s_i$, we can thus, by using (36) and also $X_i = \text{unDiff}(Z_i) \cup W_i$, obtain

$$\dot{\mathbf{z}}_i = \mathbf{g}_i(\mathbf{z}_1, \mathbf{z}_1, \dots, \mathbf{z}_i, \mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_i, \mathbf{y}) \quad (37)$$

where $\mathbf{z}_i = (\mathbf{z}_i^1, \mathbf{z}_i^2, \dots, \mathbf{z}_i^{s_i})$ is a vector of the variables in Z_i , $\mathbf{w}_i$ is a vector of the variables in W_i , $\mathbf{y}$ is a vector of the known variables in E' , and $\mathbf{g}_i = (\mathbf{g}_i^1, \mathbf{g}_i^2, \dots, \mathbf{g}_i^{s_i})$.

Now, instead, consider an element $(W_i^j, E_{W_i}^j) \in \mathcal{C}$. Since also $(W_i^j, E_{W_i}^j) \in S_{W_i}$, where S_{W_i} is given by (30), (31) holds. Since $E_{W_i}^j \subseteq E_{W_i} \subseteq E_i$ and $(X_i, E_i) \in \mathcal{S}$, we also have that

$$\text{var}_X(E_{W_i}^j) \cap \{X_{i+1} \cup X_{i+2} \cup \dots \cup X_s\} = \emptyset \quad (38)$$

for $i = 1, 2, \dots, s-1$. By using the fact that the AE tool $\mathcal{T}$ can solve $E_{W_i}^j$ for W_i^j due to lines 27–31 and that $X_i = \text{unDiff}(Z_i) \cup W_i$ and $\text{var}_D(E_{W_i}) \cap Z_i = \emptyset$ due to lines 6–8 and 12–14 and then utilizing (31) and (38), we can obtain

$$\mathbf{w}_i^j = \mathbf{h}_i^j(\dot{\mathbf{w}}_1, \dots, \dot{\mathbf{w}}_{i-1}, \mathbf{z}_1, \dots, \mathbf{z}_i, \mathbf{w}_1, \dots, \mathbf{w}_{i-1}, \mathbf{w}_i^1, \dots, \mathbf{w}_i^{j-1}, \mathbf{y}) \quad (39)$$

from $(W_i^j, E_{W_i}^j) \in \mathcal{C}$, where $\mathbf{w}_i^j$ is a vector of the variables in W_i^j , $\mathbf{z}_i$ is a vector of the variables in Z_i , and $\mathbf{h}_i^j$ is a function returned by $\mathcal{T}$ when the arguments are W_i^j and $E_{W_i}^j$. Note that the absence of vectors $\mathbf{z}_i$ in (39) is a direct implication of the assumption that each differentiated variable is present in only one equation in the original model and, therefore, also in the BLT semiexplicit DAE system. Since $\mathbf{z}_i$ is obviously present in (37), it cannot be present in (39).

By using (39), we can then obtain

$$\begin{aligned} \mathbf{w}_i^1 &= \mathbf{h}_i^1(\dot{\mathbf{w}}_1, \dots, \dot{\mathbf{w}}_{i-1}, \mathbf{z}_1, \dots, \mathbf{z}_i, \mathbf{w}_1, \dots, \mathbf{w}_{i-1}, \mathbf{y}) \\ \mathbf{w}_i^2 &= \mathbf{h}_i^2(\dot{\mathbf{w}}_1, \dots, \dot{\mathbf{w}}_{i-1}, \mathbf{z}_1, \dots, \mathbf{z}_i, \mathbf{w}_1, \dots, \mathbf{w}_{i-1}, \mathbf{w}_i^1, \mathbf{y}) \\ &\vdots \\ \mathbf{w}_i^{p_i} &= \mathbf{h}_i^{p_i}(\dot{\mathbf{w}}_1, \dots, \dot{\mathbf{w}}_{i-1}, \mathbf{z}_1, \dots, \mathbf{z}_i, \mathbf{w}_1, \dots, \mathbf{w}_{i-1}, \mathbf{w}_i^1, \dots, \mathbf{w}_i^{p_i-1}, \mathbf{y}) \end{aligned} \quad (40)$$

from the elements $(W_i^j, E_{W_i}^j) \in \mathcal{C}$, $j = 1, 2, \dots, p_i$. Comparing (37) and (40) with the system in Definition 2 shows that the elements $(Z_i^j, E_{Z_i}^j) \in \mathcal{C}$, $j = 1, 2, \dots, s_i$, and $(W_i^j, E_{W_i}^j) \in \mathcal{C}$, $j = 1, 2, \dots, p_i$, correspond to the i th block of a BLT semiexplicit DAE form. Applying the above arguments for $i = 1, 2, \dots, s$ then implies that the ordered set $\mathcal{C}$ in (32) can be used to obtain a system in BLT semiexplicit DAE form with s blocks. Thus, $\mathcal{C}$ is a computation sequence for X' with $\mathcal{T}$.

It now remains to show that $\mathcal{C}$ is a minimal and irreducible computation sequence for X' with $\mathcal{T}$. We begin with the irreducibility of $\mathcal{C}$. In the beginning of this proof, we showed that all elements of $\mathcal{C}$, given by (32), correspond to SCCs. We have also concluded that, due to the assumptions regarding the model in Section I, all elements $(Z_i^j, E_{Z_i}^j) \in \mathcal{C}$ are of size one, i.e., trivially irreducible. Now, consider an element $(W_i^j, E_{W_i}^j) \in \mathcal{C}$ and assume that we partition W_i^j as $W_i^j = W_{i1}^j \cup W_{i2}^j$ and $E_{W_i}^j$ as $E_{W_i}^j = E_{W_{i1}}^j \cup E_{W_{i2}}^j$ and form the two new elements $(W_{i1}^j, E_{W_{i1}}^j)$ and $(W_{i2}^j, E_{W_{i2}}^j)$. Due to the fact that $(W_i^j, E_{W_i}^j)$ corresponds to an SCC, $E_{W_i}^j$ is a dependent equation set with respect to the variables in W_i^j . This implies that when applying

$\mathcal{T}$ to the elements $(W_{i1}^j, E_{W_{i1}}^j)$ and $(W_{i2}^j, E_{W_{i2}}^j)$, we obtain the following two equations:

$$\begin{aligned} \mathbf{w}_{i1}^j &= \mathbf{h}_{i1}^j(\dots, \mathbf{w}_{i2}^j, \dots) \\ \mathbf{w}_{i2}^j &= \mathbf{h}_{i2}^j(\dots, \mathbf{w}_{i1}^j, \dots) \end{aligned}$$

which do not clearly have the structure of equations contained in a BLT semiexplicit DAE system, due to the cyclic dependence between the equations. Hence, a system in BLT semiexplicit DAE form cannot be obtained when the element $(W_i^j, E_{W_i}^j) \in \mathcal{C}$ is partitioned, which violates condition 2 in Definition 3. We can then conclude that no elements of $\mathcal{C}$ can further be partitioned, and hence, $\mathcal{C}$ is an irreducible computation sequence for X' with $\mathcal{T}$.

The minimality of $\mathcal{C}$ for X' with $\mathcal{T}$ trivially follows from the fact that (34) holds. Since (34) is fulfilled, all elements in $\mathcal{C}$ are needed to compute the variables in X' . This implies that any attempt to form a computation sequence for X' with $\mathcal{T}$ by using a subset of $\mathcal{C}$ will violate condition 1 in Definition 3. This completes the proof. ■

REFERENCES

- [1] United Nations, Regulation no. 49: Uniform provisions concerning the measures to be taken against the emission of gaseous and particulate pollutants from compression ignition engines for use in vehicles, and the emission of gaseous pollutants from positive-ignition engines fuelled with natural gas or liquefied petroleum gas for use in vehicles, 2008, ECE-R49.
- [2] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki, *Diagnosis and Fault-Tolerant Control*, 2nd ed. New York: Springer-Verlag, 2006.
- [3] M. Nyberg and M. Krysander, "Statistical properties and design criterions for AI-based fault isolation," in *Proc. 17th IFAC World Congr.*, Seoul, Korea, 2008, pp. 7356–7362.
- [4] M. Nyberg, "Automatic design of diagnosis systems with application to an automotive engine," *Control Eng. Pract.*, vol. 7, no. 8, pp. 993–1005, Aug. 1999.
- [5] M. Staroswiecki and P. Declerck, "Analytical redundancy in non-linear interconnected systems by means of structural analysis," in *Proc. IFAC AIPAC*, Nancy, France, 1989, pp. 51–55.
- [6] D. Dustegor, V. Cocquempot, and M. Staroswiecki, "Structural analysis for residual generation: Towards implementation," in *Proc. IEEE Int. Conf. Control Appl.*, 2004, pp. 1217–1222.
- [7] R. Izadi-Zamanabadi, "Structural analysis approach to fault diagnosis with application to fixed-wing aircraft motion," in *Proc. Amer. Control Conf.*, 2002, vol. 5, pp. 3949–3954.
- [8] V. Cocquempot, R. Izadi-Zamanabadi, M. Staroswiecki, and M. Blanke, "Residual generation for the ship benchmark using structural approach," in *Proc. UKACC Int. Conf. Control*, Sep. 1998, pp. 1480–1485.
- [9] C. Svärd and H. Wassén, "Development of methods for automatic design of residual generators," M.S. thesis, Linköpings Universitet, Linköping, Sweden, 2006.
- [10] J. Hansen and J. Molin, "Design and evaluation of an automatically generated diagnosis system," M.S. thesis, Linköpings Universitet, Linköping, Sweden, 2006.
- [11] J. Cassar and M. Staroswiecki, "A structural approach for the design of failure detection and identification systems," in *Proc. IFAC Control Ind. Syst.*, Belfort, France, 1997, pp. 841–846.
- [12] M. Staroswiecki, *Fault Diagnosis and Fault Tolerant Control*. Oxford, U.K.: Encyclopedia of Life Support Systems (EOLSS Publishers), 2002, ch. Structural Analysis for Fault Detection and Isolation and for Fault Tolerant Control.
- [13] B. Pulido and C. Alonso-González, "Possible conflicts: A compilation technique for consistency-based diagnosis," *IEEE Trans. Syst., Man, Cybern. B, Cybern.—Special Issue Diagnosis Complex Syst.*, vol. 34, no. 5, pp. 2192–2206, Oct. 2004.
- [14] S. Ploix, M. Desinde, and S. Touaf, "Automatic design of detection tests in complex dynamic systems," in *Proc. 16th IFAC World Congr.*, Prague, Czech Republic, 2005.

- [15] L. Travé-Massuyès, T. Escobet, and X. Olive, "Diagnosability analysis based on component-supported analytical redundancy," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 36, no. 6, pp. 1146–1160, Nov. 2006.
- [16] G. Katsilllis and M. Chantler, "Can dependency-based diagnosis cope with simultaneous equations?" in *Proc. 8th Int. Workshop Principles Diagnosis (DX)*, Le Mont-Saint-Michel, France, 1997, pp. 51–59.
- [17] R. W. Brockett, *Finite-Dimensional Linear Systems*. New York: Wiley, 1970.
- [18] C. De Persis and A. Isidori, "A geometric approach to nonlinear fault detection and isolation," *IEEE Trans. Autom. Control*, vol. 46, no. 6, pp. 853–865, Jun. 2001.
- [19] M. Krysander, J. Åslund, and M. Nyberg, "An efficient algorithm for finding minimal over-constrained sub-systems for model-based diagnosis," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 1, pp. 197–206, Jan. 2008.
- [20] B. Pulido, A. Bregón, and C. Alonso, "Combining state estimation and simulation in consistency-based diagnosis using possible conflicts," in *Proc. 19th Int. Workshop Principles Diagnosis (DX)*, Blue Mountains, Australia, 2008, pp. 339–346.
- [21] B. Pulido, C. Alonso, A. Bregón, V. Puig, and T. Escobet, "Analyzing the influence of temporal constraints in possible conflicts calculation for model-based diagnosis," in *Proc. 18th Int. Workshop Principles Diagnosis (DX)*, Nashville, TN, 2007, pp. 186–193.
- [22] F. Cellier and H. Elmqvist, "Automated formula manipulation supports object-oriented continuous-system modeling," *IEEE Control Syst. Mag.*, vol. 13, no. 2, pp. 28–38, Apr. 1993.
- [23] S. Narasimhan and G. Biswas, "Model-based diagnosis of hybrid systems," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 37, no. 3, pp. 348–361, May 2007.
- [24] A. S. Asratian, T. M. J. Denley, and R. Häggkvist, *Bipartite Graphs and Their Applications*. Cambridge, U.K.: Cambridge Univ. Press, 1998.
- [25] A. Dulmage and N. Mendelsohn, "Coverings of bi-partite graphs," *Can. J. Math.*, vol. 10, pp. 517–534, 1958.
- [26] K. Murota, *System Analysis by Graphs and Matroids*. Berlin, Germany: Springer-Verlag, 1987.
- [27] M. Krysander and E. Frisk, "Sensor placement for fault diagnosis," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 6, pp. 1398–1410, Nov. 2008.
- [28] P. Kunkel and V. Mehrmann, *Differential-Algebraic Equations—Analysis and Numerical Solution*. Zürich, Switzerland: Eur. Math. Soc., 2006.
- [29] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. Philadelphia, PA: SIAM, 1989.
- [30] U. M. Ascher and L. M. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Philadelphia, PA: SIAM, 1998.
- [31] S. Mattson, H. Elmqvist, and M. Otter, "Physical system modeling with Modelica," *Control Eng. Pract.*, vol. 6, no. 4, pp. 501–510, Apr. 1998.
- [32] D. V. Steward, "On an approach to techniques for the analysis of the structure of large systems of equations," *SIAM Rev.*, vol. 4, no. 4, pp. 321–342, Oct. 1962.
- [33] G. Kron, *Diakoptics—The Piecewise Solution of Large-Scale Systems*. London, U.K.: Macdonald, 1963.
- [34] D. V. Steward, "Partitioning and tearing systems of equations," *SIAM J. Numer. Anal.*, vol. 2, no. 2, pp. 345–365, Jan. 1965.
- [35] P. Fritzson, *Principles of Object-Oriented Modeling and Simulation With Modelica 2.1*. Piscataway, NJ: IEEE Press, 2004.
- [36] E. Hairer and G. Wanner, *Solving Ordinary Equations II—Stiff and Differential-Algebraic Problems*. New York: Springer-Verlag, 2002.
- [37] F. E. Cellier and E. Kofman, *Continuous System Simulation*. New York: Springer-Verlag, 2006.
- [38] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*. Philadelphia, PA: SIAM Classics, 2000.
- [39] H. K. Khalil, *Nonlinear Systems*. Englewood Cliffs, NJ: Prentice-Hall, 2002.
- [40] T. Wei and M. Li, "High order numerical derivatives for one-dimensional scattered noisy data," *Appl. Math. Comput.*, vol. 175, no. 2, pp. 1744–1759, Apr. 2006.
- [41] L. Barford, E. Manders, G. Biswas, P. Mosterman, V. Ram, and J. Barnett, "Derivative estimation for diagnosis HP Labs technical reports," HP Lab., Palo Alto, CA, Tech. Rep. HPL-1999-18, 1999.
- [42] R. Tarjan, "Depth first search and linear graph algorithms," *SIAM J. Comput.*, vol. 1, no. 2, pp. 146–160, 1972.
- [43] E. Frisk, M. Krysander, M. Nyberg, and J. Åslund, "A toolbox for design of diagnosis systems," in *Proc. IFAC SafeProcess*, Beijing, China, 2006.
- [44] J. Wahlström, "Control of EGR and VGT for emission control and pumping work minimization in diesel engines," M.S. thesis, Linköpings Universitet, Linköping, Sweden, 2006.

Carl Svärd was born in Arbrå, Sweden, in 1981. He received the M.Sc. degree in applied physics and electrical engineering in 2006 from Linköping University, Linköping, Sweden, where he is currently working toward the Ph.D. degree with the Vehicular Systems Division, Department of Electrical Engineering.

He is part of an industrial research project in collaboration with Scania CV AB, Södertälje, Sweden. His research interests are within model-based diagnosis, residual generation, fault-tolerant

control, and industrial applications.

Mattias Nyberg was born in Karlstad, Sweden, in 1970. He received the M.Sc. degree in computer science and engineering from Linköping University, Linköping, Sweden, in 1994, the M.Sc. degree in electrical engineering and applied physics from Case Western Reserve University, Cleveland, OH, in 1994, and the Ph.D. degree in electrical engineering from Linköping University in 1999.

He was with Daimler Chrysler Research, Stuttgart, Germany, for one year as a Postdoctoral Researcher.

He is currently partly an Associate Professor with the Department of Electrical Engineering, Linköping University, and partly a Technical Manager of On-Board Diagnosis with Scania CV AB, Södertälje, Sweden, where he is responsible for the development and research in onboard diagnosis. His research interest is model-based diagnosis from the perspectives of both computer science and control theory. In particular, he is interested in its uses for fault-tolerant control of mechatronic systems and decision support for repair mechanics.