Generating Semi-Explicit DAEs with Structural Index 1 for Fault Diagnosis Using Structural Analysis

Georgios Zogopoulos Papaliakos and Kostas J. Kyriakopoulos

Abstract— **Structural Analysis is a lucrative option for Fault Detection and Identification in Unmanned Aerial Vehicles (UAVs), because it handles detailed, large-scale mathematical models. It can be employed by an on-board flight computer to generate residual generators and implement automatic fault-detection. Contemporary algorithms applied on dynamic systems may yield residual generators which require the real-time solution of Differential-Algebraic Equation (DAE) systems. Depending on the form and differential index of each DAE system, its solution may not be possible exclusively by computational means. In this paper we explore the relation between Structural Analysis algorithms and the forms of DAE systems they produce, propose conditions under which all generated DAEs are Structural Index-1 and semi-explicit and provide a large-scale fixed-wing UAV model with that property.**

I. INTRODUCTION

As Unmanned Aerial Vehicles (UAVs) become deeply integrated into the civil airspace, they are required to exhibit a comparable level of autonomy to manned aircraft, in terms of navigating a dynamic environment and addressing faults.

A fault is a deviation of the system parameters or the system structure from the nominal condition. If action is not taken against it, it may result to failures, rendering the system inoperable [1], [2]. Indeed, regarding UAVs, component fault is a much more common accident cause than human error [3],[4]. But since human pilots cannot perceive the system state directly and persistently, in contrast to manned aircraft, UAVs must detect faults automatically, as early as possible.

The discipline of Fault Diagnosis (FD) establishes mathematical and physical structures that are able to detect when a fault occurs in a system. Afterwards, fault isolation methodologies can be performed to identify the system component which is at fault. This information is vital for any fault-tolerant control scheme. Both detection and isolation procedures, in a consistency-based (also known as parity-based) continuous-time diagnosis context, utilize *residual* signals $r(t)$. Given the combined input-output vector $\mathbf{y}$ and redundant model knowledge, one can exploit the ability to calculate a quantity with more than one way and design a *residual generator* function $r(t) = f(\mathbf{y}(t))$, such that under no-fault conditions $r(t) = 0$ should hold and vice versa. [1],[5],[2]. Analytical redundancy is especially useful in commercial UAV applications, where low cost and weight are primary requirements and finding Analytical Redundancy Relations (ARRs) which can be used as residual generators is a primary goal [5],[6]. Even though there have been extensive studies covering linear systems, ARR-based FD methods on non-linear systems are still under development [7],[8].

Detailed diagnosis requires large-scale mathematical models. Manually extracting the maximum number of ARRs from such large models is impossible in real-time, if at all [9],[10],[11], yet swift response is valuable in restructurable and self-repairing systems. Algorithms for automated ARR extraction on aircraft have been recently proposed, based on Structural Analysis (SA) [2],[1],[7]. However, not all residual generators generated by SA techniques are valid for implementation by an automated computing system [8],[12],[11]. In the case where ARRs include dynamic loops (dynamic systems), a Differential Algebraic Equation system (DAE) may need to be solved to calculate the corresponding residual. This numerical problem is known to have severe theoretical difficulties, especially in non-linear systems, such as UAVs [12],[13],[8].

Even though this problem is relevant to any dynamic model, in this work we focus on the application of the method on a fixed-wing UAV with electric propulsion. We propose sufficient conditions under which a model is a semi-explicit DAE with Structural Index 1 and provide a large-scale model which is compliant to these conditions and ready to be parsed by SA algorithms and numerical solution methods.

In section II, the SA methodology for residual generator extraction is presented and the resulting ARRs are formally described. In section III, DAEs are briefly introduced and conditions on the model structure for simple automated evaluation are presented. In section IV, the proposed UAV model is provided. In section V, realistic faults and sensor inputs are prescribed for the model and fault detection is performed on a simulation of the proposed system. Finally, section VI concludes the paper.

II. STRUCTURAL ANALYSIS

The ability of an on-board fault diagnostic system to automatically extract residual generators is a very desirable feature, because it allows it to respond to changes in the model in real-time. However, embedded computing systems don't have the ability of processing the symbolic equations which constitute a model within reasonable time and resource constraints. Instead, SA is employed, for which a more extensive introduction can be found in the authors' previous work [11] and in [1]. Definitions required for this work are presented in this section.

The authors are with Control Systems Lab, School of Mechanical Engineering, National Technical University of Athens, Greece {gzogop, kkyria}@mail.ntua.gr

A. The Structural Graph

SA is a methodology for abstracting the mathematical model of a system into a qualitative model which describes whether there exist relations between model equations (also referred to as constraints) and model variables. The resulting model is commonly structured as a bipartite graph. Even though the bipartite graph contains less information than the original model, it is a form suitable for processing by automated algorithms [1],[7].

Given a mathematical system model, the initial set of its constraints $\mathcal{C}_0$ with elements c_i and the initial set of its variables $\mathcal{X}_0$ with elements x_j is considered. We denote the set of variables which appear in c_i as $var(c_i)$ and the set of equations which include x_j as $eqs(x_j)$. Solving a c_i for a scalar x_j (if possible) results in the evaluation $x_j = f_{i,j}(\mathbf{x})$. Solving c_i for zero results in the evaluation $0 = f_{i,0}(\mathbf{x})$.

An assumption is made to ensure that the constraints are always well-defined:

Assumption 1 (Constraint Domain): Let there be a constraint $c_i : \mathbb{D}_i \to \mathbb{R}$. It should hold that $\mathbb{T} \subseteq \mathbb{D}_i$, where $\mathbb{T}$ is the trajectory space of $var(c_i)$. Re-wording, the system should not enter a state which would render a constraint undefined.

Definition 1 (Variables Solvable by a Constraint): Given a constraint c_i and the set of its variables $var(c_i)$, the set of variables for which the constraint can be solved is defined as:

$$var_s(c_i) = \{x_j \in var(c_i) : \exists f_{i,j}\} \tag{1}$$

Example 1 (Solving a Constraint): Given a constraint c_i describing the derivative of the body velocity x-component of an aircraft [14]

$$\dot{u} = rv - qw + F_x/m \tag{2}$$

$r \notin var_s(c_i)$, because $f_{i,r}$ is not defined in all of $\mathbb{T}$ (which includes $v = 0$).□

In practice, one might choose to limit $var_s(\cdot)$ even more during implementation, because of numerical stability issues.

The methods employed in this work require the extension of $\mathcal{C}_0$ with explicit first-order differentiation equations for those variables whose derivatives appear in the system model (e.g. $\dot{x} = d/dt \cdot x$) [1]. Let the set of these explicit differentiation equations be $\mathcal{D}$, with elements d_i.

$$\mathcal{C} = \mathcal{C}_0 \cup \mathcal{D} \tag{3}$$

The variables set is accordingly extended with the variable derivatives $\mathcal{X}_D$.

$$\mathcal{X} = \mathcal{X}_0 \cup \mathcal{X}_D \tag{4}$$

Since $\mathcal{D}$ is comprised solely of first-order differentiations, $\mathcal{X}_D$ includes only first-order derivatives.

The variables set is partitioned into $\mathcal{X}{=}\mathcal{X}_U \bigcup \mathcal{X}_K$, where:

- $\mathcal{X}_U$ is the set of unknown variables
- $\mathcal{X}_K$ is the set of known variables, e.g. measurements, inputs and constants

For the purposes of our analysis the set of known variables $\mathcal{X}_K$ can be disregarded and discarded, without loss of structural information [1].

In this work, the structural graph is defined as a *partially directed* bipartite graph $\mathbf{G}{=}(\mathcal{C}, \mathcal{X}, \mathcal{E})$ with vertex sets $\mathcal{C}$ and $\mathcal{X}$ and an edge set $\mathcal{E}{=}\{e_{ij}{=}(c_i, x_j) : x_j \in var_s(c_i)\} \cup \{e_{ji}{=}(x_j, c_i) : c_i \in eqs(x_j)\}$, to reflect Assumption 1.

B. Graph Matching

An intermediate step towards extracting residual generators from the structural graph is to produce *matching* sets. A matching is a subset of $\mathcal{E}$ such that $\mathcal{M}{=}\{m_i{=}(c_i, x_i) \in \mathcal{E} \; | m_i{\neq}m_j \text{ iff } c_i{\neq}c_j \land x_i{\neq}x_j, \forall i, j\}$. In other words, a matching is a set of edges such that, any two edges do not have a variable or a constraint in common.

Since most matching algorithms are applied on undirected graphs, the following terminology is emphasized:

Definition 2 (Realizable Matching): Let there be a *partially directed* graph $\mathbf{G} = \{\mathcal{C}, \mathcal{X}, \mathcal{E}\}$ and a matching edge $m_k \in \mathcal{M}$. m_k is *realizable* iff $m_k \in \mathcal{E}$. Similarly, $\mathcal{M}$ is realizable iff every m_k is realizable [10].

Assumption 2: For the rest of this paper, only realizable matchings will be admitted onto a structural graph.

For a given $\mathbf{G}$ and an $\mathcal{M}$ onto it, if $|\mathcal{M}| = |\mathcal{X}|$ or $|\mathcal{M}| = |\mathcal{C}|$, then the matching is called *complete* with respect to $\mathcal{X}$ or to $\mathcal{C}$ respectively. If $|\mathcal{M}| = |\mathcal{X}| = |\mathcal{C}|$ then the matching is *perfect*.

For any *undirected* bipartite graph, a unique decomposition is defined, called the Dulmage-Mendelsohn (DM) decomposition. It identifies three (possibly empty) subgraph components:

$$\begin{aligned}
\mathbf{G}^- &= \left(\mathcal{C}^-, \mathcal{X}^-, \mathcal{E}^-\right), |\mathcal{C}^-| < |\mathcal{X}^-| \\
\mathbf{G}^0 &= \left(\mathcal{C}^0, \mathcal{X}^0, \mathcal{E}^0\right), |\mathcal{C}^0| = |\mathcal{X}^0| \\
\mathbf{G}^+ &= \left(\mathcal{C}^+, \mathcal{X}^+, \mathcal{E}^+\right), |\mathcal{C}^+| > |\mathcal{X}^+|
\end{aligned}$$

The decomposition guarantees that there exists a complete matching (not necessarily unique) on $\mathcal{C}^-$ in $\mathbf{G}^-$, a perfect matching in $\mathbf{G}^0$ and a complete matching on $\mathcal{X}^+$ in $\mathbf{G}^+$. $\mathbf{G}^-$ is called the *under-constrained* part of $\mathbf{G}$, $\mathbf{G}^0$ *just-constrained* and $\mathbf{G}^+$ *over-constrained* [15].

Given a matching $\mathcal{M}$, a *directed* graph $\mathbf{G}_d{=}\{\mathcal{C}, \mathcal{X}, \mathcal{E}_d\}$ can be constructed, with edges defined as $\mathcal{E}_d{=}\{e_{ij}{=}(c_i, x_j) : e_{ij}{\in}\mathcal{M}\} \bigcup \{e_{ji}{=}(x_j, c_i){:}(c_i, x_j){\notin}\mathcal{M}\}$, $e_{ij}{\in}\mathcal{E}$, i.e. matching edges are directed from $\mathcal{X}$ to $\mathcal{C}$ and the rest from $\mathcal{C}$ to $\mathcal{X}$. The reverse of this graph is obtained by reversing the directionality of its edges and is denoted as $\mathbf{G}'_d$.

Each matching is equivalent to a pairing between constraints and variables, so that each variable covered by the matching is solved by one equation of $\mathcal{C}$, ensuring that each equation will be used only once. A matching $\mathcal{M}$ onto $\mathbf{G}$ dictates the evaluations

$$x_{J(1)} = f_{I(1),J(1)}(\cdot) \tag{5a}$$
$$x_{J(2)} = f_{I(2),J(2)}(\cdot) \tag{5b}$$
$$\dots$$
$$x_{J(k)} = f_{I(k),J(k)}(\cdot) \tag{5c}$$

where $I(\cdot)$ and $J(\cdot)$ are enumerations on the covered constraints and variables respectively.

Matchings which are complete on $\mathcal{X}$ allow for the calculation of all the unknown variables, in a structural sense.

Since there is an $\mathcal{M}$ for $\mathbf{G}^+$ complete on $\mathcal{X}^+$, there are $|\mathcal{C}^+| - |\mathcal{X}^+|$ unmatched constraints on $|\mathcal{C}^+|$. Let the set of unmatched constraints on $\mathbf{G}^+$ be $\mathcal{C}_u^+$; this set is not unique, because $\mathcal{M}$ is not necessarily unique. At the same time, all of the variables of $\mathcal{X}^+$ are structurally calculable.

As a result, the values of the variables of any $c_u \in \mathcal{C}_u^+$ are known and c_u can be evaluated into a residual $r = f_{u,0}(\cdot)$. Assuming that the system operates on its nominal condition all the equations in $\mathcal{C}$ should hold and thus c_u should evaluate to 0. If c_u or a constraint which contributed to the evaluation of the related variables fails, then the residual should depart from 0. Thus, these unmatched constraints constitute *candidate ARRs*.

Any ARR c_u can detect faults occurring on itself as well as all the constraints which are reachable from c_u in $\mathbf{G}'_d$. This information is used to construct the detectability and isolability matrices, which characterize the diagnostic performance of the system.

C. MSOs

In order to minimize the fault candidates for each triggered residual, it is of interest to find residuals which are sensitive to as few faults as possible. The notion of Minimal Structurally Overdetermined sets (MSOs) is useful for that purpose [9]. An MSO is a set of equations $\mathcal{C}^* \subseteq \mathcal{C}$ whose corresponding graph $\mathbf{G}^* = \{\mathcal{C}^*, \mathcal{X}^*, \mathcal{E}^*\}$ has the following properties:

1) $\mathcal{X}^* = var(\mathcal{C}^*)$
2) $\mathcal{E}^* = \{(c_i, x_k) \in \mathcal{E} : c_i \in \mathcal{C}^*, x_k \in \mathcal{X}^*\}$
3) $\mathbf{G}^* = \mathbf{G}^{*+}$
4) $|\mathcal{C}^*| = |\mathcal{X}^*| + 1$

From each MSO $|\mathcal{C}^*|$ different residual generators can be extracted, one for each $c_i \in \mathcal{C}^*$. The rest of the equations in $\mathcal{C}^*$ form a just-constrained system for which a perfect matching is sought [9].

In order to reduce the number of candidate MSOs, it is beneficial to consider only MSOs with at least one equation that can fail. A residual generator which involves only constraints which cannot fail is not useful and clutters the residual selection procedure.

III. DIFFERENTIAL-ALGEBRAIC EQUATION SYSTEMS

Given a just-constrained system $\mathbf{G}$ and a matching onto it $\mathcal{M}$, one can draw conclusions on how hard it is to solve by first partitioning it into its König-Hall components $\{\mathbf{G}_1, \mathbf{G}_2, ..., \mathbf{G}_i\}$. This is also known as the *fine Dulmage-Mendelsohn* decomposition [16]. The corresponding matchings are $\{\mathcal{M}_i\}$. If all $\mathbf{G}_i$ are size-1, then the system is triangular and can be solved trivially by back-substitution.

If some $\mathbf{G}_i$ are of larger size, then the system is block-triangular: simultaneous equation systems must be solved for the evaluation of the corresponding residual generator. For each $\mathbf{G}_i$ two cases exist:

1) $\nexists d_j \in \mathbf{G}_i \Rightarrow$ An algebraic equation system must be solved. We assume the most general case of a non-linear algebraic system and will not examine further this case; non-linear system solvers are usually available in all modern computing systems.
2) $\exists d_j \in \mathbf{G}_i \Rightarrow \mathbf{G}_i$ represents a DAE at the most general case.

A time-invariant DAE system is a set of differential and algebraic equations. There are many formulations of a DAE, but the most useful to our analysis is the semi-explicit one:

$$\mathbf{c}_d(\dot{\mathbf{x}}_d, \mathbf{x}_d, \mathbf{x}_a) = \mathbf{0} \tag{6a}$$

$$\mathbf{c}_a(\mathbf{x}_d, \mathbf{x}_a) = \mathbf{0} \tag{6b}$$

where $\partial\mathbf{c}_d/\partial\dot{\mathbf{x}}_d$ is non-singular. $\mathbf{x}_d$ is the vector of the dynamic variables with n_d elements (similar to an ODE formulation) and $\mathbf{x}_a$ is the vector of the algebraic variables with n_a elements. (6a) capture the dynamics of the system while (6b) impose additional algebraic constraints [17].

The *differential index* of a DAE is the number of differentiations all or a subset of equations need to undergo, in order to convert the DAE into an ODE. The difficulty of the solution of the DAE depends a lot on how high its index is. DAEs with index 0 and 1 are considered much easier to solve than DAEs with higher indices and for that reason, DAEs with index 2 and above are called *high-index DAEs*. High-index problems are hard to solve with generic solvers, because the solution accuracy may be low, regardless of the size of the time step O(1), or even have an inverse relation with the step size O(1/h). In these cases, specialized, per-problem solvers may need to be designed [17].

We shall now investigate the structure of the DAEs underlying each $\mathbf{G}_i$, as it has an impact on its solvability.

A. Producing Semi-Explicit DAE Systems

The characteristics of the matching associated with a DAE are decisive to its solvability. As a first step, it must be verified that all $d_j \in \mathbf{G}_i$ are matched for their non-differentiated variable, since dynamic systems are solved by integration, not differentiation, i.e. $\mathcal{M}_i$ obeys *integral causality* [8],[11]. Any matching violating this check must be discarded and another must be sought, in order to solve this system.

Afterwards, all explicit differentiation constraints d_i are removed from $\mathcal{M}_i$. They are implied in the following analysis and do not need to be taken into account explicitly. The resulting DAE (previously presented in (5)) is of the, most general, form:

$$\dot{\mathbf{x}}_d = \mathbf{f}_d(\mathbf{x}_d, \mathbf{x}_a, \dot{\mathbf{x}}_d) \tag{7a}$$

$$\mathbf{x}_a = \mathbf{f}_a(\mathbf{x}_d, \mathbf{x}_a, \dot{\mathbf{x}}_d) \tag{7b}$$

This is not a standard DAE structure and no straightforward conclusions can be made on its solvability properties. In the interest of ensuring the solvability of any system associated with $\mathcal{M}_i$, no such DAE should result from the matching algorithm. Instead, we should aim for a semi-explicit formulation.

To ensure that all DAEs resulting from $\mathcal{M}_i$ will be semi-explicit, the differential part (7a) is treated first.

Proposition 1: Constraints in $\mathbf{G}$ which include more than one differentiated variable and can be solved for one or more differentiated variables shall be removed from the model. Mathematically, the set of these equations can be defined as:

$$\Big\{ c_d :\in \mathcal{C} : |var_{\dot{d}}(c_d)| > 1 \wedge |var_{s,\dot{d}}(c_d)| \geq 1 \Big\} \tag{8}$$

where

$$var_{\dot{d}}(c_i) = \{x_j \in var(c_j) : x_j \in \mathcal{X}_D\} \tag{9}$$

$$var_{s,\dot{d}}(c_i) = \{x_j \in var_s(c_i) : x_j \in \mathcal{X}_D\} \tag{10}$$

The second part of the logical expression (8) inhibits unnecessary equation removals; if an equation cannot be solved for any differentiated variable, then it can never be included as a differential equation in any DAE.

Removing equations results in loss of detectability performance. A more conservative approach is to substitute in the offending equation as many derivatives as possible with another analytic expression, until it contains only one differentiated variable, which can be solved for, e.g. for the equation of the course angle ([14], see also Section IV):

$$\begin{aligned}&\chi = atan2(\dot{e}, \dot{n}) \Rightarrow \\ &\chi = atan2\left(\dot{e}, (c_\theta c_\psi)u + (-c_\phi s_\psi + s_\phi s_\theta c_\psi)v\right. \\ &\qquad \left. +(s_\phi s_\psi + c_\phi s_\theta c_\psi)w\right)\end{aligned} \tag{11}$$

This reduces the isolability performance of the overall FD system and the final decision lies with the system designer.

So far, it is certain that each resulting DAE system will have the structure

$$\dot{\mathbf{x}}_d = \mathbf{f}_d(\mathbf{x}_d, \mathbf{x}_a) \tag{12a}$$

$$\mathbf{x}_a = \mathbf{f}_a(\mathbf{x}_d, \mathbf{x}_a, \dot{\mathbf{x}}_d) \tag{12b}$$

This is also non-standard, due to the $\dot{\mathbf{x}}_d$ terms in $\mathbf{f}_a$. Converting this system into a semi-explicit DAE system would require non-linear symbolic manipulation. Notably, contemporary academic software on Fault Diagnosis [18] cannot handle this difficulty.

Example 2: The 1-D velocity of a point mass is stabilized by a PD controller, generating the driving force F:

$$\dot{v} = F/m \tag{13a}$$

$$F = -k_P v - k_D \dot{v} \tag{13b}$$

(13) is not a semi-explicit DAE, although it is index-1, as it will be shown:

$$\begin{aligned}(13) \Rightarrow &\begin{cases}\dot{v} &= (-k_P v - k_D \dot{v})/m \\ F &= -k_P v - k_D \dot{v}\end{cases} \\ \Rightarrow &\begin{cases}\dot{v} &= -k_P/(m + k_D)v \\ F &= -k_P v + k_D k_P/(m + k_D)v\end{cases} \\ \overset{d/dt}{\Rightarrow} &\begin{cases}\dot{v} &= -k_P/(m + k_D)v \\ \dot{F} &= k_P^2 m/(m + k_D)^2 v\end{cases}\end{aligned} \tag{14}$$

which is an ODE. □

Even though this example is trivial to convert into a solvable ODE, it demonstrates that even the simplest systems might pose problems to an automated FDI system which isn't capable of symbolic manipulations. Indicatively, MATLAB's symbolic solver (`solve`) cannot solve for symbolic *functions*, in presence of their derivative (v, as a function of time, in this case).

Proposition 2: In the system $\mathbf{G}$, no differentiated variable may appear more than twice, of which one is reserved for the explicit differentiation equation, i.e. the system must have the property.

$$|eqs(x_{di})| \leq 2, \quad i = 1, 2, ..., n_d \tag{15}$$

To accomplish this, one can modify the mathematical model of the system, so that each derivative appears only once throughout the entire model, by substituting $\dot{x}_{di}$ with another expression where needed. In Example 2, the system can be modified to

$$\dot{v} = a \tag{16a}$$

$$a = F/m \tag{16b}$$

$$F = -k_P v - k_D a \tag{16c}$$

which is an index-1 semi-explicit DAE.

Theorem 1: Provided a system complies with Propositions 1 and 2, any DAE resulting from a causal and realizable matching is a semi-explicit DAE with structural index 1.

Proof: Because of Propositions 1 and 2, the DAE has the semi-explicit structure

$$\dot{\mathbf{x}}_d = \mathbf{f}_d(\mathbf{x}_d, \mathbf{x}_a) \tag{17a}$$

$$\mathbf{x}_a = \mathbf{f}_a(\mathbf{x}_d, \mathbf{x}_a) \tag{17b}$$

(17) has structural index 1 iff the Jacobian $\mathbb{I} - \partial\mathbf{f}_a/\partial\mathbf{x}_a$ has full structural rank [12]. In the most general case, and because of Assumption 2, the Jacobian has the structure

$$\mathbb{I} - \frac{\partial \mathbf{f}_a}{\partial \mathbf{x}_a} = \begin{bmatrix} 1 & & & X \\ & 1 & & \\ & X & \ddots & \\ & & & 1 \end{bmatrix} \tag{18}$$

which by definition of its structural pattern [12] results to a full structural rank (X represents potentially non zero elements). Thus, the DAE system has structural index 1. Moreover, the Jacobian is well-defined because of Assumption 1. ■

Thanks to Theorem 1, evaluating the structural rank of each DAE is unneeded, which is an $O(n^3)$ problem [12]; any residual generator resulting from SA will be at the hardest a semi-explicit DAE with structural rank 1.

B. Forms of Resulting DAEs

Let us now examine two significant cases, regarding the above result.

Case 1: The DAE system is of the form

$$\dot{\mathbf{x}}_d = \mathbf{f}_d(\mathbf{x}_a, \mathbf{x}_d) \tag{19a}$$

$$x_{a1} = f_{a1}(x_{a2}, x_{a3}, \ldots, x_{an}, \mathbf{x}_d) \tag{19b}$$

$$x_{a2} = f_{a2}(x_{a3}, x_{a4}, \ldots, x_{an}, \mathbf{x}_d) \tag{19c}$$

$$\vdots$$

$$x_{ai} = f_{ai}(x_{a(i+1)}, x_{a(i+2)}, \ldots, x_{an}, \mathbf{x}_d) \tag{19d}$$

$$\vdots$$

$$x_{an} = f_{an}(\mathbf{x}_d) \tag{19e}$$

with none of the arguments of each f_{ai} mandatory.

In this case, the evaluation of x_{ai} depends only on the values of algebraic variables with higher i-index, which results in a pure back-substitution chain. No algebraic loop (algebraic equation system) needs to be solved and the system is guaranteed to be solvable, as a triangular matrix with non-zero diagonal elements. The corresponding Jacobian is:

$$\mathbb{I} - \frac{\partial \mathbf{f}_a}{\partial \mathbf{x}_a} = \begin{bmatrix} \frac{\partial x_{a1}}{\partial x_{a1}} & -\frac{\partial f_{a1}}{\partial x_{a2}} & \cdots & -\frac{\partial f_{a1}}{\partial x_{an}} \\ \vdots & \ddots & & \\ -\frac{\partial f_{an}}{\partial x_{a1}} & -\frac{\partial f_{an}}{\partial x_{a2}} & \cdots & \frac{\partial x_{an}}{\partial x_{an}} \end{bmatrix} = \begin{bmatrix} 1 & & X & \\ & 1 & & \\ & & \ddots & \\ 0 & & & 1 \end{bmatrix} \tag{20}$$

This form can be directly implemented by an automated computing system only using function evaluations, as dictated by the evaluation chain. The complexity of the calculation of each residual is $O(2n_d + n_a)$.

Case 2: The second, most general case has the structure of (18), i.e., the algebraic part is not a triangular system of equations.

As has already been mentioned, the analytic solution for $\mathbf{x}_a$ is very hard to carry out with symbolic computations in an embedded system, or even impossible altogether, since $\mathbf{f}_a$ are non-linear in the general case. For that reason, $\mathbf{x}_a$ is commonly evaluated numerically, as the result of an optimization problem.

The optimization problem is solved in recursive steps, usually employing variable-step methods. It is not considered a computationally hard problem, especially if the initialization point is close to the solution, which can be achieved with high sampling rates. Still, it is the source of a non-deterministic delay which needs to be taken into account.

However, for the numerical solution to be possible, $\mathbb{I} - \partial \mathbf{f}_a / \partial \mathbf{x}_a$ must be *numerically* non-singular. This is equivalent to requiring that the numerical rank be the same as the structural rank, which may not always hold, even though it is the common case.

Unfortunately, it is not possible to identify the singularity of that Jacobian before the evaluation of the residual generator. Calculating the finite-difference derivative $\partial \mathbf{f}_a / \partial \mathbf{x}_a$ to verify the solvability of the algebraic loop would require a linearization point for $\mathbf{x}_a$, whose value is not available a-priori solving the system.

Reasonable initialization values could be useful in this case. Still, fallback routines in case the numerical solver halts with singularity errors should be implemented.

However, a significant outcome when the semi-explicit DAE does have a constant differential index, is that the initial conditions do not have to be consistent, because the dynamic degrees of freedom equal the number of dynamic variables [12]. Arbitrary initialization values can be used to begin the residual evaluation.

IV. UAV MODEL

As our research interests include FD in fixed-wing UAVs, in this section, a large-scale mathematical model of a fixed-wing, electric propulsion UAV is given, which is set up so as to comply with Propositions 1 and 2. Thus, any DAE resulting from SA will be semi-explicit and have structural index 1.

TABLE I: Proposed Fixed-Wing UAV Model - Kinematics

Label	Constraint	$\mathbf{var_s}$
c_1	$\dot{n} = (c_\theta c_\psi)u + (-c_\phi s_\psi + s_\phi s_\theta c_\psi)v$	$\dot{n}$
c_2	$\dot{e} = (c_\theta s_\psi)u + (c_\phi c_\psi + s_\phi s_\theta s_\psi)v$	$\dot{e}$
c_3	$\dot{d} = (-s_\theta)u + (s_\phi c_\theta)v + (c_\phi c_\theta)w$	$\dot{d}$
c_4	$\dot{\phi} = p + \tan\theta s_\phi q + \tan\theta c_\phi r$	$\dot{\phi}, p$
c_5	$\dot{\theta} = c_\phi q - s_\phi r$	$\dot{\theta}, q$
c_6	$\dot{\psi} = s_\phi / c_\theta q + c_\phi / c_\theta r$	$\dot{\psi}, r$
c_7	$V_i = \sqrt{u^2 + v^2 + w^2}$	V_i
c_8	$\chi = \mathrm{atan2}\left((c_\theta s_\psi)u + (c_\phi c_\psi + s_\phi s_\theta s_\psi)v, (c_\theta c_\psi)u + (-c_\phi s_\psi + s_\phi s_\theta c_\psi)v\right)$	χ
c_9	$\gamma = \sin^{-1}\left(((-s_\theta)u + (s_\phi c_\theta)v + (c_\phi c_\theta)w) / V_i\right)$	γ, V_i
c_{10}	$V_g = V_i c_\gamma$	V_g, V_i, γ
c_{11}	$u_r = u - u_w$	u_r, u, u_w
c_{12}	$v_r = v - v_w$	v_r, v, v_w
c_{13}	$w_r = w - w_w$	w_r, w, w_w
c_{14}	$\alpha = \mathrm{atan2}\,(w_r, u_r)$	α, u_r, w_r
c_{15}	$\beta = \sin^{-1}(v_r / V_a)$	v_r, V_a
c_{16}	$V_a = \sqrt{u_r^2 + v_r^2 + w_r^2}$	V_a

TABLE II: Proposed Fixed-Wing UAV Model - Inertial

Label	Constraint	$\mathbf{var_s}$
c_{17}	$m = m_0 + m_e$	m, m_0, m_e
c_{18}	$p_{CM,x} = (p_{m_e,x} m_e) / m$	$p_{CM,x}$
c_{19}	$p_{CM,y} = (p_{m_e,y} m_e) / m$	$p_{CM,y}$
c_{20}	$p_{CM,z} = (p_{m_e,z} m_e) / m$	$p_{CM,z}$
$c_{21}-c_{23}$	$j_{xx} = j_{0,xx} + (p^2_{m_e,y} + pz^2_{m_e,z}) \cdot \left(\frac{2m_e^2 + m_0 m_e}{m_0 + m_e}\right)$ etc.	$j_{xx}, j_{0,xx}$ etc.
$c_{24}-c_{29}$	$j_{xy} = j_{0,xy} - p_{m_e,x} p_{m_e,y} \left(\frac{m_0 m_e}{m_0 + m_e}\right)$ etc.	$j_{xy}, j_{0,xy}$ etc.
$c_{30}-c_{38}$	$j^I_{ij} = \mathbf{J}^{-1}_{ij}$	j^I_{ij}

It draws from first-principles modeling and common literature models. Its high level of detail is meant to make it versatile and suitable for a wide range of systems and configurations. The complete list of model constraints is enumerated in Tables I through X, broken down into individual components. The first column contains the constraint label, the second the constraint itself while the last one is filled with the set $var_s(c_i)$. The sine and cosine functions are abbreviated as $s.$ and $c.$ respectively.

The kinematic equations (Table I) are common in the literature [14]. Table II refers to the inertial model, which provides for an additional mass m_e in the position $\mathbf{p}_{m_e}$ [19]. Rigid-body dynamics (Table III) are supplemented with typical linear-coefficient aerodynamics modeling (Table V) [14]. Propulsion is represented by polynomial propeller performance curves (Table VI) [20] and an electric motor model (Table VII).

Standard models for Earth curvature (VIII) [21], atmosphere (IX) [22] and wind (X) [14] complete the model.

From the initial set of equations, only a few needed processing to comply with Propositions 1 and 2. More specifically: the relation between inertial velocity and inertial position derivatives was removed and the ground course and flight path angle relations were re-stated to avoid inertial position derivatives.

V. IMPLEMENTATION & SIMULATION

To demonstrate the performance of the proposed model, isolability analysis and real time simulations were performed.

TABLE III: Proposed Fixed-Wing UAV Model - Dynamics

Label	Constraint	$\mathbf{var_s}$
c_{39}	$c_1 = q\left(j_{zx}p + j_{zy}q + j_{zz}r\right)$ $-r\left(j_{yx}p + j_{yy}q + j_{yz}r\right)$	c_1
c_{40}	$c_2 = r\left(j_{xx}p + j_{xy}q + j_{xz}r\right)$ $-p\left(j_{zx}p + j_{zy}q + j_{zz}r\right)$	c_2
c_{41}	$c_3 = p\left(j_{yx}p + j_{yy}q + j_{yz}r\right)$ $-q\left(j_{xx}p + j_{xy}q + j_{xz}r\right)$	c_3
c_{42}	$\dot{p} = j_{11}^I\left(T_x - c_1\right)$ $+j_{12}^I\left(T_y - c_2\right) + j_{13}^I\left(T_z - c_3\right)$	$\dot{p}$
c_{43}	$\dot{q} = j_{21}^I\left(T_x - c_1\right)$ $+j_{22}^I\left(T_y - c_2\right) + j_{23}^I\left(T_z - c_3\right)$	$\dot{q}$
c_{44}	$\dot{r} = j_{31}^I\left(T_x - c_1\right)$ $+j_{32}^I\left(T_y - c_2\right) + j_{33}^I\left(T_z - c_3\right)$	$\dot{r}$
c_{45}	$\dot{u} = rv - qw + F_x/m$	$\dot{u}, F_x$
c_{46}	$\dot{v} = -ru + pw + F_y/m$	$\dot{v}, r, F_y$
c_{47}	$\dot{w} = qu - pv + F_z/m$	$\dot{w}, q, F_z$
c_{48}	$F_x = F_{g,x} + F_{a,x} + F_{p,x}$	$F_x, F_{g,x}, F_{a,x}, F_{p,x}$
c_{49}	$F_y = F_{g,y} + F_{a,y} + F_{p,y}$	$F_y, F_{g,y}, F_{a,y}, F_{p,y}$
c_{50}	$F_z = F_{g,z} + F_{a,z} + F_{p,z}$	$F_z, F_{g,z}, F_{a,z}, F_{p,z}$
c_{51}	$T_x = T_{a,t,x} + T_{p,t,x}$	$T_x, T_{a,t,x}, T_{p,t,x}$
c_{52}	$T_y = T_{a,t,y} + T_{p,t,y}$	$T_y, T_{a,t,y}, T_{p,t,y}$
c_{53}	$T_z = T_{a,t,z} + T_{p,t,z}$	$T_y, T_{a,t,y}, T_{p,t,y}$

TABLE IV: Proposed Fixed-Wing UAV Model - Gravity

Label	Constraint	$\mathbf{var_s}$
c_{54}	$F_{gx} = -s_\theta\ mg$	F_{gx}, θ
c_{55}	$F_{gy} = s_\phi c_\theta\ mg$	F_{gx}, ϕ
c_{56}	$F_{gz} = c_\phi c_\theta\ mg$	$F_{gz}, \phi, \theta, m, g$

TABLE V: Proposed Fixed-Wing UAV Model - Aerodynamics

Label	Constraint	$\mathbf{var_s}$
c_{57}	$F_{ax} = -c_\alpha F_D - c_\alpha s_\beta F_Y + s_\alpha F_L$	F_{ax}, F_D
c_{58}	$F_{ay} = -s_\beta F_D + c_\beta F_Y$	F_{ay}, F_Y
c_{59}	$F_{az} = -s_\alpha c_\beta F_D - s_\alpha s_\beta F_Y - c_\alpha F_L$	F_{az}, F_L
$c_{60}-c_{62}$	$dx_{CL} = p_{CL,x} - p_{CM,x}$ etc.	$dx_{CL}, p_{CL,x}, p_{CM,x}$
c_{63}	$T_{ax,t} = T_{ax} - dz_{CL}F_{ay} + dy_{CL}F_{az}$	$T_{ax,t}, T_{ax}$
c_{64}	$T_{ay,t} = T_{ay} + dz_{CL}F_{ax} - dx_{CL}F_{az}$	$T_{ay,t}, T_{ay}, dz_{CL}$
c_{65}	$T_{az,t} = T_{az} - dy_{CL}F_{ax} + dx_{CL}F_{ay}$	$T_{az,t}, T_{az}, dy_{CL}$
c_{66}	$\bar{q} = 0.5\rho V_a^2$	$\bar{q}, \rho, V_a$
c_{67}	$F_D = \bar{q}SC_D$	F_D, C_D
c_{68}	$F_Y = \bar{q}SC_Y$	F_Y, C_Y
c_{69}	$F_L = \bar{q}SC_L$	F_Z, C_Z
c_{70}	$C_D = C_{D,0} + C_{D,\alpha}\alpha + C_{D,q}\frac{c}{2V_a}q$ $+C_{D,\delta_e}\delta_e$	C_D
c_{71}	$C_Y = C_{Y,0} + C_{Y,\beta}\beta + C_{Y,p}\frac{b}{2V_a}p$ $+C_{Y,r}\frac{b}{2V_a}r + C_{Y,\delta_a}\delta_a + C_{Y,\delta_r}\delta_r$	C_Y
c_{72}	$C_L = C_{L,0} + C_{L,\alpha}\alpha + C_{L,q}\frac{c}{2V_a}q$ $+C_{L,\delta_e}\delta_e$	C_L
c_{73}	$T_{ax} = \bar{q}SbC_l$	T_{ax}, C_l
c_{74}	$T_{ay} = \bar{q}ScC_m$	T_{ay}, C_m
c_{75}	$T_{az} = \bar{q}SbC_n$	T_{az}, C_n
c_{76}	$C_l = C_{l,0} + C_{l,\beta}\beta + C_{l,p}\frac{b}{2V_a}p$ $+C_{l,r}\frac{b}{2V_a}r + C_{l,\delta_a}\delta_a + C_{l,\delta_r}\delta_r$	C_l
c_{77}	$C_m = C_{m,0} + C_{m,\alpha}\alpha + C_{m,q}\frac{c}{2V_a}q$ $+C_{m,\delta_e}\delta_e$	C_m
c_{78}	$C_n = C_{n,0} + C_{n,\beta}\beta + C_{n,p}\frac{b}{2V_a}p$ $+C_{n,r}\frac{b}{2V_a}rC_{n,\delta_a}\delta_a + C_{n,\delta_r}\delta_r$	C_n

TABLE VI: Proposed Fixed-Wing UAV Model - Propulsion

Label	Constraint	$\mathbf{var_s}$
c_{79}	$F_{px} = C_t\rho n_p^2 D^4$	F_{px}, C_t
c_{80}	$F_{py} = 0$	F_{py}
c_{81}	$F_{pz} = 0$	F_{pz}
c_{82}	$T_{px} = P_p/\omega_p$	T_{px}, P_p
c_{83}	$T_{py} = 0$	F_{py}
c_{84}	$T_{pz} = 0$	F_{pz}
$c_{85}-c_{87}$	$dx_p = p_{p,x} - p_{CM,x}$ etc.	$dx_p, p_{p,x}, p_{p,x}$
c_{88}	$T_{px,t} = T_{px} - dz_pF_{py} + dy_pF_{pz}$	$T_{px,t}, T_{px}$
c_{89}	$T_{py,t} = T_{py} + dz_pF_{px} - dx_pF_{pz}$	$T_{py,t}, T_{py}$
c_{90}	$T_{pz,t} = T_{pz} - dy_pF_{px} + dx_pF_{py}$	$T_{pz,t}, T_{pz}$
c_{91}	$n_p = \omega_p/(2\pi)$	n_p, ω_p
c_{92}	$J_a = V_a/(n_pD)$	J_a, V_a, n_p
c_{93}	$C_t = C_t(J_a)$	C_t
c_{94}	$P_p = C_p\rho n_p^3 D^5$	P_p, C_p
c_{95}	$C_p = C_p(J_a)$	C_p

TABLE VII: Proposed Fixed-Wing UAV Model - Motor

Label	Constraint	$\mathbf{var_s}$
c_{96}	$\dot{\omega}_p = (P_{mot} - P_p)/(\omega_p(J_p + J_{mot}))$	$\dot{n}_p, P_{mot}, P_p$
c_{97}	$\omega_p = \omega_{mot}$	n_p, n_{mot}
c_{98}	$2\pi\omega_{mot} = K_vE_i$	n_{mot}, E_i
c_{99}	$E_i = V_{mot} - I_{mot}R_m$	E_i, V_{mot}, I_{mot}
c_{100}	$P_{mot} = E_iI_i$	P_{mot}
c_{101}	$I_i = I_{mot} - I_0 - E_i/R_1$	I_i, I_{mot}, E_i
c_{102}	$P_{elec} = V_{mot}I_{mot}$	P_{elec}
c_{103}	$V_{mot} = (V_{bat} - I_{mot}(R_{bat} + R_S))\delta_t$	V_{mot}

A. Adding Sensors and Parameters

The model presented in the previous section does not contain any sensors purposely. For each aircraft and application the sensor suit may vary and so does the available knowledge of the model parameters.

For this analysis, it is presumed that accelerometer, gyroscope, AHRS, GPS, barometer, thermometer, Pitot probe, wind vanes, voltage, current and motor RPS sensor readings are available. The corresponding equations are added in Table XI ($s_1 - s_{22}$).

The control input information is inserted with constraints $c_{117} - c_{120}$, the initial mass m_0 is considered known (c_{121}), no additional mass is placed ($c_{125} - c_{128}$) and the value of gravity acceleration is set (c_{129}).

Known, fixed values are applied for the model parameters $\mathbf{p}_p, \mathbf{p}_{CL}, \mathbf{J}_{nom}, S, b, c, C_{D,*}, C_{Y,*}, C_{L,*}, C_{l,*}, C_{m,*}, C_{n,*}, D, J_{mot}, J_p, R_m, R_1, R_{bat}, R_s, I_0, L_0, M_0, R^*$.

The set of equations which were selected to be susceptible to faults was $c_{67} - c_{78}, c_{79} - c_{81}, c_{83} - c_{84}, c_{93} - c_{95}, c_{96} - c_{99}, c_{101}, c_{103}, s_1 - c_{22}, c_{117} - c_{128}$. No fault modeling was taken into account nor it is required for this method.

The equivalent structural model of the proposed fixed-wing UAV model was encoded into a graph representation,

TABLE VIII: Proposed Fixed-Wing UAV Model - Earth

Label	Constraint	$\mathbf{var_s}$
c_{104}	$n = (R_M + z)s_{(lat - lat_0)}$	n, z, lat, lat_0
c_{105}	$e = (R_N + z)s_{(lon - lon_0)}$	e, z, lon, lon_0
c_{106}	$d = -(z - z_0)$	d, z, z_0

TABLE IX: Proposed Fixed-Wing UAV Model - Atmosphere

Label	Constraint	$\mathbf{var_s}$
c_{107}	$h = (r_0 \cdot z)/(r_0 + z)$	h
c_{108}	$z = (r_0 \cdot h)/(r_0 - h)$	z
c_{109}	$T = T_0 + L_0 \cdot (h - h_0)$	T, T_0
c_{110}	$P = P_0\left(T_0/T(h)\right)^{\left(\frac{g_0 \cdot M_0}{R^* \cdot L_0}\right)}$	P, P_0
c_{111}	$h = T_0/L_0\left(\left(P/P_0\right)^{\frac{g_0 \cdot M_0}{R^* \cdot L_0}} - 1\right) + h_0$	h, h_0
c_{112}	$\rho = (P \cdot M_0)/(R^* \cdot T)$	ρ, P, T
c_{113}	$P_t = P + 0.5\rho V_a^2$	P_t, P, ρ, V_a

TABLE X: Proposed Fixed-Wing UAV Model - Wind

Label	Constraint	$\mathbf{vars}$
c_{114}	$u_w = c_\theta c_\psi v_{w,n} + c_\theta s_\psi v_{w,e}$	u_w
c_{115}	$v_w = (s_\phi s_\theta c_\psi - c_\phi s_\psi) v_{w,n}$ $+ (s_\phi s_\theta s_\psi + c_\phi c_\psi) v_{w,e}$	v_w
c_{116}	$w_w = (c_\phi s_\theta c_\psi + s_\phi s_\psi) v_{w,n}$ $+ (c_\phi s_\theta s_\psi - s_\phi c_\psi) v_{w,e}$	w_w

TABLE XI: Measurement, Input and Parameter Constraints

Label	Constraint	Subsystem	$\mathbf{vars}$
s_1	$a_{m,x} = F_x/m + s_\theta g$	Accelerometer	F_x, θ
s_2	$a_{m,y} = F_y/m - s_\phi c_\theta g$		F_y, ϕ
s_3	$a_{m,z} = F_z/m - c_\phi c_\theta g$		F_z, ϕ, θ
s_4	$p_m = p$	Gyroscope	p
s_5	$q_m = q$		q
s_6	$r_m = r$		r
s_7	$\phi_m = \phi$	AHRS	ϕ
s_8	$\theta_m = \theta$		θ
s_9	$\psi_m = \psi$		ψ
s_{10}	$lat_{gps} = lat$	GPS	lat
s_{11}	$lon_{gps} = lon$		lon
s_{12}	$z_{gps} = z$		z
s_{13}	$V_{g,gps} = V_g$		V_g
s_{14}	$\chi_{gps} = \chi$		χ
s_{15}	$P_{bar} = P$	Barometer	P
s_{16}	$T_m = T$	Thermometer	T
s_{17}	$P_{t,m} = P_t$	Airspeed Sensor	P_t
s_{18}	$\alpha_m = \alpha$	Wind Vanes	α
s_{19}	$\beta_m = \beta$		β
s_{20}	$V_{mot,m} = V_{mot}$	Voltage Sensor	V_{mot}
s_{21}	$I_{mot,m} = I_{mot}$	Current Sensor	I_{mot}
s_{22}	$\omega_{m,m} = \omega_m$	RPM Sensor	ω_m
c_{117}	$\delta_a = \delta_{a,inp}$	System Inputs	δ_a
c_{118}	$\delta_e = \delta_{e,inp}$		δ_e
c_{119}	$\delta_t = \delta_{t,inp}$		δ_t
c_{120}	$\delta_r = \delta_{r,inp}$		δ_r
c_{121}	$m_0 = m_{0,known}$	Inertial	m_0
$c_{122}-c_{124}$	$\mathbf{J}_0 = \mathbf{J}_{0,known}$		$j_{0,ij}$
c_{125}	$m_e = 0$	Additional Mass	m_e
$c_{126}-c_{128}$	$\mathbf{p}_{m_e} = 0$		$\mathbf{p}_{m_e}$
c_{129}	$g = g_0$		g

suitable for parsing by computer programs. Software was written under the ROS framework [23], which implemented the Fault Diagnostic functionality in real-time.

B. Detectability and Isolability Performance

Most of the faults taken into account were detectable. Best-case isolability performance for the initial-nominal model is summarized in Figure 1. The only faults not covered are c_{103} and c_{119}.

By inspecting the fault isolability matrix, it can be concluded that almost all faults can be isolated within the component, with the exception of some cross-coupling between aerodynamics and the propeller model, which is expected.

C. Simulation

In order to evaluate the feasibility and performance of the extracted residuals, a simulation environment was constructed. A flight simulator was programmed, built on top of the Gazebo robot simulator [24]. Custom propulsion and aerodynamics model plugins were added to the UAV model. A screenshot of the simulation environment can be seen in Figure 2. The resulting mathematical model which was simulated was of greater detail than the one provided to the Fault Diagnosis software.

Among the residuals which were generated from the structural model, we choose to demonstrate one which incorporates a DAE system.

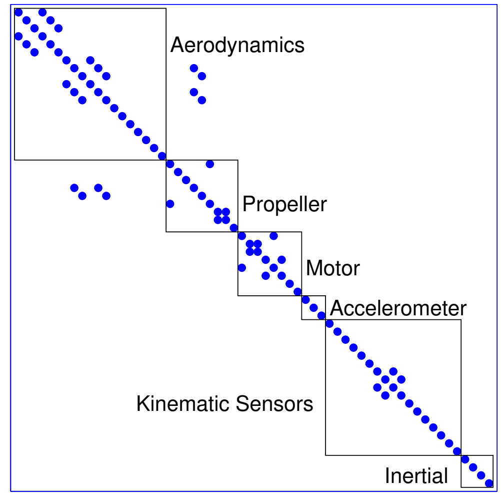

Fig. 1: Isolability Performance

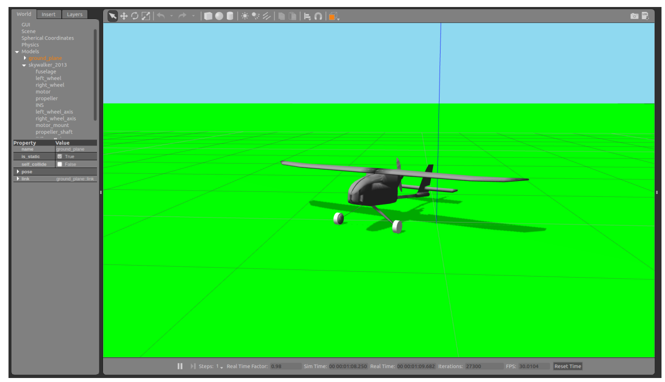

Fig. 2: A Screenshot of the Simulation Environment

It consists of the equations $c_{91}, c_{92}, c_{94}, c_{95}, c_{96}, c_{97}, c_{99} - c_{101}, c_{109}, c_{110}, c_{112}, c_{113}, s_{17}, s_{20}, s_{21}$ with s_{22} being used as the unmatched constraint. We can verify that the corresponding Jacobian is triangular with ones in each diagonal element. Hence the system is always solvable within the domain of the involved constraints with back-substitution. The calculations are omitted due to lack of space.

The measurements were corrupted with non-zero mean Gaussian white noise, with characteristics typical of each sensor. The model parameters were provided to the FDI system with a 10% margin of error. The initial value of the DAE dynamic variable was randomly selected from the operating envelope.

The response of the residual to a fault on the propeller, namely damage in one of its blades, is examined. The damage, and the corresponding loss of thrust and consumed power, are implemented as a change of the polynomial coefficients in constraints c_{95} and c_{93}. The fault is introduced during a $40\,\mathrm{s}$ flight segment at time $31.5\,\mathrm{s}$. The residual response can be seen in Figure 3.

After initialization of the residual generator, the residual reduces close to zero, as the calculated n_m state converges to its real value. Indeed, there is no consistency requirement for the initial condition. Even though the value of n_m is not constant during the pre-fault flight, the residual remains inside a bounded region of trust.

As soon as the propeller fault occurs, the residual gradu-

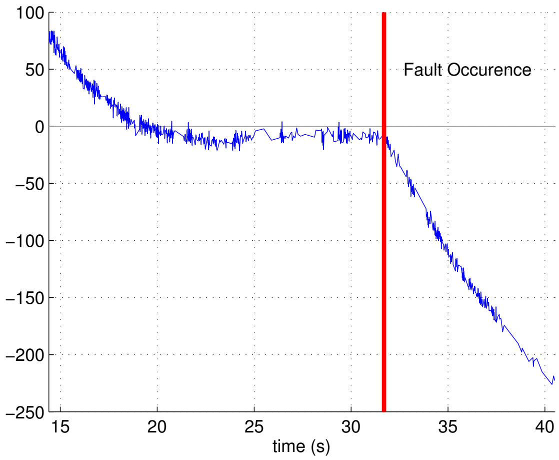

Fig. 3: Residual Response

ally but quickly builds up and exceeds detection thresholds within seconds.

VI. CONCLUSIONS

Structural Analysis is a formidable option for automated Fault Diagnosis on large-scale systems. In this work, the potential existence of Differential Algebraic Equation systems within residual generators, resulting from SA, on dynamic systems was discussed as well as the issue of their solvability restrictions, regarding their differential index.

Two conditions were given for the mathematical model of any dynamic system; if a model is adapted to satisfy them, it is guaranteed that it, or any submodel, contains only semi-explicit DAEs with structural index 1, which has significantly better solvability characteristics.

Focusing on fixed-wing UAVs, a large-scale mathematical model was given which complies to these conditions and SA was performed on it. A standard set of sensor and faults was added to the model and isolability analysis was carried out. A simulation presenting the performance of an example residual generator containing a DAE was carried out.

Future directions of this work include the investigation of more factors which affect the numerical evaluation of residuals and the experimental verification of our findings.

REFERENCES

[1] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki, *Diagnosis and fault-tolerant control*, 2nd ed. Springer Berlin Heidelberg, 2006.

[2] R. Patton, R. Clark, and P. Frank, *Issues of fault diagnosis for dynamic systems*. Springer, 2000.

[3] K. P. Valavanis and G. Vachtsevanos, *Handbook of Unmanned Aerial Vehicles*, K. P. Valavanis and G. J. Vachtsevanos, Eds. Springer Netherlands, 2015.

[4] G. Wild, J. Murray, and G. Baxter, "Exploring Civil Drone Accidents and Incidents to Help Prevent Potential Air Disasters," *Aerospace*, 2016.

[5] J. Marzat, H. Piet-Lahanier, F. Damongeot, and E. Walter, "Model-based fault diagnosis for aerospace systems: a survey," *Journal of Aerospace Engineering*, 2012.

[6] M. Fravolini, V. Brunori, G. Campa, M. Napolitano, and M. La Cava, "Structural Analysis Approach for the Generation of Structured Residuals for Aircraft FDI," *IEEE Transactions on Aerospace and Electronic Systems*, 2009.

[7] R. Izadi-Zamanabadi, "Structural analysis approach to fault diagnosis with application to fixed-wing aircraft motion," in *Proceedings of the 2002 American Control Conference*, 2002.

[8] V. Flaugergues, V. Cocquempot, M. Bayart, and M. Pengov, "On non-invertibilities for Structural Analysis," *21st International Workshop on Principles of Diagnosis*, 2010.

[9] M. Krysander and J. Aslund, "An Efficient Algorithm for Finding Over-constrained Sub-systems for Construction of Diagnostic Tests," in *16th International Workshop on Principles of Diagnosis)*, 2005.

[10] V. Flaugergues, V. Cocquempot, M. Bayart, and M. Pengov, "Structural Analysis for FDI: a modified, invertibility-based canonical decomposition," in *Proceedings of the 20th International Workshop on Principles of Diagnosis*, 2009.

[11] G. Zogopoulos Papaliakos and K. J. Kyriakopoulos, "On the selection of calculable residual generators for UAV fault diagnosis," in *24th Mediterranean Conference on Control and Automation (MED)*, 2016.

[12] J. Unger, A. Kröner, and W. Marquardt, "Structural analysis of differential-algebraic equation systemstheory and applications," *Computers & Chemical Engineering*, 1995.

[13] R. d. P. Soares and A. R. Secchi, "Structural analysis for static and dynamic models," *Mathematical and Computer Modelling*, vol. 55, no. 3-4, pp. 1051–1067, 2012.

[14] B. Stevens, F. Lewis, and E.N. Johnson, *Aircraft Control and Simulation*, 3rd ed. Wiley, 2016, no. 9.

[15] A. L. Dulmage and N. S. Mendelsohn, "Coverings of bipartite graphs," *Canadian Journal of Mathematics*, 1958.

[16] A. Pothen and C.-J. Fan, "Computing the block triangular form of a sparse matrix," *ACM Transactions on Mathematical Software*, vol. 16, no. 4, pp. 303–324, dec 1990.

[17] K. E. Brenan, S. L. V. Campbell, and L. R. Petzold, *Numerical solution of initial-value problems in differential-algebraic equations*, 1996.

[18] E. Frisk, "Fault Diagnosis Toolbox - A Matlab toolbox for fault diagnosis." [Online]. Available: http://www.fs.isy.liu.se/Software/FaultDiagnosisToolbox/

[19] J. Peraire and J. Widnall, "Lecture L26 - 3D Rigid Body Dynamics : The Inertia Tensor," in *Dynamics*, 2008.

[20] D. Allerton, *Principles of flight simulation*. Wiley, 2009.

[21] W. Mulaire, "Department of Defense: World Geodetic System 1984," National Imagery and Mapping Agency, Tech. Rep., 2000.

[22] "US Standard Atmosphere, 1976," National Oceanic and Atmospheric Administration, National Aeronautics and Space Adminisration, United States Air Force, Tech. Rep., 1976.

[23] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[24] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004.