Qualitative Event-based Diagnosis Applied to a Spacecraft Electrical Power Distribution System

Matthew J. Daigle[a,1], Indranil Roychoudhury[b,1], Anibal Bregon[c,2,*]

[a]*NASA Ames Research Center, Moffett Field, CA, 94035, USA*
[b]*SGT Inc., NASA Ames Research Center, Moffett Field, CA, 94035, USA*
[c]*Department of Computer Science, University of Valladolid, Valladolid, 47011, Spain*

Abstract

Quick, robust fault diagnosis is critical to ensuring safe operation of complex engineering systems. We develop a fault detection, isolation, and identification framework for three separate diagnosis algorithms: the first using global model; the second using minimal submodels, which allows the approach to scale easily; and the third using both the global model and minimal submodels, combining the strengths of the first two. The diagnosis framework is applied to the Advanced Diagnostics and Prognostics Testbed, that functionally represents spacecraft electrical power distribution systems. The practical implementation of these algorithms is described, and their diagnosis performance using real data is compared.

Keywords: Fault diagnosis, model-based diagnosis, structural model decomposition, electrical power systems, ADAPT

1. Introduction

Fault diagnosis plays an essential role in ensuring system safety in many application domains, from industrial power plants to aerospace vehicles. When a fault occurs in a system, diagnosis software must be able to quickly detect the presence of the fault, isolate the true fault among many potential fault candidates, and identify the fault magnitude [1–4]. With this information, automated mitigation and recovery actions can be taken. Proper recovery actions enable successful continued operation and prevention of catastrophic consequences, both of which lead to cost savings [5, 6].

In this paper, a model-based diagnosis approach for the Advanced Diagnostics and Prognostics Testbed (ADAPT), an electrical power distribution system that is representative of those on spacecrafts, is developed. ADAPT serves as a testbed through which faults can be injected to evaluate diagnostic and prognostic algorithms [7]. Located at NASA Ames Research Center, ADAPT has been established as a diagnostic benchmark system through the industrial track of the International Diagnostic Competition (DXC) [8, 9]. Within the DXC, specific diagnostic problems are defined for ADAPT, and competing algorithms are evaluated using real experimental data obtained from the ADAPT hardware. Diagnostic algorithms must deal with a variety of real-world issues in order to be successful. In particular, this paper is focused on diagnosing faults on a subset of ADAPT, called ADAPT-Lite. The application context is that of an unmanned aircraft system, and the diagnosis must be used to provide mission abort/continue commands [8]. In order to do this, faults must be correctly detected (i.e., determine if a fault is present in the system), isolated (i.e., determine which fault has occurred), and identified (i.e., estimate the parameters that define the fault behavior), under the single fault assumption. Although solutions in this work are specifically developed for ADAPT, the approach is model-based and therefore can be applied to different systems given suitable models.

[*]Corresponding author
Email addresses: `matthew.j.daigle@nasa.gov` (Matthew J. Daigle), `indranil.roychoudhury@nasa.gov` (Indranil Roychoudhury), `anibal@infor.uva.es` (Anibal Bregon)

[1]M. Daigle and I. Roychoudhury's work has been partially supported by the NASA System-wide Safety and Assurance Technologies (SSAT) project.
[2]A. Bregon's funding for this work has been provided by the Spanish MICINN DPI2013-45414-R grant.

The model-based diagnosis approach developed in this work is rooted in a qualitative fault isolation framework that is based on the analysis of residual signals, where residuals are computed as the difference between observed and predicted system variables [10]. Faults in the system are modeled as changes in the value of the system parameters [10] and as changes in component modes [11]. Faults cause discrepancies in observed behavior and model-predicted behavior, and thus manifest as deviations in the residual signals. Fault detection involves statistical testing of the residuals. The transients of residual deviations are abstracted qualitatively and compared to predicted fault transients to enable quick fault isolation. Both the qualitative change in the residual signal, expressed as + and − values in magnitude and slope, as well as the temporal ordering of these transients as they manifest in the residuals, are used as diagnostic information, establishing an event-based qualitative fault isolation framework [11].

Predicted values of system outputs are computed by using models of the system, which can be either a global model of the system or local submodels. Structural model decomposition methods can be used to systematically compute the submodels. The use of local submodels leads to increased scalability of the diagnosis algorithm [12, 13] and increased diagnosability. They been successfully used for fault diagnosis in industrial [14] and aerospace applications [15]. The main idea is to take advantage of the analytical redundancy provided by the sensors and the model to derive minimal submodels that provide additional information useful for diagnosis. In particular, this paper uses a structural model decomposition approach based upon Possible Conflicts (PCs) [16], which is a structural model decomposition technique equivalent to Analytical Redundancy Relations (ARRs) [14, 17]. PCs are computed off-line as the minimal subsets of the global model constraints that produce inconsistencies when faults occur. Residuals may be computed using PCs, and residual deviations analyzed following the qualitative fault isolation framework [10, 11, 18]. Then, quantitative fault identification can be carried out by using minimal local submodels for parameter estimation [18].

The contributions of this work are as follows. First, a novel model-based diagnosis framework is developed that addresses fault detection, isolation, and identification. It combines techniques from qualitative fault isolation and structural model decomposition. Specifically, structural model decomposition is used as an underlying technique to automatically determine the sets of submodels for each diagnosis task. From this framework, several diagnoser designs can be derived using different sets of models and submodels. We show that two previous algorithms, QED (Qualitative Event-based Diagnosis) and QED-PC (QED with Possible Conflicts) [19] can be formulated as specific instantiations of this framework, where QED uses a global system model, and QED-PC uses minimal local submodels. A new algorithm, QED-PC++, that uses both the global model and the minimal local submodels, is formulated, and it is shown how it combines the strengths of QED and QED-PC. The three algorithms are implemented as diagnostic solutions for the ADAPT case study, which includes the development of models of ADAPT suitable for diagnosis, the integration of heuristic fault isolation rules to improve fault isolation performance, and novel fault identification techniques. Using a large, comprehensive set of experimental data from the ADAPT hardware, the three algorithms are applied and their performance is compared. By analyzing the set of experimental results, we discover their limitations, and suggest possible future improvements and extensions.

The paper is organized as follows. Section 2 describes the ADAPT case study. Section 3 overviews the diagnosis approach. Section 4 provides the system model and describes the structural model decomposition approach. Section 5 describes fault detection and Section 6 describes symbol generation. Section 7 discusses fault isolation, and Section 8 describes fault identification. Section 9 covers decision-making. Section 10 presents the experimental results and discusses lessons learned. Section 11 describes related work, and Section 12 concludes the paper.

2. The Advanced Diagnostics and Prognostics Testbed

The Advanced Diagnostics and Prognostics Testbed is an electrical power distribution system that is representative of those on spacecraft, and has been established as a diagnostic benchmark system through the International Diagnostic competition [20]. As mentioned in the previous section, in this work, we focus on a subset of ADAPT, called ADAPT-Lite, which has been used to define Diagnostic Problem I of the industrial track of the DXC [8, 9], in which the ADAPT-Lite hardware is used to emulate the operation of an electrical power system aboard an Unmanned Aircraft System (UAS).

A system schematic for ADAPT-Lite is given in Fig. 1. A battery (BAT2) supplies electrical power to several loads, transmitted through several circuit breakers (CB236, CB262, CB266, and CB280), relays (EY244, EY260, EY281, EY272, and EY275), and an inverter (INV2) that converts dc to ac power. ADAPT-Lite has one dc load (DC485) and two ac loads (AC483 and FAN416). There are sensors throughout the system to report electrical voltage (names

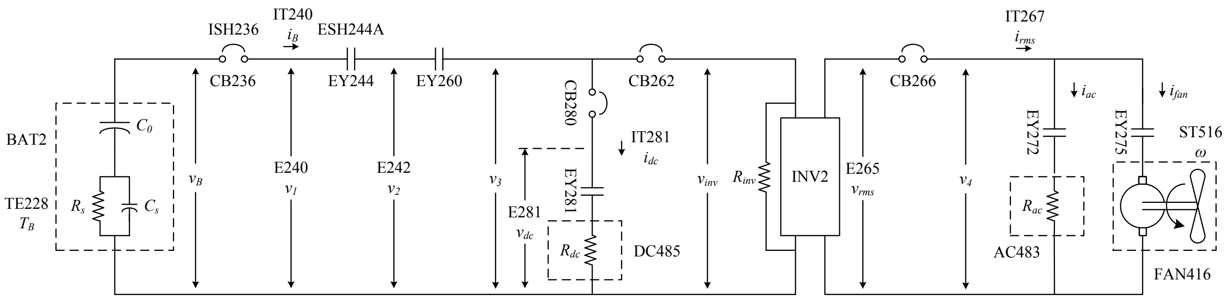


Figure 1: ADAPT-Lite schematic.

Table 1: Components of ADAPT-Lite, their failure modes, and abort recommendations by failure mode.

Components	Failure Mode	Recommendation
AC483, DC485	Failed Off	Abort
	Resistance Offset	Conditional
	Resistance Drift	Conditional
	Intermittent Resistance Offset	Conditional
CB236 CB262, CB266, CB280	Failed Open	Abort
E240, E242, E265, E281, TE228	Offset	None
	Stuck	None
	Drift	None
	Intermittent Offset	None
IT240, IT267, IT281, ST516	Offset	Conditional
	Stuck	Abort
	Drift	Conditional
	Intermittent Offset	Conditional
ESH244A, ISH236	Stuck	None
EY244, EY260, EY272, EY275, EY284	Stuck Open	Abort
FAN416	Underspeed	None
	Overspeed	Abort
	Failed Off	Abort
INV2	Failed Off	Abort

beginning with “E”), electrical current (“IT”), and the positions of relays and circuit breakers (“ESH” and “ISH”, respectively). There is one sensor to report the operating state of a load (fan speed, ST516) and another to report the battery temperature (TE228).

A diagnostic algorithm is used to inform the operator if faults have occurred, and if so, whether the fault requires aborting the mission and landing the UAS. Given the time-stamped vectors of system inputs $\mathbf{u}(t)$ and outputs $\mathbf{y}(t)$, the goal of the diagnosis algorithm is to detect the fault, isolate the faulty component and its fault mode, identify the fault magnitude, and then generate an *abort* command if necessary by $t = 4$ minutes into the mission. The necessity of an abort depends on the fault type, and, in some cases, on the fault magnitude, thus, this diagnostic problem requires the diagnostic algorithm to perform not only fault detection and isolation, but also fault identification.

Table 1 summarizes the abort recommendation for each fault mode in ADAPT-Lite. A command to abort should be given for any fault that results in a loss of power to the three loads, i.e., faults in any of the circuit breakers or relays, a failure in the inverter, and failures in the loads themselves. An overspeed fault of the fan results in an abort, but an underspeed fault does not. For a resistance change in the dc and ac loads, an offset (i.e., bias) fault triggers an abort if its magnitude is outside an acceptable range, a drift fault triggers an abort if the fault magnitude will be too large by the end of the mission, and an intermittent resistance offset fault triggers an abort if the average value of the resistance (over both nominal and faulty time periods) is outside an acceptable range. For the sensors deemed critical, IT260, IT267, IT281, and ST516, the mission must be aborted whenever the sensor is stuck, or if the offset, drift, or average intermittent value is outside an acceptable range by the end of the mission. For the remaining (noncritical) sensors, an abort is never necessary.

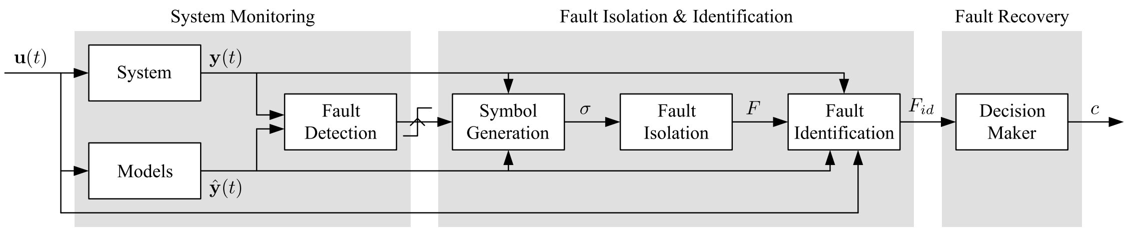


Figure 2: Diagnosis and recovery architecture.

3. Diagnosis Approach

The proposed diagnosis architecture is shown in Fig. 2. The system receives inputs $\mathbf{u}(t)$ (for ADAPT-Lite, circuit breaker and relay commands) and produces outputs $\mathbf{y}(t)$ (current measurements, voltage measurements, etc.). The set of system models (Section 4), consisting of a global model and/or a set of submodels, given inputs $\mathbf{u}(t)$, computes predicted values $\hat{\mathbf{y}}(t)$. The fault detection module (Section 5) compares $\mathbf{y}(t)$ and $\hat{\mathbf{y}}(t)$ and determines whether the differences indicate a fault. The symbol generation module (Section 6) transforms deviations from expected behavior into a symbolic representation, where a symbol is denoted by σ. The fault isolation module (Section 7) uses the sequence of these symbols to isolate faults F, by comparing to predicted symbol sequences for the faults. Each fault $f \in F$ is associated with a component, a fault mode, and a set of fault parameters. The fault identification module (Section 8) estimates, for each fault $f \in F$, the values of the fault parameters, which also helps to further isolate faults. The identified fault set is output as F_{id} and provided to the decision maker. The decision making module (Section 9) determines the command c to take (a possible *abort* command), based on the given diagnosis.

All the algorithms presented in this work (QED, QED-PC, and QED-PC++), which will be described in the following sections, are captured within this single diagnosis framework and implemented following this same general architecture. The main difference lies in the underlying models used for the diagnosis tasks: QED is based on a global system model of the system; QED-PC is based on minimal submodels computed from the global system model; and QED-PC++ is based on the combination of the global model and the minimal submodels. In the following sections, each piece of the diagnosis architecture is described in detail.

4. System Modeling

The proposed diagnosis approach is model-based, requiring a model describing both nominal and faulty behavior to be used for prediction of nominal values, and for fault detection, isolation, and identification. We seek to use the most parsimonious models to solve the diagnosis task. Because fault identification is required, the models must ultimately be quantitative.

In this work, the modeling framework described in [21] is used. Its main details are summarized here for completeness. We first define a *model*.

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

Input variables $u \in U$ are known and correspond in ADAPT-Lite to relay and circuit breaker commands (6 in total); and the output variables $y \in Y$ correspond to (measured) sensor signals (ADAPT-Lite has 11). Parameters $\theta \in \Theta$ include explicit model parameters that are used in the model constraints. Θ does not need to include all parameters in the equations, only those that must be included explicitly (i.e., fault parameters). The auxiliary variables A are extra variables that are functions of the states, and are used to simplify the model structure.

A constraint $c = (\varepsilon_c, V_c)$ is a tuple including an equation ε_c over a set of variables V_c. It does not impose any computational causality on the variables V_c, i.e., it does not specify which $v \in V_c$ is the dependent variable in equation ε_c. The notion of a *causal assignment* to define the dependent variable is required.

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

A causal assignment of a constraint is written by using the constraint's equation in a causal form, using := instead of = to make the causal direction explicit.

As described in [21], a set of causal assignments $\mathcal{A}$, for a model $\mathcal{M}$ is *valid* if (*i*) there are no causal constraints computing variables in U or Θ, (*ii*) no variable in Y acts as a dependent variable in any causal constraint, and (*iii*) every other variable has exactly one causal constraint computing it. A *causal model* is a model extended with a valid set of causal assignments.

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

We next describe the global model for ADAPT-Lite, followed by a description of the structural model decomposition methodology and the submodels derived for ADAPT-Lite from the global model.

4.1. Global Model

A system schematic for ADAPT-Lite is given in Fig. 1. BAT2 consists of two 12 V lead-acid batteries in series, which are lumped together into a single battery model. Battery models typically must include a set of complex nonlinear behaviors [22]. However, most of these nonlinear characteristics are not evident within the time frame of the experimental scenarios. Therefore, a simplified electrical circuit equivalent model, consisting of a single large capacitance, C_0, in series with a capacitor-resistor pair, C_s and R_s, that subtracts from the voltage provided by C_0 (see Fig. 1), is used. The battery may then be described as

$$\dot{v}_0 := \frac{1}{C_0}\left(-i_B\right), \tag{c_1}$$

$$v_0 := \int_{t_0}^{t} \dot{v}_0 dt, \tag{c_2}$$

$$\dot{v}_s := \frac{1}{C_s}\left(i_B R_s - v_s\right), \tag{c_3}$$

$$v_s := \int_{t_0}^{t} \dot{v}_s dt, \tag{c_4}$$

$$v_B := v_0 - v_s, \tag{c_5}$$

where i_B is the battery current, v_B is the battery voltage, v_0 is the voltage across C_0, and v_s is the voltage drop across C_s and R_s. In reality, R_s is a function of state of charge, depth of charge [22], and temperature, but, since it is not observed to change significantly in the available data, R_s is assumed to be constant. Instead, since the battery voltage decreases faster at lower voltages, C_0 is expressed as a function that decreases with voltage. In the experimental data, battery temperature, T_B, is not observed to change significantly and can be assumed to be constant, i.e.,

$$\dot{T}_B := 0, \tag{c_6}$$

$$T_B := \int_{t_0}^{t} \dot{T}_B dt. \tag{c_7}$$

The battery voltage v_B is supplied to the loads through a series of relays and circuit breakers. The intermediate voltages are described by:

$$v_1 := s_{CB236} \cdot v_B, \tag{c_8}$$

$$v_2 := s_{EY244} \cdot v_1, \tag{c_9}$$

$$v_3 := s_{EY260} \cdot v_2, \tag{c_{10}}$$

$$v_{dc} := s_{CB280} \cdot v_3, \tag{c_{11}}$$

$$v_{inv} := s_{CB262} \cdot v_3, \tag{c_{12}}$$

where the s variables are the on/off states of the relays and circuit breakers. These states are described by:

$$s_{CB236} := u_{CB236} \cdot f_{CB236}, \tag{c_{13}}$$

$$s_{EY244} := u_{EY244} \cdot f_{EY244}, \tag{c_{14}}$$

$$s_{EY260} := u_{EY260} \cdot f_{EY260}, \tag{c_{15}}$$

$$s_{CB262} := f_{CB262}, \tag{c_{16}}$$

$$s_{CB280} := f_{CB280}, \tag{c_{17}}$$

where the u variables are the input commands for those components, and the f variables are fault parameters used to capture the failed/stuck open fault modes. Note that CB262 and CB280 cannot be controlled and so have no corresponding input variables.

The battery current i_B is the sum of the dc current, i_{dc}, and the input dc current to the inverter, i_{inv}. This is described by:

$$i_B := i_{dc} + i_{inv}, \tag{c_{18}}$$

$$i_{dc} := s_{EY281} \cdot f_{dc} \cdot \frac{v_B}{R_{dc}}, \tag{c_{19}}$$

$$s_{EY281} := u_{EY281} \cdot f_{EY281}, \tag{c_{20}}$$

where R_{dc} is the resistance of the dc load, which may change due to a fault, and f_{dc} is a fault parameter used to model the "failed off" failure mode.

The inverter transforms dc power to ac power. When operating nominally, the rms voltage v_{rms} is controlled very close to 120 V ac as long as the input voltage is above 18 V:

$$v_{rms} := 120 \cdot (v_{inv} > 18) \cdot f_{INV2}, \tag{c_{21}}$$

where f_{INV2} is a fault parameter used to model the "failed off" failure mode. From a power balance of the ac and dc sides of the inverter, it results that $v_{inv} \cdot i_{inv} = e \cdot v_{rms} \cdot i_{rms}$, where e is the inverter efficiency, i_{rms} is the inverter rms current, v_{inv} is the inverter voltage on the dc side, and i_{inv} is the input dc current to the inverter. The inverter still draws a small amount of current even when $i_{rms} = 0$, and this is captured as a dc resistance parallel to the inverter, R_{inv}. Hence, the following equation is derived:

$$i_{inv} := \frac{v_{rms} \cdot i_{rms}}{e \cdot v_{inv}} + \frac{v_{inv}}{R_{inv}}. \tag{c_{22}}$$

The voltage seen by the ac loads is dependent on the state of CB266:

$$v_4 := s_{CB266} \cdot v_{rms}, \tag{c_{23}}$$

$$s_{CB266} := f_{CB266}. \tag{c_{24}}$$

The dc and ac resistive loads are modeled as pure resistances, with R_{dc} and R_{ac}, respectively. The fan has both resistive and inductive properties, so introduces a phase difference ϕ in its current from the input voltage. Since only rms values of the inverter voltage and current are available, only the steady-state ac relations are used. The total rms current drawn by the ac loads is then

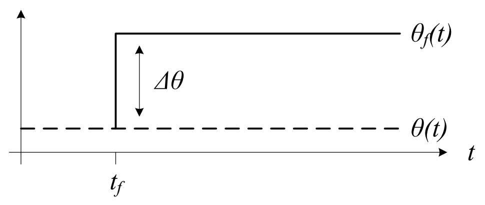
$$i_{rms} := \frac{1}{\sqrt{2}} \left| \sqrt{2} \cdot i_{fan} \cdot (\cos\phi + j \sin\phi) + \sqrt{2} \cdot i_{ac} \right|, \tag{c_{25}}$$

$$i_{ac} := s_{EY272} \cdot f_{ac} \cdot \frac{v_4}{R_{ac}}, \tag{c_{26}}$$

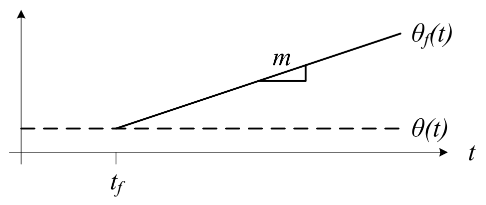
$$i_{fan} := s_{EY275} \cdot f_{fan} \cdot \frac{v_4}{R_{fan}}, \tag{c_{27}}$$

$$s_{EY272} := u_{EY272} \cdot f_{EY272}, \tag{c_{28}}$$

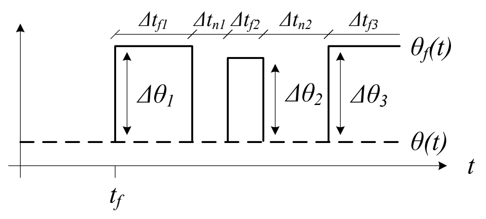
$$s_{EY275} := u_{EY275} \cdot f_{EY275}, \tag{c_{29}}$$



(a) Offset fault profile.



(b) Drift fault profile.



(c) Intermittent offset fault profile.

Figure 3: Fault profiles.

where R_{fan} is the magnitude of the fan impedance, and f_{ac} and f_{fan} are fault parameters used to model the "failed off" failure modes. The fan speed is expressed as a function of its input voltage

$$\dot{\omega} := \frac{1}{J_{fan}}\left(s_{EY275} f_{fan} \frac{v_4}{B_{fan} R_{fan}} - \omega\right), \tag{c_{30}}$$

$$\omega := \int_{t_0}^{t} \dot{\omega}\, dt, \tag{c_{31}}$$

where J_{fan} is an inertia parameter and B_{fan} is a resistance parameter.

The sensors can be modeled using the following constraints:

$$y_{E240} := b_{E240} + f_{E240} + v_1, \tag{c_{32}}$$

$$y_{E242} := b_{E242} + f_{E242} + v_2, \tag{c_{33}}$$

$$y_{E265} := b_{E265} + f_{E265} + v_{rms}, \tag{c_{34}}$$

$$y_{E281} := b_{E281} + f_{E281} + v_{dc}, \tag{c_{35}}$$

$$y_{ESH244A} := f_{ESH244A} + s_{EY244}, \tag{c_{36}}$$

$$y_{ISH236} := f_{ISH236} + s_{CB236}, \tag{c_{37}}$$

$$y_{IT240} := b_{IT240} + f_{IT240} + i_B, \tag{c_{38}}$$

$$y_{IT267} := b_{IT267} + f_{IT267} + i_{rms}, \tag{c_{39}}$$

$$y_{IT281} := b_{IT281} + f_{IT281} + i_{dc}, \tag{c_{40}}$$

$$y_{ST516} := b_{ST516} + f_{ST516} + \omega, \tag{c_{41}}$$

$$y_{TE228} := T_B + b_{TE228} + f_{TE228}, \tag{c_{42}}$$

where, for sensor S, the y_S variable is the sensor output, the b_S variable is the (nominal) sensor bias, and the f_S variable is the fault parameter.

The fault modes of the different components of ADAPT-Lite are listed in Table 1. As illustrated by the model constraints, in our modeling framework, faults explicitly correspond to changes in model parameters $\theta \in \Theta$. Table 2 lists the fault parameters in Θ of the model $\mathcal{M}$ that are associated with each component. For the loads, faults are associated with changes in their resistance parameters or discrete failures (parameter f_L for load L). For a sensor S, the fault parameter f_S captures an additive fault. For a relay or circuit breaker C, the fault parameter f_C captures a change

Table 2: Component Fault Parameters

Component	Fault Parameters
AC483	R_{ac}, f_{ac}
DC485	R_{dc}, f_{dc}
CB236	f_{CB236}
CB262	f_{CB262}
CB266	f_{CB266}
CB280	f_{CB280}
E240	f_{E240}
E242	f_{E242}
E265	f_{E265}
E281	f_{E281}
IT240	f_{IT240}
IT267	f_{IT267}
IT281	f_{IT281}
ST516	f_{ST516}
TE228	f_{TE228}
ESH244A	$f_{ESH244A}$
ISH236	f_{ISH236}
EY244	f_{EY244}
EY260	f_{EY260}
EY272	f_{EY272}
EY275	f_{EY275}
EY284	f_{EY284}
FAN416	R_{fan}, f_{fan}
INV2	f_{INV2}

in its discrete state; since in the nominal mode, all circuit breakers and relays are connected, these fault parameters becoming 0 represent the switch going from a connected to a disconnected state. Similarly, inverter failure is captured with the parameter f_{INV2}.

For the resistive load and sensor faults, the fault parameters can take on one of several profiles, including offset, drift, and intermittent offset profiles, as shown in Fig. 3 (t_f denotes the time of fault occurrence). For an offset, the faulty value $\theta_f(t)$ is defined by

$$\theta_f(t) = \theta(t) + \Delta\theta,$$

where $\theta(t)$ is the nominal value, and $\Delta\theta$ is the offset. A drift is defined by its slope m, i.e,

$$\theta_f(t) = \theta(t) + m(t - t_f).$$

For intermittent offsets, the offset randomly alternates between zero and nonzero values, where the periods of faulty values (Δt_{fi}), the periods of nominal values (Δt_{ni}), and the faulty values during each faulty period ($\Delta\theta_i$) are chosen randomly (see Fig. 3c). The profile is summarized by three parameters, the mean offset $\mu_{\Delta\theta}$, i.e, mean$(\Delta\theta_1, \Delta\theta_2, \ldots)$; the mean faulty time μ_{tf}, i.e, mean$(\Delta t_{f1}, \Delta t_{f2}, \ldots)$; and the mean time it is nominal μ_{tn}, i.e, mean$(\Delta t_{n1}, \Delta t_{n2}, \ldots)$.

For the sensor faults, there is an additional fault profile, stuck, where for sensor S, $f_S(t)$ is such that $y_S(t) = c$, where the stuck value is c and the sensor noise is absent.

In summary, the global model is defined by constraints c_1–c_{42} and the constituent variable sets. The state, parameter, input, and output variable sets are defined by:

- $X = \{T_B, v_o, v_s, \omega\}$
- $\Theta = \{R_{ac}, f_{ac}, R_{dc}, f_{dc}, R_{fan}, f_{fan}, f_{CB236}, f_{CB262}, f_{CB266}, f_{CB280}, f_{E240}, f_{E242}, f_{E265}, f_{E281}, f_{ESH244A}, f_{EY244}, f_{EY260}, f_{EY272}, f_{EY275}, f_{EY281}, f_{INV2}, f_{ISH236}, f_{IT240}, f_{IT267}, f_{IT281}, f_{ST516}, f_{TE228}\}$
- $U = \{u_{CB236}, u_{EY244}, u_{EY260}, u_{EY272}, u_{EY275}, u_{EY281}\}$
- $Y = \{y_{E240}, y_{E242}, y_{E265}, y_{E281}, y_{ESH244A}, y_{ISH236}, y_{IT240}, y_{IT267}, y_{IT281}, y_{ST516}, y_{TE228}\}$

4.2. Structural Model Decomposition

In the context of diagnosis, structural model decomposition offers several advantages. For fault isolation, its primary advantage is that we can create local submodels whose outputs are a function of only a subset of the faults, i.e., some faults become decoupled from the residuals computed from the local submodel outputs, which can improve diagnosability. For fault identification, structural model decomposition enables us to find minimal submodels to estimate parameters [18]. Further, it allows for a distributed diagnosis implementation [13].

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

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

A causal submodel is generated from a global (causal) model by defining the set of local outputs the submodel must compute, and the set of local inputs that are available to it. The algorithm for deriving a causal submodel from a global model is detailed in [21]. It works by starting at the local outputs, and propagating backwards trying to resolve variables using constraints involving the local inputs, reassigning causality in the process if needed. The procedure has the property that the generated submodels are minimal, i.e., they contain only the minimal variables and constriants needed to compute the local outputs.

The (causal) global model offers one way in which to compute predicted values of system outputs, in which the inputs used are the global inputs to the system. Instead, we can also use causal submodels, where we define for N sensors a set of N submodels, one for each sensor, where the available local inputs are the global inputs to the system and the measured values from sensors. The advantages to this approach are (*i*) improving diagnosability and (*ii*) enabling a distributed implementation. The disadvantage is that noisy sensor signals are being used as inputs to local submodels to compute predicted values of measurements, typically resulting in a slightly diminished fault detection performance.

The decomposition approach presented in this paper follows the main idea of Possible Conflicts (PCs) [16], where minimal redundant submodels as computed for fault diagnosis purposes by considering sensor measurements as inputs. However, our decomposition approach generalizes that idea and provides a framework where submodels can be computed for other tasks such as fault identification. Additionally, our algorithms allow for the computation of submodels that are not minimal, which have been proven to be useful for distributed diagnosis [13].

For ADAPT-Lite, the system has 11 sensors, so we obtain 11 minimal submodels. Table 3 shows the local state, parameter, input, and output variable sets for each, along with the submodel constraints. Note here that for a sensor S, a $y_S \in U_i$ means that the measured value from the sensor is being used as an input, and a $y_S \in Y_i$ means that a predicted value of y_S is being generated. Note that each submodel computes its values independently of all other submodels, i.e., no submodel outputs are fed into any other submodels, as the only inputs are the known inputs U and measured sensor values. In this way, the submodels can be executed in a parallel, distributed fashion if allowed by the computational hardware.

Table 3: Submodels for the ADAPT System

States (X_i)	**Parameters** (Θ_i)	**Inputs** (U_i)	**Outputs** (Y_i)	**Causal Assignments** ($\mathcal{A}_i$)

v_0, v_s	$f_{CB236}, f_{E240}, f_{IT240}$	u_{CB236}, y_{IT240}	y_{E240}	$y_{E240}{:=}b_{E240} + f_{E240} + v_1$ $v_1{:=}s_{CB236} \cdot v_B$ $v_B{:=}v_0 - v_s$ $s_{CB236}{:=}u_{CB236} f_{CB236}$ $v_o{:=}\int_{t_0}^{t} \dot{v}_0 \, dt$ $\dot{v}_0{:=}{-i_B}/C_0$ $i_B{:=}y_{IT240} - b_{IT240} - f_{IT240}$ $v_s{:=}\int_{t_0}^{t} \dot{v}_s \, dt$ $\dot{v}_s{:=}(R_s i_B - v_s)/C_s$
$\varnothing$	$f_{E240}, f_{E242}, f_{EY244}$	y_{E240}, u_{EY244}	y_{E242}	$y_{E242}{:=}b_{E242} + f_{E242} + v_2$ $v_2{:=}s_{EY244} v_1$ $s_{EY244}{:=}u_{EY244} f_{EY244}$ $v_1{:=}y_{E240} - b_{E240} - f_{E240}$
$\varnothing$	$f_{CB262}, f_{E242}, f_{E265}, f_{EY260}, f_{INV2}$	y_{E242}, y_{EY260}	y_{E265}	$y_{E265}{:=}b_{E265} + f_{E265} + v_{rms}$ $v_{rms}{:=}120 f_{INV2}(v_{inv} > 0)$ $v_{inv}{:=}s_{CB262} v_3$ $s_{CB262}{:=}f_{CB262}$ $v_3{:=}s_{EY260} v_2$ $s_{EY260}{:=}u_{EY260} f_{EY260}$ $v_2{:=}y_{E242} - b_{E242} - f_{E242}$
$\varnothing$	$f_{CB280}, f_{E242}, f_{E281}, f_{EY260}$	y_{E242}, u_{EY260}	y_{E281}	$y_{E281}{:=}b_{E281} + f_{E281} + v_{dc}$ $v_{dc}{:=}s_{CB280} v_3$ $v_3{:=}s_{EY260} v_2$ $s_{CB280}{:=}f_{CB280}$ $s_{EY260}{:=}u_{EY260} f_{EY260}$ $v_2{:=}y_{E242} - b_{E242} - f_{E242}$
$\varnothing$	$f_{ESH244A}, f_{EY244}$	u_{EY244}	$y_{ESH244A}$	$y_{ESH244A}{:=}f_{ESH244A} + s_{EY244}$ $s_{EY244}{:=}u_{EY244} f_{EY244}$
$\varnothing$	f_{CB236}, f_{ISH236}	u_{CB236}	y_{ISH236}	$y_{ISH236}{:=}f_{ISH236} + s_{CB236}$ $s_{CB236}{:=}u_{CB236} f_{CB236}$
$\varnothing$	$f_{CB262}, f_{E242}, f_{E265}, f_{EY260}, f_{IT240}, f_{IT267}, f_{IT281}$	$y_{E242}, y_{E265}, u_{EY260}, y_{IT267}, y_{IT281}$	y_{IT240}	$y_{IT240}{:=}b_{IT240} + f_{IT240} + i_B$ $i_B{:=}i_{inv} + i_{dc}$ $i_{dc}{:=}y_{IT281} - b_{IT281} - f_{IT281}$ $i_{inv}{:=}i_{rms} v_{rms}/(e v_{inv}) + v_{inv}/R_{inv}$ $v_{rms}{:=}y_{E265} - b_{E265} - f_{E265}$ $v_{inv}{:=}s_{CB262} v_3$ $s_{CB262}{:=}f_{CB262}$ $v_3{:=}s_{EY260} v_2$ $s_{EY260}{:=}u_{EY260} f_{EY260}$ $i_{rms}{:=}y_{IT267} - b_{IT267} - f_{IT267}$ $v_2{:=}y_{E242} - b_{E242} - f_{E242}$
$\varnothing$	$R_{ac}, f_{ac}, R_{fan}, f_{fan}, f_{CB266}, f_{E265}, f_{EY272}, f_{EY275}, f_{IT267}$	$y_{E265}, u_{EY272}, u_{EY275}$	y_{IT267}	$y_{IT267}{:=}b_{IT267} + f_{IT267} + i_{rms}$ $i_{rms}{:=}\frac{1}{\sqrt{2}} \left\| \sqrt{2} i_{fan} (\cos\phi + j \sin\phi) + \sqrt{2} i_{ac} \right\|$ $i_{ac}{:=}s_{EY272} f_{ac} v_4/R_{ac}$ $s_{EY272}{:=}u_{EY272} \cdot f_{EY272}$ $i_{fan}{:=}s_{EY275} f_{fan} v_4/R_{fan}$ $v_4{:=}s_{CB266} v_{rms}$ $s_{EY275}{:=}u_{EY275} f_{EY275}$ $s_{CB266}{:=}f_{CB266}$ $v_{rms}{:=}y_{E265} - b_{E265} - f_{E265}$
$\varnothing$	$R_{dc}, f_{dc}, f_{E281}, f_{EY281}, f_{IT281}$	y_{E281}, u_{EY281}	y_{IT281}	$y_{IT281}{:=}b_{IT281} + f_{IT281} + i_{dc}$ $i_{dc}{:=}s_{EY281} f_{dc} v_{dc}/R_{dc}$ $s_{EY281}{:=}u_{EY281} f_{EY281}$ $v_{dc}{:=}y_{E281} - b_{E281} - f_{E281}$

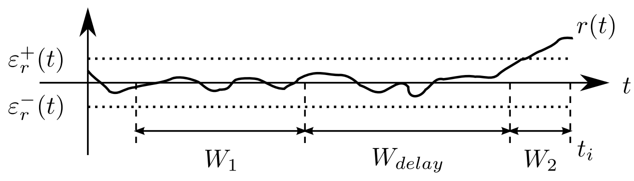


Figure 4: Sliding windows in the fault detection scheme at time t_i.

ω	$R_{fan}, f_{fan}, f_{CB266}, f_{E265}, f_{ST516}$	u_{E265}	y_{ST516}	$y_{ST516}{:=}b_{ST516} + f_{ST516} + \omega$ $\omega{:=}\int_{t_0}^{t} \dot{\omega}\, dt$ $\dot{\omega}{:=}-\omega/J_{fan} + s_{EY275} f_{fan} v_4/(B_{fan} \cdot J_{fan} \cdot R_{fan})$ $v_4{:=}s_{CB266} v_{rms}$ $s_{CB266}{:=}f_{CB266}$ $s_{EY275}{:=}u_{EY275} f_{EY275}$ $v_{rms}{:=}y_{E265} - b_{E265} - f_{E265}$
T_B	f_{TE228}	$\varnothing$	y_{TE228}	$y_{TE228}{:=}T_B + b_{TE228} + f_{TE228}$ $\dot{T}_B{:=}0$ $T_B{:=}\int_{t_0}^{t} \dot{T}_B\, dt$

For example, take IT281, where we have $U \cup (Y - \{y_{IT281}\})$ as available local inputs. To compute y_{IT281}, we need to compute the variables f_{IT281}, which is a fault parameter with a known nominal value (i.e., 0), and i_{dc} (constraint c_{40}). Following the causality backwards, to compute i_{dc}, we need to compute s_{EY281}, f_{dc}, v_{dc}, and R_{dc} (constraint c_{19}). Both f_{dc} and R_{dc} are parameters with known nominal values, and s_{EY281} can be computed with the known input u_{EY281} and f_{EY281}, a fault parameter with a known nominal value (constraint c_{20}). To compute v_{dc}, we need s_{CB262} and v_3 (constraint c_{11}), however, we know also that E281 measures this voltage, so we can use the related constraint (constraint c_{35}) in the causality where y_{E281} becomes an input, thus completing the submodel.

As we can see from Table 3, each submodel is a function of only a subset of the fault parameters. Therefore, if a fault occurs, it will cause a discrepancy in only a subset of the local outputs from their observed values. For example, if R_{fan} changes, we will see a discrepancy only in the predicted values of y_{IT267} and y_{ST516}. For the remaining submodel outputs, the submodels will correctly predict the (faulty) sensor values since they are using other sensors as inputs.

The three diagnostic algorithms, QED, QED-PC, and QED-PC++ differ in which set of models/submodels are used. QED uses only the global system model, and so computes 11 (global) outputs. QED-PC uses the set of minimal submodels shown in Table 3, and so computes 11 (local) outputs. QED-PC++, on the other hand, uses both the global model and the minimal submodels, computing a total of 22 outputs.

Recall that we seek to use the most parsimonious models to describe the system behavior. For QED, we actually do not need to model the complete dynamics, and can simplify the model in some parts because in nominal operation (for which we need to predict system behavior) the dynamics are limited to certain ranges. For example, the fan speed (ω) is constant during nominal operation because it is always operated at the same speed, where any deviation from that speed indicates a fault, and hence does not need to be considered. So, the global model can simply assume $\dot{\omega} = 0$. The local submodel computing y_{ST516}, on the other hand, must model the dynamics, because some faults independent of this submodel, e.g. a fault in the inverter, will cause the fan speed to decrease through a decrease in v_{rms}, for which y_{E265} is used as an input to the y_{ST516} submodel. Consequently, this submodel should correctly predict the gradual decrease in fan speed. Similarly, the global model may assume $v_{rms} = 120$ V rms, since any deviation implies a fault. It can also omit the C_s/R_s pair from the battery model, because it does not need to account for changes in the current drawn from the battery (any change in current implies a fault).

5. Fault Detection

For every output y, the fault detection module compares the measured signal $y(t)$ with each available prediction of that signal $\hat{y}(t)$. The difference of these signals is known as the *residual*, and our diagnosis approach is fundamentally based on the analysis of residual signals.

Definition 5 (Residual)**.** A *residual*, r_y, is a time-varying signal computed as the difference between a measurement, $y \in Y$, and a predicted value of the measurement y, denoted as $\hat{y}$.

QED computes 11 outputs and, from this, 11 residuals are computed based on the global model. QED-PC also computes 11 outputs and 11 residuals, but based on the local submodels, and so will have different fault response characteristics. QED-PC++ computes the combined 22 residuals and so has more analytical redundancy to base its diagnosis on.

Nominally, residuals are, in a statistical sense, zero, accounting for sensor noise and modeling errors. The explicit goal of fault detection is to determine *statistically significant* deviations of the residual values from zero, which indicates the presence of a fault. In this framework, for robust real-time fault detection, the Z-test with a set of sliding windows is used (see Fig. 4). This approach is described in detail in [23, 24] and it is summarized here for completeness. First, a small window, W_2, is used to estimate the current mean $\mu_r(t)$ of a residual signal, where t refers to discrete time:

$$\mu_r(t) = \frac{1}{W_2} \sum_{i=t-W_2+1}^{t} r(i).$$

The variance of the nominal residual signal, $\sigma_r^2(t)$, is computed using a large window W_1 preceding W_2, by a buffer W_{delay} that is meant to ensure that W_1 contains no samples after fault occurrence. The variance is computed by

$$\sigma_r^2(t) = \frac{1}{W_1} \sum_{i=t-W_2-W_{delay}-W_1+1}^{t-W_2-W_{delay}} (r(i) - \mu_r'(t))^2,$$

where

$$\mu_r'(t) = \frac{1}{W_1} \sum_{i=t-W_2-W_{delay}-W_1+1}^{t-W_2-W_{delay}} r(i).$$

A user-specified confidence level determines the $z^- < 0$ and $z^+ > 0$ bounds for a two-sided Z-test. The fault detection thresholds, $\varepsilon_r^-(t)$ and $\varepsilon_r^+(t)$, are dynamically computed using:

$$\varepsilon_r^-(t) = z^- \frac{\sigma_r(t)}{\sqrt{W_2}} - E,$$
$$\varepsilon_r^+(t) = z^+ \frac{\sigma_r(t)}{\sqrt{W_2}} + E,$$

where E is a modeling error term. A fault is detected if $\mu_r(t)$ lies outside of $[\varepsilon_r^-(t), \varepsilon_r^+(t)]$ at time t.

The parameters W_1, W_2, W_{delay}, the z bounds, and E must be tuned to optimize performance. The W_1 window must be large enough to accurately compute the nominal residual variance, but constrained by memory requirements. The W_2 window must be large enough to accurately compute the mean, but small enough to be respond quickly to faults. In our implementation, window sizes varied, and $100 \leq W_1 \leq 200$, $10 \leq W_2 \leq 50$, and $W_{delay} = 10$ were sufficient.

For ADAPT-Lite, the selected error terms E for QED and QED-PC are listed in Table 4 (QED-PC++ uses both). These values were tuned to obtain maximum fault sensitivity without false alarms. Overall, the thresholds are smaller for QED than for QED-PC. This is expected since the minimal submodels use typically noisy sensor signals as inputs to compute their outputs. Further, in some cases the minimal submodels must accurately capture more nominal dynamics with respect to a wider range of inputs as compared to the global model. For example, consider the submodel that takes in as input u_{E265}, the rms inverter voltage of 120 V rms, and outputs fan speed y_{ST516}. For QED, voltage v_{rms} is always computed to be 120 V rms, and so the fan speed is always the same. For QED-PC, on the other hand, the fan speed is modeled using a local submodel. In this case, v_{rms}, which is replaced by y_{E265} as a local input, can become zero if a fault such as the inverter failure occurs. Hence, the submodel has to accurately capture the decrease in fan speed when this happens, and because these dynamics are more difficult to model accurately than a constant value, a larger error threshold is needed for this residual (i.e., $E = 20$ rpm for QED and $E = 40$ rpm for QED-PC).

Table 4: E Values for Fault Detection

Residual	**QED**	**QED-PC**
r_{E240}	0.090	0.100
r_{E242}	0.080	0.100
r_{E265}	0.070	0.110
r_{E281}	0.150	0.300
$r_{ESH244A}$	0.000	0.000
r_{ISH236}	0.000	0.000
r_{IT240}	0.110	0.114
r_{IT267}	0.050	0.060
r_{IT281}	0.060	0.090
r_{ST516}	20.000	40.000
r_{TE228}	0.000	0.200

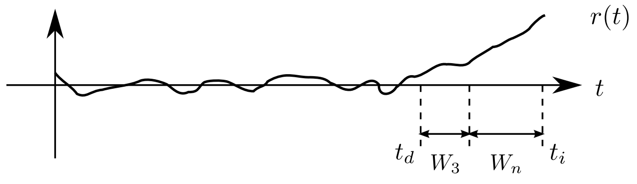


Figure 5: Slope symbol calculation.

Note that for stuck faults in sensors (recall from Section 4 that these faults exhibit no noise in the signal $y(t)$), a sensor that is stuck within nominal ranges will not be detected by the above method. Hence, a new detection test for these faults is required. For sensor y,

$$stuck_y(t) = \begin{cases} true, & \sum_{i=1}^{N_s} |y(t) - y(t-i)| = 0 \\ false, & \text{otherwise} \end{cases},$$

where N_s is a user-defined limit, i.e., if the past N_s samples of $y(t)$ are all the same, then $stuck(t) = true$. To implement this, it is sufficient to maintain a counter k_s for each sensor that keeps track of how many consecutive values of $y(t)$ up to the current time point are the same. A sensor is stuck when $k_s \geq N_s$. The selected value of N_s depends on the particular sensor. For some sensors in ADAPT-Lite, N_s must be quite large (e.g., $N_s = 400$), because the sensors normally repeat the same value for long periods of time. For the discrete sensors ISH236 and ESH244A this test is not applied, because these sensors are binary-valued and noiseless, so will nominally have long sequences of repeated measured values.

6. Symbol Generation

Robust methods based on the Z-test are also used for symbol generation [24]. Each of the three algorithms use the same method for each of their computed residual signals. The first symbol is derived from the result of fault detection. If $r(t)$ is greater than $\varepsilon_r^+(t)$ (resp. less than $\varepsilon_r^-(t)$), a + (resp. −) is obtained. The second symbol is calculated for the direction of the slope of the residual, as explained below.

The approach first starts with an estimate of the initial residual value, $\mu_{r_0}(t_d)$, at the time of fault detection, t_d, over a small window W_3 (see Fig. 5):

$$\mu_{r_0}(t_d) = \frac{1}{W_3} \sum_{i=t_d}^{t_d+W_3-1} r(i).$$

The mean of the residual slope is computed over a window from t_d to t:

$$\mu_{r_d}(t) = \frac{1}{t - t_d + 1}\left(\sum_{i=t_d}^{t} r(i) - \mu_{r_0}\right).$$

Using bounds z^- and z^+, the thresholds are:

$$\varepsilon^-_{r_d}(t) = z^- \sigma_r \left(\frac{1}{\sqrt{W_3}} + \frac{1}{\sqrt{W_n}}\right) - E_s$$
$$\varepsilon^+_{r_d}(t) = z^+ \sigma_r \left(\frac{1}{\sqrt{W_3}} + \frac{1}{\sqrt{W_n}}\right) + E_s,$$

where E_s is a modeling error term. The + (resp. −) symbol is generated when $\mu_{r_d} > \varepsilon^+_{r_d}(t)$ (resp. $\mu_{r_d} < \varepsilon^-_{r_d}(t)$). The window used to calculate the slope, W_n, is increased until the symbol is successfully generated, or $t - t_d$ becomes larger than a user-specified limit, at which point the slope is reported as `0`, implying that the true slope is either zero or unknown, but very small. The window sizes for symbol generation are decided upon in the same manner as those for fault detection. For all residuals we used $W_3 = 10$ and $W_n = 100$.

A discrete change symbol, representing whether a signal has changed between a nonzero and zero value, which is used to distinguish between parametric and discrete faults, is also computed. A `Z` symbol is reported if the estimate is nonzero and the measurement is zero, `N` symbol if the estimate is zero and the measurement is nonzero, and `X` symbol otherwise. The procedure to compute this value is detailed in [24].

Table 5: Signatures for Selected Faults for QED for ADAPT-Lite

Component	**Fault Mode**	r_{E240}	r_{E242}	r_{E265}	r_{E281}	$r_{ESH244A}$	r_{ISH236}	r_{IT240}	r_{IT267}	r_{IT281}	r_{ST516}	r_{TE228}
AC483	Offset ($\Delta p > 0$)	+0X	+0X	+0X	+0X	00X	00X	−0X	−0X	+0X	00X	00X
DC485	Offset ($\Delta p > 0$)	+0X	+0X	00X	+0X	00X	00X	−0X	00X	−0X	00X	00X
DC485	Drift ($m < 0$)	0−X	0−X	00X	0−X	00X	00X	0+X	00X	0+X	00X	00X
E240	Offset ($\Delta p > 0$)	+0X	00X	00X	00X	00X	00X	00X	00X	00X	00X	00X
E240	Drift ($m > 0$)	0+X	00X	00X	00X	00X	00X	00X	00X	00X	00X	00X
E240	Intermittent Offset ($\mu_{\Delta p} > 0$)	+0X	00X	00X	00X	00X	00X	00X	00X	00X	00X	00X
EY244	Stuck Open	+0X	−0Z	−0Z	−0Z	−0Z	00X	−0Z	−0Z	−0Z	0−X	00X
FAN416	Underspeed	+0X	+0X	+0X	00X	00X	00X	−0X	−0X	00X	−0X	00X
IT267	Offset ($\Delta p < 0$)	00X	00X	00X	00X	00X	00X	00X	−0X	00X	00X	00X

Table 6: Signatures for Selected Faults for QED-PC for ADAPT-Lite

Component	**Fault Mode**	r_{E240}	r_{E242}	r_{E265}	r_{E281}	$r_{ESH244A}$	r_{ISH236}	r_{IT240}	r_{IT267}	r_{IT281}	r_{ST516}	r_{TE228}
AC483	Offset ($\Delta p > 0$)	00X	00X	00X	00X	00X	00X	00X	−0X	00X	00X	00X
DC485	Offset ($\Delta p > 0$)	00X	00X	00X	00X	00X	00X	00X	00X	−0X	00X	00X
DC485	Drift ($m < 0$)	00X	00X	00X	00X	00X	00X	00X	00X	0+X	00X	00X
E240	Offset ($\Delta p > 0$)	+0X	−0X	00X	00X	00X	00X	00X	00X	00X	00X	00X
E240	Drift ($m >$)0	0+X	0−X	00X	00X	00X	00X	00X	00X	00X	00X	00X
E240	Intermittent Offset ($\mu_{\Delta p} > 0$)	+0X	−⋆X	00X	00X	00X	00X	00X	00X	00X	00X	00X
EY244	Stuck Open	00X	−0Z	−0Z	00X	−0Z	00X	00X	00X	00X	00X	00X
FAN416	Underspeed	00X	00X	⋆⋆X	00X	00X	00X	00X	−0X	00X	−0X	00X
IT267	Offset ($\Delta p < 0$)	00X	00X	−0X	00X	00X	00X	+0X	−0X	00X	00X	00X

7. Fault Isolation

The goal of the fault isolation module is, based on the output of the symbol generator, to generate the list of consistent faults F. As new symbols are provided, F shrinks as inconsistent faults are eliminated. We use a qualitative fault isolation methodology that isolates faults based on the transients they produce in the system behavior, which manifest as deviations in residual values [10]. The symbolic representation of these deviations, produced by the symbol generator, which represent the immediate (discontinuous) change in magnitude, the first nonzero derivative change, and discrete zero/nonzero value changes in the measurement from the estimate, are called *fault signatures*. Predicted fault signatures, derived from the system models/submodels are matched to those derived by symbol generation in order to perform fault isolation. The temporal ordering of observed deviations is also used as additional diagnostic information, based on the intuition that fault effects will manifest in some parts of the system before others. These temporal orderings of deviations produced by a fault are termed *relative residual orderings* [11]. Qualitative fault signatures and residual orderings can be computed by manual analysis of the system model, by simulation, or automatically from certain types of models, such as a temporal causal graph as presented in [10, 25].

Fault signatures for a subset of the ADAPT-Lite faults are shown in Table 5 for the QED algorithm and in Table 6 for the QED-PC algorithm. As mentioned, the first symbol represents an immediate change in magnitude, the second represents changes in the slope, and the third represents discrete changes. As an example, let us consider a positive offset in E240. This fault will cause an abrupt increase in the E240 residual (+), no change in the slope (0), and no discrete change behavior (X). As can be seen in Table 5, no other sensor is affected by this fault in the QED approach (00X). For QED-PC, the positive offset in E240 produces not only the abrupt increase in the E240 residual, but also an abrupt decrease in the E242 residual with no change in slope, and no discrete change behavior (-0X). Since measured values from E240 are used as input for the minimal submodel that estimates E242, a fault in E240 causes a deviation in the residual associated with E242. Resistance offsets in AC483 and DC485 cause multiple deviations for QED but only one in IT267 and IT281, respectively, for QED-PC. This is caused because the corresponding parameters, R_{ac} and R_{dc}, are present only in the minimal submodels for IT267 and IT281, respectively, as shown in Table 3.

Using all the signatures and orderings, it can be determined if the system is fully diagnosable with that information, i.e., that all faults can be distinguished from each other. It is clear from Tables 5 and 6, that the system is not fully diagnosable. Specifically, an offset in E240 causes the same (initial) effects in the residuals as an intermittent offset. This is the case for all other sensors as well along with DC485 and AC483, and consequently, fault identification is needed here to resolve the ambiguity. Besides these cases, there are four pairs of faults that produce exactly the same *quantitative* behavior on the given residuals: failures in CB262 and INV2, failures in EY281 and DC485, failures in EY272 and AC483, and failures in EY275 and FAN416. Further, for QED, since sensors are only used at the output to compute residuals, a sensor fault affects only a single residual. Since we are assuming no-exoneration ([26, 27]) in our work, the absence of residual activations will not be used to exonerate fault candidates. The consequence of this is that, when a sensor fault occurs, it is necessary to wait to confirm that no other residuals deviate before concluding that a sensor fault has indeed occurred. For QED-PC, since sensors are used not only as output, but also as input to the submodels, sensor faults will affect multiple residuals, so this problem is minimized. However, for QED-PC, faults in DC485 and AC483 affect only a single residual, thus having an analogous situation. Due to these diagnosability properties, in general, QED will isolate nonsensor faults faster than QED-PC, and QED-PC will isolate sensor faults faster than QED. By combining these residual sets we improve diagnosability over QED or QED-PC by themselves, and this is the prime motivation for QED-PC++.

In theory, the absence of a residual deviation cannot be used to exonerate a candidate, because the deviation could always occur some time in the future. However, in practice, candidates can be exonerated using this information if a time limit, measured from the detection of a fault, within which a deviation must be observed, can be confidently introduced. If an expected deviation is not seen within that time limit, this is taken as a direct observation of a 00X signature and so we eliminate any candidate that predicts otherwise. Such a strategy can be generally applied in any case where a nonzero signature exists in the fault signature table, but because this depends on the sensitivity of fault detection, it is possible that some deviations may not be detected, even if anticipated. Therefore, for the ADAPT case study, rules of this form are selectively applied only for residuals for which an easily detected deviation is expected. By using this strategy, the distinguishability of the faults can be improved, and the fault isolation times can be decreased.

The isolation rules implemented for ADAPT-Lite are summarized in Table 7. The table lists the components, the sensors whose residuals must deviate, the time when the rule is fired (measured from the time of fault detection), and

Table 7: Heuristic Fault Isolation Rules

Component	Residual	Firing Time (s)	QED	QED-PC	QED-PC++
AC483	r_{E265}	40	✓		
AC483	r_{IT267}	40	✓	✓	
CB236	r_{ISH236}	0.2	✓	✓	
CB236	r_{E265}	10	✓	✓	
CB262	r_{E265}	10	✓	✓	✓
CB266	r_{IT267}	10	✓	✓	✓
E265	$Any\ 3$	70		✓	
E265	r_{IT240}	70		✓	
E281	$Any\ 2$	70		✓	
EY244	$r_{ESH244A}$	0.2	✓	✓	
EY244	r_{E265}	10	✓	✓	
EY260	r_{E265}	10	✓	✓	
EY260	r_{E281}	10	✓	✓	
EY272	r_{IT267}	40	✓	✓	✓
EY272	r_{E265}	40	✓		
EY275	r_{E265}	30	✓		
EY275	r_{IT267}	30	✓	✓	
EY275	r_{ST516}	30	✓	✓	✓
EY284	r_{IT281}	10	✓	✓	✓
FAN416	r_{E265}	30	✓		
FAN416	r_{IT267}	30	✓	✓	
FAN416	r_{ST516}	30	✓	✓	✓
INV2	r_{E265}	10	✓	✓	✓
IT267	$Any\ 2$	70		✓	
IT281	$Any\ 2$	70		✓	
Sensor S	r_S	60		✓	✓

which algorithm the rule is applied to (represented using a check mark). The rules are to be read as, "if the candidate concerns component C and the residual for sensor S has not deviated within T seconds since fault detection, then eliminate the candidate". For example, if CB236 is faulty, the sensor measuring its position, ISH236, is expected to deviate almost immediately (within 0.2 s), and E265 to deviate within 10 s. If FAN416 is faulty, IT267 and ST516 are expected to deviate within 30 s, and, for QED only, E265 to deviate within 30 s. For some sensor faults for QED-PC, a certain number of residual deviations are expected to confirm a fault (see E265, E281, IT267, and IT281 in the table). Also for QED-PC, a fault in any sensor is expected to cause a deviation in the residual for that sensor within 60 s, as shown in the last row of the table. Note that for QED, this rule would be redundant because a sensor fault cannot be generated as a candidate unless a deviation in that sensor's residual is seen.

For QED-PC++, the system is diagnosable with the combined residual set, except for the known undistinguishable faults, so none of the fault isolation rules are actually needed for unique isolation of faults. Some of the rules are still kept, since they improve fault isolation times. For example, for QED a rule to rule out non sensor faults is needed if after a significant amount of time only one residual has been observed to deviate; QED-PC++ does not require that rule because multiple submodel-based residuals will deviate due to a sensor fault. Table 7 shows which rules are still used for QED-PC++.

8. Fault Identification

The goal of fault identification is to, given a hypothesized fault f (consisting of a faulty component and its fault mode), determine the parameters that define the fault. Identification is performed for each fault in the candidate list F to produce a new candidate list F_{id} augmented with estimated fault parameters. Each of the algorithms use the same approach.

As described in Section 4, each fault is mapped to a single parameter in the system model. For each parameter θ, we have a known nominal value $\theta(t)$. When a fault occurs, it takes on some new profile, $\theta_f(t)$ (offset, drift, or intermittent offset). The goal of fault identification is to identify the signal $\theta_f(t)$, and based on $\Delta\theta = \theta_f(t) - \theta(t)$, identify its specific profile and the parameters defining that profile.

The faults that require identification are faults in the resistances R_{ac} and R_{dc}, and sensor faults (f_S for a sensor S). We use structural model decomposition here in order to obtain minimal submodels that compute those parameters, where the local output of the submodel is the parameter value (i.e., the $\theta_f(t)$), and the local inputs are measured signals from the available sensors Y in the global model. Using the model decomposition algorithm, we obtain the submodels shown in Table 8.

Table 8: Fault Identification Submodels for ADAPT-Lite

States (X_i)	**Parameters** (Θ_i)	**Inputs** (U_i)	**Outputs** (Y_i)	**Causal Assignments** ($\mathcal{A}_i$)
$\varnothing$	$f_{E281}, f_{EY281}, f_{IT281}$	$y_{E281}, u_{EY281}, y_{IT281}$	R_{dc}	$v_{dc}{:=}y_{E281} - b_{E281} - f_{E281}$ $i_{dc}{:=}y_{IT281} - b_{IT281} - f_{IT281}$ $s_{EY281}{:=}u_{EY281} \cdot f_{EY281}$ $R_{dc}{:=}s_{EY281} \cdot v_{dc}/i_{dc}$
$\varnothing$	$R_{fan}, f_{CB266}, f_{E265}, f_{EY272}, f_{EY275}$	$y_{E265}, u_{EY272}, u_{EY275}$	R_{ac}	$i_{ac}{:=}-i_{fan} + i_{rms}$ $s_{EY272}{:=}u_{EY272} \cdot f_{EY272}$ $i_{fan}{:=}s_{EY275} \cdot v_4/R_{fan}$ $v_4{:=}s_{CB266} \cdot v_{rms}$ $s_{CB266}{:=}f_{CB266}$ $s_{EY275}{:=}u_{EY275} \cdot f_{EY275}$ $v_{rms}{:=}y_{E265} - b_{E265} - f_{E265}$ $i_{rms}{:=}\frac{1}{\sqrt{2}} \left\| \sqrt{2} i_{fan} (\cos\phi + j \sin\phi) + \sqrt{2} i_{ac} \right\|$ $R_{ac}{:=}s_{EY272} \cdot v_4/i_{ac}$
v_o, v_s	f_{CB236}, f_{IT240}	$u_{CB236}, y_{E240}, y_{IT240}$	f_{E240}	$v_1{:=}s_{CB236} \cdot v_B$ $v_B{:=}v_o - v_s$ $s_{CB236}{:=}f_{CB236} \cdot u_{CB236}$ $v_s{:=}\int_{t_0}^{t} \dot{v}_s \, dt$ $\dot{v}_s{:=}(R_s \cdot i_B - v_s)/C_s$ $v_o{:=}\int_{t_0}^{t} \dot{v}_o \, dt$ $i_B{:=}-b_{IT240} - f_{IT24}0 + y_{IT240}$ $\dot{v}_o{:=}-i_B/C_0$ $f_{E240}{:=}-b_{E240} - v_1 + y_{E240}$
$\varnothing$	f_{E240}, f_{EY244}	$u_{EY244}, y_{E240}, y_{E242}$	f_{E242}	$v_2{:=}s_{EY244} \cdot v_1$ $v_1{:=}-b_{E240} - f_{E240} + y_{E240}$ $s_{EY244}{:=}f_{EY244} \cdot u_{EY244}$ $f_{E242}{:=}-b_{E242} - v_2 + y_{E242}$
$\varnothing$	$f_{CB262}, f_{E242}, f_{EY260}, f_{INV2}$	$u_{EY260}, y_{E242}, y_{E265}$	f_{E265}	$v_{rms}{:=}120 \cdot f_{INV2} \cdot v_{inv}$ $v_{inv}{:=}s_{CB262} \cdot v_3$ $s_{CB262}{:=}f_{CB262}$ $v_3{:=}s_{EY260} \cdot v_2$ $v_2{:=}-b_{E242} - f_{E242} + y_{E242}$ $s_{EY260}{:=}f_{EY260} \cdot u_{EY260}$ $f_{E265}{:=}-b_{E265} - v_{rms} + y_{E265}$
$\varnothing$	$f_{CB280}, f_{E242}, f_{EY260}$	$u_{EY260}, y_{E242}, y_{E281}$	f_{E281}	$v_{dc}{:=}s_{CB280} \cdot v_3$ $s_{CB280}{:=}f_{CB280}$ $v_3{:=}s_{EY260} \cdot v_2$ $v_2{:=}-b_{E242} - f_{E242} + y_{E242}$ $s_{EY260}{:=}f_{EY260} \cdot u_{EY260}$ $f_{E281}{:=}-b_{E281} - v_{dc} + y_{E281}$
$\varnothing$	$f_{CB262}, f_{E242}, f_{E265}, f_{EY260}, f_{IT267}, f_{IT281}$	$u_{EY260}, y_{E242}, y_{E265}, y_{IT240}, y_{IT267}, y_{IT281}$	f_{IT240}	$i_B{:=}i_{inv} + i_{dc}$ $i_{dc}{:=}-b_{IT281} - f_{IT281} + y_{IT281}$ $i_{inv}{:=}i_{rms} \cdot v_{rms}/(e \cdot v_{inv}) + v_{inv}/R_{inv}$

				i_{rms}:=$-b_{IT267} - f_{IT267} + y_{IT267}$ $v_{rm}s$:=$-b_{E265} - f_{E265} + y_{E265}$ v_{inv}:=$s_{CB262} \cdot v_3$ v_3:=$s_{EY260} \cdot v_2$ s_{EY260}:=$f_{EY260} \cdot u_{EY260}$ v_2:=$-b_{E242} - f_{E242} + y_{E242}$ s_{CB262}:=f_{CB262} f_{IT240}:=$-b_{IT240} - i_B + y_{IT240}$
$\varnothing$	$R_{ac}, R_{fan}, f_{CB266}, f_{E265}, f_{EY272}, f_{EY275}$	$u_{EY272}, u_{EY275}, y_{E265}, y_{IT267}$	f_{IT267}	i_{rms}:=$\frac{1}{\sqrt{2}} \left\lvert \sqrt{2} i_{fan} (\cos\phi + j \sin\phi) + \sqrt{2} i_{ac} \right\rvert$ i_{fan}:=$s_{EY275} \cdot v_4 / R_{fan}$ v_4:=$s_{CB266} \cdot v_{rms}$ i_{ac}:=$s_{EY272} \cdot v_4 / R_{ac}$ s_{EY275}:=$f_{EY275} \cdot u_{EY275}$ s_{EY272}:=$f_{EY272} \cdot u_{EY272}$ s_{CB266}:=f_{CB266} v_{rms}:=$-b_{E265} - f_{E265} + y_{E265}$ f_{IT267}:=$-b_{IT267} - i_{rms} + y_{IT267}$
$\varnothing$	$R_{dc}, f_{E281}, f_{EY281}$	$u_{EY281}, y_{E281}, y_{IT281}$	f_{IT281}	i_{dc}:=$s_{EY281} \cdot v_{dc} / R_{dc}$ v_{dc}:=$-b_{E281} - f_{E281} + y_{E281}$ s_{EY281}:=$f_{EY281} \cdot u_{EY281}$ f_{IT281}:=$-b_{IT281} - i_{dc} + y_{IT281}$
ω	$R_{fan}, f_{CB266}, f_{E265}$	y_{E265}, y_{ST516}	f_{ST516}	ω:=$\int_{t_0}^{t} \dot{\omega}\, dt$ $\dot{\omega}$:=$-\omega / J_{fan} + v_4 / (B_{fan} \cdot J_{fan} \cdot R_{fan})$ v_4:=$s_{CB266} \cdot v_{rms}$ s_{CB266}:=f_{CB266} v_{rms}:=$-b_{E265} - f_{E265} + y_{E265}$ f_{ST516}:=$-b_{ST516} - \omega + y_{ST516}$
T_B	$\varnothing$	y_{TE228}	f_{TE228}	T_B:=$\int_{t_0}^{t} \dot{T}_B\, dt$ $\dot{T}_B$:=0 f_{TE228}:=$-T_B - b_{TE228} + y_{TE228}$

So, using these local submodels, we obtain for each fault f the history of θ_f over $[t_d, t]$. Knowing the nominal value $\theta(t)$, we can compute $\Delta\theta(t)$, and by analyzing this signal, we can compute the parameter values of the fault profile as follows:

- For stuck faults, the stuck value is taken as $\theta_f(t)$.
- For offset faults, the offset value, $\Delta\theta$, is computed at time t as the mean of $\Delta\theta$ over $[t_d, t]$.
- For drift faults, for a given interval $[t_1, t_2]$, the mean of $\Delta\theta$ is computed over a window of n samples centered at t_1, $\Delta\theta_1$, and at t_2, $\Delta\theta_2$. The slope is then computed as $m_{1,2} = (\Delta\theta_2 - \Delta\theta_1)/(t_2 - t_1)$. Three time intervals are chosen: $[t_d, (t + t_d)/2]$, $[(t + t_d)/2, t]$, and $[t_d, t]$, and the slope m is computed as the median of the three slopes ($\Delta\theta_1$, $\Delta\theta_2$, and $\Delta\theta_3$ in the figure). Taking the mean of $\Delta\theta$ over the interval endpoints and taking the median of computed slope values helps improve the robustness.
- For intermittent offset faults, a limit l. above which $\Delta\theta(t)$ is considered faulty and below which is considered nominal, is used. The limit l is typically chosen as within 1-2% of the nominal value of $y(t)$ or $\theta(t)$. The approach steps through the signal $\Delta\theta(t)$, and maintains two counters k_n and k_f. Each time a transition from a nominal value to a faulty value occurs, k_f is incremented, and when a transition from a faulty value to a nominal value occurs, k_n is incremented. In effect, these two counters keep track of the number of times the signal was faulty and nominal. For each new nominal value, a second counter τ_n, that keeps track of the total amount of time the signal is nominal, is incremented. Similarly, for each new faulty value, a counter τ_f, that keeps track of the total amount of time it is faulty, is incremented. A variable v_f, that keeps track of the sum of all faulty values, is also

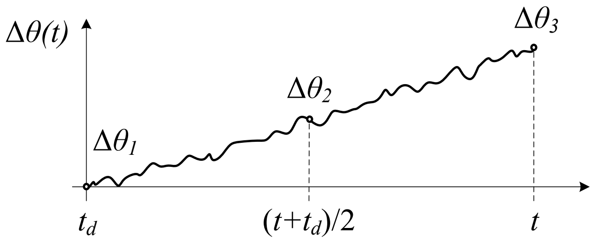


Figure 6: Identification of drift faults.

incremented. Then, the fault parameters are

$$\mu_{\Delta p} = \frac{v_f}{\tau_f},\ \mu_f = \frac{\tau_f}{k_f},\ \mu_n = \frac{\tau_n}{k_n}.$$

In addition to the parameter values of the fault profiles listed above, for each fault, we also compute the fault injection time. For offset, stuck, and intermittent faults, we take the fault injection time $t_f = t_d$. For drift faults, since we have the slope $m_{1,2}$ and the current value of the parameter θ_f, we can compute the time when θ_f crossed zero. Since there is always a delay in detecting drift faults, we expect to find $t_f < t_d$. However, due to noise and errors in the identification of the slope, the computed fault injection time is sometimes computed to also be before the actual fault injection time.

Fault identification is also used to improve fault isolation. Each candidate is represented by a faulty parameter with a proposed fault profile. If the identification results are inconsistent with the proposed profile, we can either eliminate the candidate or change its profile to a consistent one. To do this, we have implemented consistency tests for each fault profile. For example, if the proposed profile is an offset but we find that it has a significant slope, then it cannot be an offset fault and is likely a drift fault. The proposed fault profile will change according to the results of these tests, and if all tests fail, the candidate is eliminated entirely. Candidates can also be eliminated if the identified parameters are invalid (e.g., if an estimated resistance value is negative).

9. Decision Making

As shown in Fig. 2, at the end of the scenario, the decision whether to abort or continue the mission must be made. The fault identification module computes a candidate set F_{id}, with each $f \in F_{id}$ being defined by the component, its fault mode, and the associated fault parameters. The deciion maker implements a function $DM(f)$ which, for a given fault, computes a recommended command c. In ADAPT-Lite either $c = abort$ (i.e., abort the mission) or no command is provided (i.e., continue the mission). Table 1 summarizes the abort recommendation for each fault mode in ADAPT-Lite.

The cost of the decision (abort or continue) is zero when the correct command is chosen. If the algorithm recommends *abort* when the mission should be continued, the associated cost is 25, which is defined to be the cost of the mission. If the algorithm recommends to continue when it should have been aborted, the associated cost is 125 which is the cost of the mission, 25, plus the cost of the vehicle, defined to be 100. Therefore, the conservative approach is taken and *abort* is recommended if $DM(f) = abort$ for at least one $f \in F_{id}$. In the case that a fault was detected but all candidates were eliminated, then one may assume either a false positive, or a true positive with incorrect fault isolation. The latter is assumed, and in this case, the approach again conservatively recommends an abort. A decision to abort or continue is made at 4 minutes into the mission for this case study.

10. Experimental Results

In this section, the QED, QED-PC, and QED-PC++ algorithms are applied to real experimental data collected from the ADAPT hardware. The data was provided as part of the Fourth International Diagnostics Competition [9]. First, the

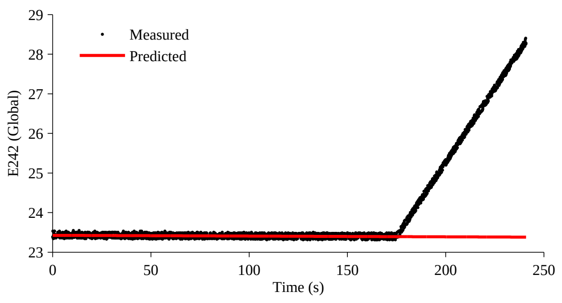


Figure 7: Measured and predicted values of E242 (global model) for E242 drift fault.

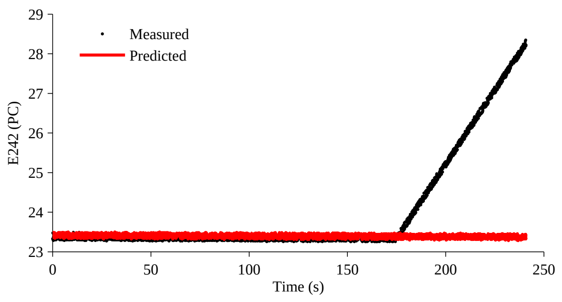


Figure 8: Measured and predicted values of E242 (local submodel) for E242 drift fault.

algorithms are demonstrated on some sample fault scenarios to show how online diagnosis works. Then, overall results are presented on both training and validation data sets, followed by an analysis of the results and a discussion of ways in which the algorithms could be improved. The algorithms are implemented using C++ with the memory requirements being linear in the number of sensors.

10.1. Demonstration of the Approach

In order to demonstrate the differences between the diagnosers, a demonstration scenario in which a drift fault in E242 is injected at 175 s with a slope of 0.075 V/s is described. Measured and predicted values for relevant outputs are shown in Figs. 7–10. QED detects the fault at 176.7 s, with an increase in the E242 residual. At this point, the possible faults are in AC483, CB262, DC485, E242, EY260, EY272, EY275, EY284, FAN416 (failed off or underspeed), and INV2. At 176.9 s, the stuck mode of E242 is eliminated since consecutive measurements were of different values. At 177.1 s, the discrete change symbol is computed as X, which does not change the candidate list. At 179.2 s, the slope of E242 is computed as +, eliminating all candidates but AC483 resistance drift, DC485 resistance drift, and E242 drift. At 220 s, since only one residual had deviated, it is concluded to be a sensor fault and E242 drift is isolated. The injection time is computed as 169.8 s and the magnitude as 0.069 V/s. The recommended action is $\varnothing$, which is correct.

QED-PC detects the fault at 178.1 s on the PC for E242. The thresholds for the residuals generated by the local submodels are larger, so, as expected, fault detection is slower than with QED. The initial candidate list consists only of faults in E240 and E242, since the remaining faults are decoupled from the E242 residual by the local submodel design. This is in contrast to QED, where most components except sensors were implicated with the first residual deviation. At 178.3 s, it is determined that neither E240 or E242 are stuck. At 182.1 s the discrete change symbol for E242 is computed as X, which does not change the candidate list. At 182.1 s, a decrease in the E281 submodel residual and increase in the IT240 submodel residual are detected, isolating the fault to E242 (drift or offset). At 183.1 s, the slope on E242 is computed as +, therefore isolating a drift in E242 as the fault. The injection time is computed as 169.7 s, and the slope as 0.069 V/s. The recommended action is $\varnothing$.

QED-PC++ detects the fault at 176.7 s with the E242 global model residual. The initial candidate list is the same as with QED; the candidate faults are in AC483, CB262, DC485, E242, EY260, EY272, EY275, EY284, FAN416

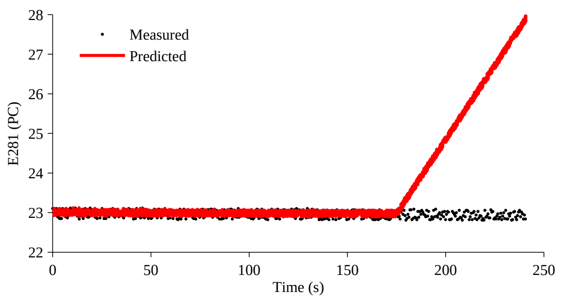


Figure 9: Measured and predicted values of E281 (local submodel) for E242 drift fault.

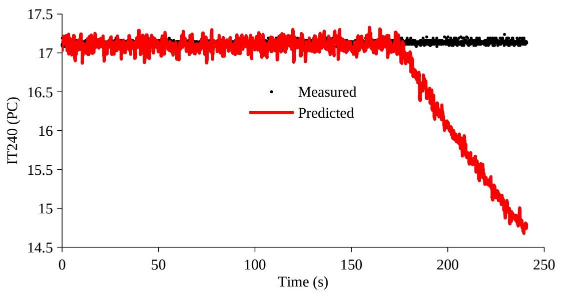


Figure 10: Measured and predicted values of IT240 (local submodel) for E242 drift fault.

(failed off or underspeed), and INV2. At 176.9 s, it is determined that E242 is not stuck. At 177.1 s, the discrete change symbol for E242 is computed as X, and the candidate list remains unchanged. At 178.1 s, an increase in the residual for the E242 submodel is detected, thus eliminating all candidates except for faults in E242. Thus, fault isolation is completed far earlier than QED (about 30 s), and also earlier than QED-PC (about 5 s). The fault is computed as a drift with an injection time of 169.8 s and a slope of 0.069 V/s. The recommended action is $\varnothing$.

10.2. Summary of Results

The set of experimental data is split into two sets, one for training (228 scenarios) and one for validation (115 scenarios). In order to evaluate the performance of the diagnosis algorithms, the following set of metrics are used [8, 9]: the mean time to detect faults M_{fd}, which is the fault detection time minus the fault injection time; the false negative rate M_{fn}, which is the number of missed detections over the number of faulty scenarios; the false positive rate M_{fp}, which is the number of times a fault was detected when a fault was not present over the number of scenarios; detection accuracy M_{da}, which is one minus the sum of false positives and false negatives over the number of scenarios; the mean time to isolate faults M_{fi}, which is the time the candidate list last changed minus the fault injection time; the number of classification errors M_{err}, which is the Hamming distance between the true and diagnosed component mode vectors; the mean CPU time M_{cpu}; the mean peak memory usage M_{mem}; and the overall recovery cost M_{rc}.

Overall results for the training data set are shown in Table 9. Ideally, the algorithms need to be tuned only with respect to the fault detectors and symbol generators. While evaluating the algorithms on the training data, these are

Table 9: Diagnosis Results over Training Data Set

Diagnostic Algorithm	M_{fd} (s)	M_{fn}	M_{fp}	M_{da}	M_{fi} (s)	M_{err}	M_{cpu} (ms)	M_{mem} (kb)	M_{rc}
QED	9.38	0.0152	0.0	0.987	125.61	19	22.72	7675	25
QED-PC	14.66	0.025	0.0	0.978	127.70	37	23.70	7743	275
QED-PC++	9.32	0.0152	0.0	0.987	124.94	19	24.49	7835	25

Table 10: Diagnosis Results over Validation Data Set

Diagnostic Algorithm	M_{fd} (s)	M_{fn}	M_{fp}	M_{da}	M_{fi} (s)	M_{err}	M_{cpu} (ms)	M_{mem} (kb)	M_{rc}
QED	3.70	0.0	0.0	1.00	72.00	27	10.70	7503	125
QED-PC	3.13	0.0	0.4087	0.5913	67.33	87	9.78	7689	1375
QED-PC++	1.65	0.0	0.4435	0.5565	43.84	67	13.62	7850	1250

tuned to be as sensitive as possible without obtaining any false alarms. Some instances of spikes in the data are also found, which are not considered to be faults and cause spurious false alarms. To filter out such data spikes, a median filter over 3 samples is implemented.

Overall, all algorithms do very well after tuning. In the training data set, there are 12 unavoidable classification errors due to the cases for the indistinguishable faults (e.g., DC485 failing versus its relay failing). For QED, the remaining errors are due to incorrect fault mode identification (e.g., identifying as intermittent offset instead of drift) and missed detections. The missed detections are acceptable, because the faults are small enough that the correct action is $\varnothing$. Except in one case, the incorrect mode identification scenarios did not result in an incorrect recommendation. The one scenario that did was one where the identified fault parameters were off just enough so that an abort was recommended when the correct action was $\varnothing$, resulting in the recovery cost of 25.

QED-PC had similar errors, and also some additional ones in which the DC485 and IT281 faults were confused. This is a difficult situation to correct, because the thresholds were difficult to tune. DC485 can only be uniquely isolated if only IT281 deviates, so if additional deviations are seen, it can be concluded that it is an IT281 fault. However, sometimes IT281 did not cause large enough deviations in other residuals so it was misdiagnosed as DC485. These scenarios did not result in a bad recommendation, though. In two cases, QED-PC recommended $\varnothing$ when the correct action was to abort. In the first case, an ST516 fault was misdiagnosed as an E265 fault. This error is due to threshold sensitivities. In the second case, an ST516 intermittent offset fault was misdiagnosed as an offset fault, for which the identified offset was not enough to trigger an abort recommendation.

QED-PC++ generally performed the same as QED. For the cases where the submodel-based residuals were not analyzed correctly, the use of the global model residuals ended up being used for reasoning instead, and so QED-PC++ avoided the errors QED-PC had. QED-PC++ also had the best fault detection and fault isolation times. This was expected since QED-PC++ has more residuals to use for fault detection and isolation.

Overall results for the validation data set are shown in Table 10. In this set of scenarios, there were 20 scenarios in which errors are expected in fault isolation due to indistinguishable faults, so each algorithm can at best have 20 classification errors. For QED, there were no false positives, and 7 classification errors. The classification errors were due in two cases to a misidentified component mode (offset instead of drift, and offset instead of intermittent offset), resulting in 4 total errors, plus 3 cases in which all candidates were eliminated due to an incorrectly generated slope symbol. None of these scenarios had an associated recovery cost. The 125 recovery cost came from one scenario in which the fault was correctly detected and isolated, but the drift was underestimated, and so an *abort* decision was missed. Fault detection and isolation times were overall lower than in the training scenarios, because on average the fault magnitudes were larger, so faults were easier to detect, and there were relatively fewer drift faults, so, on average, faults could be isolated faster.

For QED-PC, a significant increase in false positives occured. The majority of these scenarios were due to the residual for E265. Upon further inspection, it was found that the noise characteristics for E265 between the training and validation data sets were different (these data sets were gathered a year apart), i.e. a bias was introduced, and this is not something the Z-test in its current implementation can account for. This resulted in many false positives on E265. As a result, fault isolation was compromised in the scenarios in which the false alarm was obtained before isolation of the true fault was completed. The performance of QED-PC was actually comparable to its performance on the training data, omitting these cases. Additional false alarms appeared on IT267 (2 scenarios), IT240 (2 scenarios), and E242 (1 scenario), although these were more likely due to their detectors being too sensitive.

Because QED-PC++ used the same residual for E265 as QED-PC, it also had problems related to false alarms with that residual. Because QED-PC++ also had the global model residuals to rely on, its performance is improved over QED-PC. It also had the best overall fault detection and isolation times.

10.3. Lessons Learned

In this subsection, we present some lessons learned from the application of our algorithms on a real system that highlighted some of the strengths and weaknesses of the algorithms. This helps to critically identify and evaluate the shortcomings of the algorithms and explore potential solutions to address these shortcomings and improve the diagnosis framework.

The diagnosis approach presented here offers many advantages. First and foremost, the approach is model-based. Given a global model of the system, most aspects are generated automatically: generation of submodels for residual generation and fault identification and generation of fault signatures and relative residual orderings. Some manual tuning of parameters is required for the fault detectors, symbol generators, and the consistency tests for fault identification. Fault isolation is fast, with the candidate set being reduced to a small number of candidates only a short time after fault detection. This limits the amount of computation that must be performed by fault identification to only these few candidates. Further, fault isolation is efficient through the use of local submodels.

Of course, all of these approaches are dependent on correct symbol generation for correct fault isolation, and this turns out to be the source of most of the errors. Sensor noise makes correct symbol generation difficult. For the submodel-based residuals, the effect of sensor noise is greater because (noisy) sensor values are used as local inputs. The use of the Z-test attempts to mitigate the effect of noise, however, additional hand-tuning is typically required. Even when hand-tuning is not required, symbols can still be incorrectly generated if the noise characteristics of the sensor change compared to the training data used. This, in fact, is what happened with the submodel-based residual for E265 with QED-PC and QED-PC++ in the validation data set. This was particularly problematic because for all the sensor fault scenarios (about half the total number of scenarios), this false alarm was present. As a result, the fault isolation was incorrect in many of these scenarios and this resulted in classification errors and recovery costs. Incorrectly generated slope symbols, and false alarms on other sensors, also caused some diagnostic errors.

In addition, for the submodel-based residuals in particular, some fault detection thresholds were too sensitive. The tuning of these thresholds is difficult, especially since both the presence of (i.e., through predicted fault signatures) and the absence of (i.e., through the isolation rules) residual deviations for fault isolation are used. That is, in some cases a change in a sensor needs to be detected in order to properly diagnose, and in other cases it is necessary to make sure the detector does not fire when there is no change in the sensor. Perhaps a more robust approach is to resolve ambiguities using only fault identification without relying on heuristic isolation rules, although this may increase fault isolation times.

All of these issues related to symbol generation motivate the need for a more robust qualitative fault isolation approach, in which there are probabilities associated with the correctness of generated symbols, and diagnostic reasoning can be performed considering the case where the symbol was incorrectly generated and the associated confidence in the generated symbols. Current work on that front is reported in [28].

Intermittent faults also introduced some issues. For one, intermittent faults detected as $++$ (or $--$) are not accounted for. Ideally, the symbol generators should have generated $+-$ (or $-+$) and $+0$ (or -0) symbols for every time intermittent faults surfaced, but this relies on a more explicit test for discontinuity detection. Second, the algorithms can benefit from a more robust intermittent fault identification approach.

Because QED-PC++ makes use of both the global model residuals and the submodel-based residuals, it combines the strengths of both residual sets. The first set is most useful for nonsensor faults, which affect many residuals and therefore give a lot of information for fault isolation, and the second set is most useful for sensor faults, which affect more residuals than nonsensor faults (for a good model decomposition) and therefore give a lot of information for fault isolation for those types of faults. As observed in both the training and validation data sets, faster fault detection and isolation is seen with the combined residual set. Using both residual sets also eliminated most of the heuristic fault isolation rules, reducing another potential source of diagnostic errors due to problems tuning the time periods used in those rules.

11. Related Work

A variety of other diagnosis approaches have also been applied to ADAPT. Only one other was tested on the same data set, so we try here to make a fair comparison with the available metrics.

An approach similar to ours is described in [29], which shares the same qualitative fault isolation methodology, but without residual orderings. A key difference is that the different fault profiles are isolated directly within the fault detectors. The correct fault was detected and isolated 60% of the time. Like our approach, tuning the fault detectors properly becomes critical, as errors in this step propagate to reasoning.

Component models similar to ours are used in [30], and ARRs are derived for a subset of ADAPT-Lite. Changes in residual values associated to ARRs are used to diagnose the system mode and faults that have occurred, and so must precompute ARRs for each mode of the system. Since only single faults are considered and all mode changes correspond to faults in ADAPT-Lite, really only a subset of these ARRs are needed. Only a limited comparison to a previous incarnation of QED was performed, finding that this approach had similar fault detection times for abrupt faults.

In [31], a model-based diagnosis approach implemented using the Testability Engineering and Maintenance System (TEAMS) [32], is applied. TEAMS is based on functional fault modeling, and computes, similar to ARRs, a dependency matrix of ones and zeros that stores which faults affect which test points. Unlike ARRs and PCs, the tests may not correspond directly to residuals and are constructed by hand. This approach was applied to the entire ADAPT system, but relative to QED had worse fault detection and isolation performance. This could be improved with better tests, but the advantage of an approach like QED is that the tests are defined automatically in a specific way (i.e., based on residuals). A similar approach to [31] was followed in [33], where manually created tests were constructed and fault isolation was performed based on the test results. Fault isolation performance was very accurate, but with a false positive rate of 25% and a false negative rate of 7%.

A classification approach using artificial immune systems was applied to ADAPT-Lite in [34]. Nominal data was used to train a set of detectors. The set of triggered detectors was then used to isolate the faults. The approach detects faults about six times slower than QED, but isolates faults about twice as fast. Despite this, the approach had about 50% more fault isolation errors than QED, but the number of errors was still relatively low.

An approach that diagnoses faults in the dc subsystem based on convex optimization is employed in [35]. Diagnostic metrics are not provided. In contrast to such purely quantitative approaches, the main idea of QED is to start first with a quick fault isolation step that decreases significantly the size of the remaining quantitative problem (i.e., fault identification). As systems increase in size, purely quantitative approaches become more difficult to apply.

In [36], conflict-driven fault detection and isolation is performed. The inference engine is a constraint solver that checks whether a solution exists, computing every solution that explains the available observations. In a similar approach, the Hybrid Diagnosis Engine (HyDE) [37] was applied to the full ADAPT system in [38]. HyDE is a model-based diagnosis engine which generates conflicts (and eventually fault candidates) by using consistency between the model predictions and the system observations. A HyDE model may use boolean, discrete, real, or interval-valued variables to describe the behavior of a system. In [38], a steady-state model was developed, and so was very slow in fault detection and isolation, yielding only a 24% detection accuracy when applied to ADAPT-Lite with the DXC data. HyDE was also applied to the same data set used in this paper, having significantly higher costs (2650 total), due to having almost five times as many diagnostic errors. Many of these errors are due to the use of a steady-state model, and one would expect HyDE's performance to improve significantly with the model used by QED.

Steady-state diagnosis was also performed in [39], resulting in static diagnosis problems that were solved with Reiter's approach [40]. Steady-state approaches have simple fault isolation algorithms, but at the cost of delayed fault detection. Further, some faults cannot be identified using only the steady states (e.g., drift faults).

Probabilistic diagnosis algorithms are used in [41], using discrete and static Bayesian network models that are compiled with arithmetic circuits, which derive marginal probabilities using addition and multiplication operations. The approach can handle multiple faults, and achieved 96% diagnosis accuracy on its test data. A problem with this approach is that the reasoning is so simple that much effort has to be placed into careful development and tuning of the Bayesian models. With an approach like QED, the model already has much embedded information for reasoning, so less effort is expended on fine-tuning it (the fine-tuning, which is relateively less effort, moves to the fault detectors and symbol generators).

12. Conclusions

In this work, we developed a model-based diagnosis architecture that combines qualitative fault isolation and quantitative fault identification. From the diagnostic framework, three alternative algorithms are instantiated, one which uses the global model of the system (QED), one which uses model decomposition to derive local submodels (QED-PC), and one which combines both the global model and local submodels (QED-PC++). The different approaches are applied to the ADAPT system, and several enhancements in the framework are developed, such as stuck fault detection, heuristic fault isolation rules, and intermittent fault diagnosis. These new enhancements can be easily applied to other model-based fault diagnosis approaches as well.

Overall, the algorithms did very well on the case study. All approaches had very good fault detection and isolation. For the training data set, fault isolation and identification was good enough that QED and QED-PC++ never neglected to command an *abort* when it was necessary, and QED-PC only missed an *abort* command twice (out of 227 experiments). For the validation data set, QED performed very well, but QED-PC and QED-PC++ suffered, because the noise characteristics of one sensor changed enough to cause a number of false alarms, leading to several fault isolation errors, and, as a result, incorrect recovery recommendations. Although this is an easy issue to resolve, it still motivates the need for a more robust approach to fault isolation that can handle occasional incorrect symbol generation.

In the future, these algorithms will be applied to larger systems, e.g., the complete ADAPT system. ADAPT is a hybrid system, and can have multiple faults. Hence, in the future, the current approaches will be extended to diagnosis of hybrid systems, and multiple fault diagnosis. Additionally, ongoing work is investigating more robust fault isolation frameworks.

References

[1] J. Chen, R. Patton, Robust model-based fault diagnosis for dynamic systems, Kluwer Academic Publishers, Norwell, MA, USA, 1999.

[2] R. Patton, P. Frank, R. Clark, Issues of Fault Diagnosis for Dynamic Systems, Springer, 2000.

[3] R. Isermann, Supervision, Fault-Detection and Fault-Diagnosis Methods - An Introduction, Ctrl. Eng. Practice 5 (5) (1997) 639–652.

[4] J. Gertler, Fault Detection and Diagnosis in Engineering Systems, Marcel Dekker, Inc., 1998.

[5] P. Goupil, Oscillatory failure case detection in the A380 electrical flight control system by analytical redundancy, Control Engineering Practice 18 (9) (2010) 1110 – 1119.

[6] P. Goupil, Airbus state of the art and practices on FDI and FTC in flight control system, Control Engineering Practice 19 (6) (2011) 524 – 539.

[7] S. Poll *et al.*, Evaluation, selection, and application of model-based diagnosis tools and approaches, in: AIAA Infotech@Aerospace 2007 Conference and Exhibit, 2007.

[8] T. Kurtoglu, S. Narasimhan, S. Poll, D. Garcia, L. Kuhn, J. de Kleer, A. van Gemund, A. Feldman, First international diagnosis competition – DXC'09, in: Proceedings of 20th International Workshop on Principles of Diagnosis, 2009, pp. 383–396.

[9] A. Sweet, A. Feldman, S. Narasimhan, M. Daigle, S. Poll, Fourth international diagnostic competition – DXC'13, in: Proc. of the 24th Intl. Workshop on Principles of Diagnosis, 2013, pp. 224–229.

[10] P. J. Mosterman, G. Biswas, Diagnosis of continuous valued systems in transient operating regions, IEEE Trans. on Systems, Man and Cybernetics, Part A 29 (6) (1999) 554–565.

[11] M. J. Daigle, X. Koutsoukos, G. Biswas, A qualitative event-based approach to continuous systems diagnosis, IEEE Trans. on Control Systems Technology 17 (4) (2009) 780–793.

[12] H. Thompson, Parallel processing architectures for aerospace applications, Ctrl. Engineering Practice 2 (3) (1994) 509 – 520.

[13] A. Bregon, M. Daigle, I. Roychoudhury, G. Biswas, X. Koutsoukos, B. Pulido, An event-based distributed diagnosis framework using structural model decomposition, Artificial Intelligence 210 (2014) 1–35. doi:http://dx.doi.org/10.1016/j.artint.2014.01.003.

[14] S. Arogeti, D. Wang, C. B. Low, M. Yu, Fault detection isolation and estimation in a vehicle steering system, IEEE Transactions on Industrial Electronics 59 (12) (2012) 4810–4820. doi:10.1109/TIE.2012.2183835.

[15] M. Fravolini, G. Campa, Design of robust redundancy relations for a semi-scale yf-22 aircraft model, Control Engineering Practice 17 (7) (2009) 773 – 786.

[16] B. Pulido, C. Alonso-González, Possible conflicts: a compilation technique for consistency-based diagnosis, IEEE Trans. on Systems, Man, and Cybernetics, Part B 34 (5) (2004) 2192–2206.

[17] S. Arogeti, D. Wang, C. B. Low, Mode identification of hybrid systems in the presence of fault, IEEE Transactions on Industrial Electronics 57 (4) (2010) 1452–1467. doi:10.1109/TIE.2009.2030213.

[18] A. Bregon, G. Biswas, B. Pulido, A decomposition method for nonlinear parameter estimation in TRANSCEND, IEEE Trans. Syst. Man. Cy. Part A 42 (3) (2012) 751–763.

[19] M. Daigle, A. Bregon, I. Roychoudhury, Qualitative event-based diagnosis with possible conflicts applied to spacecraft power distribution systems, in: Proceedings of the 8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, 2012, pp. 265–270.

[20] S. Poll, A. Patterson-Hine, J. Camisa, D. Garcia, D. Hall, C. Lee, O. Mengshoel, C. Neukom, D. Nishikawa, J. Ossenfort, A. Sweet, S. Yentus, I. Roychoudhury, M. Daigle, G. Biswas, X. Koutsoukos, Advanced diagnostics and prognostics testbed, in: Proceedings of the 18th International Workshop on Principles of Diagnosis, 2007, pp. 178–185.

[21] I. Roychoudhury, M. Daigle, A. Bregon, B. Pulido, A Structural Model Decomposition Framework for Systems Health Management, in: Proceedings of the 2013 IEEE Aerospace Conference, 2013.

[22] M. Ceraolo, New dynamical models of lead-acid batteries, IEEE Trans. on Power Systems 15 (4) (2000) 1184–1190.

[23] G. Biswas, G. Simon, N. Mahadevan, S. Narasimhan, J. Ramirez, G. Karsai, A robust method for hybrid diagnosis of complex systems, in: Proceedings of the 5th Symposium on Fault Detection, Supervision and Safety for Technical Processes, 2003, pp. 1125–1131.

[24] M. Daigle, I. Roychoudhury, G. Biswas, X. Koutsoukos, A. Patterson-Hine, , S. Poll, A comprehensive diagnosis methodology for complex hybrid systems: A case study on spacecraft power distribution systems, IEEE Transactions of Systems, Man, and Cybernetics, Part A 4 (5) (2010) 917–931.

[25] C. H. Lo, Y. Wong, A. Rad, Intelligent system for process supervision and fault diagnosis in dynamic physical systems, IEEE Trans. on Industrial Electronics 53 (2) (2006) 581–592. doi:10.1109/TIE.2006.870707.

[26] M. Cordier, P. Dague, F. Lévy, J. Montmain, M. Staroswiecki, L. Travé-Massuyès, Conflicts versus Analytical Redundancy Relations: a comparative analysis of the Model-based Diagnosis approach from the Artificial Intelligence and Automatic Control perspectives, IEEE Trans. on Systems, Man, and Cybernetics. Part B: Cybernetics 34 (5) (2004) 2163–2177.

[27] J. de Kleer, J. Kurien, Fundamentals of model-based diagnosis, in: Proceedings of the 5th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes, Washington D.C., USA, 2003, pp. 25–36.

[28] M. Daigle, I. Roychoudhury, A. Bregon, Qualitative event-based fault isolation under uncertain observations, in: Annual Conference of the Prognostics and Health Management Society 2014, 2014, pp. 347–355.

[29] J. D. Carl, D. L. C. Mack, A. Tantawy, G. Biswas, X. D. Koutsoukos, Fault isolation for spacecraft systems: An application to a power distribution testbed, in: Proceedings of the 8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, Mexico City, Mexico, 2012, pp. 168–173.

[30] M. Maiga, E. Chanthery, L. Trave-Massuyes, Hybrid system diagnosis: Test of the diagnoser hydiag on a benchmark of the international diagnostic competition dxc 2011, in: Proceedings of the 8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, SAFEPROCESS12, Mexico City, Mexico, 2012, pp. 271–276.

[31] M. Wilson, T. Kurtoglu, TRT4ADAPT - a model-based algorithm for the second international diagnostic competition, in: Proc. of the 21st International Workshop on Principles of Diagnosis, Portland, OR, 2010, pp. 387–394.

[32] K. Pattipati, V. Raghavan, M. Shakeri, S. Deb, R. Shrestha, Teams: testability engineering and maintenance system, in: American Control Conference, 1994, Vol. 2, 1994, pp. 1989–1995.

[33] E. Almqvist, D. Eriksson, A. Lundberg, E. Nilsson, N. Wahlstrom, E. Frisk, M. Krysander, Solving the ADAPT benchmark problem: A student project study, in: Proceedings of the 21st International Workshop on Principles of Diagnosis, Portland, OR, 2010, pp. 153–160.

[34] J. Mange, D. Daniszewski, , A. Dunn, Artificial immune systems for diagnostic classification problems, in: Proc. of the 22nd International Workshop on Principles of Diagnosis, Murnau, Germany, 2011, pp. 279–284.

[35] D. Gorinevsky, S. Boyd, S. Poll, Estimation of faults in dc electrical power system, in: American Control Conference, 2009, pp. 4334–4339.

[36] P. Bunus, O. Isaksson, B. Frey, B. Munker, RODON: A model-based diagnosis approach for the DX diagnostic competition, in: Proceedings of the 20th International Workshop on Principles of Diagnosis, Stockholm, Sweden, 2009, pp. 423–430.

[37] S. Narasimhan, L. Brownston, HyDE–a general framework for stochastic and hybrid model-based diagnosis, Proc. of the 18th Intl. Workshop on Principles of Diagnosis (2007) 162–169.

[38] A. Sweet, Testing HyDE on ADAPT, Tech. Rep. NASA/TM 2008-214570, NASA Ames Research Center, Moffet Field, CA, USA (2008).

[39] A. Grastien, P. Kan-John, Wizards of oz: Description of the 2009 DXC entry, in: Proceedings of the 20th International Workshop on Principles of Diagnosis, Stockholm, Sweden, 2009, pp. 409–413.

[40] R. Reiter, A theory of diagnosis from first principles, Artificial intelligence 32 (1) (1987) 57–95.

[41] B. Ricks, O. Mengshoel, The diagnostic challenge competition: Probabilistic techniques for fault diagnosis in electrical power systems, in: Proceedings of the 20th International Workshop on Principles of Diagnosis, Stockholm, Sweden, 2009, pp. 415–422.