Integrated Diagnostics and Prognostics for the Electrical Power System of a Planetary Rover

Matthew Daigle[1], Indranil Roychoudhury[2], and Anibal Bregon[3]

[1] *NASA Ames Research Center, Moffett Field, California, 94035, USA*
matthew.j.daigle@nasa.gov

[2] *SGT Inc., NASA Ames Research Center, Moffett Field, California, 94035, USA*
indranil.roychoudhury@nasa.gov

[3] *Department of Computer Science, University of Valladolid, Valladolid, Spain*
anibal@infor.uva.es

Abstract

For electric vehicles, technology for monitoring, diagnosis, and prognosis of the electrical power system (EPS) becomes essential for safe and efficient operation. To this end, we develop a general system-level integrated diagnosis and prognosis framework, which detects, isolates, and identifies EPS faults, and predicts when the EPS will fail to deliver sufficient power. The approach takes advantage of recent work in structural model decomposition in order to distribute the global diagnosis and prognosis problems into local subproblems that can be solved in parallel, thus enabling implementation on distributed computational platforms. The framework is applied to the EPS of a planetary rover testbed, and is demonstrated using data from field experiments.

1. Introduction

For electric vehicles, technology for monitoring, diagnosis, and prognosis of the electrical power system (EPS) is critical. In order to ensure safety, algorithms are needed that are able to predict the end-of-discharge (EOD) of the batteries powering the vehicle. The EOD time depends both on the current state of the batteries, including state-of-charge (SOC), and the future power requirements of the batteries. The future power requirements for the batteries depend both on the power required for future vehicle maneuvers and on any fault present in the system, which may cause increases in power demands. Therefore, both diagnosis (determining the current system state and faults) and prognosis (predicting the EOD of the system) are required.

A large body of research exists for both model-based diagnosis (Gertler, 1998; Blanke et al., 2006) and prognosis methods (Luo et al., 2008; Saha & Goebel, 2009; Orchard & Vachtsevanos, 2009), however, most of the approaches in the literature focus in either solely the diagnosis or the prognosis task. A few works have proposed the integration of both tasks within a common framework (Patrick et al., 2007; Orchard & Vachtsevanos, 2009; Roychoudhury & Daigle, 2011; Zabi et al., 2013), however, unlike our approach, these approaches perform the diagnosis and prognosis tasks in a centralized way, thus suffering from scalability issues due to the large number of states and parameters in real-world systems. Moreover, most solutions do not approach the system-level problem. To the best of our knowledge, there is no approach in the literature which combines, in a distributed fashion, the system-level diagnosis and prognosis tasks.

In previous work, we have developed an integrated model-based diagnosis and prognosis framework (Roychoudhury & Daigle, 2011). The main contribution of this work was a unified modeling framework. In an extension of this work, we used structural model decomposition to develop a distributed integrated diagnosis and prognosis framework (Bregon, Daigle, & Roychoudhury, 2012), based on other work in distributed diagnosis (Bregon et al., 2014) and distributed prognosis (Daigle, Bregon, & Roychoudhury, 2012, 2014). Through structural model decomposition, a global model is transformed into a set of local submodels. For model-based diagnosis and prognosis, this results in the global diagnosis and prognosis problems being transformed into local diagnosis and prognosis subproblems. These subproblems can be solved independently by assigning them to different processing units, thus enabling a scalable and computationally efficient distributed diagnosis and prognosis solution.

In this paper, we apply these frameworks and ideas to the EPS of a planetary rover testbed at NASA Ames Research Center (Balaban et al., 2013). The applied architecture constitutes a new framework for integrated *system-level* diagnosis and prognosis. For the rover, we are interested in a system-level prediction, that is, when the EPS can no longer supply sufficient power to the loads. The rover is powered by several batteries, and this condition is a function of the state of all the batteries. Hence, component-level prognostics algorithms cannot be used, and a system-level prognosis framework is required (Daigle, Bregon, & Roychoudhury, 2012). We utilize recent work in structural model decomposition (Roychoudhury, Daigle, Bregon, & Pulido, 2013) to achieve a distributed implementation of the framework. We demonstrate the complete approach using real experimental data from the rover operating in the field.

The paper is organized as follows. Section 2 formulates the system-level diagnosis and prognostics problems. Section 3 describes the background on structural model decomposition, distributed diagnosis, and distributed diagnosis. Section 4 presents the rover EPS case study. Sections 5 and 6 present the system-level diagnosis and prognostics solutions, respectively, for the rover EPS. Section 7 presents the results for different scenarios. Finally, Section 8 concludes the paper.

2. PROBLEM FORMULATION

In this section, we formulate the integrated system-level diagnosis and prognosis problem. Ultimately, the goal is to predict when some event occurs in the system, such as the rover running out of power. In order to make such a prediction, we need to know the state of the system, including any faults that are present, therefore, diagnosis is needed in order to perform prognosis. We first formulate the system-level diagnosis problem, followed by the system-level prognosis problem.

2.1. System-Level Diagnosis

The problem of system-level diagnosis consists of three parts: (*i*) detecting whether a fault is present, (*ii*) isolating the correct fault, and (*iii*) identifying the faulty system state. In each of these parts, different models may be used. We assume that a model $\mathcal{M}$ can be succinctly represented in the following general formulation:

$$\mathbf{x}(k+1) = \mathbf{f}(k, \mathbf{x}(k), \boldsymbol{\theta}(k), \mathbf{u}(k), \mathbf{v}(k)), \quad (1)$$

$$\mathbf{y}(k) = \mathbf{h}(k, \mathbf{x}(k), \boldsymbol{\theta}(k), \mathbf{u}(k), \mathbf{n}(k)), \quad (2)$$

where k is the discrete time variable, $\mathbf{x}(k) \in \mathbb{R}^{n_x}$ is the state vector, $\boldsymbol{\theta}(k) \in \mathbb{R}^{n_\theta}$ is the unknown parameter vector, $\mathbf{u}(k) \in \mathbb{R}^{n_u}$ is the input vector, $\mathbf{v}(k) \in \mathbb{R}^{n_v}$ is the process noise vector, $\mathbf{f}$ is the state equation, $\mathbf{y}(k) \in \mathbb{R}^{n_y}$ is the output vector, $\mathbf{n}(k) \in \mathbb{R}^{n_n}$ is the measurement noise vector, and $\mathbf{h}$ is the output equation.[1] We will describe in Section 3 an equivalent structural representation of a model $\mathcal{M}$ that will be used for structural model decomposition.

In the model-based paradigm, we assume that in the nominal (fault-free) case, the system behaves according to some model $\mathcal{M}_n$, and, given the inputs $\mathbf{u}(k)$, produces measured outputs $\mathbf{y}(k)$. The problem of fault detection is to determine when model-predicted (nominal) outputs $\hat{\mathbf{y}}_n(k)$ are different from the measured outputs $\mathbf{y}(k)$ in a statistically significant manner. The difference $\mathbf{y}(k) - \hat{\mathbf{y}}_n(k)$ is called a *residual*; a (statistically significant) nonzero residual indicates a fault.

Faults are generally represented as changes in the model (i.e., in parameter values and/or model structure). So, in general, each fault $f \in F$, where F is the complete set of potential faults, is represented as a new model, $\mathcal{M}_f$. Given that a fault is present, the problem of fault isolation is to determine which model $\mathcal{M}_f$ now represents the system. The problem of fault identification is to determine the fault parameter estimate for the isolated fault, $p(\boldsymbol{\theta}_f(k)|\mathbf{y}(k_0\!:\!k))$, where $\mathbf{y}(k_0\!:\!k)$ denotes all measurements observed from the initial time k_0 to the current time k.

2.2. System-Level Prognosis

Rather than being focused on individual components, system-level prognostics is focused on the system as a whole, and on predictions for the system. As such, it is a more general formulation of the prognostics problem. System-level prognostics was previously defined in (Daigle, Bregon, & Roychoudhury, 2012). Here, we generalize the problem formulation based on (Daigle & Kulkarni, 2014) and explicitly integrate it with the diagnosis problem. Specifically, predictions must be made for a given fault hypothesis, which consists of a fault model $\mathcal{M}_f$ and joint state-parameter estimate $p(\mathbf{x}_f(k), \boldsymbol{\theta}_f(k)|\mathbf{y}(k_0\!:\!k))$. Fault identification computes an estimate of $\boldsymbol{\theta}_f(k)$, and the initial step of prognostics is to compute the full joint-state parameter estimate for the new faulty model.

System-level prognostics is concerned with predicting the occurrence of some system-level event E that is defined with respect to the states, parameters, and inputs of the system. We define the event as the earliest instant that some event threshold function $T_{E_f} : \mathbb{R}^{n_{x_f}} \times \mathbb{R}^{n_{\theta_f}} \times \mathbb{R}^{n_u} \rightarrow \mathbb{B}$, where $\mathbb{B} \triangleq \{0, 1\}$ changes from the value 0 to 1. That is, the time of the event k_{E_f} at some time of prediction k_P given some fault f is defined as

$$k_{E_f}(k_P) \triangleq \inf\{k \in \mathbb{N} \colon k \geq k_P \wedge T_{E_f}(\mathbf{x}_f(k), \boldsymbol{\theta}_f(k), \mathbf{u}(k)) = 1\}. \quad (3)$$

[1]Bold typeface denotes vectors, and n_a denotes the length of a vector $\mathbf{a}$.

The time remaining until that event, Δk_{E_f}, is defined as

$$\Delta k_{E_f}(k_P) \triangleq k_{E_f}(k_P) - k_P. \qquad (4)$$

The prognostics problem is inherently uncertain, due to the random nature of the system evolution (represented with $\mathbf{v}(k)$), and unknown future inputs ($\mathbf{u}(k)$ for $k > k_P$). Therefore, k_{E_f} and Δk_{E_f} are random variables, and we must compute the probability distribution $p(k_{E_f}(k_P)|\mathbf{y}(k_0{:}k_P))$ (Daigle, Saxena, & Goebel, 2012; Sankararaman, Daigle, Saxena, & Goebel, 2013; Sankararaman, Daigle, & Goebel, 2014).

3. BACKGROUND

For a large system, both the diagnosis and prognosis problems are correspondingly large. A centralized approach does not scale well, can be computationally expensive, and prone to single points of failure. Therefore, we propose to decompose the *global* diagnosis and prognosis problems into independent *local* subproblems. In this work, we build on the ideas from structural model decomposition (Blanke et al., 2006; Pulido & Alonso-González, 2004) to compute local independent subproblems, which may be solved in parallel, thus providing scalability and efficiency.

We adopt here the structural model decomposition framework described in (Roychoudhury et al., 2013). This approach allows us to make guarantees of the minimality of the derived submodels and allows to generate different submodels for each one of the diagnosis and prognosis tasks. In the following, we review the main details and refer the reader to (Roychoudhury et al., 2013) for additional explanation. We define a model as follows:

Definition 1 (Model)**.** A *model* $\mathcal{M}^*$ is a tuple $\mathcal{M}^* = (V, C)$, where V is a set of variables, and C is a set of constraints among variables in V. 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 $c = (\varepsilon_c, V_c)$, such that $c \in C$, consists of an equation ε_c involving variables $V_c \subseteq V$.

Input variables, U, are known, and the set of output variables, Y, correspond to the (measured) sensor signals. Parameters, Θ, include explicit model parameters that are used in the model constraints. Auxiliary variables, A, are additional variables that are algebraically related to the state and parameter variables, and are used to reduce the structural complexity of the equations.

The notion of a *causal assignment* is used to specify the computational causality for a constraint c, by defining which $v \in V_c$ is the dependent variable in equation ε_c.

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 ε_c.

We write a causal assignment of a constraint using its equation in a causal form, with $:=$ to explicitly denote the causal (i.e., computational) direction.

Definition 3 (Valid Causal Assignments)**.** We say that a set of causal assignments $\mathcal{A}$, for a model $\mathcal{M}^*$ is *valid* if

- For all $v \in U \cup \Theta$, $\mathcal{A}$ does not contain any α such that $\alpha = (c, v)$.
- For all $v \in Y$, $\mathcal{A}$ does not contain any $\alpha = (c, v_c^{out})$ where $v \in V_c - \{v_c^{out}\}$.
- For all $v \in V - U - \Theta$, $\mathcal{A}$ contains exactly one $\alpha = (c, v)$.

The definition of valid causal assignments states that (*i*) input or parameter variables cannot be the dependent variables in the causal assignment, (*ii*) a measured variable cannot be used as an independent variable in any constraint, and (*iii*) every variable, which is not input or parameter, is computed by only one (causal) constraint.

Based on this, a *causal model* is a model extended with a valid set of causal assignments.

Definition 4 (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.

3.1. Structural Model Decomposition

To decompose a model into submodels, we need to break internal variable dependencies. We do this by selecting certain variables as local inputs. Given the set of potential local inputs (in general, selected from V), and the set of variables to be computed by the submodel (selected from $V - U - \Theta$), we create from a causal model $\mathcal{M}$ a causal submodel $\mathcal{M}_i$, in which a subset of the variables in V are computed using a subset of the constraints in C. In this way, each submodel computes independently from all other submodels. A causal submodel can be defined as follows.

Definition 5 (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 $\mathcal{A}_i \cap \mathcal{A} \neq \varnothing$.

When using measurements (from Y) as local inputs, the causality of these constraints must be reversed, and so, in general, $\mathcal{A}_i$ is not a subset of $\mathcal{A}$.

The procedure for generating a submodel from a causal model is given as Algorithm 1 (`GenerateSubmodel`) in (Roychoudhury et al., 2013). Given a causal model $\mathcal{M}$, a set of variables that are considered as local inputs, U^*, and a set of variables to be computed, V^*, the `GenerateSubmodel` algorithm derives a causal submodel $\mathcal{M}_i$ that computes V^* using U^*. The algorithm works by starting at the variables in V^*, and propagating backwards through the causal dependencies. Propagation along a dependency chain stops once a variable in U^* is reached, or once a constraint is reached in which the causality can be reversed so that a variable in U^* can become a local input. We refer

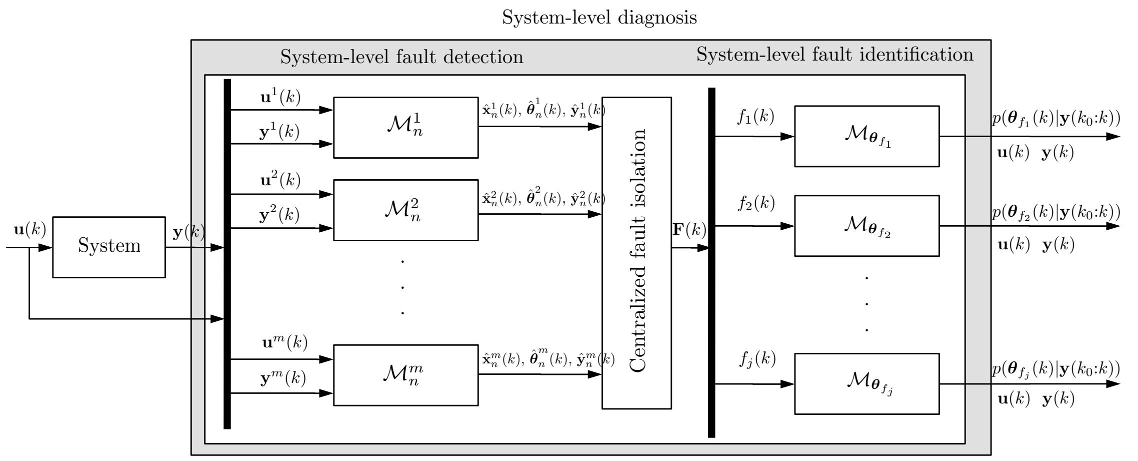

Figure 1. System-level diagnosis architecture.

the reader to (Roychoudhury et al., 2013) for the algorithm and additional details.

3.1.1. Structural Model Decomposition for System-Level Diagnosis

In this work, we use model decomposition to simplify the fault detection and fault identification problems (Bregon, Biswas, & Pulido, 2012; Bregon, Daigle, & Roychoudhury, 2012). For fault detection, we compute a set of residuals based on the sensors, and so derive a set of minimal local submodels to compute the nominal values of these sensors, i.e., one submodel for each $y \in Y$. In the submodel computing the output y, we use the other sensors $Y - \{y\}$ as local inputs, thus allowing decomposition. So, given the nominal model $\mathcal{M}_n$, for each output $y \in Y$, we create a submodel with $V^* = \{y\}$ and $U^* = \{U \cup (Y - \{y\})\}$.

Fault identification requires estimating a set of parameters associated with faults. Here, we also add Y as local inputs. Given a fault model $\mathcal{M}_f$, we create a submodel with $V^* = \boldsymbol{\theta}_f$, where $\boldsymbol{\theta}_f$ denotes the set of fault parameters, and $U^* = U \cup Y$.

3.1.2. Structural Model Decomposition for System-Level Prognosis

Prediction requires determining k_{E_f} for a given fault hypothesis f, which is computed based on T_{E_f}, which, in turn, is a function of the system states, parameters, and inputs. Often, the system-level, global threshold T_{E_f} can be expressed as the logical or of other local thresholds, i.e., $T_{E_f} = T_{E_f^1} \vee T_{E_f^2} \vee \ldots \vee T_{E_f^n}$ for n conditions. With each local threshold $T_{E_f^i}$ we can associate a local event E_f^i and compute times $k_{E_f^i}$, such that k_{E_f} can now also be defined as $min(k_{E_f^1}, k_{E_f^2}, \ldots, k_{E_f^n})$. This leads to a natural decomposition where each $k_{E_f^i}$ is computed independently, and allows us to decompose the prediction problem. So, to create the prediction submodels, we use the `GenerateSubmodel` algorithm in (Roychoudhury et al., 2013) with U^* set to $\{U_P\}$ and V^* set to $\{k_{E_f^i}\}$ for each local threshold $T_{E_f^i}$, where $U_P \subseteq V$ is the set of variables that can be predicted a priori.

The decomposition that can be achieved depends also on the selected U_P. If no variables exist that can be predicted a priori outside of U, then the `GenerateSubmodel` algorithm may not result in any decomposition and it will suffice to simply use the global model.

The initial state needed for prediction can be generated from a set of local estimators. The global prediction model is decomposed into local state estimators for the needed states, in the same way as in estimation for diagnosis.

3.2. Integrated System-level Diagnosis and Prognostics Architecture

Figs. 1 and 2 illustrate the architecture for our system-level diagnosis and prognosis frameworks, respectively. Regarding system-level diagnosis (Fig. 1), at each discrete time step, k, the system takes as input $\mathbf{u}(k)$ and produces outputs $\mathbf{y}(k)$. These are split into local inputs $\mathbf{u}^i(k)$ and local outputs $\mathbf{y}^i(k)$ for each one of the m system-level fault detection submodels, $\mathcal{M}_n^i$. Within each submodel $\mathcal{M}_n^i$, nominal tracking is performed, computing estimates of nominal states, $\hat{\mathbf{x}}_n^i(k)$, parameters, $\hat{\boldsymbol{\theta}}_n^i(k)$, and the measurements, $\hat{\mathbf{y}}_n^i(k)$. The fault isolator performs detection first by comparing the estimated measurement values against the observed values, to determine statistically significant deviations for the residual, $\mathbf{r}^i(k) = \mathbf{y}^i(k) - \hat{\mathbf{y}}^i(k)$. Deviations in the residuals are then transformed to qualitative symbols used by the centralized fault isolation block to generate a set of isolated fault candidates, $\mathbf{F}(k)$. For each one of the isolated fault candidates, $f_i(k)$, local models for fault identification, $\mathcal{M}_{\boldsymbol{\theta}_{f_i}}$, are used to compute local parameter estimates $p(\boldsymbol{\theta}_{f_i}(k)|\mathbf{y}(k_0{:}k))$. These

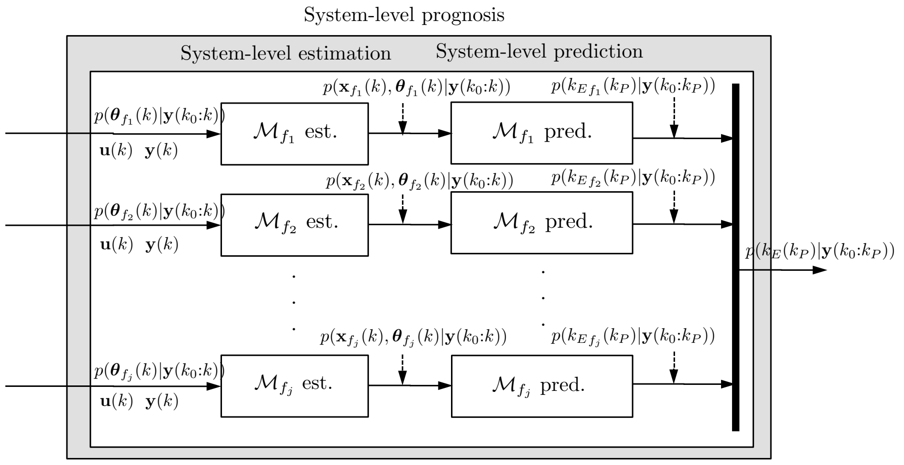
System-level prognosis
System-level estimation
System-level prediction
$p(\boldsymbol{\theta}_{f_1}(k)|\mathbf{y}(k_0{:}k))$
$\mathbf{u}(k)$ $\mathbf{y}(k)$
$\mathcal{M}_{f_1}$ est.
$p(\mathbf{x}_{f_1}(k),\boldsymbol{\theta}_{f_1}(k)|\mathbf{y}(k_0{:}k))$
$\mathcal{M}_{f_1}$ pred.
$p(k_{E_{f_1}}(k_P)|\mathbf{y}(k_0{:}k_P))$
$\mathcal{M}_{f_2}$ est.
$\mathcal{M}_{f_2}$ pred.
$\mathcal{M}_{f_j}$ est.
$\mathcal{M}_{f_j}$ pred.
$p(k_E(k_P)|\mathbf{y}(k_0{:}k_P))$

Figure 2. System-level prognosis architecture.

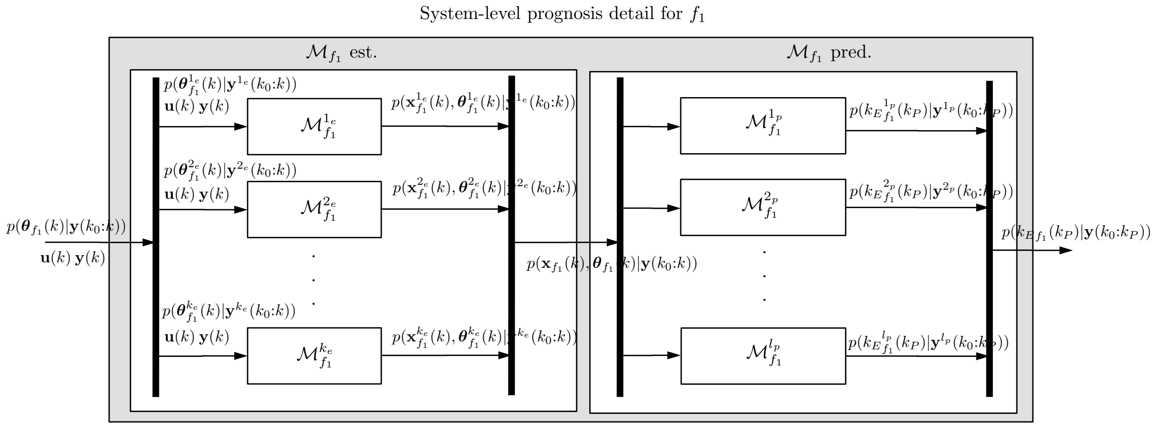
System-level prognosis detail for f_1
$\mathcal{M}_{f_1}$ est.
$\mathcal{M}_{f_1}$ pred.
$p(\boldsymbol{\theta}_{f_1}(k)|\mathbf{y}(k_0{:}k))$
$\mathbf{u}(k)\,\mathbf{y}(k)$
$\mathcal{M}^{1_e}_{f_1}$
$\mathcal{M}^{2_e}_{f_1}$
$\mathcal{M}^{k_e}_{f_1}$
$\mathcal{M}^{1_p}_{f_1}$
$\mathcal{M}^{2_p}_{f_1}$
$\mathcal{M}^{l_p}_{f_1}$
$p(\mathbf{x}_{f_1}(k),\boldsymbol{\theta}_{f_1}(k)|\mathbf{y}(k_0{:}k))$
$p(k_{E_{f_1}}(k_P)|\mathbf{y}(k_0{:}k_P))$

Figure 3. Detail of the system-level prognosis architecture.

local parameter estimates are then used as input to system-level prognosis (Fig. 2).

The system-level prognosis block of the architecture is divided into two phases: system-level estimation and system-level prediction. Parameter estimates from the local fault identification blocks, together with the inputs and outputs of the system, are used as input for the local estimation blocks, $\mathcal{M}_{f_i}$ *est.*, to compute state-parameter estimates $p(\mathbf{x}_{f_i}(k), \boldsymbol{\theta}_{f_i}(k)|\mathbf{y}(k_0{:}k))$. Finally, the local state-parameter estimates are used as input to the system-level prediction blocks, $\mathcal{M}_{f_i}$ *pred.*, to compute predictions, $p(k_{E\,f_i}(k_P)|\mathbf{y}(k_0{:}k_P))$, at given prediction time k_P. Predictions for each fault hypothesis are combined into the global prediction $p(k_E(k_P)|\mathbf{y}(k_0{:}k_P))$.

Fig. 3 shows the detail of the system-level estimation and prediction blocks for fault f_1, namely $\mathcal{M}_{f_1}$ *est.* and $\mathcal{M}_{f_1}$ *pred.* The system-level estimation task is decomposed using local estimation submodels, $\mathcal{M}^{1e}_{f_1}$ to $\mathcal{M}^{ke}_{f_1}$. As shown in the figure, subsets of the the local parameter estimates $p(\boldsymbol{\theta}_{f_1}(k)|\mathbf{y}(k_0{:}k))$, the system inputs, $\mathbf{u}(k)$, and the system outputs, $\mathbf{y}(k)$, are used as input for each one of the local state-parameter estimation submodels (this, of course, is similar to the estimation problem using the nominal model in the diagnosis part). The output of all the local submodels is then combined to compute the local state-parameter estimate for fault f_1, $p(\mathbf{x}_{f_1}(k), \boldsymbol{\theta}_{f_1}(k)|\mathbf{y}(k_0{:}k))$. The system-level prediction problem is also decomposed using local prediction submodels. The state estimate for the fault is split into local estimates for the prediction submodels, which then each compute a local $k^i_{E_{f_1}}$ value; these are then merged into the system-level prediction $k_{E_{f_1}}$ for the fault.

4. ROVER EPS MODELING

We are interested in integrated diagnosis and prognosis of the EPS of the rover. Thus, our system under consideration consists of the batteries, the battery current sensor, and the voltage sensors. The rover motors, which produce the electrical loads experienced by the EPS, are considered outside of our system under consideration, and so the loads the motors demand are viewed as inputs to the EPS.

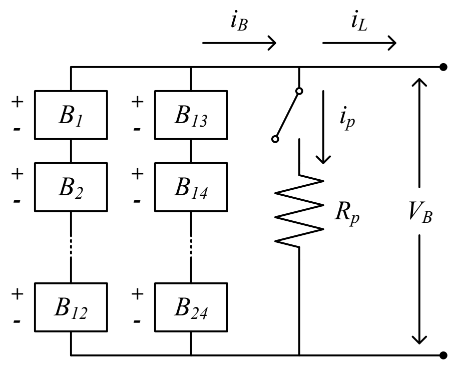

Figure 4. Rover EPS schematic.

The circuit schematic for the rover EPS is shown as Fig. 4. There are 24 lithium-ion cells in total, with two parallel branches of 12 cells in series. In parallel is a parasitic load, modeled as a resistance, R_p, that may appear as a fault. The battery current, i_B, is split into the current going to the load, i_L, and the current going to the parasitic load (if present), i_p. The total voltage provided by the EPS to the load is denoted as V_B. The cell model computes the voltage as a function of time given the current drawn from the cell, and is described in detail in (Daigle & Kulkarni, 2013). For completeness, the model is summarized in the appendix, and we refer the reader to (Daigle & Kulkarni, 2013) for additional explanation.

We assume that all cells start fully charged, so the voltage over each parallel branch is the same, and the current is split evenly ($i_B/2$). As the cells discharge, the total voltages must stay balanced, since the two sets of cells are in parallel, and therefore the current into each branch remains $i_B/2$.

The causal graph corresponding to the EPS model is shown in Fig. 5. The boxes in the figure indicate the battery cell models (for brevity, the internal variables are not shown). Also indicated are the sensor models. A measured value y^* (the * superscript indicates the measured value of a physical variable y) is equal to the physical variable y plus a bias, indicated with the ^{b} superscript. The biases, when present, produce a constant offset to the true value. Here, it also makes clear that we use the measured value of the load current, i_L^*, as an input to the system, which we assume is faultless.

The causal graph also indicates the computation of the time k_E (in the following, and in the figures, we drop the f subscript, as these submodels are not specific to a given fault). For the rover, E corresponds to any of the batteries reaching end-of-discharge (EOD), which is what must be predicted. EOD is defined by a voltage threshold V_{EOD}, where T_E is defined by $V_1 < V_{EOD}$ or $V_2 < V_{EOD}$, ..., $V_{24} < V_{EOD}$. When any cell voltage is less than V_{EOD}, EOD is reached for that battery and T_E evaluates to 1. The rover cannot be used beyond that point, as it will damage any batteries whose voltage is below the cutoff voltage.

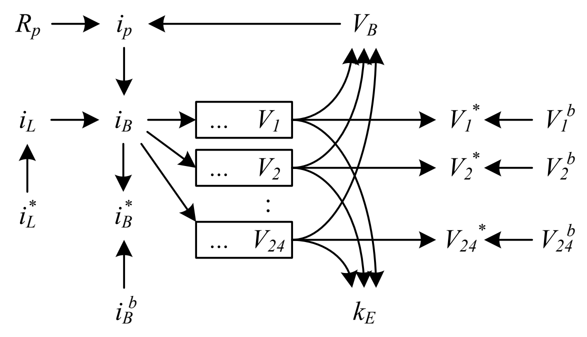

Figure 5. Causal graph for rover EPS model.

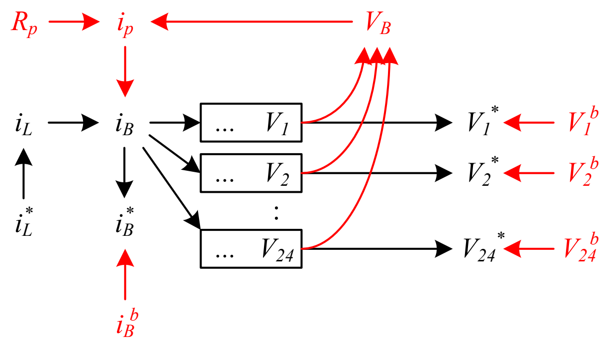

Figure 6. Causal graph for global nominal model.

5. ROVER EPS DIAGNOSIS

As described in Section 2, for diagnosis, models are used for the three phases of the diagnosis process: (*i*) fault detection, consisting of state estimation and residual generation, (*ii*) fault isolation, and (*iii*) fault identification. We describe the models used for each in the following subsections.

5.1. Fault Detection

Recall that in order to detect faults, we produce residuals, for which we need to compute model-predicted values of the outputs. We denote a residual using r_{y^*}, where y^* is the variable name for the sensor output. The causal graph for the global model for residual generation is shown as Fig. 6. It is generated by calling `GenerateSubmodel` with $U^* = \{i_L^*\}$, and $V^* = \{V_1^*, V_2^*, \ldots, V_{24}^*, i_B^*\}$. For residual generation, only the nominal model is needed, because the aim is only to detect when the nominal model is no longer valid, due to the appearance of a fault. In Fig. 6, the nominal parts of the model are colored black, and the fault-related parts in red. Since the faults are free from the nominal version of the model, only the black portion is needed for residual generation. We retain the red parts in the figures to indicate that the measured values will be causally effected by the faults.

As described in Section 3, we can decompose the residual generation problem, by creating local models for each sen-

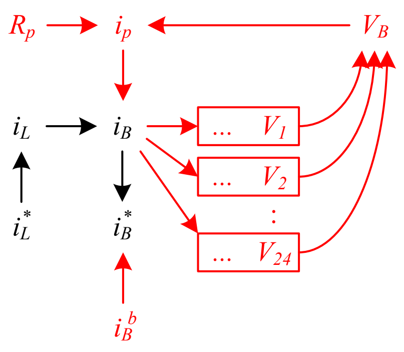

Figure 7. Causal graph for local i_B^* residual generator.

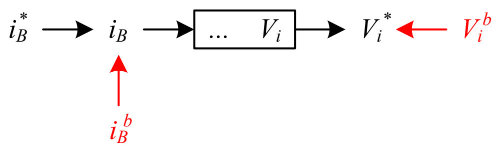

Figure 8. Causal graph for local V_i^* residual generator.

sor to compute predicted values. The causal graph for the local model for i_B^* is shown in Fig. 7, and is generated by calling `GenerateSubmodel` on the global model with $U^* = \{i_L^*, V_1^*, V_2^*, \ldots, V_{24}^*\}$ and $V^* = \{i_B^*\}$. The predicted value of i_B^*, in the nominal case, is simply equal to the measured load current, i_L^*. For each residual generator that has states we use the unscented Kalman filter (UKF) for estimation (Julier & Uhlmann, 2004).

The causal graph for the local model for V_i^* ($i \in [1, 24]$) is shown in Fig. 8, and is generated by calling `GenerateSubmodel` on the global model with $U^* = \{i_L^*, i_B^*, V_1^*, V_2^*, \ldots, V_{24}^*\} - \{V_i^*\}$ and $V^* = \{V_i^*\}$. The voltage for each cell is computed independently, using i_B^* as an input (this is divided by 2 to be used as input to the cell model).

5.2. Fault Isolation

Fault isolation is performed by analysis of the residual signals. Due to the decomposition used in the residual generation step, each fault manifests in only a subset of the complete residual set. As is clear in Fig. 7, $r_{i_B^*}$ will deviate (in a statistically significant way from zero) due only to the R_p fault and the i_B^b fault. As is clear in Fig. 8, $r_{V_i^*}$ will deviate due to V_i^b and i_B^b. Note that the relation $i_B^* = i_B + i_B^b$ holds, so when i_B^* is used as a local input, the causal relation is modified so that i_B becomes the dependent variable, and the causal constraint is $i_B := i_B^* - i_B^b$. That is, the true value of i_B is equal to the measured value minus the bias. For residual generation, the bias is not included, so by using the measured value, i_B^* as a local input, when a bias is present the wrong (i.e., biased) current will be fed to the cell model and used to compute V_i^*, thus causing a deviation in the corresponding residual.

The effects of the faults on the residuals are shown in Table 1. Faults are indicated both by the model parameter and the direction of its change, e.g., R_p^- denotes a decrease in the parasitic resistance.[2] Fault effects on residuals are represented as qualitative fault signatures (Mosterman & Biswas, 1999) and relative residual orderings (Daigle, Koutsoukos, & Biswas, 2007). Fault signatures express the qualitative change in a signal as the result of a fault. In general, they can be used to represent changes in magnitude, slope, and higher-order derivatives of a signal, but here, we represent changes in magnitude only, as this is sufficient to obtain unique diagnoses. For example, the parasitic load fault causes an increase in $r_{i_B^*}$. An ordering between a residual r_1 and r_2 for fault f, denoted as $r_1 \prec_f r_2$, indicates that the fault will cause an observable deviation in r_1 before r_2. For example, a bias in the V_1^* sensor will produce a deviation in $r_{V_1^*}$ before every other residual (since the fault affets no other residuals). Both signatures and orderings can be derived from the model automatically (Daigle, 2008).

Both signatures and orderings are reasoned over in an event-based framework to perform fault isolation (Daigle, Koutsoukos, & Biswas, 2009). When a residual deviation is first detected, the fault isolation algorithm checks for the faults that could have produced that deviation. As more residuals deviate, the algorithm checks for consistency with the current sequence of deviations, retaining only faults that can produce the observed sequence according to the predicted signatures and orderings. In addition, we can also eliminate candidates as inconsistent when no deviation is observed in a residual by using timeouts (this is equivalent to “observing” a `0` signature) (Daigle, Roychoudhury, & Bregon, 2013). For each residual we set a time limit under which we expect a residual deviation to occur after a fault. If we detect a fault and that residual has not deviated by that time, we observe a `0` signature and reason with that information. Including this information, we can distinguish qualitatively between all faults, and therefore obtain unique diagnoses based on the qualitative signatures and orderings alone.[3]

5.3. Fault Identification

The fault identification submodels are generated from the global model shown in Fig. 5, with the faulty parts included. In the call to `GenerateSubmodel`, U^* is set to the set of measured variables, and V^* is set to the fault parameter that is to be estimated.

[2]In the nominal model, when the parasitic load is absent, this is equivalent to an infinite resistance in parallel. Thus, the appearance of the parasitic load is denoted as a *decrease* in the parasitic resistance.

[3]Without using the `0` signatures for isolation, if a a voltage sensor bias occurred, we would have to wait infinitely long to ensure there were no further deviations and rule out i_B^b as a possibility.

Table 1. Fault Signatures and Residual Orderings

Fault	$r_{V_1^*}$	$r_{V_2^*}$	$\dots$	$r_{V_{24}^*}$	$r_{i_B^*}$	Residual Orderings
R_p^-	0	0	$\dots$	0	+	$r_{i_B^*} \prec r_{V_1^*}, r_{i_B^*} \prec r_{V_2^*}, \dots, r_{i_B^*} \prec r_{V_{24}^*}$
V_1^{b+}	+	0	$\dots$	0	0	$r_{V_1^*} \prec r_{V_2^*}, \dots, r_{V_1^*} \prec r_{V_{24}^*}, r_{V_1^*} \prec r_{i_B^*}$
V_1^{b-}	−	0	$\dots$	0	0	$r_{V_1^*} \prec r_{V_2^*}, \dots, r_{V_1^*} \prec r_{V_{24}^*}, r_{V_1^*} \prec r_{i_B^*}$
V_2^{b+}	0	+	$\dots$	0	0	$r_{V_2^*} \prec r_{V_1^*}, \dots, r_{V_2^*} \prec r_{V_{24}^*}, r_{V_2^*} \prec r_{i_B^*}$
V_2^{b-}	0	−	$\dots$	0	0	$r_{V_2^*} \prec r_{V_1^*}, \dots, r_{V_2^*} \prec r_{V_{24}^*}, r_{V_2^*} \prec r_{i_B^*}$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
V_{24}^{b+}	0	0	$\dots$	+	0	$r_{V_{24}^*} \prec r_{V_1^*}, \dots, r_{V_{24}^*} \prec r_{V_2^*}, r_{V_{24}^*} \prec r_{i_B^*}$
V_{24}^{b+}	0	0	$\dots$	−	0	$r_{V_{24}^*} \prec r_{V_1^*}, \dots, r_{V_{24}^*} \prec r_{V_2^*}, r_{V_{24}^*} \prec r_{i_B^*}$
i_B^{b+}	+	+	$\dots$	+	+	$\varnothing$
i_B^{b-}	−	−	$\dots$	−	−	$\varnothing$

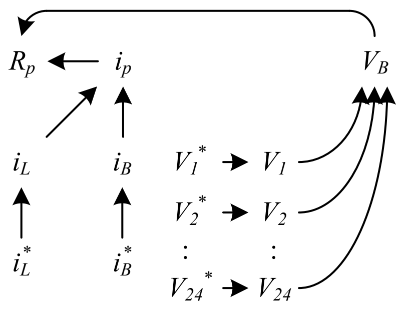

Figure 9. Causal graph for local R_p estimation.

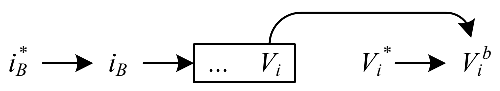

Figure 10. Causal graph for local V_i^b estimation.

For the parasitic load fault, the causal graph for the local estimation model is shown in Fig. 9. The parasitic resistance R_p is computed using i_P and V_B, where i_p is computed based on the difference between the measured load and battery currents, and V_B is computed based on the measured voltages.

The causal graph for the local model for the voltage sensor bias estimation is shown in Fig. 10. The voltage bias is computed based on the measured voltage and the model-predicted voltage, computed using the measured battery current.

The causal graph for the local model for the current sensor bias estimation is shown in Fig. 11. i_B^b is computed as the difference between the measured battery and load currents.

6. ROVER EPS PROGNOSIS

As described in Section 2, prognosis requires a prediction model, an initial state estimate, and future trajectories of the inputs, $\mathbf{U}_{k_P}$ and the process noise, $\mathbf{V}_{k_P}$. The prediction model must be able to compute the event threshold T_E, given the local inputs for prediction.

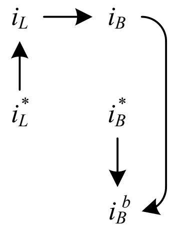

Figure 11. Causal graph for local i_B^b estimation.

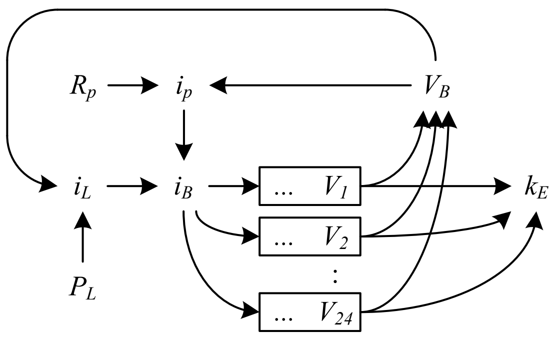

Figure 12. Causal graph for system-level prediction.

The causal graph for the global model for prediction is shown in Fig. 12. We need only to compute T_E, so none of the sensor outputs are included. Note also that for prediction, we use as an input the load power, P_L, instead of the load current. This is because it is much easier in practice to predict load power a priori. With a given speed command to the rover motors, power is constant, but current will increase as the battery cells discharge and V_B decreases.

It is important also to note that the prediction problem cannot be decomposed in general. Given i_B, we can compute each V_i independently, and evaluate $V_i < V_{EOD}$. Since E occurs when any one of the cells drops below the cutoff voltage, we can compute EOD for each cell and take the minimum to determine when E will occur (since E occurs when the first cell reaches EOD). However, i_B depends on i_P and i_L, both

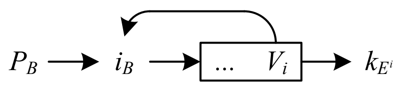

Figure 13. Causal graph for local prediction for cell i.

of which depend on V_B. There are no local inputs to break this dependency.

If we make a simplifying assumption, however, we can decompose the prediction problem and thus achieve the benefits of a distributed implementation. The causal graph for this case is shown in Fig. 13. In this case, we use as a local input the cell power P_B, where $P_B = P_L/24$, thus allowing local EOD thresholds, T_{E^i}, and, hence, local events E^i, to be computed independently. This assumption is only valid if the cells are all approximately equal in voltage, otherwise the assumption of $P_B = P_L/24$ will be violated. Further, this is not valid when R_p is present, as in that case P_B is a function of both P_L and i_p.

In general, the state estimates required for the prediction models must be produced by new estimators derived using structural model decomposition, for the global prediction model. For some faults, however, the needed estimates may be available from the residual generators, if those residual generators were not affected by the fault. In this case, new estimators do not need to be derived. For the parasitic load fault, the residual generator for each V_i^* has the state estimates for the battery cells, and the fault identifier has the value of R_p. We can then reconstruct a global state estimate for use in prediction. For a voltage sensor fault, a new local estimator (same as that used for residual generation, see Fig. 8) is needed to reestimate the states for the corresponding battery cell model. From the time of fault detection onwards, the corrected value of the sensor, computed by removing the estimated bias, is used to reestimate the states. For the current sensor fault, the case is more complex, because a faulty sensor reading was used in all of the local voltage estimators. Therefore, new local estimators are needed for all cells, in which the bias-corrected value must be fed as an input from the time of fault detection onward, once the fault bias has been identified.

In this work, we assume that process noise is negligible compared to the future input uncertainty, so represent the uncertainty only in the future input trajectories $\mathbf{U}_{k_P}$ (i.e, the trajectory of P_L). We use the surrogate variable method to represent the future input trajectories (Daigle & Sankararaman, 2013). In this method, we represent $\mathbf{U}_{k_P}$ through a set of surrogate variables, such that $\mathbf{U}_{k_P}$ can be constructed in a deterministic way given values of the surrogate variables. In this way, we can represent the probability distributions of the surrogate variables to indirectly represent the probability distribution of the input trajectories. For the rover, we consider an equivalent constant-loading distribution for the future inputs. That is, we assume that the future load power, P_L, will be constant with the value drawn from some distribution. In the case of the rover, the operator really only needs to know EOD predictions for best-, average-, and worst-case usage scenarios (Daigle & Kulkarni, 2014). For the state estimate, we use as samples the sigma points provided by the UKF. Each sample is simulated forward three times, once for each use case. From this we obtain best-, average-, and worst-case EOD predictions, each with some small variance (due to the state estimate variance).

It is important to note that since R_p is included in the prediction model, the prediction input does not change in the nominal and faulty cases. If, however, R_p was considered part of the load, i.e., part of P_L, then P_L prediction would have to change in the faulty case and would be complicated, since the additional power required by R_p is actually a function of battery voltage (as shown in Fig. 12). This is an advantage of viewing the prediction problem in a system-level perspective (the EPS perspective), rather than a component-level perspective (the battery cell perspective).

7. RESULTS

In this section, we demonstrate the integrated system-level diagnosis and prognosis framework on the rover case study, using real experimental field data. The task of the rover is to travel to different waypoints to complete some science objective. We must predict how long the rover will be able to execute its mission before having to return to the start point. Faults must be diagnosed so that the mission can be replanned if the rover is unable to meet all of its objectives due to the fault, and does not become stranded before returning to the start point.

We consider first a nominal scenario, in which the rover has enough energy to visit all waypoints and return successfully to the start point. Fig. 14 shows the measured and estimated values of V_1^* (results are similar for the remaining voltage sensors). With $V_{EOD} = 2.5$ V, EOD is clearly not reached. Fig. 15 shows tracking of the battery current sensor. Although the measured value is very noisy, the residual remains within the nominal range, and no fault is detected in any of the residuals. Fig. 16 shows the system-level EOD predictions for the rover. Each prediction consists of three points, for best-, average-, and worst-case future loading. Here, even in the worst-case scenario the predictions indicate that the rover will be able to complete the mission.

We next consider a parasitic load fault of 20 Ω, appearing as an additional load on the batteries, draining additional current and causing the batteries to discharge more quickly. The fault occurs at 780 s, and is detected at 801 s on the battery current residual, as shown in Fig. 17. Given the increase in the battery current, the parasitic load fault and a positive bias in

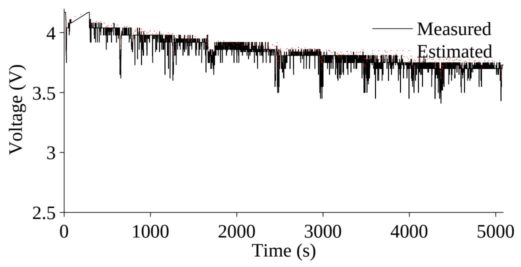

Figure 14. Estimation of V_1^*.

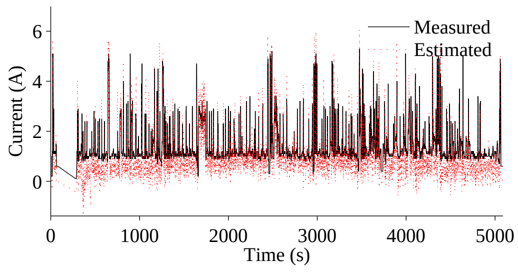

Figure 15. Estimation of i_B^*.

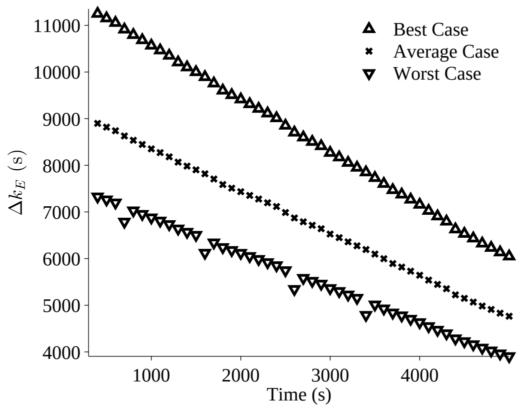

Figure 16. Predictions of Δk_E for worst-, average-, and best-case future usage scenarios.

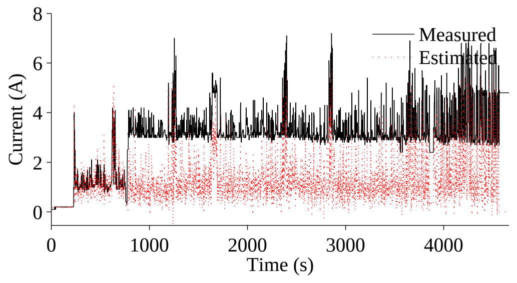

Figure 17. Estimation of i_B^* for a parasitic load.

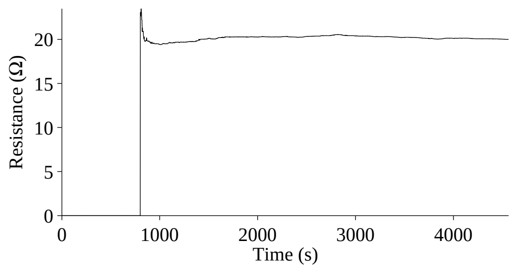

Figure 18. Estimation of R_p for a parasitic load.

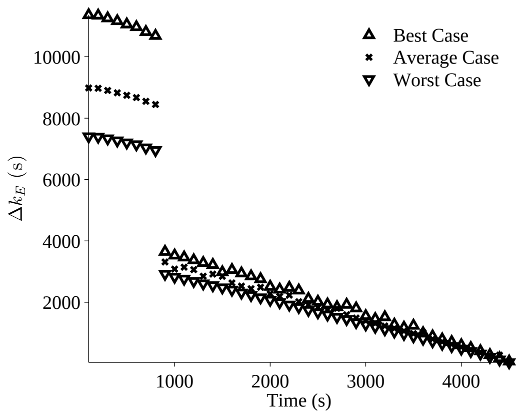

Figure 19. Predictions of Δk_E for worst-, average-, and best-case future usage scenarios with a parasitic load fault.

the battery current sensor are the only possible faults (see Table 1). At 922 s, two minutes after fault detection, we observe a 0 symbol on all the voltage sensor residuals, since they have not yet deviated. Given these observations, the only consistent candidate is the parasitic load fault. The estimated parasitic resistance over time is shown in Fig. 18. The estimate converges to the true value in less than 50 s, and stays very close to the true value. As described in Section 6, the prediction problem in this case cannot be decomposed, because the parasitic current depends on the battery voltages, so the local input for prediction is the total motor power. The system-level predictions are shown in Fig. 19. Before the fault is diagnosed, the predictions indicate that the rover will be able to complete its mission. After the fault is diagnosed, the predictions reflect the fact that more power is being demanded from the batteries, and EOD will be reached much sooner, requiring the mission to be shortened.

We next consider a battery voltage sensor fault, manifesting as a constant offset (bias) of 0.2 V on the voltage sensor for battery 1. The fault is injected at 600 s and detected at 634 s in the residual for the faulty sensor, as shown in Fig. 20. It is immediately diagnosed, as no other fault can produce a deviation first in the voltage sensor, according to the residual orderings. In order to recover from this fault, the estimator for the voltage is reset back to the estimated time of the fault, and is updated up to the current time using the unbiased signal, computed as the measured signal value minus the estimated bias. From the current time on, the present value of the estimated bias is used to correct the measured value sent to

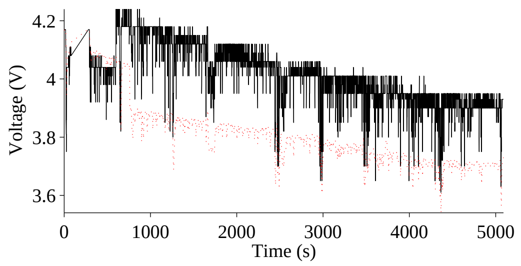

Figure 20. Estimation of V_1^* for the voltage sensor bias fault.

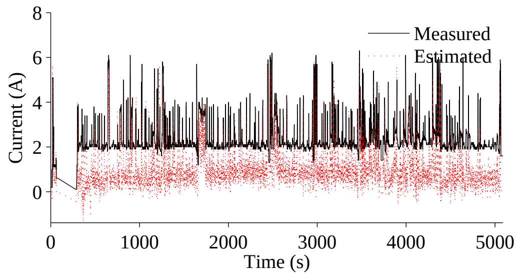

Figure 21. Estimation of i_B^* with a battery current sensor fault.

the estimator. Because this fault does not actually have any effect on the energy required by the rover, the predictions are the same as in the nominal condition.

Finally, we consider an offset fault in the battery current sensor. The fault is injected at 300 s, and is detected at 344 s. Detection time is slow due to the high amount of noise in the sensor. The tracking of the sensor is shown in Fig. 21, where the bias is clear visually (c.f. Fig. 15). The initial diagnosis is either the parasitic load fault, which can also cause an increase in the current, and a current sensor fault. Because a faulty current sensor value is being used as a local input to the voltage estimators, these residuals deviate as well. Tracking for V_1^* is shown in Fig. 22. Because a larger current is used, the estimated voltage drains faster than actual, and a deviation is detected at 415 s, thus isolating the current sensor fault as the true fault. Since the state estimates for the batteries will be corrupted, this will propagate to the predictions, giving incorrect results. So, to recover from the fault, once the fault is identified, the battery estimators are reset to the time of fault detection, and the corrected measurement value, based on the estimated bias is fed up to the current time and in the future. There is no physical effect on the energy consumption of the rover due to the fault, and therefore the predictions match those in the nominal case.

8. CONCLUSIONS

In this paper, we developed and implemented an approach for integrated system-level diagnosis and prognosis of the electrical power system of a planetary rover testbed. The algorithms monitor the behavior of the EPS and generate symbols for fault isolation in a distributed fashion. Fault isolation is performed, and for each fault hypothesis, system-level prognosis is performed, starting with distributed estimation of the state and fault parameters, and followed by distributed prediction. The distributed nature of the architecture is based upon the use of local submodels that enable the decomposition of global diagnosis and prognosis problems into local subproblems, applying ideas established in previous works. The approach was demonstrated using field data from the rover, showing successful detection, isolation, identification, and prediction for a set of realistic faults.

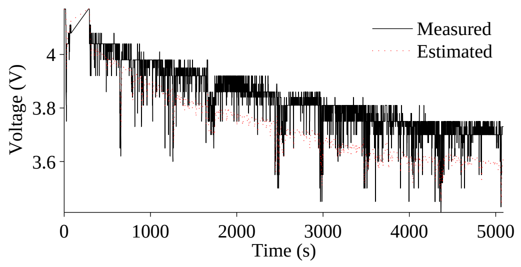

Figure 22. Estimation of V_1^* with a battery current sensor fault.

Future work will extend the application of the framework to the entire rover system, not just the EPS, which will enable the diagnosis of faults in the rover motors, and incorporation of that information into system-level predictions. We will also apply the approach to other systems, and make further theoretical extensions of the work, e.g., by including multiple faults, and hybrid systems.

ACKNOWLEDGEMENTS

The authors acknowledge Adam Sweet, NASA Ames Research Center, and George Gorospe, SGT, Inc., NASA Ames Research Center, for obtaining the experimental data used in this work. M. Daigle's and I. Roychoudhury's funding for this work was provided by the NASA System-wide Safety and Assurance Technologies (SSAT) Project. A. Bregon's funding for this work was provided by the Spanish MICINN DPI2013-45414-R grant.

REFERENCES

Balaban, E., Narasimhan, S., Daigle, M., Roychoudhury, I., Sweet, A., Bond, C., & Gorospe, G. (2013). Development of a mobile robot test platform and methods for validation of prognostics-enabled decision making algorithms. *Intl. Journal of Prognostics and Health Management*, *4*(1).

Blanke, M., Kinnaert, M., Lunze, J., & Staroswiecki, M. (2006). *Diagnosis and fault-tolerant control*. Springer.

Bregon, A., Biswas, G., & Pulido, B. (2012). A Decomposition Method for Nonlinear Parameter Estimation in TRANSCEND. *IEEE Trans. Syst. Man. Cy. Part A*, *42*(3), 751-763.

Bregon, A., Daigle, M., & Roychoudhury, I. (2012, September). An integrated framework for model-based distributed diagnosis and prognosis. In *Annual conference of the prognostics and health management society 2012* (p. 416-426).

Bregon, A., Daigle, M., Roychoudhury, I., Biswas, G., Koutsoukos, X., & Pulido, B. (2014, May). An event-based distributed diagnosis framework using structural model decomposition. *Artificial Intelligence*, *210*, 1-35.

Daigle, M. (2008). *A qualitative event-based approach to fault diagnosis of hybrid systems*. Unpublished doctoral dissertation, Vanderbilt University.

Daigle, M., Bregon, A., & Roychoudhury, I. (2012, September). A distributed approach to system-level prognostics. In *Annual conference of the prognostics and health management society 2012* (p. 71-82).

Daigle, M., Bregon, A., & Roychoudhury, I. (2014, June). Distributed prognostics based on structural model decomposition. *IEEE Trans. on Reliability*, *63*(2), 495-510.

Daigle, M., Koutsoukos, X., & Biswas, G. (2007, April). Distributed diagnosis in formations of mobile robots. *IEEE Trans. on Robotics*, *23*(2), 353–369.

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

Daigle, M., & Kulkarni, C. (2013, October). Electrochemistry-based battery modeling for prognostics. In *Annual conference of the prognostics and health management society 2013* (p. 249-261).

Daigle, M., & Kulkarni, C. (2014). A battery health monitoring framework for planetary rovers. In *Proceedings of the ieee aerospace conference.*

Daigle, M., Roychoudhury, I., & Bregon, A. (2013, October). Qualitative event-based diagnosis with possible conflicts: Case study on the fourth intl. diagnostic competition. In *Proc. of the 24th intl. workshop on principles of diagnosis* (p. 230-235).

Daigle, M., & Sankararaman, S. (2013, October). Advanced methods for determining prediction uncertainty in model-based prognostics with application to planetary rovers. In *Annual conference of the prognostics and health management society 2013* (p. 262-274).

Daigle, M., Saxena, A., & Goebel, K. (2012, September). An efficient deterministic approach to model-based prediction uncertainty estimation. In *Annual conference of the prognostics and health management society* (p. 326-335).

Gertler, J. J. (1998). *Fault detection and diagnosis in engineering systems*. New York, NY: Marcel Dekker, Inc.

Julier, S. J., & Uhlmann, J. K. (2004, March). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, *92*(3), 401–422.

Karthikeyan, D. K., Sikha, G., & White, R. E. (2008). Thermodynamic model development for lithium intercalation electrodes. *Journal of Power Sources*, *185*(2), 1398–1407.

Luo, J., Pattipati, K. R., Qiao, L., & Chigusa, S. (2008, September). Model-based prognostic techniques applied to a suspension system. *IEEE Trans. on Systems, Man and Cybernetics, Part A: Systems and Humans*, *38*(5), 1156 -1168.

Mosterman, P. J., & Biswas, G. (1999). Diagnosis of continuous valued systems in transient operating regions. *IEEE Trans. on Systems, Man, and Cybernetics, Part A: Systems and Humans*, *29*(6), 554-565.

Orchard, M. E., & Vachtsevanos, G. (2009). A particle-filtering approach for on-line fault diagnosis and failure prognosis. *Trans. of the Institute of Measurement and Control*, *31*(3/4), 221-246.

Patrick, R., Orchard, M. E., Zhang, B., Koelemay, M., Kacprzynski, G., Ferri, A., & G., V. (2007, September). An integrated approach to helicopter planetary gear fault diagnosis and failure prognosis. In *Proc. of the 42nd annual systems readiness technology conf.* Baltimore, MD, USA.

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

Rahn, C. D., & Wang, C.-Y. (2013). *Battery systems engineering*. Wiley.

Roychoudhury, I., & Daigle, M. (2011, October). An integrated model-based diagnostic and prognostic framework. In *Proc. of the 22nd intl. workshop on principles of diagnosis* (p. 44-51).

Roychoudhury, I., Daigle, M., Bregon, A., & Pulido, B. (2013, March). A structural model decomposition framework for systems health management. In *Proceedings of the 2013 ieee aerospace conference.*

Saha, B., & Goebel, K. (2009, September). Modeling Li-ion battery capacity depletion in a particle filtering framework. In *Proceedings of the annual conference of the prognostics and health management society 2009.*

Sankararaman, S., Daigle, M., & Goebel, K. (2014, June). Uncertainty quantification in remaining useful life prediction using first-order reliability methods. *IEEE Trans. on Reliability*, *63*(2), 603-619.

Sankararaman, S., Daigle, M., Saxena, A., & Goebel, K. (2013, March). Analytical algorithms to quantify the uncertainty in remaining useful life prediction. In *Proc. of the 2013 IEEE aerospace conference.*

Zabi, S., Ribot, P., & Chanthery, E. (2013, October). Health monitoring and prognosis of hybrid systems. In *Proc. of the annual conference of the prognostics and health management society 2013* (p. 300-311).

BIOGRAPHIES

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

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

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

APPENDIX: BATTERY CELL MODELING

The battery cell model computes the voltage as a function of time given the current drawn from the cell, and is described in detail in (Daigle & Kulkarni, 2013). We summarize the model here and refer the reader to (Daigle & Kulkarni, 2013) for additional explanation.

The voltage terms of the battery are expressed as functions of the amount of charge in the electrodes (the states of the model). Each electrode, positive (subscript p) and negative (subscript n), is split into two volumes, a surface layer (subscript s) and a bulk layer (subscript b). The differential equations for the battery describe how charge moves through these volumes. The charge (q) variables are described using

$$\dot{q}_{s,p} = i_{app} + \dot{q}_{bs,p} \tag{5}$$
$$\dot{q}_{b,p} = -\dot{q}_{bs,p} + i_{app} - i_{app} \tag{6}$$
$$\dot{q}_{b,n} = -\dot{q}_{bs,n} + i_{app} - i_{app} \tag{7}$$
$$\dot{q}_{s,n} = -i_{app} + \dot{q}_{bs,n}, \tag{8}$$

where i_{app} is the applied electric current The term $\dot{q}_{bs,i}$ describes diffusion from the bulk to surface layer for electrode i:

$$\dot{q}_{bs,i} = \frac{1}{D}(c_{b,i} - c_{s,i}), \tag{9}$$

where D is the diffusion constant. The c terms are lithium ion concentrations:

$$c_{b,i} = \frac{q_{b,i}}{v_{b,i}} \tag{10}$$
$$c_{s,i} = \frac{q_{s,i}}{v_{s,i}}, \tag{11}$$

where, for CV v in electrode i, $c_{v,i}$ is the concentration and $v_{v,i}$ is the volume. We define $v_i = v_{b,i} + v_{s,i}$. Note now that the following relations hold:

$$q_p = q_{s,p} + q_{b,p} \tag{12}$$
$$q_n = q_{s,n} + q_{b,n} \tag{13}$$
$$q^{\max} = q_{s,p} + q_{b,p} + q_{s,n} + q_{b,n}. \tag{14}$$

We can also express mole fractions (x) based on the q vari-

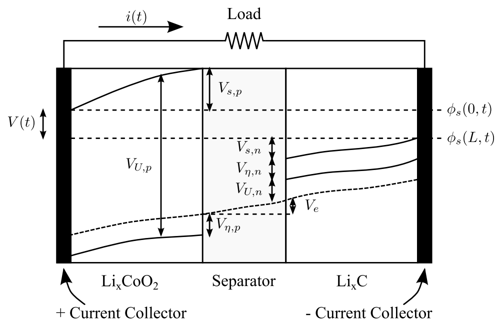

Figure 23. Battery voltages.

ables:

$$x_i = \frac{q_i}{q^{\max}}, \tag{15}$$

$$x_{s,i} = \frac{q_{s,i}}{q_{s,i}^{\max}}, \tag{16}$$

$$x_{b,i} = \frac{q_{b,i}}{q_{b,i}^{\max}}, \tag{17}$$

where $q^{\max} = q_p + q_n$ refers to the total amount of available Li ions. It follows that $x_p + x_n = 1$. For lithium ion batteries, when fully charged, $x_p = 0.4$ and $x_n = 0.6$. When fully discharged, $x_p = 1$ and $x_n = 0$ (Karthikeyan, Sikha, & White, 2008).

The different potentials are summarized in Fig. 23 (adapted from (Rahn & Wang, 2013)). The overall battery voltage $V(t)$ is the difference between the potential at the positive current collector, $\phi_s(0,t)$, and the negative current collector, $\phi_s(L,t)$, minus resistance losses at the current collectors (not shown in the diagram). At the positive current collector is the equilibrium potential $V_{U,p}$. This voltage is then reduced by $V_{s,p}$, due to the solid-phase ohmic resistance, and $V_{\eta,p}$, the surface overpotential. The electrolyte ohmic resistance then causes another drop V_e. At the negative electrode, there is a drop $V_{\eta,n}$ due to the surface overpotential, and a drop $V_{s,n}$ due to the solid-phase resistance. The voltage drops again due to the equilibrium potential at the negative current collector $V_{U,n}$. These voltages are described by the following set of equations (see (Daigle & Kulkarni, 2013) for details):

$$V_{U,i} = U_0 + \frac{RT}{nF}\ln\left(\frac{1-x_{s,i}}{x_{s,i}}\right) + V_{\text{INT},i}, \tag{18}$$

$$V_{\text{INT},i} = \frac{1}{nF}\left(\sum_{k=0}^{N_i} A_{i,k}\left((2x_i-1)^{k+1} - \frac{2x_ik(1-x_i)}{(2x_i-1)^{1-k}}\right)\right), \tag{19}$$

$$V_o = i_{app}R_o, \tag{20}$$

$$V_{\eta,i} = \frac{RT}{F\alpha}\text{arcsinh}\left(\frac{J_i}{2J_{i0}}\right), \tag{21}$$

$$J_i = \frac{i}{S_i}, \tag{22}$$

$$J_{i0} = k_i(1-x_{s,i})^{\alpha}(x_{s,i})^{1-\alpha}, \tag{23}$$

$$V = V_{U,p} - V_{U,n} - V_o' - V_{\eta,p}' - V_{\eta,n}', \tag{24}$$

$$\dot{V}_o' = (V_o - V_o')/\tau_o \tag{25}$$

$$\dot{V}_{\eta,p}' = (V_{\eta,p} - V_{\eta,p}')/\tau_{\eta,p} \tag{26}$$

$$\dot{V}_{\eta,n}' = (V_{\eta,n} - V_{\eta,n}')/\tau_{\eta,n}. \tag{27}$$

Here, U_0 is a reference potential, R is the universal gas constant, T is the electrode temperature (in K), n is the number of electrons transferred in the reaction ($n = 1$ for Li-ion), F is Faraday's constant, J_i is the current density, and J_{i0} is the exchange current density, k_i is a lumped parameter of several constants including a rate coefficient, electrolyte concentration, and maximum ion concentration. $V_{\text{INT},i}$ is the activity correction term (0 in the ideal condition). We use the Redlich-Kister expansion with $N_p = 12$ and $N_n = 0$ (see (Daigle & Kulkarni, 2013)). The τ parameters are empirical time constants (used since the voltages do not change instantaneously).

The model contains as states $\mathbf{x}$, $q_{s,p}$, $q_{b,p}$, $q_{b,n}$, $q_{s,n}$, V_o', $V_{\eta,p}'$, and $V_{\eta,n}'$. The single model output is V.

The state of charge (SOC) of a battery is defined to be 1 when the battery is fully charged and 0 when the battery is fully discharged by convention. In this model, it is analogous to the mole fraction x_n, but scaled from 0 to 1. We distinguish here between nominal SOC and *apparent* SOC (Daigle & Kulkarni, 2013). Nominal SOC is computed based on the combination of the bulk and surface layer CVs in the negative electrode, whereas apparent SOC is be computed based only on the surface layer. When a battery reaches the voltage cutoff, apparent SOC is 0, and nominal SOC is greater than 0 (how much greater depends on the difference between the diffusion rate and the current drawn). Once the concentration gradient settles out, the surface layer will be partially replenished and apparent SOC will rise while nominal SOC remains the same. Nominal (n) and apparent (a) SOC are defined using

$$SOC_n = \frac{q_n}{0.6q^{\max}} \tag{28}$$

$$SOC_a = \frac{q_{s,n}}{0.6q^{\max_{s,n}}}, \tag{29}$$

where $q^{\max_{s,n}} = q^{\max}\frac{v_{s,n}}{v_n}$.[4]

[4]Note that SOC of 1 corresponds to the point where $q_n = 0.6q^{\max_{s,n}}$, since the mole fraction at the positive electrode cannot go below 0.4, as described earlier.