A Distributed Approach to System-Level Prognostics

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

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

[2] *University of Valladolid, Valladolid, Spain*
anibal@infor.uva.es

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

ABSTRACT

Prognostics, which deals with predicting remaining useful life of components, subsystems, and systems, is a key technology for systems health management that leads to improved safety and reliability with reduced costs. The prognostics problem is often approached from a component-centric view. However, in most cases, it is not specifically component lifetimes that are important, but, rather, the lifetimes of the systems in which these components reside. The system-level prognostics problem can be quite difficult due to the increased scale and scope of the prognostics problem and the relative lack of scalability and efficiency of typical prognostics approaches. In order to address these issues, we develop a distributed solution to the system-level prognostics problem, based on the concept of structural model decomposition. The system model is decomposed into independent submodels. Independent local prognostics subproblems are then formed based on these local submodels, resulting in a scalable, efficient, and flexible distributed approach to the system-level prognostics problem. We provide a formulation of the system-level prognostics problem and demonstrate the approach on a four-wheeled rover simulation testbed. The results show that the system-level prognostics problem can be accurately and efficiently solved in a distributed fashion.

1. INTRODUCTION

Prognostics is the process of predicting the end of (useful) life (EOL) and/or the remaining useful life (RUL) of components, subsystems, or systems. The prognostics problem itself can be divided into two distinct problems: (*i*) the *estimation* problem, which determines the current state of the system, and (*ii*) the *prediction* problem, which, using the current system state estimate, computes EOL and/or RUL. In this paper, we focus on a model-based prognostics approach (Orchard & Vachtsevanos, 2009; Daigle & Goebel, 2011b; Saha & Goebel, 2009; Luo et al., 2008). In model-based prognostics, an underlying model of the system, its components, and how they fail is leveraged, where health state estimation is formulated as a joint state-parameter estimation problem, typically using a filtering approach, and prediction is formulated as a simulation problem (Daigle, Saha, & Goebel, 2012).

To the best of our knowledge, all prognostics research to date has been focused on *individual components*, and determining their EOL and RUL, e.g., (Orchard & Vachtsevanos, 2009; Saha & Goebel, 2009; Daigle & Goebel, 2011a; Celaya et al., 2011; Bolander et al., 2010; Luo et al., 2008; Byington et al., 2004). However, in many cases, the desired information is the EOL of the *system*, which is obtained through *system-level prognostics*. Generally, the EOL of a system depends on its constituent components and how they interact. Approaching this problem from the centralized perspective becomes very difficult, as common (centralized) prognostics algorithms may not scale to the system level.

In order to address the problems with centralized approaches, in recent work, we have developed a distributed model-based prognostics architecture that allows the decomposition of a large prognostics problem into several independent local subproblems from which local results can be merged into a global result (Daigle et al., 2011; Daigle, Bregon, & Roychoudhury, 2012). Since each local subproblem can be solved independently, each can be assigned to a different processing unit and be solved in parallel. Such a distributed approach is in contrast to other proposed distributed prognostics architectures in which the global problem is not decomposed and the computation is distributed onto multiple processing units, e.g., (Saha, Saha, & Goebel, 2009). Our distributed approach

scales well and the resulting subproblems are typically small and easy to solve, resulting in an efficient and flexible distributed solution to the prognostics problem. Such an approach has obvious advantages when applied to the system-level prognostics problem. In this paper, we formulate the system-level prognostics problem and propose a solution using this distributed prognostics framework. We apply our system-level prognostics approach to a rover testbed and provide results in simulation to empirically demonstrate and validate the approach.

The paper is organized as follows. Section 2 formulates the system-level prognostics problem and overviews the proposed distributed solution. Section 3 describes the estimation problem, and Section 4 describes the prediction problem. Section 5 presents the rover case study, and shows prognostics results in simulation. Section 6 concludes the paper.

2. System-level Prognostics

While most prognostics approaches focus on individual components, in most practical cases it is actually the EOL of the system that must be determined. With this prediction, the future usage of the system may be optimally planned to maximize system life and to schedule system-wide maintenance activities. It is often important to take a system-level perspective of prognostics, because the degradation of individual components is often coupled, i.e., the way one component degrades is dependent on how a connected component degrades. This may occur, for example, if one component provides the inputs to another component, in which case, prognostics of the latter component cannot be performed in isolation.

In this section, we first define the system-level prognostics problem. We then introduce the system-level prognostics approach and architecture using a distributed prognostics framework that is based on structural model decomposition.

2.1. Problem Formulation

The goal of system-level prognostics is the prediction of the EOL and/or RUL of a system. We assume the system model may be generally defined as

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t), \boldsymbol{\theta}(t), \mathbf{u}(t), \mathbf{v}(t)),$$
$$\mathbf{y}(t) = \mathbf{h}(t, \mathbf{x}(t), \boldsymbol{\theta}(t), \mathbf{u}(t), \mathbf{n}(t)),$$

where $\mathbf{x}(t) \in \mathbb{R}^{n_x}$ is the state vector, $\boldsymbol{\theta}(t) \in \mathbb{R}^{n_\theta}$ is the unknown parameter vector, $\mathbf{u}(t) \in \mathbb{R}^{n_u}$ is the input vector, $\mathbf{v}(t) \in \mathbb{R}^{n_v}$ is the process noise vector, $\mathbf{f}$ is the state equation, $\mathbf{y}(t) \in \mathbb{R}^{n_y}$ is the output vector, $\mathbf{n}(t) \in \mathbb{R}^{n_n}$ is the measurement noise vector, and $\mathbf{h}$ is the output equation.[1] This model describes both the nominal behavior and faulty behavior, including the fault progression functions.

[1] Here, we use bold typeface to denote vectors, and use n_a to denote the length of a vector $\mathbf{a}$.

In system-level prognostics, we are interested in when the performance of a system lies outside some desired region of acceptable behavior. The desired performance is expressed through a set of n_c constraints, $C_{EOL} = \{c_i\}_{i=1}^{n_c}$, where $c_i : \mathbb{R}^{n_x} \times \mathbb{R}^{n_\theta} \times \mathbb{R}^{n_u} \rightarrow \mathbb{B}$ maps a given point in the joint state-parameter space given the current inputs, $(\mathbf{x}(t), \boldsymbol{\theta}(t), \mathbf{u}(t))$, to the Boolean domain $\mathbb{B} \triangleq [0, 1]$, where $c_i(\mathbf{x}(t), \boldsymbol{\theta}(t), \mathbf{u}(t)) = 1$ if the constraint is satisfied. If $c_i(\mathbf{x}(t), \boldsymbol{\theta}(t), \mathbf{u}(t)) = 0$, then the constraint is not satisfied, and the behavior of the system is deemed to be unacceptable. These deterministic constraints may refer to component-level, subsystem-level, or system-level specifications or requirements and define a fixed partition of the state-parameter-input space into acceptable and unacceptable regions of behavior. When the constraints are violated, it does not necessarily refer to a hard failure, but any point at which the operational risk is too large to continue system operation, or future behaviors of the system will be in some way unacceptable. At this point we say the system has no *useful* life remaining.

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

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

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

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

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

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

Note that because $\mathbf{x}(t)$ is a random variable, EOL and RUL must necessarily be random variables also.

2.2. Prognostics Approach

In order to make an EOL or RUL prediction for the system, the initial state from which to make a prediction is required. In general, this initial state is not directly observed, and must be estimated. Therefore, there are two sequential problems for prognostics: the *estimation* problem and the *prediction* problem. The estimation problem is to find a joint state-parameter estimate $p(\mathbf{x}(t), \boldsymbol{\theta}(t) | \mathbf{y}_{0:t})$ based on the history of observations up to time t, $\mathbf{y}_{0:t}$. This estimate is represented as a probability distribution because, generally, the system

state is not directly observed, and there is sensor noise, $\mathbf{n}(t)$, and process noise, $\mathbf{v}(t)$. At a given prediction time, t_P, the prediction algorithm uses the joint state-parameter estimate $p(\mathbf{x}(t_P), \boldsymbol{\theta}(t_P)|\mathbf{y}_{0:t_P})$ and computes $p(EOL(t_P)|\mathbf{y}_{0:t_P})$ and $p(RUL(t_P)|\mathbf{y}_{0:t_P})$. Along with the uncertainty in the state-parameter estimate, process noise and uncertainty in the future inputs to the system all contribute to the uncertainty in the EOL/RUL prediction.

This system-level prognostics problem, consisting of estimating the system state and then predicting its evolution to EOL, can be solved using component-level approaches by treating the entire system as a single component and applying these approaches directly. However, for a large system, both the estimation and prediction problems are correspondingly large. Due to the large state-parameter dimension, a centralized approach does not scale well, and can be very inefficient.

Therefore, we propose to decompose the *global* system-level prognostics problem into independent *local* subproblems, such that the solutions to the local subproblems may be easily merged to form the solution to the global prognostics problem. This forms a naturally distributed approach in which the local subproblems, since they are independent, may be solved in parallel, thus providing scalability and efficiency. Further, the approach allows different algorithms to be employed on each subproblem. The subproblems often correspond directly to component-level prognostics problems, and the approach provides a mechanism to combine component-level prognostics results into system-level results.

In (Daigle et al., 2011), we developed such a distributed solution to the estimation part of the prognostics problem, based on the concept of structural model decomposition (Pulido & Alonso-González, 2004). In recent work, the same concept was used to decompose the prediction problem (Daigle, Bregon, & Roychoudhury, 2012). Structural model decomposition allows one to decompose a system model into a set of submodels for which local prognostics problems can be directly defined. The global model of the system, denoted as $\mathcal{M}$, is defined as follows.

Definition 1 (Model)**.** The model of a system, $\mathcal{M}$, is a tuple $\mathcal{M} = (X, \Theta, U, Y, C)$, where X is the set of state variables of $\mathbf{x}$, Θ is the set of unknown parameters of $\boldsymbol{\theta}$, U is the set of input variables of $\mathbf{u}$, Y is the set of output variables of $\mathbf{y}$, and C is the set of model constraints of $\mathbf{f}$, $\mathbf{h}$, and C_{EOL}.

Informally, a model consists of a set of variables and a set of constraints among the variables. While technically $\mathbf{f}$ and $\mathbf{h}$ themselves are (complex) constraints, we represent them instead as sets of simple constraints. This view is also more consistent with the way modelers describe $\mathbf{f}$ and $\mathbf{h}$, i.e., as sets of equations, each describing a single state or output variable.

Model decomposition is accomplished by assigning some variables as local inputs for which the values are known (e.g., they are directly measured). In this way, the submodels are made computationally independent of each other. Within this scheme, a submodel is then defined as follows.

Definition 2 (Submodel)**.** A submodel $\mathcal{M}_i$ of a system model $\mathcal{M} = (X, \Theta, U, Y, C)$ is a tuple $\mathcal{M}_i = (X_i, \Theta_i, U_i, Y_i, C_i)$, where $X_i \subseteq X$, $\Theta_i \subseteq \Theta$, $U_i \subseteq X \cup U \cup Y$, and $Y_i \subseteq Y$ are the state, parameter, input, and output variables, respectively, and $C_i \subseteq C$ are the submodel constraints.

For distributed prognostics, we find a set of submodels that satisfy a certain set of properties. For distributed estimation, the submodels use $U_i \subseteq U \cup (Y - Y_i)$, and we find a set of minimal submodels such that each Y_i is a singleton, and over all Y_i, Y_j where $i \neq j$, $Y_i \cap Y_j = \varnothing$. So, each submodel uses some global model inputs and some measured values as local inputs, and, in this way, the submodels become decoupled and may be computed independently from each other. By creating submodels with one output variable each, we maximize the number of estimation submodels and the opportunity for parallelization of the estimation task. By making the submodels minimal, they require no constraints or variables that are not strictly necessary to compute Y_i. An algorithm for computing the set of submodels with these properties is given in (Daigle et al., 2011), which is based on the model decomposition algorithms presented in (Pulido & Alonso-González, 2004; Bregon, Biswas, & Pulido, 2012).

For distributed prediction, the submodels use $U_i \subseteq U_P$, where $U_P \subseteq X \cup U$. Here, U_P is a set of variables whose future values can be hypothesized. In the centralized case, $U_P = U$. We find a set of minimal submodels such that each submodel has at least one $c \in C_{EOL}$ belonging to C_i, and over all submodels, C_{EOL} is covered. This ensures that T_{EOL} may be computed for the system; since T_{EOL} is 1 whenever any of the constraints in C_{EOL} are violated, we can independently evaluate when those individual constraints will be violated and then take the minimum to obtain the system EOL. An algorithm for computing the set of submodels with these properties is given in (Daigle, Bregon, & Roychoudhury, 2012). Both decomposition algorithms work in a similar way; essentially, they start with a variable or constraint that must be computed in the local submodel, and then trace the dependencies backwards until local inputs are reached, including all variables and constraints found throughout the search within the submodel.

Note that the problem of defining U_P is critical to obtaining accurate results for system-level EOL in a distributed manner. On average, the most accurate result will be achieved when the system model is directly used for prediction, because it captures all the interdependencies between the components. In the general case, damage could be progressing in multiple components, and damage progression in one component may have an effect on damage progression in another component due to their coupling. In such cases, for system-level prog-

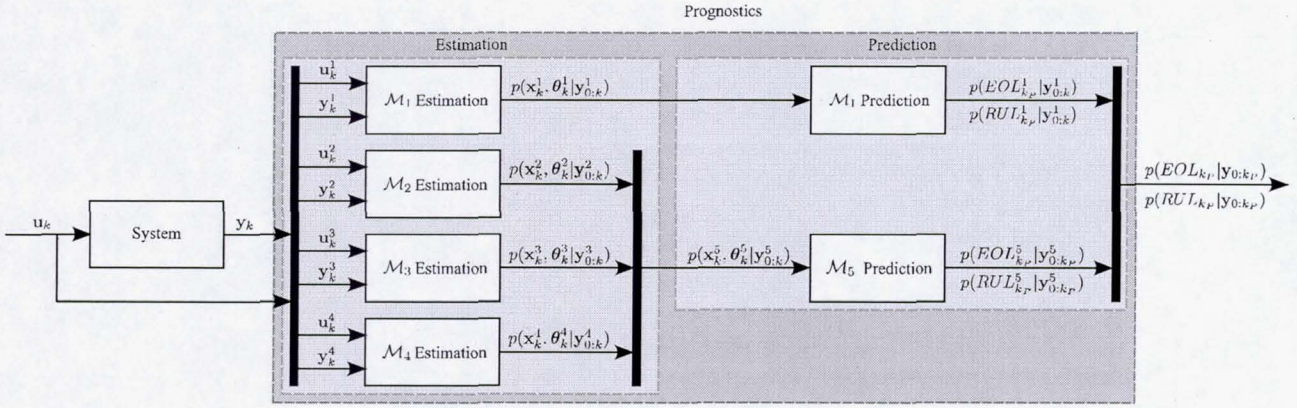

Figure 1. Sample system-level prognostics architecture.

nostics the components cannot be decoupled due to these interactions, and the prediction problem cannot be decomposed into two independent problems, one for each component. It is only appropriate to neglect these interactions when they are either negligible or predictable a priori. It will be shown in Section 5 how this is an important consideration.

2.3. Prognostics Architecture

A sample system-level prognostics architecture based on the distributed framework is shown in Fig. 1. In discrete time k, and using a discrete-time version of the model, the damage estimation module takes as input both $\mathbf{u}_k$ and $\mathbf{y}_k$ and splits them into local inputs and outputs for the submodels. Estimation is performed for each submodel using an appropriate algorithm, computing local state-parameter estimates $p(\mathbf{x}_k^i, \boldsymbol{\theta}_k^i | \mathbf{y}_{0:k}^i)$. Some of these local estimates are merged corresponding to the prediction submodels. For example, submodel $\mathcal{M}_5$ builds its local state using the estimates from the estimators of $\mathcal{M}_2$, $\mathcal{M}_3$, and $\mathcal{M}_4$. The local predictors compute local EOL/RUL predictions $p(EOL_{k_P}^i | \mathbf{y}_{0:k_P}^i)$ and $p(RUL_{k_P}^i | \mathbf{y}_{0:k_P}^i)$ at given prediction time k_P based on the local EOL constraints. Local predictions are then merged into global predictions $p(EOL_{k_P} | \mathbf{y}_{0:k_P})$ and $p(RUL_{k_P} | \mathbf{y}_{0:k_P})$ by taking the minimum of the local predictions.

3. Distributed Estimation

As described in Section 2, in our distributed estimation scheme, the local estimator for each submodel $\mathcal{M}_i$ produces a local estimate $p(\mathbf{x}_k^i, \boldsymbol{\theta}_k^i | \mathbf{y}_{0:k}^i)$, where $\mathbf{x}_k^i \subseteq \mathbf{x}_k$, $\boldsymbol{\theta}_k^i \subseteq \boldsymbol{\theta}_k$, and $\mathbf{y}_k^i \subseteq \mathbf{y}_k$. Here, the local inputs used, $\mathbf{u}^i$, consist of elements from both $\mathbf{u}$ and $\mathbf{y}$, where measured values are directly used as local inputs. The estimation problem is decomposed by finding a set of minimal submodels that together cover the subset of $\mathbf{x}$ and $\boldsymbol{\theta}$ required for prediction, by using these local inputs. This approach to distributed estimation is different from approaches like the distributed decentralized extended Kalman filter (Mutambara, 1998) or other estimation fusion techniques (Sinha et al., 2008) where local estimates are communicated between local estimators. Here, local estimators do not communicate and operate completely independently.

In order to effectively perform joint state-parameter estimation, the system should be observable, among other requirements. If the global model is structurally observable, then we are guaranteed that the local submodels for estimation are as well (Moya et al., 2010).

Any suitable algorithm may be used for joint state-parameter estimation. In this paper, we use an unscented Kalman filter (UKF) (Julier & Uhlmann, 1997, 2004) with a variance control algorithm (Daigle, Saha, & Goebel, 2012). The UKF assumes the general nonlinear form of the state and output equations described in Section 2, but restricted to additive Gaussian noise.

We summarize the main details of the UKF below, and refer the reader to (Julier & Uhlmann, 1997, 2004) for details. In the UKF, distributions are approximated using the unscented transform (UT). The UT takes a random variable $\mathbf{x} \in \mathbb{R}^{n_x}$, with mean $\bar{\mathbf{x}}$ and covariance $\mathbf{P}_{xx}$, that is related to a second random variable $\mathbf{y} \in \mathbb{R}^{n_y}$ by some function $\mathbf{y} = \mathbf{g}(\mathbf{x})$, and computes the mean $\bar{\mathbf{y}}$ and covariance $\mathbf{P}_{yy}$ using a minimal set of *deterministically* selected weighted samples, called *sigma points* (Julier & Uhlmann, 1997). $\boldsymbol{\mathcal{X}}^i$ denotes the ith sigma point from $\mathbf{x}$ and w^i denotes its weight.[2] The sigma points are always chosen such that the mean and covariance match those of the original distribution, $\bar{\mathbf{x}}$ and $\mathbf{P}_{xx}$. Each sigma point is passed through $\mathbf{g}$ to obtain new sigma points $\boldsymbol{\mathcal{Y}}$, i.e.,

$$\boldsymbol{\mathcal{Y}}^i = \mathbf{g}(\boldsymbol{\mathcal{X}}^i)$$

[2] Sigma point weights do not directly represent discrete probabilities, so are not restricted to $[0, 1]$.

with mean and covariance calculated as

$$\bar{\mathrm{y}} = \sum_i w^i \boldsymbol{\mathcal{Y}}^i$$
$$\mathbf{P}_{yy} = \sum_i w^i (\boldsymbol{\mathcal{Y}}^i - \bar{\mathrm{y}})(\boldsymbol{\mathcal{Y}}^i - \bar{\mathrm{y}})^T.$$

In this paper, we use the symmetric unscented transform, in which $2n_x + 1$ sigma points are symmetrically selected about the mean according to (Julier & Uhlmann, 2004):

$$w^i = \begin{cases} \dfrac{\kappa}{(n_x + \kappa)}, & i = 0 \\ \dfrac{1}{2(n_x + \kappa)}, & i = 1, \ldots, 2n_x \end{cases}$$
$$\boldsymbol{\mathcal{X}}^i = \begin{cases} \bar{\mathrm{x}}, & i = 0 \\ \bar{\mathrm{x}} + \left(\sqrt{(n_x+\kappa)\mathbf{P}_{xx}}\right)^i, & i = 1, \ldots, n_x \\ \bar{\mathrm{x}} - \left(\sqrt{(n_x+\kappa)\mathbf{P}_{xx}}\right)^i, & i = n_x+1, \ldots, 2n_x, \end{cases}$$

where $\left(\sqrt{(n_x + \kappa)\mathbf{P}_{xx}}\right)^i$ refers to the ith column of the matrix square root of $(n_x + \kappa)\mathbf{P}_{xx}$. Here, κ is a free parameter that can be used to tune higher order moments of the distribution. If x is assumed Gaussian, then selecting $\kappa = 3 - n_x$ is recommended (Julier & Uhlmann, 1997).

In the filter, first, n_s sigma points $\hat{\boldsymbol{\mathcal{X}}}_{k-1|k-1}$ are derived from the current mean $\hat{\mathrm{x}}_{k-1|k-1}$ and covariance estimates $\mathbf{P}_{k-1|k-1}$ using a sigma point selection algorithm. The prediction step is:

$$\hat{\boldsymbol{\mathcal{X}}}^i_{k|k-1} = \mathrm{f}(\hat{\boldsymbol{\mathcal{X}}}^i_{k-1|k-1}, \mathrm{u}_{k-1}), i = 1, \ldots, n_s$$
$$\hat{\boldsymbol{\mathcal{Y}}}^i_{k|k-1} = \mathrm{h}(\hat{\boldsymbol{\mathcal{X}}}^i_{k|k-1}), i = 1, \ldots, n_s$$
$$\hat{\mathrm{x}}_{k|k-1} = \sum_i^{n_s} w^i \boldsymbol{\mathcal{X}}^i_{k|k-1}$$
$$\hat{\mathrm{y}}_{k|k-1} = \sum_i^{n_s} w^i \boldsymbol{\mathcal{Y}}^i_{k|k-1}$$

with

$$\mathbf{P}_{k|k-1} = \mathbf{Q} + \sum_i^{n_s} w^i (\boldsymbol{\mathcal{X}}^i_{k|k-1} - \hat{\mathrm{x}}_{k|k-1})(\boldsymbol{\mathcal{X}}^i_{k|k-1} - \hat{\mathrm{x}}_{k|k-1})^T,$$

where $\mathbf{Q}$ is the process noise covariance matrix. The update step is:

$$\mathbf{P}_{yy} = \mathbf{R} + \sum_i^{n_s} w^i (\boldsymbol{\mathcal{Y}}^i_{k|k-1} - \hat{\mathrm{y}}_{k|k-1})(\boldsymbol{\mathcal{Y}}^i_{k|k-1} - \hat{\mathrm{y}}_{k|k-1})^T$$
$$\mathbf{P}_{xy} = \sum_i^{n_s} w^i (\boldsymbol{\mathcal{X}}^i_{k|k-1} - \hat{\mathrm{x}}_{k|k-1})(\boldsymbol{\mathcal{Y}}^i_{k|k-1} - \hat{\mathrm{y}}_{k|k-1})^T$$
$$\mathbf{K}_k = \mathbf{P}_{xy}\mathbf{P}_{yy}^{-1}$$
$$\hat{\mathrm{x}}_{k|k} = \hat{\mathrm{x}}_{k|k-1} + \mathbf{K}_k(\mathrm{y}_k - \hat{\mathrm{y}}_{k|k-1})$$
$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{P}_{yy}\mathbf{K}_k^T,$$

where $\mathbf{R}$ is the sensor noise covariance matrix.

Joint state-parameter estimation is accomplished in the UKF by augmenting the state vector with the unknown parameters, and the corresponding diagonal elements of the process noise matrix, $\mathbf{Q}$, are set to nonzero values. In this way, the parameter estimates become time-varying and are modified by the filter using the measured outputs.

The variance values in $\mathbf{Q}$ associated with the unknown parameters determine both the rate of parameter estimation convergence and the estimation performance once convergence is achieved, therefore, techniques have been developed to tune this value online to maximize performance, e.g., (Liu & West, 2001; Orchard, Tobar, & Vachtsevanos, 2009; Daigle, Saha, & Goebel, 2012). We adopt the approach presented in (Daigle, Saha, & Goebel, 2012), in which the algorithm tries to control the variance of the hidden wear parameter estimate to a user-specified range by modifying the process noise variance. Effectively, the algorithm increases the variance when the relative parameter spread is below the desired level, and decreases it otherwise. With the proper settings, the parameter estimates converge quickly and track with high accuracy and precision.

4. Distributed Prediction

Each local prediction module takes as input local state-parameter estimates formed from the local estimators, as discussed in Section 2. The required estimates must be constructed from the local estimates of the submodels used for estimation. A prediction submodel has a set of states X_i and parameters Θ_i, and it must construct a local distribution $p(\mathrm{x}^i_k, \boldsymbol{\theta}^i_k | \mathrm{y}^i_{0:k})$. To do this, we assume that the local state-parameter estimates may be sufficiently represented by a mean $\boldsymbol{\mu}^i$ and covariance matrix $\boldsymbol{\Sigma}^i$. For each prediction submodel $\mathcal{M}_i$, we combine the estimates from estimation submodels that estimate states and parameters in $X_i \cup \Theta_i$ into $\boldsymbol{\mu}^i$ and covariance $\boldsymbol{\Sigma}^i$. If there is overlap in the state-parameter estimates, i.e., if two submodels both estimate the same state variable x or parameter θ, then this may be resolved by a number of techniques, e.g., taking the estimate with the smallest variance, or taking an average. Note that, due to the decomposition into independent local submodels,

Algorithm 1 EOL Prediction

Inputs: $\{(\mathbf{x}_{k_P}^{i(j)}, \boldsymbol{\theta}_{k_P}^{i(j)}), w_{k_P}^{i(j)}\}_{j=1}^{N}$
Outputs: $\{EOL_{k_P}^{i(j)}, w_{k_P}^{i(j)}\}_{j=1}^{N}$
for $j = 1$ **to** N **do**
  $k \leftarrow k_P$
  $\mathbf{x}_k^{i(j)} \leftarrow \mathbf{x}_{k_P}^{i(j)}$
  $\boldsymbol{\theta}_k^{i(j)} \leftarrow \boldsymbol{\theta}_{k_P}^{i(j)}$
  Predict $\hat{\mathbf{u}}_k^i$
  while $T_{EOL}^i(\mathbf{x}_k^{i(j)}, \boldsymbol{\theta}_k^{i(j)}, \hat{\mathbf{u}}_k^i) = 0$ **do**
    Predict $\hat{\mathbf{u}}_k^i$
    $\boldsymbol{\theta}_{k+1}^{i(j)} \sim p(\boldsymbol{\theta}_{k+1}^i | \boldsymbol{\theta}_k^{i(j)})$
    $\mathbf{x}_{k+1}^{i(j)} \sim p(\mathbf{x}_{k+1}^i | \mathbf{x}_k^{i(j)}, \boldsymbol{\theta}_k^{i(j)}, \hat{\mathbf{u}}_k^i)$
    $k \leftarrow k + 1$
    $\mathbf{x}_k^{i(j)} \leftarrow \mathbf{x}_{k+1}^{i(j)}$
    $\boldsymbol{\theta}_k^{i(j)} \leftarrow \boldsymbol{\theta}_{k+1}^{i(j)}$
  end while
  $EOL_{k_P}^{i(j)} \leftarrow k$
end for

we recover only an approximation to the joint posterior distribution as would have been found by a global estimator. In particular, covariance information is lost due to the decoupling and will appear as zeros in the merged covariance matrix. As shown in (Daigle et al., 2011) and as will be seen in Section 5, the approximation still results in accurate predictions.

Given the mean and covariance information, we represent the distribution with a set of sigma points derived using the unscented transform. Then, as in (Daigle & Goebel, 2010), each sigma point is simulated forward to EOL, and we recover the statistics of the EOL distribution given by the sigma points.

The prediction algorithm is executed for each submodel i, deriving local EOL predictions using its local threshold function based on the local EOL constraints. The pseudocode for the prediction procedure is given as Algorithm 1 (Daigle & Goebel, 2011b). For a given submodel $\mathcal{M}_i$, each sigma point j is propagated forward until $T_{EOL}^i(\mathbf{x}_k^{i(j)}, \boldsymbol{\theta}_k^{i(j)})$ evaluates to 1. The algorithm hypothesizes future inputs $\hat{\mathbf{u}}_k^i$.

Each prediction submodel $\mathcal{M}_i$ computes a local EOL/RUL distribution, i.e., $p(EOL_{k_P}^i | \mathbf{y}_{0:k_P}^i)$ and $p(RUL_{k_P}^i | \mathbf{y}_{0:k_P}^i)$. The system EOL is determined by the minimum of all the local distributions, since T_{EOL} of the system is 1 whenever any of the local constraints are violated, and each local distribution is associated with a subset of these constraints. Specifically, for m prediction submodels,

$$p(EOL_{k_P} | \mathbf{y}_{0:k_P}) = \min(\{p(EOL_{k_P}^i | \mathbf{y}_{0:k_P}^i)\}_{i=1}^m).$$

To compute this, we sample from each local EOL distribution and take the minimum of the local samples. This is repeated many times and the statistics of the global EOL distribution are computed.

5. Case Study

In this section, we apply our system-level prognostics approach to a four-wheeled rover testbed developed at NASA Ames Research Center. We develop a model of the rover, and demonstrate the approach using simulated scenarios.

5.1. Rover Modeling

The rover model was originally presented in (Balaban et al., 2011). In this section we summarize the main features and include some extensions to the model.

The rover consists of a symmetric rigid frame with four independently-driven wheels. The wheel speeds are governed by

$$\dot{\omega}_{FL} = \frac{1}{J_{FL}} \left(\tau_{mFL} - \tau_{fFL} - \tau_{glFL} + \tau_{grFL} \right) \quad (c_1)$$

$$\dot{\omega}_{FR} = \frac{1}{J_{FR}} \left(\tau_{mFR} - \tau_{fFR} - \tau_{glFR} - \tau_{grFR} \right) \quad (c_2)$$

$$\dot{\omega}_{BL} = \frac{1}{J_{BL}} \left(\tau_{mBL} - \tau_{fBL} - \tau_{glBL} + \tau_{grBL} \right) \quad (c_3)$$

$$\dot{\omega}_{BR} = \frac{1}{J_{BR}} \left(\tau_{mBR} - \tau_{fBR} - \tau_{glFR} - \tau_{grBR} \right) \quad (c_4)$$

The F, B, L, and R subscripts stand for *front*, *left*, *back*, and *right*, respectively. Here, for wheel $w \in \{FL, FR, BL, BR\}$, J_w denotes the wheel inertia; $\tau_{mw} = k_\tau i_w$ is the motor torque, where i_w is the motor current and k_τ is an energy transformation gain; $\tau_{fw} = \mu_{fw} \omega_w$ is the wheel friction torque, where μ_{fw} is a friction coefficient; $\tau_{glw} = r_w \mu_{gls} (v_w - v)$ is the torque due to slippage, where r_w is the wheel radius, μ_{gls} is a friction coefficient, v_w is the translational wheel velocity, and v is the translation velocity of the rover body; and $\tau_{grw} = r_w \mu_{grw} \omega \cos\gamma$ is the torque due to the rotational movement of the rover body, where μ_{grw} is a friction coefficient, ω is the rotational velocity of the rover body, and $\gamma = \arctan l/b$ with l being the rover length and b being its width.

We consider here friction-based damage progression in the motors, resulting in an increase in motor friction over time, which will lead to an increase in power consumption. For wheel w, μ_{fw} is governed by (Daigle & Goebel, 2011b)

$$\dot{\mu}_{fFL} = \nu_{fFL}\, \mu_{fFL}\, \omega_{FL}^2 \quad (c_5)$$

$$\dot{\mu}_{fFR} = \nu_{fFR}\, \mu_{fFR}\, \omega_{FR}^2 \quad (c_6)$$

$$\dot{\mu}_{fBL} = \nu_{fBL}\, \mu_{fBL}\, \omega_{BL}^2 \quad (c_7)$$

$$\dot{\mu}_{fBR} = \nu_{fBR}\, \mu_{fBR}\, \omega_{BR}^2, \quad (c_8)$$

where for wheel w, ν_{fw} is an unknown wear coefficient.

The translational velocity v of the rover is described by

$$\dot{v} = \frac{1}{m} \left(F_{glFL} + F_{glFR} + F_{glBL} + F_{glBR} \right), \quad (c_9)$$

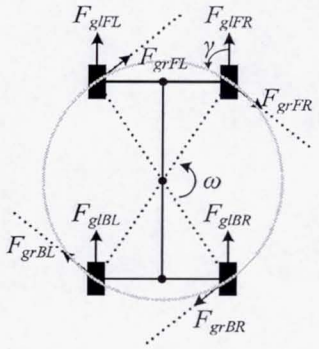

Figure 2. Rover forces.

where m is the rover mass, and for wheel w, $F_{glw} = \mu_{glw}(v_w - v)$ is the force due to slippage. The rotational velocity ω is described by

$$\dot{\omega} = \frac{1}{J}(d\cos\gamma F_{glFR} + d\cos\gamma F_{glBR} - d\cos\gamma F_{glFL} - d\cos\gamma F_{glBL} - dF_{grFL} - dF_{grFR} - dF_{grBL} - dF_{grBR}). \quad (c_{10})$$

Here, J is the rotational inertia of the rover, d is the distance from the center of the rover to each wheel, and for wheel w, $F_{grw} = \mu_{grw}\omega$ is the force due to the rotational movement of the rover body. The rover forces are illustrated in Fig. 2.

The wheels are driven by DC motors with PI control that sets the voltages V applied to the motors. The motor currents are governed by

$$\dot{i}_{FL} = \frac{1}{L}(V_{FL} - i_{FL}R_{FL} - k_\omega\omega_{FL}) \quad (c_{11})$$

$$\dot{i}_{FR} = \frac{1}{L}(V_{FR} - i_{FR}R_{FR} - k_\omega\omega_{FR}) \quad (c_{12})$$

$$\dot{i}_{BL} = \frac{1}{L}(V_{BL} - i_{BL}R_{BL} - k_\omega\omega_{BL}) \quad (c_{13})$$

$$\dot{i}_{BR} = \frac{1}{L}(V_{BR} - i_{BR}R_{BR} - k_\omega\omega_{BR}). \quad (c_{14})$$

Here, L is the motor inductance, k_ω is an energy transformation term, and for wheel w, R is the motor resistance. The voltages applied to the motors are determined by the controllers, where for wheel w, $V_w = P * (u_w - \omega_w) + I * e_{iw}$, where P is a proportional gain, u_w is the commanded wheel speed, I is an integral gain, and e_{iw} is the integral error term. The integral error terms are governed by

$$\dot{e}_{iFL} = u_{FL} - \omega_{FL} \quad (c_{15})$$

$$\dot{e}_{iFR} = u_{FR} - \omega_{FR} \quad (c_{16})$$

$$\dot{e}_{iBL} = u_{BL} - \omega_{BL} \quad (c_{17})$$

$$\dot{e}_{iBR} = u_{BR} - \omega_{BR}. \quad (c_{18})$$

The motor windings heat up as current passes through them. The temperature of the windings for the motors are governed by

$$\dot{T}_{dFL} = \frac{1}{J_d}\left(i_{FL}^2R - h_{dFL}(T_{dFL} - T_{mFL})\right) \quad (c_{19})$$

$$\dot{T}_{dFR} = \frac{1}{J_d}\left(i_{FR}^2R - h_{dFR}(T_{dFR} - T_{mFR})\right) \quad (c_{20})$$

$$\dot{T}_{dBL} = \frac{1}{J_d}\left(i_{BL}^2R - h_{dBL}(T_{dBL} - T_{mBL})\right) \quad (c_{21})$$

$$\dot{T}_{dBR} = \frac{1}{J_d}\left(i_{BR}^2R - h_{dBR}(T_{dBR} - T_{mBR})\right), \quad (c_{22})$$

where J_d is the thermal inertia of the windings, and for wheel w, h_{dw} is a heat transfer coefficient, and T_{mw} is the motor surface temperature. It is assumed that heat is lost only to the motor surface, and that winding resistance R is approximately constant for the temperature range considered. The surface temperature of the motor for wheel w is given by

$$\dot{T}_{mFL} = \frac{1}{J_s}(h_{dFL}(T_{dFL} - T_{mFL}) - h_{aFL}(T_{mFL} - T_a)) \quad (c_{23})$$

$$\dot{T}_{mFR} = \frac{1}{J_s}(h_{dFR}(T_{dFR} - T_{mFR}) - h_{aFR}(T_{mFR} - T_a)) \quad (c_{24})$$

$$\dot{T}_{mBL} = \frac{1}{J_s}(h_{dBL}(T_{dBL} - T_{mBL}) - h_{aBL}(T_{mBL} - T_a)) \quad (c_{25})$$

$$\dot{T}_{mBR} = \frac{1}{J_s}(h_{dBR}(T_{dBR} - T_{mBR}) - h_{aBR}(T_{mBR} - T_a)), \quad (c_{26})$$

where J_s is the thermal inertia of the motor surface, and for wheel w, h_{aw} is a heat transfer coefficient, and T_a is the ambient temperature. Heat is transferred from the windings to the surface and lost to the environment.

The batteries, which are connected in series, are described by a simple electrical circuit equivalent model that includes a large capacitance C_b in parallel with a resistance R_p, together in series with another resistance R_s.[3] The battery charge variables q_i are governed by

$$\dot{q}_1 = -V_1/R_{p1} - (i_{FL} + i_{FR} + i_{BR} + i_{BL}) \quad (c_{27})$$

$$\dot{q}_2 = -V_2/R_{p2} - (i_{FL} + i_{FR} + i_{BR} + i_{BL}) \quad (c_{28})$$

$$\dot{q}_3 = -V_3/R_{p3} - (i_{FL} + i_{FR} + i_{BR} + i_{BL}) \quad (c_{29})$$

$$\dot{q}_4 = -V_4/R_{p4} - (i_{FL} + i_{FR} + i_{BR} + i_{BL}). \quad (c_{30})$$

[3] We use a simple model here only for demonstration purposes. More detailed battery models for prognostics can be found in the literature, e.g., (Saha & Goebel, 2009).

Submodel	X_i	Θ_i	U_i	Y_i	C_i
$\mathcal{M}_1$	q_1	C_{b1}, R_{s1}	i_b^*	V_1^*	c_{27},c_{31},c_{35}
$\mathcal{M}_2$	q_2	C_{b2}, R_{s2}	i_b^*	V_2^*	c_{28},c_{32},c_{35}
$\mathcal{M}_3$	q_3	C_{b3}, R_{s3}	i_b^*	V_3^*	c_{29},c_{33},c_{35}
$\mathcal{M}_4$	q_4	C_{b4}, R_{s4}	i_b^*	V_4^*	c_{30},c_{34},c_{35}
$\mathcal{M}_5$	T_{dFL}, T_{mFL}	h_{dFL}, h_{aFL}	i_{FL}^*	T_{mFL}^*	c_{19},c_{23},c_{36},c_{40}
$\mathcal{M}_6$	T_{dFR}, T_{mFR}	h_{dFR}, h_{aFR}	i_{FR}^*	T_{mFR}^*	c_{20},c_{24},c_{37},c_{41}
$\mathcal{M}_7$	T_{dBL}, T_{mBL}	h_{dBL}, h_{aBL}	i_{BL}^*	T_{mBL}^*	c_{21},c_{25},c_{38},c_{42}
$\mathcal{M}_8$	T_{dBR}, T_{mBR}	h_{dBR}, h_{aBR}	i_{BR}^*	T_{mBR}^*	c_{22},c_{26},c_{39},c_{43}
$\mathcal{M}_9$	i_{FL}, e_{iFL}	$\varnothing$	u_{FL}, ω_{FL}^*	i_{FL}^*	c_{11},c_{15},c_{36},c_{44}
$\mathcal{M}_{10}$	i_{FR}, e_{iFR}	$\varnothing$	u_{FR}, ω_{FR}^*	i_{FR}^*	c_{12},c_{16},c_{37},c_{45}
$\mathcal{M}_{11}$	i_{BL}, e_{iBL}	$\varnothing$	u_{BL}, ω_{BL}^*	i_{BL}^*	c_{13},c_{17},c_{38},c_{46}
$\mathcal{M}_{12}$	i_{BR}, e_{iBR}	$\varnothing$	u_{BR}, ω_{BR}^*	i_{BR}^*	c_{14},c_{18},c_{39},c_{47}
$\mathcal{M}_{13}$	$\omega_{FL}, v, \omega, \mu_{fFL}$	ν_{fFL}	$i_{FL}^*, \omega_{FR}^*, \omega_{BL}^*, \omega_{BR}^*$	ω_{FL}^*	c_1,c_5,c_9,c_{10},c_{36},c_{45},c_{46},c_{47}
$\mathcal{M}_{14}$	$\omega_{FR}, v, \omega, \mu_{fFR}$	ν_{fFR}	$i_{FR}^*, \omega_{FL}^*, \omega_{BL}^*, \omega_{BR}^*$	ω_{FR}^*	c_2,c_6,c_9,c_{10},c_{37},c_{44},c_{46},c_{47}
$\mathcal{M}_{15}$	$\omega_{BL}, v, \omega, \mu_{fBL}$	ν_{fBL}	$i_{BL}^*, \omega_{FL}^*, \omega_{FR}^*, \omega_{BR}^*$	ω_{BL}^*	c_3,c_7,c_9,c_{10},c_{38},c_{44},c_{45},c_{47}
$\mathcal{M}_{16}$	$\omega_{BR}, v, \omega, \mu_{fBR}$	ν_{fBR}	$i_{BR}^*, \omega_{FL}^*, \omega_{FR}^*, \omega_{BL}^*$	ω_{BR}^*	c_4,c_8,c_9,c_{10},c_{39},c_{44},c_{45},c_{46}

Table 1. Estimation Submodels

The available sensors measure the voltages of the batteries,

$$V_1^* = q_1/C_{b1} - R_{s1} * (i_{FL} + i_{FR} + i_{BR} + i_{BL}) \quad (c_{31})$$
$$V_2^* = q_2/C_{b2} - R_{s2} * (i_{FL} + i_{FR} + i_{BR} + i_{BL}) \quad (c_{32})$$
$$V_3^* = q_3/C_{b3} - R_{s3} * (i_{FL} + i_{FR} + i_{BR} + i_{BL}) \quad (c_{33})$$
$$V_4^* = q_4/C_{b4} - R_{s4} * (i_{FL} + i_{FR} + i_{BR} + i_{BL}), \quad (c_{34})$$

the battery current,

$$i_b^* = i_{FL} + i_{FR} + i_{BR} + i_{BL}, \quad (c_{35})$$

the motor currents,

$$i_{FL}^* = i_{FL} \quad (c_{36})$$
$$i_{FR}^* = i_{FR} \quad (c_{37})$$
$$i_{BL}^* = i_{BL} \quad (c_{38})$$
$$i_{BR}^* = i_{BR}, \quad (c_{39})$$

the motor surface temperatures,

$$T_{mFL}^* = T_{mFL} \quad (c_{40})$$
$$T_{mFR}^* = T_{mFR} \quad (c_{41})$$
$$T_{mBL}^* = T_{mBL} \quad (c_{42})$$
$$T_{mBR}^* = T_{mBR}, \quad (c_{43})$$

and the wheel speeds,

$$\omega_{FL}^* = \omega_{FL} \quad (c_{44})$$
$$\omega_{FR}^* = \omega_{FR} \quad (c_{45})$$
$$\omega_{BL}^* = \omega_{BL} \quad (c_{46})$$
$$\omega_{BR}^* = \omega_{BR}. \quad (c_{47})$$

Here, the * superscript indicates a measured value.

We are interested in predicting when any of the rover batteries are at their voltage threshold, beyond which the batteries will be damaged (Saha & Goebel, 2009). The constraints are given as

$$V_1 > V^- \quad (c_{48})$$
$$V_2 > V^- \quad (c_{49})$$
$$V_3 > V^- \quad (c_{50})$$
$$V_4 > V^-, \quad (c_{51})$$

where the voltage threshold is given by $V^- = 9.6$ V, and for battery i, $V_i = q_i/C_{bi} - R_{si} * (i_{FL} + i_{FR} + i_{BR} + i_{BL})$. We are also interested in when the motor temperature gets too high. The motor windings are designed to withstand temperatures up to a certain point, after which, the insulation breaks down, the windings short, and the motor fails (Balaban et al., 2010). The constraints are given as

$$T_{mFL} < T_m^+ \quad (c_{52})$$
$$T_{mFR} < T_m^+ \quad (c_{53})$$
$$T_{mBL} < T_m^+ \quad (c_{54})$$
$$T_{mBR} < T_m^+, \quad (c_{55})$$

where the temperature limit is given by $T_m^+ = 70^\circ$ C. The rover cannot be operated when any of these constraints, c_{48}–c_{55}, are violated.

In the general case, we consider uncertainty in the friction wear parameters ν_{fFL}, ν_{fFR}, ν_{fBL}, and ν_{fBR}; the heat transfer coefficients h_{dFL}, h_{dFR}, h_{dBL}, h_{dBR}, h_{aFL}, h_{aFR}, h_{aBL}, and h_{aBR}; the battery capacitances C_{b1}, C_{b2}, C_{b3}, and C_{b4}; and the battery resistances R_{s1}, R_{s2}, R_{s3}, and R_{s4}. Sensor and process noise were estimated based on data from the actual rover testbed.

5.2. Results

To demonstrate the validity of the approach, we describe two scenarios for system-level prognostics of the rover. In the first, the rover is operating nominally without any faults present, and in the second, friction damage is progressing

Submodel	X_i	Θ_i	U_i	Y_i	C_i
$\mathcal{M}_{17}$	q_1	C_{b1}, R_{s1}	$i_{FL}, i_{FR}, i_{BL}, i_{BR}$	$\varnothing$	c_{27},c_{48}
$\mathcal{M}_{18}$	q_2	C_{b2}, R_{s2}	$i_{FL}, i_{FR}, i_{BL}, i_{BR}$	$\varnothing$	c_{28},c_{49}
$\mathcal{M}_{19}$	q_3	C_{b3}, R_{s3}	$i_{FL}, i_{FR}, i_{BL}, i_{BR}$	$\varnothing$	c_{29},c_{50}
$\mathcal{M}_{20}$	q_4	C_{b4}, R_{s4}	$i_{FL}, i_{FR}, i_{BL}, i_{BR}$	$\varnothing$	c_{30},c_{51}
$\mathcal{M}_{21}$	T_{dFL}, T_{mFL}	h_{dFL}, h_{aFL}	i_{FL}	$\varnothing$	c_{19},c_{23},c_{52}
$\mathcal{M}_{22}$	T_{dFR}, T_{mFR}	h_{dFR}, h_{aFR}	i_{FR}	$\varnothing$	c_{20},c_{24},c_{53}
$\mathcal{M}_{23}$	T_{dBL}, T_{mBL}	h_{dBL}, h_{aBL}	i_{BL}	$\varnothing$	c_{21},c_{25},c_{54}
$\mathcal{M}_{24}$	T_{dBR}, T_{mBR}	h_{dBR}, h_{aBR}	i_{BR}	$\varnothing$	c_{22},c_{26},c_{55}

Table 2. Prediction Submodels Using Motor Currents as Local Inputs

on one motor. In both cases, the rover travels between various waypoints, moving at an average speed of 0.5 m/s. The unknown parameters are initialized incorrectly (with around 10% error) so that the local estimators must converge to the true values. In both cases, the estimation step is performed in a distributed manner using the set of submodels derived by using measured values as local inputs, shown in Table 1. For example, submodel $\mathcal{M}_1$ computes an estimate of V_1^* using the measured value of i_b^* as a local input, and using the minimal set of constraints to do this. For the prediction submodels, as will be shown, the correct submodels to use depends on the scenario, and illustrates when and when not the prediction step can be decomposed.

5.2.1. Nominal Operation

We first consider a scenario involving nominal, fault-free operations. In this case, EOL occurs around 3 h. An RUL prediction is made every 500 s. With the rover traveling at an average speed of 0.5 m/s, the motor currents average to about 0.15 A each and so the total current draining from the four batteries is 0.6 A. Since these values do not vary much during nominal operation, we can use the motor currents as local inputs for the model decomposition. These submodels are shown in Table 2. Note that the estimates from the estimation submodels $\mathcal{M}_1$–$\mathcal{M}_8$ are used directly in the prediction submodels $\mathcal{M}_{17}$–$\mathcal{M}_{24}$, respectively, and that estimation submodels $\mathcal{M}_9$–$\mathcal{M}_{16}$ are not necessary. Note also that the prediction submodels do not compute any outputs, rather, their goal is to compute EOL constraints (e.g., $\mathcal{M}_{17}$ computes c_{48}).

The system-level prediction results are shown in Fig. 3. Predictions from the battery submodels are shown in Fig. 4. In this case, the motor temperatures reach a steady-state that is below the temperature threshold, so only the batteries have an impact on system EOL, which is the minimum of the EOLs predicted for the battery submodels. In particular, it is the first and fourth batteries (corresponding to $\mathcal{M}_{17}$ and $\mathcal{M}_{20}$, respectively) that discharge the fastest, as shown explicitly in Fig. 4. The figures show the means of the predicted RUL distributions, the true RUL, RUL^*, and an accuracy cone of $\alpha = 10\%$ around it. In Fig. 3, we show both the system-level predictions using the distributed approach with $\mathcal{M}_{17}$–$\mathcal{M}_{24}$ and the centralized approach using the global prediction model $\mathcal{M}_0$. The global prediction model contains all the states, parameters, and constraints given in the previous subsection, minus the output constraints, and uses the commanded wheel speeds (known a priori) as hypothesized inputs. Since the currents are also known a priori, the system-level prediction can be decomposed, and the predictions made using the local submodels closely match those made using the global model, as shown in the figure.

We use the relative accuracy (RA) metric (Saxena et al., 2010) for prediction accuracy. Averaged over all predictions, RA is 97.48% for the distributed approach and 98.74% for the centralized approach. Using relative standard deviation (RSD) as a measure of spread, and averaged over all prediction points, RSD is 0.40% for the distributed approach and 0.43% for the centralized approach. The distributed approach is only slightly less accurate but has better precision. Here, both approaches are very accurate since the system state-parameter estimates are very accurate, and there is only a small amount of error associated with assuming a constant average motor current or wheel speed. Correspondingly, the prediction spread is relatively small because the uncertainty in the state-parameter estimate is very small.

5.2.2. Friction Damage Progression

We now consider a scenario in which for the front-left motor, there is nonlinear friction damage progression with $\nu_{fFL} = 1 \times 10^{-4}$ s. As a result of the continuously increasing friction, the current drawn by the motor increases as well in order for the motor controller to maintain the same desired wheel speed. Hence, the total current drawn from the batteries is increased, and EOL occurs around 2 h. Because i_{FL} is constantly changing, and in a way that is dependent on the motor state, it cannot be predicted *a priori*, and so cannot be used as a local input because the resulting predictions will not be accurate. Therefore, we require a submodel that estimates i_{FL}, and we so employ submodels using as local inputs average values for the remaining motor currents, average commanded wheel speeds, and average rover translational velocity v and rotational velocity ω. The prediction submodels for this case are shown in Table 3. For comparison, we demonstrate also prediction using $\mathcal{M}_{17}$–$\mathcal{M}_{24}$, and, for this strategy, at each prediction point the average value of current measured over the last minute is used as the future hypothesized value. Of course, this will not yield accurate results since future values

Submodel	X_i	Θ_i	U_i	Y_i	C_i
$\mathcal{M}_{25}$	$q_1, i_{FL}, e_{iFL}, \omega_{FL}, \mu_{fFL}$	$C_{b1}, R_{s1}, \nu_{fFL}$	$u_{FL}, v, \omega, i_{FR}, i_{BL}, i_{BR}$	$\varnothing$	c_1,c_5,c_{11},c_{15},c_{27},c_{48}
$\mathcal{M}_{26}$	$q_2, i_{FL}, e_{iFL}, \omega_{FL}, \mu_{fFL}$	$C_{b2}, R_{s2}, \nu_{fFL}$	$u_{FL}, v, \omega, i_{FR}, i_{BL}, i_{BR}$	$\varnothing$	c_1,c_5,c_{11},c_{15},c_{28},c_{49}
$\mathcal{M}_{27}$	$q_3, i_{FL}, e_{iFL}, \omega_{FL}, \mu_{fFL}$	$C_{b3}, R_{s3}, \nu_{fFL}$	$u_{FL}, v, \omega, i_{FR}, i_{BL}, i_{BR}$	$\varnothing$	c_1,c_5,c_{11},c_{15},c_{29},c_{50}
$\mathcal{M}_{28}$	$q_4, i_{FL}, e_{iFL}, \omega_{FL}, \mu_{fFL}$	$C_{b4}, R_{s4}, \nu_{fFL}$	$u_{FL}, v, \omega, i_{FR}, i_{BL}, i_{BR}$	$\varnothing$	c_1,c_5,c_{11},c_{15},c_{30},c_{51}
$\mathcal{M}_{29}$	$T_{dFL}, T_{mFL}, i_{FL}, e_{iFL}, \omega_{FL}, \mu_{fFL}$	$h_{dFL}, h_{aFL}, \nu_{fFL}$	u_{FL}, v, ω	$\varnothing$	c_{19},c_{23},c_{52},c_{11},c_{15},c_1,c_5
$\mathcal{M}_{30}$	T_{dFR}, T_{mFR}	h_{dFR}, h_{aFR}	i_{FR}	$\varnothing$	c_{20},c_{24},c_{53}
$\mathcal{M}_{31}$	T_{dBL}, T_{mBL}	h_{dBL}, h_{aBL}	i_{BL}	$\varnothing$	c_{21},c_{25},c_{54}
$\mathcal{M}_{32}$	T_{dBR}, T_{mBR}	h_{dBR}, h_{aBR}	i_{BR}	$\varnothing$	c_{22},c_{26},c_{55}

Table 3. Prediction Submodels Using Commanded Wheel Speeds and Rover Velocities as Local Inputs

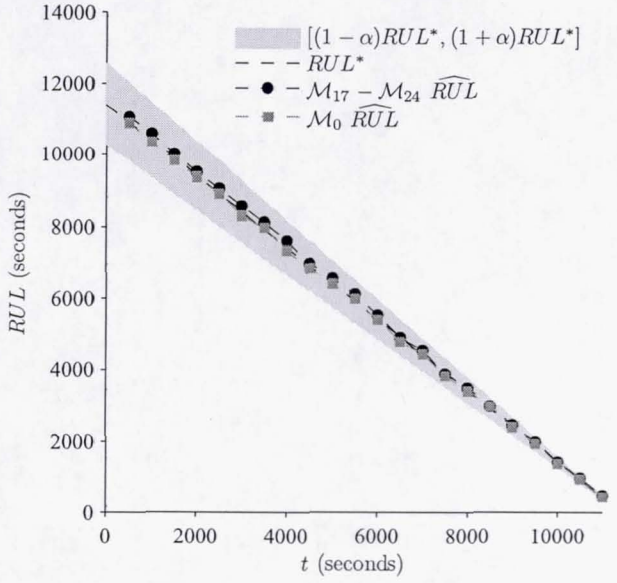

Figure 3. System RUL prediction performance under nominal conditions with $\alpha = 0.1$.

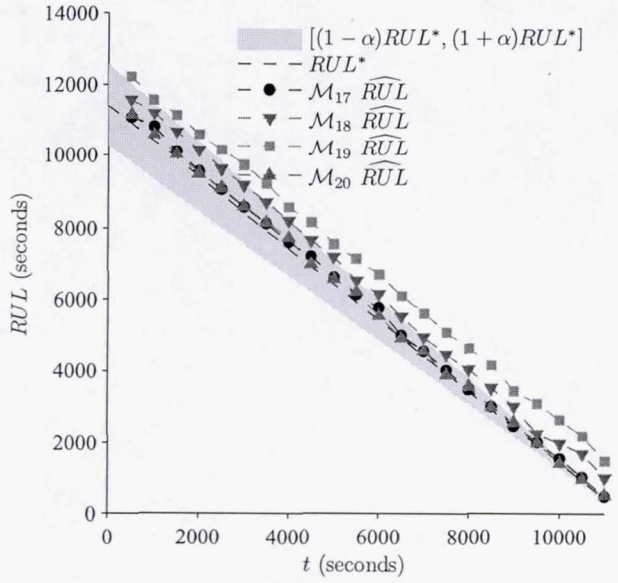

Figure 4. Individual battery submodel RUL prediction performance under nominal conditions with $\alpha = 0.1$.

of the current will actually be larger. Note that the prediction submodels used in this case do not correspond directly to those used for estimation. So, when constructing the estimate for $\mathcal{M}_{25}$, for example, it takes the estimates from $\mathcal{M}_1$, $\mathcal{M}_9$, and $\mathcal{M}_{13}$.

The system-level prediction results are shown in Fig. 5. Although the increased friction causes the temperature of the front-left motor to increase, it is still the batteries discharging that dominates the system-level EOL in this case. We show the predictions using $\mathcal{M}_{17}$–$\mathcal{M}_{24}$, $\mathcal{M}_{25}$–$\mathcal{M}_{32}$, and the global model $\mathcal{M}_0$. For $\mathcal{M}_{25}$–$\mathcal{M}_{32}$, average values of $v = 0.5$ m/s and $\omega = 0$ rad/s are used. Here, the predictions using the latter two approaches are virtually identical (the predictions using $\mathcal{M}_{25}$–$\mathcal{M}_{32}$ are hidden under those for $\mathcal{M}_0$), and fairly accurate. In contrast, as expected, the predictions using $\mathcal{M}_{17}$–$\mathcal{M}_{24}$ are very inaccurate, and only converge towards the true RUL at the very end. This quite effectively demonstrates that, in this scenario, it is incorrect to use the front-left motor current as a local input for predictions, since it cannot be predicted independently from the front-left motor submodel, and therefore a submodel that itself predicts this current is required to obtain accurate predictions.

Here, RA averages to 58.95% using $\mathcal{M}_{17}$–$\mathcal{M}_{24}$, 94.24% using $\mathcal{M}_{25}$–$\mathcal{M}_{32}$, and 94.32% using $\mathcal{M}_0$. RSD averages to 0.76% using $\mathcal{M}_{17}$–$\mathcal{M}_{24}$, 1.62% using $\mathcal{M}_{25}$–$\mathcal{M}_{32}$, 1.73% using $\mathcal{M}_0$. Here, we also observe an increase in prediction spread using the centralized approach with only a slight increase in accuracy over the distributed approach.[4] Overall, accuracy and precision are both decreased compared to the nominal scenario because there is more uncertainty in the state-parameter estimate, specifically, that dealing with the estimate of ν_{fFL}. This uncertainty in the state-parameter estimate contributes to the additional uncertainty in the RUL predictions.

[4]The RSD for $\mathcal{M}_{17}$–$\mathcal{M}_{24}$ is the lowest because those submodels do not include the motor friction component, so do not have the additional uncertainty associated with the estimation of the wear parameter.

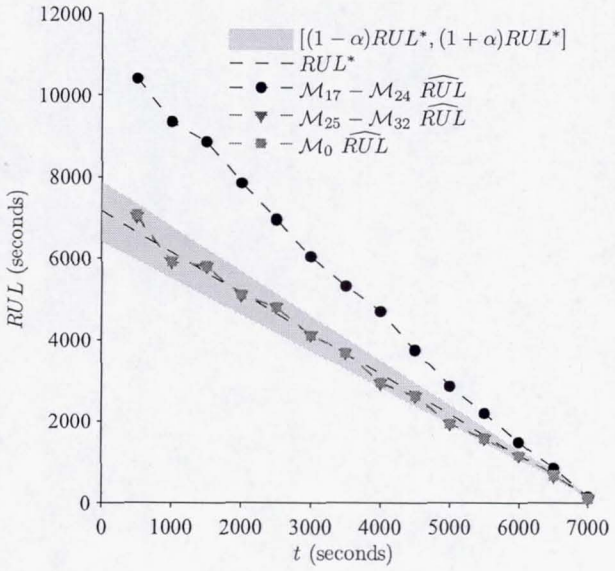

Figure 5. System RUL prediction performance with friction damage progression with $\alpha = 0.1$.

6. Conclusions

In this paper, we formulated the system-level prognostics problem and proposed a distributed solution based on structural model decomposition. Using a four-wheeled rover as a simulation-based case study, we demonstrated the effectiveness of the approach. Most importantly, the distributed approach allows for, in many practical circumstances, the decomposition of the system-level prognostics problem into component-level prognostics problems and provides a mechanism to merge local prognostics results into a system-level result. Further, since the local subproblems are independent, this allows component experts to focus on prognostics solutions for their components. However, we showed also that this approach is not always possible if accurate results are desired, since in some cases the prediction problem cannot be so easily decomposed, and it depends crucially on correct assumptions about what variables may serve as local inputs for the prediction problem.

Although in this paper we focused on the model-based prognostics paradigm, our approach is flexible in that data-driven algorithms may be used also, once the local subproblems are defined. For example, in previous works, structural model decomposition was used to automatically design gray box diagnosis models that were implemented using different data-driven techniques (for instance, state space neural networks in (Pulido, Zamarreno, Merino, & Bregon, 2012) or machine learning techniques in (Alonso-Gonzalez, Rodríguez, Prieto, & Pulido, 2008)). By decomposing the system-level problem into independent subproblems through structural model decomposition, we can apply similar ideas to solve each prognostics subproblem by using the most appropriate technique, whether it is a model-based, data-driven, or hybrid approach.

An important direction of future work is in algorithms for optimal placement of sensors for model decomposition, because the level of model decomposition that can be achieved is dependent on the number of sensors and where they are placed. This results in the most efficient decomposition of the system-level prognostics problem. Current work also addresses combining the distributed prognostics framework with a distributed diagnostic approach for integrated diagnostics and prognostics (Bregon, Daigle, & Roychoudhury, 2012).

Acknowledgments

M. Daigle 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 MCI TIN2009-11326 grant.

References

Alonso-Gonzalez, C. A., Rodríguez, J. J., Prieto, O., & Pulido, B. (2008, September). Machine learning and model-based diagnosis using possible conflicts and system decomposition. In *Proc. of the 19th international workshop on principles of diagnosis* (p. 215-222). Blue Mountains, Australia.

Balaban, E., Narasimhan, S., Daigle, M., Celaya, J., Roychoudhury, I., Saha, B., et al. (2011, September). A mobile robot testbed for prognostics-enabled autonomous decision making. In *Annual conference of the prognostics and health management society* (p. 15-30). Montreal, Canada.

Balaban, E., Saxena, A., Narasimhan, S., Roychoudhury, I., Goebel, K., & Koopmans, M. (2010, September). Airborne electro-mechanical actuator test stand for development of prognostic health management systems. In *Annual conference of the prognostics and health management society.*

Bolander, N., Qiu, H., Eklund, N., Hindle, E., & Rosenfeld, T. (2010, October). Physics-based remaining useful life prediction for aircraft engine bearing prognosis. In *Proceedings of the annual conference of the prognostics and health management society 2010.*

Bregon, A., Biswas, G., & Pulido, B. (2012, May). A decomposition method for nonlinear parameter estimation in TRANSCEND. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, 42*(3), 751-763.

Bregon, A., Daigle, M., & Roychoudhury, I. (2012, July). An integrated model-based distributed diagnosis and prognosis framework. In *Proceedings of the 23rd interna-*

tional workshop on principles of diagnosis.

Byington, C. S., Watson, M., Edwards, D., & Stoelting, P. (2004, March). A model-based approach to prognostics and health management for flight control actuators. In *Proceedings of the 2004 ieee aerospace conference* (Vol. 6, pp. 3551–3562).

Celaya, J. R., Kulkarni, C., Biswas, G., Saha, S., & Goebel, K. (2011, September). A model-based prognostics methodology for electrolytic capacitors based on electrical overstress accelerated aging. In *Proceedings of the annual conference of the prognostics and health management society 2011.*

Daigle, M., Bregon, A., & Roychoudhury, I. (2011, September). Distributed damage estimation for prognostics based on structural model decomposition. In *Proceedings of the annual conference of the prognostics and health management society 2011* (p. 198-208).

Daigle, M., Bregon, A., & Roychoudhury, I. (2012). *Distributed prognostics based on structural model decomposition.* (Manuscript submitted for publication.)

Daigle, M., & Goebel, K. (2010, October). Improving computational efficiency of prediction in model-based prognostics using the unscented transform. In *Proc. of the annual conference of the prognostics and health management society 2010.*

Daigle, M., & Goebel, K. (2011a, August). A model-based prognostics approach applied to pneumatic valves. *International Journal of Prognostics and Health Management*, *2*(2).

Daigle, M., & Goebel, K. (2011b, March). Multiple damage progression paths in model-based prognostics. In *Proceedings of the 2011 ieee aerospace conference.*

Daigle, M., Saha, B., & Goebel, K. (2012, March). A comparison of filter-based approaches for model-based prognostics. In *Proceedings of the 2012 ieee aerospace conference.*

Julier, S. J., & Uhlmann, J. K. (1997). A new extension of the Kalman filter to nonlinear systems. In *Proceedings of the 11th international symposium on aerospace/defense sensing, simulation and controls* (pp. 182–193).

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

Liu, J., & West, M. (2001). Combined parameter and state estimation in simulation-based filtering. *Sequential Monte Carlo Methods in Practice*, 197–223.

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

Moya, N., Biswas, G., Alonso-Gonzalez, C. J., & Koutsoukos, X. (2010, October). Structural observability: Application to decompose a system with possible conflicts. In *Proceedings of the 21st international workshop on principles of diagnosis* (p. 241-248).

Mutambara, A. G. (1998). *Decentralized estimation and control for multisensor systems*. Boca Raton: CRC Press.

Orchard, M., Tobar, F., & Vachtsevanos, G. (2009, December). Outer feedback correction loops in particle filtering-based prognostic algorithms: Statistical performance comparison. *Studies in Informatics and Control*(4), 295-304.

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

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

Pulido, B., Zamarreno, J., Merino, A., & Bregon, A. (2012, July). *Using structural decomposition methods to design gray-box models for fault diagnosis of complex industrial systems: a beet sugar factory case study.*

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.*

Saha, B., Saha, S., & Goebel, K. (2009). A distributed prognostic health management architecture. In *Proceedings of the 2009 conference of the society for machinery failure prevention technology.*

Saxena, A., Celaya, J., Saha, B., Saha, S., & Goebel, K. (2010). Metrics for offline evaluation of prognostic performance. *International Journal of Prognostics and Health Management*, *1*(1).

Sinha, A., Chen, H., Danu, D., Kirubarajan, T., & Farooq, M. (2008). Estimation and decision fusion: A survey. *Neurocomputing*, *71*(13), 2650–2656.