Distributed Prognostics Based on Structural Model Decomposition

Matthew J. Daigle, *Member, IEEE,* Anibal Bregon, *Member, IEEE,* and Indranil Roychoudhury, *Member, IEEE*

***Abstract*—Within systems health management, prognostics focuses on predicting the remaining useful life of a system. In the model-based prognostics paradigm, physics-based models are constructed that describe the operation of a system and how it fails. Such approaches consist of an estimation phase, in which the health state of the system is first identified, and a prediction phase, in which the health state is projected forward in time to determine the end of life. Centralized solutions to these problems are often computationally expensive, do not scale well as the size of the system grows, and introduce a single point of failure. In this paper, we propose a novel distributed model-based prognostics scheme that formally describes how to decompose both the estimation and prediction problems into independent local subproblems whose solutions may be easily composed into a global solution. The decomposition of the prognostics problem is achieved through structural decomposition of the underlying models. The decomposition algorithm creates from the global system model a set of local submodels suitable for prognostics. Independent local estimation and prediction problems are formed based on these local submodels, resulting in a scalable distributed prognostics approach that allows the local subproblems to be solved in parallel, thus offering increases in computational efficiency. Using a centrifugal pump as a case study, we perform a number of simulation-based experiments to demonstrate the distributed approach, compare the performance with a centralized approach, and establish its scalability.**

***Index Terms*—model-based prognostics, distributed prognostics, structural model decomposition**

Abbreviations & Acronyms

EOL	end of life
PRMSE	percent root mean square error
RA	relative accuracy
RPM	revolutions per minute
RSD	relative standard deviation
RUL	remaining useful life
UKF	unscented Kalman filter
UT	unscented transform

Corresponding author. M. J. Daigle is with NASA Ames Research Center, Moffett Field, CA 94035 USA (e-mail: matthew.j.daigle@nasa.gov).

A. Bregon is with the Department of Computer Science, University of Valladolid, Valladolid, Spain (e-mail: anibal@infor.uva.es).

I. Roychoudhury is with Stinger Ghaffarian Technologies, at NASA Ames Research Center, Moffett Field, CA 94035 USA (e-mail: indranil.roychoudhury@nasa.gov).

M. Daigle and I. Roychoudhury's work has been partially supported by the NASA System-wide Safety Assurance (SSAT) Project within the Aviation Safety Program (ASP) under the Aeronautics Mission Directorate (ARMD), and the NASA Autonomous Cryogenic Loading Operations (ACLO) Project under the Office of the Chief Technologies (OCT). A. Bregon's work has been partially supported by Spanish MCI TIN2009-11326 grant.

Notation

$\mathbf{x}$	state vector
$\boldsymbol{\theta}$	parameter vector
$\mathbf{u}$	input vector
$\mathbf{y}$	output vector
r	performance requirement
$\mathcal{R}$	set of performance requirements
ω	rotational velocity
τ	torque
p	pressure or probability
Q	volumetric flow
T	temperature
r	friction coefficient
w	wear parameter
$\mathcal{M}$	model/submodel
v	variable
V	voltage or set of variables
X	set of states
Θ	set of parameters
U	set of inputs
Y	set of outputs
c	constraint
C	set of constraints
ϵ_c	equation of constraint c
α	causal assignment
$\mathcal{A}$	set of causal assignments

I. Introduction

Systems health management is an engineering discipline that seeks to improve the design and operation of complex systems in the presence of faults and degradations. Prognostics is an essential technology for systems health management that centers on predicting the useful life of components, subsystems, or systems. This information may be used to slow damage progression, prolong system life, and optimize maintenance activities. Model-based prognostics approaches capture knowledge of how a system and its components fail through the use of physics-based models that describe the underlying physical phenomena [1]–[7]. These algorithms consist of two parts: (*i*) *estimation*, which computes the current joint state-parameter estimate of the system to determine the current health state, and (*ii*) *prediction*, which projects the current joint state-parameter estimate forward in time to determine end of life (EOL) and/or remaining useful life (RUL).

To date, virtually all prognostics approaches employ a centralized architecture. However, centralized approaches have several drawbacks: they embody a single point of failure, are computationally expensive, and do not scale well as the size of the system increases. Distributed architectures, on the other hand, offer several advantages. In particular, implementation platforms are becoming increasingly distributed, involving systems of smart sensors and smart components, in addition to multi-core processors [8]. Distributed approaches naturally take advantage of these new architectural paradigms, and hence improve scalability and computational efficiency. Distributed implementations on large systems are also easier to maintain when components are added or removed from the system.

Specifically, we propose a novel distributed prognostics approach that exploits structural model decomposition [9]. In a model-based prognostics paradigm, the prognostics problem is defined by the underlying model. So, by decomposing the system model, we decompose the model-based prognostics problem. Several methods for structural model decomposition have, in fact, been developed for the purposes of diagnosis [10]–[13], but none for prognosis. In this work, we adopt the model decomposition framework developed previously in [14]. Like other structural model decomposition approaches, the key feature of the derived submodels is that they are *computationally independent*. Therefore, local prognostics problems based on the submodels can be solved independently. As a result, solution of the subproblems requires no communication between the algorithms. This approach also provides more flexibility, allowing different algorithms to be applied to each subproblem, and, thus, each subproblem can be solved with the most appropriate strategy. The proposed approach to distributed prognostics developed in this paper is a fundamentally different approach from previous distributed prognostics approaches, e.g. [15], [16]. In such approaches, the global problem is still solved, and the computation is simply distributed, whereas in our approach, the approach is distributed by decomposing the global problem into local subproblems that are solved in parallel.

In earlier work [17], preliminary results were presented in which only the estimation problem was decomposed using structural model decomposition as described in [10]. In this paper, we show how the more general model decomposition framework of [14] can be used to decompose both the estimation and prediction problems for model-based prognostics. The work of [14] shows how the estimation and prediction problems can be decomposed, however, it does not provide any algorithms for distributed prognostics. In this paper, we develop a distributed prognostics architecture based on the derived submodels, and includes the algorithms for distributed prognostics. The model decomposition corresponds to a set of *local* estimation and prediction problems that are smaller, and, therefore, easier to solve and require less computation than the *global* problem. Each local estimator computes a local joint state-parameter estimate. The local predictors use the outputs of the local estimators as inputs to their prediction routine, yielding local EOL/RUL predictions. Global EOL/RUL predictions are then determined based on the local EOL/RUL predictions.

We demonstrate our distributed prognostics methodology on a centrifugal pump that is used for liquid oxygen transfer in spacecraft fueling operations at Kennedy Space Center [2]. We derive submodels for local estimation and for local prediction, forming a distributed prognostics architecture for the pump. We perform a number of simulation-based experiments, and show that our distributed approach performs comparably to the centralized approach in terms of accuracy and precision of the life predictions, and at decreased computational cost. In addition, since the value of a distributed approach is only readily apparent with large-scale systems, we demonstrate the improved scalability of the distributed approach using a large-scale system composed of multiple pumps.

The contributions of the paper are as follows: (*i*) a novel distributed model-based prognostics framework based upon structural model decomposition, including algorithms for distributed estimation and prediction and the merging of local results into global results; (*ii*) the application of the distributed prognostics approach to a centrifugal pump with comprehensive simulation-based experimental results that validate the approach and compare to a centralized approach; and (*iii*) a proof that the distributed approach is more efficient and scalable than the centralized approach, with corroborating experimental results.

The paper is organized as follows. Section II formally defines the prognostics problem, describes the centralized prognostics architecture, and introduces our proposed distributed architecture. Section III describes the modeling methodology and develops the centrifugal pump model. Section IV overviews the model decomposition framework. Section V formulates the distributed prognostics problem using model decomposition, and provides an architecture for the pump. Section VI describes distributed estimation, and Section VII describes distributed prediction. Section VIII provides results from simulation-based experiments, evaluates the approach, and shows its scalability. Section IX compares our approach with related work, and Section X concludes the paper.

II. Model-based Prognostics

In this section we first formulate the model-based prognostics problem. We then describe the typical centralized prognostic architecture, followed by our proposal for a distributed architecture.

A. Problem Formulation

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

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

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]

In prognostics, we are interested in the time at which the performance of a system lies outside some desired region

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

of acceptable behavior. Outside this region, we say that the system has failed. The desired performance is expressed through a set of n_r *requirements*, $\mathcal{R} = \{r_i\}_{i=1}^{n_r}$, where $r_i : \mathbb{R}^{n_x} \times \mathbb{R}^{n_\theta} \times \mathbb{R}^{n_u} \to \mathbb{B}$ 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 $r_i(\mathbf{x}(t), \boldsymbol{\theta}(t), \mathbf{u}(t)) = 1$ if the requirement is satisfied, and $r_i(\mathbf{x}(t), \boldsymbol{\theta}(t), \mathbf{u}(t)) = 0$ if the requirement is not satisfied.

These individual requirements are combined into a single *threshold function* $T_{EOL} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_\theta} \to \mathbb{B}$, defined as

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

That is, T_{EOL} evaluates to 1, i.e., the system has failed, when any of the requirements are violated [2]. 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. RUL is expressed using EOL as

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

B. Centralized Architecture

In order to compute EOL, we need the current state of the system, which is unknown. Therefore, in the model-based prognostics paradigm, the problem of predicting EOL/RUL is split into two sequential problems: (*i*) estimation, which computes the state-parameter estimate, and (*ii*) prediction, which simulates the current joint state-parameter estimate forward in time to determine EOL/RUL [1], [2], [4].

The centralized architecture implementing the model-based prognostics approach works as follows. In discrete time k, the system receives inputs $\mathbf{u}_k$ and provides measured outputs $\mathbf{y}_k$. With the system model, the estimation module uses this information to compute an estimate $p(\mathbf{x}_k, \boldsymbol{\theta}_k | \mathbf{y}_{0:k})$, accounting for the presence of process noise $\mathbf{v}(t)$ and sensor noise $\mathbf{n}(t)$. Given this state-parameter estimate, the prediction algorithm uses the model to simulate this distribution out to EOL to compute $p(EOL_{k_P} | \mathbf{y}_{0:k_P})$ and $p(RUL_{k_P} | \mathbf{y}_{0:k_P})$ at given prediction times k_P. In order to do this, the prediction step must hypothesize the future inputs to the system $\mathbf{u}_k$ for $k \geq k_P$.

The centralized approach solves the *global* prognostics problem by solving global estimation and prediction problems. Centralized approaches, however, introduce a host of potential problems. Aside from the fact that most modern computational architectures are distributed, be it through multi-core processors or networked systems, the most significant issue is scalability. As the size of the problem increases, the methods to solve it become more and more costly, and can suffer from problems such as convergence of the estimates. We therefore propose a distributed approach that solves the global problem through a set of *local* subproblems.

C. Distributed Architecture

The key idea of the distributed approach is to decompose the global model into a set of local submodels, with each local submodel defining a local estimation or prediction subproblem. For a given model, we generate a set of submodels with local variables $\mathbf{x}^i \subseteq \mathbf{x}$, $\boldsymbol{\theta}^i \subseteq \boldsymbol{\theta}$, $\mathbf{u}^i \subseteq \mathbf{u}$, $\mathbf{y}^i \subseteq \mathbf{y}$, $\mathcal{R}^i \subseteq \mathcal{R}$, and with local equations $\mathbf{f}^i$, $\mathbf{h}^i$, and T_{EOL}^i. Once the submodels have been defined, the distributed architecture works as follows. In discrete time k, the system is provided with inputs $\mathbf{u}_k$ and provides measured outputs $\mathbf{y}_k$. The inputs $\mathbf{u}_k$ and the outputs $\mathbf{y}_k$ are split into local inputs $\mathbf{u}_k^i$ and outputs $\mathbf{y}_k^i$. Local estimators compute $p(\mathbf{x}_k^i, \boldsymbol{\theta}_k^i | \mathbf{y}_{0:k}^i)$. From the local estimates, the inputs to the local predictors are constructed. 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 times k_P. Local predictions are then merged into global predictions $p(EOL_{k_P} | \mathbf{y}_{0:k_P})$ and $p(RUL_{k_P} | \mathbf{y}_{0:k_P})$.

Models are decomposed by selecting some set of variables as local inputs in addition to the global inputs $\mathbf{u}$ to form $\mathbf{u}^i$. In this way, we can derive submodels that can be computed independently of other submodels given the local inputs. This means that local estimation and prediction subproblems are independent and can be solved in parallel without communication. For the estimation phase, the measured sensor signals can be used as additional inputs, exploiting the redundancy they provide [18]. For the prediction phase, the insight is to select, as local inputs, variables that can be predicted a priori (e.g., a controlled quantity). The sets of local inputs chosen for estimation and prediction may be different, in which case the resulting submodels used for estimation and prediction will be different, so the estimates required for prediction must be reconstructed from the results of the local estimators.

Because the subproblems are smaller than the global problem and can be solved in parallel, this approach is more efficient and scalable than a centralized approach, as will be shown in Section VIII. However, the distributed approach does have some limitations. For the estimation phase, information will be lost due to the decomposition (specifically, the covariance of decoupled variables), and noisy sensor signals will be used as local inputs. Therefore, the distributed approach will not obtain the same exact answer as the centralized approach. We will show in Section VIII that, despite these limitations, the performance of the distributed approach is comparable to the performance of the centralized approach. Before presenting the details of the distributed prognostics approach, in the next section, we introduce the pump case study and its model.

III. Centrifugal Pump Modeling

In this work, we use a centrifugal pump as a case study. The particular pump under study is used to transfer liquid oxygen for spacecraft fueling operations, the model for which was originally presented in [2]. In practice, distributed prognostics may not be warranted for a single component, but we use the pump as a case study here because it is a complex but small enough system to fully describe and demonstrate our approach. In Section VIII, we will investigate the scalability properties of our approach using a large-scale multi-pump system.

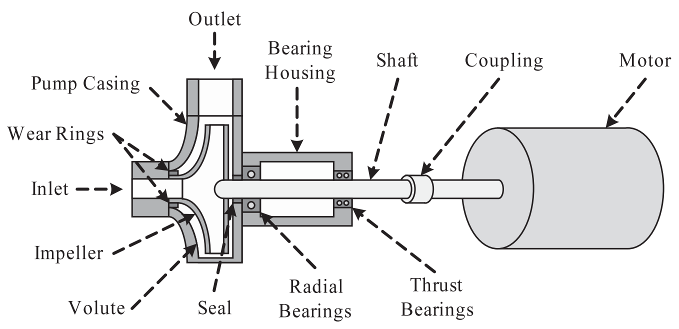


Fig. 1. Centrifugal pump.

In order to apply model-based prognostics, we must develop a model of the system under consideration. This includes identifying the state vector $\mathbf{x}(t)$, the parameter vector $\boldsymbol{\theta}(t)$, the input vector $\mathbf{u}(t)$, the output vector $\mathbf{y}(t)$, the state equation $\mathbf{f}$, the output equation $\mathbf{h}$, and the set of performance requirements $\mathcal{R}$. In this section, we summarize the main features of the pump model, first describing the nominal model, and then describing its damage progression models.

A. Nominal Model

Centrifugal pumps are used for the delivery of fluids in a system. A schematic is shown in Fig. 1. Fluid enters the inlet, and the impeller rotation, driven by an electric motor, forces it through the outlet. Bearings help minimize friction along the shaft, and are lubricated by oil residing in the bearing housing. The pump state includes $\omega(t)$, the rotational velocity of the pump; $Q(t)$, the discharge flow; $T_t(t)$, the thrust bearing temperature; $T_r(t)$ the radial bearing temperature; and $T_o(t)$, the oil temperature.

The rotational velocity of the pump is described using a torque balance,

$$\dot{\omega} = \frac{1}{J}\left(\tau_e - r\omega - \tau_L\right), \tag{1}$$

where J is the lumped motor/pump inertia, τ_e is the electromagnetic torque provided by the motor, r is the lumped friction parameter, and τ_L is the load torque. We assume the pump is driven by an induction motor, in which torque is produced only when there is a slip, s, between the synchronous speed of the supply voltage, ω_s and the mechanical rotation, ω:

$$s = \frac{\omega_s - \omega}{\omega_s}. \tag{2}$$

The expression for the torque τ_e is derived from an equivalent circuit representation for a three-phase induction motor [19]:

$$\tau_e = \frac{npR_2}{s\omega_s}\frac{V^2}{(R_1 + R_2/s)^2 + (\omega_s L_1 + \omega_s L_2)^2}, \tag{3}$$

where R_1 is the stator resistance, L_1 is the stator inductance, R_2 is the rotor resistance, L_2 is the rotor inductance, n is the number of phases (typically 3), p is the number of magnetic pole pairs, and V is the applied rms motor voltage. The dependence of torque on slip creates a feedback loop that causes the rotor to follow the rotation of the magnetic field. Rotor speed is controlled by changing ω_s, e.g., through the use of a variable-frequency drive, which will also change V to manage power usage.

Load torque τ_L is a polynomial function of the pump flow rate and the impeller rotational velocity [20], [21]:

$$\tau_L = a_0\omega^2 + a_1\omega Q - a_2 Q^2, \tag{4}$$

where a_0, a_1, and a_2 are coefficients derived from the pump geometry [21].

The rotation of the impeller creates a pressure difference from the inlet to the outlet of the pump, which drives the pump flow, Q. The pump pressure is computed as

$$p_p = b_0\omega^2 + b_1\omega Q - b_2 Q^2, \tag{5}$$

where b_0, b_1, and b_2 are coefficients derived from the pump geometry. The parameter b_0 is proportional to impeller area A [22]. Flow through the impeller, Q_i, is computed using the pressure differences:

$$Q_i = c\sqrt{|p_s + p_p - p_d|}\,sign(p_s + p_p - p_d), \tag{6}$$

where c is a flow coefficient, p_s is the suction pressure, and p_d is the discharge pressure. To account for fluid inertia, the discharge flow is described by

$$\dot{Q} = \frac{1}{J_Q}(Q_o - Q_i), \tag{7}$$

where J_Q is the flow inertia.

Pump temperatures are often monitored as indicators of pump condition. The oil heats up due to the radial and thrust bearings and cools to the environment:

$$\dot{T}_o = \frac{1}{J_o}(H_{o,1}(T_t - T_o) + H_{o,2}(T_r - T_o) - H_{o,3}(T_o - T_a)), \tag{8}$$

where J_o is the thermal inertia of the oil, and the $H_{o,i}$ terms are heat transfer coefficients. The thrust bearings heat up due to the friction between the pump shaft and the bearings, and cool to the oil and the environment:

$$\dot{T}_t = \frac{1}{J_t}(r_t\omega^2 - H_{t,1}(T_t - T_o) - H_{t,2}(T_t - T_a)), \tag{9}$$

where J_t is the thermal inertia of the thrust bearings, r_t is the friction coefficient for the thrust bearings, and the $H_{t,i}$ terms are heat transfer coefficients. The radial bearings behave similarly:

$$\dot{T}_r = \frac{1}{J_r}(r_r\omega^2 - H_{r,1}(T_r - T_o) - H_{r,2}(T_r - T_a)), \tag{10}$$

where J_r is the thermal inertia of the radial bearings, r_r is the friction coefficient for the radial bearings, and the $H_{r,i}$ terms are heat transfer coefficients.

The overall input vector $\mathbf{u}$ is given by

$$\mathbf{u}(t) = \begin{bmatrix} p_s(t) & p_d(t) & T_a(t) & V(t) & \omega_s(t) \end{bmatrix}^T. \tag{11}$$

The available pump sensors form the measurement vector $\mathbf{y}$ given by

$$\mathbf{y}(t) = \begin{bmatrix} \omega(t) & Q(t) & T_o(t) & T_t(t) & T_r(t) \end{bmatrix}^T. \tag{12}$$

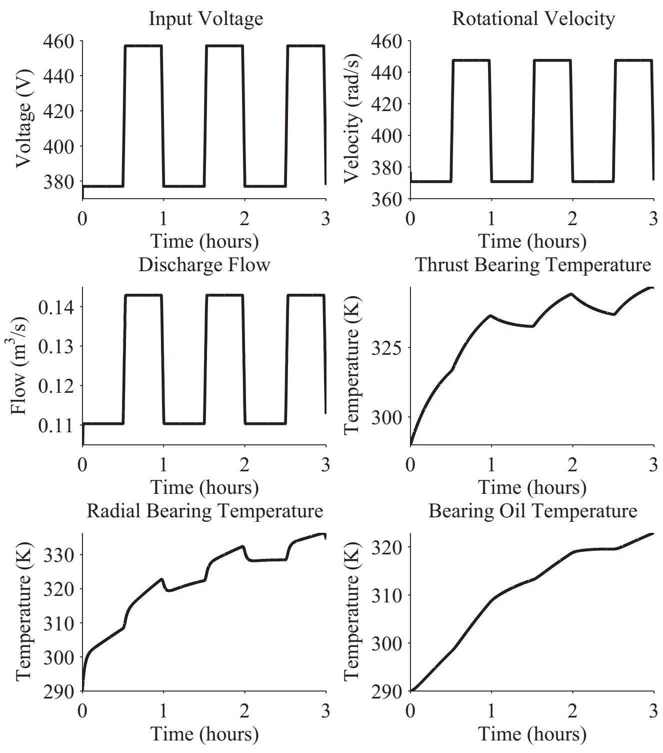


Fig. 2. Nominal pump operation.

Fig. 2 shows nominal pump operation, with the parameters given in Table I. The input voltage and line frequency are varied to control the pump speed (commanded line frequency is equal to the observed pump speed). Initially, slip is 1 so an electromagnetic torque is produced, causing a rotation of the motor to match the rotation of the magnetic field, with a small amount of slip remaining. The pump rotation creates fluid flow and heats up the bearings.

B. Damage Modeling

The performance requirements of the pump are specified by efficiency and temperature limits:

$$\eta > \eta^- \tag{13}$$
$$T_o < T_o^+ \tag{14}$$
$$T_t < T_t^+ \tag{15}$$
$$T_r < T_r^+, \tag{16}$$

where the $^-$ superscript denotes a minimum and the $^+$ superscript denotes a maximum, and efficiency η is defined as $\eta = \frac{VI}{(p_d - p_s)Q}$ for nominal inputs (I is rms motor current). We take $\eta^- = 0.75\eta_0$, where η_0 is the nominal efficiency. When the maximum temperatures are reached, irreversible damage occurs. Here, we use $T_o^+ = 333$ K, $T_t^+ = 370$ K, and $T_r^+ = 370$ K.

The most significant damage mechanism for pumps is impeller wear. It is represented as a decrease in impeller area A [22], [23]. Since the impeller area is proportional to b_0, a decrease in impeller area causes a decrease in the pump pressure, and, hence, the pump efficiency. We use the erosive

TABLE I
NOMINAL PUMP PARAMETERS

Parameter	Value
$\omega(0)$	376 rad/s
J	50 kg m^2
r	8.0×10^{-3} N m s
n	3 phases
p	1 pole pair
R_1	3.6×10^{-1} Ω
R_2	7.6×10^{-2} Ω
$L_1 + L_2$	6.3×10^{-4} H
$Q(0)$	0 m^3/s
a_0	1.5×10^{-3} kg m^2
a_1	5.8 kg/m
a_2	9.2×10^3 kg/m^4
$b_0(0)$	12.7 kg/m
b_1	1.8×10^4 kg/m^4
b_2	0 kg/m^7
c	8.2×10^{-5} m$^{7/2}$/kg$^{1/2}$
c_l	1.0×10^{-10} m$^{7/2}$/kg$^{1/2}$
J_Q	5.0 s^{-1}
$T_o(0)$	290 K
J_o	8.0×10^3 K/J/s
$H_{o,1}$	1.0 W/K
$H_{o,2}$	3.0 W/K
$H_{o,3}$	1.5 W/K
$T_r(0)$	290 K
J_r	2.4 K/J/s
$r_r(0)$	1.8×10^{-6} N m s
$H_{r,1}$	1.8×10^{-3} W/K
$H_{r,2}$	2.0×10^{-2} W/K
$T_t(0)$	290 K
J_t	7.3 K/J/s
$r_t(0)$	1.4×10^{-6} N m s
$H_{t,1}$	3.4×10^{-3} W/K
$H_{t,2}$	2.6×10^{-2} W/K

wear equation [24] to describe how the impeller area changes over time. The erosive wear rate is proportional to fluid velocity times friction force. Fluid velocity is proportional to volumetric flow rate, and friction force is proportional to fluid velocity. We lump the proportionality constants into the wear coefficient w_A to obtain [2]

$$\dot{A} = -w_A Q_i^2. \tag{17}$$

Because A is proportional to b_0, then $\dot{b}_0 = k\dot{A} = -kw_A Q_i^2$, so we estimate b_0 and $w_{b_0} = kw_A$.

Another significant damage mechanism for pumps is bearing wear, which is captured as an increase in the friction coefficient. Sliding and rolling friction generate wear of material which increases the coefficient of friction [2], [3], [24]:

$$\dot{r}_t = w_t r_t \omega^2, \tag{18}$$
$$\dot{r}_r = w_r r_r \omega^2, \tag{19}$$

where w_t and w_r are the wear coefficients. The slip compensation provided by the electromagnetic torque generation masks small changes in friction, so it is only with very large increases that a change in ω will be observed. Changes in friction manifest more strongly in the bearing temperatures, eventually driving them to the temperature limits.

So, the full state vector is

$$\mathbf{x}(t) = \begin{bmatrix}\omega(t) & Q(t) & T_o(t) & T_t(t) & T_r(t) & b_0(t) & r_t(t) & r_r(t)\end{bmatrix}^T. \tag{20}$$

The initial conditions for b_0, r_t, and r_r are given in Table I. The wear parameters form the unknown parameter vector, i.e.,

$$\boldsymbol{\theta}(t) = \begin{bmatrix}w_{b_0} & w_t & w_r\end{bmatrix}^T. \tag{21}$$

IV. Model Decomposition

To decompose the problem of model-based prognostics, we decompose the underlying structural model. We use the model decomposition framework described in [14], but simplify it, without loss of generality, by removing the notion of auxiliary variables (intermediate variables derived from the states, parameters, and inputs). This simplifies the model decomposition algorithm and allows us to make guarantees of the minimality of the derived submodels.

We introduce the requisite notation and concepts of the model decomposition framework in the following. Additional details and the full version of the framework can be found in [14]. We begin with the definition of a *model*.

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

Input variables $u \in U$ are known or measured, and correspond to the input signals $\mathbf{u}(t)$. The subset of the outputs corresponding to the (measured) sensor signals $\mathbf{y}(t)$ are denoted as $Y^* \subseteq Y$. Parameters $\theta \in \Theta$ include explicit model parameters corresponding to $\boldsymbol{\theta}(t)$ that are used in the model constraints. Θ consists only of those parameters that are to be made explicit for joint state-parameter estimation.

As shown in Defn. 1, a constraint $c = (\varepsilon_c, V_c)$ includes an equation ε_c over the set of variables V_c. These constraints are essentially representative of the vector functions $\mathbf{f}$ and $\mathbf{h}$, along with the requirements $\mathcal{R}$. We associate explicit variables for the evaluation of the performance requirements, e.g., $e_i = r_i(\mathbf{x}(t), \boldsymbol{\theta}(t), \mathbf{u}(t))$. Note that, typically, a given r_i is only a function of a subset of the states, parameters, and inputs. Here, the e_i variables become part of Y, and are not included in Y^*. We denote by $E \subset Y$ the variable set associated with the performance requirement evaluations.

For the pump model, we have the variable sets $X = \{\omega$, Q, T_o, T_t, T_r, b_0, r_t, $r_r\}$, $\Theta = \{w_{b_0}$, w_t, $w_r\}$, $U = \{p_s$, p_d, T_a, V, $\omega_s\}$, and $Y = \{\omega^*$, Q^*, T_o^*, T_t^*, T_r^*, e_1, e_2, e_3, $e_4\}$. Here, $Y^* = \{\omega^*$, Q^*, T_o^*, T_t^*, $T_r^*\}$ and $E = \{e_1$, e_2, e_3, $e_4\}$, where the variables e_1 to e_4 correspond to the requirements described in Eq. 13 to 16. The * superscript is used on output variables that are associated with sensors.

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 denote explicitly the causal (i.e., computational) direction.

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 $\alpha = (c, v)$.

A *causal model* is a model extended with a valid set of causal assignments.

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

For the pump model, the causal constraints are as follows. For the states, we have

$$\omega := \int_0^t \dot{\omega}\ dt, \tag{α_1}$$

$$Q := \int_0^t \dot{Q}\ dt, \tag{α_2}$$

$$T_o := \int_0^t \dot{T}_o\ dt, \tag{α_3}$$

$$T_t := \int_0^t \dot{T}_t\ dt, \tag{α_4}$$

$$T_r := \int_0^t \dot{T}_r\ dt, \tag{α_5}$$

$$b_0 := \int_0^t -w_{b_0} Q_i^2\ dt \tag{α_6}$$

$$r_t := \int_0^t \dot{r}_t\ dt, \tag{α_7}$$

$$r_r := \int_0^t \dot{r}_r\ dt, \tag{α_8}$$

where $\dot{\omega}$ is given by Eq. 1, $\dot{Q}$ by Eq. 7, $\dot{T}_o$ by Eq. 8, $\dot{T}_t$ by Eq. 9, $\dot{T}_r$ by Eq. 10, Q_i by Eq. 6, r_t by Eq. 18, and r_r by Eq. 19. The initial conditions are provided in Table I. For the outputs, we have

$$\omega^* := \omega \tag{α_9}$$

$$Q^* := Q, \tag{α_{10}}$$

$$T_o^* := T_o, \tag{α_{11}}$$

$$T_t^* := T_t, \tag{α_{12}}$$

$$T_r^* := T_r. \tag{α_{13}}$$

For the performance requirements, we have

$$e_1 := (\eta > \eta^-), \tag{α_{14}}$$

$$e_2 := (T_o < T_o^+), \tag{α_{15}}$$

$$e_3 := (T_t < T_t^+), \tag{α_{16}}$$

$$e_4 := (T_r < T_r^+). \tag{α_{17}}$$

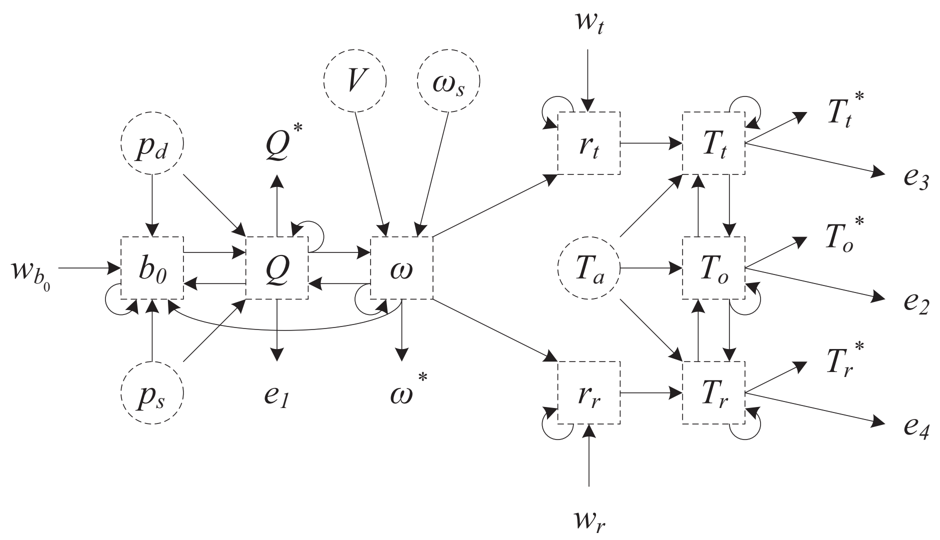


Fig. 3. Causal graph for the pump model.

We visualize a causal model $\mathcal{M}$ using a directed graph $\mathcal{G} = (N, A)$, where N is the set of nodes corresponding directly to the variables V in $\mathcal{M}$, and A is the set of arcs, where for every $(c, v_c^{out}) \in \mathcal{A}$, we include an arc (v', v_c^{out}) for each $v' \in V_c - \{v_c^{out}\}$. The causal graph corresponding to the pump model is given in Fig. 3. In the graph, we mark inputs with dashed circles and states with dashed squares.

In order 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 its variable values independently from all other submodels. Further, if the local input values are exactly the same as the corresponding variables in the global model, the values of local outputs for the submodel will exactly reproduce the values of the corresponding variables in the global model. A causal submodel can be defined as follows.

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

When using outputs (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 $\mathcal{A}$. All remaining causal assignments in $\mathcal{A}_i$ will still be found in $\mathcal{A}$.

The procedure for generating a submodel from a causal model is given as Algorithm 1. 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^*. We provide here a simplified version of the algorithm presented in [14], and refer the reader to [14] for the extended algorithm and additional details. We briefly summarize the algorithm below.

Algorithm 1 $\mathcal{M}_i = \texttt{GenerateSubmodel}(\mathcal{M}, U^*, V^*)$

```
1: $V_i \leftarrow V^*$
2: $C_i \leftarrow \varnothing$
3: $\mathcal{A}_i \leftarrow \varnothing$
4: $variables \leftarrow V^*$
5: while $variables \neq \varnothing$ do
6:     $v \leftarrow \texttt{pop}(variables)$
7:     $c \leftarrow \texttt{GetBestConstraint}(v, V_i, U^*, \mathcal{A})$
8:     $C_i \leftarrow C_i \cup \{c\}$
9:     $\mathcal{A}_i \leftarrow \mathcal{A}_i \cup \{(c, v)\}$
10:    for all $v' \in V_c$ do
11:        if $v' \notin V_i$ and $v' \notin \Theta$ and $v' \notin U^*$ then
12:            $variables \leftarrow variables \cup \{v'\}$
13:        end if
14:        $V_i \leftarrow V_i \cup \{v'\}$
15:    end for
16: end while
17: $\mathcal{M}_i \leftarrow (V_i, C_i, \mathcal{A}_i)$
```

In Algorithm 1, the $variables$ queue represents the set of variables that have been added to the submodel but have not yet been resolved, i.e., they cannot yet be computed by the submodel. This queue is initialized to V^*, the set of variables that must be computed by the submodel. The algorithm then iterates until this queue has been emptied, i.e., the submodel can compute all variables in V^* using only variables in U^*. For each variable v that must be resolved, we use the `GetBestConstraint` subroutine (Subroutine 2) to find the constraint that should be used to resolve v in the *minimal* (in the number of constraints) way.

The `GetBestConstraint` subroutine (simplified from [14]) tries to find a constraint that *completely* resolves the variable, i.e., resolves v without further backward propagation (all other variables involved in the constraint are in $V_i \cup \Theta \cup U^*$). Such a constraint may be the one that computes v in the current causality, if all needed variables are already in the submodel (in V_i) or are available local inputs

Subroutine 2 $c = \texttt{GetBestConstraint}(v, V_i, U^*, \mathcal{A})$

```
1:  $c_v \leftarrow$ find $c$ where $(c, v) \in \mathcal{A}$
2:  if $(V_{c_v} - \{v\}) \subseteq V_i \cup U^*$ then
3:      return $c_v$
4:  else
5:      for all $y \in Y^* \cap U^*$ do
6:          $c_y \leftarrow$ find $c$ where $(c, y) \in \mathcal{A}$
7:          if $v \in V_{c_y}$ and $(V_{c_y} - \{v\}) \subseteq V_i \cup U^*$ then
8:              return $c_y$
9:          end if
10:     end for
11: end if
12: return $c_v$
```

(in U^*); or such a constraint may be one that computes a measured output $y^* \in U^*$, in which case the causality will be modified such that y^* becomes an input, i.e., the constraint in the new causality will compute v rather than y^*. If no such constraint exists, then the constraint that computes v in the current causal assignment is chosen, and further backward propagation will be necessary.

For example, consider generating a submodel for the pump with $U^* = \{V, \omega_s, Q^*\}$ and $V^* = \{\omega^*\}$. We first try to resolve ω^* (see Fig. 3). To compute ω^* in the given causality we need to include ω in the submodel. To compute ω we need to include V, ω_s, Q, and ω. Both V and ω_s are in U^*, and ω is in V, so these variables are resolved. To compute Q, we have two options: compute using p_d, b_0, p_s, Q, and ω, or compute using Q^* with the corresponding constraint in the causality such that Q is computed. The minimal resolution is the second option, so Q^* is added to the submodel and the causality of the involved constraint is modified. Since Q^* is in U^*, it is resolved. Now all variables in the submodel are resolved and the algorithm is complete.

Clearly, there are many submodels that compute any given V^* using a given U^*. The global model is one such solution. Algorithm 1 finds a minimal submodel that satisfies this, which is guaranteed in Subroutine 2 by resolving a variable without further backward propagation whenever possible. There may be multiple submodels that are equally minimal (i.e., due to a choice of which local input to use), and the algorithm returns the first that it finds.

The algorithm also generates only *complete* submodels, i.e., the submodels contain at least the variables needed to compute its V^*. This is guaranteed because the algorithm only stops propagation at variables included in $V_i \cup \Theta \cup U^*$ [14].

In the worst case, the algorithm must visit all variables and constraints. On each variable, Subroutine 2 is called, which in the worst case considers all variables in $Y \cap U^*$. So the overall worst-case time complexity is $O((|V| + |E|) \cdot |Y \cap U^*|)$. Since $(Y \cap U^*) \subset V$, the algorithm is polynomial in the model size. On average some amount of decomposition will be possible so the complexity will be much lower in practice.

In the next section we describe how this model decomposition algorithm is used to decompose a model for the explicit purposes of distributed estimation and distributed prediction.

V. Distributed Prognostics Architecture

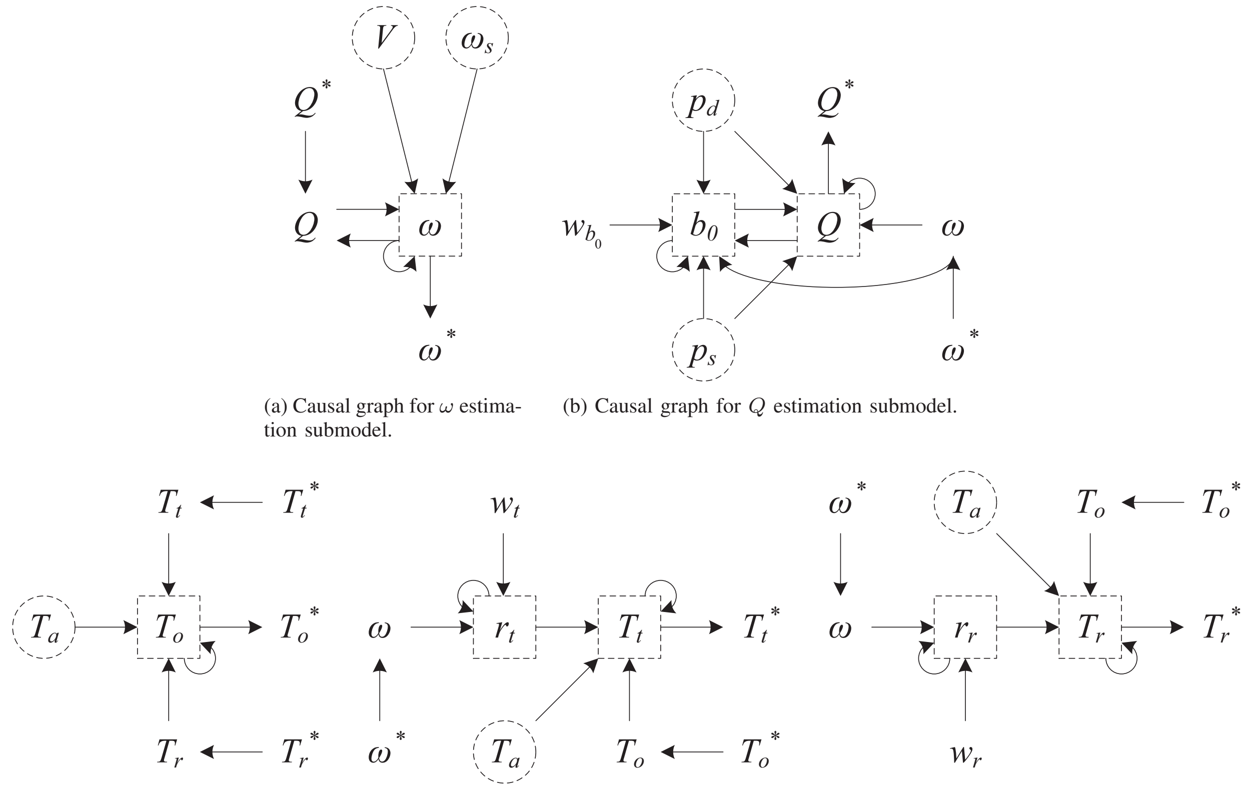
The distributed model-based prognostics architecture is based on structural model decomposition, with local estimation and prediction subproblems based on derived local submodels. For estimation, we construct minimal submodels, one for each output of the model that corresponds to a sensor, i.e., for each $y^* \in Y^*$. As discussed in Section II-C, we use measured sensor signals as local inputs in addition to U. For each output $y^* \in Y^*$, we create a submodel using $\texttt{GenerateSubmodel}(\mathcal{M}, U \cup (Y^* - \{y^*\}), \{y^*\})$, i.e., we use as local inputs the inputs to the global model along with all sensor outputs except for y^*, and the only local output is y^*. We define a local estimator based on that local submodel (e.g., Kalman filter, unscented Kalman filter, particle filter, etc.).

Using noisy sensors as local inputs to the estimation subproblems may, of course, result in a loss of accuracy and robustness of the local estimators. This is the cost of deriving independent local estimators. Without sensor noise, the local estimators would produce the same results as a centralized estimator, and as noise is added, performance may degrade. Note of course that the centralized estimator must deal also with sensor noise and so its performance will degrade as well. In Section VIII we investigate the effects of increased sensor noise on estimation performance for both the distributed and centralized cases. In the situation where some sensors are unreliable or extremely noisy, they can be removed from the set of local inputs. To improve robustness, multi-output estimators can also be derived, instead of the proposed single-output estimators, but setting $V^* \subseteq Y^*$ (the global model can be recovered by setting $V^* = Y^*$).

The causal graphs for the resulting submodels for the pump are shown in Fig. 4. We obtain five submodels. The submodel for ω^* has $X = \{\omega\}$ and $\Theta = \varnothing$; the submodel for Q^* has $X = \{Q, b_0\}$ and $\Theta = \{w_{b_0}\}$; the submodel for T_o^* has $X = \{T_o\}$ and $\Theta = \varnothing$; the submodel for T_t^* has $X = \{T_t, r_t\}$ and $\Theta = \{w_t\}$; and the submodel for T_r^* has $X = \{T_r, r_r\}$ and $\Theta = \{w_r\}$. Note that the estimation submodels do not contain the performance requirements, as these are not required to compute the outputs, so are not included in the submodels derived by the decomposition algorithm.

If the measurement set changed, then the resulting local submodels for estimation would change also. For example, consider a reduced measurement set of $Y^* = \{\omega^*, Q^*, T_t^*, T_r^*\}$, i.e., T_o^* is no longer measured. Then only four submodels would result, one for each sensor. The submodels for w^* and Q^* remain the same, but since T_o^* can no longer be used as a local input, the T_t^* and T_r^* submodels would have to include the T_o state and instead use T_r^* and T_t^* as local inputs, respectively. The corresponding causal graphs are shown in Fig. 5.

Prediction requires hypothesizing future inputs to the system. Therefore, when selecting local inputs for model decomposition, we can select only those variables that can be predicted a priori. For example, for the pump, we can use ω as a local input because it is a controlled variable, and we know what the future controlled values are. On the other hand, T_r cannot be used, since it is evolving due to r_t, which



(a) Causal graph for ω estimation submodel.

(b) Causal graph for Q estimation submodel.

(c) Causal graph for T_o estimation submodel.

(d) Causal graph for T_t estimation submodel.

(e) Causal graph for T_r estimation submodel.

Fig. 4. Causal graphs for pump estimation submodels.

is changing in time depending on w_t, and T_o, which is in turn affected by T_t and r_t which are changing in time due to w_t. The fault propagation among the pump temperatures prohibits the use of any of the temperature variables being used as local inputs. If no variables exist that can be predicted a priori outside of U, then the prediction problem cannot be decomposed, and the global model must be used for prediction.

Besides U, local inputs can come from X or Y. In some cases, it is advantageous to add additional "virtual" outputs if these variables can be predicted a priori but are not already in X or Y, to include in U^*. We construct for each performance requirement a submodel that evaluates the requirement. Because EOL is reached when any one of the performance requirements are violated, we can evaluate them independently to obtain local EOL distributions and then take the minimum to get the global EOL distribution. For each variable $e \in E$, we create a submodel using $\texttt{GenerateSubmodel}(\mathcal{M}, U_P, \{e\})$, where $U_P \subseteq X \cup U \cup Y \supseteq U$ is the set of variables that can be predicted a priori.

For the pump model, U_P consists of U and ω, as just mentioned. The causal graphs for the resulting submodels are shown in Fig. 6. We obtain one submodel associated with the efficiency requirement, and three submodels associated with the temperature requirements. For the temperature requirements, however, aside from the performance requirement included, each submodel is exactly the same (e.g., the submodel for e_2 is that corresponding to the causal graph in Fig. 6b with the e_3 and e_4 variables and constraints removed). This means that the temperatures cannot be decomposed (due to the fault propagation between them). Therefore, we merge these three submodels into one submodel, to avoid unnecessary computation, using $\texttt{GenerateSubmodel}(\mathcal{M}, U_P, \{e_2, e_3, e_4\})$.

As in the centralized scheme, the prediction algorithm uses the state-parameter estimates as input. So, 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 must construct a local distribution $p(\mathbf{x}_k^i, \boldsymbol{\theta}_k^i | \mathbf{y}_{0:k}^i)$ from the estimates provided by the local estimators. 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 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 we take the average value for common means and covariances (alternate strategies may also be used). Some covariance information lost due to the decoupling will appear as zeros in the recovered covariance matrix. Each prediction submodel 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 global EOL is determined by the minimum of all the local distributions, since T_{EOL} is 1 whenever any of the local constraints are violated.

The distributed prognostics architecture for the pump is shown in Fig. 7. Here we had derived five submodels for the

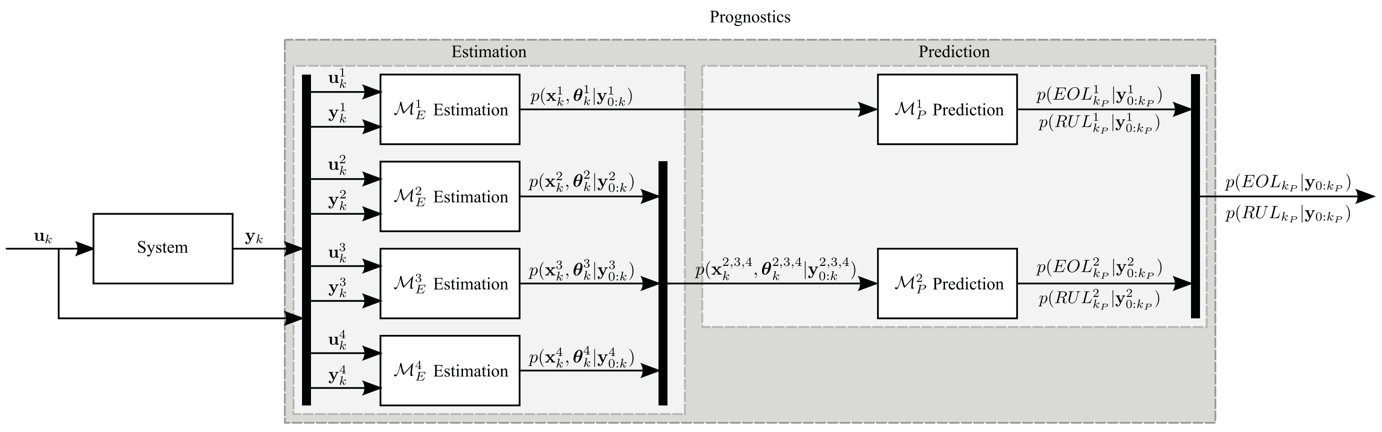
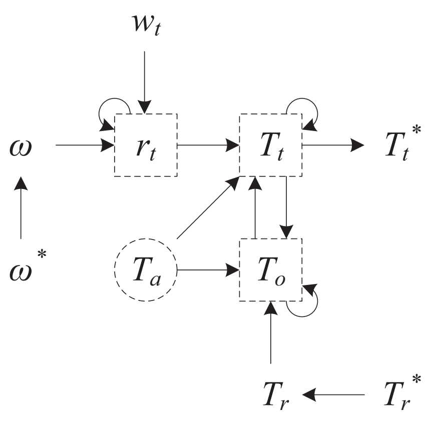
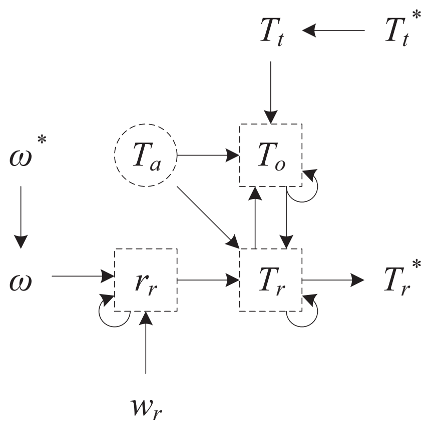

Prognostics
Estimation
Prediction
$\mathbf{u}_k$
System
$\mathbf{y}_k$
$\mathbf{u}_k^1$
$\mathbf{y}_k^1$
$\mathcal{M}_E^1$ Estimation
$p(\mathbf{x}_k^1, \boldsymbol{\theta}_k^1 | \mathbf{y}_{0:k}^1)$
$\mathbf{u}_k^2$
$\mathbf{y}_k^2$
$\mathcal{M}_E^2$ Estimation
$p(\mathbf{x}_k^2, \boldsymbol{\theta}_k^2 | \mathbf{y}_{0:k}^2)$
$\mathbf{u}_k^3$
$\mathbf{y}_k^3$
$\mathcal{M}_E^3$ Estimation
$p(\mathbf{x}_k^3, \boldsymbol{\theta}_k^3 | \mathbf{y}_{0:k}^3)$
$\mathbf{u}_k^4$
$\mathbf{y}_k^4$
$\mathcal{M}_E^4$ Estimation
$p(\mathbf{x}_k^4, \boldsymbol{\theta}_k^4 | \mathbf{y}_{0:k}^4)$
$p(\mathbf{x}_k^{2,3,4}, \boldsymbol{\theta}_k^{2,3,4} | \mathbf{y}_{0:k}^{2,3,4})$
$\mathcal{M}_P^1$ Prediction
$p(EOL_{k_P}^1 | \mathbf{y}_{0:k_P}^1)$
$p(RUL_{k_P}^1 | \mathbf{y}_{0:k_P}^1)$
$\mathcal{M}_P^2$ Prediction
$p(EOL_{k_P}^2 | \mathbf{y}_{0:k_P}^2)$
$p(RUL_{k_P}^2 | \mathbf{y}_{0:k_P}^2)$
$p(EOL_{k_P} | \mathbf{y}_{0:k_P})$
$p(RUL_{k_P} | \mathbf{y}_{0:k_P})$

Fig. 7. Distributed prognostics architecture for the pump.

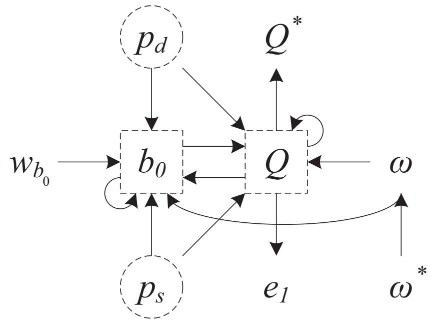


(a) Causal graph for T_t estimation submodel.

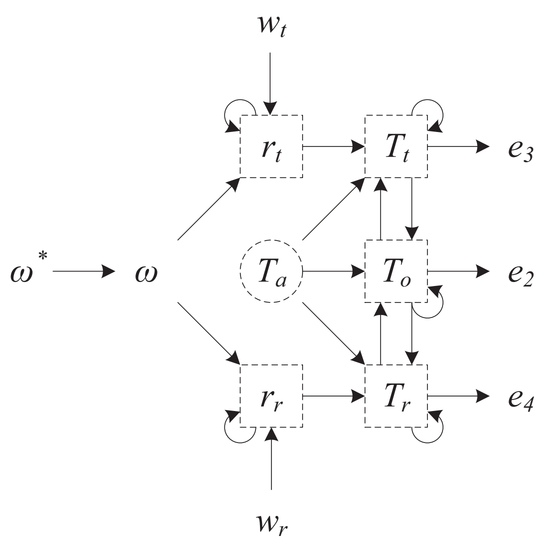


(b) Causal graph for T_r estimation submodel.

Fig. 5. Causal graphs for pump estimation submodels when T_o^* is not measured.



(a) Causal graph for flow prediction submodel.



(b) Causal graph for temperature requirements submodel.

Fig. 6. Causal graphs for pump prediction submodels.

purposes of estimation, and two for prediction. We find that the submodel estimating ω^* is actually not needed, because the only state it estimates is ω (see Fig. 4a), and this state is not required by any of the prediction submodels (see Fig. 6). Therefore, we need only four submodels on which to base our local estimators, $\mathcal{M}_E^1$ that estimates Q^*, $\mathcal{M}_E^2$ that estimates T_o^*, $\mathcal{M}_E^3$ that estimates T_t^*, and $\mathcal{M}_E^4$ that estimates T_r^*. For prediction, we have $\mathcal{M}_P^1$ that predicts the violation of the efficiency requirement, and $\mathcal{M}_P^2$ that predicts the violation of the temperature requirements.

The global inputs and outputs are first split into the local inputs and outputs based on the U_i and Y_i of the submodels derived for estimation. For example, $\mathcal{M}_E^1$ uses as inputs p_s, p_d, and ω^* (the measured value of ω), and computes a single output, Q^*. The local estimates are computed. $\mathcal{M}_P^2$ builds its local state using the estimates of T_o from the $\mathcal{M}_E^2$ estimator, w_t, r_t, and T_t from the $\mathcal{M}_E^3$ estimator, and w_r, r_r, and T_r from the $\mathcal{M}_E^4$ estimator. The local predictors compute the local EOL/RUL predictions, with the predictor for $\mathcal{M}_P^1$ computing the EOL for the efficiency requirement, and the predictor for $\mathcal{M}_P^2$ computing the EOL for the temperature requirements. The local predictions are then merged into the global prediction. The next two sections describe the algorithms used for local estimation and local prediction.

VI. Distributed Estimation

As described in Section V, in our distributed estimation scheme, the local estimator for each submodel $\mathcal{M}_E^i$ produces a local estimate $p(\mathbf{x}_k^i, \boldsymbol{\theta}_k^i | \mathbf{y}_{0:k})$, where $\mathbf{x}_k^i \subseteq \mathbf{x}_k$ and $\boldsymbol{\theta}_k^i \subseteq \boldsymbol{\theta}_k$. Any suitable algorithm may be used for joint state-parameter estimation on any of the local subproblems.

In this paper, we use an unscented Kalman filter (UKF) [25], [26] with a variance control algorithm [27] for the estimation problems. The UKF assumes the general nonlinear form of the state and output equations described in Section II, but restricted to additive Gaussian noise. The pump model satisfies these constraints.

We review here the UKF, and refer the reader to [25], [26] for details. The UKF approximates a distribution 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 $\mathbf{P}_{xx}$, which is related to a second random variable $\mathbf{y}$ by some nonlinear function $\mathbf{y} = \mathbf{g}(\mathbf{x})$, and computes the mean $\bar{\mathbf{y}}$ and covariance $\mathbf{P}_{yy}$ using a (small) set of *deterministically* selected weighted samples, called *sigma points* [25]. $\boldsymbol{\mathcal{X}}^i$ denotes the ith sigma point from $\mathbf{x}$ and w^i denotes its weight. 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.,

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

with mean and covariance calculated as

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

We use here the symmetric unscented transform, in which $2n_x + 1$ sigma points are selected symmetrically about the mean in the following way:

$$w^i = \begin{cases} \dfrac{\kappa}{(n_x+\kappa)}, & i = 0 \\ \dfrac{1}{2(n_x+\kappa)}, & i = 1, \ldots, 2n_x \end{cases}$$

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

where $\left(\sqrt{(n_x+\kappa)\mathbf{P}_{xx}}\right)^i$ refers to the ith column of the matrix square root of $(n_x+\kappa)\mathbf{P}_{xx}$ [26]. The number κ is a free parameter that can be used to tune the higher order moments of the distribution. Note that the sigma point weights do not directly represent probabilities, so are not restricted to the interval $[0, 1]$. If $\mathbf{x}$ is assumed Gaussian, then selecting $\kappa = 3 - n_x$ is recommended [25]. A smaller value of κ will bring the sigma points closer together than a larger value.

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

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

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

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

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

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

Joint state-parameter estimation can be accomplished in the UKF by augmenting the state vector with the unknown parameters. The corresponding diagonal elements of the process noise matrix, $\mathbf{Q}$, for the parameters $\boldsymbol{\theta}$ 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 assigned to the parameters determine both the rate of parameter estimation convergence and the estimation performance once convergence is achieved. Therefore, several heuristic approaches have been developed to tune this value online to optimize performance, e.g., [2], [27]–[29]. We adopt the approach presented in [2], [27], in which the algorithm modifies the variance in order to control the variance of the parameter estimate to a user-specified range. Note that the purpose of the algorithm is to adapt only the (artificial) process noise terms associated with the parameters, and process noise associated with the states and sensor noise is assumed to be known and the associated variance values are not adjusted.

The algorithm for the adaptation of the variance vector associated with $\boldsymbol{\theta}$, $\mathbf{v}_\theta$, is given as Algorithm 3 (see [27] for details), and is called at each time step. We assume that

Algorithm 3 $\mathbf{v}_\theta$ Adaptation

```
Inputs: $p(\mathbf{x}_k, \boldsymbol{\theta}_k | \mathbf{y}_{0:k})$
State: $\mathbf{v}_{\theta,k-1}, \mathbf{l} \leftarrow \mathbf{1}$
Outputs: $\mathbf{v}_{\theta,k}$
for all $j \in \{1, 2, \ldots, n_\theta\}$ do
    $v_j \leftarrow \texttt{RelativeSpread}(p(\boldsymbol{\theta}_k(j)|\mathbf{y}_{0:k}))$
    if $v_j < \mathbf{t}_j(\mathbf{s}(j))$ then
        $\mathbf{s}(j) \leftarrow \mathbf{s}(j) + 1$
    end if
    $\mathbf{v}_{\theta,k}(j) \leftarrow \mathbf{v}_{\theta,k-1}(j)\left(1 - \mathbf{P}_j(\mathbf{s}(j))\frac{v_j - \mathbf{v}_j^*(\mathbf{s}(j))}{\mathbf{v}_j^*(\mathbf{s}(j))}\right)$
end for
$\mathbf{v}_{\theta,k-1} \leftarrow \mathbf{v}_{\theta,k}$
```

the variance values are tuned initially based on the minimum expected EOLs. The adaptation proceeds in stages, maintained with the $\mathbf{s}_j$ variable for each parameter (with j referring to the parameter index). The relative spread is computed as v_j. If this value is below the threshold value for the the current stage, $\mathbf{t}_j(\mathbf{s}(j))$, then the stage number is increased. Then the new variance $\mathbf{v}_{\theta,k}(j)$ is computed. The error between the the actual and the desired spread value for the current stage, $v_j - \mathbf{v}_j^*(\mathbf{s}(j))$, is normalized by $\mathbf{v}_j^*(\mathbf{s}(j))$. This normalized error is then multiplied by the proportional gain term for the current stage, $\mathbf{P}_j(\mathbf{s}(j))$, and the corresponding variance $\mathbf{v}_{\theta,k-1}(j)$ is increased or decreased by that percentage to compute the new variance value $\mathbf{v}_{\theta,k}(j)$. Tuning of the algorithm parameters is necessary, but we have found that the number of stages $S_j = 2$ with $\mathbf{v}_j^* = [50, 10]$, $\mathbf{t}_j = [60, 0]$, and $\mathbf{P}_j = [1 \times 10^{-3}, 1 \times 10^{-4}]$ for all j works well in many cases. In the first stage, the variance is kept large to allow for convergence, and in the second stage, once convergence has begun, the variance is kept small for accurate tracking.

VII. Distributed Prediction

Each local prediction module takes as input local state-parameter estimates formed from the local estimators, as discussed in Section V. Given the mean and covariance information, we represent the distribution with a set of sigma points derived using the unscented transform. Then, as in [30], 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, deriving local EOL predictions using its local threshold function. The pseudocode for the prediction procedure is given as Algorithm 4 [3]. For a given submodel $\mathcal{M}_P^i$, each sigma point j is propagated forward until $T_{EOL}(\mathbf{x}_k^{i(j)}, \boldsymbol{\theta}_k^{i(j)}, \hat{\mathbf{u}}_k^i)$ evaluates to 1. The algorithm hypothesizes future inputs of the system, $\hat{\mathbf{u}}_k$. In this work, we consider only the situation where a single future input trajectory is known, because the pump in our application undergoes a strict pumping schedule [31]. Approaches to handle the case with uncertain future inputs are described in [32], [33].

As discussed in Section V, the global EOL prediction is taken as the minimum of the local EOL predictions. To compute this, we sample from each local EOL distribution and take the minimum of the local samples. This is repeated

Algorithm 4 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
```

many times and the statistics of the global EOL distribution are computed.

Note that prediction for some submodels may complete (i.e., simulate all their sigma points to EOL) before others, because the damage progression is faster in one submodel than in another. To avoid the distributed approach simulating beyond the point where a centralized approach would stop, we may run the local predictors simultaneously, and terminate all predictors whenever the first completes. The unfinished samples in the predictors can be ignored, since when taking the minimum, they would not be selected anyways. Prediction on some submodels may also be avoided altogether if the wear rate is clearly dominated by the wear rates on other submodels.

VIII. Results

We performed a number of simulation-based experiments to analyze the performance of the distributed prognostics approach compared to a centralized prognostics approach for the pump case study. For the distributed approach, we implemented the architecture given in Fig. 7. In this section, we first provide a demonstration of the approach, followed by a summary of a large number of experiments to compare the two approaches. We then argue for the improved scalabilty of the distributed approach and provide experimental results in support of it.

A. Demonstration of Approach

Here, we use percent root mean square error (PRMSE) as a measure of estimation accuracy, relative accuracy (RA) [34] as a measure of prediction accuracy (computed as the difference in true and predicted RUL over the true RUL, and expressed as a percentage), and RSD as a measure of spread. Each prediction metric is averaged over multiple prediction points (one every hour of usage) for a single scenario (see [2], [34] for the mathematical definitions of the metrics used here).

As an example scenario, consider the case where $w_{b_0} = 1 \times 10^{-3}$, $w_t = 2 \times 10^{-11}$, and $w_r = 2.5 \times 10^{-11}$ with the nominal noise level. Estimation results for the wear parameters for the

TABLE II
CENTRALIZED ESTIMATION AND PREDICTION PERFORMANCE

n	PRMSE$_{w_{b_0}}$	PRMSE$_{w_t}$	PRMSE$_{w_r}$	$\overline{\text{RSD}}_{w_{b_0}}$	$\overline{\text{RSD}}_{w_t}$	$\overline{\text{RSD}}_{w_r}$	$\overline{\text{RA}}$	$\overline{\text{RSD}}_{RUL}$
1	3.04	1.90	3.36	9.44	9.55	9.37	96.32	6.95
10	3.79	2.28	3.97	9.84	9.49	9.56	96.07	7.16
100	4.15	2.83	4.15	11.11	9.21	10.15	95.26	7.27
1000	3.59	3.21	4.50	11.78	9.37	10.78	94.98	7.49

TABLE III
DISTRIBUTED ESTIMATION AND PREDICTION PERFORMANCE

n	PRMSE$_{w_{b_0}}$	PRMSE$_{w_t}$	PRMSE$_{w_r}$	$\overline{\text{RSD}}_{w_{b_0}}$	$\overline{\text{RSD}}_{w_t}$	$\overline{\text{RSD}}_{w_r}$	$\overline{\text{RA}}$	$\overline{\text{RSD}}_{RUL}$
1	2.98	1.85	4.10	10.39	10.49	10.30	96.21	8.12
10	3.77	2.35	5.28	10.79	10.42	10.55	95.78	7.99
100	4.28	2.88	5.69	11.81	10.14	11.02	95.32	7.75
1000	3.76	3.55	5.39	13.09	10.23	12.11	94.25	7.99

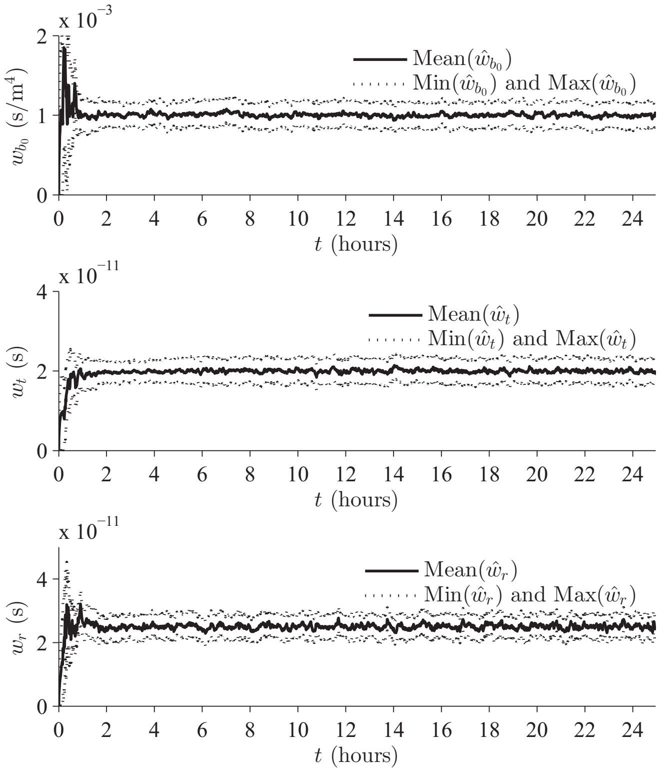


Fig. 8. Centralized estimation results for $w_{b_0} = 1\times10^{-3}$, $w_t = 2\times10^{-11}$, and $w_r = 2.5 \times 10^{-11}$.

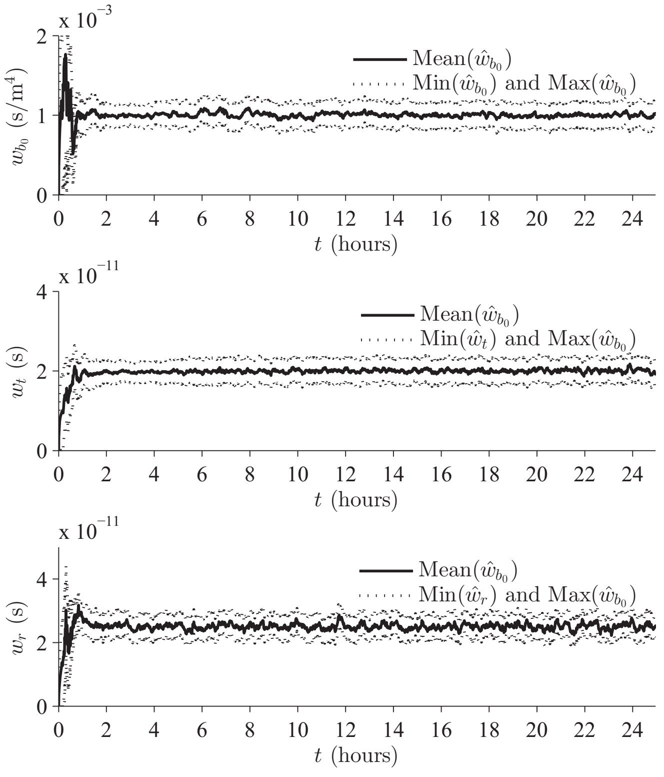


Fig. 9. Distributed estimation results for $w_{b_0} = 1\times10^{-3}$, $w_t = 2\times10^{-11}$, and $w_r = 2.5 \times 10^{-11}$.

centralized and distributed approaches are shown in Figs. 8 and 9, respectively. Note that the minimum and maximum values shown are those from the sigma points. Clearly, both approaches do very well, and there is no discernible difference between the two approaches. Due to the variance control algorithm, both approaches converge very quickly to the true values of the wear parameters, and remain close to the true values with small variance. In both cases, PRMSE for all unknown parameters is within 2–3%, with RSD of the wear parameters within 9–10% for the centralized case and within 10–11% for the distributed case.

For the same scenario, prediction results are given in Figs. 10 and 11 for the centralized and distributed approaches, respectively, as α-λ plots. The α-λ metric requires that at a given prediction point (λ), β of the predicted RUL distribution must come within α of the true RUL [34]. Here, we use $\alpha = 0.1$ and $\beta = 0.5$ for all λ, i.e., we require that at each prediction point, 50% of the distribution lies within 10% of ground truth. Both approaches pass the test at all prediction points, so either approach will obtain the desired performance. Figs. 10 and 11 show the result of the test and the percentage of the distribution lying within the α-bounds.

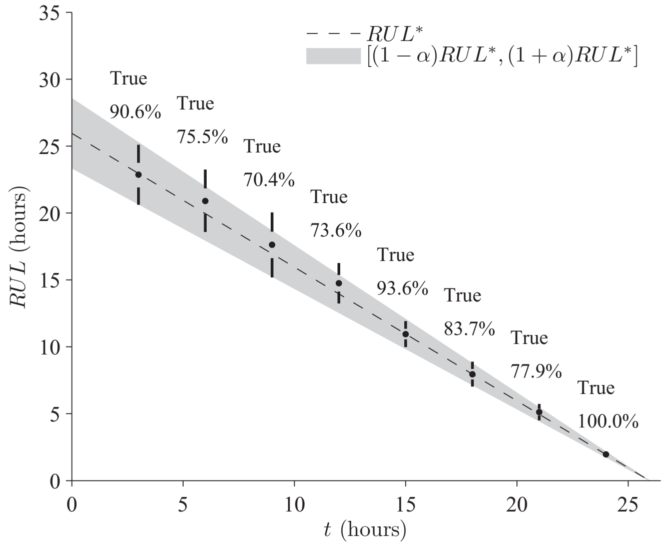


Fig. 10. Centralized prognosis results for $w_{b_0} = 1\times10^{-3}$, $w_t = 2\times10^{-11}$, and $w_r = 2.5 \times 10^{-11}$ with $\alpha = 0.1$ and $\beta = 0.5$.

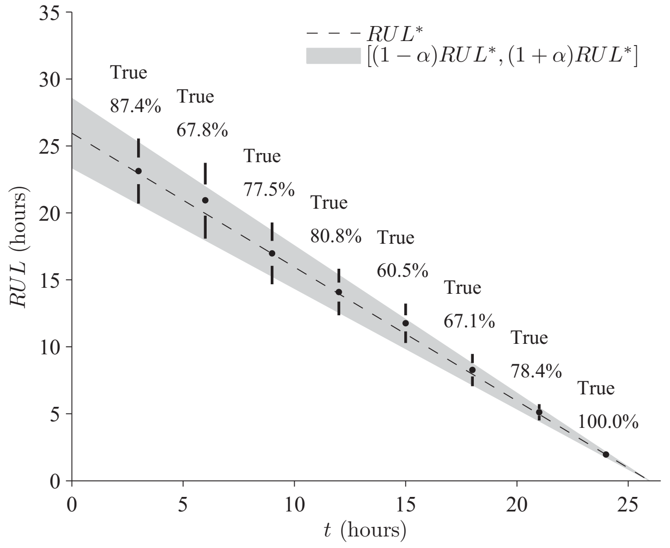


Fig. 11. Distributed prognosis results for $w_{b_0} = 1\times10^{-3}$, $w_t = 2\times10^{-11}$, and $w_r = 2.5 \times 10^{-11}$ with $\alpha = 0.1$ and $\beta = 0.5$.

The corresponding RA is 96.90% for the centralized case and 96.54% for the distributed case, with the RSD of the RUL at 7.44% for the centralized case and 6.85% for the distributed case.

If T_o^* is not measured (see Section V), both the estimation and prediction performance are virtually the same. PRMSE is within 2–3%, being only slightly higher (less than 1%) for the distributed case. RSD of the wear parameters is again within 9–10% for the centralized case and within 10–11% for the distributed case. Prediction results are also similar, with RA being 96.79% for the centralized case and 96.82% for the distributed case, with RSD of the RUL being 7.15% for the centralized case and 6.95% for the distributed case. Since dropping the T_o^* measurement does not significantly affect observability of the system, both centralized and distributed prognostics still perform well.

B. Prognostics Performance

In a single experiment, combinations of wear parameter values were selected randomly within the following ranges: $[1\times10^{-3}, 3\times10^{-3}]$ for w_{b_0}, in $[1\times10^{-11}, 3\times10^{-11}]$ for w_t, and in $[2\times10^{-11}, 5\times10^{-11}]$ for w_r, such that the maximum wear rates corresponded to a minimum EOL of 20 hours. For the variance control algorithm, with relative standard deviation (RSD) as the measure of spread, we used $S_j = 2$ with $\mathbf{v}_j^* = [50, 10]$, $\mathbf{t}_j = [60, 0]$, and $\mathbf{P}_j = [1\times10^{-3}, 1\times10^{-4}]$ for all j, in all experiments. Since the local estimators use measured values as inputs, performance will degrade as sensor noise is increased, so we varied the sensor noise variance by factors of 1, 10, 100, and 1000, to explore this situation. We performed 30 experiments for each sensor noise level for both the centralized and distributed approaches. We considered the case where the future input of the pump is known, in order to limit the uncertainty to only that involved in the noise terms and that introduced by the filtering algorithms. The pump operates at two different RPM values, changing every half hour, as shown in Fig. 2. For the pump model, we used a first-order discrete-time approximation using a step size of 1 s.

The averaged estimation and prediction performance results are shown in Table II for the centralized approach, and Table III for the distributed approach. The column labeled $\mathbf{n}$ lists the sensor noise variance multipliers. Note that all metrics are expressed as percentages.

We expect that in going from a centralized implementation to a distributed implementation there will be some loss of performance, due to the information lost in the decomposition, but that this performance loss will not be significant. As shown in Tables II and III, both the centralized and distributed approaches obtain high accuracy and precision, with RA over 94% and RSD$_{RUL}$ under 9%, and the performance of the distributed approach is virtually the same as the centralized approach. The distributed approach yields only small decreases in prediction accuracy (less than 1%) and small increases in spread (less than 1.2%). The decrease in performance of the distributed approach is expected, since the local estimators use noisy measurement values as inputs. Consequently, estimation performance decreases slightly and this translates to decreases in prediction performance. The distributed approach also must hypothesize the future value of ω, which is not completely accurate, and therefore also contributing to the slight decrease in performance.

As expected, both approaches perform worse as sensor noise increases, but with only small decreases in performance. The centralized approach loses 1.34% for RA, whereas the distributed approach loses 1.96% for RA. The decrease in performance from one noise level to the next higher level is larger with the distributed approach since the local estimation approach is more sensitive to noise.

As shown in this specific example and comparing Tables II and III, for the pump model, the distributed approach achieves prognostics performance virtually identical to the centralized approach. Although these results are only empirical, they should extend to other systems as well. For distributed estimation, the covariance information lost due to the decoupling is, in this case (and likely many others), negligible, and is not needed for prediction, so only a small loss in performance is expected, if any. For distributed prediction, the quality of the predictions will depend on the estimation results, so errors in estimation will propagate into prediction. Within the distributed prediction itself though, there is no information loss due to the decomposition.

C. Computational Efficiency

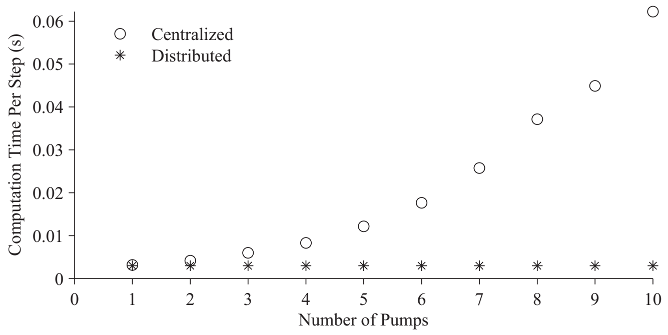
The distributed approach will always yield more efficient local estimators and predictors compared to the centralized approach, as long as some amount of decomposition is achieved, because each local submodel will be smaller than the global model, and the complexity of the estimation and prediction algorithms is a function of the model size. So, if each local estimator (predictor) is implemented on an independent processor, the distributed approach will be faster compared to the centralized approach on a single processor.

In particular, for the UKF, the computational complexity is polynomial in the state-parameter dimension ($O(n^3)$ for dimension n [35]). Using the symmetric unscented transform, there are $2n + 1$ sigma points for a state-parameter vector of size n. For the centralized approach, the state-parameter vector is of size 11, yielding 23 sigma points. For the distributed approach, the state-parameter vectors are of size 3, 3, 3, and 1, yielding 7, 7, 7, and 3 sigma points, respectively. In the implementation for the pump experiments, on average, we find that a single local estimator operates around 14% faster.[2]

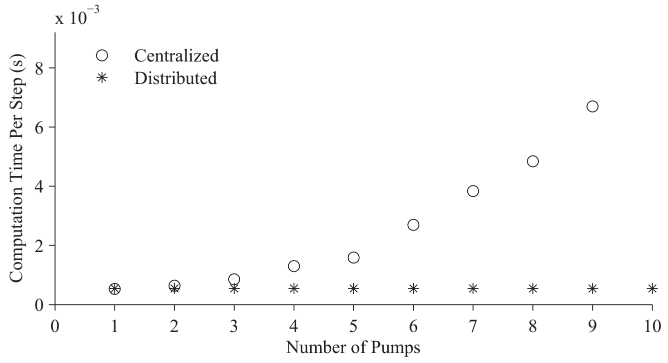
For the prediction algorithm, using the sigma points as the sample set, each local predictor has less samples than the global predictor, so, all things being equal, will be faster than the centralized predictor. Overall computational efficiency depends also on the spread of the samples, i.e., a sample with a wear rate value closer to zero will take longer to simulate to EOL than one with a larger wear rate value, given the same inputs [30]. If the distributed approach can achieve the same spread, then, it should be faster than the centralized approach since less samples are simulated forward. For the pump, the centralized approach uses the 24 sigma points obtained by the global estimator. The distributed approach has two submodels with state-parameter vectors of size 3 and 7, yielding 7 and 15 sigma points. The centralized and distributed approaches do achieve approximately the same amount of spread (e.g., compare Figs. 8 and 9), and we find that the distributed approach is, on average, about 15% faster.

D. Scalability

As the size of the system increases, we expect that the computational cost of the distributed approach grows more slowly than that of the centralized approach, i.e., the distributed approach is more scalable. For the case of UKF estimation, assume we have the least possible decomposition, where for global model dimension n the largest local submodel has dimension $n - 1$. Then the complexities are $O(n^3)$ and $O((n-1)^3)$, and if the size of the system increases by one state, the complexities are $O((n+1)^3)$ and $O((n)^3)$, and the complexity of the centralized approach grows by a larger margin than the distributed approach, therefore the distributed approach is more scalable. The argument is similar for distributed prediction.



(a) Estimation.



(b) Prediction.

Fig. 12. Scalability of centralized and distributed approaches.

The improved scalability is confirmed experimentally also. As a large-scale system we adopted a system consisting of n pumps. In this case, the decomposition results remain the same, where each pump results in 4 submodels for estimation and 2 submodels for prediction. For example, for a 5-pump system, there are 20 submodels for estimation and 10 for prediction. The scalability results for estimation are shown in Fig. 12a. As the size of the system increases, the amount of computation time required per step increases exponentially for the centralized approach. The distributed approach, on the other hand, stays constant, because even though there are more submodels, they are all operating in parallel. Even if the distributed approach was implemented sequentially on a single processor, the total amount of computation would grow only linearly, since each new pump adds 4 new submodels of fixed size. The scalability results for prediction are shown in Fig. 12b, and the results are similar to the case for estimation.

[2]We find the performance gain of the distributed approach smaller than expected due to both the overhead associated with the UKF implementation, and the fact that the implementation platform, MATLAB®, is very efficient at matrix multiplications; for relatively small matrices the performance difference is quite small.

IX. Related Work

Model-based prognostics approaches have been developed previously and applied to other components and fault modes, such as batteries [4], [36], fatigue cracks [37], [38], and automotive suspension systems [5]. Most model-based approaches are based on using filters for state estimation. Kalman filters have been used for prognostics of electrolytic capacitors in [39]. A model-based prognostics methodology is developed in [5] using an interacting multiple model filter for state-parameter estimation and prediction. An application of the approach of [5] to a centrifugal pump is developed in [23], but considers only a single degradation mode. Particle filters have been the most popular and have been used in [4], [37], [38], [40], [41], among others.

Some distributed prognostics approaches have also been explored. A distributed prognostics approach based on particle filters is developed in [15], and one based on Gaussian process regression in [16]. In contrast to our approach, these approaches still solve the global problem, and distribute only the computation. We propose a fundamentally different and novel distributed architecture, in which the global problem is decomposed into subproblems that can be solved independently and computation trivially distributed. This type of architecture is favorable, because there may be parts of the global problem that are not relevant to prognostics, and do not need to be solved (e.g., estimation of ω in the pump model). In a global approach where only the computation is distributed, these parts of the problem are still being solved. The local subproblems themselves can then be solved in a distributed fashion using an approach such as that described in [15].

The idea of using model decomposition to distribute state and parameter estimation is not new. Subspace methods [42], [43] have been used for solving identification problems in large dimensional systems by employing QR-factorization and singular-value decomposition [44]. These methods have been successfully used for linear systems, but face robustness problems when applied to nonlinear systems. Moreover, methods to automatically derive the decomposition directly from the system model have not been proposed. Regarding structural model decomposition, in [9], Williams and Millar propose an approach for decomposing a system model into smaller hierarchically organized subsystems, called *dissents*, applied to learning problems. Similar techniques, like Analytical Redundancy Relations (ARRs) [45] and Possible Conflicts (PCs) [10], both used for diagnosis, are also based on the idea of model decomposition. Dissents, ARRs and PCs are all conceptually equivalent [10]. PCs have been previously applied to generate a more robust and computationally simpler parameter estimation approach for fault identification [18]. Simulation results in that case showed an improvement in estimation accuracy while having a faster convergence to true solutions. Similar work was proposed in [46] using a dynamic Bayesian network (DBN) modeling framework, in which an automatic approach for model decomposition into submodels based on structural observability was developed for efficient state estimation and fault identification. We instead use a more general model decomposition framework, distributing the estimation problem in a way similar to these previous approaches, but distributing also the prediction problem in a novel way.

X. Conclusions

In this paper, we developed a novel distributed model-based prognostics approach based on structural model decomposition. The global model of a system is decomposed into a set of local submodels, from which *independent* local estimation and prediction problems are posed to solve the global prognostics problem in a distributed fashion that scales well. We applied a general model decomposition framework to generate minimal submodels for estimation and prediction. The local estimators compute local state-parameter estimates that define the system health state, and this information is used as an input to the local predictors, which compute EOL and RUL predictions for their submodels. The system EOL prediction can then be formed as the minimum of the local EOL predictions.

A centrifugal pump model was used for a simulation-based case study, demonstrating that, for all practical purposes, the distributed scheme has identical prognostics performance to the centralized scheme. Minor decreases in performance are observed, as expected, since the distributed scheme decomposes the models by using noisy sensor values as local submodel inputs, however, the impact was not significant and did not increase significantly as sensor noise increased. The distributed approach also offers improvements in computational efficiency and scalability in both the estimation and prediction steps.

In future work, we will apply this framework to system-level prognosis of large-scale systems. Further, the amount of model decomposition that can be achieved, for the estimation problem, is dependent on the number of sensors and where they are placed, so algorithms are needed for optimal placement of sensors to achieve the best model decompositions, and, hence, the best decomposition of the prognostics problem.

References

[1] M. Orchard and G. Vachtsevanos, “A particle filtering approach for on-line fault diagnosis and failure prognosis,” *Transactions of the Institute of Measurement and Control*, no. 3-4, pp. 221–246, June 2009.

[2] M. J. Daigle and K. Goebel, “Model-based prognostics with concurrent damage progression processes,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, to appear.

[3] M. Daigle and K. Goebel, “Model-based prognostics under limited sensing,” in *2010 IEEE Aerospace Conference*, Mar. 2010.

[4] B. Saha and K. Goebel, “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*, Sept. 2009.

[5] J. Luo, K. R. Pattipati, L. Qiao, and S. Chigusa, “Model-based prognostic techniques applied to a suspension system,” *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 38, no. 5, pp. 1156 –1168, Sept. 2008.

[6] P. Lall, R. Lowe, and K. Goebel, “Extended kalman filter models and resistance spectroscopy for prognostication and health monitoring of lead-free electronics under vibration,” *IEEE Transactions on Reliability*, vol. 61, no. 4, pp. 858–871, Dec. 2012.

[7] P. Baraldi, F. Mangili, and E. Zio, “A kalman filter-based ensemble approach with application to turbine creep prognostics,” *IEEE Transactions onReliability*, vol. 61, no. 4, pp. 966–977, Dec. 2012.

[8] H. Thompson, “Parallel processing architectures for aerospace applications,” *Control Engineering Practice*, vol. 2, no. 3, pp. 509–520, 1994.

[9] B. Williams and B. Millar, “Decompositional model-based learning and its analogy to diagnosis,” in *Proc. of the Fifteenth National Conference on Artificial Intelligence*, 1998, pp. 197–204.

[10] B. Pulido and C. Alonso-González, “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*, vol. 34, no. 5, pp. 2192–2206, 2004.

[11] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki, *Diagnosis and Fault-Tolerant Control*. Springer, 2006.

[12] L. Travé-Massuyès, T. Escobet, and X. Olive, “Diagnosability analysis based on component supported analytical redundancy relations,” *IEEE Trans. on Systems, Man, and Cybernetics, Part A*, vol. 36, no. 6, 2006.

[13] M. Krysander, J. Åslund, and M. Nyberg, “An efficient algorithm for finding minimal over-constrained sub-systems for model-based diagnosis,” *IEEE Trans. on Systems, Man, and Cybernetics, Part A*, vol. 38, no. 1, 2008.

[14] I. Roychoudhury, M. Daigle, A. Bregon, and B. Pulido, “A structural model decomposition framework for systems health management,” in *Proceedings of the 2013 IEEE Aerospace Conference*, Mar. 2013.

[15] B. Saha, S. Saha, and K. Goebel, “A distributed prognostic health management architecture,” in *Proceedings of the 2009 Conference of the Society for Machinery Failure Prevention Technology*, 2009.

[16] S. Saha, B. Saha, A. Saxena, and K. Goebel, “Distributed prognostic health management with Gaussian process regression,” in *Aerospace Conference, 2010 IEEE*, Mar. 2010.

[17] M. Daigle, A. Bregon, and I. Roychoudhury, “Distributed damage estimation for prognostics based on structural model decomposition,” in *Proceedings of the Annual Conference of the Prognostics and Health Management Society 2011*, Sept. 2011, pp. 198–208.

[18] A. Bregon, G. Biswas, and B. Pulido, “A decomposition method for nonlinear parameter estimation in TRANSCEND,” *IEEE Trans. on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 42, no. 3, pp. 751–763, May 2012.

[19] S. E. Lyshevski, *Electromechanical Systems, Electric Machines, and Applied Mechatronics*. CRC, 1999.

[20] A. Wolfram, D. Fussel, T. Brune, and R. Isermann, “Component-based multi-model approach for fault detection and diagnosisof a centrifugal pump,” in *Proceedings of the 2001 American Control Conference*, vol. 6, 2001, pp. 4443–4448.

[21] C. Kallesøe, “Fault detection and isolation in centrifugal pumps,” Ph.D. dissertation, Aalborg University, 2005.

[22] G. Biswas and S. Mahadevan, “A hierarchical model-based approach to systems health management,” in *Proc. of the 2007 IEEE Aerospace Conference*, Mar. 2007.

[23] F. Tu, S. Ghoshal, J. Luo, G. Biswas, S. Mahadevan, L. Jaw, and K. Navarra, “PHM integration with maintenance and inventory management systems,” in *Proc. of the 2007 IEEE Aerospace Conference*, Mar. 2007.

[24] I. M. Hutchings, *Tribology: friction and wear of engineering materials*. CRC Press, 1992.

[25] S. J. Julier and J. K. Uhlmann, “A new extension of the Kalman filter to nonlinear systems,” in *Proc. of the 11th Intl. Symp. on Aerospace/ Defense Sensing, Simulation and Controls*, 1997, pp. 182–193.

[26] ——, “Unscented filtering and nonlinear estimation,” *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, Mar. 2004.

[27] M. Daigle, B. Saha, and K. Goebel, “A comparison of filter-based approaches for model-based prognostics,” in *Proceedings of the 2012 IEEE Aerospace Conference*, Mar. 2012.

[28] J. Liu and M. West, “Combined parameter and state estimation in simulation-based filtering,” *Sequential Monte Carlo Methods in Practice*, pp. 197–223, 2001.

[29] M. Orchard, F. Tobar, and G. Vachtsevanos, “Outer feedback correction loops in particle filtering-based prognostic algorithms: Statistical performance comparison,” *Studies in Informatics and Control*, no. 4, pp. 295–304, Dec. 2009.

[30] M. Daigle and K. Goebel, “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*, Oct. 2010.

[31] C. Goodrich, S. Narasimhan, M. Daigle, W. Hatfield, R. Johnson, and B. Brown, “Applying model-based diagnosis to a rapid propellant loading system,” in *Proceedings of the 20th International Workshop on Principles of Diagnosis*, June 2009, pp. 147–154.

[32] M. Daigle, A. Saxena, and K. Goebel, “An efficient deterministic approach to model-based prediction uncertainty estimation,” in *Annual Conference of the Prognostics and Health Management Society 2012*, Sept. 2012, pp. 326–335.

[33] S. Sankararaman, M. Daigle, A. Saxena, and K. Goebel, “Analytical algorithms to quantify the uncertainty in remaining useful life prediction,” in *Proceedings of the 2013 IEEE Aerospace Conference*, Mar. 2013.

[34] A. Saxena, J. Celaya, B. Saha, S. Saha, and K. Goebel, “Metrics for offline evaluation of prognostic performance,” *International Journal of Prognostics and Health Management*, vol. 1, no. 1, 2010.

[35] F. Daum, “Nonlinear filters: beyond the Kalman filter,” *IEEE Aerospace and Electronic Systems Magazine*, vol. 20, no. 8, pp. 57–69, 2005.

[36] M. Abbas, A. A. Ferri, M. E. Orchard, and G. J. Vachtsevanos, “An intelligent diagnostic/prognostic framework for automotive electrical systems,” in *2007 IEEE Intelligent Vehicles Symp.*, 2007, pp. 352–357.

[37] M. E. Orchard, “A particle filtering-based framework for on-line fault diagnosis and failure prognosis,” Ph.D. dissertation, Georgia Institute of Technology, 2007.

[38] E. Zio and G. Peloni, “Particle filtering prognostic estimation of the remaining useful life of nonlinear components,” *Reliability Engineering & System Safety*, vol. 96, no. 3, pp. 403–409, 2011.

[39] J. R. Celaya, C. Kulkarni, G. Biswas, S. Saha, and K. Goebel, “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*, Sept. 2011, pp. 31–39.

[40] N. Bolander, H. Qiu, N. Eklund, E. Hindle, and T. Rosenfeld, “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*, Oct. 2010.

[41] M. Daigle and K. Goebel, “A model-based prognostics approach applied to pneumatic valves,” *International Journal of Prognostics and Health Management*, vol. 2, no. 2, Aug. 2011.

[42] T. Katayama, *Subspace Methods for System Identification*. Springer, 2005.

[43] M. Viberg, “Subspace-based state-space system identification,” *Circuits, Systems, and Signal Processing*, vol. 21, no. 1, pp. 23–37, 2002.

[44] P. Overschee and B. D. Moor, *Subspace Identification for Linear Systems*. Boston, MA, USA: Kluwer Academic Publishers, 1996.

[45] M. Staroswiecki and P. Declerck, “Analytical redundancy in nonlinear interconnected systems by means of structural analysis,” in *IFAC Symp. on Advanced Information Processing in Automatic Control*, July 1989.

[46] I. Roychoudhury, G. Biswas, and X. Koutsoukos, “Factoring dynamic Bayesian networks based on structural observability,” in *Proc. of the 48th IEEE Conference on Decision and Control*, Dec. 2009, pp. 244–250.

Matthew J. Daigle (S'07–M'08) 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. 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 recipient of two Staff Recognition and Development Awards from the University of California, Santa Cruz, a best paper award at the Annual Conference of the Prognostics and Health Management Society, an Ames Contractor Council Excellence Award, and a NASA Ames Research Center Group Achievement Award. He is a member of the Prognostics and Health Management Society and the IEEE.

Anibal Bregon (S'08-M'10) received his B.Sc., M.Sc., and Ph.D. degrees in Computer Science from the University of Valladolid, Valladolid, Spain, in 2005, 2007, and 2010, respectively.

From September 2005 to June 2010, he was a Graduate Research Assistant with the Intelligent Systems Group, at the Department of Computer Science, University of Valladolid, Spain. Dr. Bregon has been a visiting researcher with the Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, USA, with the Department of Electrical Engineering, Linköping University, Linköping, Sweden, and with the Diagnostics and Prognostics Group, NASA Ames Research Center, Moffett Field, CA, USA. Currently he is Assistant Professor and Research Assistant with the Department of Computer Science, 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 and prognostics of complex physical systems.

Indranil Roychoudhury (S'07–M'08) 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 Stinger Ghaffarian Technologies, at NASA Ames Research Center as a Computer Scientist. His research interests include hybrid systems modeling, model-based diagnostics and prognostics, distributed diagnostics and prognostics, and Bayesian diagnostics of complex physical systems.

Dr. Roychoudhury is the recipient of the ISRDS Team Recognition Award from Stinger Ghaffarian Technologies and a best paper award at the Annual Conference of the Prognostics and Health Management Society. He is a member of the Prognostics and Health Management Society and the IEEE.