Predicting Remaining Driving Time and Distance of a Planetary Rover under Uncertainty

Matthew Daigle
NASA Ames Research Center
Moffett Field, CA 94035
Email: matthew.j.daigle@nasa.gov

Shankar Sankararaman*
NASA Ames Research Center (SGT Inc.)
Moffett Field, CA 94035
Email: shankar.sankararaman@nasa.gov

The operations of a planetary rover depend critically upon the amount of power that can be delivered by its batteries. In order to plan the future operation, it is important to make reliable predictions regarding the end-of-discharge time, which can be used to estimate the remaining driving time and distance. These quantities are stochastic in nature, not only because there are several sources of uncertainty that affect the rover's operation, but also since the future operating conditions cannot be known precisely. This paper presents a computational methodology to predict these stochastic quantities, based on a model of the rover and its batteries. We utilize a model-based prognostics framework that characterizes and incorporates the various sources of uncertainty into these predictions, thereby assisting operational decision-making. We consider two different types of driving scenarios, and develop methods for each to characterize the associated uncertainty. Monte Carlo sampling and the inverse first-order reliability method are used to compute the stochastic predictions of end-of-discharge time, remaining driving time, and remaining driving distance.

1 Introduction

Robots are used to facilitate automation in several industrial, mechanical, and aerospace applications [1, 2].In planetary exploration, rovers must have the capability to autonomously perform path planning [3, 4], fault mitigation [5, 6] and mission re-routing [7, 8]. In this context, it is important to predict how much longer the rover can be operated, i.e, what is the time until the end-of-discharge (EOD) of the batteries powering the rover, and the corresponding remaining driving time (RDT) and/or the remaining driving distance (RDD) [9], by anticipating the future operating demands on the rover. Research in the topic of prognostics [10–12] has focused on developing computational methods for predicting the future behavior of engineering components, subsystems, and systems, and this paper focuses on applying these methods to planetary rover operations.

This paper uses model-based prognostic methods [13–15] (as against data-driven approaches [16, 17] that mainly rely on the presence of large failure data sets) to solve the prediction problem by first estimating the system state, and then simulating it forward in time until a predefined event occurs. This event may relate to the violation of certain usability, safety, and/or serviceability constraints (such as reaching the end-of-discharge of a battery [18, 19]). Prediction of the future system behavior is affected by several sources of uncertainty; therefore, a rigorous approach for prognostics needs to systematically account for these sources of uncertainty and quantify their effect on the desired predictions. Due to the presence of uncertainty, the system evolution is a random process. A systematic approach to prognostics, therefore, needs to address two important issues: (*i*) to identify and characterize the various sources of uncertainty, and (*ii*) to quantify the combined effect of the different sources of uncertainty within the prediction and thereby compute the prediction uncertainty.

Some common sources of uncertainty in model-based prognostics include state estimation uncertainty, model uncertainty, and future input (loading, environmental, and usage conditions) uncertainty. While state estimation uncertainty and model uncertainty can be reduced by using better sensors, robust state estimators, and improved models, future input uncertainty is practically irreducible since it is almost impossible to precisely predict the future loading conditions for many practical engineering systems. Uncertainty regarding future inputs is, in fact, typically the most significant source of uncertainty [15, 20–22].

Once each source of uncertainty has been characterized and quantified, it must be accounted for within the prediction

*Address all correspondence related to ASME style format and figures

procedure, and thereby, the overall prediction uncertainty needs to be computed. In previous work [23], we explained that such a prediction problem is, fundamentally, an uncertainty propagation problem and investigated different types of sampling-based methods [18] and analytical methods [24] for this purpose. We also investigated future input characterization [25] for the power system of a rover, and compared different methods for EOD prediction.

While most prior work was performed only in the context of component-level prognostics, this paper focuses on the operations of a planetary rover, and we develop a generic model-based prognostics framework [14] that systematically incorporates system-level information. We predict the EOD, RDT, and RDD of a planetary rover. While EOD only indicates how much time is left until the battery discharges, RDT indicates how much of that time is actually spent driving the rover, and RDD indicates how much distance the rover may travel until EOD. All of these quantities are related, and are determined by the commanded speeds and corresponding power requirements of the rover. Operationally, RDT and RDD are more meaningful than EOD, because they are explicitly related to the driving of the rover. We consider two different driving scenarios that require different future input characterization and prediction methods: unstructured driving and structured driving. Unstructured driving describes a free driving scenario by considering a set of unplanned maneuvers: "going straight", "turning", and "stopping", each with stochastic duration and power demands. Here, we use Monte Carlo sampling. In constrast, in structured driving, a set of waypoints must be traversed at specific speeds until EOD. Here, we use the inverse FORM approach. In both scenarios, we quantify the uncertainty in the inputs to the rover, and use this information to compute the uncertainty in EOD, RDT, and RDD.

The rest of this paper is organized as follows. Section 2 defines the prognostics problem and discuss the various sources of uncertainty in prognostics. Section 3 presents the modeling aspects of the rover and explains the two driving scenarios in detail. Sections 4 and 5 discuss the estimation and prediction methodologies, while Section 6 presents numerical results. Finally, Section 7 concludes the paper and discusses possible directions for further research.

2 Model-based Prognostics

In this section, we discuss the general problem of model-based prognostics, discuss the importance of uncertainty in prognostics, develop a new approach for uncertainty representation, and finally, present a computational architecture for prognostics under uncertainty.

2.1 Formal Definition of Prognostics

Consider a system that is represented using a generic state-space model, as:

$$\mathbf{x}(k+1) = \mathbf{f}(k, \mathbf{x}(k), \theta(k), \mathbf{u}(k), \mathbf{v}(k)), \quad (1)$$
$$\mathbf{y}(k) = \mathbf{h}(k, \mathbf{x}(k), \theta(k), \mathbf{u}(k), \mathbf{n}(k)), \quad (2)$$

where k is the discrete time variable, $\mathbf{x}(k) \in \mathbb{R}^{n_x}$ is the state vector, $\theta(k) \in \mathbb{R}^{n_\theta}$ is the unknown parameter vector, $\mathbf{u}(k) \in \mathbb{R}^{n_u}$ is the input vector, $\mathbf{v}(k) \in \mathbb{R}^{n_v}$ is the process noise vector, $\mathbf{f}$ is the state equation, $\mathbf{y}(k) \in \mathbb{R}^{n_y}$ is the output vector (corresponding to the system sensors), $\mathbf{n}(k) \in \mathbb{R}^{n_n}$ is the sensor noise vector, and $\mathbf{h}$ is the output equation.[1] The unknown parameter vector $\theta(k)$ is used to capture explicit model parameters whose values are unknown and time-varying.

In prognostics, we are interested in predicting the occurrence of some event E that corresponds to the violation of certain performance, usability, safety, and/or serviceability constraints. We define the event E as the earliest instant that some event threshold, that is a Boolean function of the states, parameters, and inputs of the system, $T_E : \mathbb{R}^{n_x} \times \mathbb{R}^{n_\theta} \times \mathbb{R}^{n_u} \to \mathbb{B}$, where $\mathbb{B} \triangleq \{0, 1\}$, changes from the value 0 to 1 [14]. Let k_E denote the time at which the event E occurs, and let k_P denote a generic time at which it is desired to perform prediction. Then, k_E is defined as a function of k_P, as:

$$k_E(k_P) \triangleq \inf\{k \in \mathbb{N} \colon k \geq k_P \wedge T_E(\mathbf{x}(k), \theta(k), \mathbf{u}(k)) = 1\}. \quad (3)$$

Note that k_E is an explicit function of the time of prediction k_P, since the time-to-event E depends on the state estimated at k_P and anticipated future conditions ($\forall k > k_P$). The time remaining until the occurrence of that event E, denoted by Δk_E, is defined as:

$$\Delta k_E(k_P) \triangleq k_E(k_P) - k_P. \quad (4)$$

We are interested in predicting k_E and the values of desired system-level variables, $\mathbf{z}$, at time k_E, which are defined as a function of the state, parameters, and inputs using $\mathbf{g}$:

$$\mathbf{z}(k) = \mathbf{g}(k, \mathbf{x}(k), \theta(k), \mathbf{u}(k)). \quad (5)$$

As shorthand, we use $\mathbf{z}_E(k_P)$ to refer to $\mathbf{z}(k_E(k_P))$, i.e., the prediction of $\mathbf{z}(k_E)$ at prediction time k_P. We may also be interested in the difference in these quantities from the time of prediction k_P, i.e.,

$$\Delta \mathbf{z}_E(k_P) = \mathbf{z}_E(k_P) - \mathbf{z}(k_P). \quad (6)$$

The system evolution is a random process due to the process noise $\mathbf{v}(k)$ and nondeterministic inputs $\mathbf{u}(k)$. In addi-

tion, the state and parameter values are not known exactly, and so only uncertain estimates can be derived. At the time of prediction k_P, the predicted quantities $k_E(k_P)$, $\Delta k_E(k_P)$, $\mathbf{z}_E(k_P)$, and $\Delta\mathbf{z}_E(k_P)$ can each be mathematically expressed as a function of the following quantities, all of which are uncertain [18, 24]:

1. the values of the state variables at k_P, $\mathbf{x}(k_P)$;
2. the model parameter values from time k_P until k_E, a time-trajectory $\Theta_{k_P} = [\theta(k_P)\ \theta(k_P+1)\ \ldots\ \theta(k_E)]$;
3. the system inputs from time k_P until k_E, a time-trajectory $\mathbf{U}_{k_P} = [\mathbf{u}(k_P)\ \mathbf{u}(k_P+1)\ \ldots\ \mathbf{u}(k_E)]$; and
4. the process noise values from time k_P until k_E, a time-trajectory denoted by $\mathbf{V}_{k_P} = [\mathbf{v}(k_P)\ \mathbf{v}(k_P+1)\ \ldots\ \mathbf{v}(k_E)]$.

So, the prediction problem is to compute the *probability distributions* of $k_E(k_P)$, $\Delta k_E(k_P)$, $\mathbf{z}_E(k_P)$, and $\Delta\mathbf{z}_E(k_P)$, using estimated probability distributions for $\mathbf{x}(k_P)$, Θ_{k_P}, $\mathbf{U}_{k_P}$, and $\mathbf{V}_{k_P}$ and the model that includes $\mathbf{f}$, $\mathbf{g}$, and T_E.

2.2 Representing Uncertainty in Prognostics

In order to solve the prediction problem, the first step is to quantify and appropriately represent the uncertainty in each of the uncertain inputs. Let $p(\mathbf{x}(k_P))$, $p(\Theta_{k_P})$, $p(\mathbf{U}_{k_P})$, and $p(\mathbf{V}_{k_P})$ denote their respective probability distributions.

Representing $p(\mathbf{x}(k_P))$ is straightforward, often captured through the mean and the covariance of the multi-dimensional state-space, but, in general, representing the trajectories is more difficult. One can always represent the distribution of these trajectories directly, but when that becomes difficult, the concept of *surrogate variables* can be used [25]. Surrogate variables are used to transform a time-varying stochastic process into a combination of time-invariant random numbers and a deterministic (non-random), time-varying function; the former random numbers serve as parameters of the function, thereby constituting a set of random trajectories. The time-invariant random numbers are referred to as the surrogate variables. A similar concept has been used in stochastic processes, for instance, in Karhunen-Loeve expansion [26], where the eigenvalues of the autocorrelation function are chosen as "surrogate variables" that can represent the time-varying stochastic process.

For describing the probability distribution of a generic trajectory $\mathbf{A}_k$, we introduce a finite set of surrogate random variables $\lambda_a = [\lambda_{a,1}\ \lambda_{a,2}\ \ldots\ \lambda_{a,m}]$. We describe a trajectory using λ_a and instead define $p(\lambda_a)$, which in turn defines $p(\mathbf{A}_k)$ through some function $\mathbf{f}_a(k,\lambda_a)$. The surrogate variables, λ_a, and the function $\mathbf{f}_a$ can be defined to describe trajectories in a variety of ways. The use of the surrogate variables provides flexibility to the user in defining the distribution of trajectories.

2.3 Prognostics Architecture

We adopt a model-based prognostics architecture [14], in which there are two sequential problems, (*i*) the *estimation* problem, which requires determining a joint state-parameter estimate, $p(\mathbf{x}(k),\theta(k)|\mathbf{y}(k_0{:}k_P))$, based on the history of observations up to time k, $\mathbf{y}(k_0{:}k_P)$; and (*ii*) the *prediction* problem, which, at a specified prediction time, k_P, using estimates of $p(\mathbf{x}(k_P))$, $p(\Theta_{k_P})$, $p(\mathbf{U}_{k_P})$, and $p(\mathbf{V}_{k_P})$, computes $p(k_E(k_P)|\mathbf{y}(k_0{:}k_P))$, $p(\Delta k_E(k_P)|\mathbf{y}(k_0{:}k_P))$, $p(\mathbf{z}_E(k_P)|\mathbf{y}(k_0{:}k_P))$, and/or $p(\Delta\mathbf{z}_E(k_P)|\mathbf{y}(k_0{:}k_P))$.

The prognostics architecture is shown in Fig. 1. In discrete time k, the system is provided with inputs $\mathbf{u}(k)$ and provides measured outputs $\mathbf{y}(k)$. The estimation module uses this information, along with the system model, to compute an estimate of the states and parameters at time k, $p(\mathbf{x}(k),\theta(k)|\mathbf{y}(k_0{:}k))$. At time k_P, the prediction module uses this information along with the information regarding future input, parameter, and process noise trajectories represented using the surrogate variables, $p(\lambda_\theta)$, $p(\lambda_u)$, and $p(\lambda_v)$, to compute $p(k_E(k_P)|\mathbf{y}(k_0{:}k_P))$, $p(\Delta k_E(k_P)|\mathbf{y}(k_0{:}k_P))$, $p(\mathbf{z}_E(k_P)|\mathbf{y}(k_0{:}k_P))$, and/or $p(\Delta\mathbf{z}_E(k_P)|\mathbf{y}(k_0{:}k_P))$.

The estimator solves the problem of representing the probability distribution for $\mathbf{x}(k_P)$.

For the future input trajectories, the probability distribution depends on the particular system under consideration and the environment it is operating within. As with the other trajectories, we describe $p(\mathbf{U}_{k_P})$ using surrogate variables, λ_u. Often, there is some knowledge about what the future input will be and only a few random variables are needed in λ_u. For example, in a constant-loading scenario with a single input, the input trajectory can be defined using a single surrogate variable, i.e., $\mathbf{u}(k) = \lambda_{u,1}$ for $k \geq k_P$.

In order to account for process noise, we define $p(\mathbf{V}_k)$ using λ_v. Typically, this quantity is represented as white noise and needs to be sampled at every time-instant. Alternatively, an equivalent time-invariant process noise, i.e., a single random variable that defines a constant value of process noise for all k [27], can be computed using the principle of likelihood and used in analysis.

3 Rover Modeling for Prognostics

Prior to the application of the above generic framework for prognostics and uncertainty quantification, it is necessary to understand the physical system of interest, and develop appropriate models. In this paper, we consider a four-wheeled, skid-steered rover, powered by a set of lithium-ion batteries [28]. We are interested in predicting the time of EOD, and the corresponding RDT and RDD. In order to predict these quantities, we need to develop a model that describes the relationship between the rover inputs, i.e., the commanded wheel speeds, and the battery voltage, driving time, and driving distance. This model is described first in Section 3.1.

In order to make predictions, we must also develop models of the uncertain inputs to the prediction problem. That is, we need to determine appropriate and accurate descriptions of the distributions of the future trajectories, $p(\Theta_{k_P})$, $p(\mathbf{U}_{k_P})$, and $p(\mathbf{V}_{k_P})$. Section 3.2 describes these for both structured and unstructured driving scenarios.

3.1 System Modeling

As part of the modeling process, the states ($\mathbf{x}$), the model parameters (θ), the model inputs ($\mathbf{u}$), the model outputs ($\mathbf{y}$),

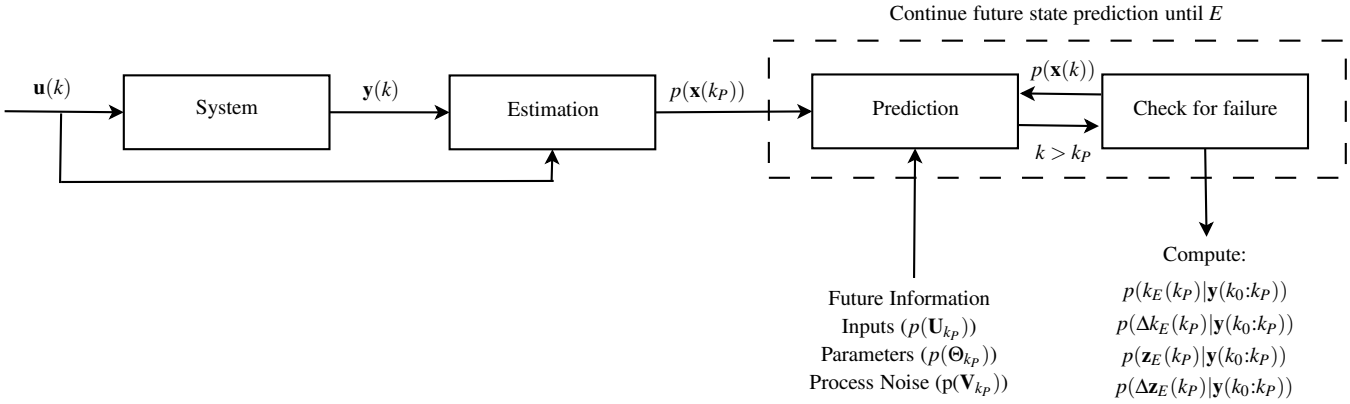

Fig. 1. Model-Based Prognostics Architecture

the state equation (**f**), the output equation (**h**), the event of interest (E), the threshold function (T_E), the additional quantities of interest (**z**), and the equation used to compute them (**g**) must all be defined.

The system inputs, **u**, are defined by the velocities of the four wheels: the front-left wheel speed, v_{FL}; the front-right wheel speed, v_{FR}; the back-left wheel speed, v_{BL}; and the back-right wheel speed, v_{BR}. As a control policy, we have the following constraints:

$$v_L = v_{FL} = v_{BL}, \tag{7}$$
$$v_R = v_{FR} = v_{BR}, \tag{8}$$

where v_L is the left-side velocity and v_R is the right-side velocity; in other words, we always control the wheels on one side of the rover to the same speed. Therefore, we consider only v_L and v_R as the independent system inputs:

$$\mathbf{u}(t) = \begin{bmatrix} v_L & v_R \end{bmatrix}^T. \tag{9}$$

We have also the following kinematic constraints:

$$v = \frac{v_L + v_R}{2}, \tag{10}$$
$$\omega = \frac{v_R - v_L}{2b}, \tag{11}$$

where v is the rover translational velocity, and ω is the rover rotational velocity. Here, b is the distance from the rover longitudinal axis to the wheels. Adopting a Cartesian coordinate system, it follows that

$$\dot{x} = v\cos\theta, \tag{12}$$
$$\dot{y} = v\sin\theta, \tag{13}$$
$$\dot{\theta} = \omega. \tag{14}$$

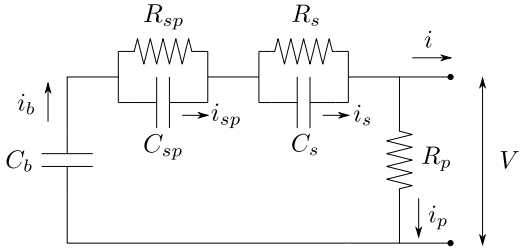

Fig. 2. Battery equivalent circuit

The driving distance d and driving time are described by:

$$\dot{d} = v, \dot{t}_d = (v > 0). \tag{15}$$

Required power is computed as an empirical function of v, v_L, and v_R (determined from data):

$$P = 4.44v + 69.54(v_L - v_R)^2 + 1.55, \tag{16}$$

where velocity is expressed in m/s and power in W. Note that power increases with the overall velocity, but increases also with differences in left- and right-side velocities, i.e, turning requires more power than moving straight (due to increased friction encountered when turning). When the rover is not moving, a small amount of power (1.55 W) is required to keep the components powered.

The rover is powered by two parallel sets of 12 lithium-ion batteries in series, which we lump into one single abstracted battery. We employ an extended version of the equivalent circuit battery model presented in [18]. While more complex battery models are available, equivalent circuit models provide a desirable trade off between execution speed and accuracy [29–31]. The schematic for the equivalent circuit model is shown in Fig. 2, and the equations are summarized in Table 1. We briefly describe the model here, and refer to [18] for additional details.

The large capacitance C_b holds the charge q_b of the battery. The nonlinear C_b captures the open-circuit potential and concentration over-potential, and C_b is a nonlinear function

Table 1. Battery Model Equations

$$
\begin{aligned}
i &= P/V \\
i_p &= \frac{V_p}{R_p} \\
i_b &= i_p + i \\
i_{sp} &= i_b - \frac{V_{sp}}{R_{sp}} \\
i_s &= i_b - \frac{V_s}{R_s} \\
V_b &= \frac{q_b}{C_b} \\
V_{sp} &= \frac{q_{sp}}{C_{sp}} \\
V_s &= \frac{q_s}{C_s} \\
V_p &= V_b - V_{sp} - V_s \\
V &= V_b - V_{sp} - V_s \\
\dot{q}_b &= -i_b \\
\dot{q}_{sp} &= i_{sp} \\
\dot{q}_s &= i_s \\
SOC &= 1 - \frac{q_{max} - q_b}{C_{max}} \\
R_{sp} &= R_{sp_0} + R_{sp_1} \exp\left(R_{sp_2}(1 - SOC)\right) \\
C_b &= C_{b_0} + C_{b_1} SOC + C_{b_2} SOC^2 + C_{b_3} SOC^3
\end{aligned}
$$

Table 2. Battery Model Parameters

Parameter	Value	Parameter	Value
C_{b_0}	312.43 F	C_{sp}	2.47 F
C_{b_1}	346.65 F	R_{sp_0}	0.42 Ω
C_{b_2}	0.20 F	R_{sp_1}	$9. \times 10^{-17}$ Ω
C_{b_3}	−38.33 F	R_{sp_2}	37.22
R_s	0.32 Ω	q_{max}	1.57×10^4 C
C_s	39.06 F	C_{max}	15,554 C
R_p	1×10^4 Ω	V_{EOD}	30.00 V

of state-of-charge (SOC) [32]. The R_{sp}-C_{sp} pair captures the major nonlinear voltage drop due to surface over-potential, R_s captures the Ohmic drop, and R_p models the parasitic resistance that accounts for self-discharge. R_{sp} is also a nonlinear function of SOC, increasing exponentially as SOC decreases (R_{sp_0}, R_{sp_1}, and R_{sp_2} are empirical parameters). SOC is computed based on the amount of charge in the battery, q_b, the maximum possible charge, q_{max}, and the maximum possible capacity, C_{max}. The parameter values of the battery model are given in Table 2. All voltages are given in Volts, resistances in Ohms, charges in Coulombs, and capacitances in Farads[2]. The model is implemented considering a discrete-time version with a time step of 1 s.

The complete state vector is:

$$\mathbf{x} = \begin{bmatrix} x & y & \theta & d & t_d & q_b & q_{sp} & q_s \end{bmatrix}^T. \tag{17}$$

We assume that all parameters are known, so $\theta = \varnothing$. The available measurements include the battery current and voltage:

$$\mathbf{y} = \begin{bmatrix} i & V \end{bmatrix}^T. \tag{18}$$

The event E that we want to predict is EOD, and T_E is specified using a voltage threshold, V_{EOD}. Specifically, we define T_E as:

$$T_E = (V < V_{\text{EOD}}). \tag{19}$$

For prediction purposes, we define $\mathbf{z}$ as:

$$\mathbf{z} = \begin{bmatrix} d & t_d \end{bmatrix}^T. \tag{20}$$

3.2 Uncertainty Modeling

While the uncertainty in the state estimates, i.e., $p(\mathbf{x}(k_P))$, will be directly provided by the state estimation algorithm, the probability distributions for future trajectories $p(\Theta_{k_P})$, $p(\mathbf{U}_{k_P})$, and $p(\mathbf{V}_{k_P})$ must all be defined. In our system, all parameters are assumed to be known, and so the probability density function $p(\Theta_{k_P})$ need not be considered. Further, in this application, we observe that the effect of process noise is negligible relative to the uncertainty in future inputs (such conclusion is arrived based on sensivitity analysis, as explained later in this paper), so we also do not consider $p(\mathbf{V}_{k_P})$. On the other hand, output measurement data may be subject to sensor errors ($\mathbf{n}$), and such errors are assumed to Gaussian (mean and covariance are estimated from data). The most challenging aspect of uncertainty modeling is the characterization of $p(\mathbf{U}_{k_P})$. This probability distribution should incorporate as much as possible any knowledge about how the rover will be driven in the future, and so depends on the particular driving scenario under consideration.

The navigation of a rover depends on several factors such as the mission goals, ground terrain, desired navigation speed, etc. Our prediction model requires defining the wheel speed inputs, which in turn defines the battery power consumed, and, ultimately, when EOD will occur. The commanded wheel speeds depend, obviously, on the type of driving scenario being considered. In this paper, we consider two different types of driving scenarios: *unstructured driving*, and *structured driving*. Unstructured driving refers to free driving of the rover by an unplanned sequence of maneuvers, either by an operator or autonomously. Structured driving refers to the situation where the rover needs to navigate through a predetermined set of waypoints in an open terrain. We must transform information regarding these two scenarios into future commanded wheel speeds, $v_L(k)$ and $v_R(k)$, in order to quantify the uncertainty in $\mathbf{U}_{k_P}$.

3.2.1 Unstructured Driving

In unstructured driving, the rover may take one of the following three maneuvers:

[2]Note that C_{b_0}, C_{b_1}, C_{b_2}, and C_{b_3} are simply fitting parameters and do not have physical meaning.

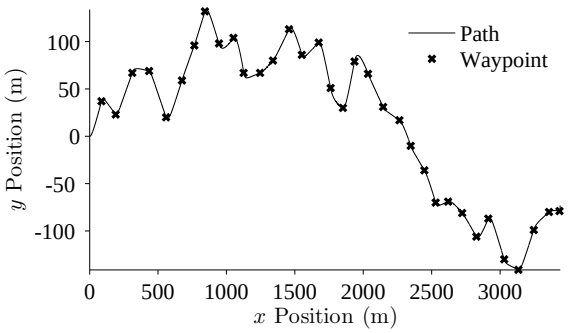

Fig. 3. Rover path for structured driving.

1. Straight: when $v_L = v_R \neq 0$;
2. Turning: when $v_L \neq v_R$, for $v_L > 0$ and $v_R > 0$; and
3. Stopping: when $v_L = v_R = 0$, and $P > 0$ but small.

We assume that in each maneuver segment, the commanded wheel velocities remain constant. So, the surrogate variables include the maneuvers, their time durations, and the wheel speeds. For each maneuver, we must sample a time duration and either zero, one, or two different wheel speeds, for stopping, straight, and turning maneuvers, respectively. We first sample the maneuver and its time duration, and then the appropriate number of commanded speeds. So, we need to specify distributions for the maneuver selection, the time duration, and the wheel speeds conditioned on the selected maneuver. From these surrogate variables, we can construct completely the trajectories for v_L and v_R.

From past unstructured driving scenarios, we can estimate the discrete probability distribution for maneuver selection, the probability distribution for mauever duration, and for each manuever, the probability distributions for the wheel speeds.

3.2.2 Structured Driving

In structured driving, the rover is required to traverse a predetermined sequence of waypoints. Between each pair of waypoints in the sequence, the rover is commanded to go one of three speeds, S_L, S_M, and S_U, representing slow speed, normal speed, and fast speed, respectively. A sample set of waypoints and the associated rover path are shown in Fig. 3.

For structured driving, the trajectories for v_L and v_R are ultimately determined from the waypoints. As it is clear from the figure, the rover does not take a straight path from one waypoint to the other. This is because, in reality, the rover is controlled to maintain a given forward speed while turning towards the desired waypoint. As a result, (*i*) the actual path taken between two sequential waypoints is curved, and the distance traveled is actually larger than the straight-line distance between the waypoints, and (*ii*) since the rover requires more power to turn than to move forward (see Eq. 16), the power used when going between two sequential waypoints is not constant.

To simplify the problem, we transform to an equivalent set of waypoints placed along the same axis, for which the distance between consecutive waypoints will be longer (due to the turns in the actual path) and the power demands will be higher than that required for traveling that same distance without turning (traveling a given distance with turns requires more power than traveling that same distance without turns, as specified by Eq. 16). So, instead of constructing trajectories for v_L and v_R we perform a change of inputs to

$$\mathbf{u} = \left[v\ P\right]^T, \tag{21}$$

constructing trajectories for v and P.

So, surrogate variables are needed to describe trajectories for v and P. Given a waypoint sequence, we consider two surrogate variables for each consecutive pair of waypoints: the distance bias (defined as the additional distance beyond the straight-line distance between the two waypoints) and the constant (average) power. With probability distributions for these surrogate variables, we can construct future input trajectories for v and P. At a given prediction time, for n remaining waypoints, $2n$ surrogate variables are required to describe $p(\mathbf{U}_{k_P})$.

We assume our surrogate variables are normally distributed, defined by a mean and variance. In order to compute the statistics, we construct a large set of random structured driving scenarios, and simulate the rover for each of them using the full dynamic model. For each pair of sequential waypoints, we compute the average power and distance bias as a function of commanded speed (S_L, S_M, or S_U). We then compute the mean and variance of average power and distance bias. In constructing a realization of the future input trajectory, for a given sequence of waypoints, we compute the straight-line distance between each pair of sequential waypoints, sample a realization of the distance bias for the given commanded forward speed for this waypoint pair, and sample a realization of the average power. Using this, we can then construct corresponding realizations of $v(k)$ and $P(k)$ for all $k \geq k_P$, and thus define a realization of $\mathbf{U}_{k_P}$.

4 Estimation

In order to accurately predict the future behavior of the system, we must first estimate its state. For this purpose, we use the unscented Kalman filter (UKF) [33, 34]. Here, we summarize the UKF and refer the reader to [33, 34] for mathematical and implementation details, and to [35] for its application to prognostics.

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* [33]. $\mathcal{X}^i$ 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 $\mathcal{Y}$. There are several different implementations of the unscented trans-

form with different features; here, we use the symmetric unscented transform, in which $2n_x+1$ sigma points are selected symmetrically about the mean as described in [34]. It has a free parameter, κ, used to tune the higher-order moments, for which we choose the recommended value of $3-n_x$, which, in our case, is -5.

In the UKF, first, n_s sigma points $\hat{\mathcal{X}}_{k-1|k-1}$ are derived from the current mean $\hat{\mathbf{x}}_{k-1|k-1}$ and covariance estimates $\mathbf{P}_{k-1|k-1}$ using a sigma point selection algorithm. A one-step-ahead prediction is made using the system model, which is then corrected based on the observations.

For the rover, the model described in Section 3.1 is used. As inputs, the UKF takes $\mathbf{u}(t) = \begin{bmatrix} v_L & v_R \end{bmatrix}^T$. It provides a mean and covariance estimate for the states, $\mathbf{x} = \begin{bmatrix} x & y & \theta & d & t_d & q_b & q_{sp} & q_s \end{bmatrix}^T$.

5 Prediction

In this section, we describe the approach to the prediction problem, which includes prediction of the probability distributions for k_E, Δk_E, $\mathbf{z}_E(k_P)$, and $\Delta\mathbf{z}_E(k_P)$. Although we do not consider uncertainty in the parameters or process noise in our application, we present the general method that incorporates these uncertainties.

5.1 Prediction as an Uncertainty Propagation Problem

Section 2 illustrated that k_E depends on the initial state, $\mathbf{x}(k_P)$; the parameter trajectory up to k_E, $\Theta_{k_P} = [\theta(k_P),\ldots,\theta(k_E)]$; the process noise trajectory up to k_E, $\mathbf{V}_{k_P} = [\mathbf{v}(k_P),\ldots,\mathbf{v}(k_E)]$; and the input trajectory up to k_E, $\mathbf{U}_{k_P} = [\mathbf{u}(k_P),\ldots,\mathbf{u}(k_E)]$. Consider one realization of each of these uncertain quantities at prediction time k_P. Then, the corresponding realization of k_E can be computed by simulating the system model until the threshold T_E evaluates to 1. The time-index at which this event happens is k_E. This procedure can be graphically represented, as in Fig. 4.

Two functions – $R = P(\Xi)$ and $R = P^{\lambda}(\Omega)$ – are explained in Fig. 4. R represents the set of predictions that is uncertain and $\mathbf{r}$ represents a realization of R. The inputs to the function P consist of the present state ($\mathbf{x}(k_P)$), the input loading trajectory ($\mathbf{U}_{k_P}$), the process noise trajectory ($\mathbf{V}_{k_P}$), and the model parameter trajectory (Θ_{k_P}). A concatenated vector of these inputs is represented by Ξ. Since these quantities are uncertain, a realization of Ξ is denoted by ξ. On the other hand, the inputs to the function P^{λ} consist of the present state ($\mathbf{x}(k_P)$) and the surrogate variables corresponding to loading (λ_θ), process noise (λ_u), and model parameter (λ_v) trajectories. A concatenated vector of these quantities is represented by Ω. Similarly, the quantities in Ω are also uncertain, and a realization of Ω is represented by ω.

For every trajectory, it is necessary to specify a function that can generate the entire trajectory as a function of the surrogate variables. In Fig. 4, there are three trajectories and `constructΘ`, `constructU`, and `constructV` serve as those functions to generate model parameter, loading, and process noise trajectories, respectively. Note that, in the case of the trajectory of the model parameters, the model parameter values at k_P are obtained through the estimator, and therefore, this estimate is also used along with the model parameter surrogate variables to generate the trajectory.

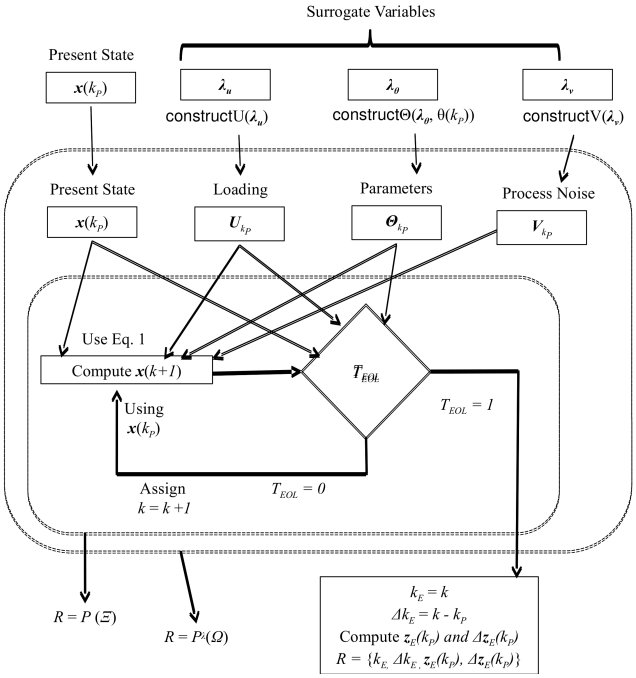

Fig. 4. Definition of P

Algorithm 1 $\mathbf{r} \leftarrow P(\xi)$

$[\mathbf{x}(k_P), \Theta_{k_P}, \mathbf{U}_{k_P}, \mathbf{V}_{k_P}] \leftarrow \xi$
$k \leftarrow k_P$
$\mathbf{x}(k) \leftarrow \mathbf{x}(k_P)$
$\mathbf{z}(k_P) \leftarrow \mathbf{g}(\mathbf{x}(k), \Theta_{k_P}(k), \mathbf{U}_{k_P}(k))$
while $T_E(\mathbf{x}(k), \Theta_{k_P}(k), \mathbf{U}_{k_P}(k)) = 0$ **do**
  $\mathbf{x}(k+1) \leftarrow \mathbf{f}(k, \mathbf{x}(k), \Theta_{k_P}(k), \mathbf{U}_{k_P}(k), \mathbf{V}_{k_P}(k))$
  $k \leftarrow k+1$
  $\mathbf{x}(k) \leftarrow \mathbf{x}(k+1)$
end while
$k_E \leftarrow k$
$\Delta k_E \leftarrow k_E - k_P$
$\mathbf{z}_E \leftarrow \mathbf{g}(\mathbf{x}(k), \Theta_{k_P}(k), \mathbf{U}_{k_P}(k))$
$\Delta\mathbf{z}_E \leftarrow \mathbf{z}_E - \mathbf{z}(k_P)$
$\mathbf{r} \leftarrow [k_E \ \Delta k_E \ \mathbf{z}_E \ \Delta\mathbf{z}_E]^T$

Algorithm 2 $\mathbf{r} \leftarrow P^{\lambda}(\omega)$

$[\mathbf{x}(k_P), \theta(k_P), \lambda_\theta, \lambda_u, \lambda_v] \leftarrow \omega$
$\Theta_{k_P} \leftarrow \texttt{constructΘ}(\lambda_\theta, \theta(k_P))$
$\mathbf{U}_{k_P} \leftarrow \texttt{constructU}(\lambda_u)$
$\mathbf{V}_{k_P} \leftarrow \texttt{constructV}(\lambda_v)$
$\mathbf{r} \leftarrow P([\mathbf{x}(k_P), \Theta_{k_P}, \mathbf{U}_{k_P}, \mathbf{V}_{k_P}])$

The functions P and P^{λ} are described in terms of the random variables in Fig. 4. For the purpose of computation, these two functions can also be described in terms of realizations of these random variables as $\mathbf{r} = P(\xi)$ and $\mathbf{r} = P^{\lambda}(\omega)$, respectively. These functions are presented as Algorithm 1 and Algorithm 2, respectively.

Algorithm 3 $\{\mathbf{r}^{(i)}\}_{i=1}^{N_{sam}} = \mathtt{MC}(f_\Omega(\omega), N_{sam}))$

1: **for** $i = 1$ **to** N_{sam} **do**
2: $\quad \omega^{(i)} \sim f_\Omega(\omega)$
3: $\quad \mathbf{r}^{(i)} \leftarrow P^\lambda(\omega^{(i)})$
4: **end for**

Since it is easier to define trajectories in terms of surrogate variables, the uncertainty in the surrogate variables is represented using probability distributions, and the function P^λ is used for prediction purposes. The goal in prediction is to compute the probability distribution of R, given the probability distributions of Ω, which consists of the present state and the surrogate variables corresponding to input, parameter, and process noise trajectories. The computation of the probability distribution of R is, fundamentally, an uncertainty propagation problem where it is necessary to propagate the uncertainty in Ω through this procedure in order to obtain the distribution for R [24, 27], and thereby estimate the uncertainty in the predicted quantities.

5.2 Monte Carlo Sampling

Monte Carlo sampling is the most straightforward approach to compute the uncertainty in the prediction of R. Several realizations of the state, parameter trajectory, input trajectory, and process noise trajectory are sampled from their respective distributions. It is simple to generate multiple realizations of the state variables, but in the case of trajectories, a two-step approach is followed; first, several realizations of the corresponding surrogate variables are sampled, and then the corresponding realizations of the trajectories are generated. For each realization, $\mathbf{r}$ is computed. The resulting set of $\mathbf{r}$ values characterizes its distribution; kernel density estimation can be used to construct the probability density function of R based on these values of $\mathbf{r}$. Algorithm 3 shows the Monte Carlo prediction algorithm, where ω represents the concatenation $[\mathbf{x}(k_P),\ \theta(k_P),\ \lambda_\theta,\ \lambda_u,\ \lambda_v]$, and N_{sam} the number of Monte Carlo samples.

The computational complexity of Monte Carlo sampling is driven by N_{sam}, the number of evaluations of P, which also determines the accuracy. It can be shown that the covariance estimate (δ_F) of the cumulative distribution function (F) value is related to the number of samples (N_{sam}) as [36]:

$$\delta_F = \frac{\sigma_F}{F} = \sqrt{\frac{(1-F)}{FN_{sam}}} \tag{22}$$

Ideally, it is desired that δ_F is small, in order to ensure accuracy. δ_F approaches zero as N_{sam} approaches infinity, i.e., the accuracy increases with the number of samples.

5.3 Analytical First-Order Reliability Method

While the previous section discussed a sampling-based approach to predict the uncertainty in $\mathbf{R}$, this section discusses an *analytical*, *optimization-based* method, the inverse first-order reliability method, for this purpose. These methods focuses on calculating the cumulative distribution of $\mathbf{R}$ while the previously described Monte Carlo approach directly generates samples from the probability distribution of $\mathbf{R}$. An important advantage of these methods is that they provide measures of sensitivity of the output to the different sources of uncertainty and hence, preliminary studies can be helpful to isolate insignificant sources of uncertainty so that such quantities can be treated deterministically (process noise and parameter uncertainty, in the case study later presented in this paper).

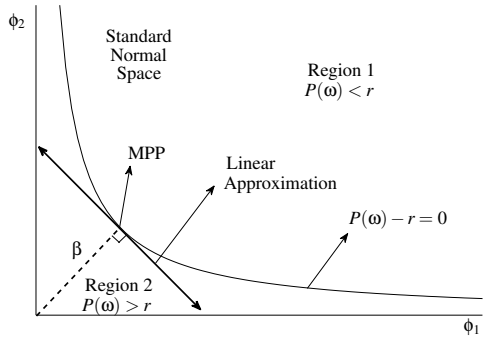

Fig. 5. Most Probable Point Estimation

The basic concept of the inverse FORM approach is to linearize the curve represented by the equation $r_q = P_q^\lambda(\omega)$ and transform all the variables in Ω to the uncorrelated standard normal space (where each variable follows the standard Gaussian distribution with zero mean and unit variance) using well-defined, popular transformation functions [36] (denoted by T_N and the corresponding inverse-transformation denoted by T_N^{-1} in Algorithm 4). Thus, R_q can be expressed as a linear sum of Gaussian variables, and therefore the probability distribution of R_q can be computed easily.

For the purpose of implementation, this method considers each output quantity (k_E, Δk_E, $\mathbf{z}_E$, and $\Delta\mathbf{z}_E$) separately. In general, let N_q denote the number of output quantities of interest, and these outputs are denoted as $\mathbf{r} = \{r_q\}_1^{N_q}$. Consider a vector of functions denoted by $r_q = P_q^\lambda(\omega)$, for $q = 1$ to N_q. The goal is calculate the uncertainty in R_q (r_q is a realization of R_q) through the cumulative distribution function of R_q, denoted by $F_{R_q}(r_q) = \eta$. While the FORM approach calculates the value of η corresponding to a given value of r_q, the inverse FORM approach calculates the value of r_q corresponding to a given value of η. By repeating the FORM procedure for multiple values of r_q, or by repeating the inverse FORM procedure for multiple values of η, the entire CDF $F_{R_q}(r_q)$ can be calculated. In a practical scenario,it is easier to select values of η (say, 0.1, 0.2, 0.3 and so on until 0.9) which span the entire probability range and implement the inverse FORM procedure for each of these η values.

Similar to the Monte Carlo algorithm, the joint probability distribution of ω (representing the concatenation $[\mathbf{x}(k_P),\ \theta(k_P),\ \lambda_\theta,\ \lambda_u,\ \lambda_v]$) is input to the inverse-FORM algorithm as shown in Algorithm 4. In addition, N_q, the number of η

Algorithm 4 $\{\mathbf{r}^{(i)}, \eta^{(i)}\}_{i=1}^{N} \leftarrow$ `InverseFORM`$(f_\Omega(\omega), N_q, N_\eta, M_\omega, T_N, T_N^{-1})$

for $q = 1$ **to** N_q (For each output quantity of interest) **do**
  for $i = 1$ **to** N_η (For each η value) **do**
    $\beta^{(i)} \leftarrow -\Phi^{-1}(\eta^{(i)})$
    $\omega_0 \leftarrow$ Select initial guess for optimization
    Convergence $= 0$, $j = 0$ {Initialize optimization loop}
    while Convergence $\leftarrow 0$ **do**
      $\phi_j \leftarrow T_N(\omega_j)$ {Transform to Standard Normal Space}
      $\phi_j \leftarrow [\phi_{jk}; k = 1 \text{ to } M_\omega]$
      $\alpha_j \leftarrow [\alpha_{jk}; k = 1 \text{ to } M_\omega]$ where $\alpha_{jk} = \frac{\partial P_q^\lambda}{\partial \phi_{jk}}$
      $\phi_{j+1} \leftarrow -\frac{\alpha_j}{|\alpha_j|} \times \beta^{(i)}$
      $\omega_{j+1} \leftarrow T_N^{-1}(\phi_{j+1})$ {Transform to Original Space}
      if $\omega_{j+1} \approx \omega_j$ **then**
        Convergence $\leftarrow 1$
      end if
      $j \leftarrow j + 1$
    end while
    $r_q^{(i)} \leftarrow P_q^\lambda(\omega_j)$
  end for
end for

values (N_η), the number of elements in ω (denoted by M_ω) are also input to the algorithm.

The computational effort (in terms of number of evaluations of P) involved in implementing the inverse-FORM approach for each output quantity (each value of q) can be approximated to be equal to "$4 \times (N_\Omega + 1) \times (N_\eta)$", considering 4 iterations for optimization convergence, "$(N_\Omega + 1)$" evaluations of P for each iteration, and N_η repetitions of the inverse-FORM algorithm (one of each η value).

6 Results

In this section, we demonstrate the prognostics framework for prediction of EOD (k_E), and RDT/RDD ($\Delta \mathbf{z}_E$). We use the relative accuracy (RA) metric as defined in [37] as a measure of accuracy and relative standard deviation (RSD) as a measure of spread for the predictions. We present some scenarios in detail and provide overall performance results culled from a large, comprehensive set of simulation experiments.

We use the full dynamic model of the rover described in [28] as a stand-in for the real system. Since the system model described in Section 3 is an abstraction of this model, there are some actual parameter uncertainties and process noise, but these sources of uncertainty are negligible compared to the future input trajectory uncertainty. The insignificance of these sources of uncertainties were concluded based on the sensitivities obtained as a result of the inverse-FORM methodology, during a preliminary, offline study. Thus, as described in Section 3, we assume that all parameters are known exactly and no process noise is present. We consider only initial state uncertainty, related to the state estimate obtained by the UKF, and the future input uncerainty. In each experiment, predictions are made every 500 s until EOD, and the accuracy and precision metrics are averaged over all these

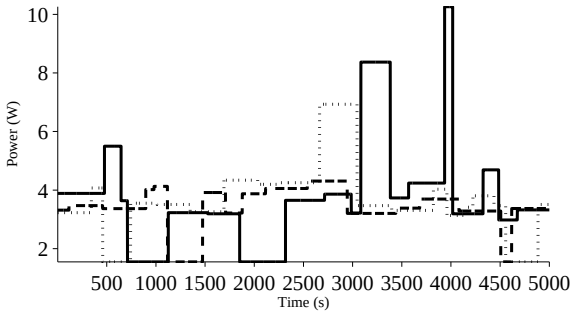

Fig. 6. Example power trajectories for unstructured driving.

predictions. In the following plots, the * superscript indicates the ground truth values. Results for unstructured driving are presented in Section 6.1, followed by results for structured driving in Section 6.2.

6.1 Unstructured Driving

We generate a set of unstructured driving scenarios as follows. We first select a random maneuver (moving straight, turning, or stopping) with probabilities p_1, p_2, and p_3 associated with each of the above maneuvers ($p_1 + p_2 + p_3 = 1$). Second, the duration of the maneuver is sampled from a uniform distribution with bounds $[50, 500]$ s. If the sampled maneuver is "straight", then the forward speed value is sampled. If "turning", then two speed values (for the left and right-side wheels) are sampled. If "stopping", then no speed value is sampled. Here, we use $p_1 = 1/2$, $p_2 = 1/3$, and $p_3 = 1/6$, and all speed values are sampled from a uniform distribution with bounds $[0.315, 0.630]$ m/s.

Due to the vast number of possible unstructured driving trajectories, there is a significant amount of uncertainty in the wheel speeds, and, correspondingly, the future power usage, as shown by Fig. 6. Power is maximum when the rover is turning with a large speed differential between the two sides of the rover, and minimum when the rover is stopped.

As discussed in Section 3.2.1, the surrogate variables here are the maneuvers, their durations, and the corresponding wheel speeds. Since we do not know in advance how many maneuvers will be executed before EOD, it is difficult to use inverse FORM, and so Monte Carlo sampling is used here (1000 samples). In this case, we assume the probability distributions can be accurately determined, and the prediction algorithm uses the same distributions as used to generate the unstructured driving scenarios.

We first demonstrate the prognosis approach for unstructured driving using an example. Battery voltage and current are shown in Figs. 7 and 8. The voltage rises and falls with decreasing and increasing current, respectively, and drops precipitously towards 2.5 V as EOD is approached. Prediction results are shown in Figs. 9-11. Due to the significant uncertainty in the possible future input trajectories, accuracy and precision are poor, with relative accuracy averaging to 78.87%, with RSD to 13.41%. The prediction algorithm captures the set of possible trajectories, but the rover only performs one of those trajectories. So, if the actual trajectory is

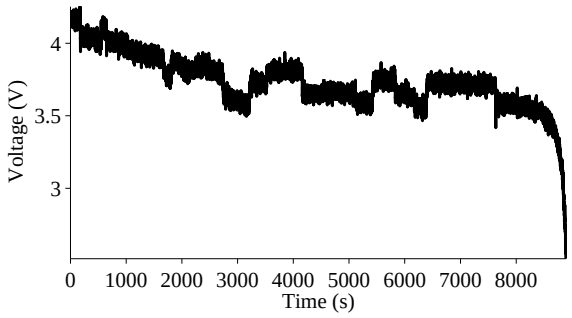

Fig. 7. Battery voltage.

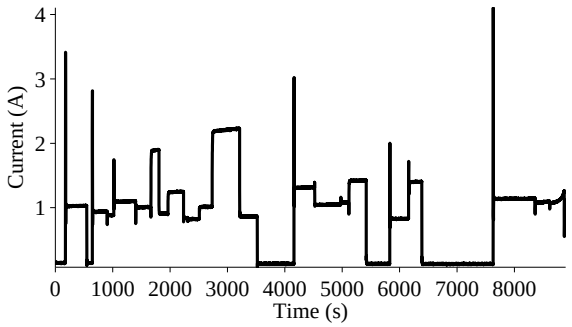

Fig. 8. Battery current.

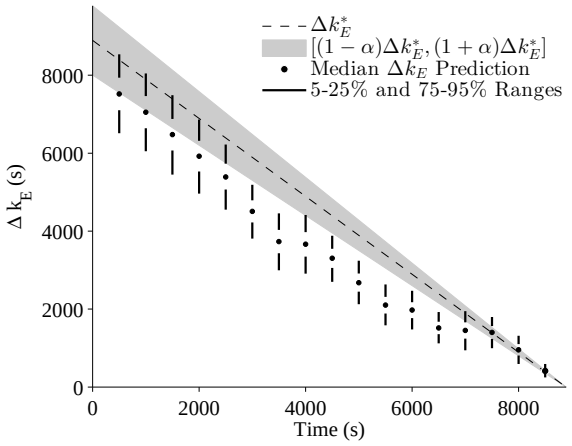

Fig. 9. Δk_E prediction results using Monte Carlo.

not close to the most probable trajectory in terms of power demands, then the predictions will not be very accurate, but the predicted spread should contain the actual trajectory that occurred. If the actual power demands are more than than the most probable, the mean predictions will underestimate EOD, and if smaller, they will overestimate EOD. In this scenario, initially EOD is overestimated, because future power demands are higher than on average. As EOD is approached, accuracy improves since the battery state is being continuously tracked with high accuracy.

We expect the RDT and RDD predictions, besides being more useful to an operator and automated planners, to be more accurate and precise, because they are less sensitive to the rover inputs than EOD. For example, if one input

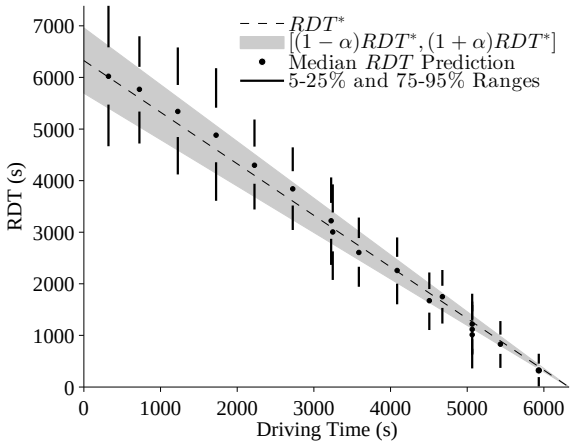

Fig. 10. RDT prediction results using Monte Carlo.

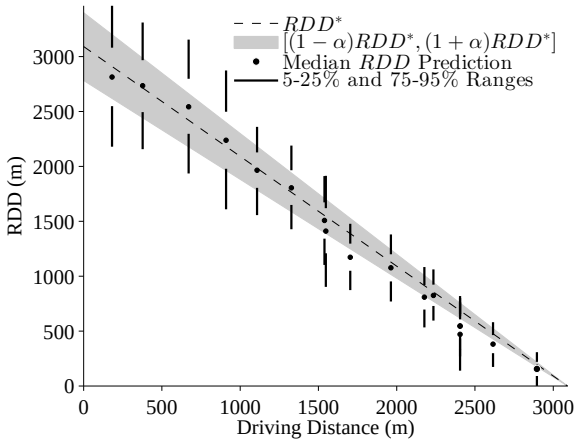

Fig. 11. RDD prediction results using Monte Carlo.

trajectory at a given speed has twice as many stops as another of the same speed, EOD will be twice as large. On the other hand, RDT and RDD will remain approximately the same. RDD is the least sensitive to the future inputs, e.g., for a trajectory going twice the speed as another the battery will discharge in half the time, so driving time will be halved but driving distance will remain unchanged. Here, for RDT, RA averages to 91.16% and RSD to 20.77%; for RDD, RA averages to 94.56% and RSD to 20.25%. Clearly, these predictions are much more accurate.

We performed over 100 different unstructured driving scenarios. For Δk_E, RA averages to 86.55% and RSD to 12.71%. For RDT, RA averages to 89.63% and RSD to 18.76%. For RDD, RA averages to 89.26 and RSD to 19.39%. Despite the large amount of uncertainty in the potential trajectories, accurate predictions of RDT and RDD can be achieved.

6.2 Structured Driving

We generate a set of structured driving scenarios as follows. First, it is necessary to select a set of waypoints for navigation on the *x*-*y* co-ordinate axes. It is assumed that

Table 3. Input Trajectory Surrogate Variable Statistics

Commanded Speed	Variable	Mean	Variance
0.3150 m/s	Average Power (W)	2.93	0.0020
0.3150 m/s	Distance Bias (m)	0.79	0.2600
0.4725 m/s	Average Power (W)	3.64	0.0023
0.4725 m/s	Distance Bias (m)	1.21	0.6900
0.6300 m/s	Average Power (W)	4.37	0.0027
0.6300 m/s	Distance Bias (m)	1.88	2.2500

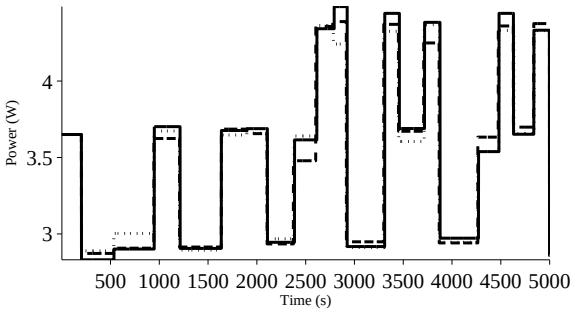

Fig. 12. Sampled power trajectories for structured driving.

the rover starts navigation from an edge of the exploration region and therefore, the rover keeps moving away from the origin with respect to one of the axes. Without loss of generality, this axis is chosen to be the x-axis. The location of the waypoint is selected by drawing a random Δx value (uniformly from a distribution with bounds $[75, 125]$ m) and a random Δy value (uniformly from a distribution with bounds $[-50, 50]$ m). If (x_P, y_P) denotes the location of the current waypoint, then the location of the next waypoint is given by ($x_P + \Delta x$, $y_P + \Delta y$). Without loss of generality, the starting point is chosen to be the origin. The commanded forward speed between two waypoints is constant and is randomly chosen from a discrete probability distribution for three different speeds ($S_L = 0.3150$ m/s, $S_M = 0.4725$ m/s, and $S_U = 0.6300$ m/s), with each speed being equally likely

As discussed in Section 3.2.2, we require $2n$ surrogate variables for n remaining waypoints: for each pair of sequential waypoints, average power and driving distance are random. We compute the statistics of the distributions of the surrogate variables, as a function of commanded forward speed, by analyzing past structured driving scenarios, given in Table 3. As speed increases, both the mean and variance of both power and the distance bias increase. Fig. 12 shows sampled power trajectories for the structured driving case, for a given set of waypoints, based on these statistics. Because the variance on the distance bias is relatively small, the sampled trajectories all arrive at the waypoints at roughly the same times. The variance on the power usage between two waypoints is relatively larger, and accounts for the differences in power levels shown in the figure.

We first demonstrate the prognosis approach for structured driving using an example. Battery voltage and current are shown in Figs. 13 and 14, and prediction results in Figs. 15 and 16. The predictions are very accurate, with an average RA of 98.64%, and with average RSD of 0.46% for

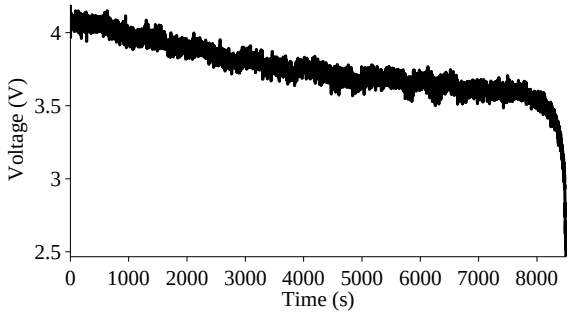

Fig. 13. Battery voltage.

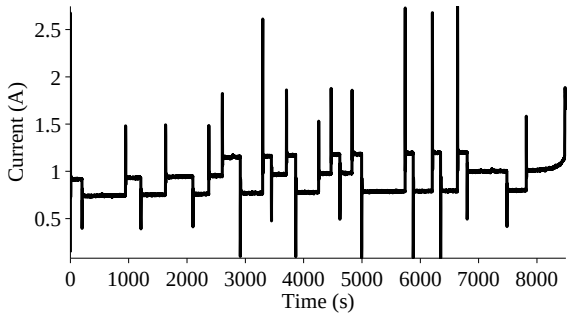

Fig. 14. Battery current.

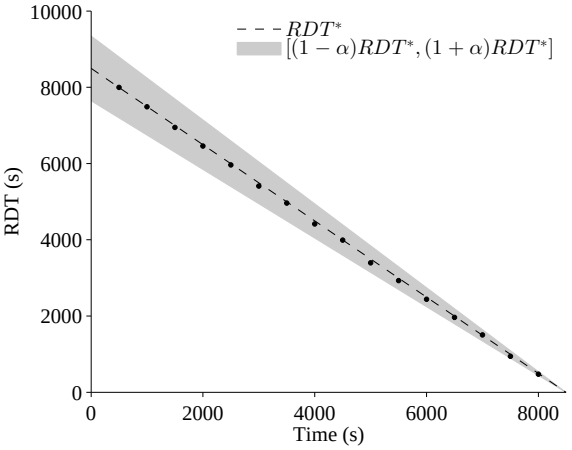

Fig. 15. RDT predictions using inverse FORM.

the RDT predictions (in this case, RDT is equivalent to Δk_E since the rover does not stop). For the RDD predictions, RA averages to 98.44 and RSD to 0.66.

We performed 25 total scenarios to confirm that the approach consistently achieves accurate and precise prediction results. For RDT, RA averages over all the scenarios to 98.42%, and RSD to 0.55%. For RDD, RA averages to 98.52% and RSD to 0.71%. For comparison, we applied Monte Carlo sampling (1000 samples) as the prediction algorithm to these scenarios, and similar results are achieved. For RDT, RA averages to 98.41% and RSD to 0.68%. For RDD, RA averages to 98.55% and RSD to 0.65%.

These results clearly show the reduction in uncertainty in the structured driving scenario. Since state estimation er-

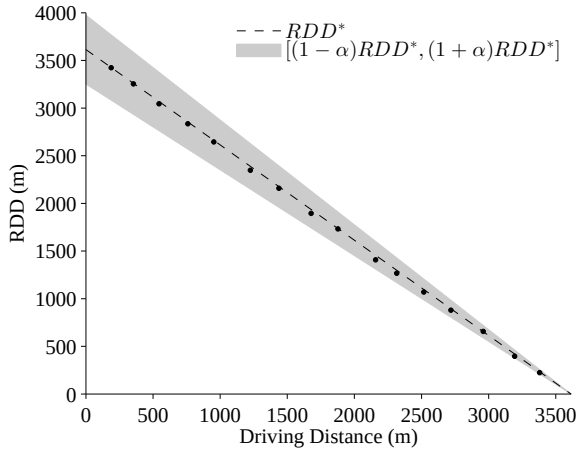

Fig. 16. RDD predictions using inverse FORM.

ror is the same for the two driving scenarios, the key difference is in the future inputs. Since much more information is known for structured driving, these scenarios are much more constrained than for unstructured driving, and hence have less variability and associated uncertainty. This only enforces the point that all information known about future input trajectories should be considered in modeling the uncertainty. The more information, the less uncertainty, and the more useful the predictions will be.

7 Conclusions

In this paper, we presented a general computational framework for predicting end-of-discharge, the remaining driving time, and the remaining driving distance of planetary rovers, by systematically accounting for the different sources of uncertainty that affect rover operations. For this purpose, we developed a prognostics architecture that consists of an estimation step, using the unscented Kalman filter, and a prediction step, using Monte Carlo sampling and the inverse first-order reliability method. Future input trajectories were characterized using the concept of surrogate variables, several simulations were performed to estimate the uncertainty in such trajectories for two types of driving scenarios (structured and unstructured). The quality of the prediction was assessed through comparison against the ground truth and satisfactory performance was achieved. The results demonstrate that, besides being more useful predictive quantities, RDT and RDD predictions are more accurate and precise than EOD predictions, since they are less sensitive to variability in the rover inputs.

In future work, we will consider other realistic driving scenarios for the rover. Further, we will also incorporate system-level modeling approaches into our framework and focus on providing useful information for decision-making activities [38] such as fault mitigation, path planning, mission routing, etc. For example, while we assumed for structured driving that the waypoints were given, the method described here can be used within a larger decision-making framework that considers different sets of possible waypoint plans and selects an optimal path by comparing the RDT and RDD predictions corresponding to each of the different sets of waypoint plans.

Acknowledgements

This work was funded in part by the NASA System-wide Satefy Assurance Technologies (SSAT) project under the Aviation Safety (AvSafe) Program of the Aeronautics Research Mission Directorate (ARMD), and by the NASA Automated Cryogenic Loading Operations (ACLO) project under the Office of the Chief Technologist (OCT) of Advanced Exploration Systems (AES).

References

[1] Koren, Y., and Borenstein, J., 1991. "Potential field methods and their inherent limitations for mobile robot navigation". In Proc. of IEEE International Conf. on Robotics and Automation, pp. 1398–1404.

[2] Brooks, R., 1986. "A robust layered control system for a mobile robot". *IEEE Journal of Robotics and Automation,* **2**(1), pp. 14–23.

[3] Ge, S. S., and Cui, Y. J., 2000. "New potential functions for mobile robot path planning". *IEEE Trans. on Robotics and Automation,* **16**(5), pp. 615–620.

[4] Hu, Y., and Yang, S. X., 2004. "A knowledge based genetic algorithm for path planning of a mobile robot". In Proc. of IEEE Intl. Conf. on Robotics and Automation, Vol. 5, pp. 4350–4355.

[5] Lee, M. K., Kiesler, S., Forlizzi, J., Srinivasa, S., and Rybski, P., 2010. "Gracefully mitigating breakdowns in robotic services". In Proc. of ACM/IEEE Intl. Conf. on Human-Robot Interaction, pp. 203–210.

[6] Dixon, W., Walker, I., and Dawson, D., 2001. "Fault detection for wheeled mobile robots with parametric uncertainty". In 2001 IEEE/ASME Intl. Conf. on Advanced Intelligent Mechatronics, Vol. 2, pp. 1245–1250.

[7] Cesarone, J., and Eman, K. F., 1989. "Mobile robot routing with dynamic programming". *Journal of Manufacturing Systems,* **8**(4), pp. 257–266.

[8] Yuan, B., Orlowska, M., and Sadiq, S., 2007. "On the optimal robot routing problem in wireless sensor networks". *IEEE Trans. on Knowledge and Data Engineering,* **19**(9), pp. 1252–1261.

[9] Oliva, J., Weihrauch, C., and Bertram, T., 2013. "A model-based approach for predicting the remaining driving range in electric vehicles". In Annual Conf. of the Prognostics and Health Management Society, pp. 438–448.

[10] Vichare, N. M., and Pecht, M. G., 2006. "Prognostics and health management of electronics". *IEEE Trans. on Components and Packaging Technologies,* **29**(1), pp. 222–229.

[11] Sheppard, J. W., Kaufman, M. A., and Wilmer, T. J., 2009. "Ieee standards for prognostics and health man-

agement". *Aerospace and Electronic Systems Magazine, IEEE,* **24**(9), pp. 34–41.

[12] Byington, C. S., Watson, M., Edwards, D., and Stoelting, P., 2004. "A model-based approach to prognostics and health management for flight control actuators". In IEEE Aerospace Conf., Vol. 6, pp. 3551–3562.

[13] Orchard, M., and Vachtsevanos, G., 2009. "A particle filtering approach for on-line fault diagnosis and failure prognosis". *Trans. of the Institute of Measurement and Control,* **31**(3-4), June, pp. 221–246.

[14] Daigle, M., and Goebel, K., 2013. "Model-based prognostics with concurrent damage progression processes". *IEEE Trans. on Systems, Man, and Cybernetics: Systems,* **43**(4), May, pp. 535–546.

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

[16] Schwabacher, M., 2005. "A survey of data-driven prognostics". In Proc. of the AIAA Infotech@Aerospace Conf.

[17] Si, X.-S., Wang, W., Hu, C.-H., and Zhou, D.-H., 2011. "Remaining useful life estimation–a review on the statistical data driven approaches". *European Journal of Operational Research,* **213**(1), pp. 1–14.

[18] Daigle, M., Saxena, A., and Goebel, K., 2012. "An efficient deterministic approach to model-based prediction uncertainty estimation". In Annual Conf. of the Prognostics and Health Management Society, pp. 326–335.

[19] Tampier, C., Pérez, A., Jaramillo, F., Quintero, V., Orchard, M. E., and Silva, J. F., 2015. "Lithium-ion battery end-of-discharge time estimation and prognosis based on bayesian algorithms and outer feedback correction loops: A comparative analysis". In Annual conference of the PHM Society.

[20] Ling, Y., Shantz, C., Mahadevan, S., and Sankararaman, S., 2011. "Stochastic prediction of fatigue loading using real-time monitoring data". *Intl. Journal of Fatigue,* **33**(7), pp. 868–879.

[21] Sankararaman, S., Ling, Y., Shantz, C., and Mahadevan, S., 2011. "Uncertainty quantification in fatigue crack growth prognosis". *Intl. Journal of Prognostics and Health Management,* **2**(1).

[22] Saha, B., Quach, C. C., and Goebel, K., 2012. "Optimizing battery life for electric UAVs using a Bayesian framework". In Proc. of the 2012 IEEE Aerospace Conf.

[23] Sankararaman, S., and Goebel, K., 2013. "Why is the remaining useful life prediction uncertain?". In Annual Conf. of the Prognostics and Health Management Society, pp. 337–349.

[24] Sankararaman, S., Daigle, M., Saxena, A., and Goebel, K., 2013. "Analytical algorithms to quantify the uncertainty in remaining useful life prediction". In Proc. of the 2013 IEEE Aerospace Conf.

[25] Daigle, M., and Sankararaman, S., 2013. "Advanced methods for determining prediction uncertainty in model-based prognostics with application to planetary rovers". In Annual Conf. of the Prognostics and Health Management Society, pp. 262–274.

[26] Phoon, K., Huang, S., and Quek, S., 2002. "Simulation of second-order processes using karhunen–loeve expansion". *Computers & Structures,* **80**(12), pp. 1049–1060.

[27] Sankararaman, S., and Goebel, K., 2013. "Uncertainty quantification in remaining useful life of aerospace components using state space models and inverse form". In Proc. of the 15th Non-Deterministic Approaches Conf.

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

[29] Daigle, M., and Goebel, K., 2010. "Improving computational efficiency of prediction in model-based prognostics using the unscented transform". In Proc. of the Annual Conf. of the Prognostics and Health Management Society, pp. 326–335.

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

[31] Saha, B., Goebel, K., Poll, S., and Christophersen, J., 2007. "An integrated approach to battery health monitoring using Bayesian regression and state estimation". In 2007 IEEE Autotestcon, pp. 646–653.

[32] Chen, M., and Rincon-Mora, G. A., 2006. "Accurate electrical battery model capable of predicting runtime and I-V performance". *IEEE Trans. on Energy Conversion,* **21**(2), June, pp. 504 – 511.

[33] Julier, S. J., and Uhlmann, J. K., 1997. "A new extension of the Kalman filter to nonlinear systems". In Proc. of the 11th Intl. Symposium on Aerospace/Defense Sensing, Simulation and Controls, pp. 182–193.

[34] Julier, S. J., and Uhlmann, J. K., 2004. "Unscented filtering and nonlinear estimation". *Proc. of the IEEE,* **92**(3), Mar, pp. 401–422.

[35] Daigle, M., Saha, B., and Goebel, K., 2012. "A comparison of filter-based approaches for model-based prognostics". In Proc. of the 2012 IEEE Aerospace Conf.

[36] Haldar, A., and Mahadevan, S., 2000. *Probability, reliability, and statistical methods in engineering design*. John Wiley & Sons, Inc.

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

[38] Sweet, A., Gorospe, G., Daigle, M., Celaya, J. R., Balaban, E., Roychoudhury, I., and Narasimhan, S., 2014. "Demonstration of prognostics-enabled decision making algorithms on a hardware mobile robot test platform". In Annual Conf. of the Prognostics and Health Management Society, pp. 142–150.