An Efficient Deterministic Approach to Model-based Prediction Uncertainty Estimation

Matthew Daigle[1], Abhinav Saxena[2], and Kai Goebel[1]

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

[2] *SGT, Inc., NASA Ames Research Center, Moffett Field, CA 94035, USA*
abhinav.saxena@nasa.gov

ABSTRACT

Prognostics deals with the prediction of the end of life (EOL) of a system. EOL is a random variable, due to the presence of process noise and uncertainty in the future inputs to the system. Prognostics algorithms must account for this inherent uncertainty. In addition, these algorithms never know exactly the state of the system at the desired time of prediction, or the exact model describing the future evolution of the system, accumulating additional uncertainty into the predicted EOL. Prediction algorithms that do not account for these sources of uncertainty are misrepresenting the EOL and can lead to poor decisions based on their results. In this paper, we explore the impact of uncertainty in the prediction problem. We develop a general model-based prediction algorithm that incorporates these sources of uncertainty, and propose a novel approach to efficiently handle uncertainty in the future input trajectories of a system by using the unscented transform. Using this approach, we are not only able to reduce the computational load but also estimate the bounds of uncertainty in a deterministic manner, which can be useful to consider during decision-making. Using a lithium-ion battery as a case study, we perform several simulation-based experiments to explore these issues, and validate the overall approach using experimental data from a battery testbed.

1. INTRODUCTION

Prognostics deals with the prediction of the end of life (EOL) and remaining useful life (RUL) of components, subsystems, and systems. At its core, prognostics is a prediction problem. But, the future evolution of the system is a random process due to (*i*) process noise, and (*ii*) uncertainty in the future inputs to the system. In practice, these two sources of uncertainty cannot be avoided and thus EOL and RUL are random variables. The prognostics algorithm itself introduces additional uncertainty because, in general, (*i*) it does not know exactly the state of the system at the time of prediction, (ii) it does not know exactly the description of the process noise, (*iii*) it does not know exactly the description of the future input uncertainty, and (*iv*) it does not know exactly the model of the future system behavior. All these sources of uncertainty contribute to the difficulty of the prognostics problem (Sankararaman, Ling, Shantz, & Mahadevan, 2011). While uncertainty cannot be eliminated from prognostics, an accurate assessment can be crucial in decision-making. Making decisions based on uncertain information requires characterizing the uncertainty itself to tune the risk level as needed in a particular application. In safety-critical systems it is of even higher importance, which is reflected in the fact that verification, validation, and certification protocols in the aerospace domain require provably deterministic and bounded systems.

Although the presence of prediction uncertainty is clearly a practical issue, only a few works have explored it. In (Sankararaman et al., 2011), the authors examine the various sources of uncertainty in fatigue crack growth prognostics and analyze their effects in an offline setting. In dealing with input uncertainty, future input trajectories are constructed as sequential blocks of constant-amplitude loading, and such trajectories are sampled in the prediction algorithm. In a similar approach applied to batteries in an unmanned aerial vehicle (UAV), in (Saha et al., 2012) the authors determine statistics of the battery loading for typical UAV maneuvers based on past flight data, and construct future input trajectories as constrained sequences of flight maneuvers. In (Luo et al., 2008), constant loading is assumed for a vehicle suspension system, and predictions are made for a weighted set of three different loading values. The approach of (Edwards et al., 2010) also considers constant loading, and several uncertainty measures are defined that are then used within a framework for system life extension through actions that modify

the system loading.

In this paper, we develop a general framework within the model-based prognostics paradigm for representing arbitrarily complex future input trajectories, and develop a general sample-based algorithm for predicting EOL and RUL that accounts for the sources of uncertainty in the prediction process. In particular, we introduce the unscented transform (Julier & Uhlmann, 1997), which predicts the mean and covariance of a random variable passed through a nonlinear function, as a method to efficiently sample from future input trajectories while still maintaining the statistics of the end result. This approach offers substantial computational savings as compared to a semi-exhaustive random sampling approach. Additionally, we show that since the UT allows sampling in a deterministic manner, and in this particular case where uncertainty in inputs is assumed to be uniformly distributed, we are able to bound the RUL predictions. Therefore, using the UT we realize a threefold benefit, (*i*) obtaining bounds for the prediction uncertainty, (ii) obtaining bounds in a deterministic manner, and (*iii*) keeping the statistical information intact with a considerably reduced computational burden as compared to traditional sampling approaches. Using a lithium-ion battery as a case study, we analyze the impact of uncertainty and various performance trade-offs of the prediction algorithm under several cases, and demonstrate and validate the approach with experimental data from a battery testbed.

The paper is organized as follows. Section 2 formulates the prognostics problem and describes the sources of uncertainty. Section 3 develops the general prediction algorithm and its different instantiations. Section 4 presents the battery case study and provides results using both experimental and simulated data. Section 5 concludes the paper.

2. Prognostics Approach

This section first formulates the prognostics problem. It then describes how uncertainty arises in prognostics, and examines the implications on prognostics algorithms. Finally, it provides an architecture for model-based prognostics.

2.1. 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

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

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

Prognostics and health management is concerned with system performance that lies outside a given region of acceptable behavior. The desired performance is expressed through a set of n_c constraints, $C_{EOL} = \{c_i\}_{i=1}^{n_c}$, where $c_i : \mathbb{R}^{n_x} \times \mathbb{R}^{n_\theta} \times \mathbb{R}^{n_u} \to \mathbb{B}$ 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 $c_i(\mathbf{x}(t), \boldsymbol{\theta}(t), \mathbf{u}(t)) = 1$ if the constraint is satisfied, and 0 otherwise.

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

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

T_{EOL} evaluates to 1 when any of the constraints are violated. EOL is then defined as the earliest time point at which this occurs:

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

RUL is expressed using EOL as

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

2.2. Prediction Uncertainty

The above definitions of EOL and RUL are for their exact values, i.e., the system takes some path *out of many possible paths* through the state space until EOL. The actual path the system will take cannot be known in advance because the system evolution is a random process, therefore, EOL and RUL at any prediction time $t_P < EOL$, are actually random variables. System evolution is random due to the process noise $\mathbf{v}(t)$ and because $\mathbf{u}(t)$ for $t > t_P$ is never known exactly. Since EOL is a function of $(\mathbf{x}(t_P), \boldsymbol{\theta}(t_P))$ and $\mathbf{u}(t)$, which are all random variables, EOL (and RUL) must also be a random variable. This uncertainty is inherent to the system itself and cannot be avoided. Note that as t approaches EOL, the variability in the actual EOL will naturally reduce, simply because $EOL - t$ becomes smaller.

The goal of a prognostics algorithm, then, is to compute the true distribution of the EOL and RUL. A decision that is made based on a misrepresentation of this true distribution could have a significant impact, especially if the true variability is underestimated. It is therefore critical that a prognostics algorithm comes as close to this true distribution as possible.

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

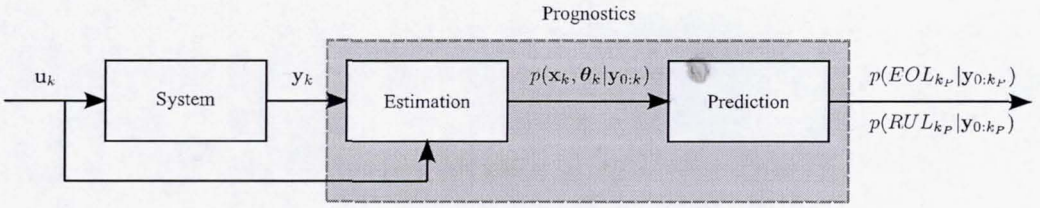

Figure 1. Model-based prognostics architecture.

However, additional uncertainty is also introduced by the prognostics algorithm itself. In order to make a prediction, the state of the system at the time of prediction must be known. At best, only a probability distribution can be estimated since (*i*) the system state may not be directly measured and, (*ii*) even if it is, there is sensor noise, (*iii*) the initial state of the system is not known exactly, (*iv*) the system model is not known exactly, and (*v*) there is process noise. Even if the system state is known exactly, uncertainty is introduced in the prediction process since, in general, (*i*) the model used for prediction is not known exactly, (*ii*) the correct representation of process noise is not known exactly, and (*iii*) the correct representation of the space of possible future input trajectories is not known exactly. Due to these additional sources of uncertainty inherent to the prognostics algorithm, the uncertainty in the predicted EOL/RUL will nominally be larger than the true variability in EOL/RUL.

The uncertainty inherent to the system cannot be eliminated, and neither can the uncertainty inherent to the algorithm. However, the uncertainty associated with the algorithm can be limited by using the best known model including the best known representation of the process noise, and by constraining the space of possible future input trajectories as much as possible and representing the associated probability distribution as accurately as possible. The potential trajectories of $\mathbf{u}(t)$ for $t \geq t_P$ depend on the system being monitored and can, in general, take on any number of arbitrary signals. We denote the space of possible future input trajectories as $\mathcal{U}_{t_P}$. A single trajectory in the set $\mathcal{U}_{t_P}$ is denoted as $\mathbf{U}_{t_P}$, and defines the values of $\mathbf{u}(t)$ for all $t \geq t_P$. Each possible trajectory has a certain probability of occurring in the real system, and so this is defined by a probability distribution $p(\mathcal{U}_{t_P})$. In practice, it is entirely possible that this exact distribution is unknown and must be approximated.

2.3. Prognostics Architecture

To predict EOL/RUL, first an initial state to use for the prediction must be determined. In the model-based paradigm, this is referred to as the *estimation* problem, and requires determining a joint state-parameter estimate $p(\mathbf{x}(t), \boldsymbol{\theta}(t)|\mathbf{y}_{0:t})$ based on the history of observations up to time t, $\mathbf{y}_{0:t}$. The *prediction* problem is to determine, using this estimate, EOL and RUL probability distributions, $p(EOL_{t_P}|\mathbf{y}_{0:t_P})$ and $p(RUL_{t_P}|\mathbf{y}_{0:t_P})$.

The prognostics architecture is shown in Fig. 1 (Daigle & Goebel, 2011). 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 $p(\mathbf{x}_k, \boldsymbol{\theta}_k|\mathbf{y}_{0:k})$.[2] The prediction module uses the joint state-parameter distribution and the system model, along with hypothesized future inputs, to compute EOL and RUL as probability distributions $p(EOL_{k_P}|\mathbf{y}_{0:k_P})$ and $p(RUL_{k_P}|\mathbf{y}_{0:k_P})$ at given prediction times k_P.

3. PREDICTION

Prediction is initiated at a given time of prediction k_P using the current joint state-parameter estimate, $p(\mathbf{x}_{k_P}, \boldsymbol{\theta}_{k_P}|\mathbf{y}_{0:k_P})$. Approaches to determine this estimate are reviewed in (Daigle et al., 2012) and will not be described here. The goal is to compute $p(EOL_{k_P}|\mathbf{y}_{0:k_P})$ and $p(RUL_{k_P}|\mathbf{y}_{0:k_P})$. The representation of $p(\mathbf{x}_{k_P}, \boldsymbol{\theta}_{k_P}|\mathbf{y}_{0:k_P})$ is determined by the algorithm used for the estimation step. In any case, here, we assume it is given as a set of weighted samples $\{(\mathbf{x}_{k_P}^{i_x}, \boldsymbol{\theta}_{k_P}^{i_x}), w_{x,k_P}^{i_x}\}$. In the case of the unscented Kalman filter (Julier & Uhlmann, 1997, 2004) and the particle filter (Arulampalam et al., 2002), the distribution is provided in this format, otherwise, the provided distribution can be sampled.

Here, we follow a sample-based approach (as opposed to analytical methods) to the prediction problem to incorporate the uncertainty (Sankararaman et al., 2011), in which each sample is simulated to EOL. The approach is shown as Algorithm 1. For each of the N_x samples of the state-parameter distribution, we sample from the input distribution, $\mathcal{U}_{k_P}$, N_u future input trajectories (where a single trajectory $\mathbf{U}_{k_P} \triangleq \{\mathbf{u}_{k_P}, \mathbf{u}_{k_P+1}, \ldots\}$) with weights w_{u,k_P}, and for each of these trajectories, simulate N_v trajectories with process noise (lines 10–14). At the end, we obtain a weighted set of EOL predictions for each of these simulations, totaling $N_x \times N_u \times N_v$, i.e., $\{EOL_{k_P}^j, w_{k_P}^j\}_{j=1}^{N_x \times N_u \times N_v}$ (lines 15 and 16).

In the algorithm, line 11 samples the next state from the prior probability distribution. Effectively, this is implemented by

[2]Estimation does not need to be performed by the prognoser if it is provided by some other module, such as a diagnoser (Roychoudhury & Daigle, 2011).

Algorithm 1 EOL Prediction with Uncertainty

1: **Inputs:** $\{(\mathbf{x}_{k_P}^{i_x}, \boldsymbol{\theta}_{k_P}^{i_x}), w_{x,k_P}^{i_x}\}_{i_x=1}^{N_x}, N_u, N_v$
2: **Outputs:** $\{EOL_{k_P}^{j}, w_{k_P}^{j}\}_{j=1}^{N_x \times N_u \times N_v}$
3: $\{\mathbf{U}_{k_P}^{i_u}, w_{u,k_P}^{i_u}\}_{i_u=1}^{N_u} \sim p(\mathcal{U}_{k_P})$
4: **for** $i_x = 1$ **to** N_x **do**
5: **for** $i_u = 1$ **to** N_u **do**
6: **for** $i_v = 1$ **to** N_v **do**
7: $j \leftarrow (i_x, i_u, i_v)$
8: $k \leftarrow k_P$
9: $\mathbf{x}_k^j \leftarrow \mathbf{x}_{k_P}^j$
10: **while** $T_{EOL}(\mathbf{x}_k^j, \boldsymbol{\theta}_k^j, \mathbf{U}_{k_P}^{i_u}(k)) = 0$ **do**
11: $\mathbf{x}_{k+1}^j \sim p(\mathbf{x}_{k+1} | \mathbf{x}_k^j, \boldsymbol{\theta}_{k_P}^j, \mathbf{U}_{k_P}^j(k))$
12: $k \leftarrow k + 1$
13: $\mathbf{x}_k^j \leftarrow \mathbf{x}_{k+1}^j$
14: **end while**
15: $EOL_{k_P}^j \leftarrow k$
16: $w_{k_P}^j \leftarrow w_{x,k_P}^{i_x} \times w_{u,k_P}^{i_u} / N_v$
17: **end for**
18: **end for**
19: **end for**

sampling the process noise and executing the state equation with that process noise. Each of these trajectories individually are set to have equal weight ($1/N_v$), and a statistically meaningful result can only be obtained by sampling a sufficient number of times. The longer the time to EOL, the more of an effect process noise will have and the more samples may be necessary to accurately capture the statistics. If desired, process noise can be set to zero.

Line 3 in the algorithm samples the future input trajectories. What this space looks like is highly dependent on the underlying application, and how to sample this space depends on what the space looks like. It is up to the modeler to define this space and its probability distribution. There are a few simple approaches to take that are generally applicable. One way to handle this is to define a family of parameterized functions that define $\mathbf{u}(t)$ for all $t \geq t_P$. For example, let $\mathbf{u}(t) = p$, where p is an unknown value drawn from a known (or assumed) distribution. To sample an input trajectory one needs only to sample a value for p. Or, let $\mathbf{u}(t) = p_1 t + p_2 t^2$. Here, an input trajectory is sampled by sampling values for p_1 and p_2. More complicated functions may also be defined, and as long as they are parameterized then it is easy to sample such functions. Another general approach is to define the input as a set of blocks where within each block the input is constant, such as in (Sankararaman et al., 2011; Saha et al., 2012). One must then sample how long the next block will last and at what magnitude it will be.

The computational complexity of the algorithm is mainly a function of the number of unique samples ($N_x \times N_u \times N_v$). Secondary to this is how long each sample takes to simulate to EOL. Samples with higher rates of of damage progression (e.g., due to increased loading) will simulate faster than those with lower rates of progression.

The general algorithm presented here can be instantiated in different ways, depending on how the future input trajectories are sampled (line 3). In the following subsections we describe different sampling methods.

3.1. Exhaustive Sampling

If the input trajectory space $\mathcal{U}_{k_P}$ is finite, then it is possible to do predictions over the entire space. This is limited by the imposed computational requirements, because even if the space is finite, the number of discrete elements may be too large. In this case the sampling would be deterministic, so repeated executions would always get the same results.

3.2. Random Sampling

If the input trajectory space is infinitely large or finite but too large for exhaustive sampling, then random sampling may be used to obtain a number of sufficient samples. Because the process is stochastic, the results will be nondeterministic, which could have a significant impact on performance if too few samples are drawn. Further, repeated executions would obtain different results, which could make validation of the algorithm difficult. However, it is the most generally applicable approach.

3.3. Sampling with the Unscented Transform

As an alternative to random sampling, nonexhaustive deterministic sampling can also be performed. One method is to use the unscented transform (UT). The UT takes a random variable $\mathbf{x} \in \mathbb{R}^{n_x}$, with mean $\bar{\mathbf{x}}$ and covariance $\mathbf{P}_{xx}$, that is related to a second random variable $\mathbf{y} \in \mathbb{R}^{n_y}$ by some function $\mathbf{y} = \mathbf{g}(\mathbf{x})$, and computes the mean $\bar{\mathbf{y}}$ and covariance $\mathbf{P}_{yy}$ with high accuracy using a minimal set of deterministically selected weighted samples, called *sigma points* (Julier & Uhlmann, 1997). The number of sigma points is only linear in the dimension of the random variable, and so the statistics of the transformed random variable, i.e., mean and covariance, can be computed much more efficiently than by random sampling.[3]

Here, $\boldsymbol{\mathcal{X}}^i$ 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.,

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

[3] Versions of the unscented transform also exist that compute also higher-order moments like skew (Julier, 1998).

with mean and covariance calculated as

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

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

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

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

where $\left(\sqrt{(n_x + \kappa)\mathbf{P}_{xx}}\right)^i$ refers to the ith column of the matrix square root of $(n_x + \kappa)\mathbf{P}_{xx}$. Note that the required number of samples is only linear in the size of the state space. Here, κ is a free parameter that can be used to tune higher order moments of the distribution. If $\mathbf{x}$ is assumed Gaussian, then selecting $\kappa = 3 - n_x$ is recommended (Julier & Uhlmann, 1997). Note that with the UT, weights may be negative, and are not to be directly interpreted as probabilities.

If we consider the random variable in this case to be a representation of our input space, then the UT can be used to sample the space of input trajectories. Here, the simulation to EOL is the nonlinear transformation, as in (Daigle & Goebel, 2010). A representation of the input space for this framework is required. If the future input trajectories are defined by parameterized functions, where the function parameters are themselves sampled from some distribution, then the input space is defined by these parameters and the UT can be used to sample from this parameter space. The number of samples would be linear in the number of parameters (as this defines the state space for the UT). By using the UT to sample this parameter space, in effect we obtain representative samples of the input trajectory space.

If the input space cannot be transformed to a representation amenable to the UT, then this approach cannot be used. Such cases, however, may not appear often in practice since this would imply that it is difficult for the modeler to define the input space in the first place. That is, an easy and practical way to define the input trajectory space is by sampling a finite set of parameters that define a particular input trajectory.

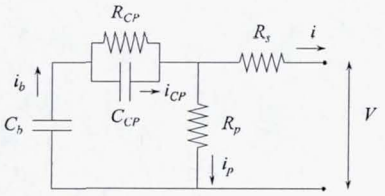

Figure 2. Battery equivalent circuit.

4. Case Study

We select a lithium-ion battery as a case study on which to demonstrate and validate our approach. We first present the battery model. We then apply the approach to experimental data and demonstrate the impact of prognostics uncertainty. We then present a number of simulation experiments to more systematically explore these issues.

4.1. Modeling

The battery model is based on an electrical circuit equivalent shown in Fig. 2, similar to models presented in (Chen & Rincon-Mora, 2006; Barsali & Ceraolo, 2002; Ceraolo, 2000). The large capacitance C_b holds the charge q_b of the battery. The R_{CP}-C_{CP} pair captures the major nonlinear voltage drop due to concentration polarization, R_s captures the so-called I-R drop, and R_p models the parasitic resistance that accounts for self-discharge. This simple battery model is enough to capture the major dynamics of the battery, but ignores temperature effects and other minor battery processes.

The state-of-charge, SOC, is computed as

$$SOC = 1 - \frac{q_{max} - q_b}{C_{max}},$$

where q_b is the current charge in the battery (related to C_b), q_{max} is the maximum possible charge, and C_{max} is the maximum possible battery capacity (i.e., nominally, its rated capacity). The concentration polarization resistance is a nonlinear function of SOC:

$$R_{CP} = R_{CP0} + R_{CP1} \exp R_{CP2}(1 - SOC),$$

where R_{CP0}, R_{CP1}, and R_{CP1} are empirical parameters. The resistance, and, hence, the voltage drop, increases exponentially as SOC decreases (Saha et al., 2012).

Voltage drops across the individual circuit elements are given by

$$V_b = q_b / C_b$$
$$V_{CP} = q_{CP} / C_{CP}$$
$$V_p = V_b - V_{CP},$$

where q_{CP} is the charge associated with the capacitance

Parameter	Value
C_b	9844
R_s	0.143014
R_p	500
C_{CP}	70.3767
R_{CP0}	0.019829
R_{CP1}	3.68606×10^{-14}
R_{CP2}	31.9213
q_{max}	41400
C_{max}	6900

Table 1. Battery Model Parameters

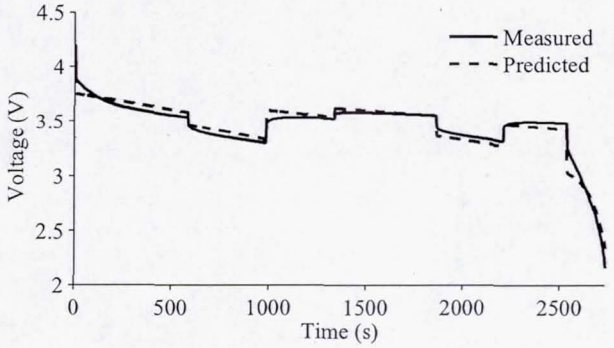

Figure 3. Battery model comparison with experimental data.

C_{CP}. The terminal voltage of the battery is

$$V = V_b - V_{CP} - R_s i,$$

where i is the battery current at the terminals. Currents associated with the individual circuit elements are given by

$$i_p = V_p / R_p$$
$$i_b = i_p + i$$
$$i_{CP} = i_b - V_{CP}/R_{CP}.$$

The charges are then governed by

$$\dot{q}_b = -i_b$$
$$\dot{q}_{CP} = i_{CP}$$

We are interested in predicting end-of-discharge as defined by a voltage threshold V_{EOD}. So, C_{EOL} consists of only one constraint:

$$c_1 : V > V_{EOD}.$$

From experimental data of battery discharges, we have identified the parameters of the battery model through model fitting, and their values are shown in Table 1. A comparison of measured and predicted behavior for known current inputs is shown in Fig. 3. The associated inputs are shown in Fig. 4. Clearly, the model is not perfect, so prognosis will have to account for the model uncertainty.

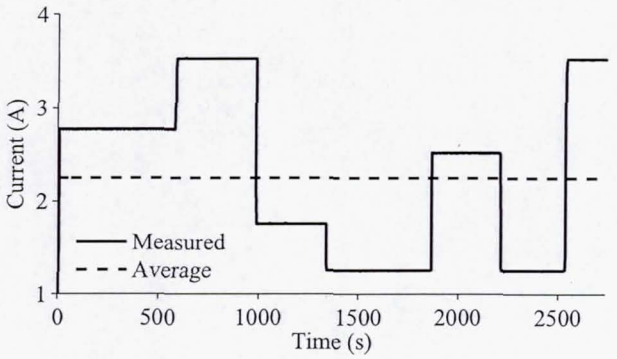

Figure 4. Battery current inputs.

4.2. Experimental Results

To demonstrate and validate the approach, we apply our prognostics algorithms to the experimental data given in Fig. 3. For state estimation, we use the unscented Kalman filter (UKF) (Julier & Uhlmann, 1997; Daigle et al., 2012). Here, we use the UKF since the model is nonlinear and generally performs better (and is easier to apply) than the extended Kalman filter (Julier & Uhlmann, 1997). The model uncertainty is captured through process noise, represented using normal distributions with zero mean. Sensor noise is generally very small but is also assumed to be zero-mean Gaussian.

The battery current is viewed as an input to the model, and its future values must be hypothesized. First, we assume that the future inputs are known exactly (we use the exact load profile available from the experimental data shown in Fig. 4), and that there is no process noise. The RUL predictions versus time are shown in Fig. 5. The predictions are shown against the true RUL (denoted as RUL^*) along with an accuracy cone defined by $\alpha = 0.15$. Predictions are made every 100 s. Here, we see that the predictions are quite accurate and remain within 15% of the true RUL until about 2500 s. The UKF can partially correct for the model uncertainty, but towards the end of the discharge the error cannot be fully corrected and the relative accuracy is reduced, due to the high sensitivity to the final voltage drop. The only uncertainty captured by these predictions is that in the state estimate, which is very small (so is not visible in the figure). Clearly, these predictions do not capture the true uncertainty so would be incorrect to use for decision-making.

Process noise must be correctly represented in order to yield usable results. Fig. 6 shows the RUL predictions versus time in this case, using $N_v = 100$. Now, we see that the uncertainty represented in the prediction covers the decrease in accuracy observed towards the end of the discharge, i.e., the true RUL is now contained within the uncertainty bounds.

If the future inputs are not known, then some assumption must be made about what they look like. First, we assume

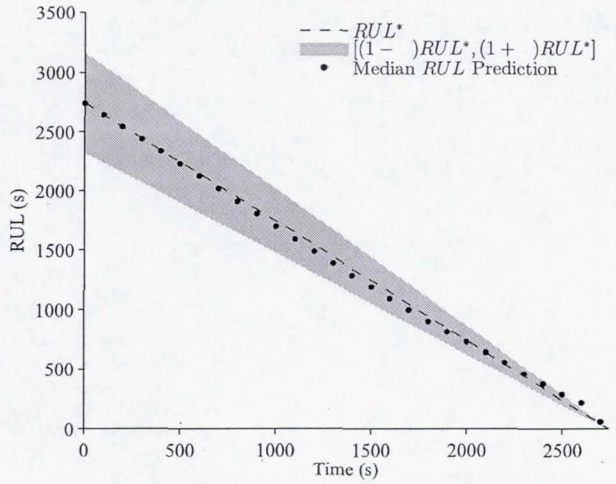

Figure 5. Prediction performance assuming known future input trajectory and no process noise with $\alpha = 0.15$.

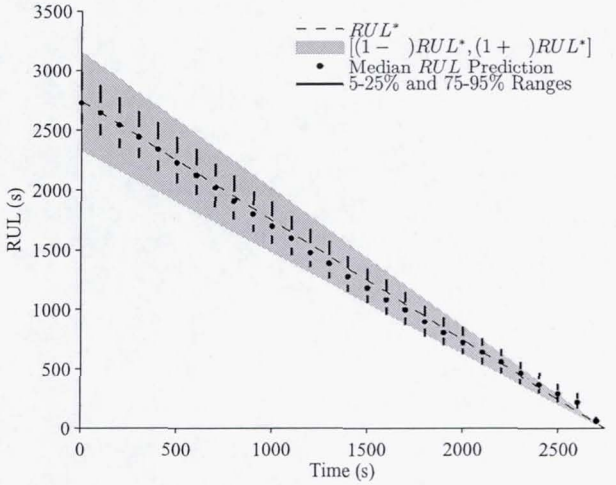

Figure 6. Prediction performance assuming known future input trajectory and 100 process noise samples with $\alpha = 0.15$.

just a constant current of 2.25 A, which is the average current drawn during the actual experiment (shown as the dashed line in Fig. 4). Here, we assume there is no process noise. The results are shown in Fig. 7. Clearly, assuming the average current results in a significant performance degradation. Around 1000 s, the average current until discharge is lower than hypothesized, so RUL is underestimated. Around 2500 s, the average current until discharge is higher than hypothesized, so RUL is overestimated. It is apparent that the RUL prediction is very sensitive to the uncertainty in the future inputs. In this case, the predictions are much more sensitive to input uncertainty than model uncertainty, since the model itself is more sensitive to changes in input than to the added process noise.

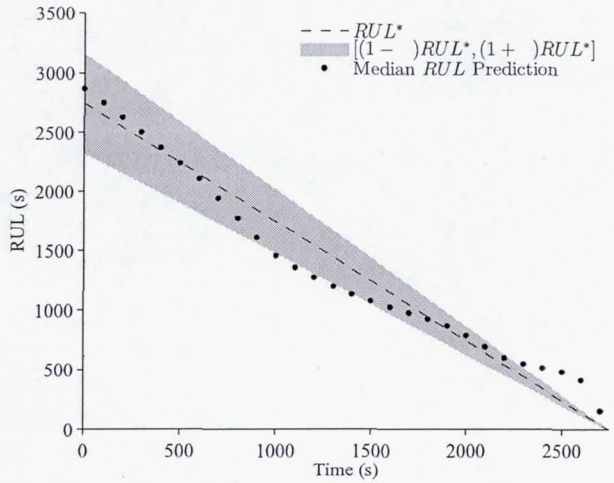

Figure 7. Prediction performance assuming average future input trajectory and no process noise with $\alpha = 0.15$.

Instead of assuming a single possible future input trajectory, we now consider multiple trajectories. At each prediction step, we assume the future current is drawn from a uniform distribution between 1 and 4 A and remains constant for the remainder of the discharge. Looking at Fig. 4, the current does not remain constant, but serves as a reasonable assumption for prediction purposes, i.e., rather than assuming variable discharge currents we assume constant discharge currents within the range of possible currents, which is much easier to sample from and still captures the best- and worst-case inputs. Fig. 8 shows the results using 10 samples and Fig. 9 shows the results using 100 samples. The uncertainty in the RUL predictions now is much more accurately represented. Clearly, the more samples used, the smoother the predictions and the better the description of the uncertainty. It is also clear here that the uncertainty in the future inputs causes about an order of magnitude more spread in the RUL predictions than the uncertainty associated with the process noise, comparing the figures (specifically, relative median absolute deviation averaged 6.2% in Fig. 6 and 27.9% in Fig. 8). Therefore, it would be acceptable to drop the process noise, which would save also on the required amount of computation.

Further improvements in computation can be achieved by using the unscented transform to select input trajectories. In this case, the future input trajectories are parameterized by a single number, representing the future current draw, taken from a uniform distribution. Applying the UT to this case yields only three future input trajectories that need to be simulated. There is no guidance to choosing a value for κ when the distribution is uniform, so we use the suggested value of κ for when the distribution is Gaussian, which, for a one-dimensional input

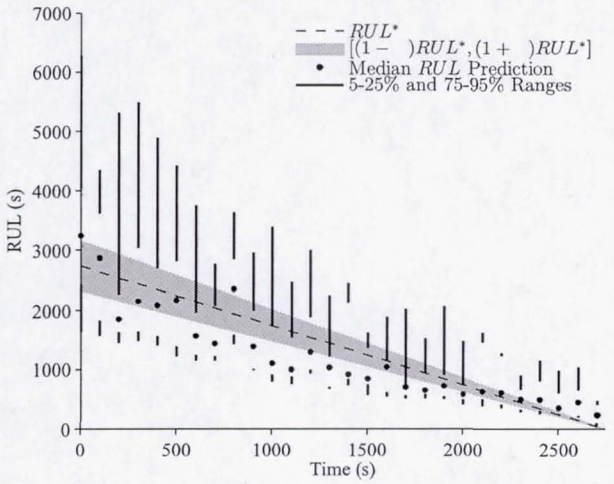

Figure 8. Prediction performance with 10 future input trajectories drawn from a uniform distribution and assuming no process noise with $\alpha = 0.15$.

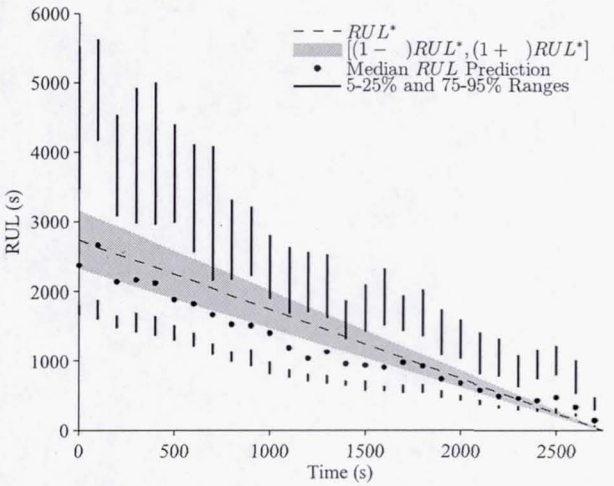

Figure 9. Prediction performance with 100 future input trajectories drawn from a uniform distribution and assuming no process noise with $\alpha = 0.15$.

space, is $\kappa = 2$.[4] With $\kappa = 2$, the UT happens to choose the three points as the mean of the distribution and its two endpoints, thus naturally capturing the input bounds. The results are shown in Fig. 10. Comparing to the case where 100 random input trajectories were generated, using the UT we are able to capture approximately the same distribution with only a fraction of the computational effort. The figure shows the results from the three sigma points directly, but the distribution can be reconstructed from this minimal set of samples. We will show in the following subsection that the UT is able to do this accurately. In this case the UT provides both the

[4] A smaller value of κ will bring the sigma points closer together, and a larger value spreads them out.

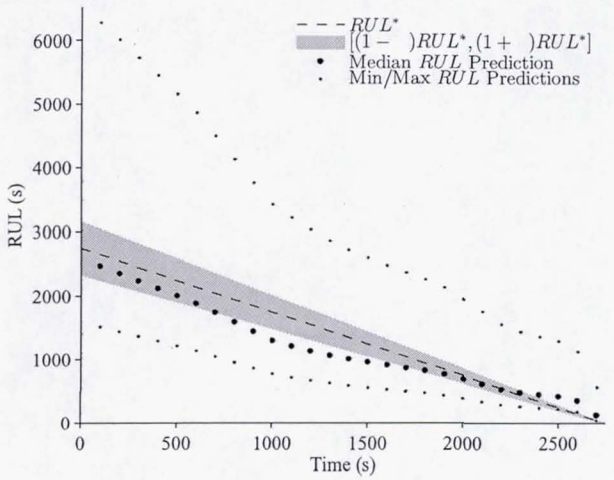

Figure 10. Prediction performance with future input trajectories drawn from a uniform distribution using the unscented transform and assuming no process noise with $\alpha = 0.15$.

RUL distribution and its bounds deterministically. Note that for a bounded distribution one may always choose the endpoints to determine the RUL bounds, however, in this case the UT does this automatically with the added benefit of being able to reconstruct the RUL distribution from those two points and the mean.

4.3. Simulation Results

For a more careful analysis, we ran a set of comprehensive simulation experiments. In these experiments, by using a simulation model, we eliminate several sources of uncertainty: (*i*) the state at the time of prediction is assumed to be known exactly, (*ii*) the system model is known exactly, (*iii*) the process noise distribution is known exactly, and (*iv*) the future input trajectory distribution is known exactly. This focuses the resulting uncertainty to only that associated with the process noise and the future input trajectories. In each experiment, the true input trajectory is sampled from the known distribution, while the algorithm knows only the distribution.

To analyze prognostics performance we use the relative accuracy (RA) metric to characterize the accuracy (Saxena, Celaya, Saha, Saha, & Goebel, 2010). For RA we use the median as the measure of central tendency since the RUL distributions are skewed. For spread, we use relative median absolute deviation (RMAD). For each experiment we perform 100 iterations, and average the results over these iterations. We compute also a computation time metric T_{cpu} which is computed as the fraction of computation time taken for a prediction t_{cpu} over the true RUL, RUL^*, i.e, $T_{cpu} = t_{cpu}/RUL^*$.

Results with process noise but without input uncertainty are shown in Table 2. In this case process noise has little effect

N_v	RA_{RUL}	$RMAD_{RUL}$	T_{cpu}
0	99.42	0.00	5.91×10^{-4}
10	99.28	1.34	6.75×10^{-4}
50	99.15	1.47	6.97×10^{-4}
100	99.02	1.39	7.34×10^{-4}

Table 2. Prognostics Performance with different values of N_v and no input uncertainty.

on accuracy, so even when ignoring process noise (indicated by $N_v = 0$ in the table) accuracy is pretty good, although the RMAD is 0 so the true spread is being underrepresented by the RUL prediction. Even with only 10 samples the prediction spread is quite close to the case with 100 samples (with a difference of only 0.26 in RA and 0.05 in RMAD), so only having a about 10 samples is acceptable for this level of process noise. As process noise increases, more samples will be needed to properly cover that space.

We now focus on the effect of the uncertainty in future inputs, so eliminate process noise from the simulation. The actual input current is drawn from a uniform distribution between 0.75 and 2.00 A. If the prediction algorithm always selects the mean, 1.375 A, as the future input current, then the relative accuracy of the first prediction point varies from about 52-97%. RA is high when the actual current is close to the mean but low when far from the mean. Because no uncertainty is taken into account, when the actual current is far from the mean not only will the accuracy be low, but the predictions will be of high precision and therefore be presented as predictions with high confidence, which would be incorrect and lead to poor decision-making.

If we instead sample randomly from the input distribution, the relative accuracy on average does not really change from when assuming only the mean input. This is because RA is computed based on the median RUL prediction, and when the input distribution is sampled enough, then we will get the correct mean of that distribution, which, because the distribution is symmetric, will correspond to the median RUL prediction. The key difference, however, is that now the RUL spread is more accurately represented by the predicted RUL distribution. The true RMAD is around 27%. Table 3 shows the prediction performance for different values of N_u. RA is about the same and the prediction spread, as computed by RMAD, approaches the true spread with around 25 samples (with about 7 times the computation as when predicting with only one sample). Using the UT (in which only 3 samples are needed), the performance is similar with only about 2.6 times as much computation needed compared to using only a single sample. Using $\kappa = 2$ for the UT, it chooses the mean and the endpoints of the assumed uniform distribution, so naturally provides the median and best- and worst-case RULs. Using the weights of the sigma points we can reconstruct the distribution it represents, which is not possible when just choosing these values in an ad hoc manner.

N_u	RA_{RUL}	$RMAD_{RUL}$	T_{cpu}
1	62.98	0.00	6.31×10^{-4}
10	74.57	25.96	2.36×10^{-3}
25	75.90	27.07	5.39×10^{-3}
50	78.54	27.50	9.61×10^{-3}
75	74.31	27.55	1.48×10^{-2}
100	76.71	27.61	2.08×10^{-2}
UT	76.77	26.72	1.61×10^{-3}

Table 3. Prognostics Performance with different values of N_u and no process noise.

5. CONCLUSIONS

In this paper, we analyzed the sources of uncertainty in prognostics and developed a general model-based prediction algorithm that incorporates this uncertainty to provide EOL and RUL results that correctly capture the true uncertainty in EOL and RUL. We also introduced the use of the unscented transform for efficiently sampling from the space of possible future input trajectories, which can achieve the same results as random sampling but at a fraction of the computational effort (see Table 3).

We applied the approach to both real and simulated lithium-ion battery data, where end-of-discharge was predicted. Here it was demonstrated that it is important to realize that for any source of uncertainty that is ignored in the prediction, even though in one particular case the result may be accurate, the actual uncertainty is underrepresented and so is not capturing the right information needed for decision-making. If the uncertainty is accounted for, on average the accuracy will be the same as if just average performance is assumed, but the spread will be correctly represented and the actual system behavior will fall within the predicted spread. Since the actual path the system will take is uncertain, it is best practice to capture the uncertainty as accurately as possible. It is important also not to overestimate the uncertainty. For example, when defining the space of possible future input trajectories it is important to constrain these trajectories as much as possible.

In this paper, we have shown that it is possible, by using the unscented transform, to estimate the uncertainty in predictions in a deterministic manner and with reduced computational burden while still keeping the important statistical information intact. Furthermore, since the uncertainty in future inputs happens to be uniformly distributed in our example, the UT method also determines the bounds of the uncertainty, which can be extremely valuable information in making decisions based on prognostic estimates in order to contain the risk. Minimizing the computational cost maintains the real-time application potential of the algorithm without compromising on the quality of prediction. This allows us to move a step closer towards adapting prediction algorithms, which are generally inherently stochastic, to meet the needs of current certification procedures and protocols that require determin-

istic system outputs. This also lays the foundation to start investigating other methods that can be used for generalized situations that do not assume a specific distribution type for future input uncertainty.

In the future, we will further investigate these issues on other systems and with additional experimental data. It is also important to investigate more closely the applicability of the UT for sampling the input trajectory space in a variety of practical applications.

ACKNOWLEDGMENT

Funding for this work was provided by the NASA Systemwide Safety and Assurance Technologies (SSAT) Project.

REFERENCES

Arulampalam, M. S., Maskell, S., Gordon, N., & Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, *50*(2), 174–188.

Barsali, S., & Ceraolo, M. (2002, March). Dynamical models of lead-acid batteries: Implementation issues. *IEEE Transactions on Energy Conversion*, *17*(1), 16–23.

Ceraolo, M. (2000, November). New dynamical models of lead-acid batteries. *IEEE Transactions on Power Systems*, *15*(4), 1184–1190.

Chen, M., & Rincon-Mora, G. A. (2006, June). Accurate electrical battery model capable of predicting runtime and I-V performance. *IEEE Transactions on Energy Conversion*, *21*(2), 504 - 511.

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

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

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

Edwards, D., Orchard, M. E., Tang, L., Goebel, K., & Vachtsevanos, G. (2010, September). Impact in input uncertainty on failure prognostic algorithms: Extending the remaining useful life of nonlinear systems. In *Annual conference of the prognostics and health management society.*

Julier, S. J. (1998, April). A skewed approach to filtering. In *Proc. aerosense: 12th int. symp. aerospace/defense sensing, simulation and controls* (Vol. 3373, p. 54-65).

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

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

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

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

Saha, B., Quach, C. C., & Goebel, K. (2012, March). Optimizing battery life for electric UAVs using a Bayesian framework. In *Proceedings of the 2012 ieee aerospace conference.*

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

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