Improving Computational Efficiency of Prognostics Algorithms in Resource-Constrained Settings

Katelyn J. Jarvis
NASA Ames Research Center
Moffett Field, CA 94035
katelyn.j.jarvis@nasa.gov

Christopher Teubert
NASA Ames Research Center
Moffett Field, CA 94035
christopher.a.teubert@nasa.gov

Wendy A. Okolo
NASA Ames Research Center
Moffett Field, CA 94035
wendy.a.okolo@nasa.gov

Chetan S. Kulkarni
KBR, Inc.
NASA Ames Research Center
Moffett Field, CA 94035
chetan.s.kulkarni@nasa.gov

***Abstract*—The field of prognostics and health management provides quantitative methods for monitoring and predicting the health of physical systems. Prognostic algorithms are useful in that they can be employed to assess the current state of a system, propagate the system state throughout time, and predict potential anomalies or failures that may occur. However, effective prognosis can be challenging to achieve in resource-constrained settings due to computational limitations and high computational latency, leading to obsolete predictions. Thus, computationally efficient and accurate algorithms are necessary for some prognostics applications. In this work, we implement three new algorithmic approaches to prediction (sampling methods, variable prediction time step, variable prediction sample size) with the goal of improving computational efficiency while minimizing decrease in model accuracy. To quantitatively analyze our results, we examine a use-case of degradation of a Lithium-ion battery. Notably, through this work it was found that none of the sampling approaches had a significant impact on computational efficiency or model accuracy in predicting EOD of the battery. However, our results show that prediction accuracy is highly dependent on the time step used, and that an appropriate time step can optimize both model accuracy and simulation efficiency. Finally, implementing a variable sample size also affected prediction, and our results show that tuning both the magnitude and timing of the sample size adjustment in an application-specific manner may prove useful in some applications. Taken together, our findings highlight the challenge of performing prognostics in resource-constrained settings, and illustrate the potential of developing new prediction algorithms to improve computational efficiency of prognosis.**

***Index Terms*—Prognostics, Health Management, Systems Health Monitoring, Prediction, Failure Analysis, Bayes Methods, Belief Networks, Probability, Reliability, Maintenance Engineering**

I. Introduction

The field of prognostics and health management involves the development of quantitative tools to monitor, predict, and proactively manage the health of physical systems. In general, prognostics tools are utilized to monitor the current state of a system and to propagate that state forward in time, thus enabling the prediction of when a particular state or event of interest may occur. In this way, prognostics algorithms provide insight as to when system anomalies or failures may take place, allowing for potential mitigation before a failure event transpires.

Prognostics is particularly imperative in engineering and aerospace applications, where accurate and timely assessment of system health and performance is necessary to reduce risk and provide safety. Safety management becomes more critical as aerospace systems and vehicles become more autonomous. One such application that will benefit from "in-time" prognosis is the integration of unmanned air vehicles (UAV) in the national airspace (NAS). With this integration, air transportation will be revolutionized as the number and diversity of aircraft in the NAS increases. To maintain the current levels of safety while airspace complexity increases, airspace safety technologies must be adapted to meet the needs of the new NAS [1]–[3]. Such technologies will need to monitor the state of the system, assess current risks, predict future hazards, and mitigate potential issues. Additionally, these capabilities must span from component prognostics, such as UAV battery health, up to larger system of systems prognostics, such as predicting aircraft intersections in a sector of the airspace. Prognostics methodologies provide a promising framework for achieving these requirements.

However, while prognostics is useful for risk monitoring and mitigation in many engineering contexts, it is challenging to perform real-time prognosis in resource-constrained settings. In many applications, such as on-board a UAV or spacecraft, computational capabilities are limited. Thus, computationally expensive algorithms, like those used in prediction, become time-intensive, resulting in predictions that are obsolete by the time they are generated and communicated to a decision-maker. Additionally, scaling prediction algorithms to enable numerous predictions across multiple system components can be computationally costly, making it challenging to perform optimization and decision making across a variety of conditions (e.g. [4]). Thus, computationally efficient and accurate algorithms are necessary for the successful implementation of a prognostics framework.

Given the necessity of timely outcomes for successful prognosis in resource-constrained settings, the objective of this work is to improve the computational cost and efficiency of a prognostics framework while minimizing any decrease in overall model accuracy. To achieve this, three key components of the prediction step are explored in this work: sampling method, time step, and sample size. New algorithmic approaches are implemented for each component, and model accuracy and computational efficiency are evaluated using a variety of metrics. In the following sections, we introduce the general

prognostics framework and our algorithmic methodologies (Section II), describe the implementation of our techniques on a use-case for the prediction of UAV battery degradation (Section III), highlight prominent findings (Section IV), and suggest future work (Section V). Our analysis provides insight into potential improvements to computation time, and suggests algorithmic enhancements that make real-time prediction more feasible in resource-constrained settings.

II. Methods

A. Model-based Prognostics

This work utilizes a model-based approach to prognostics, as implemented in the Python Prognostics Models and Algorithms Packages [5], [6] and similar to that described in [7]. This approach is intended to be generic, capable of describing system behavior based on physical principles (i.e., physics-based), learning from data (i.e., data-based), or hybrid approaches (e.g., Physics-Informed Machine Learning).

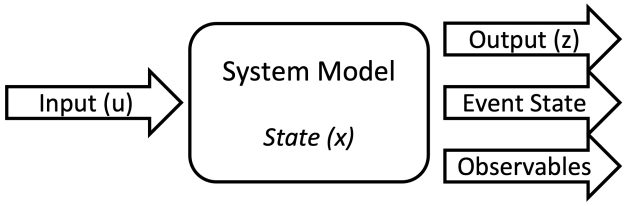

Fig. 1. Prognostics System Model

Under this approach, prognostics models describe how a system is expected to behave based on how it is used/loaded (i.e., input). The primary properties of a Prognostics Model under this framework are defined below, and described in Figure 1.

parameters (Θ) Parameters describe a system. They are used to specialize a generic model to the specific system of interest.

inputs (u) System usage (i.e., loading). For example, a battery is loaded by drawing a current from the battery.

states (x) Describes the state of the system at any point of time. For example, the state of an object in motion could be described as its position and velocity. States can be divided into model states (i.e., healthy states) and degradation states, which describe the degraded behavior of the system.

outputs (z) Measurable variables that are a function of state (x). For example, for a battery, outputs could be battery voltage and temperature.

events (E) Events to be predicted. For example, events for a battery could include End of Discharge (EOD), Capacity Degradation and Thermal Runaway conditions.

event states Each event has an event state that indicates progress towards the event occurring. Event states are defined as being between 0-1 where 1 corresponds to a state with no progress towards an event, and 0 corresponds to a state where the event has occurred. Event states are a function of system state (x).

observables Variables that describe the performance of the system, as a function of state (x). Observables are not directly measured, but can affect other systems dependent on the target system. For example, an actuator might include an observable of maximum achievable position. In this example, depending on the way a system degrades, some positions might not be achievable for a degraded system.

Model states transition as a function of time (t), parameters (Θ), state (x), and input/loading (u), as shown in Equation 1. Output is only a function of state (x) and parameters (Θ), as shown in Equation 2. Similarly, event states and observables are functions of state (x) and parameters (Θ).

$$x(t + dt) = f(t, x(t), u(t), dt, \Theta) \tag{1}$$

$$z(t) = g(x(t), \Theta) \tag{2}$$

In practice, it is impossible to have absolute knowledge of future states due to uncertainties in the system. There is uncertainty in the estimates of the present state, future inputs, models, and prediction methods [7]. This model-based prognostic approach incorporates this uncertainty in three forms: initial state uncertainty (x_0), process noise ($v(t)$) added at each application of the state transition equation (Equation 1), and measurement noise ($n(t)$) added at each application of the output equation (Equation 2).

Prognostics frameworks are frequently divided into two steps: state estimation and prediction, illustrated in Figure 2. State estimation is the process of determining the current system state (x), with some uncertainty, given the system parameters (Θ), system loading (u) and measured variables (z). There are various methods used for this, such as Kalman Filters and Particle Filters. These methods utilize the prognostics model, comparing measured parameters (z) with those predicted from the system output equation (Equation 2).

In the prediction step, the state estimate at the prediction time and system model are used together to estimate system degradation with time. This is most commonly done using a variant of the Monte Carlo method with the model state transition equation (Equation 1). Prediction is often computationally expensive, especially for sample-based approaches with strict precision requirements (which therefore require large number of samples).

With this framework, there are a number of results that can be predicted. The exact prediction results are selected based on the needs of the end-user. The most common prediction results are Time of Event (ToE) and Time to Event (TtE). Time of Event (see Equation 3) at a specific prediction time (t_P) is defined as the time when the event is expected to occur (with uncertainty), or equivalently, the time where the event state for that event is zero. Time to Event (see Equation 4) is defined as the time to ToE. In prognostics, ToE and TtE are often referred to as End of Life (EOL) and Remaining Useful Life (RUL), respectively. In subsequent sections of this paper, we use the terms EOL and RUL to refer to ToE and TtE.

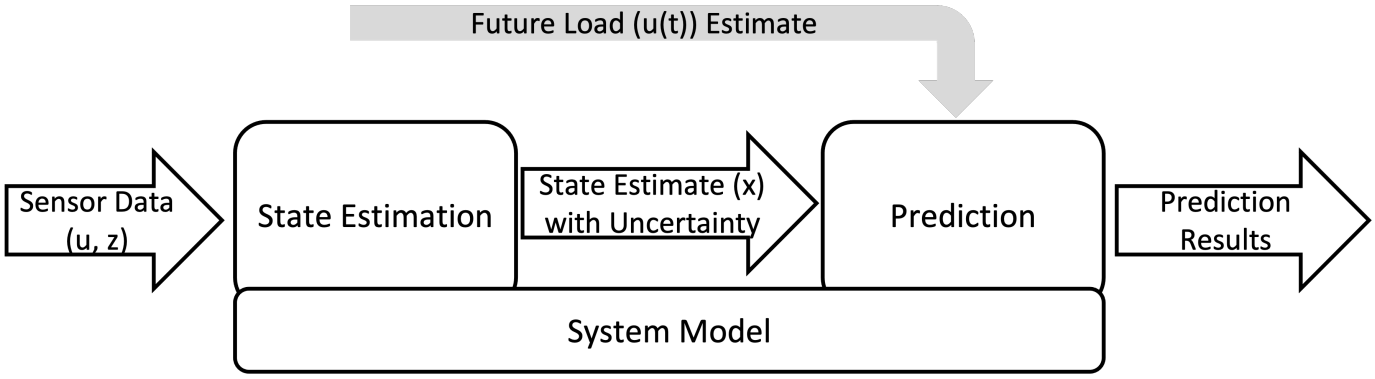

Fig. 2. Model-based Prognostics Process

$$ToE(t_P) \triangleq t \big| (event\_state(\Theta, x(t)) = 0) \tag{3}$$

$$TtE(t_P) \triangleq ToE(t_P) - t_P \tag{4}$$

Beyond these, results of prediction can also include event state, outputs, observables, and system states at different future times, including at ToE. For approaches that predict ToE with uncertainty, some users consider Probability of Success (PoS) or Probability of Failure (PoF). PoF is defined as the percentage of predictions that result in failure before the prognostics horizon ($PoS \triangleq 1 - PoF$).

B. Approaches to Reduce Prediction Time

To evaluate and increase computational efficiency of prediction, three new algorithmic approaches are implemented within the prognostics framework, including 1) distinct sampling methods, 2) variable time step in prediction, and 3) variable prediction sample size. In this section, we discuss the details involved in the implementation of each algorithm.

The first algorithmic approach focuses on the method in which samples are generated from the current state estimate in preparation for prediction (see Section IV-A). In this case, the current state estimate is in the form of a normal distribution, resulting from the use of an Unscented Kalman Filter (UKF) in state estimation (see Section II-A). The goal of this first algorithmic approach is to reduce the number of samples required to effectively represent this distribution.

A standard Monte Carlo (MC) random sampling method is common in prognostics [7]–[9], and therefore we use MC sampling as a baseline. To improve upon this random sampling, we additionally implement Quasi-Monte Carlo (QMC) and Latin Hypercube Sampling (LHS). Both methods are advantageous over a standard MC because they choose samples in a more ordered, quasi-random manner [10]. In QMC, for example, sample points are chosen from quasi-random, or low-discrepancy, sequences in order to maximize the spread of points within a particular interval to generate a more even distribution [10]. In our implementation of QMC sampling, we use Halton sequences [10] to create a uniformly distributed low-discrepancy sequence that is then transformed to be normally distributed and directly implemented with the current state information to generate representative samples. Using a similar approach, we can employ LHS to generate state samples by calling Matlab's `lhs` function with the mean and covariance of the normal state distribution.

Beyond the pre-processing step, changes to two key internal components of the sample-based prediction algorithms are being considered: time step and sample size. First, to explore the effect of prediction time step, we develop a variable time step prediction algorithm, in which the size of the time step by which we iterate a prediction forward is variable (see Section IV-B). This means that when the algorithm makes a prediction of remaining useful life, the time step of this forward prediction 1) can be fixed at any user-defined value, or 2) can be dynamic, changing to a larger or smaller time step based on the starting prediction time (t_P).

For a final approach to prediction, we develop an algorithm that includes a variable sample size within prediction (see Section IV-C). As discussed previously (see Section II-A), during each RUL prediction, a set of samples are drawn from the current state and propagated to the event horizon. The sample size affects model accuracy, with more samples producing better predictions, while also determining computational cost, with sample size proportional to simulation run-time. In this prediction algorithm, we explore approaches for intelligently varying sample size in such a way as to reduce computational costs while minimizing loss in the accuracy of prediction. In the algorithm, this sample size can be 1) defined by the user, or 2) can be dynamic, changing to a larger or smaller sample size at any point during the prediction. In the case of a "sample shed", a subset of samples are randomly chosen to be discarded at a particular time during prediction, and only the remaining smaller set of samples is iterated to event horizon. In the opposite case of a "sample gain", at a particular time during prediction, the current state is re-sampled using bootstrapping to generate new samples, increasing the number of samples that are propagated to end-of-life.

Note: We implement the gain or shed at an absolute time, t^*. In other words, a prediction begins at time t_P and is

propagated forward in time. When the current time reaches $t = t^*$, the sample gain or shed is performed. This means that, for starting prediction times less than t^*, (i.e. $t_P < t^*$), a change in sample size will occur when the system is propagated forward in time and passes $t = t^*$. However, for predictions that start at times beyond this absolute gain/shed time, (i.e. $t_P > t^*$), no sample gain or shed will occur, since presumably the shed or gain has occurred before prediction began. Instead, in this case the sample size is fixed throughout the simulation at the post-shed or post-gain size.

C. Metrics for Comparing Algorithms

With three algorithmic approaches to compare, we next develop a set of metrics to quantitatively analyze each method in the context of accurate and efficient prediction. To do so, we must first define a ground truth value for end-of-life, or "true EOL value", to which we will compare our model predictions (see Section II-A). Keeping in mind potential end users who are interested in safety assessment services that can inform decision-makers in-time, we develop the following metrics to quantify each algorithm's ability to capture the ground truth and provide worthwhile findings to the user.

At a basic level, users want to ensure that the prognostics framework they employ is accurately capturing the average EOL behavior. An accuracy measure of this form can be obtained from two of our metrics. We quantify the accuracy of the predicted EOL mean compared to the true EOL value by calculating relative accuracy (RA), which we define as the absolute value of the difference between the true EOL (EOL_T) value and the predicted mean EOL ($\overline{EOL}_P$), normalized to the true EOL value, and subtracted from 1 (i.e. perfect accuracy is 1, and poor accuracy approaches 0).

$$RA = 1 - \frac{|EOL_T - \overline{EOL}_P|}{EOL_T} \tag{5}$$

Additionally, we obtain insight into the overall accuracy of the predicted distribution in capturing the ground truth by calculating Total Error (TE). Total Error is defined as the sum of the squared difference between all predicted EOL values (EOL_{P_i}) and the true EOL value (EOL_T), normalized by the sample size (N). In addition to overall accuracy, many users are interested in the accuracy of prognostics results within a specific risk tolerance. To quantify this, we calculate specific percentile error calculations, defined as the sum of the squared difference between predicted EOL values that fall within a specific user-defined percentile and the true EOL, normalized by the sample size (N).

$$TE = \frac{1}{N}\sum(EOL_T - EOL_{P_i})^2 \tag{6}$$

Many stakeholders require precise predictions to complement desired accuracy. To quantify the dispersion of our results, we calculate the mean absolute deviation (MAD) of the predicted RUL values by taking the mean of the absolute value of the difference between each predicted RUL value (RUL_{P_i}) and the average of the predicted distribution ($\overline{RUL}_P$). This metric provides information on the spread of the predicted distribution.

$$MAD = \overline{|RUL_{P_i} - \overline{RUL}_P|} \tag{7}$$

Finally, one of the most common needs of stakeholders is a computationally efficient prognostics framework. To compute run-time for our algorithms, we employ Matlab functions to measure both CPU time and wall-clock time (i.e. Matlab's `cputime` and `tic/toc` functions, respectively).

With this collection of metrics, we are able to both quantify the ability of each algorithm to perform accurate and fast computation, as well as to discern differences between algorithms and draw conclusions regarding the most optimal approach.

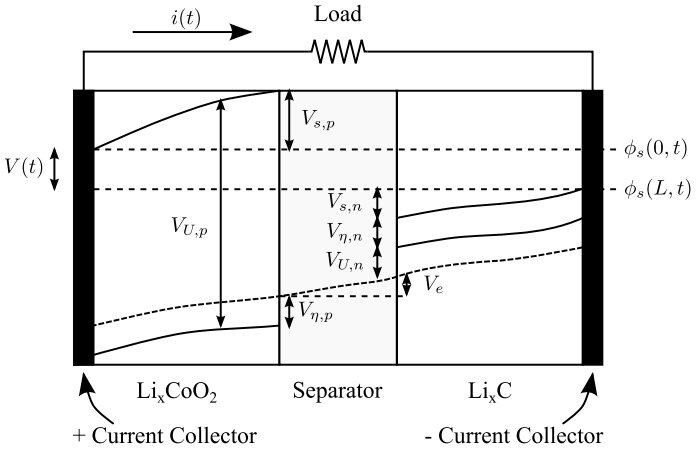

Fig. 3. Battery voltages [11]

III. Application: Lithium-ion Battery Degradation

In this work, two models of battery degradation are implemented to test and validate the proposed algorithms. The first is a physics-based electro-chemical battery model developed in [11], while the second is a generalized equivalent circuit based model [12].

A. Electro-chemical Battery Model

A battery is a collection of electrochemical cells that convert between chemical and electrical energy. Each cell consists of a positive and a negative electrode with electrolyte. In this paper, we focus on Li-ion cells. The electrolyte enables the diffusion of lithium ions (Li^+) between the positive and negative electrodes. The lithium ions insert or de-insert from the active material depending upon the electrode and whether the active process is charging or discharging [13].

This work implements an electrochemistry-based model developed for Li-ion cells, as described in [11]. Unlike other electrochemistry-based models that rely on complex partial differential equations that are unsuitable for online estimation and prediction algorithms, this model uses ordinary differential equations, and is fast enough for real-time use [11]. In this section, the model is briefly summarized and its key features are discussed.

TABLE I
STAKEHOLDER NEEDS AND CORRESPONDING METRICS

Stakeholder Needs	Metric	Equation
Accurately capture mean EOL behavior	Relative Accuracy (RA) of EOL Mean	$RA = 1 - \frac{\lvert EOL_T - \overline{EOL}_P \rvert}{EOL_T}$
	Total Error (TE)	$TE = \frac{1}{N_s} \sum_{i \in N} (EOL_T - EOL_{P_i})^2$
Accurately capture EOL in specific risk tolerance	Specific Percentile Error (SPE)	$SPE = \frac{1}{N} \sum_{i \in N_p} (EOL_T - EOL_{P_i})^2$
Precise prediction	Mean Absolute Deviation (MAD) of predicted RUL	$MAD = \overline{\lvert RUL_{P_i} - \overline{RUL}_P \rvert}$
Computational efficiency	CPU-Time and Wall-clock Time	`Matlab` functions

Table notations: EOL_T is true EOL, RUL_T is true RUL, EOL_{P_i} refers to all predicted EOL values, RUL_{P_i} refers to all predicted RUL values, $\overline{EOL}_P$ is predicted mean EOL, $\overline{RUL}_P$ is predicted mean RUL, N_s is the set of all samples, N_p is the set of samples in a user-defined percentile, and N is the total number of samples

The overall battery voltage, $V(t)$, consists of several electrochemical potentials (illustrated in Fig. 3). At the positive current collector is the equilibrium potential $V_{U,p}$. This voltage is then reduced by $V_{s,p}$, due to the solid-phase ohmic resistance, and $V_{\eta,p}$, the surface overpotential. The electrolyte ohmic resistance then causes another drop V_e. At the negative electrode, there is a drop $V_{\eta,n}$ due to the surface overpotential, and a drop $V_{s,n}$ due to the solid-phase resistance. The voltage drops again due to the equilibrium potential at the negative current collector $V_{U,n}$.

The state vector, input vector, and output vector of the EOD model are defined follows:

$$\mathbf{x}_{\text{EOD}}(t) = \begin{bmatrix} q_{s,p} & q_{b,p} & q_{b,n} & q_{s,n} & V_o' & V_{\eta,p}' & V_{\eta,n}' \end{bmatrix}^T \quad (8)$$

$$\mathbf{u}(t) = \begin{bmatrix} i_{app} \end{bmatrix} \quad (9)$$

$$\mathbf{y}(t) = \begin{bmatrix} V \end{bmatrix} \quad (10)$$

where $q_{i,j}$ represents the charge of the positive or negative electrode (for $j = p, n$, respectively) in the surface layer (for $i = s$) or the bulk (for $i = b$) volumes, and i_{app} is the applied electric current. Differential equations describe how charge moves through these volumes. Parameter values for a typical Li-ion cell are given in Daigle et al. [11].

An example discharge cycle is shown in Figure 4. The cell starts fully charged at 4.2 V and is discharged at 1 A. Around 8000 s, the voltage hits the lower voltage limit, 3.2 V, which defines EOD. At this point the load is removed and the cell voltage recovers back to the equilibrium potential. In this case, the capacity is computed as the discharge time (7630 s) times the applied current (1 A), yielding 2.12 Ah.

Note that capacity can only be measured consistently for a discharge cycle at reference conditions, since measured capacity is a nonlinear function of the load and environmental conditions. In this work, reference conditions are defined by a 1 A discharge at room temperature. Additionally, while it is important to consider battery aging, which results in a loss of capacity and an increase in internal resistance due to a variety of physical processes [14], this work only implements measurements from new batteries, and therefore modeling battery aging is beyond the scope of this work.

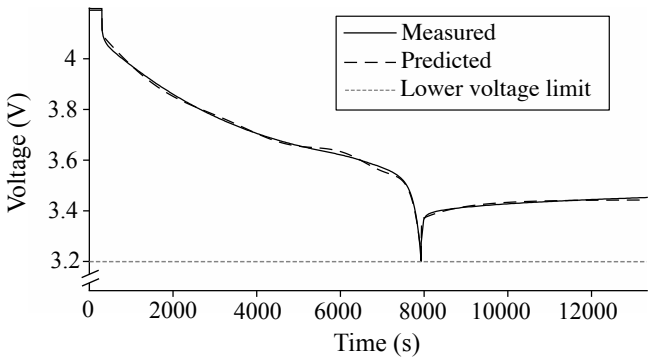

Fig. 4. Battery voltage during 1 A discharge cycle [11].

B. Equivalent Circuit based Model

Previous work on batteries has implemented the equivalent circuit diagram based battery model presented in [15], [16], and has used Unscented Kalman filtering (UKF) to update battery state estimates based on observations of current and voltage at the battery output terminals. Figure 5 shows the equivalent circuit battery model implemented here to represent

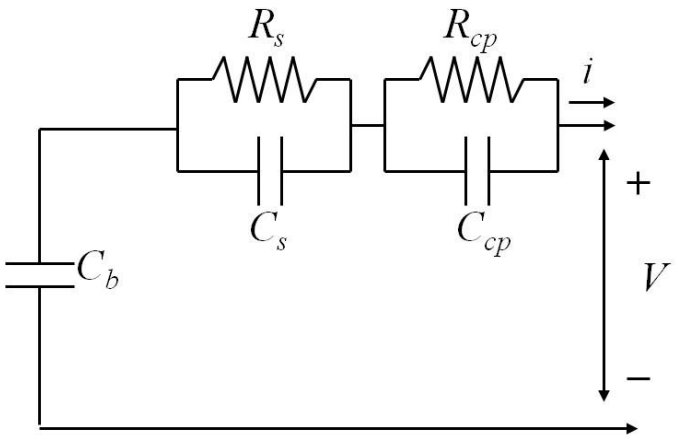

Fig. 5. Equivalent Circuit Based Model [16]

battery output voltage dynamics as a function of the battery current control input. This battery model contains six electrical components that are tuned to recreate the observed current-voltage dynamics of a battery pack used in the flight tests. The bulk of the battery charge is assumed to be stored in the capacitor, C_b. The (R_s, C_s) and (R_{cp}, C_{cp}) circuit element pairs are used to simulate standard battery phenomenon, such as internal resistance drops and hysteresis effects [16].

To better match such standard battery phenomenon, additional pairs of series connected RC parallel circuits are added to the model. The R_s, C_s pair are added for the internal resistance drop and the R_{cp}, C_{cp} pair are added for the concentration polarization effect. The correspondence of these RC circuits to actual battery chemical phenomena is notional for modeling purposes. The State of Charge (SOC) is a battery charge estimate of the bulk charge. The battery input-output voltage dynamics will change as a function of this bulk charge estimate. The ratio of a battery's charge at a given instant to its maximum charge storage capacity is typically referred to as the State of Charge (SOC). Battery SOC is defined here as:

$$SOC = 1 - \frac{q_{max} - q_b}{C_{max}} \tag{11}$$

where q_b represents the charge stored in capacitor C_b, q_{max} is the maximum charge that the battery can hold, and C_{max} is the maximum charge that can be drawn from the battery during operation at the lowest threshold. C_{max} will always be less than q_{max}, due to the ongoing internal electrochemical side-reactions that make some portion of a battery's charge carriers unavailable to be used during the discharge cycle. The UKF receives the measured battery current and voltage as input, approximates the mean and covariance of the state variable, and propagates selected "sigma points" that when transformed by the non-linearity of the model are used to estimate the output mean and covariance of the estimated battery terminal voltage. Implementation details for the equivalent circuit model and UKF state estimation are discussed in [17].

IV. Results

To test the ability of the developed approaches to produce accurate and computationally efficient predictions, each approach is implemented within a prognostics framework aimed at predicting battery discharge voltage (see Sections II-A and III). Through implementation of our user-driven metrics, we analyze each algorithm's ability to accurately capture the battery system, and then extend our analysis to quantitatively compare the different approaches. Our findings illustrate that prediction quality is largely independent of sampling method, but that prediction algorithms can be optimized by appropriately choosing both time step and sample size. These results have implications for developing robust and efficient prediction algorithms.

In the following sections, we implement the electrochemical battery model to illustrate our findings (though see application of the equivalent circuit model in Section IV-B). However, implementation of the equivalent circuit battery model produced similar results. Additionally, note that we adjust our notation for "end-of-life" (EOL) to more specifically be "end-of-discharge" (EOD) to emphasize that in our battery use-case, the specific event of interest is battery discharge.

A. Sampling methods have minimal effect on prediction results

First, three distinct methods of sampling the current state at the start of a prediction are implemented. In particular, we compare Quasi-Monte Carlo (QMC) and Latin Hypercube (LHS) techniques to a standard Monte Carlo (MC) sampling method. While QMC and LHS sampling techniques will approach standard MC for sufficiently large sample sizes (see Fig. 6A, 100,000 samples), for smaller sample sizes (like those feasible in the context of resource-constrained prognostics), both QMC and LHS produce smoother distributions that more accurately capture the state estimate mean (see Fig. 6A, 50 samples). The superior distributions generated by QMC and LHS suggest that these sampling methods may require fewer samples to achieve comparable prediction accuracy, thus decreasing overall computation time by reducing the necessary sample size.

Comparing sampling methods, however, we find that the differences in initial sampling distributions do not translate to significant differences in prediction results. In particular, relative accuracy, mean absolute deviation, total error, and 10^{th} percentile error as functions of sample size are comparable for the three sampling methods (Fig. 6B), as well as runtime (Fig. 6C). Given the similarity across the metrics, we are unable to implement a sampling method to maintain prediction accuracy with a smaller sample size, and we conclude that prediction results are largely independent of MC, QMC, and LHS sampling techniques.

B. Prediction time step has a significant impact on computational efficiency and model accuracy

Given the model's insensitivity to sampling method, we next explored the effect of the prediction time step size on model accuracy and efficiency. To do so, we develop a variable time

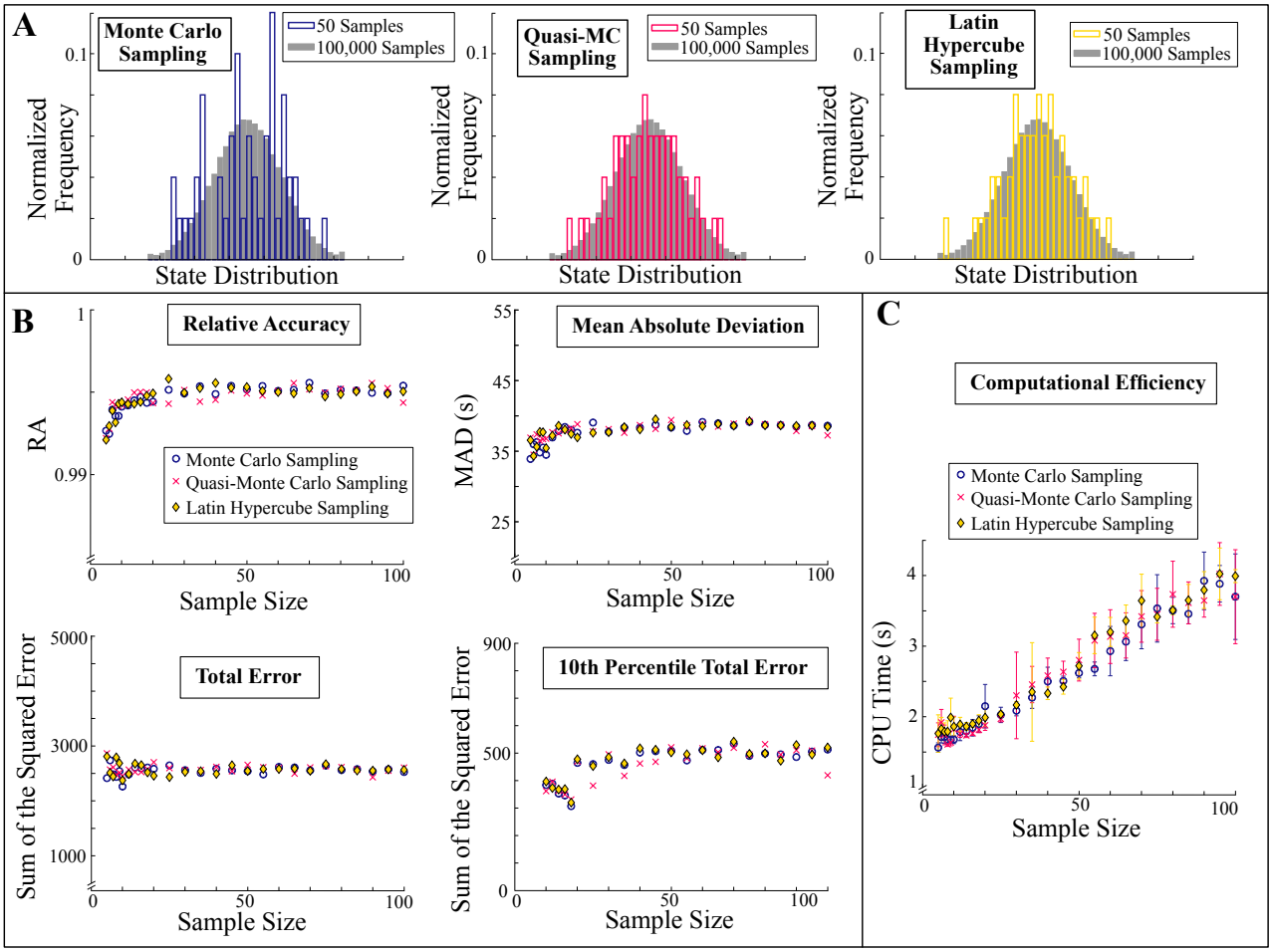

Fig. 6. Effect of sampling method on prediction results. **A)** Distributions of an arbitrary state using MC, Quasi-MC, and LHS sampling with 50 (colored) and 100,000 (gray) samples. **B)** Metrics (RA, MAD, TE, and 10th percentile error) measured as a function of sample size for MC, Quasi-MC, and LHS sampling techniques illustrate the similarities among methods. **C)** CPU time (average $\pm$ standard deviation) of prediction run-time as a function of sample size for each sampling technique. For **B** and **C**, results are an average of 100 simulations.

step algorithm, in which the time step incremented in prediction can either be fixed or dynamic (see Section II). Such an algorithm has two potential improvements upon computational efficiency. First, this prediction algorithm allows us to explore the overall optimal prediction time step, or Δt, value. The size of the steps taken to propagate forward in time will have a direct impact on computational efficiency and model accuracy, with large time steps resulting in faster computation but with risk of decreased accuracy or model instability. Our prediction algorithm allows for exploration into optimizing Δt.

Second, this algorithm allows for a variable time step, with Δt changing based on the time at which prediction is made, t_P (see Section II). Intuitively, it is plausible that tuning the time step to be bigger with less accuracy, or smaller with more accuracy, at different points during a simulation may optimize both computational efficiency and prediction accuracy to the specific needs of the stakeholder. For example, a user may need precise knowledge of near-term events, i.e. to know if a failure event will occur in the next 2-3 minutes of run-time. In this case, it may be beneficial to use small time steps for the start of a prediction, to get very accurate results in the near term, and larger time steps past the 2-3 minute mark in prediction, where accuracy is less important and we can improve upon computational efficiency with a larger Δt. By tuning the time step throughout the simulation with the needs of the stakeholder in mind, prediction may be improved and optimized.

To test these conjectures in the context of battery degradation, we first employ our model to predict a set of samples to end-of-discharge at a collection of fixed Δt values. As expected, we find that for sufficiently small Δt values, the resulting relative accuracy is near 1 (optimal) and the total error is nearly 0 (Fig. 7A, panel 1 and 3). As Δt values increase, these metrics stay relatively constant at their respective values, until a critical Δt value is reached, and as Δt increases further, we observe a significant decrease in relative accuracy

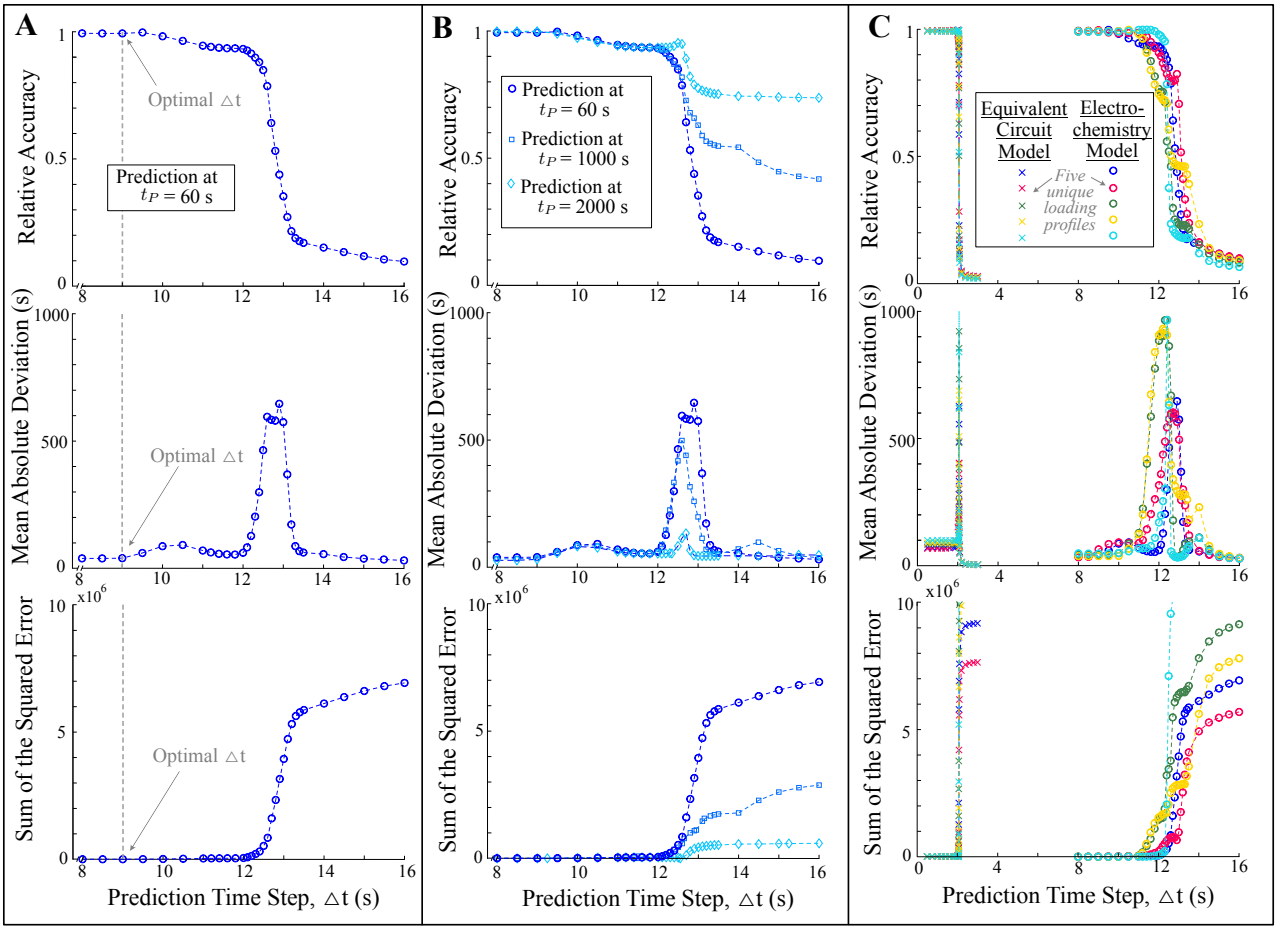

Fig. 7. Effect of time step on prediction results. **A)** RA, MAD, and TE as a function of Δt for battery data predicted from $t_P = 60$ s (data comparable to that used in Fig. 6 and Fig. 8). **B)** Same as panel **A**, with addition of predictions made closer to EOD, starting at $t_P = 1000$ s and $t_P = 2000$ s. **C)** Same as in panel **A**, but including results from 5 other data sets with unique loading profiles (distinguished by color), and using both the electrochemistry battery model and the equivalent circuit battery model (distinguished by symbol). All panels reflect an average of 100 simulations.

and an increase in error (Fig. 7A, panel 1 and 3). Notably, the transition between these two regimes occurs over a small Δt interval.

To explore optimal Δt further, we considered the effect of time step on mean absolute deviation to see how the spread of predicted end-of-discharge values changes with Δt. For smaller Δt values that result in acceptable accuracy, we observe small MAD values indicating minimal dispersion around the predicted EOD mean (Fig. 7A, panel 2). As Δt is increased and accuracy begins to decline, our findings display an increase in MAD. This increase in dispersion occurs as Δt increases because samples are beginning to fail with the large Δt, resulting in a larger spread of predicted EOD in comparison to the average predicted EOD value. As Δt increases further, the accuracy of the model reaches a minimum. Mean absolute deviation, however, returns to an acceptable level, because even though all samples have failed, they have done so uniformly, resulting in minimal dispersion and thus a low MAD value (Fig. 7A, panel 2).

Taken together, the effect of prediction time step on relative accuracy, total error, and mean absolute deviation illustrate that the most optimal Δt for prediction is slightly below the transition from optimal to sub-optimal accuracy. Such a Δt will be small enough to avoid any samples failing early and a corresponding increase in dispersion, while also being large enough to optimize run-time. Specifically, our results show that an optimal Δt (e.g. $\Delta t \approx 9$ for our selected data) maintains accuracy (RA = 0.99 ± 0.002) but is more computationally efficient than smaller Δt values (i.e. compare $\Delta t = 1$ and $\Delta t = 9$ in Table II and Fig. 7A), while also achieving similar computational efficiency but significantly improved model accuracy compared to larger Δt (i.e. compare $\Delta t = 9$ and $\Delta t = 14$ in Table II and Fig. 7A).

To further understand the effect of optimal prediction time step, we ran comparable simulations of battery degradation using two different models (as discussed in Section III), and under a variety of loading conditions. In each simulation, we found the same qualitative behavior, a small Δt resulting in good accuracy, and a steep transition to larger Δt values that result in poor model accuracy (Fig. 7C). However, the critical

TABLE II
MEAN ± STANDARD DEVIATION OF METRICS FOR RESULTS IN FIGURE 7A.

Δt (s)	RA	Total Error	CPU Time (s)
$\Delta t = 1$	1.00 ± 0.002	2587.45 ± 351.23	4.01 ± 0.330
$\Delta t = 9$	0.99 ± 0.002	2947.86 ± 446.583	1.38 ± 0.019
$\Delta t = 14$	0.15 ± 0.003	6.12 x 10^6 ± 4.09 x 10^4	1.12 ± 0.025

Δt where this transition occurred was dependent on the battery model implemented, with the less accurate equivalent circuit model requiring a smaller Δt, and the more accurate electrochemistry model allowing for a much larger Δt (Fig. 7C).

As a final examination of optimal Δt, the dynamic feature of our variable time step prediction algorithm is implemented to measure model accuracy as a function of Δt at various times of prediction (t_P). The objective of these simulations was to understand if tuning Δt based on proximity of prediction time and true EOD would improve accuracy or efficiency. It is observed that the relationship between our metrics and Δt is independent of starting prediction time, t_P (Fig. 7B). As t_P is varied, RA, MAD, and TE are nearly identical for a given Δt, until Δt increases enough for failure to occur, in which case the final RA and TE values are dependent on how close the prediction time is to the final end-of-discharge time (Fig. 7B). These findings suggest that a dynamic Δt value dependent on prediction time (or event state, which is correlated to prediction time) is not an effective means to improving prediction.

This exploration into prediction Δt has two key implications for prognostics algorithms. First, picking an appropriate prediction time step optimizes computational efficiency. Additionally, this optimal Δt value is, at least in part, influenced by the details of the model describing the physical system, but is not dependent on the proximity of the prediction time to event time. Thus, by selecting an optimal time step, prediction algorithms can be optimized to be both efficient and accurate.

C. Variable sample size affects prediction results in an application-dependent manner

As a final exploration into prediction efficiency, different methods are considered for strategically reducing or increasing the number of samples during specific segments of the prediction. By its nature, reducing the number of samples will decrease computation time, but this will also result in a loss of information and therefore a decrease in prediction accuracy. The focus of this exploration is on methods minimizing the negative impact on model accuracy from reducing samples in prediction. This is achieved by developing a variable sample size prediction algorithm, which includes features for both a sample shedding or a sample gain at any point during prediction (see Section II). Intuitively, tuning sample size throughout prediction may allow for high accuracy in specific user-defined intervals (with a large sample size) where there might be more complicated effects (and therefore changes in the shape of the distribution), while reducing computational cost when a less accurate prediction is sufficient (with a smaller sample size).

To test this algorithmic approach, we ran a collection of predictions with the battery model, performed at consecutive times throughout a battery discharge. As a baseline, we ran simulations with a standard prediction algorithm (i.e. implementation of MC sampling for fixed time step and sample size) for a small sample size (10 samples) and a larger sample size (100 samples). As expected, there is an increase in relative accuracy along with a significant increase in run-time with the addition of extra samples (compare gray and black circles in Fig. 8A, and gray and black bars in Fig. 8B). This validates the potential impact of a variable sample size. If a sample gain or shed method is successful, when compared to a constant sample size prediction of equivalent RA, it should take less time to perform the prediction (i.e., be more computationally efficient). To investigate this, we implement our variable sample size prediction algorithm for 1) sample gain: a starting sample size of 10, increased to 100 at $t^* = 1500$ s, and 2) sample shed: a starting sample size of 100, decreased to 10 at $t^* = 1500$ s, and measure their impact on accuracy and run-time.

First, implementing a sample gain approach, we observe a favorable decrease in run-time compared to the larger sample size case (Fig. 8B). Analyzing the predictions made with this algorithm, however, we find an accuracy similar to the standard predictor with the smaller sample size, until a starting prediction time that is beyond the time index of sample gain (i.e. $t_P > t^* = 1500$ s), in which case the sample gain results are comparable to the standard predictor with the larger sample size (since predictions for $t_P > t^*$ have a sample size of 100, see Section II-B for explanation, Fig. 8A). These results show that there is no benefit to increasing sample size during prediction using our approach, as our sample gain method is insufficient to capture the system more accurately than the standard predictor with the smaller sample size. However, the findings do suggest a potential benefit of varying total sample size based on starting prediction time, t_P, in an application-specific manner. For example, suppose a UAV operator is monitoring battery health, and requires a lead time of $\sim$3 minutes [16], to safely land the vehicle if the battery degrades beyond some prior set threshold. In this case, for monitoring of battery health when EOD is far away (e.g. > 10 mins), less accuracy is sufficient, since no action is needed. Therefore, the prediction of battery health can be achieved with a small sample size, and thus greater computational efficiency. Then, as t_P increases towards the failure event, total sample size can be increased to improve accuracy of the predictions near EOD, so the controller has a more refined prediction and can safely land the UAV with enough time.

For the sample shed method, we again observe that while computation time is decreased (Fig. 8B), the shedding of samples also reduces the model accuracy to that of the smaller sample size (Fig. 8A). Our findings suggest that a simple sample shed method cannot preserve accuracy after samples

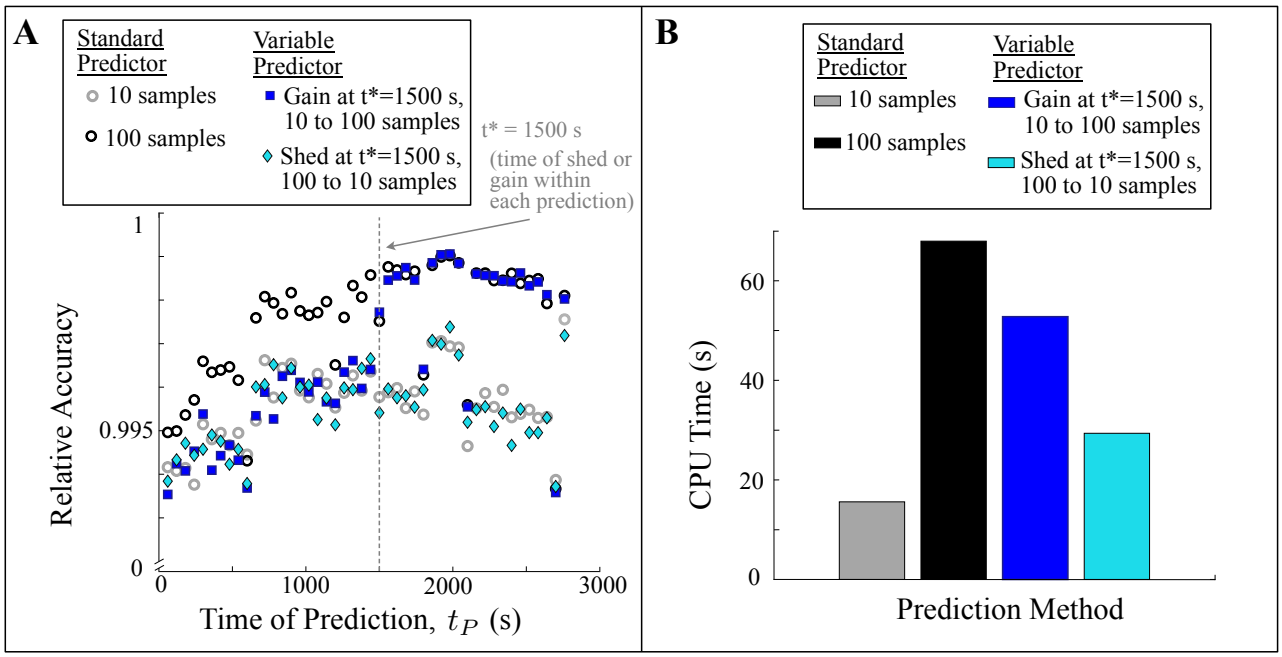

Fig. 8. Effect of sample size on prediction results. **A)** Relative accuracy as a function of time of prediction, t_P, for baseline simulations using a standard predictor with 10 samples (gray circles) and 100 samples (black circles) as well as a variable sample size prediction algorithm with a sample gain (dark blue squares) and a sample shed (light blue diamonds). Dashed line indicates the absolute time during each prediction when samples were gained or shed. **B)** CPU time for each prediction method in panel **A**.

are shed, but this technique may still be useful for users who require accuracy in the short-term, but are willing to sacrifice accuracy for efficiency past some time threshold. For example, for a UAV operator using this algorithm to inform flight decisions, if the battery is going to fully discharge in the next 5 minutes, the operator may want to know a precise time to inform a response. In the same scenario, if the battery is going to reach end of discharge around 60 minutes, the same operator may be indifferent to an EOD time of 55 minutes or one of 65 minutes; less accuracy is required. Thus, our findings suggest that a sample gain or shed method does improve computational efficiency, and may prove favorable for specific stakeholder needs.

V. Conclusions and Future Work

To improve computational efficiency while maintaining model accuracy of prognosis, we explored three key components of a prediction algorithm in the context of battery degradation. While the quantitative details of our findings are specific to this use-case, they also provide insight into potential improvements for a general prognostics framework, as well as future areas of study.

Most notably, our analysis highlights the importance of correctly defining the time step of prediction (Δt). Implementation of our variable time step prediction algorithm illustrates a steep transition between Δt values that result in acceptable model accuracy and those that are insufficient to capture system behavior. While the steepness of this curve was observed for both models we considered, the magnitude of the optimal Δt was model specific, suggesting that optimal time step is, at least in part, a function of the physical model used. In future work, it will be worthwhile to explore the specific data characteristics that may influence optimal Δt, as well as potential causes of variations in accuracy with different step sizes (e.g. proximity to event, load intensity, model accuracy, etc.). Such knowledge may allow for Δt to be optimally defined. With these variable time step findings, it is concluded that defining prediction time step in an application-specific manner will be crucial for successful and efficient prognostic estimates.

Additionally, our variable sample size findings further indicate the need to consider application-specific details. In exploring the potential of sample gain and shed algorithms, it was found that while a simple implementation of these methods was not broadly applicable to improve efficiency and maintain accuracy, they may prove advantageous depending on users needs. In future work, various iterations of sample gain and shed should be considered. For example, these algorithms may be useful in more contexts if samples are strategically shed or gained based on certain conditions. Future research could explore methods for strategic selection of which samples to keep or shed. Such a technique may preserve the ability to represent the distribution with fewer samples. Additionally, exploring variations and combinations of shedding and gaining samples with specific user objectives in mind may improve prediction. Beyond these sample-based considerations, future work should additionally explore varying the time (t^*) at which the sample gain or shed occurs. Taken together, our

battery degradation results suggest that tuning the specific details of prognostics algorithms in application-specific ways can enhance the ability of general prognostics algorithms to capture the system behavior in an efficient manner.

ACKNOWLEDGMENT

This work was supported by the System-Wide Safety (SWS) project under the Airspace Operations and Safety Program within the NASA Aeronautics Research Mission Directorate (ARMD).

REFERENCES

[1] S. Young, E. Ancel, A. Moore, E. Dill, C. Quach, J. Foster, K. Darafsheh, K. Smalling, S. Vazquez, E. Evans, W. Okolo, M. Corbetta, J. Ossenfort, J. Watkins, C. Kulkarni, and L. Spirkovska, "Architecture and information requirements to assess and predict flight safety risks during highly autonomous urban flight operations," 2020.

[2] K. K. Ellis, P. Krois, J. Koelling, L. J. Prinzel, M. Davies, and R. Mah, "A concept of operations (conops) of an in-time aviation safety management system (iasms) for advanced air mobility (aam)," in *AIAA Scitech 2021 Forum*, 2021, p. 1978.

[3] K. Ellis, P. Krois, M. D. Davies, and J. Koelling, "In-time system-wide safety assurance (issa) concept of operations," 2019.

[4] G. Sierra, M. Orchard, K. Goebel, and C. Kulkarni, "Battery health management for small-size rotary-wing electric unmanned aerial vehicles: An efficient approach for constrained computing platforms," *Reliability Engineering & System Safety*, vol. 182, pp. 166–178, 2019.

[5] C. Teubert, M. Corbetta, C. Kulkarni, and M. Daigle, "Prognostics models python package," Aug. 2021. [Online]. Available: https://github.com/nasa/prog_models

[6] ——, "Prognostics algorithm python package," Apr. 2021. [Online]. Available: https://github.com/nasa/prog_algs

[7] K. Goebel, M. J. Daigle, A. Saxena, I. Roychoudhury, S. Sankararaman, and J. R. Celaya, *Prognostics: The science of making predictions*, 2017.

[8] J. Guo, Z. Li, and M. Li, "A review on prognostics methods for engineering systems," *IEEE Transactions on Reliability*, vol. 69, no. 3, pp. 1110–1129, 2019.

[9] A. Bregon and M. J. Daigle, *Fundamentals of Prognostics*. Cham: Springer International Publishing, 2019, pp. 409–432. [Online]. Available: https://doi.org/10.1007/978-3-030-17728-7_17

[10] M. A. G. Dias. (1995) Quasi-monte carlo simulation. [Online]. Available: http://marcoagd.usuarios.rdc.puc-rio.br/quasi_mc.html

[11] M. Daigle and C. Kulkarni, "Electrochemistry-based battery modeling for prognostics," in *Annual Conference of the Prognostics and Health Management Society 2013*, Oct. 2013, pp. 249–261.

[12] E. Hogge, C. S. Kulkarni, S. Vazquez, K. M. Smalling, T. H. Strom, B. Hill, and C. Quach, "Flight tests of a remaining flying time prediction system for small electric aircraft in the presence of faults," 2017.

[13] *Development of a First Principles Equivalent Circuit Model for a Lithium Ion Battery*, 10 2012. [Online]. Available: https://doi.org/10.1115/DSCC2012-MOVIC2012-8607

[14] G. Ning, R. E. White, and B. N. Popov, "A generalized cycle life model of rechargeable li-ion batteries," *Electrochimica Acta*, vol. 51, no. 10, pp. 2012–2022, 2006.

[15] B. Bole, C. Teubert, Q. C. Chi, H. Edward, S. Vazquez, K. Goebel, and G. Vachtsevanos, "SIL/HIL replication of electric aircraft powertrain dynamics and inner-loop control for V&V of system health management routines," in *Annual Conference of the Prognostics and Health Management Society*, 2013.

[16] C. Quach, B. Bole, E. Hogge, S. Vazquez, M. Daigle, J. Celaya, A. Weber, and K. Goebel, "Battery charge depletion prediction on an electric aircraft," in *Annual Conference of the Prognostics and Health Management Society 2013*, October 2013, pp. 503–512.

[17] B. Bole, M. Daigle, and G. Gorospe, "Online prediction of battery discharge and estimation of parasitic loads for an electric aircraft," in *Second European Conference of the Prognostics and Health Management Society 2014*, July 2014, pp. 23–32.

Dr. Katelyn Jarvis is an applied mathematician and research engineer at NASA Ames Research Center. Katy received her B.S. in Mathematics and M.S. and Ph.D. in Applied Mathematics from the University of California, Davis. At NASA, Katy is a researcher within the System-Wide-Safety project and a member of the Diagnostics and Prognostics group. Her research interests include development of novel algorithmic approaches to prognostics, improvement of computational efficiency of these methods, and application of prognostics in human-health related fields.

Christopher Teubert is the group lead for the Diagnostics and Prognostics group at NASA Ames Research Center. Christopher received his B.S. in Aerospace Engineering from Iowa State University and his M.S. in Computer Science and Engineering from Santa Clara University. While at NASA, Christopher performed research in software architectures for prognostics, is the lead of NASA's Prognostics CoE, and is the lead developer for the Prognostics Python Packages (`prog_models` and `prog_algs`).

Dr. Wendy Okolo is the Associate Project Manager for NASA's System-Wide Safety project, focused on the development of new research tools, innovative aerospace technologies, and re-defined operational methods that will enable the safe operations of unmanned vehicles in the national airspace. As an aerospace controls research engineer, her expertise is in unconventional controls system design and optimization for air and space vehicles.

Dr. Chetan Kulkarni is a Research Engineer with KBR Inc, at NASA Ames Research Center, Calif. He received the B.E. degree in Electronics and Electrical Engineering from University of Pune, India in 2002 and the M.S. and Ph.D. degrees in Electrical Engineering from Vanderbilt University, Nashville, TN, in 2009 and 2013, respectively. His current research interests include physics-based modeling, model-based diagnosis and prognosis for complex systems. Dr. Kulkarni is a member of the Prognostics and Health Management (PHM) Society, SM AIAA and SM IEEE.