Reconceptualizing the Prognostics Digital Twin for Smart Manufacturing with Data-Driven Evolutionary Models and Adaptive Uncertainty Quantification

Jack Murray [1], Brandon Chamberlain [2], Nicholas Hemleben[3], Daniel Ospina Acero[4], Indranil Nayak [5], Andrew VanFossen[6], Frank Zahiri[7], Mrinal Kumar[8]

[1,2,4,6,8] *Mechanical & Aerospace Engineering, The Ohio State University, 201 W. 19th Avenue, Columbus, OH 43210, USA*
murray.964@buckeyemail.osu.edu
chamberlain.187@buckeyemail.osu.edu
ospinaacero.1@osu.edu
vanfossen.24@buckeyemail.osu.edu
kumar.672@osu.edu

[3] *PointPro Inc., 7658 Johntimm Ct, Dublin, OH 43017, USA*
nicholas@pointprousa.com

[5] *Electrical & Computer Engineering, The Ohio State University, 2015 Neil Avenue, Columbus, OH 43210, USA*
nayak.77@buckeyemail.osu.edu

[7] *402 Commodities Maintenance Group (CMXG), Warner Robins Air Logistics Complex, Robins AFB, GA 31098*
feraidoon.zahiri@us.af.mil

ABSTRACT

This work presents an integrated architecture for a prognostic digital twin for smart manufacturing subsystems. The specific case of cutting tool wear (flank wear) in a CNC machine is considered, using benchmark data sets provided by the Prognostics and Health Management (PHM) Society. This paper emphasizes the role of robust uncertainty quantification, especially in the presence of data-driven black- and gray-box dynamic models. A surrogate dynamic model is constructed to track the evolution of flank wear using a reduced set of features extracted from multi-modal sensor time series data. The digital twin's uncertainty quantification engine integrates with this dynamic model along with a machine emulator that is tasked with generating future operating scenarios for the machine. The surrogate dynamic model and emulator are combined in a closed-loop architecture with an adaptive Monte Carlo uncertainty forecasting framework that allows prediction of quantities of interest critical to prognostics within user-prescribed bounds. Numerical results using the PHM dataset are shown illustrating how the adaptive uncertainty forecasting tools deliver a trustworthy forecast by maintaining predictive error within the prescribed tolerance.

1. INTRODUCTION

Industries with high-value physical assets incur significant expense, to the tune of millions of dollars in a year, due to a lack of usable insights into productivity optimization and, in particular, unnecessary downtime created by scheduled preventive maintenance (Menon, Shah, & Coutroubis, 2018; Ramakrishna, Khong, & Leong, 2017; Vogl, Weiss, & Helu, 2019). There is significant recent literature on the development of digital (computational) solutions for solving prognostics related problems for such systems and subsystems. At its core, prognostics requires uncertainty forecasting using a sufficiently accurate dynamic model of system degradation. Addressing issues of data sufficiency, accuracy and scalability of model construction and uncertainty forecasting are open questions, especially as it pertains to timely and trustworthy prediction of critical events that foreshadow system failure (Wright & Davidson, 2020; Vogl et al., 2019). There is keen interest in the academia and industry alike to develop innovative tools that, by capturing their specific evolutionary characteristics, effectively predict failure before it happens.

This allows operators to take corrective action in time, ultimately maximizing asset productivity.

Computational representations of complex evolutionary models, like those used for prognostics, have been referred to in the last few years as *digital twins*. Unfortunately, the term "digital twin" has transformed into a catch-all phrase that conveys little meaning in the absence of proper context (Wright & Davidson, 2020; Errandonea, Beltrán, & Arrizabalaga, 2020). As explained in Ref. (Wright & Davidson, 2020), not only does the term digital twin vary from application scenario and industry, but also it has started to face the natural backlash associated with being identified as a wildcard term. With the object of reducing the vagueness of the term, Ref.(Wright & Davidson, 2020) proposes a definition of digital twin with three pillars: ***(i)*** a computational model of real-world object or process, ***(ii)*** an evolving data-set relating to the object or process, and ***(iii)*** a means of dynamically updating the model to better conform to the data. Worth highlighting in this definition is the central role that data plays. This is the case because it enables the direct and continuous connection with the behavior of the real-life object/process as it evolves in its operation. Naturally, such a strong reliance on data from the real-world object/process automatically sheds light onto the treatment of uncertainty in the measurements, as it is a necessary component in imbuing physical measurements with adequate meaningfulness. Thus, properly accounting for and handling uncertainty, especially of the output quantities, is another key feature of any adequate digital twin, since it offers avenues to discern how robust the model results are and how much trust can be placed in them (Wright & Davidson, 2020)(Jimenez, Schwartz, Vingerhoeds, Grabot, & Salaün, 2020).

The use of digital twins in smart manufacturing is hardly a new idea. Digital twins have been conceptually and, in some cases, experimentally studied in almost all stages and dimensions of the manufacturing process of different industries: design stage, production stage and service stage (Tao, Zhang, & Nee, 2019, Ch. 2). See also refs. (Lu, Liu, Wang, Huang, & Xu, 2020; Qi & Tao, 2018; Li, Lei, & Mao, 2022; Lattanzi, Raffaeli, Peruzzini, & Pellicciari, 2021) for a list of comprehensive reviews on digital twin mediated smart manufacturing. The specific case of prognostics and health management has been investigated in some depth as well, with developments in the sub-stages of observation, analysis and decision (Tao et al., 2019, Ch. 7). Notwithstanding, proper treatment of the uncertainty inherent in the input data, the model and (especially) the output data, continues to be mostly unaddressed by the research community and manufacturing industries, resulting in considerable difficulties when evaluating the robustness and the trustworthiness of the digital twins' operation and output. We address this issue in the present work.

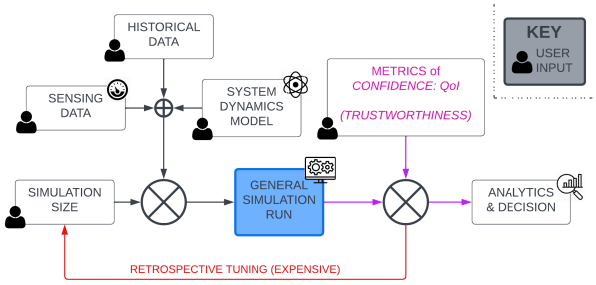
This paper concerns the framework of digital twins for the purpose of system prognostics, i.e. the task of predicting future system states over a process-dependent time window, leading to insights on system performance, including divergence from nominal and potential failure. Therefore, all characterizations and definitions must be understood within the boundaries of this stated prognostics application. Notably, in addition to incorporating the three-component framework of digital twin described above (model, evolving data, dynamic model updates) via a modular architecture, this paper puts front and center the characterization and forecasting of uncertainty emergent from models and data, leading to a quantifiably trustworthy prediction of system behavior for performance optimization and better maintenance scheduling.

2. CONTRIBUTION AND PROPOSED ARCHITECTURE

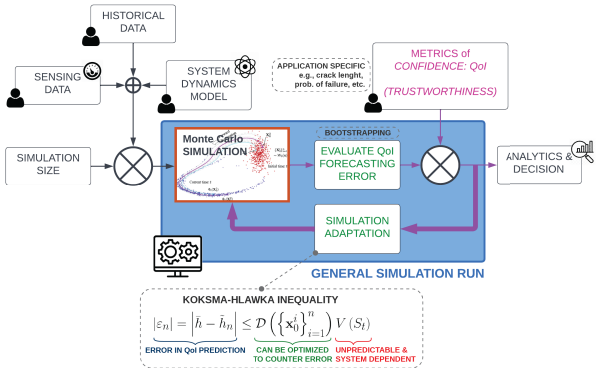
As stated in the previous section, a central point in the present work is the characterization and controllably accurate forecasting of uncertainty in the dynamic models employed with the digital twin. There are many frameworks for uncertainty forecasting (Yang & Kumar, 2019b) and each seeks to accurately propagate the state probability distribution through time. Traditionally, forecasting tools have been open-loop: see Fig.(1a) for an open-loop Monte Carlo propagation framework. The shown approach combines sensor data with system dynamics models and run simulations of future performance by *a priori* guessing the size of the simulation (number of initial conditions). The accuracy of the forecast of the quantities of interest (QoI) can only be estimated after the simulation is completed. If required error bounds are not achieved, a new simulation must be executed, and there exists no clear guidance on how the size of the simulation must be modified. This backward looking (retrospective) tuning is expensive and can easily take days to complete. Achieving a stipulated level of accuracy in uncertainty forecasting in complex systems, however technically difficult, configures a crucial aspect in the development of any digital twin. This has two supporting ideas:

• *Trustworthiness.* Decisions that depend on system forecasts need the forecast to be trustworthy, i.e., of decision-quality. For example, a milling machine with a predicted tool fracture length (QoI) exceeding $10^{-6}\ m$ would have to be serviced. A simulation that can predict this QoI within a guaranteed accuracy bound of $10^{-8}\ m$ (1% error) is trustworthy, whereas a simulation with unknown or unpredictable error bound would not be considered so.

• *Confidence.* Prognostics engineers identify key indicators of failure, whose expected order of magnitude estimates are known. E.g., in a milling machine operating in certain conditions, it may be known that the cutting speed (QoI) is about 4584 RPM, and speeds in excess of 5000 RPM indicate off-nominal operation and malfunction. A forecast that guar-



(a) Complex System Forecasting via Open-Loop Monte Carlo



(b) Closed-loop Adaptive Monte Carlo (AMC)

Figure 1. Traditional Monte Carlo framework for complex systems forecasting and the Adaptive Monte Carlo platform.

antees prediction of this QoI within 4.5 RPM (1% differential error) may instill sufficient confidence to recommend that the machine is maintained and repaired.

The work in this paper differentiates itself by making adaptive uncertainty forecasting the cornerstone of the prognostic digital twin. The closed-loop, adaptive Monte Carlo (AMC) forecasting platform shown in Fig.(1b) eliminates guesswork from achieving trustworthy uncertainty forecasting in complex systems. It enables front-end accuracy control in the prediction of user-defined quantities of interest (QoI: see Sec.(3.1.2)). It has a closed-loop architecture such that the simulation continuously monitors its performance vis-à-vis user stipulated QoI error bounds. When off-nominal forecasting error is detected, it determines *optimal modifications* to reestablish specified forecasting accuracy without any user intervention. In contrast to the feedback loop in Fig.(1a), which requires user adjustment after the general simulation has been finished, the feedback loop for simulation adjustment in Fig.(1b) is automatic and takes place while the general simulation is executing (see Secs.(3.1.3) and (3.1.4)). Fig.(2) now illustrates the overall proposed prognostic digital twin. There are three main modules, moving from the right and moving to the left:

1. The Uncertainty Quantification (UQ) Engine: which includes the adaptive AMC platform and a machine emulator. The machine emulator is a dynamic model tasked with generating future operational time series data for the process under expected operating conditions. In this work, a seasonal ARIMA model is employed to construct the machine emulator: Sec.(3.2).
2. The Dynamics Learning Engine: which employs machine learning tools in conjunction with available physics insights to build a data-driven surrogate of the the machine degradation process. This paper considers tool-wear dynamics. To keep the development straightforward, a simple LASSO regressor is employed to build the surrogate dynamic model (Sec.(3.4)).
3. The Sensing Module: which performs pre-processing on available multi-modal sensor and historical data and extracts key features ($\mathcal{F}^\star$ in Fig.(2)) that impact the dynamic evolution of machine degradation (tool flank wear, in this case). This key step of dimensionality reduction is achieved in this paper by employing a simple LASSO framework, described in Sec.(3.3.2).

In summary, the AMC platform integrates with two data-driven sub-modules: i.) the dynamics surrogate in the dynamics learning engine that translates key features ($\mathcal{F}^\star$ of a single run of the machine) to a numerical representation of machine degradation (flank wear in this paper), and, ii.) the machine emulator that creates time series sensor data representative of future *runs* of the machine, allowing a sweep of operational scenarios to support predictive uncertainty quantification in the evolution of flank wear. The next section describes individual sub-modules of the proposed prognostic digital twin.

3. DESCRIPTION OF SUB-MODULES

3.1. Adaptive Monte Carlo and Uncertainty Quantification

The architecture of the Adaptive Monte Carlo (AMC) platform (Fig.(1b)), as originally conceived in (Yang & Kumar, 2019b), requires a complete description of the system dynamics via differential equations. In almost all practical application scenarios, however, such a description is exceedingly scarce, and data-driven evolutionary models must be used instead. Measurement and simulation (from finite-element-method-based software packages, for example) data is abundant nowadays, as well as powerful machine machine learning tools that can exploit the data to construct dynamic evolutionary models. In this paper we present the extension of the AMC platform to operate with data-driven evolutionary models in the context of prognostics for smart manufacturing. A schematic of the proposed data-driven system is presented in Fig.(2).

3.1.1. System Dynamics

A nonlinear dynamic system with initial condition uncertainty and random excitation is given by the following

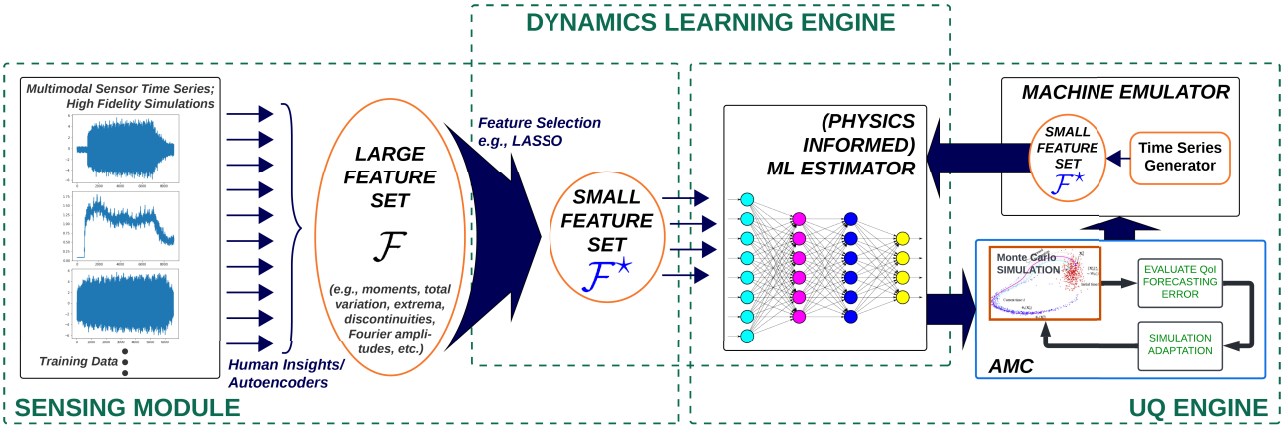


Figure 2. Schematic of the Prognostic Digital Twin.

stochastic differential equation (SDE) (Øksendal, 2003):

$$d\mathbf{x} = \mathbf{f}(t,\mathbf{x})dt + \mathbf{g}(t,\mathbf{x})d\mathbf{B}(t), \qquad \mathbf{x}_0 \sim \mathcal{W}_0(\mathbf{x}) \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^N$ denotes the system state and $\mathbf{x}_0$ the initial condition with associated probability density function (pdf) $\mathcal{W}_0$. The "process noise" term, $d\mathbf{B}(t)$, is an M-dimensional Brownian motion term with zero mean and correlation function $Q\delta(t_1 - t_2)$. The nonlinear vector function $\mathbf{f}(t,\mathbf{x}) : [0,\infty) \times \mathbb{R}^N \to \mathbb{R}^N$ corresponds to the deterministic part of the system and $\mathbf{g}(t,\mathbf{x}) : [0,\infty) \times \mathbb{R}^N \to \mathbb{R}^{N\times M}$ is a nonlinear matrix noise-influence function. For stochastic systems given in Eq.(1), the propagation of state pdf $\mathcal{W}(t)$ is given by the corresponding Fokker-Planck equation. In the case where process noise is absent, Eq.(1) reduces to: $d\mathbf{x} = \mathbf{f}(t,\mathbf{x})dt$, $\mathbf{x}_0 \sim \mathcal{W}_0(t_0,\mathbf{x})$, and the time evolution of the state-pdf $\mathcal{W}_0(t_0,\mathbf{x})$ is given by the stochastic Liouville equation (SLE). In Monte Carlo simulations (MCS), realizations of initial uncertainty are generated via random sampling $\{\mathbf{X}_0^i\}_{i=1}^n \sim \mathcal{W}_0$, where n denotes the total number of particles in the ensemble. Each particle is forward propagated in time through system dynamics to obtain an approximate representation of the evolved state pdf. That is,

$$\mathcal{W}_t(\mathbf{x}) \approx \{\Phi_t(\mathbf{X}_0^i)\}_{i=1}^n \tag{2}$$

where $\Phi_t(\cdot)$ is the system dynamics map that maps initial conditions to the current state. For dynamic systems with no process noise, Φ_t is the state-transition function $\mathbf{x}(t) = \Phi_t(\mathbf{x}_0)$ (Yang & Kumar, 2019a). The state-transition map $\Phi_t(\cdot)$ can be either physics-based (e.g., ordinary or stochastic differential equations) or data-driven, such as recurrent neural networks, autoencoders, etc.

3.1.2. Quantities of Interest (QoI) and Error Bounds

The AMC platform performs ensemble adaptations based on the difference between its measured forecasting accuracy and stipulated error bounds. Quantities of interest (QoI) are used to characterize the transient performance of MCS as it relates to $\mathcal{W}_t$. Each QoI is application specific and can be defined as anything from a simple state mean to the instantaneous heat flux on a vehicle. In general, QoIs are defined as the expected value of a function of the state, $h(\mathbf{x}_t)$, where $\mathbf{x}_t$ is the current state with density function $\mathcal{W}(\mathbf{x}_t) \equiv \mathcal{W}_t$ (Yang & Kumar, 2019b):

$$\underbrace{\overline{h}(\mathbf{x}_t) = \mathbb{E}_{\mathcal{W}_t}[h(\mathbf{x}_t)]}_{\text{Quantity of Interest: QoI}} = \int_{\Omega_t} h(\mathbf{x}_t)\mathcal{W}_t d\mathbf{x}_t \tag{3}$$

In the above expression, Ω_t is the state-space at time t. Prior to executing the AMC platform, its user must decide what application specific quantities ($\overline{h}(\mathbf{x}_t)$) must be forecast within prescribed bounds, and, what those prescribed bounds need to be to achieve a trustworthy forecast. While this is a nontrivial task, the relationship between the QoI and the MC approximation error developed in Ref. (Yang & Kumar, 2019b) allows AMC to be implemented in a wide variety of scenarios. Continuing under the assumption that the state-transition map $\Phi_t(\mathbf{x}_0) = \mathbf{x_t}$ is injective and continuously differentiable, Eq.(3) can also be expressed in terms of the initial state-pdf:

$$\mathbb{E}_{\mathcal{W}_t}[h(\mathbf{x}_t)] = \int_{\Omega_0} \underbrace{h[\Phi_t(\mathbf{x}_0)]}_{S_t(x_0)} \mathcal{W}_0 d\mathbf{x}_0 = \mathbb{E}_{\mathcal{W}_o}[S_t(\mathbf{x}_0)] \ , \tag{4}$$

where Ω_0 is the state-space at time t_0. $S_t(\cdot) \triangleq (h \circ \Phi_t)(\cdot) = h[\Phi_t(\cdot)]$ is an integrable composite function and $h(\cdot)$ is application dependent. For the complete derivation of Eq.(4), see Ref.(Yang & Kumar, 2019b). QoIs form the basis of ensemble adaptations in the AMC platform through the so-called Koksma-Hlawka inequality given below (Niederreiter, 1978):

$$|\overline{h} - \tilde{h}_n| \leq \mathcal{D}(\{\mathbf{X}_0^i\}_{i=1}^n)V(S_t) \ , \tag{5}$$

where $\tilde{h}_n$ is the sample-based approximation of $\overline{h}$. The left hand side of the above equation represents the departure of the the AMC forecast from the true QoI value. On the right

hand side, $V(S_t)$ is the variation of the composite function S_t, and $\mathcal{D}(\{\mathbf{X}_0^i\}_{i=1}^n)$ denotes the discrepancy of the ensemble at the initial time t_0. The function $V(S_t)$ is not "controllable", but the discrepancy function, $\mathcal{D}(\{\mathbf{X}_0^i\}_{i=1}^n)$ is, since the initial state distribution is known. As a result the product of terms on the right hand side of Eq.(5) can be controlled by performing an optimization of the discrepancy function. Eq.(6) below denotes the performance measure utilized in AMC to quantify the departure of the MC approximation ($\tilde{h}_n$) from the truth ($\overline{h}$) in direct terms of the quantity of interest (Yang & Kumar, 2019b).

$$\begin{aligned} \underbrace{\sigma_{\epsilon_n}}_{\text{Performance measure}} &= \sqrt{\mathbb{E}[(\overline{h}(\mathbf{x}_t) - \tilde{h}_n(\mathbf{x}_t))^2]} \\ &= \sqrt{\mathbb{E}[\epsilon_n^2]} \to \frac{\sigma(S_t((x)_0))}{\sqrt{n}} \end{aligned} \tag{6}$$

In AMC, the performance guarantee is defined as the ability to hold the estimation accuracy of the user-defined QoI within the prescribed accuracy bound. To estimate the quantity defined in Eq.(6), an approximation technique known as bootstrapping is employed. As described in (Efron, 1979), bootstrapping is introduced to solve the problem of estimating a sampling distribution of a specified random variable given a random sample $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ from an unknown probability distribution F (Efron, 1979).

3.1.3. Particle Addition

The theoretical basis of particle addition in the AMC platform is the Koksma-Hlawka inequality. Per Eq.(5), QoI forecasting error can be reduced by adding new particles that minimize ensemble discrepancy at the initial time. This optimization problem is difficult, on account of high dimensionality and non-convexity of the discrepancy function. To introduce the ensemble enhancer sub-module within the platform, first assume that the current particle ensemble at time t can be represented by P_t^p with p particles. Let the corresponding initial ensemble at time t_0 be $P_{t_0}^p \sim \mathcal{U}[0,1)^N$ since a uniform ensemble can be transformed into any target state-pdf. Without loss of generality, assume the propagated mean is the tracked QoI which allows the current MC estimation error (standard deviation of the MC estimation error) to be estimated via bootstrapping and denoted by $\mathcal{E}_t^p = \mathcal{E}(P_t^p)$. Now, if the estimation error is greater than the user-prescribed threshold ($\mathcal{E}_t^P > \mathcal{E}_t^{U^*}$), the ensemble enhancer is activated and new particles are introduced until the MC accuracy falls back within the defined thresholds(Yang & Kumar, 2019b). It should be noted that all particles are added at time t_0 to $P_{t_0}^p$ and then forward propagated to join the current ensemble at time t. The ensemble enhancer within the AMC platform (particle addition process) reduces discrepancy by exploiting the *space-filling* property of the ensemble to select particles for addition to $P_{t_0}^p$. A space-filling sample design leads to particles that fill out the domain of interest as homogeneously as possible (Janssen, 2013). Examples of some space-filling designs are Latin hypercubes, fractional designs, and orthogonal arrays (Grosso, Jamali, & Locatelli, 2009; Simpson, Poplinski, Koch, & Allen, 2001; Fang & Lin, 2003). There also exists several measures for quantifying this property, which include, but are not limited to, distance, entropy, and discrepancy. For the adaptive Monte Carlo platform, the discrepancy measure was chosen due to the relationship between the QoI (i.e., $\overline{h}(\mathbf{x}_t) = \mathbb{E}[h(\mathbf{x}_t)]$) and discrepancy given by the Koksma-Hlawka inequality (Eq.(5)). A numerically tractable approximation of discrepancy, $\mathcal{D}(P)$, was derived by Hickernell (Hickernell, 1998) and is given by Eq.(7):

$$\begin{aligned} \mathcal{D}_{CL_2}^2(P) \approx & \left(\frac{13}{12}\right)^N \\ & - \frac{2}{n}\sum_{k=1}^{n}\prod_{i=1}^{N}\left(1 + \frac{1}{2}\left|x_{ki} - 0.5\right| - \frac{1}{2}\left|x_{ki} - 0.5\right|^2\right) \\ & + \frac{1}{n^2}\sum_{k=1}^{n}\sum_{l=1}^{n}\prod_{i=1}^{N}\Bigg[1 + \frac{1}{2}\left|x_{ki} - 0.5\right| \\ & \qquad + \frac{1}{2}\left|x_{li} - 0.5\right| - \frac{1}{2}\left|x_{ki} - x_{li}\right|\Bigg], \end{aligned} \tag{7}$$

where N is the dimension of the state space, and n is the current number of samples plus additional candidates. With this, the particle addition procedure thus corresponds to solving the optimization problem defined by Eq.(8) directly

$$\mathbf{x}_*^{p+1} = \arg\min_{\mathbf{x}\in\mathbb{R}^N}\left[\mathcal{D}_{CL_2}^2(P_{t_0}^{(p+1)})\right], \tag{8}$$

where $\mathcal{D}_{CL_2}^2$ is Eq.(7) and $\mathbf{x}_*^{p+1}$ denotes the next optimal candidate to be included in $P_{t_0}^p$ thereby defining the $P_{t_0}^{(p+1)}$ ensemble. In other words, discrepancy (Eq.(7)) is minimized within the ensemble enhancer for every particle addition.

3.1.4. Particle Removal

During forward propagation, the error related to the current ensemble may decrease depending on the change in variation of the function S_t. That is, the error of the current ensemble may outperform its desired boundary ($\mathcal{E}_t^P < \mathcal{E}_t^{L^*}$) such that a reduction in ensemble size can be performed in the interest of reducing the computational load of future propagation (Yang & Kumar, 2019a). Within the AMC platform, particles are "halted" and not carried with the ensemble to the next time step rather than completely "removed." This allows particles to be reactivated at a later time step if needed without having to execute the optimization routine and propagate the particle(s) starting from t_0. Candidates are identified for removal based on their current significance with respect to the evaluated state probability density function. Particles with lower weights (evaluated state-pdf values) are considered for

removal with a greater probability. The method of characteristics (MOC) can be used to compute the solution of the SLE at a particle's current location in the state space at the current time (Yang, Buck, & Kumar, 2015). Using MOC, the ODE governing the evolution of the state pdf for each ensemble particle is given via $(\text{d/dt})\mathcal{W}(t,\mathbf{x}) = -\mathcal{W}(\mathbf{t},\mathbf{x}) \cdot \nabla \cdot \mathbf{f}(\mathbf{t},\mathbf{x})$ with the solution

$$\mathcal{W}(t,\mathbf{x}(t)) = \mathcal{W}_0[\mathbf{x}_0(\mathbf{x}(t))] \exp\left[-\int_{t_0}^{t} \nabla \cdot \mathbf{f}(\tau,\mathbf{x}_c(\tau))d\tau\right], \tag{9}$$

where $\mathbf{x}_0[\mathbf{x}(t)] = \Phi_t^{-1}(\mathbf{x}_t)$ are the initial conditions with respect to the particle's current location $\mathbf{x}(t)$. Considering the current ensemble (P_t^p), the probability of removing a member particle (x_t^i) from the ensemble is given by

$$Pr(\text{Remove } \mathbf{x}_t^i) = 1 - \frac{\mathcal{W}(t,\mathbf{x}_t^i)}{\sum_{j=1}^{p} \mathcal{W}(t,\mathbf{x}_t^j)} \tag{10}$$

where $i = 1, 2, \ldots, p$. Particle removal continues until the measured error is above the specified lower bound, $\mathcal{E}_t^n > \mathcal{E}_t^{L^*}$, thereby alleviating additional computational cost.

3.2. The Machine Emulator

In order to achieve robust predictive uncertainty quantification, a machine emulator is needed that creates realistic future operational time-series data. This is similar in effect to a data-augmentation system that create realistic time series sensing data "from the future". While further work is required in this area to achieve high-fidelity emulation, this paper employs a lightweight and functional method that can predict the time series generated from CNC use. A suitable tool for modeling and predicting time-series data is autoregressive integrated moving average (ARIMA) models (Hamilton, 1994). Recall first an autoregressive moving average (ARMA) model of order (p,q), in which given a time series X_t, we have

$$X_t = \sum_{i=1}^{p} \alpha_i X_{t-i} + \sum_{i=0}^{q} \theta_i \varepsilon_{t-i}\ , \quad \text{with } \theta_0 = 1\ .$$

where α_i, θ_i are coefficients that are determined by applying least squares on the predicted values of a exiting data set of the time series. Intuitively one can think of these coefficients as a measure of the contribution of the i^{th} previous time step to the value at the current time step. This reflects the belief that the time series is a stationary process after we have taken enough differences with the previous time steps (and scale the previous time steps by α_i). More concretely, if we look at

$$X_t - \alpha_1 X_{t-1} - \cdots - \alpha_{p'} X_{t-p'} = \varepsilon_t + \cdots + \theta_q \varepsilon_{t-q},$$

we can see that the left hand side is just an aggregate of several time steps of Brownian motion. To get to an ARIMA model we can refactor this with the lag operator L:

$$(1 - \sum_{i=1}^{p'} \beta_i L^i) X_t = (1 - \sum_{i=1}^{q} \varphi_i L^i)\varepsilon_t$$

where the β_i and φ_i are such that once we expand them we get equality with the expressions above. Now, if we believe that a major factor in the recurrence is just the previous time steps without scaling, then some of the β's above are simply 1. In this case we expect to be able to factor out of the form $(1-L)^d$. If we add in the factor above to the expansion, our expression for the time series is similar to how it was before:

$$(1 - \sum_{i=1}^{p'} \beta_i L^i)(1-L)^d X_t = (1 - \sum_{i=1}^{q} \varphi_i L^i)\varepsilon_t$$

We could simply adjust the β_i's to approximate the original formula. Now we have an ARIMA model (not just an ARMA) of order (p,d,q):

$$(1 - \sum_{i=1}^{p} \beta_i L^i)(1-L)^d X_t = (1 - \sum_{i=1}^{q} \varphi_i L^i)\varepsilon_t\ .$$

The advantage here with the order d terms is that they do not have to have coefficients trained by the data. Of course, we now have 3 instead of 2 hyper parameters to fit ((p,d,q) versus (p,q)).

Seasonal ARIMA Model: The final iteration we have here is to add a seasonal component to the ARIMA model. The motivation behind such a model can be seen in the following example: If one considers predicting construction prices, then the last 4 weeks of construction spending are very relevant, but the construction spending of 12 months ago (in a 4 week window) are also relevant. Thus, a seasonal ARIMA model with order (p,d,q), and with *seasonal* order (P,D,Q,m) is as follows:

$$\begin{aligned}(1 - \sum_{i=1}^{p} \beta_{i,1} L^i)(1-L)^d (1 - \sum_{i=1}^{P} \beta_{i,2} L^{i*m})(1-L^m)^D X_t \\ = (1 - \sum_{i=1}^{q} \varphi_{i,1} L^i)(1 - \sum_{i=1}^{Q} \varphi_{i,2} L^{i*m})\varepsilon_t\end{aligned}$$

Now we can look back a full period of data, and with the way the polynomials expand out, we have terms for the previous p time steps. Additionally, if we go back a period we have terms for the p previous time steps of that period, and we have that for the previous P periods.

Generating Time series: The ARIMA sub-module generates a time series from a ARIMA model trained on each run of the PHM data set. In the case of the PHM data set we generate 7 time series for the 7 sensor series of each machine run. To predict an individual sensor's time series, a time series is

generated from each of the runs within the training data set. These time series are then combined weighted on each of the AIRMA model's performances in predicting the last 10% of it's runs time series. That is to say if run 1's ARIMA model is twice as accurate as run 2's ARIMA model than it will receive twice the weighting in the aggregate sum.

3.3. Sensing Module

The sensing module is tasked with pre-processing the training multi-modal sensor. It identifies a small set of features that are most impactful to the prediction of flank wear (QoI).

3.3.1. Data Pre-processing

Data of manufacturing machines is of three distinct forms, namely **(i)** evolution data, **(ii)** operation data and **(iii)** time-series data. The pre-processing procedure of the data diverges into their own respective sub-procedures:

(i) Evolution data is information that can only be measured when a machine is not in operation, such as the wear of a machine and the cumulative lifetime of a machine. Therefore, the evolution data relevant to a given operation are measurements made before and after along with their differences. However, only measurements made after operation are usually reported. So when missing, the remaining evolution data is populated using data of preceding operations.

(ii) Operation data is information relating to how a machine was operated, such as machine settings, use case identification numbers, etc. Data of this form can either be continuous or categorical. For continuous information, the data is scaled, shifted, and normalized such that all values lie within the interval $[0, 1]$. For categorical information, the data is encoded as new binary features.

(iii) Time-series data is information carrying sensor or control signals recorded during machine operation. When the length of operation is constant, many standard signal processing methods are applicable. However, in industry, operation of manufacturing machines occurs on infrequent intervals. As a result, the methods used to characterize the time-series are agnostic to its length and are as follows:

1. The first of such methods quantifies any jumps present across operations by differencing the final time-series value of the preceding operation with the initial value of the current operation.
2. The second of which quantifies the joint statistical properties of the time-series' by determining parameters of the joint moment generating function, calculated up to the 5th moment.
3. The third of which quantifies the key frequencies present in the time-series with the Fourier Transform. Ten frequencies with the greatest Fourier coefficient magnitudes are sampled from high, medium, and low frequency ranges.

The result of data pre-processing is a large set of potential features; however, identification of a more tractable small set of features to construct the evolutionary model is still required.

3.3.2. Feature Selection

Selection of the best characteristic data features to serve as model parameters is performed by the Least Absolute Shrinkage and Selection Operator (LASSO) (Tibshirani, 1996) and was chosen for its well documented applications in the machine learning field. LASSO is a regression analysis technique that adds a 1-norm regularization term to the standard least-squares formulation of a regression procedure. The LASSO problem can be formulated as follows:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1 \,, \tag{11}$$

where $\|\cdot\|_2$ and $\|\cdot\|_1$ denote respectively the 2- and 1-norm. $\boldsymbol{\beta}$ represents the vector of p regression coefficients and $\lambda - 0$ represents the regularization parameter. The appropriate value of λ for best performance must be determined, and although numerous techniques exist, by far the most commonly adopted is cross-validation.

The most relevant feature of LASSO is encapsulated by the structure of the second term of the cost function in Eq.(11) and its influence on the behavior of the regression coefficients $\boldsymbol{\beta}$ once they are found: for a large enough λ, a great degree of sparsity is induced in the LASSO coefficients $\boldsymbol{\beta}$, as the majority of them are made 0. This then means that only a handful of predictors in $\mathbf{X}$ are active in the solution of Eq.(11). Such a feature establishes LASSO as one of the main mechanisms to perform feature subset selection, as it automatically allows one to represent the set of responses with a minimal set of "relevant" predictors (Hastie, Tibshirani, Friedman, & Friedman, 2009, Chapter 3). Once the solution $\boldsymbol{\beta}^*$ to Eq.(11) is obtained, the estimate $\hat{\mathbf{y}}_{\text{LASSO}}$ is then found via:

$$\widehat{\mathbf{y}}_{\text{LASSO}} = \mathbf{X}\boldsymbol{\beta}^* \,. \tag{12}$$

To employ the LASSO procudure, the QoI (flank wear) and all features dependent on its known value are first extracted from the large feature set and defined as a set of potential true responses $\mathbf{y}$. Then, the remaining large feature set is defined as predictors $\mathbf{X}$. Finally, the LASSO framework determines the set of $\boldsymbol{\beta}^*$ containing at most the number of predictors demanded by the modeling method. Of the found $\widehat{\mathbf{y}}$ solutions, that which constructs the QoI most accurately defines the feature selection by its $\boldsymbol{\beta}^*$ used to downsize the large feature set to the small feature set.

3.4. Data-driven Evolutionary Model

The Dynamics Learning Engine seeks to model the true underlying dynamics of a system, without knowing the dynamics themselves. As a result, we consider data-driven methods to map a subset of generated statistical features to the QoI (flank wear). Initially, we consider methods that perform this mapping with significant nonlinearity, which includes linear regression with autoregressive terms or nonlinear basis, as well as Feed-Forward Neural-Network (FFN) architectures. Upon further evaluation, we found that regression with L1 regularization produced noticeably lower mean-squared error losses when compared against FFN's trained on the same data. As a result, we utilize this L1 regularization when predicting the flank wear. The constructed small feature set in Sec.(3.3.2) isolates the characteristics which impact the system most significantly. From here, the system's dynamics are approximated by mapping the "differenced" QoI, or the difference between the QoI of the prior and current operating conditions, from the small set of features. We make our predictions of the QoI (flank wear) by directly using this LASSO model, as shown in Eq.(12).

4. RESULTS AND DISCUSSION

4.1. Application Scenario

The 2010 Prognostic and Health Management (PHM) Society's annual data challenge competition tasked participants with estimating the remaining useful life on a CNC milling machine tool bit from force, vibration, and acoustic sensor information (Society, May 18,2021). The dataset contains flank wear measurements of the three cutting edges of 6 mm ball nose tungsten carbide cutters. These measurements were taken proceeding CNC operations with sensor time series measurements gathered during CNC operations on 50 kHz sensing channels. Data was collected for 3 separate cutters, each conducting 315 operations. For all operations, the CNC machine was set with 10,400 RPM spindle speed, 1,555 mm/min feed rate, 0.125 mm radial depth of cut, and 0.2 mm axial depth of cut (Society, May 18,2021). To be compatible with the proposed twin, the QoI was defined as the average flank wear of all three flutes. For training across all feature selection, QoI prediction, and ARIMA modeling sub-processes, the first 90% of operations from the data were selected. The remaining 10% of operations were used in system-level validation and analysis.

4.2. Numerical Results

4.2.1. LASSO Performance

This section is under development. Early results are presented here for guidance. The final paper will carry improved results with a more robust training datasets and iterative model improvements. The training data for the LASSO model was the first 284 data points of the PHM data set, the remaining 31 runs are then used as the testing set. LASSO regression identified 1771 features that then made up the small feature set. The data was initially mapped into the small feature set, then subdivided into the training/testing sets. **Fig.**(3) **depicts early results of the LASSO model. Note that these results are the outcome of the very first instance of training. They will be improved before inclusion in the final manuscript.**

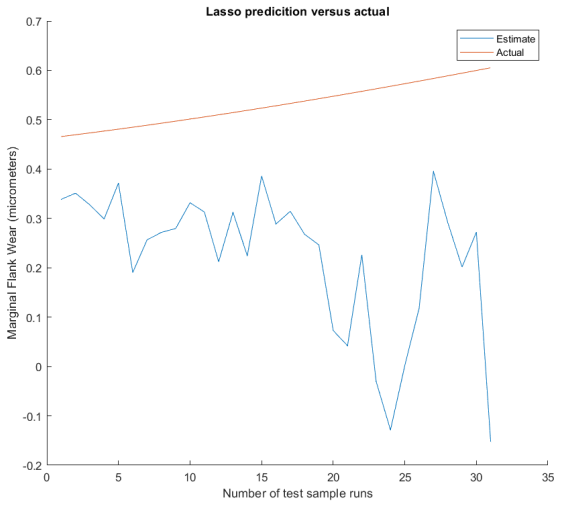


Figure 3. **Early Results:** LASSO Model prediction against actual value

4.2.2. Digital Twin Performance

We note that these are very early results. The final paper will carry results with much greater accuracy as the machine learning models are refined and trained with a larger training data set.

Results correspond to use of the AMC platform for 31 time steps and an error threshold of 1 μm. The particles are initialized to begin simulation at the end of the 90% mark of the PHM data. This is done such that the remaining 10% of data can be used to validate predicted flank wear values. Fig.(4) displays the accuracy of the uncertainty forecasting operation vis-à-vis prescribed error thresholds ($= 1\ \mu m$). In Fig.(4), blue circles show where the simulation was found to be above the prescribed error threshold of 1 μm. Particles were added and the accuracy improved until within tolerances once again (corresponding black diamond mark directly underneath the blue). As can be seen in Fig.(4) the simulation required ensemble enhancements at numerous time steps throughout propagation with the consistent increase in the number of particles needed to meet the accuracy threshold depicted in Fig.(5). This behavior is typical in uncertainty propagation as it becomes increasingly difficult to bound the

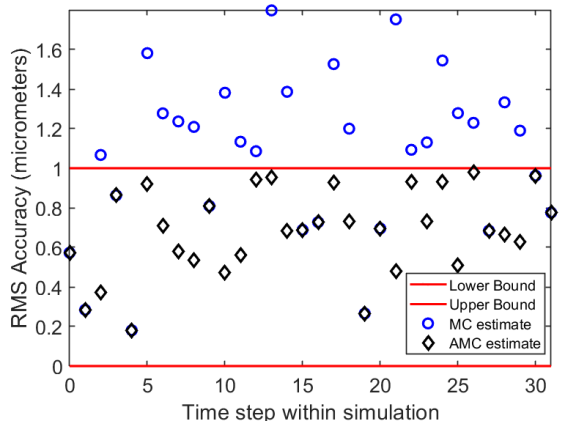


Figure 4. Accuracy of AMC prediction against simulation time

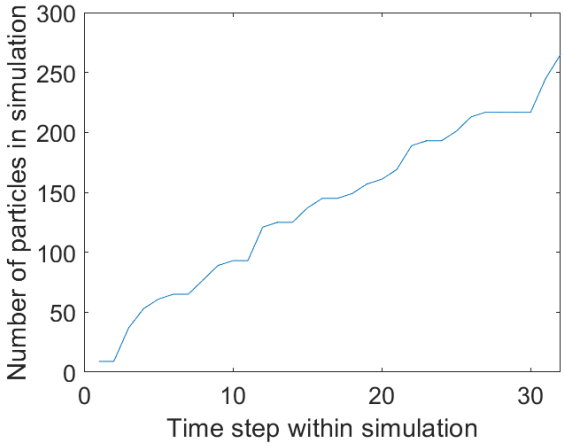


Figure 5. Size of the AMC Ensemble against simulation time

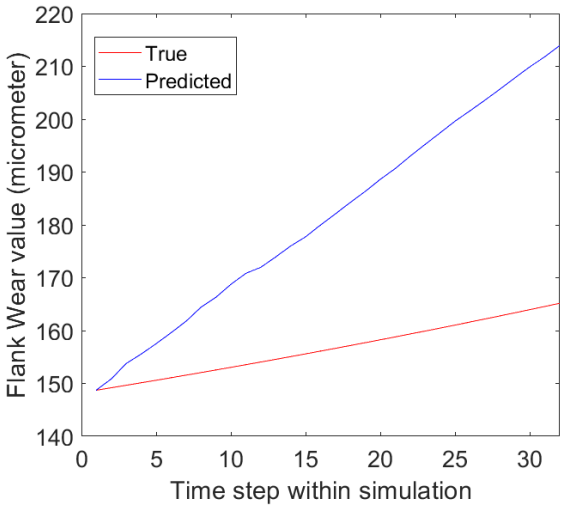


Figure 6. **Early Results:** predicted flank wear against measured values

accuracy of QoI at future time steps. The flank wear simulation concluded with 265 particles after an initial ensemble size of 5, with the first ensemble adaptation occurring at time step 3. That is, the initial ensemble size was sufficient to accurately compute the QoI through time step 3 with an upper threshold of 1μm. Afterwards, additional particles were included in the simulation to continue to bound the estimate of the QoI (flank wear) below 1μm.

Figure 6 plots the predicted flank wear by the AMC platform against the measured values of the last 10% of the PHM data set. **These are early results with a very small training data set. The final version of the paper will carry improved results with refined machine emulator and dynamics models and increased size of the training datasets.**

5. CONCLUSION

This paper presents a precise framework for a smart manufacturing prognostic digital twin by integrating dynamic data-driven degradation models with an adaptive uncertainty forecasting framework based on closed-loop Monte Carlo simulations. The quality of uncertainty forecasting is controlled by defining application specific quantities of interest (in this case, flank wear) and associated bounds on its forecasting error to maintain trustworthiness. Three sub-modules constitute the twin: i.) the uncertainty quantification engine, combining the AMC platform with a machine emulator, ii.) the dynamics learning engine providing a surrogate dynamic model of flank wear evolution, iii.) the sensing module that preprocesses multi-modal sensing data and identifies key features for building surrogate dynamics and the machine emulator. The current iteration of the digital twin includes simple regression-based sub-modules for feature selection (LASSO),

surrogate dynamic modeling (LASSO) and machine emulation (ARIMA). **This version of the manuscript contains very early results for the digital twin. As mentioned before, the final paper will carry improved results on account of refinements to the machine emulator and the dynamics learning engine.**

ACKNOWLEDGMENT

This work is supported by the US Air Force AFWERX SBIR Program under Grant No. FA8649-22-P-0701.

REFERENCES

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, *7*(1), 1–26.

Errandonea, I., Beltrán, S., & Arrizabalaga, S. (2020). Digital twin for maintenance: A literature review. *Computers in Industry*, *123*, 103316.

Fang, K.-T., & Lin, D. K. (2003). Ch. 4. uniform experimental designs and their applications in industry. In *Statistics in industry* (Vol. 22, pp. 131–170). Elsevier.

Grosso, A., Jamali, A., & Locatelli, M. (2009). Finding maximin latin hypercube designs by iterated local search heuristics. *European Journal of Operational Research*, *197*(2), 541–547.

Hamilton, J. (1994). Time series analysis. Princeton University Press.

Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction* (Vol. 2). Springer.

Hickernell, F. (1998). A generalized discrepancy and quadrature error bound. *Mathematics of Computation*, *67*(221), 299–322.

Janssen, H. (2013). Monte-carlo based uncertainty analysis: Sampling efficiency and sampling convergence. *Reliability Engineering & System Safety*, *109*, 123–132.

Jimenez, J. J. M., Schwartz, S., Vingerhoeds, R., Grabot, B., & Salaün, M. (2020). Towards multi-model approaches to predictive maintenance: A systematic literature survey on diagnostics and prognostics. *Journal of Manufacturing Systems*, *56*, 539–557.

Lattanzi, L., Raffaeli, R., Peruzzini, M., & Pellicciari, M. (2021). Digital twin for smart manufacturing: A review of concepts towards a practical industrial implementation. *International Journal of Computer Integrated Manufacturing*, *34*(6), 567–597.

Li, L., Lei, B., & Mao, C. (2022). Digital twin in smart manufacturing. *Journal of Industrial Information Integration*, *26*, 100289. doi: https://doi.org/10.1016/j.jii.2021.100289

Lu, Y., Liu, C., Wang, K. I.-K., Huang, H., & Xu, X. (2020). Digital twin-driven smart manufacturing: Connotation, reference model, applications and research issues. *Robotics and Computer-Integrated Manufacturing*, *61*, 101837. doi: https://doi.org/10.1016/j.rcim.2019.101837

Menon, S., Shah, S., & Coutroubis, A. (2018). An overview of smart manufacturing for competitive and digital global supply chains. In *2018 ieee international conference on technology management, operations and decisions (ictmod)* (p. 178-183). doi: 10.1109/ITMC.2018.8691224

Niederreiter, H. (1978). Quasi-monte carlo methods and pseudo-random numbers. *Bulletin of the American Mathematical Society*, *84*(6), 957–1041.

Øksendal, B. (2003). *Stochastic differential equations: An introduction with applications* (6th ed.). Springer-Verlag Berlin Heidelberg.

Qi, Q., & Tao, F. (2018). Digital twin and big data towards smart manufacturing and industry 4.0: 360 degree comparison. *IEEE Access*, *6*, 3585-3593. doi: 10.1109/ACCESS.2018.2793265

Ramakrishna, S., Khong, T. C., & Leong, T. K. (2017). Smart manufacturing. *Procedia Manufacturing*, *12*, 128-131. (International Conference on Sustainable and Intelligent Manufacturing, RESIM 2016, 14-17 December 2016, Leiria, Portugal)

Simpson, T., Poplinski, J., Koch, P. N., & Allen, J. (2001). Metamodels for computer-based engineering design: Survey and recommendations. *Engineering with Computers*, *17*, 129–150.

Society, P. (May 18,2021). 2010 phm society conference data challenge phmsociety. *International Journal of Prognostics and Health Management*.

Tao, F., Zhang, M., & Nee, A. Y. C. (2019). *Digital twin driven smart manufacturing*. Academic Press.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267-288.

Vogl, G. W., Weiss, B. A., & Helu, M. (2019). A review of diagnostic and prognostic capabilities and best practices for manufacturing. *Journal of Intelligent Manufacturing*, *30*(1), 79–95.

Wright, L., & Davidson, S. (2020). How to tell the difference between a model and a digital twin. *Advanced Modeling and Simulation in Engineering Sciences*, *7*(1), 1–13.

Yang, C., Buck, K., & Kumar, M. (2015). An evaluation of monte carlo for nonlinear initial uncertainty propagation in keplerian mechanics. In *2015 18th international conference on information fusion (fusion)* (p. 119-125).

Yang, C., & Kumar, M. (2019a). An adaptive monte carlo method for uncertainty forecasting in perturbed two-body dynamics. *Acta Astronautica*, *155*, 369–378.

Yang, C., & Kumar, M. (2019b). Closed-loop adaptive monte carlo framework for uncertainty forecasting in nonlin-

ear dynamic systems. *Journal of Guidance, Control, and Dynamics*, *42*(6), 1218-1236.