Sequential Monte Carlo: Enabling Real-time and High-fidelity Prognostics

Patrick E. Leser[1], Jacob D. Hochhalter[2], James E. Warner[3], Geoffrey F. Bomarito[4], William P. Leser[5], and Fuh-Gwo Yuan[6]

[1,2,3,4,5] *NASA Langley Research Center, Hampton, VA, 23681, USA*
patrick.e.leser@nasa.gov
jacob.d.hochhalter@nasa.gov
james.e.warner@nasa.gov
geoffrey.f.bomarito@nasa.gov
william.leser@nasa.gov

[6] *North Carolina State University, Raleigh, NC, 27695, USA*
yuan@ncsu.edu

Abstract

Uncertainty quantification and propagation form the foundation of a prognostics and health management (PHM) system. Particle filters have proven to be a valuable tool for this reason but are generally restricted to state-space damage models and lack a natural approach for quantifying model parameter uncertainty. Both of these issues tend to inhibit the real-world application of PHM. While Markov chain Monte Carlo (MCMC) sampling methods avoid some of these restrictions, they are also inherently serial, and, thus, MCMC can become intractable as model fidelity increases. Over the past two decades, sequential Monte Carlo (SMC) methods, of which the particle filter is a special case, have been adapted to sample from a single, static posterior distribution, eliminating the state-space requirement and providing an alternative to MCMC. Additionally, SMC samplers of this type can be run in parallel, resulting in drastic reductions in computation time. In this work, a potential path toward real-time, high-fidelity prognostics using a combination of surrogate modeling and a parallel SMC sampler is explored. The use of SMC samplers to enable tractable parameter estimation for full-fidelity (i.e., non-surrogate-assisted) damage models is also discussed. Both of these topics are studied in the context of fatigue crack growth in a geometrically complex, metallic specimen subjected to variable amplitude loading.

1. Introduction

Uncertainty quantification and propagation are essential to the implementation of prognostics and health management (PHM) systems. By acquiring periodic diagnoses of structural health via non-destructive evaluation or an on-board structural health monitoring system, Bayesian inference can be used to quantify the uncertainty in a predictive model. Propagating this uncertainty via simulation enables probabilistic predictions of a quantity of interest (e.g., remaining useful life, residual strength, mission readiness). As more data is acquired, predictions can be updated to reflect the change in uncertainty over time. Two approaches to uncertainty quantification are commonly used in prognostics. The first involves the quantification of model state uncertainty over time, commonly referred to as Bayesian filtering or Bayesian tracking. The second approach involves estimating model parameter uncertainty as a static, joint distribution where uncertainty is reduced as more data is acquired. This latter approach is referred to as model calibration or parameter estimation.

Particle filters, a particular sub-class of sequential Monte Carlo (SMC) methods, have been used extensively for Bayesian filtering and Bayesian tracking problems (Doucet, Godsill, & Andrieu, 2000; Arulampalam, Maskell, Gordon, & Clapp, 2002; Theodoridis, 2015). Particle filters work by estimating a model's state probability distribution as it evolves in time via Bayesian inference. Upon the receipt of new data, the state distribution is updated recursively using a form of sequential importance sampling (SIS). SIS requires a state transition relation that predicts the subsequent state distribution based on the current state. In practice, this is accomplished by propagating the uncertainty in the current estimated state through the model to the time at which new data will be acquired. This predicted probability distribution, which, by definition, does not incorporate the new data, is updated via importance sampling and Bayes' Theorem when new data is observed. At any given time, non-deterministic predictions of a quantity of interest can be attained by propagating uncertainty in the current model state to a given time or until a stopping criterion is met.

Numerous examples of particle filter implementations can be found in the PHM literature; see, for example, (Cadini, Zio, & Avram, 2009; Orchard & Vachtsevanos, 2009; Saha & Goebel, 2009; Zio & Peloni, 2011). Studying these works yields two observed shortcomings. First, particle filtering algorithms have generally been restricted to state-space representations. Second, particle filters lack a method for estimating uncertainty in static model parameters (e.g., material properties) (Liu & West, 2001), the values of which are often unknown and uncertain. Researchers have attempted to address the latter issue by introducing methods such as artificial dynamics (Daigle & Goebel, 2011) and through a hybrid approach in which parameter uncertainty is estimated via Markov chain Monte Carlo (MCMC) at each resampling step in a standard particle filtering algorithm (Corbetta, Sbarufatti, Manes, & Giglio, 2015).

MCMC sampling methods are an alternative to particle filters for uncertainty quantification that are generally less restrictive of model class and are naturally suited for static parameter estimation (Andrieu, De Freitas, Doucet, & Jordan, 2003; Kaipio & Somersalo, 2006; Smith, 2014; Theodoridis, 2015). These methods involve constructing a Markov chain through the model parameter space, for which the stationary distribution is an approximation of the parameter posterior distribution of interest. The chain is built sample by sample, where a new sample is randomly proposed based only on the previous sample (i.e., a Markov process). A set of simple rules based on Bayes Theorem dictate whether or not proposals are accepted into the chain or rejected. This approximation approaches the true distribution as the number of total samples in the chain approaches infinity. The primary disadvantage of MCMC methods is that they are computationally intensive due to the Markovian nature in which samples are drawn and the model is evaluated (i.e., these methods are difficult to parallelize).

Depending prognosticator's choice of model, the state-space requirement of particle filters or the serial nature of MCMC methods may be prohibitive. This is especially the case for high-fidelity models which involve solving large systems of equations numerically (e.g., finite element analysis) in order to simulate damage progression. Flexibility in model fidelity is critical to the continued adoption of PHM methodologies for real-world applications, as engineering structures are often complex and might necessitate higher fidelity models. The statistical methods used to form non-deterministic predictions should not restrict the breadth of available models. While particle filtering can be computationally efficient in certain cases, MCMC is relatively slow, even when using quick-to-evaluate analytical models. In addition to facilitating model flexibility, statistical approaches used for static parameter estimation in a PHM system need to be, in general, faster than current MCMC methods.

A potential solution to all of the aforementioned issues is the parallel SMC sampler, which is a generalization of the particle filter. Over the past two decades, SMC methods have been adapted to sample from a single, static posterior distribution (Peters, 2005; Del Moral, Doucet, & Jasra, 2006; Peters, Fan, & Sisson, 2012). As a result, the state-space restriction of particle filtering is removed and SMC becomes an attractive alternative to MCMC. A key advantage of SMC samplers over their MCMC counterparts is that model evaluations are independent and can be run in parallel, potentially improving parameter estimation times by orders of magnitude.

The work presented here focuses on the efficiency and performance of a generalized parallel SMC sampler for static parameter estimation in the context of non-deterministic prognostics. In particular, two critical benefits provided by SMC to the PHM field are discussed. First, a potential path toward real-time, high-fidelity prognostics using a combination of surrogate modeling and a parallel SMC sampler is demonstrated. Second, the use of SMC samplers to enable tractable parameter estimation for full-fidelity (i.e., non-surrogate-assisted) damage models is discussed. Both of these topics are explored in the context of fatigue crack growth in a geometrically complex metallic specimen. Three introductory sections precede the results and conclusions. These sections include discussion of (i) the relevant SMC theory, (ii) the case study used to demonstrate the proposed method, and (iii) an MCMC-based benchmark for method evaluation.

2. ADAPTING SEQUENTIAL MONTE CARLO FOR STATIC PARAMETER ESTIMATION

The theory presented in this section is based on work by Nguyen et al. (Nguyen, Septier, Peters, & Delignon, 2016), and an attempt was made to maintain consistency in notation. The review here is brief, and the reader is encouraged to refer to (Nguyen et al., 2016) for more detail.

Assume that the relationship between a predictive damage model, $\mathcal{M}$ and a set of available damage diagnoses, or observations, is given by

$$Y = \mathcal{M}(\boldsymbol{\theta}) + \epsilon, \tag{1}$$

where Y, $\boldsymbol{\theta}$, and ϵ are random variables representing the measurements, model parameters and measurement errors, respectively. The goal of parameter estimation is to quantify the uncertainty in $\boldsymbol{\theta}$ given realizations y. In the context of prognostics, this amounts to a non-deterministic calibration of model parameters using noisy diagnostic information obtained throughout the life of the monitored component or structure. Probabilistic predictions of some quantity of interest are obtained via propagation of the quantified parameter uncertainty through the model to some failure condition.

According to Bayes' Theorem, the parameter posterior distribution of interest is

$$p(\boldsymbol{\theta}|y) = \frac{p(y|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int_{\Theta^d} p(y|\boldsymbol{\theta})p(\boldsymbol{\theta})} = \frac{p(y|\boldsymbol{\theta})p(\boldsymbol{\theta})}{Z}, \tag{2}$$

where $p(y|\boldsymbol{\theta})$ is the likelihood expression, which defines the probability of observing y given $\boldsymbol{\theta}$, and $p(\boldsymbol{\theta})$ is the prior distribution, which aggregates any knowledge that is available *a priori* regarding the parameter probability distributions. A direct solution to Equation 2 is typically intractable, particularly when considering the normalizing constant, Z, which requires integration over the entire d-dimensional parameter space Θ^d. However, making the assumption that the errors are normally distributed, $\epsilon \sim \mathcal{N}(0, \sigma^2)$, allows for the point-by-point evaluation of the unnormalized posterior distribution; i.e., $p(\boldsymbol{\theta}|y) \propto p(y|\boldsymbol{\theta})p(\boldsymbol{\theta})$.

In general, the goal of Sequential Monte Carlo (SMC) methods is to approximate Equation 2 via sequential importance sampling (Theodoridis, 2015). One of the keys to successful importance sampling is selecting an importance distribution, $\eta(\boldsymbol{\theta})$, that produces samples in the region(s) where the target distribution, $\pi(\boldsymbol{\theta})$, exhibits high probability density. That is, if the two distributions are vastly different, estimators constructed using importance sampling will exhibit high variance. This issue is exacerbated when sequentially sampling in this manner because the effects of poor sampling distributions are propagated to each subsequent step.

The base SMC sampler used herein is designed to reduce the effect of selecting a $\eta(\boldsymbol{\theta})$ that differs significantly from $\pi(\boldsymbol{\theta}) = p(\boldsymbol{\theta}|y)$. This is accomplished by using what is referred to as a likelihood tempered sequence of target distributions, or cooling schedule. This sequence is defined as

$$\pi_t(\boldsymbol{\theta}) = \frac{p(\boldsymbol{\theta})p(y|\boldsymbol{\theta})^{\phi_t}}{Z_t} = \frac{\gamma_t(\boldsymbol{\theta})}{Z_t}, \tag{3}$$

where $\{\phi_t\}_{t=1}^{T}$ is a non-decreasing sequence with boundaries $\phi_0 = 0$ and $\phi_T = 1$. Defining the sequence in this manner means that the sampler begins by estimating what is typically a known distribution – the parameter prior probability distribution, $p(\boldsymbol{\theta})$ – which can be accomplished by setting $\eta_1(\boldsymbol{\theta}) = p(\boldsymbol{\theta})$. In a monotonic manner, the target distribution is then gradually transitioned to the final target distribution of interest, $\pi_T(\theta)$, which is equal to Equation 2. This transition helps to ensure that, at each sampling step, the importance distribution is similar to the target distribution. As with any Bayesian inference procedure, it is thus important to make a judicious choice of prior distribution, particularly one that encompasses potential regions of high posterior probability density.

Leveraging the existing maturity of particle filters, the sequential importance sampler (SIS) (Doucet et al., 2000; Arulampalam et al., 2002) can be applied to the likelihood tempered sequence. To do so, the sequence of target distributions, $\{\tilde{\pi}_t\}_{t=1}^{T}$, is defined on the path-space which emits $\pi_t(\boldsymbol{\theta})$ as marginals. The dimension of the joint distribution support increases with time; i.e., $\text{supp}(\tilde{\pi}_t) = \Theta^d \times \Theta^d \times \ldots \times \Theta^d = (\Theta^d)^t$. The target distribution in path-space at time t is thus the joint distribution of the parameters at all times along the path,

$$\tilde{\pi}_t(\boldsymbol{\theta}_{1:t}) = \frac{\tilde{\gamma}_t(\boldsymbol{\theta}_{1:t})}{Z_t} = \frac{\gamma_t(\boldsymbol{\theta}_t)}{Z_t} \prod_{k=1}^{t-1} \mathcal{L}_k(\boldsymbol{\theta}_{k+1}, \boldsymbol{\theta}_k), \tag{4}$$

where $1:t$ represents all times along the path up to time t, and $\{\mathcal{L}_k(\boldsymbol{\theta}_{k+1}, \boldsymbol{\theta}_k)\}_{k=1}^{t-1}$ are backward Markov kernels representing the probability density of a backward move from $\boldsymbol{\theta}_{k+1}$ to $\boldsymbol{\theta}_k$.

Recast as a particle method, the estimate of each target distribution π_t is given by

$$\pi_t^N(\boldsymbol{\theta}_t) \approx \sum_{m=1}^{N} \widetilde{W}_t^{(m)} \delta(\boldsymbol{\theta}_t - \boldsymbol{\theta}_t^{(m)}), \tag{5}$$

where the particle system consists of N particles with corresponding normalized weights that form the set $\{\boldsymbol{\theta}_{1:t}^{(m)}, \widetilde{W}_t^{(m)}\}_{m=1}^{N}$ and where δ denotes the Dirac delta function. An estimate of the expectation of any integrable function $\varphi(\cdot)$ with respect to this distribution is

$$\mathbb{E}_{\pi_t^N}[\varphi(\boldsymbol{\theta}_t)] = \sum_{m=1}^{N} \widetilde{W}_t^{(m)} \varphi(\boldsymbol{\theta}_t^{(m)}). \tag{6}$$

The transition to each subsequent target distribution is carried out via recursive importance sampling. In traditional particle filtering, the particles are transitioned to the next target through a two step process: (i) predict the future state using the state-space model, and (ii) correct the prediction by reweighting the particles given new data. The method used here is similar, except that the target being approximated is static, requiring a different approach to transitioning the particles. To propagate the particles from $\tilde{\pi}_t$ to $\tilde{\pi}_{t+1}$, the following proposal distribution is used:

$$\eta_t(\boldsymbol{\theta}_{1:t}^{(m)}) = \eta_1(\boldsymbol{\theta}_1^{(m)}) \prod_{k=2}^{t} \mathcal{K}_k(\boldsymbol{\theta}_{k-1}^{(m)}, \boldsymbol{\theta}_k^{(m)}), \tag{7}$$

where $\{\mathcal{K}_k(\boldsymbol{\theta}_{k-1}^{(m)}, \boldsymbol{\theta}_k^{(m)})\}_{k=2}^{t}$ are forward Markov kernels, representing the probability density of a forward move from $\boldsymbol{\theta}_{t-1}$ to $\boldsymbol{\theta}_t$. From importance sampling, the unnormalized importance weights are

$$W_t^{(m)} \propto \frac{\tilde{\pi}_t(\boldsymbol{\theta}_{1:t}^{(m)})}{\eta_t(\boldsymbol{\theta}_{1:t}^{(m)})} \propto w_t(\boldsymbol{\theta}_{t-1}^{(m)}, \boldsymbol{\theta}_t^{(m)}) W_{t-1}^{(m)}, \tag{8}$$

where w_t is termed the incremental weight function and takes the form

$$w_t(\boldsymbol{\theta}_{t-1}^{(m)}, \boldsymbol{\theta}_t^{(m)}) = \frac{\gamma_t(\boldsymbol{\theta}_t^{(m)})\mathcal{L}_{t-1}(\boldsymbol{\theta}_t^{(m)}, \boldsymbol{\theta}_{t-1}^{(m)})}{\gamma_{t-1}(\boldsymbol{\theta}_{t-1}^{(m)})\mathcal{K}_t(\boldsymbol{\theta}_{t-1}^{(m)}, \boldsymbol{\theta}_t^{(m)})}. \tag{9}$$

The optimal choice for $\mathcal{K}_t(\boldsymbol{\theta}_{t-1}, \boldsymbol{\theta}_t) = \pi_t(\boldsymbol{\theta}_t)$, which is only known up to some proportionality. While computationally intensive, Markov chain Monte Carlo (MCMC) methods are excellent tools for approximating probability distributions of this type. Thus, $\mathcal{K}_t$ is chosen to be a MCMC kernel with invariant target $\pi_t(\boldsymbol{\theta}_t)$. The maturity of MCMC methods in the literature can be leveraged to choose the specific MCMC sampling algorithm that is most appropriate for a given problem.

As shown in (Peters, 2005; Del Moral et al., 2006), the choice of the backward kernel can be arbitrary and still provide asymptotically consistent estimates, but it can also be optimized for performance. Omitting some detail for brevity, if the backward kernel takes the form

$$\mathcal{L}_{t-1}(\boldsymbol{\theta}_t, \boldsymbol{\theta}_{t-1}) = \frac{\pi_t(\boldsymbol{\theta}_{t-1})\mathcal{K}_t(\boldsymbol{\theta}_{t-1}, \boldsymbol{\theta}_t)}{\pi_t(\boldsymbol{\theta}_t)}, \tag{10}$$

then the incremental weight function simplifies to

$$w_t(\boldsymbol{\theta}_{t-1}^{(m)}, \boldsymbol{\theta}_t^{(m)}) = \frac{\gamma_t(\boldsymbol{\theta}_{t-1}^{(m)})}{\gamma_{t-1}(\boldsymbol{\theta}_{t-1}^{(m)})} = p(y|\boldsymbol{\theta}_{t-1}^{(m)})^{\Delta\phi_t}, \tag{11}$$

where $\Delta\phi_t = \phi_t - \phi_{t-1}$. Substituting Equation 11 into Equation 8, a recursive relationship is obtained to update particle weights. Note that the update only depends on the particles from time $t-1$; i.e., those yet to be transitioned via the MCMC forward kernel. In practice, this means that the weights are actually updated prior to the MCMC step.

As with SIS for particle filtering applications, degeneration of the particle population is often unavoidable, especially when the initial proposal distribution, $\eta(\boldsymbol{\theta})$, differs significantly from the target distribution. A degenerate population is one where only a few of the particles have significant weights and the variance of the weights is large. Resampling with replacement can alleviate this issue. A common method for determining when resampling is required is to monitor the equivalent sample size,

$$\mathbb{ESS}_t = \left[\sum_{m=1}^{N} (\widetilde{W}_t^{(m)})^2\right]^{-1}. \tag{12}$$

If this value falls below a user-defined threshold value, $\overline{\mathbb{ESS}}$, then resampling is conducted.

In an effort to increase accessibility for general scientific and engineering applications, Nguyen et al. produced a general algorithm for implementation of the SMC theory presented in this section (Nguyen et al., 2016). Their algorithm was adapted for parallel computing on distributed memory systems herein, as outlined in Algorithm 1. This parallel version was implemented using the `mpi4py` module (L. Dalcín, Paz, & Storti, 2005; L. Dalcín, Paz, Storti, & DElía, 2008; L. D. Dalcín, Paz, Kler, & Cosimo, 2011) in Python 2.7. Since the particles are independent of one another for a given iteration, the steps of the algorithm that involve actual model evaluations – namely those associated with lines 4 and 15 in Algorithm 1 – can be carried out in an embarrassingly parallel fashion across all available processors. While not currently available, an effort will be made post-publication of this conference paper to release an open source version of the software.

Algorithm 1 Parallel SMC Sampler (adapted from (Nguyen et al., 2016))

```
 1: procedure INITIALIZE PARTICLES, $t = 1$:
 2:     Sample $\{\theta_1^{(m)}\}_{m=1}^N \sim \eta_1(\cdot)$.
 3:     Partition $\{\theta_1^{(m)}\}_{m=1}^N$; scatter to available processors.
 4:     Compute $W_1^{(m)} = \frac{p(\theta)p(y|\theta)^{\phi_1}}{\eta_1(\theta_1^{(m)})}$ for all $m$ in parallel.
 5:     Gather unnormalized weights on main processor.
 6:     Compute normalized weights, $\widetilde{W}_1^{(m)} = \frac{W_1^{(m)}}{\sum_{j=1}^N W_1^{(j)}}$.
 7:     Resample if $\mathbb{ESS} < \overline{\mathbb{ESS}}$.
 8: end procedure
 9: procedure ITERATE OVER TARGET SEQUENCE:
10:     for $t = 2, \ldots, T$ do
11:         Compute $W_t^{(m)} = \widetilde{W}_{t-1}^{(m)} p(y|\theta_{t-1}^{(m)})^{\Delta\phi_t}$.
12:         Compute $\widetilde{W}_t^{(m)} = \frac{W_t^{(m)}}{\sum_{j=1}^N W_t^{(j)}}$.
13:         Resample if $\mathbb{ESS} < \overline{\mathbb{ESS}}$.
14:         Partition and scatter $\{\theta_{t-1}^{(m)}\}_{m=1}^N$.
15:         Mutate particles; sample $\theta_t^{(m)} \sim \mathcal{K}_t(\boldsymbol{\theta}_{t-1}^{(m)}; \cdot)$.[1]
16:         Gather $\theta_t^{(m)}$ and $p(y|\theta_t^{(m)})$ on main processor.
17:     end for
18: end procedure
```

As a final note, while not discussed or implemented herein, Nguyen et al. developed methods for choosing an optimal cooling schedule, $\{\phi_t\}_{t=0}^T$ based on a user-defined T. See (Nguyen et al., 2016) for more information.

[1] $\mathcal{K}_t(\cdot; \cdot)$ is an n-step, $\pi_t(\cdot)$-invariant Markov chain Monte Carlo (MCMC) kernel.

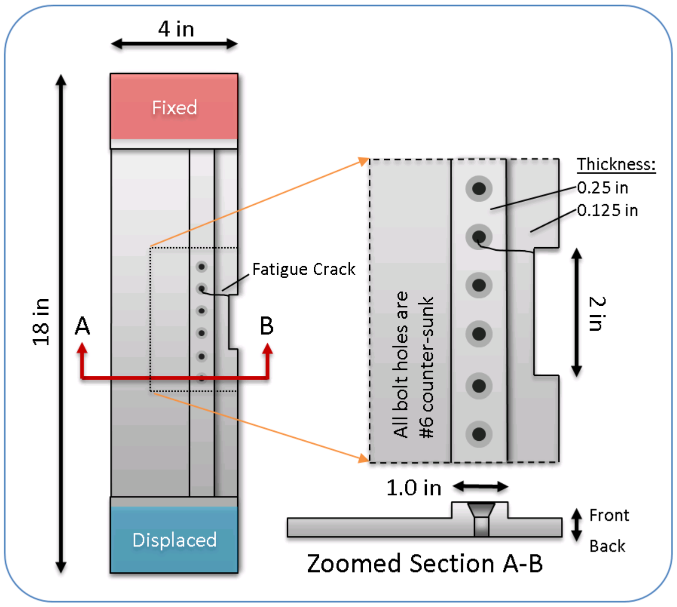

Figure 1. Diagram of the fatigue test specimen.

3. CASE STUDY: TEST SETUP AND MODEL DESCRIPTION

The specimen used in this work was developed as part of NASA's Convergent Aeronautics Solutions (CAS) Digital Twin project. The specimen design incorporated the primary geometric features of an experimental aircraft component that was deemed fatigue-critical. The aircraft was part of the Subsonic Ultra Green Aircraft Research (SUGAR) Project (Bradley, Droney, & Allen, 2015; Bradley & Droney, 2015; Bradley, Allen, & Droney, 2015), a joint effort between Boeing and NASA to develop next-generation aircraft. These features were combined to form the tensile specimen shown in Figure 1, which comprises fourteen potential crack initiation sites – two at each hole and one at each notch fillet. The specimen was intentionally complex, providing an excellent sandbox for testing prognostic algorithms and structural health monitoring techniques. The counter-sunk holes and thickness changes necessitated the types of high-fidelity crack growth models that were the motivation of the present study.

A quasi-random-amplitude load spectrum was applied to the specimen to simulate real-world operation. The load spectrum was formed by generating a series of load blocks, each defined by a randomly selected maximum load and duration, measured in cycles. Within each load block, the fatigue loading was constant amplitude, and the minimum load was defined by a load ratio, R, of 0.1. When implementing the prognostic system described in this paper, it was assumed that the load up to the current cycle was known (i.e., it could be measured with negligible error but future loads were unknown and had to be predicted).

Table 1. Visual measurements of fatigue crack length with added noise. The cycles are displayed as offsets relative to the cycle at which a crack was first observed, which was 2,277,380 cycles.

Relative Cycle	Crack length (in)
+0	0.0523
+3,756	0.0884
+10,932	0.1316
+15,325	0.1392
+20,299	0.2113
+26,456	0.2440
+30,686	0.2784
+34,631	0.2985
+36,439	0.3064
+40,778	0.3885
+45,635	0.3995
+48,991	0.4627
+53,222	0.6117

A fatigue crack initiated at the top fillet and grew into the second hole, as indicated in Figure 1. Two-dimensional measurements of the surface crack tip location were obtained throughout the test using an optical microscope. These measurements were converted into a one-dimensional surface crack length. Gathering visual measurements with the option to manually add noise was preferred for the experiment over other diagnostic methods as it provided flexibility when studying the effect of noise. Tests conducted previously as part of the Digital Twin project (yet to be published) used a strain-based diagnostic approach to measure the crack length with quantified uncertainty. The noise was found to be approximately Gaussian in these tests with average variance, $\nu_{\text{avg}} = 4.83 \times 10^{-4}\text{in}^2$. This variance was used to define the Gaussian white noise added to the visual measurements in the present study; i.e., $\epsilon \sim \mathcal{N}(0, \nu_{\text{avg}})$. The noisy data is included in Table 1.

The damage model used in this work was based on the fracture simulation software FRANC3D ("FRANC3D Reference Manual, Version 7", 2016). The code inserts and grows cracks within an existing finite element model through a local/global remeshing scheme. Although it enables arbitrary, three-dimensional crack growth, FRANC3D is currently limited to linear elastic fracture mechanics (LEFM). Once a crack is inserted, the new mesh is passed to any one of a variety of finite element codes that FRANC3D is compatible with including commercial codes such as ANSYS and ABAQUS and government codes such as SIERRA Mechanics (Sierra Solid Mechanics Team, 2011) and ScIFEN[2] (Warner, Bomarito, Heber, & Hochhalter, 2016). ScIFEN, which is a parallel finite element code developed at NASA Langley Research Center, was used in this work.

In the spirit of the Digital Twin concept, as-built dimensions were used to create a three-dimensional finite element model

[2]Scalable Implementation of Finite Elements by NASA

of the test specimen. Before testing, the specimen was outfitted with an optimized pattern for digital image correlation (DIC); see (Bomarito, Hochhalter, Ruggles, & Cannon, 2017; Bomarito, Hochhalter, & Ruggles, 2017) for more details on the optimization and pattern application. DIC data was obtained during the initial fatigue cycle and the results were used to calibrate the boundary conditions of the model to capture any test stand misalignment or twist during testing.

The specimen geometry and the nature of the applied loading resulted in fourteen highly-localized regions of elevated stress concentration in the specimen, located at the left and right sides of each of the six holes and at each of the two notch fillets. It was assumed that the crack would start at one of these fourteen locations, and, thus, fourteen crack growth simulations were run in advance of testing. The bolt hole cracks were all placed at the shoulder formed between the countersink and the hole. These cracks were semicircular in nature with a radius of 0.02 inches.[3] The cracks inserted at the fillet were through-cracks with a length of 0.02 inches. All initial cracks were planar and oriented perpendicular to the loading direction. The simulated fatigue crack growth was non-planar.

Due to the variability in load amplitude, it was expected that near-threshold driving forces would be observed. Therefore, the NASGRO equation (version 3) (Mettu et al., 1999) was chosen to define the crack growth rate within the FRANC3D simulation,

$$\frac{da}{dN} = C\left[\left(\frac{1-f}{1-R}\right)\Delta K\right]^n \frac{\left(1 - \frac{\Delta K_{th}}{\Delta K}\right)^p}{\left(1 - \frac{K_{max}}{K_C}\right)^q}. \tag{13}$$

Here, da is the infinitesimal crack extension at a particular point along the front, and dN is the infinitesimal number of cycles consumed during the extension. The crack driving force, ΔK, is the difference in stress intensity factor at the maximum and minimum load of a given fatigue cycle; i.e., $\Delta K = K_{\text{max}} - K_{\text{min}}$. The threshold stress intensity factor is denoted as ΔK_{th}. The parameters C, n, p and q are empirical constants, $R = K_{min}/K_{max} = 0.1$ for each load cycle, f is the Newman crack closure term, and K_C is the critical stress intensity factor beyond which unstable crack growth ensues.

At each crack growth step, the displacements in the cracked finite element model are computed by ScIFEN. These displacements are then used by FRANC3D to compute the stress intensity factors along the three-dimensional crack front. The crack front is advanced at the i^{th} point along the discretized crack front based on a user-defined median extension size,

$$\Delta a_i = \Delta a_{\text{median}} \left[\frac{\frac{da}{dN}_i}{\frac{da}{dN}_{\text{median}}}\right], \tag{14}$$

taking into account any out-of-plane kinking due to mode II driving forces and defining $\frac{da}{dN}_{\text{median}}$ as the crack growth rate induced by the median stress intensity factor along the crack front. The user-defined crack growth rate equation (e.g., Equation 13) is then integrated over the computed crack extension to determine the number of cycles consumed by the step. The process is iterated until a failure condition is met.

On average, the FRANC3D-based crack growth simulations took approximately 11 minutes to complete under constant amplitude fatigue loading. For probabilistic parameter estimation, for which most methods typically require thousands to millions of model evaluations, a simulation time of 11 minutes could result in months or even years of analysis. Surrogate modeling or reduced order modeling is commonly used to increase the speed of high-fidelity simulations while maintaining the primary features of the original, full-fidelity model response. Examples of surrogate modeling for high-fidelity fracture simulation can be found in (Li & Lee, 2005; Hombal, Ling, Wolfe, & Mahadevan, 2012; Hombal & Mahadevan, 2013; Leser et al., 2017).

In this work, a fairly simple surrogate approach was adopted. Since the loading direction was invariant throughout testing (i.e., only amplitude was random), it was assumed that a finite set of fourteen potential crack paths existed, $\{\mathcal{C}_i\}_{i=1}^{14}$, corresponding to the aforementioned initiation sites. The median a vs. K_{max} curve was stored for each simulated $\mathcal{C}_i$. Given a known load ratio, R, these curves defined ΔK as a function of crack length in Equation 13.

Assuming the set of potential crack paths was fixed, all remaining uncertainty about fatigue life, the prognostic quantity of interest, was attributed to the parameters describing the linear region of the crack growth rate curve, C and n, and the initial crack length at the zeroth[4] cycle, a_0. These were the random variables to be estimated via Bayesian inference. The remaining model parameters were fixed at the calibrated values found in the NASGRO database. Evaluating the surrogate model thus involved re-integrating Equation 13 given a realization $\theta = [a_0, C, n]^T$ and a crack initiation location. In this particular study, the crack initiation location was known and fixed at the observed initiation coordinates.

4. MARKOV CHAIN MONTE CARLO BENCHMARK

Markov chain Monte Carlo (MCMC) sampling is a widely accepted method for parameter estimation in the case of a single, static posterior distribution (i.e., estimating the in-

[3]Initial crack radius was chosen based on the crack size at which microscale fatigue crack growth simulations were terminated; this multi-scale modeling is not discussed in this paper. See (Leser, 2017) for more detail.

[4]Note that the zeroth cycle is reported as an offset from the cycle at which a crack was first detected, which was 2,277,380 cycles. This definition is consistent with the data in Table 1.

Table 2. Prior distribution definitions

Parameter	a_0 (in)	$\log_{10} C$	n
Distribution	Trunc. Normal	Uniform	Uniform
Mean	5.3×10^{-2}	-	-
Variance	4.8×10^{-4}	-	-
Lower limit	0.0	-50.0	0.0
Upper limit	1.0	0.0	50.0

verse solution to Equation 2). In this study, MCMC sampling served as a benchmark with which to evaluate the performance of the proposed parallel SMC implementation. The comparison was focused on both total simulation time and posterior sample statistics.

To approximate the posterior parameter distribution, the data set shown in Table 1 was used along with the model described in Section 3. The marginal prior distributions are defined in Table 2. It was assumed that these parameters were independent from one another, thus giving a full definition of the joint prior distribution, $p(\boldsymbol{\theta})$. Noninformative distributions were used for $\log_{10} C$ and n, while a truncated normal distribution was used for a_0. The mean of the a_0 prior distribution was set equal to the first-available diagnosis of the crack length, while the variance was assumed to be $1.5\nu_{\text{avg}}$. For simplicity of the demonstration, the measurement noise variance, ν_{avg}, was assumed to be deterministic and known, and was thus fixed during MCMC sampling; i.e., $\nu_{\text{avg}} = 4.83 \times 10^{-4}\text{in}^2$.

The Delayed Rejection Adaptive Metropolis (DRAM) algorithm (Haario, Laine, Mira, & Saksman, 2006) was used to generate a Markov chain with 50,000 samples. The first 10,000 were discarded as burn-in (i.e., the period during which it was assumed the chain had yet to reach a stationary condition and was still exploring the parameter space). For adaptive sampling within the DRAM algorithm, the covariance of the proposal distribution was updated every 1,000 samples based on the current state of the Markov chain. Chain convergence was diagnosed by monitoring the Geweke scores (Geweke et al., 1991). The MCMC sampling took 52,524 seconds, or 14.6 hours, to complete.

In Bayesian prognostics, parameter probability density functions (PDFs) are estimated from MCMC-derived chains using methods such as kernel density estimation. Samples are drawn from the PDFs and propagated through the model via Monte Carlo simulation to provide probabilistic predictions of some quantity of interest (e.g., remaining useful life). This process is typically referred to as uncertainty propagation. In practice, the samples in the chain can be used directly as long as they are independent. Low autocorrelation is commonly used as a measure of sample independence. Sub-sampling, or thinning of the chains, can be conducted when the autocorrelation is high.[5] Depending on the computational expense associated with the forward simulation, the chains may also be sub-sampled to reduce workload.

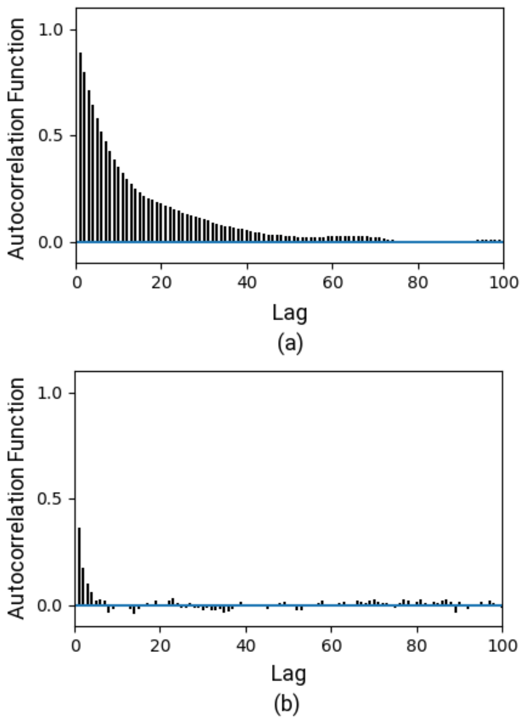

Figure 2. Autocorrelation of the n trace of the parameter chain (a) before thinning and (b) after thinning.

Table 3. Statistics estimated by the MCMC-based sampler.

Parameter	**Mean**	**Variance**
a_0	4.877×10^{-2}	9.752×10^{-5}
$\log_{10} C$	-7.546	0.1925
n	3.005	0.2436

Autocorrelation of the sampled parameters in the present benchmark study was in fact high, as shown in Figure 2 (a). This is typically a sign of sub-optimal mixing. To reduce this autocorrelation and to simulate a sub-sampling for prognostic purposes, the chains were thinned by discarding all but every 10^{th} sample. Parameter estimation results for the thinned chain are shown in Figure 3. The resulting autocorrelation of the chain after thinning is shown to have been reduced significantly in Figure 2 (b). The thinned chains will be used to compare this benchmark with the SMC results in Section 5. Statistics obtained using the thinned chains are shown in Table 3.

[5]See (Link & Eaton, 2012) for more discussion on thinning chains and when it is appropriate in practice.

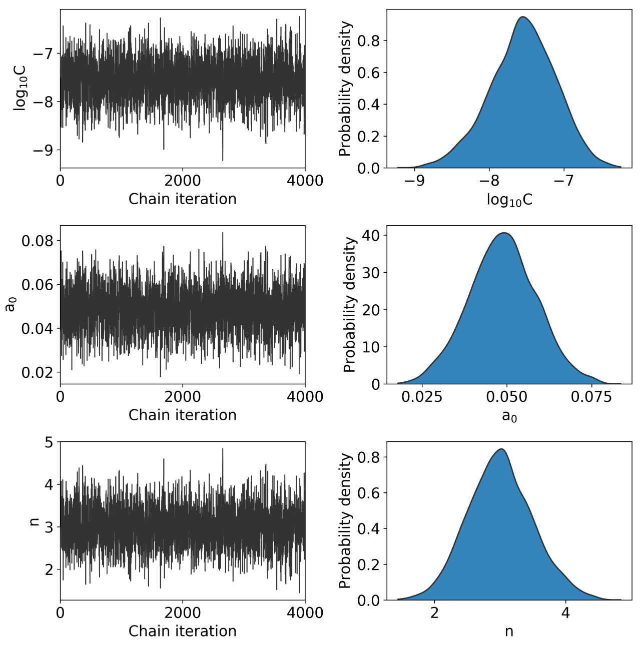

Figure 3. Benchmark multi-dimensional parameter chain sampled via MCMC and the kernel density estimates of the associated marginal PDFs.

5. SMC STATIC PARAMETER ESTIMATION AND COMPARISON WITH BENCHMARK

The goal of the present study was to use the parallel implementation of the sequential Monte Carlo (SMC) methodology presented in Section 2, Algorithm 1, to produce results similar to the benchmark of Section 4. The expected outcome was a parameter estimation process that could match the performance of an MCMC benchmark while expending a fraction of the computation time.

To facilitate a fair comparison of computation time, both the parallel SMC code and the MCMC code were implemented on the Pleiades supercomputer, which is housed at the NASA Advanced Supercomputing (NAS) facility at Ames Research Center. All sampling procedures were executed on Sandy Bridge nodes with two eight-core, 2.6 GHz Intel Xeon E5-2670 processors. The three primary tuning parameters of the SMC sampler were set as shown in Table 4 such that the total number of model evaluations required, $N \times T \times Z = 50,000$ evaluations. This was equivalent to the minimum number of model evaluations required for the MCMC benchmark in the ideal case (i.e., assuming perfect mixing and that all proposed samples were accepted). This assumption was conservative in favor of MCMC. As was the case for the MCMC benchmark, it was assumed that the measurement error $\epsilon \sim \mathcal{N}(0, \nu_{\text{avg}})$ with $\nu_{\text{avg}} = 4.83 \times 10^{-4}$.

When distributing the workload of the SMC sampler on a high-performance computing platform, a theoretical maximum exists on the number of processors used, $\mathcal{P}$, such that $\mathcal{P} \leq N$, the total number of particles. This limit is due to the fact that the parallelization is being carried out over the particle population, meaning processors exceeding this limit would not have a corresponding particle to evaluate. The model evaluation itself could be sped up with additional cores via a nested parallel operation, but this is outside the scope of this work.

Table 4. SMC Tuning Parameters

Tuning Parameter	Value
Number of particles, N	1,000
Number of cooling steps, T	10
Number of MCMC steps, Z	5

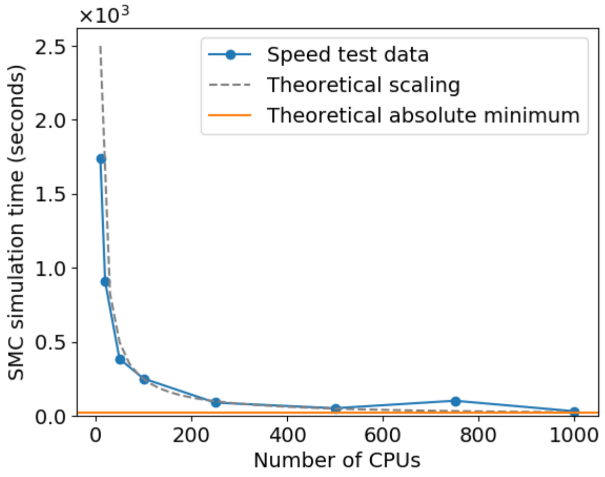

Figure 4. Theoretical and observed computation times for parameter estimation using the parallel SMC sampler of Algorithm 1 and the surrogate model with an increasing number of CPUs.

By assuming an average surrogate model evaluation time of 0.5 seconds, a theoretical scaling curve was be generated. The time to complete an SMC sampling procedure is approximately equal to the product of the evaluation time and the total number of non-parallel model evaluations, as defined by the SMC tuning parameters. The scaling behavior of the parallel algorithm presented in this study was examined by repeating the SMC-based parameter estimation process using the tuning parameters in Table 4 and varying the number of processors used, $\mathcal{P}$. The theoretical scaling and true scaling results are shown in Figure 4. At the maximum number of processors, $\mathcal{P} = 1,000$, the SMC sampling took 32 seconds, three orders of magnitude less than the MCMC benchmark time (52,524 seconds).

Qualitatively, the SMC results can be compared to the benchmark through examination of the joint sample plots in Fig-

Table 5. Statistics estimated by the SMC-based sampler using 1,000 particles.

Parameter	Mean	Variance
a_0	4.798×10^{-2}	5.678×10^{-5}
$\log_{10} C$	-7.492	9.515×10^{-2}
n	2.946	0.1229

Table 6. Percent relative difference (PRD) in estimated statistics (SMC vs. MCMC)

	1,000 particles		4,000 particles	
Parameter	Mean	Variance	Mean	Variance
a_0	1.65%	0.08%	0.59%	0.08%
$\log_{10} C$	0.71%	1.30%	0.73%	1.31%
n	1.98%	4.06%	2.22%	4.11%

ure 5. Correlation and spread appear to be similar between the two sampling methods. The estimated means from each method are plotted as dotted lines, and general agreement is observed.

Quantitatively, the SMC-based estimates of mean and variance are displayed in Table 5, and the percent relative difference (PRD) of those estimates with respect to the MCMC benchmark are shown in Table 6 under the "1,000 particles" heading. The PRDs for mean and variance were calculated according to the equations

$$\text{PRD}_\mu = \frac{2\left|\mu_{\text{SMC}} - \mu_{\text{MCMC}}\right|}{\mu_{\text{SMC}} + \mu_{\text{MCMC}}} \times 100\% \tag{15}$$

and

$$\text{PRD}_{\sigma^2} = \frac{2\left|\sigma^2_{\text{SMC}} - \sigma^2_{\text{MCMC}}\right|}{\mu_{\text{SMC}} + \mu_{\text{MCMC}}} \times 100\%, \tag{16}$$

where, μ and σ^2 represent an estimated mean and variance, respectively, and the subscripts indicate whether the estimator was built using SMC or MCMC sampling. As shown, all of the PRDs are less than 5%, indicating good agreement. It should be noted that, although serving as the benchmark, both methods are approximations of an unknown quantity, and there is no guarantee that the MCMC estimator is any more correct than the SMC estimator in this case. Instead, the focus here was on verifying the SMC approach and comparing performance with a commonly used method.

According to Equation 6, expectations of some function of the model parameters (e.g., the model itself) can be estimated directly using the final particle system, $\{\boldsymbol{\theta}_T^{(m)}, \widetilde{W}_T^{(m)}\}$. This provides a means of predicting expectations of a quantity of interest, such as remaining useful life. Perhaps of more use in fatigue crack growth prognostics is the ability to obtain an estimate of the parameter joint PDF. Propagating the uncertainty represented by this PDF via Monte Carlo sampling enables calculation of credible and prediction intervals,[6] and increases the information available for decision making.

For MCMC sampling, this PDF is typically obtained by generating a sample histogram and using kernel density estimation. In SMC, particles can be resampled with replacement to produce a set of equally likely samples; however, it is not uncommon for the number of particles chosen for SMC to be less than the desired number of samples for histogram construction, as was the case for the 1,000 particle example presented previously. One solution would be to use a stochastic reduced order model for uncertainty propagation with limited samples (Warner, Grigoriu, & Aquino, 2013; Warner, 2018), an approach that will be left for future work. Instead, a brute force approach was adopted here in which the number of particles was increased to the desired sample size and the parallel SMC implementation was used to maintain sampling efficiency, albeit at the expense of more processors.

Using $N = 4,000$ particles (i.e., the same number of samples in the thinned set of MCMC benchmark samples) and $\mathcal{P} = 4,000$ processors, the weighted samples shown in Figure 6 were obtained. Compared to the 1,000 particle system, the variance of the weights in the 4,000 particle system was reduced,[7] which was expected due to the asymptotic convergence properties of SMC estimators as the number of particles $N \to \infty$. However, the PRDs in Table 6 are largely unaffected. Again, it is important to note that the true posterior distribution is unknown and the MCMC and SMC results are approximations.

The sampling was completed in 70.5 seconds, over twice the 32 seconds achieved for the 1,000 particle case. This increase in sampling time may be due to the increased single-processor workload between parallel operations in Algorithm 1, inefficiencies in the implementation of the algorithm, or communication overhead. On the topic of efficient implementation, the gather operations in Algorithm 1 are blocking, meaning that all particle evaluations must be completed before proceeding with the next step in the cooling schedule. This is an inherent source of inefficiency that will be hard to avoid. In this particular case, a larger number of particles meant that the parameter space was explored more thoroughly in the early cooling steps. It is likely that samples were drawn from regions where simulations took longer to complete than the anticipated average simulation time. Regardless, the results of this study suggest that SMC is a valid alternative to MCMC, capable of significantly faster sampling. The parallel SMC sampler can approach real-time capabilities when coupled with rapid modeling techniques such as the surrogate-based approach to high-fidelity modeling presented herein.

[6]Credible intervals are a measure of model fit; prediction intervals are a range of values within which the next observation is expected to reside, with some probability (Smith, 2014).

[7]The variance of particle weights were 1.440×10^{-8} and 1.712×10^{-8} for the 1,000 particle and 4,000 particle samples, respectively.

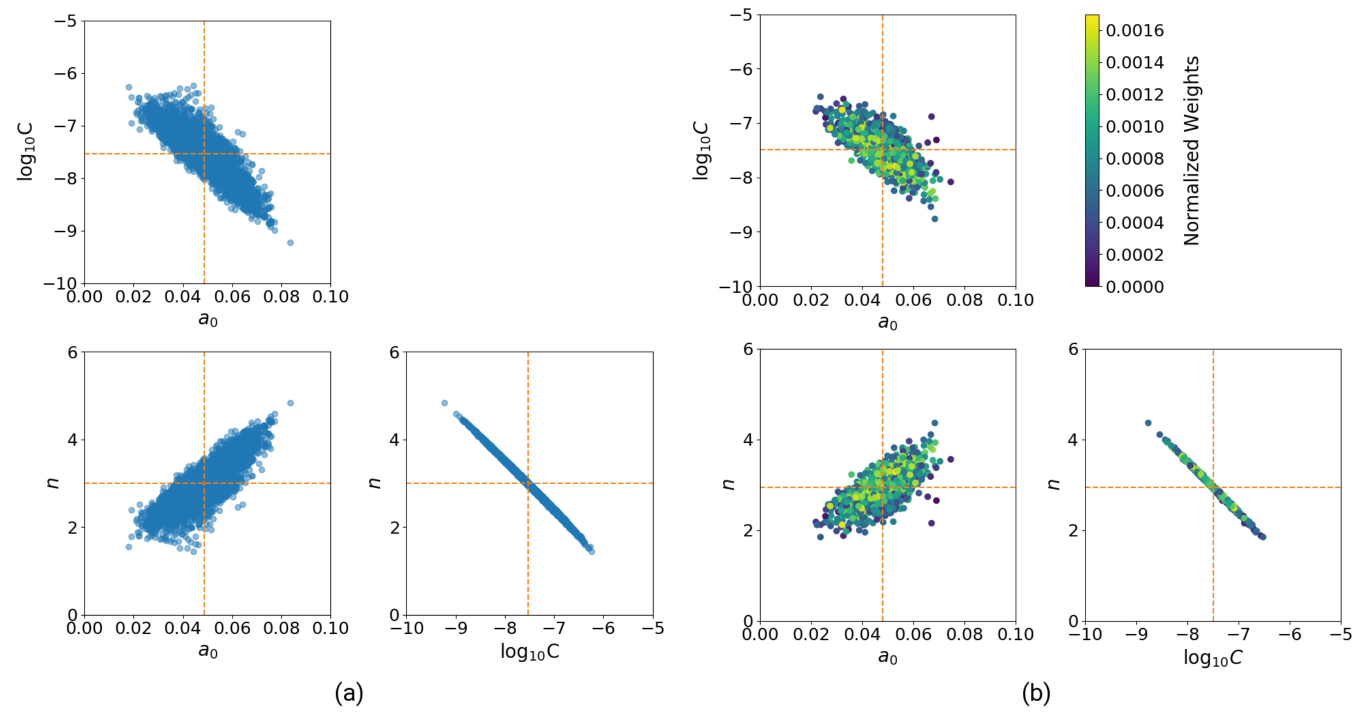

Figure 5. Joint sample points generated from (a) the MCMC benchmark and (b) the SMC samplers (number of processors used, $\mathcal{P} = 1,000$; the dotted lines represent the estimated parameter means.

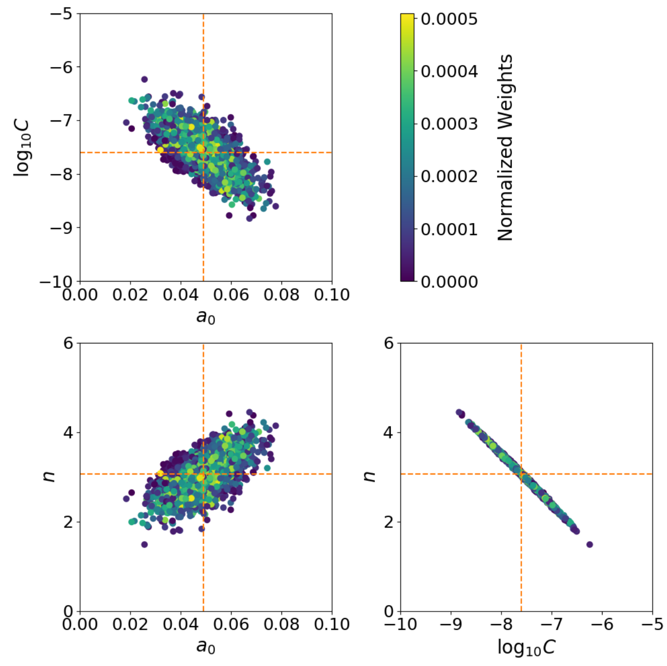

Figure 6. Joint sample points generated from the SMC sampler (number of processors used, $\mathcal{P} = N = 4,000$); the dotted lines represent the estimated parameter means.

6. POTENTIAL OF SMC SAMPLERS FOR FULL-FIDELITY PROGNOSTICS

While not demonstrated in this work, there is significant potential for the proposed parallel SMC implementation to yield full-fidelity prognostics – that is, those that do not require surrogate modeling to achieve computational efficiency. Computation times will certainly increase compared to those reported in Section 5 if using a full-fidelity fatigue crack growth simulator. However, SMC can provide tractability to analyses that are otherwise intractable with MCMC. This is true for high-fidelity fatigue simulations as well as many other applications of interest to the prognostics community.

To further argue this point, a brief thought experiment is presented. The theoretical simulation time shown in Figure 4 was based on the average time required to complete a single, deterministic fatigue crack growth simulation using the surrogate model, which was 0.5 seconds. Replacing this value with the average simulation time required to complete a simulation using the full-fidelity, FRANC3D model, which was 11 minutes on a single processor, provides a reliable prediction of SMC performance. As shown in Figure 7, maximizing the number of processors (i.e., setting $\mathcal{P} = N$) would yield a total parameter estimation time of 9 hours and 10 minutes. In comparison, the approximate minimum time to generate 50,000 samples via MCMC based on the average FRANC3D evaluation time of 11 minutes is, conservatively, 9,167 hours, or 382 days.

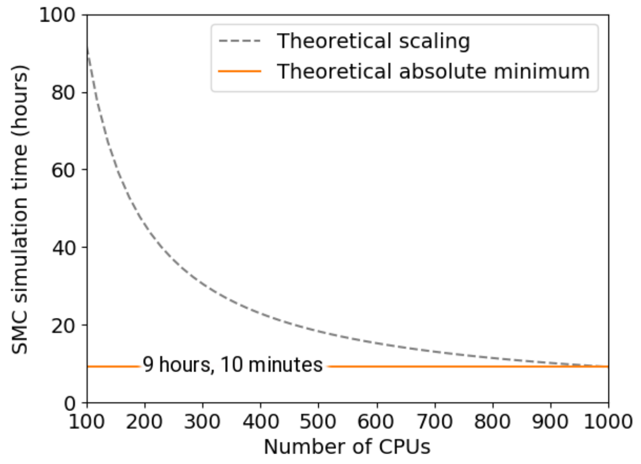

Figure 7. Theoretical computation times for parameter estimation utilizing the parallel SMC sampler of Algorithm 1 and FRANC3D to simulate fatigue crack growth.

7. SUMMARY AND CONCLUSION

A parallel sequential Monte Carlo (SMC) algorithm was developed and implemented using the `mpi4py` module in Python 2.7. The algorithm was applied to prognosis of crack growth in a geometrically complex, metallic specimen subjected to variable amplitude fatigue loading. A surrogate model was developed for crack growth simulation, and the uncertainty in model parameters was quantified using the parallel SMC sampler. The results were compared to parameters estimated via Markov chain Monte Carlo (MCMC) methods for verification and performance evaluation. The SMC sampler provided estimates of mean and variance within a relative percent difference of 5% when compared to the MCMC results and decreased total computation time by three orders of magnitude.

Uncertainty propagation was not explicitly demonstrated in this work. Future work should demonstrate methods for propagating parameter uncertainty obtained from SMC static sampling. Some methods worth investigating include: (i) Monte Carlo sampling using a brute force approach where the number of particles is set equal to the desired number of samples and (ii) smart Monte Carlo methods such as stochastic reduced order modeling (SROM). Efforts will also be made to release an open source version of the parallel SMC code used in this study and to implement some of the other self-tuning methods available in the literature.

In conclusion, parallel SMC sampling was shown to be a legitimate tool for real-time prognostics and health management (PHM) when coupled with a rapid model. For applications requiring expensive high-fidelity models, SMC can be coupled with a surrogate modeling approach to maintain this real-time capability. Based on preliminary study, the parallel SMC framework could also enable parameter estimation for the full-fidelity models as well, without the need for a surrogate. This is in contrast to MCMC sampling, which would be intractable in many of these cases. Based on these benefits and the flexibility of the algorithm, parallel SMC samplers have the potential to make a lasting impact on the field of PHM and could be instrumental in accelerating the mainstream adoption of PHM methodologies.

REFERENCES

Andrieu, C., De Freitas, N., Doucet, A., & Jordan, M. I. (2003). An introduction to MCMC for machine learning. *Machine learning*, *50*(1-2), 5–43.

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

Bomarito, G., Hochhalter, J., & Ruggles, T. (2017). Development of optimal multiscale patterns for digital image correlation via local grayscale variation. *Experimental Mechanics*, 1–12.

Bomarito, G., Hochhalter, J., Ruggles, T., & Cannon, A. (2017). Increasing accuracy and precision of digital image correlation through pattern optimization. *Optics and Lasers in Engineering*, *91*, 73–85.

Bradley, M. K., Allen, T. J., & Droney, C. K. (2015). *Subsonic Ultra Green Aircraft Research: Phase II – Volume III – Truss Braced Wing Aeroelastic Test Report* (Tech. Rep.). NASA/CR-2015-218704/Volume III.

Bradley, M. K., & Droney, C. K. (2015). *Subsonic Ultra Green Aircraft Research: Phase II – Volume II – Hybrid Electric Design Exploration* (Tech. Rep.). NASA/CR-2015-218704/Volume II.

Bradley, M. K., Droney, C. K., & Allen, T. J. (2015). *Subsonic Ultra Green Aircraft Research: Phase II – Volume I – Truss Braced Wing Design Exploration* (Tech. Rep.). NASA/CR-2015-218704/Volume I.

Cadini, F., Zio, E., & Avram, D. (2009). Monte Carlo-based filtering for fatigue crack growth estimation. *Probabilistic Engineering Mechanics*, *24*(3), 367–373.

Corbetta, M., Sbarufatti, C., Manes, A., & Giglio, M. (2015). Real-time prognosis of crack growth evolution using sequential Monte Carlo methods and statistical model parameters. *IEEE Transactions on Reliability*, *64*(2), 736–753.

Daigle, M. J., & Goebel, K. (2011). A model-based prognostics approach applied to pneumatic valves. *International journal of prognostics and health management*, *2*(2), 84–99.

Dalcín, L., Paz, R., & Storti, M. (2005). MPI for Python. *Journal of Parallel and Distributed Computing*, *65*(9), 1108–1115.

Dalcín, L., Paz, R., Storti, M., & DElía, J. (2008). MPI for

Python: Performance improvements and MPI-2 extensions. *Journal of Parallel and Distributed Computing*, *68*(5), 655–662.

Dalcín, L. D., Paz, R. R., Kler, P. A., & Cosimo, A. (2011). Parallel distributed computing using Python. *Advances in Water Resources*, *34*(9), 1124–1139.

Del Moral, P., Doucet, A., & Jasra, A. (2006). Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *68*(3), 411–436.

Doucet, A., Godsill, S., & Andrieu, C. (2000). On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and computing*, *10*(3), 197–208.

FRANC3D Reference Manual, Version 7 [Computer software manual]. (2016).

Geweke, J., et al. (1991). *Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments* (Vol. 196). Federal Reserve Bank of Minneapolis, MN, USA.

Haario, H., Laine, M., Mira, A., & Saksman, E. (2006). DRAM: efficient adaptive MCMC. *Statistics and Computing*, *16*(4), 339–354.

Hombal, V., Ling, Y., Wolfe, K., & Mahadevan, S. (2012). Two-stage planar approximation of non-planar crack growth. *Engineering Fracture Mechanics*, *96*, 147–164.

Hombal, V., & Mahadevan, S. (2013). Surrogate modeling of 3D crack growth. *International Journal of Fatigue*, *47*, 90–99.

Kaipio, J., & Somersalo, E. (2006). *Statistical and computational inverse problems*. Springer Science & Business Media.

Leser, P. E. (2017). *Probabilistic prognostics and health management for fatigue-critical components using high-fidelity models* (PhD dissertation). Dept. of Mechanical and Aerospace Engineering, North Carolina State University.

Leser, P. E., Hochhalter, J. D., Warner, J. E., Newman, J. A., Leser, W. P., Wawrzynek, P. A., & Yuan, F.-G. (2017). Probabilistic fatigue damage prognosis using surrogate models trained via three-dimensional finite element analysis. *Structural Health Monitoring*, *16*(3), 291–308.

Li, C. J., & Lee, H. (2005). Gear fatigue crack prognosis using embedded model, gear dynamic model and fracture mechanics. *Mechanical systems and signal processing*, *19*(4), 836–846.

Link, W. A., & Eaton, M. J. (2012). On thinning of chains in MCMC. *Methods in ecology and evolution*, *3*(1), 112–115.

Liu, J., & West, M. (2001). Combined parameter and state estimation in simulation-based filtering. In *Sequential Monte Carlo methods in practice* (pp. 197–223). Springer.

Mettu, S., Shivakumar, V., Beek, J., Yeh, F., Williams, L., Forman, R., … Newman Jr, J. (1999). Nasgro 3.0: A software for analyzing aging aircraft.

Nguyen, T. L. T., Septier, F., Peters, G. W., & Delignon, Y. (2016). Efficient sequential Monte-Carlo samplers for Bayesian inference. *IEEE Transactions on Signal Processing*, *64*(5), 1305–1319.

Orchard, M. E., & Vachtsevanos, G. J. (2009). A particle-filtering approach for on-line fault diagnosis and failure prognosis. *Transactions of the Institute of Measurement and Control*, *31*(3-4), 221–246.

Peters, G. W. (2005). *Topics in sequential Monte Carlo samplers* (M.Sc. thesis). Dept. of Engineering, University of Cambridge.

Peters, G. W., Fan, Y., & Sisson, S. A. (2012). On sequential Monte Carlo, partial rejection control and approximate Bayesian computation. *Statistics and Computing*, *22*(6), 1209–1222.

Saha, B., & Goebel, K. (2009). Modeling Li-ion battery capacity depletion in a particle filtering framework. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society* (pp. 2909–2924). New York: Prognostics and Health Management Society.

Sierra Solid Mechanics Team. (2011). Sierra/Solid Mechanics 4.22 Users Guide [Computer software manual].

Smith, R. C. (2014). *Uncertainty quantification: theory, implementation, and applications*. SIAM.

Theodoridis, S. (2015). *Machine learning: a Bayesian and optimization perspective*. Academic Press.

Warner, J. E. (2018). *Stochastic Reduced Order Models with Python (SROMPy)* (Tech. Rep.). NASA/TM-2018-219824.

Warner, J. E., Bomarito, G. F., Heber, G., & Hochhalter, J. D. (2016). *Scalable implementation of finite elements by NASA _ Implicit (ScIFEi)* (Tech. Rep.). NASA/TM-2016-219180.

Warner, J. E., Grigoriu, M., & Aquino, W. (2013). Stochastic reduced order models for random vectors: Application to random eigenvalue problems. *Probabilistic Engineering Mechanics*, *31*, 1–11.

Zio, E., & Peloni, G. (2011). Particle filtering prognostic estimation of the remaining useful life of nonlinear components. *Reliability Engineering & System Safety*, *96*(3), 403–409.