PFsuper: Simulation-Based Prognostics to Monitor and Predict Sparse Time Series

Javier Echauz[1], Andrew Gardner[2], Ryan R. Curtin[3], Nikolaos Vasiloglou[4], and George Vachtsevanos[5]

[1-4]*Symantec Corporation, Atlanta, GA,30328, USA*
{Javier_Echauz, Andrew_Gardner, Ryan_Curtin, Nikolaos_Vasiloglou}@symantec.com

[5]*Georgia Institute of Technology, Atlanta, GA, 30332, USA*
gjv@ece.gatech.edu

ABSTRACT

Commercial systems for predicting remaining useful life (RUL) of serviceable parts like engine oil tend to use either generic regression models (practical, e.g., widely deployed in the automotive industry), or dynamic models for which software lags behind theory (impractical, 'one-trick' hardcodings, etc.). We describe an arguably more realistic framework using both generic and vehicle-specific dynamic models of time-series for simulation-based condition monitoring and RUL forecasting, suitable in situations where: (a) measured time-series are sparse or slowly sampled, and (b) health condition signals tend to follow relatively simple paths (low-degree polynomial stationary trends, unit-root stochastic trends, exponential growths, quasiperiodic oscillations). This combination unlocks affordability of PFsuper, a CPU/GPU-intensive prognostics algorithm that implements online Bayesian learning with particle filters to jointly estimate hidden condition state and optionally a handful of unknown parameters, coupled with subsimulations characterizing failure progression and RUL probability density function. The overall method converts a generic static time-as-a-regressor model into stochastic differential, then has PFsuper adapt the initally generic model into a vehicle-specific one as data measurements arrive.

1. INTRODUCTION

The existing patented and commercialized systems for predicting oil remaining life, such as GM's Oil-Life System, Daimler's FSS, and others (Hitch, 2015; Di et al., 2013; Discenzo, 2009), monitor known correlates of oil degradation (without oil parameter sensors) including engine revolutions and temperature, to suggest time and distance to next oil change. These are reasonable regression models; however the input-output relationship is static (no dynamics equations), and is one-vehicle-group-fits-all (i.e., there is no knowledge of the specific oil specimen in a vehicle). Zhu et al. (2013) worked extensively on PF-based oil RUL prediction, which is dynamic, but implemented code deviating from their proposed methods. Specifically, their simulations were hardcoded to follow an exponential growth with no provision for online model adaptation, uncertainty management, nonuniform or under-sampled observations, and several other details needed for application in practice.

Our developing technology for oil RUL prediction is summarized below in reference to Fig. 1:

1. *Precompute generic models*. From historical oil sample databases, identify clusters/families for which oil degrades in a coarsely similar way. This could be a function of oil type, machine asset class, and application. For each identified group, precompute a generic, prior model, which is generally a static mapping with time-as-the-regressor. These can obtained from standard curve fitting or symbollic regression packages.
2. *Convert regressions to stochastic dynamic models*. Although the generic model incorporates time, it is still in a 'frozen' form with time-as-the-regressor. Dynamic state-space models are preferred as they encode not just behavior but *rules of behavior*. (For example, if data were samples of a sinusoid over a finite interval, a polynomial fit extrapolates incorrectly, whereas the differential equation $\ddot{x} = -x$ allows prediction indefinitely.) From the generic training data, also use the aggregate regression residuals and vehicles' mean sampling rates to estimate state and output noise levels in the stochastic differential equations. This conversion step remains largely manual, but can be automated in some polynomially-fit cases,

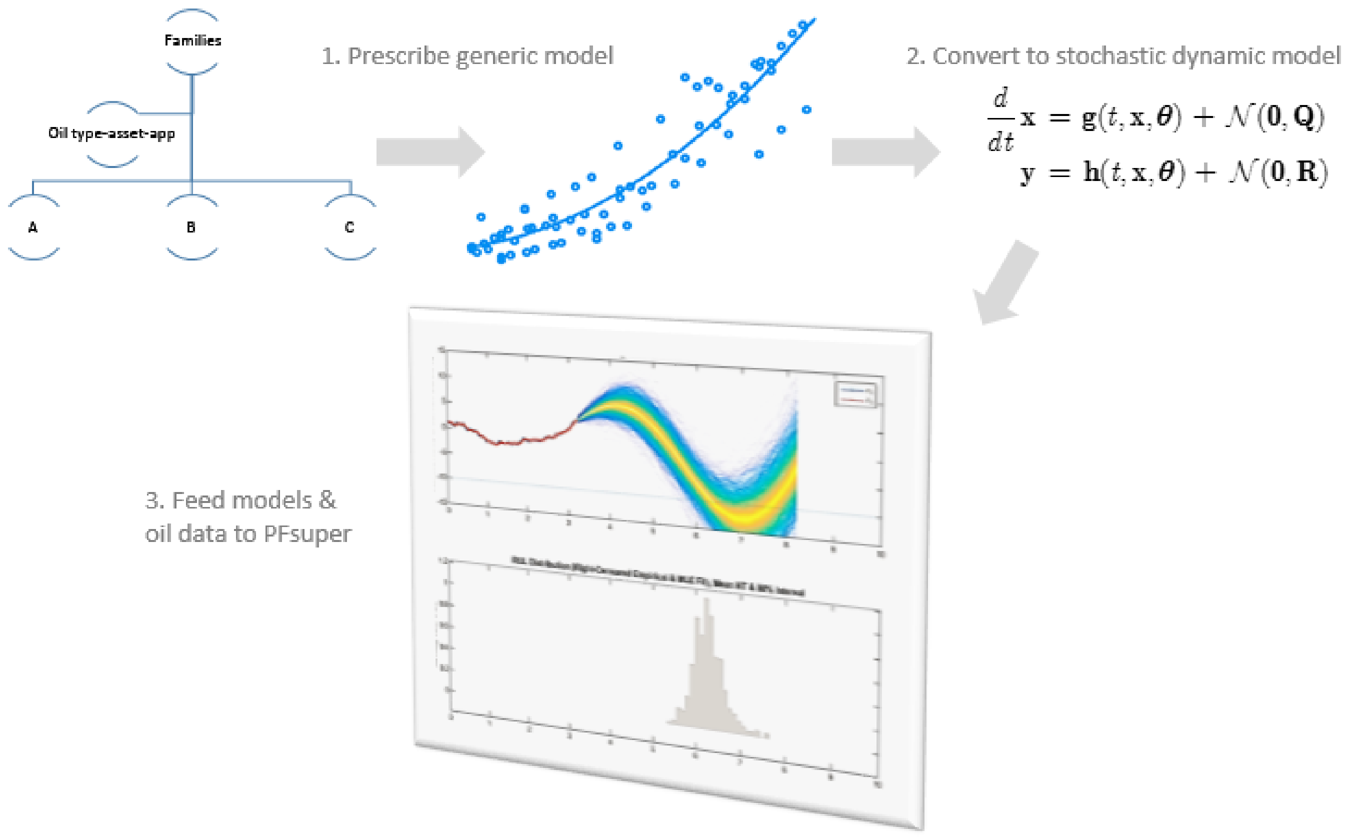
Families
Oil type-asset-app
A
B
C
1. Prescribe generic model
2. Convert to stochastic dynamic model
$\frac{d}{dt}\mathbf{x} = \mathbf{g}(t, \mathbf{x}, \boldsymbol{\theta}) + \mathcal{N}(\mathbf{0}, \mathbf{Q})$
$\mathbf{y} = \mathbf{h}(t, \mathbf{x}, \boldsymbol{\theta}) + \mathcal{N}(\mathbf{0}, \mathbf{R})$
3. Feed models & oil data to PFsuper

Figure 1. Framework for advanced oil prognostics using PFsuper.

by nonseasonal differencing to make the time profile stationary, accounted for by integrators (delays in discrete case) in Jordan-canonical form in the model. Alternatively, up to 2 neural networks can be trained to directly provide discrete-time state and output equations from delay-embedded pseudostates obtained from time-series measurements in the training set.

3. *Perform PFsuper*. Feed oil analysis data to our prognostic program called [**P**]article [**F**]ilter [**S**]tate & [**U**]nknown [**P**]arameter [**E**]stimation plus [**R**]UL, with the family-appropriate generic model as initial guess. As new data measurements $\mathbf{y}$ arrive, PFsuper will: (a) track the expected value of unobserved state $\mathbf{x}$, while simultaneously performing: (b) Bayesian recursive estimation of unknown parameters $\boldsymbol{\theta}$ using parallel populations of PFs, (c) Monte Carlo simulation-based prognosis over a receding prediction horizon, (d) statistical characterization of RUL for given exceedance limits, including pdfs that account for the uncertainty of right-censored exceedance hitting times, and (e) optionally several visualizations: signals and parameter histories, parameter likelihood profiles, animated particles in state space, undersampled data y_i arrivals, RUL distribution in relation to prognosed output, and RUL burndown chart.

Thus, the overall strategy is to start prognosing oil health with the best we know about the oil-asset-application family prior to actual use (i.e,. the selected generic model), then let PFsuper learn a more specific model from the oil samples as they arrive, with the evolving pdfs and concomitant propagated uncertainties handled under principled data science and rational Bayesian formalism. Since the models can just as well be time-varying, our RUL predictions can evolve based on, e.g., parameter drift, sudden change in operating conditions, or incipient failure. We are unaware of prior software implementations getting all these aspects together to actually work. For example, the simultaneous state filtering with parameter estimation is widely considered unsolved in the field due to mixing issues; it had been done insatisfactorily using the ‘state augmentation trick’ (adding parameter as a random-walk state), or nonrecursively (Conrad et al., 2017; Särkkä, 2013). Similarly the censored statistics treatment for RUL prediction horizon appears to be a new contribution in this context.

2. PFSUPER ALGORITHM DESCRIPTION

PFsuper implements a bootstrap particle filter enhanced by a stratified type of resampling that tracks unobserved states **x** of a dynamic system with uncertainties, from causally observed measurements **y**, while simultaneously performing:

- Bayesian recursive estimation of unknown parameters $\boldsymbol{\theta}$ using parallel PFs (one population per $\boldsymbol{\theta}$ gridpoint)
- Monte Carlo simulation-based prognosis over a receding prediction horizon
- Statistical characterization of remaining useful life for given exceedance limits
- Optionally several visualizations: signals and parameter histories, parameter likelihood profiles, animated particles in state space, undersampled data arrivals, RUL distribution in relation to prognosed output, and RUL burndown chart.

Use cases:

- Given a ground-truth model (state **x** and output **y** mappings and noises), generate a realization path for (**x**,**y**) and perform PFsuper on that. This "true" model could be from physics or domain knowledge.
- Given a streaming time-series $\{t_i,y_i\}$ observed at possibly nonuniform or undersampled times, a *putative model* (proxy for "true") that could have generated that data, and an initial model that PF is allowed to use, stochastically interpolate to finer uniform times, recreate a realization for (**x**,**y**) passing thru $\{t_i,y_i\}$, and perform PFsuper on that. The putative model is only used for visualization and stopping; PF has no access to it other than measurements. The implementation is *causal*, so data $\{t_i,y_i\}$ are presented gradually over time as if only when available.

The general time-varying dynamic systems handled are stochastic differential equations converted to discrete-time state-space, possibly inhomogeneous, hidden Markov models of the form

State model:

$$\mathbf{x}_k \sim p(\mathbf{x}_k \mid \mathbf{x}_{k-1},\boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}_k \mid \mathbf{f}_k(\mathbf{x}_{k-1},\boldsymbol{\theta}),\mathbf{Q}_k), \quad \mathbf{x}_0 \sim p(\mathbf{x}_0 \mid \boldsymbol{\theta}), \quad \boldsymbol{\theta}_0 \sim p(\boldsymbol{\theta}) \tag{1}$$

Output model:

$$\mathbf{y}_k \sim p(\mathbf{y}_k \mid \mathbf{x}_k,\boldsymbol{\theta}) = \mathcal{N}(\mathbf{y}_k \mid \mathbf{h}_k(\mathbf{x}_k,\boldsymbol{\theta}),\mathbf{R}_k) \tag{2}$$

where $\mathbf{f}(\cdot)$ is the deterministic part of the state transition map obtained after discretizing the vector field $\mathbf{g}(\cdot)$ in the continuous-time equation of motion

$$\frac{d}{dt}\mathbf{x} = \mathbf{g}(t,\mathbf{x},\boldsymbol{\theta})\,. \tag{3}$$

The deterministic part of transition map becomes

$$\mathbf{f}_k^{\mathrm{E}}(\mathbf{x}_{k-1},\boldsymbol{\theta}) = \mathbf{x}_{k-1} + \mathbf{g}(t_k,\mathbf{x}_{k-1},\boldsymbol{\theta})\,\Delta t \tag{4}$$

with simple forward Euler, or

$$\mathbf{f}_k(\mathbf{x}_{k-1},\boldsymbol{\theta}) = \mathbf{x}_{k-1} + \frac{\mathbf{g}(t_k,\mathbf{x}_{k-1},\boldsymbol{\theta}) + \mathbf{g}(t_k,\mathbf{f}^{\mathrm{E}}(t_k,\mathbf{x}_{k-1},\boldsymbol{\theta}),\boldsymbol{\theta})}{2}\Delta t \tag{5}$$

with Heun 2-stage trapezoidal rule, which reuses Euler as a sort of look-ahead for next state. We implemented Heun for its better stability at a given fixed step size $\Delta t=h$. In software, this means that the next state is computed as

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{G}(t_k,\mathbf{x}_{k-1},\boldsymbol{\theta})\,h + \mathcal{N}(\mathbf{0},\mathbf{Q}) \tag{6}$$

where $\mathbf{G}(\cdot)$ is the improved Heun gradient, and covariance $\mathbf{Q}_k=\mathbf{Q}$ is constant[1]. This makes the state evolution a Brownian motion with possibly nonlinear drift $\mathbf{G}(\cdot)h$ and diffusion rate controlled by **Q**. A noise-free **Q**=0 (along with **R**>0 below) yields a constant-trend stationary process, whereas **Q**>0 yields a stochastic-trend/difference-stationary process. For example, in 1D, if $q=\sigma^2/h$ then the solution x_k for all k will be a random path with time-conditional mean G (the deterministic component) and standard deviation that grows as $\sigma\sqrt{t_k}$, thus the forecast fan can look like a sideways snaking parabola. Similarly, the output equation is implemented as

$$\mathbf{y}_k = \mathbf{h}(t_k,\mathbf{x}_k,\boldsymbol{\theta}) + \mathcal{N}(\mathbf{0},\mathbf{R})\,. \tag{7}$$

A wide class of models including ARIMA with seasonality are subsumed under this framework. Some multiplicative noise processes can still be modeled as above via log transformation that makes noise additive. We used Gaussian noise, but normality is not a strict requirement in the method.

Uncertainties consist of process/state/disturbance noise, measurement/output/innovation noise, and unknown parameters. Unknown parameters are estimated at each k as the mean of the marginal posterior density $p(\boldsymbol{\theta} \mid \mathbf{y}_{1:k})$, which is obtained by accumulating the negative log-likelihood (energy-like) function over a grid of parallel PF populations, each conditioned on specific $\boldsymbol{\theta}$ tuples. The marginal likelihoods $p(\mathbf{y}_{1:k} \mid \boldsymbol{\theta})$ come as a by-product from the unnormalized weights assigned to the particles via the output model at each update. Whereas the particles come as samples from $\mathcal{N}(\mathbf{x}_k \mid \mathbf{f}_k(\mathbf{x}_{k-1},\boldsymbol{\theta}),\mathbf{Q})$, the weights come as samples from the corresponding $\mathcal{N}(\mathbf{y}_k \mid \mathbf{h}_k(\mathbf{x}_k,\boldsymbol{\theta}),\mathbf{R})$. Our method is recursive for online implementation (though possibly slow), handles nonlinearity and time-variance of parameters, and resists local-minima entrapment frequently seen with gradient-based prediction error or MLE methods. Currently only up to 2 unknowns are supported (e.g., 2 physical parameters or quadratic trend coefficients)[2].

The core task for PF is to recursively estimate the filtering distribution $p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$ with the data so far. With that in hand, any expected value of a function of $\mathbf{x}_k$ can be easily estimated, e.g. the expected state $\hat{\mathbf{x}}_k$, or filtered output $\hat{\mathbf{y}}_k$. At time k, the main PF plugs the best estimate of $\boldsymbol{\theta}$ so far into the next-state model $p_k(\mathbf{x}_k \mid \mathbf{x}_{k-1},\boldsymbol{\theta})$, which is used as importance distribution to sample N_p predicted particles $\mathbf{X}_k$. Meanwhile, the measure $\mathbf{y}_k$ is plugged into the output model $p_k(\mathbf{y}_k \mid \mathbf{x}_k,\boldsymbol{\theta})$ to evaluate likelihoods of $\mathbf{y}_k$ given each of those particles, yielding weights $\mathbf{w}=\{w^{(j)}\}$ for each, and thus an updated particle distribution. Continued iteration would quickly degenerate the weights (all but a few =0), so a *re*sampling step is done where particle children $\mathbf{X}^*$ are sampled anew from the updated parents $\{\mathbf{w},\mathbf{X}_k\}$. This tends to make "good" particles become repeated while degenerates are weeded out, but introduces its own impoverishment issue (just a few clones representing the whole distribution).

When data $\{t_i,y_i\}$ are supplied, the updating and resampling steps above happen only at those times. In between data arrivals, our PF and parallel PFs continue iterating only the predictive step with whatever model they had so far. This implements a form of stochastic interpolation that we found to be superior to Brownian bridges (bursty), and thus provides "soft-sensing" in between real sensor readings.

2.1.1. Monte Carlo Forecasting & Failure Prediction

The prognosis routine generates a nested Monte Carlo subsimulation at each time, i.e., an ensemble of iterated prediction steps without updates (as future measuremetns are not available) over a receding prediction horizon of fixed duration T_PH, starting with the last main PF population as initial conditions; see Fig. 2. The resulting shape of the paths is referred to as *forecast fan*. RUL analysis is currently applied by defining exceedance bounds to either the 1st component of output, y_1, or that of the state vector, x_1. More complicated joint constraints representing "useful life" are possible and still an open research question. Each prognosed trajectory is tracked for exceedance and its *hitting time* (HT; a.k.a. first passage time) is recorded. For practical reasons, the prognosis is time-limited by T_PH, so many trajectories within that window may have not crossed threshold yet (or never will), especially during early life, so HTs are statistically right-censored. On any hits, the empirical histogram is derived from the Kaplan-Meier CDF, which gives MLE estimates of survival accounting for censored values. It can be shown that when the drift is a linear trend, the Brownian hitting times will follow an inverse Gaussian (IG) distribution. Under a monotonicity test applied to the forecast fan trend, we perform an MLE fitting of IG to the censored HTs. That makes a big difference in the ability to predict RUL statistics of interest, namely the expected HT and the [5% 50% 95%] HT percentiles, especially when most HTs are still censored during the early portions of simulation.

3. VERIFICATION WITH KNOWN CASES

To verify that the rich dynamic behaviors encapsulated by PFsuper are correct, we show 3 test cases where the "true" system is known by construction: nonlinear pendulum, sudden breakdown, and unstable ramping oscillation.

3.1. Nonlinear Classic Pendulum

The unforced pendulum has state equations

$$\begin{aligned}\dot{x}_1 &= x_2\\ \dot{x}_2 &= -\frac{g}{l}\sin x_1 - \frac{k_\text{f}}{ml}x_2\end{aligned} \tag{8}$$

where state x_1=angle, m=mass, l=length, g=gravity acceleration constant, and k_f=coefficient of viscous friction at pivot. We take unity length and mass, and define gravity 9.81 and friction 0.25 as parameters to be identified recursively from measurements

$$y = \sin x_1 + r\mathcal{N}(0,1) \tag{9}$$

where r=0.1. Note parameters are nonlinear with respect to output and this is where many other methods including recursive least squares get stuck in local minima. PFsuper resists such entrapment by sampling the target likelihood function over a grid using parallel PFs. A grid size of 40x40 was found to be reasonable. A small ellipsoid state noise with covariance $\mathbf{Q} = \left[h^3/3 \;\; h^2/2;\;\; h^2/2 \;\; h\right]$ was set for both system and PF. Integration step size was h=.01. We impose no exceedance thresholds to let simulation run its full course over a time span of 10 seconds, with receding prediction horizon 10/3 seconds. Figs. 3-4 show that PFsuper can successfully solve this problem.

[1]Here, time variance is injected thru free scalar t (and possibly varying $\boldsymbol{\theta}$) in state and output mappings, not thru $\mathbf{Q}_k,\mathbf{R}_k$ which are constant matrices.
[2]Up to ~8 unknowns would be feasible but discouraged in PHM due to notoriously unreliable time-series forecasts (e.g., extrapolations from a high-degree polynomial).

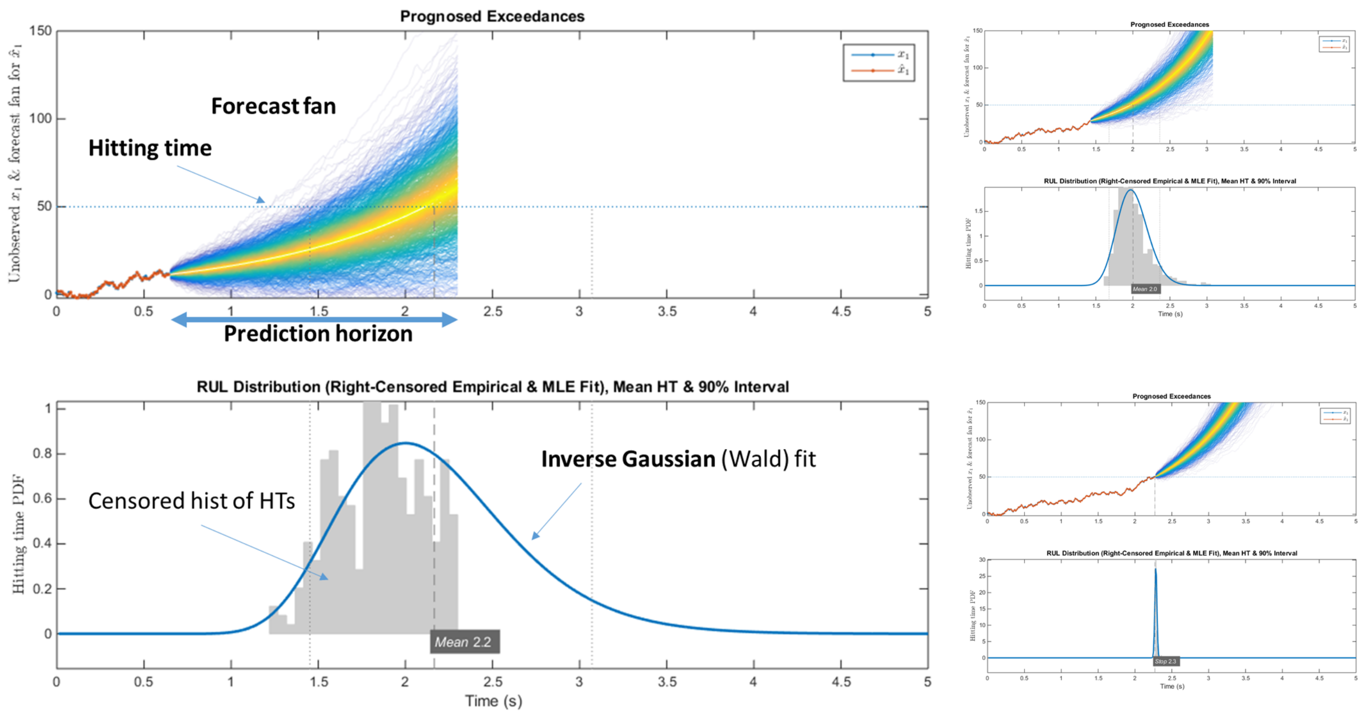

Figure 2. At each time *k*, PFsuper MC-simulates *N* paths forward, conditional on the current particles as initial condition. RUL is inferred from empirical Kaplan-Meier cdf (or its censored hitting-times histogram as shown), plus the MLE fit of inverse Gaussian distribution as pdf whenever the mean of forecast fan is monotonic. Two additional time frames on the right show this pdf continuously revised as more and more data measurements arrive, until complete certainty at failure time.

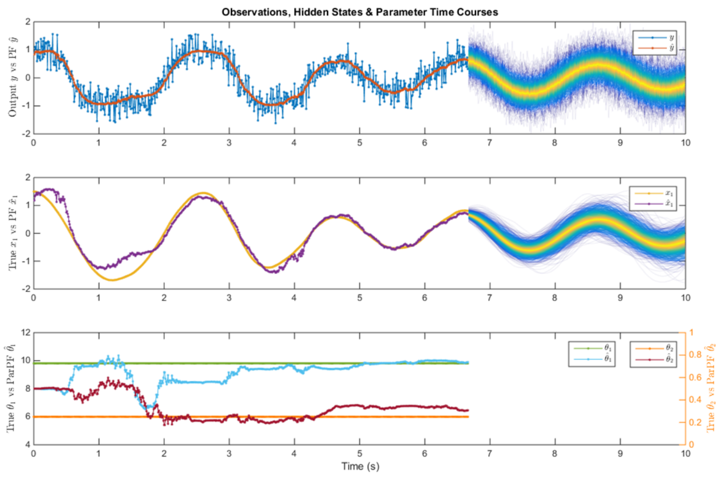

Figure 3. Output y, state x_1, forecast fans, and estimated parameters at 6.67s into the nonlinear pendulum simulation.

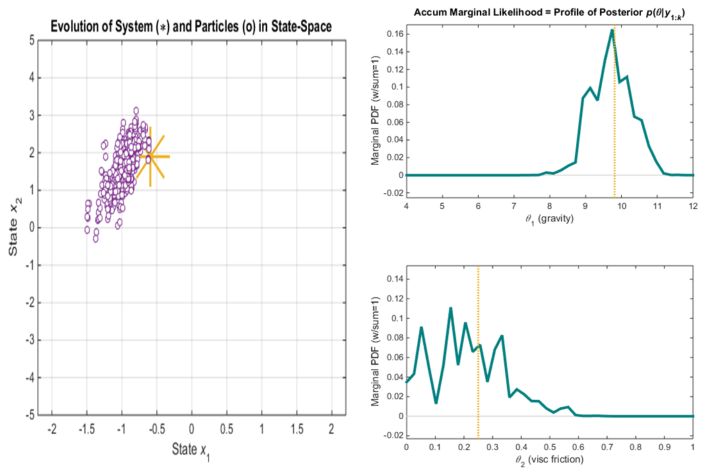

Figure 4. Particle population in state-space and likelihood profiles of the identified parameters at end of the simulation (10s; ground-truth values shown in gold).

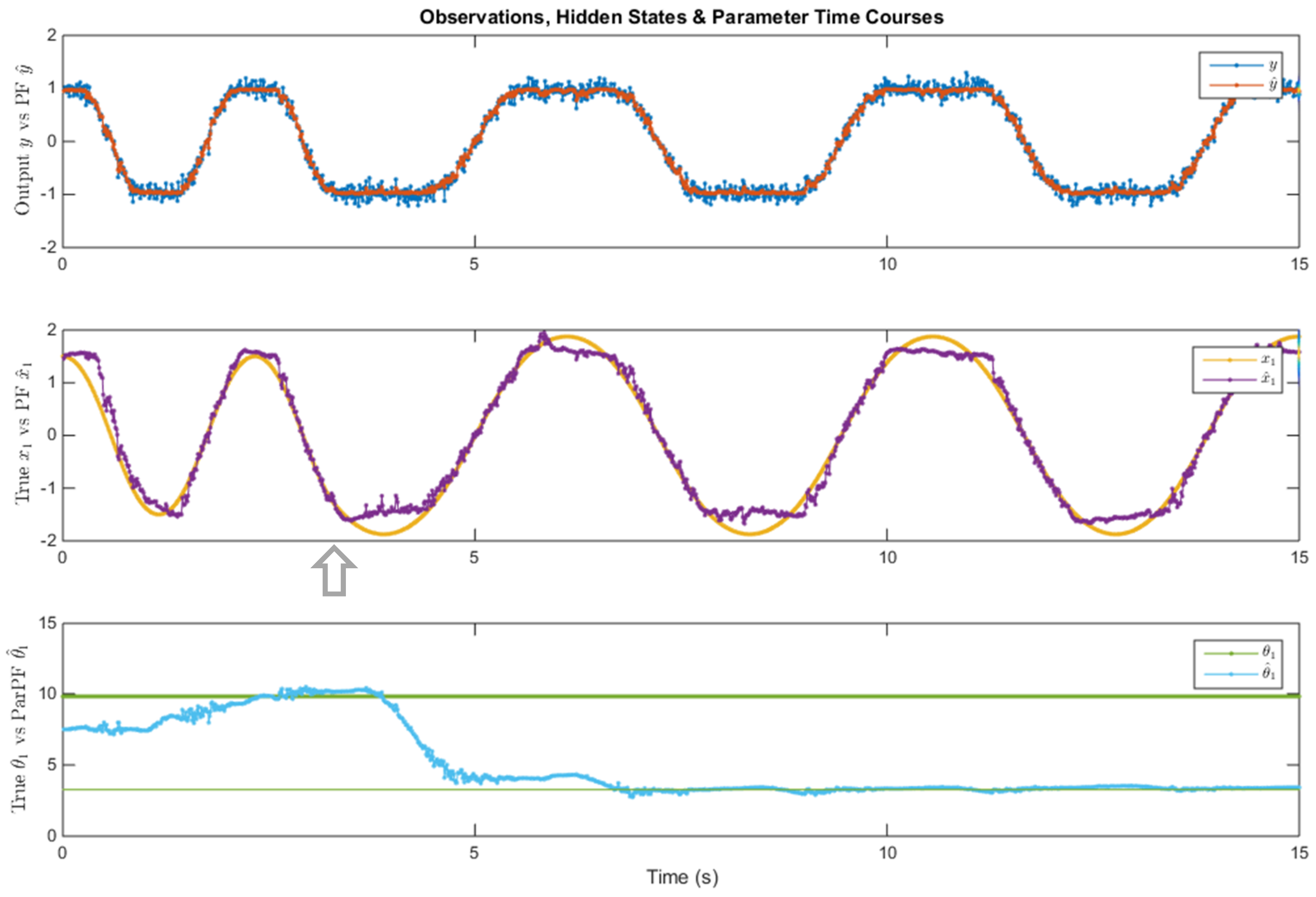

Figure 5. At time t=3.33s, there is a sudden change (unbeknownst to PFsuper) in which gravity was cut by 1/3 (or equivalently, length parameter tripled). By around 5s, the algorithm has started to learn the new value.

3.2. Nonlinear Pendulum with Sudden Breakdown

Using the same pendulum above, we injected a sudden disturbance created by changing g abruptly to 1/3 of its value (3.27) at time 3.33s, or equivalently attaching something so that its length tripled, unbeknownst to PFsuper. This internal breakdown manifests in observed frequency being $\sqrt{3}$ slower. To more clearly separate the effects, we had PFsuper identify a single unknown parameter and ran system with 0 state noise, small spherical state noise $\mathbf{Q} = .01 \times \mathbf{I}_2$ for PF, and r=0.01 shared by both system and PF. Time span was 15s and prediction horizon 5s. Fig. 5 shows the end of the run. We see that PFsuper could adapt to the sudden malfunction within a couple of seconds.

3.3. Unstable Ramping Oscillation

A ramping sinusoid of the form $t \sin at$ can be the response of a linear system with transfer function $H(s) = 2as \,/\, (s^2 + a)^2$, which has a repeated pair of poles at $\pm j\sqrt{a}$. Instead of zero-state response to an impulsive input, we use the zero-input response to some initial condition of the states. A controllable-canonical representation of the state dynamics with a=1 is

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} = \begin{bmatrix} 0 & -2 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, \mathbf{x}(0) = \mathbf{x}_0 \qquad (10)$$

whose explicit solution is $\mathbf{x}_0 e^{\mathbf{A}t}$. We declared the coefficients θ_1 =–2, θ_2 =–1 as unknown parameters for PFsuper to identify recursively, defined a spherical noise $\mathbf{Q} = .01 \times \mathbf{I}_4$ in the state model used by both system and PF, and a measurement noise r=0.5 just for PF's quasi-dummy identity output model, which has to be positive definite for evaluation of likelihoods using the normal pdf. We set 500 particles, prediction horizon 20/3 seconds, and exceedance threshold 10 on state x_1. We defined uniform priors on PF state and parameters, and let the system run from $x_0 = [1\ 1\ 1\ 1]^\top$. A bigger step size h=.1 was used to accelerate simulation but in other problems this can lead to integrator instability. Fig. 6 shows snapshot at end of simulation.

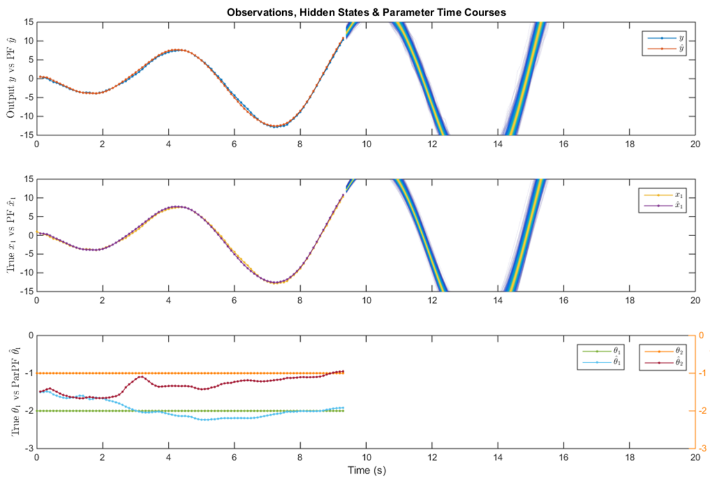

Figure 6. Exceedance occurred on upper rail of x_1 at 9.4s.

4. VALIDATION ON REAL-WORLD TRUCK-FLEET DATA

To test whether our approach meets user needs and intended uses, we performed leave-one-out cross validation (LOOCV) by simulation of strictly causal usage of a military truck-fleet data collection. It had oil analyses of 12 trucks sampled 11 times (not uniformly) over nearly a year. We picked TBN as degradation indicator for its clear pattern, but several of the other oil parameters correlated to it.

The raw TBN data were preprocessed to obtain single oil-drain cycle histories of individual vehicles, then the aggregate scatter plot and generic fit with time-as-the-regressor looked like Fig. 1, step-1 of the general scheme, except in this case it was a linear trend going down. The global fit had parameters θ_1 =slope; θ_0 =intercept with values [–0.0013; 5.9749]. During LOOCV, we do training using all vehicles minus one (simulating a new unseen case), while testing on the missing vehicle in each fold, therefore we will actually have 12 different generic "whole-fleet" fits. For each truck, the generic fit predicts that TBN will hit low threshold c=2.5 at the offset mileage $(c-\theta_0)/\theta_1$, which is a constant. In contrast, each truck monitored by PFsuper will have time-dependent RUL statistics, which are highly variable in the beginning and tend to stabilize towards the end as more oil data samples accumulate. Thus, we will compare the mean absolute error (MAE) between the RUL prediction profile and the actual exceedance mileage for both methods.

In this linear deterioration case, the conversion to stochastic dynamic model (scheme step-2) is as follows. A putative model that could have generated an individual truck's data is

$$\begin{aligned} \dot{x} &= \theta_1, \quad x(0) = \theta_0 \\ y &= x + \sigma \mathcal{N}(0,1) \end{aligned} \tag{11}$$

where the state model has constant-derivative equal to slope θ_1 plus 0 noise, and the output model has identity mapping plus normal noise with standard deviation σ. The state is unobserved (hidden) and only discrete-time samples of the observations at possibly nonuniform times, i.e., data $\{t_i, y_i\}$, are available to PFsuper. A PF also always needs some nonzero "state noise" (as long as it is 0-mean, which preserves expectation) so that it can spread particles without instant degeneracy. Thus, the model that the PF is allowed to "see" is

$$\begin{aligned} \dot{x} &= \hat{\theta}_1 + q\mathcal{N}(0,1), \\ &\quad x(0) \sim \mathcal{U}(y_0 - 3\sigma_{\text{gen}}, y_0 + 3\sigma_{\text{gen}}), \\ &\quad \hat{\theta}_1(0) \sim \mathcal{U}(\theta_1^{\min}, \theta_1^{\max}) \\ y &= x + r\mathcal{N}(0,1) \end{aligned} \tag{12}$$

where $\hat{\theta}_1$ is PFsuper' fluctuating estimate of the unknown slope θ_1, initially drawn from interval seen in the individual trucks' slopes from the database. The prior $p(x_0)$ for the state particles is normal $\mathcal{N}$ or uniform $\mathcal{U}$ but preferably located around mean $\mathbf{h}^{-1}(y_0) = y_0$, the maximum-likelihood estimate of the unknown intercept θ_0.

We couple the magnitude of the output noise to the generic residual: r=σ_{gen} (also learnable online from the data y_i), and that of the state noise to a *diffusion-consistent* scale:

$$q = \sigma_{\text{gen}} \sqrt{h / \tau} \tag{13}$$

where h is the fixed integrator step size and τ is mean intersample interval (1/average sampling rate) for data coming in. This quantity is justified in reference to prognosis: it says that on average, the size to which uncertainty is going to grow while waiting between data updates is of a scale comparable to the underlying σ (recall the spread of Brownian motion grows with $\sqrt{t}$).

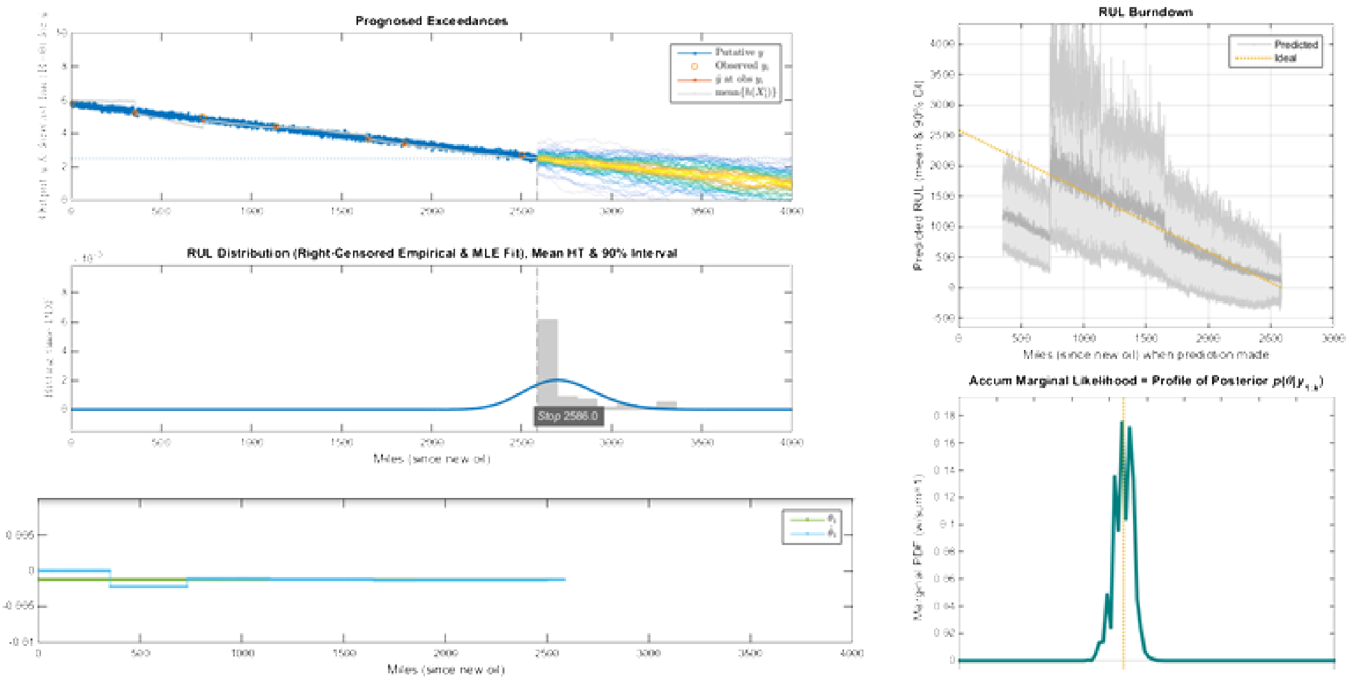

Figure 7. Typical PFsuper simulation run during cross validation (truck #7 shown). Abrupt jumps in estimated θ_1 and RUL burndown are due to the scarcity of data updates (red-orange circles in top panel).

Qualitatively, we want the confidence intervals to grow with longer prediction periods, recognizing propagation of uncertainty into the future. Forecasts with stochastic trends are "safer" (Hyndman & Koehler, 2006) than point estimates.

The simulation was carried out with only 50 particles in the interest of speed, but 500 or 5000 is more common and yields lower variance of the RUL quantiles. Prediction horizon was set at 1500 miles, and exceedance threshold was 2.5 applied to the output (particle-filtered output is used to know when to stop simulation). Fig. 7 shows one of the 12 runs (truck #7) at the end of a LOOCV fold. The truck-specific slope was correctly identified well ahead of the oil lifetime. The RUL burndown shows a typical pattern of unreliable estimates during the beginning until the arrival of at least the first data update. We defined a "break-in" period of 1000 miles across the fleet during which we ignore RUL predictions and exclude them from the MAE performance metric. Alternatively, the method could simply stick to the generic prediction during that initial period.

Fig. 8 shows that in 10 out of 12 cases, dynamic RUL estimates from PFsuper had lower error than the static generic ones. The cross-validation estimate of expected error across the ensemble is 178.86 for PFsuper vs. 358.11 for generic, which suggests the procedure would meet a need in oil RUL prediction, at least with respect to the currently generic methods used in commercial systems.

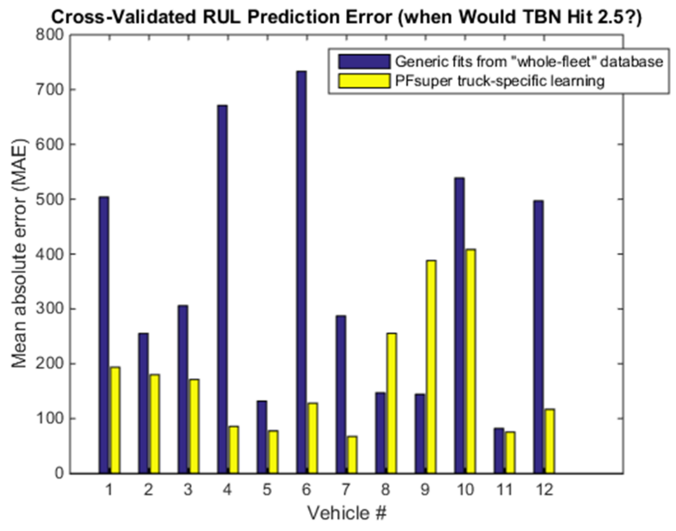

Figure 8. LOO cross-validated mean absolute error between the predicted expected value of RUL and the actual exceedance mileage for static generic fits vs. dynamic PFsuper learning from scarce data updates.

We note that although the approach to online parameter ID via a grid of parallel PF populations is not scalable, it was deemed sufficient in this low-dimensionality PHM application and well worth the cost of avoiding local minima. Additionally, our censored-statistics approach to RUL forecasting 'squeezed out the last drop' of prediction with the inverse Gaussian fit giving the best estimate of mean hitting time before complete data are even available to complete the histogram.

5. CONCLUSION

We have formalized an online learning prognostics framework suitable when time-series data are sparse and health/damage signals tend to follow low-complexity dynamics, as is the case in oil PHM applications. The conversion of generic static models from a historical database into specific dynamic models unlocks the ability to track and adapt to changes in the field. PFsuper attempts to fill prognostic software gaps in online model adaptation, uncertainty management, sparse, nonuniform or under-sampled time-series observations, and other areas needed for reduction to practice. Because the framework is brutally honest about uncertainty propagation, the lack of updates in the context of sparse/slow data makes for wide forecast fans. However, the generic-vs-PFsuper truck-fleet oil validation taught us that even with data samples as information-starved as 11 points per truck (only about once per month), PFsuper offers advantage.

ACKNOWLEDGMENT

We gratefully acknowledge the collaboration of Patrick Henning, CTO at Spectro Scientific, who provided guidance and the truck-fleet validation data.

REFERENCES

Conrad, P., Girolami, M., Särkkä, S., Stuart, A., Zygalakis, K. (2017). Statistical analysis of differential equations: introducing probability measures on numerical solutions. *Statistics and Computing*, vol. 27, no. 4, pp. 1065–1082. doi:10.1007/s11222-016-9671-0.

Di, S., Haijun, W., & Haifeng, L. (2013). Recent Patents on Oil Analysis Technologies of Mechanical Equipment. *Recent Patents on Mechanical Engineering*, vol. 6, no. 1, pp. 11-25.

Discenzo, F.M., Chung, D., Kendig, M.W., Loparo, K.A. (2009). Intelligent fluid sensor for machinery diagnostics, prognostics, and control. US7581434.

Hitch, J. (2015) Determining Proper Oil and Filter Change Intervals: Can Onboard Automotive Sensors Help? *Machinery Lubrication*.Noria publications.

Hyndman, R. J. & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, vol. 22, pp. 679-688. doi:10.1016/j.ijforecast.2006.03.001

Särkkä, S. (2013). *Bayesian Filtering and Smoothing*. Cambridge University Press.

Zhu, J., Yoon, J., He, D., Qu, Y., & Bechhoefer, E. (2013). Lubrication Oil Condition Monitoring and Remaining Useful Life Prediction with Particle Filtering. *International Journal of Prognostics and Health Management*, ISSN 2153-2648, 020.

BIOGRAPHIES

Javier Echauz is a Technical Director at Symantec Corporation. He obtained a PhD in ECE/BioEng from the Georgia Institute of Technology.

Andrew B. Gardner is a Senior Technical Director at Symantec Corporation. He obtained a PhD in ECE from the Georgia Institute of Technology.

Ryan R. Curtin is a Principal Research Scientist at Symantec Corporation. He wanted to be an astronaut, but instead obtained a PhD in ECE from the Georgia Institute of Technology.

Nikolaos Vasiloglou is a Technical Director at Symantec Corporation. He obtained a PhD in ECE from the Georgia Institute of Technology.

George Vachtsevanos is Professor Emeritus of Electrical and Computer Engineering at the Georgia Institute of Technology.