On the Use of Particle Flow to Enhance the Computational Performance of Particle-Filtering-based Prognostics

Javier A. Oliva[1], Torsten Bertram[2]

[1,2] *Institute of Control Theory and Systems Engineering, Technische Universität Dortmund, Germany*
javier.oliva@tu-dortmund.de
torsten.bertram@tu-dortmund.de

ABSTRACT

Prognostic approaches based on particle filtering employ physical models in order to estimate the remaining useful life (RUL) of systems. To this aim a set of particles is used to first estimate the degradation state of the system and then to predict the distribution of the RUL through simulation. The computational complexity of this approach is a function of the number of particles used in the state estimation and of the time each particle needs to simulate the RUL. It is therefore clear that enhancing the computational performance of this approach requires reducing the number of particles. In this paper we investigate the applicability and suitability of the particle flow particle filter for particle-filtering-based prognostics. The estimation of the remaining driving range (RDR) of an electric vehicle is used as the case study to illustrate the improvement in computational performance of the proposed approach in comparison to the standard particle filter.

1. INTRODUCTION

Model-based prognostic approaches have gained in importance during the last decade due to their versatility and ease of implementation in practical engineering applications. From the methodologies available in the literature, a model-based framework using particle filters (PF) has emerged as a solid solution for many prognostics applications. Particle-filtering based approaches for prognostics employ physics-based models in order to estimate the remaining useful life (RUL) of systems or components. To this aim a set of discrete weighted samples, known as particles, is used to first estimate the degradation state of the system or component and then to predict a distribution of the RUL by propagating the set of particles forward in time through simulation until an established failure threshold is reached. The computational complexity of this approach is a function of the number of particles used in the state estimation and of the time each particle needs to sim-

ulate the RUL. It is therefore clear that enhancing the computational performance of this approach requires minimizing the number of particles used without sacrificing the accuracy of both the estimation of the degradation state and the prediction of the RUL distribution. An approach that aims to solve this issue is introduced by (Daigle & Goebel, 2010). This approach is based on the Unscented Transform (UT) (Julier & Uhlmann, 2004), in which the particles are chosen deterministically instead of using a random sampling method. Although this method is more computationally efficient than standard particle filters, the UT may only be applied to nonlinear systems where all sources of noise are Gaussian; otherwise this approach should not be used. In this paper we investigate the use and the suitability of a well known variation of the particle filter based on particle flow and optimal transport methods. The main idea behind this approach is to reduce the number of particles needed in the particle filter by introducing a particle flow, in which the particles are progressively transported without needing to randomly sample from any distribution. This allows us to optimally move the particles to the correct locations according to the Bayes' rule, reducing in this way the number of particles needed and thereby the computational effort in both the estimation and the prediction step. To the best of our knowledge the present study is the first in applying the the particle flow particle filter in model-based prognostics. This paper evaluates the use of the particle flow, which until now has been just investigated in filtering problems of nonlinear systems (Daum & Huang, 2008), with the aim of presenting a computationally efficient alternative to state of the art simulation-based approaches, namely UKF (Daigle & Goebel, 2010) and PF (Orchard & Vachtsevanos, 2010) based approaches, for reducing the number of simulations and therefore the simulation time in the prediction step of model-based prognostics. We use the remaining driving range (RDR) estimation of an electric vehicle (Oliva, Weihrauch, & Bertram, 2013) as the case of study for illustrating and validating the enhancement in the computational performance of the presented approach in comparison to the standard particle filter.

The remainder of this paper is organized as follows. Section 2 formulates the RUL estimation problem in the context of particle filters. Section 3 explains in detail the theoretical foundations of the particle flow particle filter (PFPF) and afterwards presents the steps needed for its implementation within the prognostics framework presented in section 2. In section 4 the case of study used for validating the proposed approach is described. Section 5 presents the experimental and simulation results. Finally, section 6 concludes the findings of this work and provides an outlook on our future work.

2. PARTICLE-FILTERING BASED RUL ESTIMATION

This section is concerned with formulating the RUL estimation problem and briefly explains the particle-filtering-based framework for prognostics employed in this work.

2.1. Problem Statement

Consider the following nonlinear system represented, in a discrete-time form by

$$\begin{aligned} \mathbf{x}_k &= \mathbf{f}\left(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{v}_k, \mathbf{w}_k\right) \\ \mathbf{y}_k &= \mathbf{h}\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{n}_k, \mathbf{w}_k\right), \end{aligned} \quad (1)$$

where $\mathbf{x}_k$ is the state vector, $\mathbf{w}_k$ is the parameter vector, $\mathbf{v}_k$ is the process noise vector, $\mathbf{u}_k$ is the input vector, $\mathbf{y}_k$ is the output vector and $\mathbf{n}_k$ is the measurement noise vector. The terms $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ stand for the state and output function, respectively. The system exhibits a degradation which accumulates in time until a deterministic degradation threshold $\boldsymbol{T}(\mathbf{x})$ is reached, at which the system fails. The degradation of the system is attributed to the environment and to the operation conditions. The RUL estimation problem is concerned with first estimating the degradation state of the system and then to predict its future operation conditions in order to determine the distribution of the time at which the performance of the system fails to fulfill its tasks, i.e. the time at which the threshold is exceeded. Thus, $\boldsymbol{T}(\mathbf{x}) = 1$ if the system fails and $\boldsymbol{T}(\mathbf{x}) = 0$, otherwise. The RUL is a random variable that is influenced by many sources of uncertainty. The lack of knowledge about the state variables, the noise presented in the measurements or the randomness of the operation environment, are some of the factors that largely contribute to the uncertainty of the RUL. Therefore, properly predicting the RUL requires accounting for these sources of uncertainty. In the context of particle filters the RUL estimation proceeds basically in two phases, namely the *state estimation* (I) and the RUL *prediction* (II), as shown in Fig. 1. For the sake of clarity, Fig. 1 depicts the RUL estimation of just one particle.

In the first phase the PF recursively approximates the posterior probability $p(\mathbf{x}_k|\mathbf{Y}_k)$ of the state variables by a set of N_x weighted particles $\mathbf{S}_k = \left\{\mathbf{x}_k^i, w_k^i\right\}_{i=1}^{N_x}$. Here $\mathbf{x}_k^i$ is the set of particles representing the state space, w_k^i are the associated importance weights and $\mathbf{Y}_k = \mathbf{y}_{0:k}$ is the set of all mea-

surements done until time k. Each particle is sampled from an *a priori* estimation of the state space and it is propagated through the function $\mathbf{f}(\cdot)$ in the *prediction step*. Then, the value of each particle is updated from measurements through the output function $\mathbf{h}(\cdot)$ in the *measurement update step*. In this step the weight of each particle is updated according to the likelihood of a new measurement given the particle. Afterwards the *resampling step* occurs. The idea behind this step is to duplicate those particles with large weights and to eliminate those with small weights.
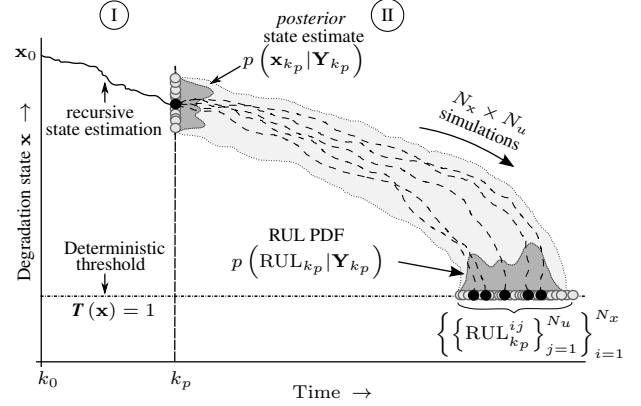

Figure 1. Particle-filtering based RUL estimation approach.

In this way the so called *particle degeneracy* (Daum & Huang, 2011) can be overcome. Particle degeneracy, i.e the situation in which all but few particles have negligible weights leads to a poor approximation of the state variables and, since most weights are close to zero, valuable computational effort is wasted by updating insignificant particles. Finally, the probability distribution of the state variables at time k is approximated by

$$p(\mathbf{x}_k|\mathbf{Y}_k) \approx \frac{1}{N_x} \sum_{i=1}^{N_x} w_k^i \delta\left(\mathbf{x}_k - \mathbf{x}_k^i\right) \quad (2)$$

where $\delta(\cdot)$ describes the Dirac delta function located at $\mathbf{x}_k^i$. The posterior state estimate establishes the starting point for the second phase, in which the particle filter is employed for predicting the RUL at given time k_p. To this aim the posterior estimate $p(\mathbf{x}_{k_p}|\mathbf{y}_{k_p})$ is set as initial condition.

By assuming that the set of particles $\mathbf{S}_k$ accurately represents the unknown states at the time of prediction, it is possible to approximate the probability density function of system states at any time $k_p + m$ in the future by means of the law of total probabilities (Orchard & Vachtsevanos, 2010)

$$\hat{p}\left(\mathbf{x}_{k_p+m}|\hat{\mathbf{x}}_{k_p:k_p+m-1}\right) \approx \\ \sum_{i=1}^{N_x} w_{k_p+m-1}^i \hat{p}\left(\hat{\mathbf{x}}_{k_p+m}^i|\hat{\mathbf{x}}_{k_p+m-1}^i\right). \quad (3)$$

To account for the fact, that during the prediction the shape of the states probability distribution may change, due to noise and process nonlinearities, Eq.(3) requires the set of weights to be updated at each iteration. However, during the prediction step no new measurements, which could serve for updating the weights, can be acquired. This implies that an update procedure for the particle weights, as it would happen in a typical filtering problem, cannot be carried out. This issue is addressed by assuming the weights as invariant from the time k_p to $k_p + m$. This assumption is justified by considering the uncertainty added by model inaccuracies or by the ignorance about future operation conditions to be large in comparison to the uncertainty which comes from considering constant particle weights. In this way, the set of weighted particles $\mathbf{S}_{k_p}$ is simply propagated forward into the future by simulating the behavior of the system as reaction to a future operation condition, until the determined failure condition is reached.

Once all particles have reached this point, i.e. $\boldsymbol{T}_{k_p}^i = 1$, the $\mathrm{RUL}_{k_p}^i$ of each particle is determined and combined with its weight $w_{k_p}^i$ to approximate $p\left(\mathrm{RUL}_{k_p}|\mathbf{Y}_{k_p}\right)$ as follows

$$p\left(\mathrm{RUL}_{k_p}|\mathbf{Y}_{k_p}\right) \approx \sum_{i=1}^{N_x} w_{k_p}^i \mathrm{RUL}_{k_p}^i. \qquad (4)$$

The RUL prediction, as formulated in Eq.(4), requires propagating the set of particles through a single hypothesized predicted profile of the future operation conditions of the system. However, such a propagation accounts just for the uncertainty introduced in the state estimation step but it does not consider the uncertainty related to the predicted operation profile. Taking this uncertainty into account would require propagating the set of particles through multiple predicted profiles, and not through a single one. Thus, the computational complexity of such a prediction becomes a function of $N_x \times N_u$ (Daigle, Saxena, & Goebel, 2012), where N_u is the number of predicted operation profiles. The set of weighted particles is then propagated through multiple profiles until all particles along all predicted profiles, have reached the threshold, i.e. $\boldsymbol{T}_{k_p}^{ij} = 1$. Here j represents each predicted operation profile. Accordingly, the probability distribution $p\left(\mathrm{RUL}_{k_p}|\mathbf{Y}_{k_p}\right)$ is approximated by

$$p\left(\mathrm{RUL}_{k_p}|\mathbf{Y}_{k_p}\right) \approx \frac{1}{N_u} \sum_{j=1}^{N_u} \sum_{i=1}^{N_x} w_{k_p}^i \mathrm{RUL}_{k_p}^{ij}. \qquad (5)$$

It must be noted that all predicted profiles are equally weighted by means of $\frac{1}{N_u}$.

3. PARTICLE FLOW PARTICLE FILTER

From the previous section it can be inferred that the computational performance of the particle-filter-based RUL estimation approach can be enhanced through the reduction of the particles employed during the estimation step and therefore during the prediction step. However, this cannot be done straightforward specially in those systems where the dimensionality of the state space is high. This problem becomes more significant in a joint state/parameter estimation since the dimensionality of the state space can increase considerably. In this paper we aim to investigate the suitability of an approach for reducing the number of particles needed in the estimation of the state space without sacrificing the accuracy of the state estimation.

Standard particle filters might reduce the computational performance of the prognostics algorithm during the estimation step by wasting computational resources during the propagation of those particles with negligible weights. Furthermore, since either particles with very low weight or duplicated particles have to be propagated forward in time until they reach the predefined threshold, additional resources might be wasted during the prediction step of the prognostics framework.

The approach presented in this paper aims to overcome the aforementioned issues by implementing an update schema, which progressively transforms the prior $p\left(\mathbf{x}_k|\mathbf{Y}_{k-1}\right)$ into the posterior state estimate $p\left(\mathbf{x}_k|\mathbf{Y}_k\right)$ by smoothly moving the particles in an optimal manner as new measurements become available without needing to employ any resampling algorithm. This is achieved by solving a differential equation to determine the flow of particles in the state space as they migrate from the prior to the posterior distribution. In a generic Bayesian framework, the posterior $p\left(\mathbf{x}_k|\mathbf{Y}_k\right)$ is obtained in the prediction step by a single computation of the Bayes' rule given by

$$\overbrace{p\left(\mathbf{x}_k|\mathbf{Y}_k\right)}^{\text{posterior}} = \frac{\overbrace{p\left(\mathbf{x}_k|\mathbf{Y}_{k-1}\right)}^{\text{prior}}\overbrace{p\left(\mathbf{y}_k|\mathbf{x}_k\right)}^{\text{likelihood}}}{\underbrace{\int_{\mathbb{R}} p\left(\mathbf{x}_k|\mathbf{Y}_{k-1}\right) p\left(\mathbf{y}_k|\mathbf{x}_k\right) \mathrm{d}x_k}_{\text{normalization factor}}}. \qquad (6)$$

By denoting a new set of density functions given by

$$\begin{aligned} \psi\left(\mathbf{x}_{k,\lambda}|\mathbf{Y}_k\right) &= p\left(\mathbf{x}_k|\mathbf{Y}_k\right) \\ g\left(\mathbf{x}_{k,\lambda}|\mathbf{Y}_{k-1}\right) &= p\left(\mathbf{x}_k|\mathbf{Y}_{k-1}\right) \end{aligned} \qquad (7)$$

it is possible to compute $\psi\left(\mathbf{x}_{k,\lambda}|\mathbf{Y}_k\right)$ in a *B*-fold recursive manner by progressively introducing the likelihood density, here denoted as $l\left(\mathbf{y}_k|\mathbf{x}_k\right)$, such that the prior $g\left(\mathbf{x}_{k,\lambda}|\mathbf{Y}_{k-1}\right)$ gradually deforms into $g\left(\mathbf{x}_{k,\lambda}|\mathbf{Y}_{k-1}\right) l\left(\mathbf{y}_k|\mathbf{x}_k\right)$. This can be achieved by using a homotopy of the form

$$\overbrace{\psi\left(\mathbf{x}_{k,\lambda}|\mathbf{Y}_k\right)}^{\text{posterior}} = \frac{\overbrace{g\left(\mathbf{x}_{k,\lambda}|\mathbf{Y}_{k-1}\right)}^{\text{prior}}\overbrace{l\left(\mathbf{y}_k|\mathbf{x}_{k,\lambda}\right)^{\lambda}}^{\text{likelihood}}}{\underbrace{K_{k,\lambda}}_{\text{normalization factor}}}, \qquad (8)$$

where $\lambda \in [0, 1]$ is the progression parameter and the term $l\left(\mathbf{y}_k|\mathbf{x}_{k,\lambda}\right)^\lambda$ is understood as an incremental likelihood. Thus, Eq. (8) represents the prior when $\lambda = 0$ and the posterior when $\lambda = 1$. The number of iterations in the recursion, namely B, depends on the step size $\Delta\lambda$, which determines the rate at which $\lambda_{0\to1}$. The way $l\left(\mathbf{y}_k|\mathbf{x}_{k,\lambda}\right)^\lambda$ is incrementally incorporated into the Bayes' update step can be seen in the Algorithm 1. For the sake of clarity, from now on we express the states variables as $\mathbf{x}_\lambda$ instead as $\mathbf{x}_{k,\lambda}$. This is due to the fact that the evolution of the probability distribution as $\lambda_{0\to1}$ always occurs at the discrete time step k. In order to avoid numerical issues the log-density of Eq. (8) is applied yielding to

$$\Psi\left(\mathbf{x}_\lambda\right) = G\left(\mathbf{x}_\lambda\right) + \lambda L\left(\mathbf{x}_\lambda\right) - \log K_\lambda, \qquad (9)$$

where the posterior is given by $\Psi\left(\mathbf{x}_\lambda\right) = \log\psi\left(\mathbf{x}_\lambda|\mathbf{Y}_k\right)$, the prior is represented by $G\left(\mathbf{x}_\lambda\right) = \log g\left(\mathbf{x}_\lambda|\mathbf{Y}_{k-1}\right)$ and the likelihood is $L\left(\mathbf{x}_\lambda\right) = \log l\left(\mathbf{y}_k|\mathbf{x}_\lambda\right)$. The evolution of the probability distribution given by Eq. (9) in the pseudo-time is known as *log-homotopy* (Daum & Huang, 2008). As it can be seen in Fig. 2, the task of this homotopy is to move the particles through a sequence of densities from the prior to the posterior as λ continuously increases from zero to one.
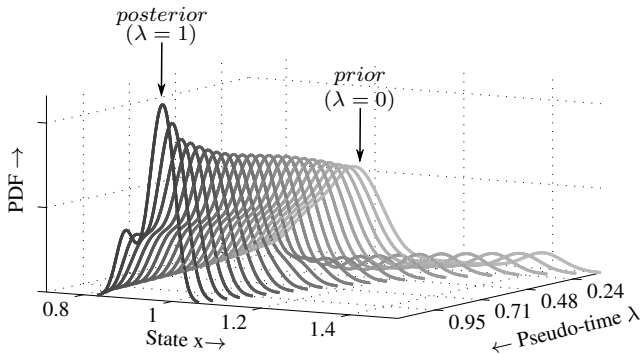

Figure 2. Evolution of the probability distribution from the prior at $\lambda = 0$ to the posterior at $\lambda = 1$.

As it can be observed in Fig. 3, it becomes necessary to find a flow $\frac{d\mathbf{x}}{d\lambda}$ that dictates the motion of particles as they move following the log-homotopy given by Eq. (9).

To this aim we differentiate Eq. (9) with respect to λ

$$\frac{\partial\Psi\left(\mathbf{x}_\lambda\right)}{\partial\lambda} = L\left(\mathbf{x}_\lambda\right) - \frac{d}{d\lambda}\log K_\lambda. \qquad (10)$$

Replacing the left hand side of Eq. (10) by the logarithm identity

$$\frac{\partial\Psi\left(\mathbf{x}_\lambda\right)}{\partial\lambda} = \frac{1}{\psi\left(\mathbf{x}_\lambda\right)}\frac{\partial\psi\left(\mathbf{x}_\lambda\right)}{\partial\lambda} \qquad (11)$$

and multiplying both sides by $\psi\left(\mathbf{x}_\lambda\right)$ yields to

$$\frac{\partial\psi\left(\mathbf{x}_\lambda\right)}{\partial\lambda} = \psi\left(\mathbf{x}_\lambda\right)\left[L\left(\mathbf{x}_\lambda\right) - \frac{d\log K_\lambda}{d\lambda}\right]. \qquad (12)$$

A way to find the desired flow $\frac{d\mathbf{x}}{d\lambda}$ is by considering that the particles move, as $\lambda_{0\to1}$, obeying the following stochastic differential equation (SDE)

$$d\mathbf{x}_\lambda = \boldsymbol{\zeta}\left(\mathbf{x}_\lambda\right)d\lambda + \boldsymbol{\eta}\left(\mathbf{x}_\lambda\right)d\boldsymbol{\xi}_\lambda, \qquad (13)$$

where $\mathbf{x}_\lambda$ is the particle position at given time k and pseudo-time λ, $\boldsymbol{\zeta}\left(\mathbf{x}_\lambda\right)$ can be understood as a *vector field* that induces the motion of particles from the prior to the posterior distribution, $\eta\left(\cdot\right)$ is a multiplicative noise matrix and $\boldsymbol{\xi}_\lambda$ is a noise resulting from the randomness of process.

By considering $\frac{d\mathbf{x}}{d\lambda}$ to be given by $\boldsymbol{\zeta}\left(x_\lambda\right)$, the desired particle flow can be obtained by using the conditional probability density $\psi\left(\mathbf{x}_\lambda\right)$ together with the forward Kolmogorov equation, also known as the Fokker-Planck-Kolmogorov (FPK) equation. In this context the FPK equation is employed to relate the flow $\frac{d\mathbf{x}}{d\lambda}$ of a particle with the evolution of $\psi\left(\mathbf{x}_\lambda\right)$ as $\lambda_{0\to1}$ under the influence of drift and diffusion processes.

The FPK equation can be written as

$$\frac{\partial\psi\left(\mathbf{x}_\lambda\right)}{\partial\lambda} = \overbrace{-\text{tr}\left[\frac{\partial}{\partial\mathbf{x}_\lambda}\left(\boldsymbol{\zeta}\left(\mathbf{x}_\lambda\right)\psi\left(\mathbf{x}_\lambda\right)\right)\right]}^{\text{drift}} + \overbrace{\frac{1}{2}\text{tr}\left[\frac{\partial}{\partial\mathbf{x}_\lambda}\left(\mathbf{Q}\left(\mathbf{x}_\lambda\right)\frac{\partial\psi\left(\mathbf{x}_\lambda\right)}{\partial\mathbf{x}_\lambda}\right)\right]}^{\text{diffusion}}, \qquad (14)$$

where $\mathbf{Q}\left(\mathbf{x}_\lambda\right) = \boldsymbol{\eta}\left(\mathbf{x}_\lambda\right)\boldsymbol{\eta}^{\mathrm{T}}\left(\mathbf{x}_\lambda\right)$ is the process covariance matrix and $\text{tr}\left(\cdot\right)$ states for the *trace* of $\left(\cdot\right)$.

Reformulating Eq. (14) in a more proper way yields

$$\begin{aligned}\frac{\partial\psi\left(\mathbf{x}_\lambda\right)}{\partial\lambda} &= -\text{tr}\left[\psi\left(\mathbf{x}_\lambda\right)\frac{\partial\boldsymbol{\zeta}\left(\mathbf{x}_\lambda\right)}{\partial\mathbf{x}_\lambda} + \boldsymbol{\zeta}\left(\mathbf{x}_\lambda\right)^{\mathrm{T}}\frac{\partial\psi\left(\mathbf{x}_\lambda\right)}{\partial\mathbf{x}_\lambda}\right] + \\ &\quad + \frac{1}{2}\text{div}\left(\mathbf{Q}\left(\mathbf{x}_\lambda\right)\frac{\partial\psi\left(\mathbf{x}_\lambda\right)}{\partial\mathbf{x}_\lambda}\right) \\ &= -\boldsymbol{\zeta}\left(\mathbf{x}_\lambda\right)^{\mathrm{T}}\frac{\partial\psi\left(\mathbf{x}_\lambda\right)}{\partial\mathbf{x}_\lambda} - \psi\left(\mathbf{x}_\lambda\right)\text{tr}\left(\frac{\partial\boldsymbol{\zeta}\left(\mathbf{x}_\lambda\right)}{\partial\mathbf{x}_\lambda}\right) + \\ &\quad + \frac{1}{2}\text{div}\left(\mathbf{Q}\left(\mathbf{x}_\lambda\right)\frac{\partial\psi\left(\mathbf{x}_\lambda\right)}{\partial\mathbf{x}_\lambda}\right), \end{aligned} \qquad (15)$$

where $\text{div}\left(\cdot\right)$ states for the *divergence* of $\left(\cdot\right)$. As it can be seen, Eq. (12) and Eq. (15) are equivalent. Thus, equating them and by dividing both sides by $\psi\left(\mathbf{x}_\lambda\right)$ we can write

4
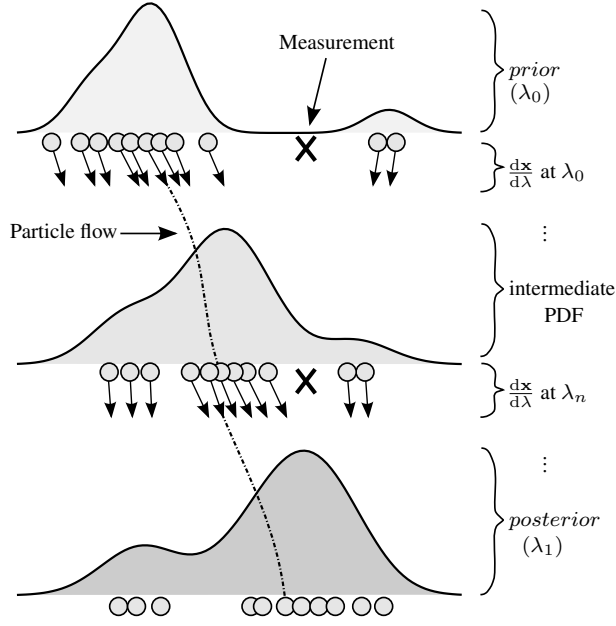
Figure 3. Particle flow at different values of λ.

$$
L\left(\mathbf{x}_\lambda\right) - \frac{\mathrm{d}\log K_\lambda}{\mathrm{d}\lambda} = -\zeta\left(\mathbf{x}_\lambda\right)^\mathrm{T} \frac{1}{\psi\left(\mathbf{x}_\lambda\right)} \frac{\partial \psi\left(\mathbf{x}_\lambda\right)}{\partial \mathrm{x}_\lambda} +
$$
$$
-\mathrm{tr}\left(\frac{\partial \zeta\left(\mathbf{x}_\lambda\right)}{\partial \mathrm{x}_\lambda}\right) +
$$
$$
+\frac{1}{2\psi\left(\mathbf{x}_\lambda\right)}\mathrm{div}\left(\mathbf{Q}\left(\mathbf{x}_\lambda\right)\frac{\partial \psi\left(\mathbf{x}_\lambda\right)}{\partial \mathrm{x}_\lambda}\right),
$$
$$
(16)
$$

under the assumption that $\psi\left(\mathbf{x}_\lambda\right)$ is nowhere vanishing. The desired particle flow is found by solving Eq. (16) wrt. $\zeta\left(\mathbf{x}_\lambda\right)$.

To this aim we first compute the gradient wrt. $\mathbf{x}_\lambda$. This yields to a system of partially differential equations (PDEs) with the same number of unknowns and equations given by

$$
\frac{\partial L\left(\mathbf{x}_\lambda\right)}{\partial \mathbf{x}_\lambda} = -\zeta^\mathrm{T}\left(\mathbf{x}_\lambda\right) \frac{\partial^2 \Psi\left(\mathbf{x}_\lambda\right)}{\partial \mathbf{x}_\lambda^2} - \frac{\partial \Psi\left(\mathbf{x}_\lambda\right)}{\partial \mathbf{x}_\lambda} \frac{\partial \zeta\left(\mathbf{x}_\lambda\right)}{\partial \mathbf{x}_\lambda} +
$$
$$
-\frac{\partial}{\partial \mathbf{x}_\lambda}\left[\mathrm{tr}\left(\frac{\partial \zeta\left(\mathbf{x}_\lambda\right)}{\partial \mathbf{x}_\lambda}\right)\right] +
$$
$$
+\frac{\partial}{\partial \mathbf{x}_\lambda}\left[\frac{1}{2\psi\left(\mathbf{x}_\lambda\right)}\mathrm{div}\left(\mathbf{Q}\left(\mathbf{x}_\lambda\right)\frac{\partial \psi\left(\mathbf{x}_\lambda\right)}{\partial \mathbf{x}_\lambda}\right)\right].
$$
$$
(17)
$$

There are many methods to solve the system of PDE's given by Eq. (17) (Daum & Huang, 2010). In this work we employ the approach presented by (Daum & Huang, 2013) in which it is assumed that both the process noise matrix $\mathbf{Q}\left(\mathbf{x}_\lambda\right)$ and the vector field given by $\zeta\left(\mathbf{x}_\lambda\right)$ are chosen such that sum of the

last three terms of Eq. (17) is zero. In this manner the system of PDE's is drastically simplified yielding to the following equation

$$
\frac{\partial L\left(\mathbf{x}_\lambda\right)}{\partial \mathbf{x}} = -\zeta^\mathrm{T}\left(\mathbf{x}_\lambda\right) \frac{\partial^2 \Psi\left(\mathbf{x}_\lambda\right)}{\partial \mathbf{x}_\lambda^2} \qquad (18)
$$

As stated by (Daum & Huang, 2013), if it is assumed that $\frac{\partial^2 \Psi(\mathbf{x}_\lambda)}{\partial \mathbf{x}_\lambda^2}$ is non-singular, the solution of Eq. (18) for $\zeta\left(\mathbf{x}_\lambda\right)$ can be computed as

$$
\zeta\left(\mathbf{x}_\lambda\right) = -\left[\frac{\partial^2 \Psi\left(\mathbf{x}_\lambda\right)}{\partial \mathbf{x}_\lambda^2}\right]^{-1}\left[\frac{\partial L\left(\mathbf{x}_\lambda\right)}{\partial \mathbf{x}_\lambda}\right]^\mathrm{T}. \qquad (19)
$$

The task now is to compute the terms of the right hand side of Eq. (19). First, the Hessian $\frac{\partial^2 \Psi(\mathbf{x}_\lambda)}{\partial \mathbf{x}_\lambda^2}$ can be obtained in closed form by differentiating twice Eq. (9) wrt. $\mathbf{x}_\lambda$

$$
\frac{\partial^2 \Psi\left(\mathbf{x}_\lambda\right)}{\partial \mathbf{x}_\lambda^2} = \frac{\partial^2 G\left(\mathbf{x}_\lambda\right)}{\partial \mathbf{x}_\lambda^2} + \lambda \frac{\partial^2 L\left(\mathbf{x}_\lambda\right)}{\partial \mathbf{x}_\lambda^2}. \qquad (20)
$$

In this work we use a hybrid approach for computing Eq. (20) in which the Hessian $\frac{\partial^2 G(\mathbf{x}_\lambda)}{\partial \mathbf{x}_\lambda^2}$ is approximated by

$$
\frac{\partial^2 G\left(\mathbf{x}_\lambda\right)}{\partial \mathbf{x}_\lambda^2} \approx -\hat{\mathbf{S}}_{N_x}^{-1}, \qquad (21)
$$

where $\hat{\mathbf{S}}_{N_x}$ is the sample covariance matrix (SCM) of the prior distribution computed from the set of N_x particles. The SCM offers an unbiased estimate of the true covariance matrix. However, it has to be noted that if the number of particles employed is smaller than the number of states to be estimated the SCM may suffer from high variance. To overcome this issue the Kronecker product expansion can be used to estimate the covariance matrix in high dimensional spaces (Tsiligkaridis & Hero, 2013).

If it is assumed that the prior $g\left(\cdot\right)$ is represented by a Gaussian distribution, then the approximation given by Eq. (21) is exact. For practical purposes the likelihood function $l\left(\cdot\right)$ can be assumed to follow an univariate or a multivariate Gaussian distribution depending on the dimension of the output vector. Accordingly, $L\left(\mathbf{x}_\lambda\right)$ is expressed as

$$
L\left(\mathbf{x}_\lambda\right) = -\frac{N}{2}\log\left(2\pi\right) - \frac{1}{2}\log|\mathbf{R}| - \frac{1}{2}\mathbf{z}_{k,\lambda}^\mathrm{T}\mathbf{R}^{-1}\mathbf{z}_{k,\lambda}, \quad (22)
$$

where $\mathbf{z}_{k,\lambda} = \left(\mathbf{y}_k - \mathbf{h}\left(\mathbf{x}_\lambda\right)\right)$ and $\mathbf{R}$ is the covariance matrix of the measurement noise. Computing the gradient of Eq. (22) wrt. $\mathbf{x}_\lambda$ gives

$$
\frac{\partial L\left(\mathbf{x}_\lambda\right)}{\partial \mathbf{x}_\lambda} = \left[\frac{\partial \mathbf{h}\left(\mathbf{x}_\lambda\right)}{\partial \mathbf{x}_\lambda}\right]^\mathrm{T}\mathbf{R}^{-1}\mathbf{z}_{k,\lambda}
$$
$$
= \hat{\mathbf{H}}\left(\mathbf{x}_\lambda\right)^\mathrm{T}\mathbf{R}^{-1}\left(\mathbf{y}_k - \mathbf{h}\left(\mathbf{x}_\lambda\right)\right),
$$
$$
(23)
$$

where $\hat{\mathbf{H}}\left(\mathbf{x}_\lambda\right)$ is the linearized output matrix around $\mathbf{x}_\lambda$. Com-

puting the Hessian $\frac{\partial^2 L(\mathbf{x}_\lambda)}{\partial \mathbf{x}_\lambda^2}$ might be computationally expensive. We instead approximate it by computing the expected Hessian by means of the Monte Carlo approximation method as follows

$$
\begin{aligned}
\frac{\partial^2 L(\mathbf{x}_\lambda)}{\partial \mathbf{x}_\lambda^2} &\approx E\left[\frac{\partial^2 L(\mathbf{x}_\lambda)}{\partial \mathbf{x}_\lambda^2}\right] \\
&\approx -\frac{1}{N_x}\sum_{i=1}^{N_x}\left[\frac{\partial \mathbf{z}_{k,\lambda}(\mathbf{x}_\lambda)}{\partial \mathbf{x}_\lambda^i}\right]^{\mathrm{T}}\mathbf{R}^{-1}\frac{\partial \mathbf{z}_{k,\lambda}(\mathbf{x}_\lambda)}{\partial \mathbf{x}_\lambda^i},
\end{aligned}
\tag{24}
$$

where $E[\cdot]$ is the expected value with respect to the likelihood function. After having computed $\frac{\partial L(\mathbf{x}_\lambda)}{\partial \mathbf{x}_\lambda}$ and $\frac{\partial^2 L(\mathbf{x}_\lambda)}{\partial \mathbf{x}_\lambda^2}$ both Eq. (19) and Eq. (20) can be evaluated in order to obtain the particle flow. As it can be seen, evaluating Eq. (20) requires computing the inverse of $\hat{\mathbf{S}}_{N_x}$, which can lead to numerical problems if $\hat{\mathbf{S}}_{N_x}$ is close to be singular. To overcome this issue we apply the matrix inversion lemma known as Woodbury's formula in order to invert Eq. (20) as follows

$$
\begin{aligned}
\left[\frac{\partial^2 \Psi(\mathbf{x}_\lambda)}{\partial \mathbf{x}_\lambda^2}\right]^{-1} &= -\hat{\mathbf{S}}_{N_x} + \\
&- \hat{\mathbf{S}}_{N_x}\lambda\frac{\partial^2 L(\mathbf{x}_\lambda)}{\partial \mathbf{x}_\lambda^2}\left(I - \hat{\mathbf{S}}_{N_x}\lambda\frac{\partial^2 L(\mathbf{x}_\lambda)}{\partial \mathbf{x}_\lambda^2}\right)^{-1}\hat{\mathbf{S}}_{N_x}.
\end{aligned}
\tag{25}
$$

Algorithm 1 summarizes the steps needed for implementing the presented particle flow particle filter for state estimation. It is worth noting that the rate at which $\lambda_{0\to1}$ is determined by the step size $\Delta\lambda$. Numerical experiments presented by (Daum & Huang, 2013) have shown that employing a fixed step size, such as in the case of the Euler method, works properly just if the number of particles is high. Therefore, to reduce the number of particles employed a variable $\Delta\lambda$ has to be used. A proper strategy is to use a very small value of $\Delta\lambda$ at the beginning and to gradually increase it as $\lambda \to 1$, which makes sense, since the uncertainty a the beginning of the measurement update step is higher. We therefore use an exponentially increasing step size (George & Powell, 2006) given by

$$
\Delta\lambda = 1 - \frac{1}{n^b},
\tag{26}
$$

where n is the number of iteration and $b \in \left(\frac{1}{2}, 1\right]$. In the case of initial transient conditions the a small value of b can lead to a slower learning rate of the step size. The value of b should be chosen according to the desired rate of convergence of the step size.

4. CASE STUDY

For validating the applicability of the particle flow particle filter for prognostics we chose the remaining driving range

(RDR) estimation of an electric vehicle (Oliva et al., 2013). In this context, the RDR estimation is concerned with predicting the power demand of the electric vehicle and identifying the distance that it can drive with the energy stored in its battery before recharging is required. To this aim we consider the battery state of charge (SOC) to be the indicator that determines the threshold condition.

Algorithm 1 Particle flow particle filter for state estimation

---

Initialization
Draw a set of particles $\{\mathbf{x}_0^i\}_{i=1}^{N_x}$ from the prior $p(\mathbf{x}_0)$
for $k = 1$ **to** ∞ **do**
  State prediction
  Propagate the particles through the system equation:
  $\mathbf{x}_{k|k-1}^i = \mathbf{f}\left(\mathbf{x}_{k-1}^i, \mathbf{u}_k, \mathbf{v}_k n; \mathbf{w}_k\right)$
  Initialize the pseudo-time $\lambda = 0$
  Set $\left\{\mathbf{x}_{k,\lambda}^i\right\}_{i=1}^{N_x} = \left\{\mathbf{x}_{k|k-1}^i\right\}_{i=1}^{N_x}$
  Measurement update:
  Propagate the particles through the output equation:
  $\mathbf{y}_{k|k}^i = \mathbf{h}\left(\mathbf{x}_k^i, \mathbf{u}_k, \mathbf{n}_k, \mathbf{w}_k\right)$
  while $\lambda \leq 1$ **do**
    Compute $\hat{\mathbf{S}}_{N_x}$ from $\left\{\mathbf{x}_{k,\lambda}^i\right\}_{i=1}^{N_x}$
    Calculate the state estimation from $\left\{\mathbf{x}_{k,\lambda}^i\right\}_{i=1}^{N_x}$
    $\hat{\mathbf{x}}_{k,\lambda} = \frac{1}{N_x}\sum_{i=1}^{N_x}\mathbf{x}_{k,\lambda}^i$
    Linearize $\mathbf{h}(\cdot)$ around $\hat{\mathbf{x}}_{k,\lambda}$ to compute $\hat{\mathbf{H}}$
    for $i = 1$ **to** N_x **do**
      Compute the flow $\zeta\left(\mathbf{x}_{k,\lambda}^i\right)$ for each particle
      Set $\frac{\mathrm{d}\mathbf{x}_{k,\lambda}^i}{\mathrm{d}\lambda} = \zeta\left(\mathbf{x}_{k,\lambda}^i\right)$
      Move the particles according their respective flow:
      $\mathbf{x}_{k,\lambda}^i = \mathbf{x}_{k,\lambda}^i + \Delta\lambda\frac{\mathrm{d}\mathbf{x}_{k,\lambda}^i}{\mathrm{d}\lambda}$
    end for
    Increment the pseudo-time $\lambda \leftarrow \lambda + \Delta\lambda$
  end while
  Update the state estimation:
  $\hat{\mathbf{x}}_k = \frac{1}{N_x}\sum_{i=1}^{N_x}\mathbf{x}_{k,\lambda}^i$
end for

Accordingly, the threshold is expressed as $\boldsymbol{T}(\mathrm{SOC})$. Thus, $\boldsymbol{T}(\mathrm{SOC}) = 1$ if $\mathrm{SOC}_{\min}$ (the minimum allowable state of charge) is reached and $\boldsymbol{T}(\mathrm{SOC}) = 0$, otherwise. The $\mathrm{SOC}_{\min}$ is usually dictated by the battery management system (BMS) of the electric vehicle in order to protect the battery cells from a possible total charge depletion.

4.1. Battery Model

We employ the model of a Li-ion cell shown in Fig. 4. The model combines the Kinetic Battery Model (Manwell & McGowan, 1994) for capturing the nonlinear effects in the battery capacity, such as the recovery and the rate capacity effect, with a second order equivalent circuit based model which

captures the dynamic response of the Li-ion cell. Furthermore, the combined model demands low computational effort, which makes it suitable for real-time applications. Even though the KiBaM was initially developed for lead acid batteries, it has been shown to be suitable for modeling the capacity behavior of Li-ion cells (Jongerden & Haverkort, 2009).
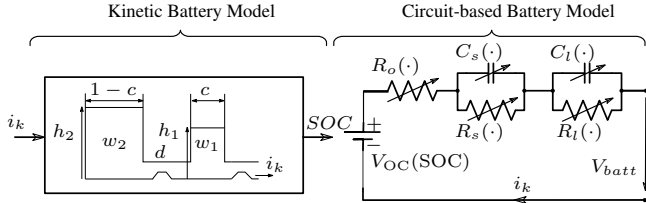

Figure 4. Combined battery model.

The Kinetic Battery Model abstracts the chemical processes of the battery discharge to its kinetic properties. The model assumes that the total charge of the battery is distributed with a capacity ratio $0 < c < 1$ between two charge wells. The first well contains the available charge and delivers it directly to the load. The second well supplies charge only to the first well by means of the parameter d. The rate of charge that flows from the second to the first well depends on both d and on the height difference between the wells $(h_2 - h_1)$. If the first well is empty, then the battery is considered to be fully discharged. By applying load to the battery, the charge in the first well is reduced, which leads to an increment in the height difference between both wells. After removing the load, certain amount of charge flows from the second well to the first well until the height of both wells is the same. In this way the recovery effect is taken into account by the model. The rate capacity effect is also considered in this model. For high discharge currents, the charge in the first well is delivered faster to the load in comparison to the charge that flows from the second well. In this scenario there is an amount of charge that remains unused. The consideration of this effect is especially important for applications in electric vehicles, since the unused charge might eventually increase the driving range. The KiBaM yields two difference equations which describe the change of capacity in both wells in dependence of the load i_k, the conductance d and the capacity ratio c:

$$w_{1,k+1} = a_1 w_{1,k} + a_2 w_{2,k} + b_1 i_k, \qquad (27)$$

$$w_{2,k+1} = a_3 w_{1,k} + a_4 w_{2,k} + b_2 i_k, \qquad (28)$$

where

$$\begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} = e^{\begin{pmatrix} -\frac{d}{c} & \frac{d}{1-c} \\ \frac{d}{c} & -\frac{d}{1-c} \end{pmatrix} \Delta t}$$

$$\begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \int_0^{\Delta t} e^{\begin{pmatrix} -\frac{d}{c} & \frac{d}{1-c} \\ \frac{d}{c} & -\frac{d}{1-c} \end{pmatrix} \vartheta} d\vartheta \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

The term Δt is the sampling time used in the discretization of the model. The battery SOC is given by

$$\text{SOC}_k = \frac{w_{1,k}}{cC_n 3600}, \qquad (29)$$

where C_n is the nominal capacity of the battery. The right-hand-side equivalent circuit of Fig. 4 is compounded of three parts, namely, the open circuit voltage V_{OC}, a resistance R_o and two RC networks.

The voltage V_{OC} changes at different SOC levels, as depicted in Fig. 5. The ohmic resistance R_o captures the I-R drop, i.e., the instantaneous voltage drop due to a step load current event. The $R_s C_s$ and $R_l C_l$ networks capture the voltage drops due to the electrochemical and the concentration polarization, respectively. In Fig. 4 the dependency of these parameters on the temperature and on the SOC is represented by the term $(\cdot)$.
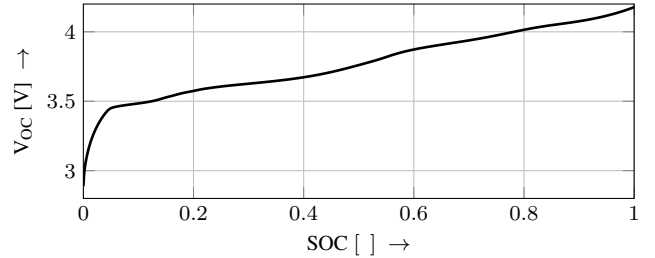

Figure 5. $V_{\text{OC}} - \text{SOC}$ relationship.

This part of the model yields two difference equations which describe the transient response of the battery:

$$v_{s,k+1} = e^{-\frac{\Delta t}{R_s C_s}} v_{s,k} + \left(-R_s e^{-\frac{\Delta t}{R_s C_s}} + R_s \right) i_k, \qquad (30)$$

$$v_{l,k+1} = e^{-\frac{\Delta t}{R_l C_l}} v_{l,k} + \left(-R_l e^{-\frac{\Delta t}{R_l C_l}} + R_l \right) i_k. \qquad (31)$$

Accordingly, the state vector of the battery model is given by

$$\mathbf{x}_k = \begin{bmatrix} w_{1,k} & w_{2,k} & v_{s,k} & v_{l,k} \end{bmatrix}^T. \qquad (32)$$

The output y_k of the system, represented by the terminal voltage $V_{batt,k}$, is then computed as follows

$$y_k = V_{batt,k}(\text{SOC}) = V_{\text{OC}}(\text{SOC}) + R_o i_k + v_{l,k} + v_{s,k}. \qquad (33)$$

5. RESULTS AND DISCUSSIONS

This section evaluates the particle flow particle filter in both accuracy and computational performance in the estimation of the RDR of an electric vehicle. To measure the accuracy of the RDR estimation we employ the relative accuracy (RA) and the alpha-lambda $(\alpha - \lambda)$ metric (Saxena, Celaya, Saha, Saha, & Goebel, 2009). In the context of the RDR estimation

the RA is given by

$$\mathrm{RA}_{k_p} = 100 \left(1 - \frac{\left| \mathrm{RDR}_{k_p}^* - \mathrm{RDR}_{k_p} \right|}{\mathrm{RDR}_{k_p}^*} \right), \quad (34)$$

where $\mathrm{RDR}_{k_p}^*$ is the ground truth RDR at time k_p and RDR_{k_p} is the estimated RDR at that time. The $\alpha - \lambda$ metric serves to evaluate whether the estimated RDR lies withing specified bounds.

5.1. Experimental results

The first set of experiments aims to test the suitability of the PFPF in prognostics on the one hand, and to compare its performance in contrast to the PF, on the other hand. To this aim the load profile shown in the top part of Fig.6 is applied to a Li-ion cell until the pre established $\mathrm{SOC}_{\mathrm{min}}$ is reached. For this experiment a cell with a nominal capacity $C_n = 2.15$ Ah, a nominal voltage $V_{\mathrm{nom}} = 4.2$ V and a $\mathrm{SOC}_{\mathrm{min}} = 0.15$ is used. The load profile is computed by scaling down the theoretical load of an electric vehicle driving the standard UDDS (Urban Dynamometer Driving Schedule) drive cycle. In this way it is possible to directly relate the load with the speed of the vehicle and therefore to compute the RDR.

First, the accuracy of the SOC estimation is investigated. To this aim both filters run in parallel and recursively estimate the SOC. The bottom part of Fig.6 depicts the results of the state estimation. As it can be seen, both filters are very accurate while estimating the SOC. The main difference lies on the number of particles used. For the estimation shown just 10 particles are employed by the PFPF, whereas the PF needs 100 in order to estimate the SOC with the same accuracy as the PFPF. This is by no means a claim of improvement of the particle filter for state estimation, but a suggestion that the PFPF successfully manage to estimate states in nonlinear systems with many less particles.

After having proved the applicability of the PFPF for estimating the SOC, the second step is to validate the accuracy and the computational performance of the RDR estimation. To this aim a series of predictions are carried out at different stages of the discharge process every 500 s. Since for this experiment the future load profile of the battery is assumed to be known, the error presented in the RDR estimation is attributed to the model inaccuracy and to the SOC estimation error. A RDR prediction proceeds by simulating the evolution of the battery SOC, from a given time k_p, as a response to the predicted load and by determining the point in the future, at which the $\mathrm{SOC}_{\mathrm{min}}$ is reached. The initial state values at the time of prediction are dictated by the value of the particles obtained from the state estimation step. This procedure is repeated for all particles. The RDR distribution is then computed by means of Eq.(4). As it can be appreciated in Fig.7, the RDR prediction shows a high RA, with the ex-
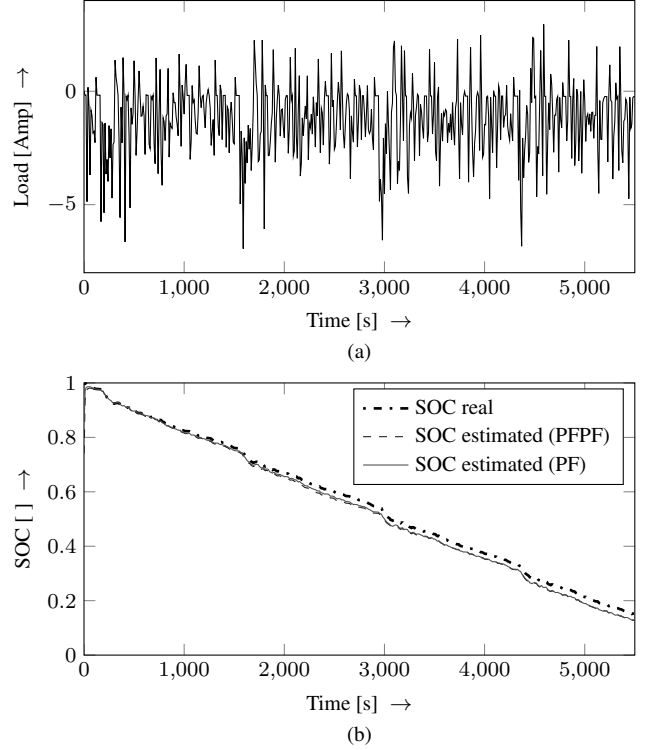

Figure 6. a) Load profile derived from the UDDS drive cycle. b) SOC estimation with the PFPF and the PF.
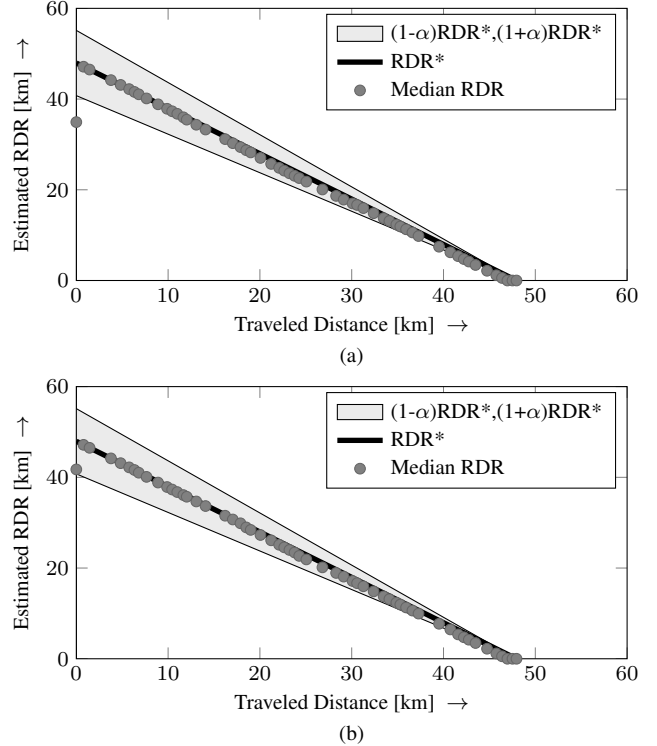

Figure 7. RDR estimation with a) PFPF and b) PF.

ception of the first prediction and the predictions carried out near the end of discharge of the battery. This first deviation is due to the fact that the state estimation in both cases is initialized by uniformly spreading the particles among the entire state space, which causes the estimation to deviate from the real value. Once the filters converge to the real SOC, the RA increases remarkably. As it can be seen, the RA decreases towards the end of discharge at $k_p = 45$ and $k_p = 50$. This phenomenon is attributed to the abrupt voltage drop that the battery exhibits at around $SOC = 5\%$, as it is shown in Fig. 5. The battery model doesn't accurately capture the behavior of the terminal voltage in this region, which causes the filter algorithm to slightly diverge from the real SOC. Since the uncertainty presented in the filtering step is the only uncertainty considered in this case study, a reduction in the accuracy of the state estimation directly causes a reduction in the RA.

Table 1 presents the RA and the time needed to complete a prediction, here referred as t_{cpu}, for different prediction times. As it can be noted, in average the t_{cpu} of those predictions done with the PFPF are three times faster than those carried out with the PF.

Table 1. RDR prediction performance.

| | Urban | | | |
| | RA [%] | | t_{cpu} [s] | |
k_p	PFPF	PF	PFPF	PF
1	72.83	87.05	3.16	3.91
5	100.0	100.0	0.327	1.078
10	99.48	99.48	0.305	0.927
15	98.72	99.44	0.287	0.808
20	97.82	99.33	0.273	0.730
25	96.06	97.22	0.253	0.671
30	95.25	95.67	0.235	0.568
35	95.88	95.98	0.222	0.479
40	94.33	94.53	0.206	0.407
45	88.26	91.41	0.109	0.324
50	76.75	77.80	0.176	0.241

5.2. Simulation results

A series of simulations is carried out in order to incorporate the uncertainty introduced by the randomness of the driving environment into the RDR estimation. To this aim the methodology previously presented in together with the model of an electric vehicle is used to compute power demand as response to a predicted driving profile, i.e., speed, acceleration and slope profile. The approach for predicting the driving profiles is however out of the scope of this work. The reader is referred to (Oliva et al., 2013) for a detailed explanation about the methodology employed for estimating the RDR.

The RDR prediction proceeds similarly as in the previous section with the difference that in this case each particle is simulated through 50 different predicted driving profiles, i.e, $N_u = 50$. In this case 10 particles are employed by the PFPF and 50 by the PF in order to obtain similar accuracy in the
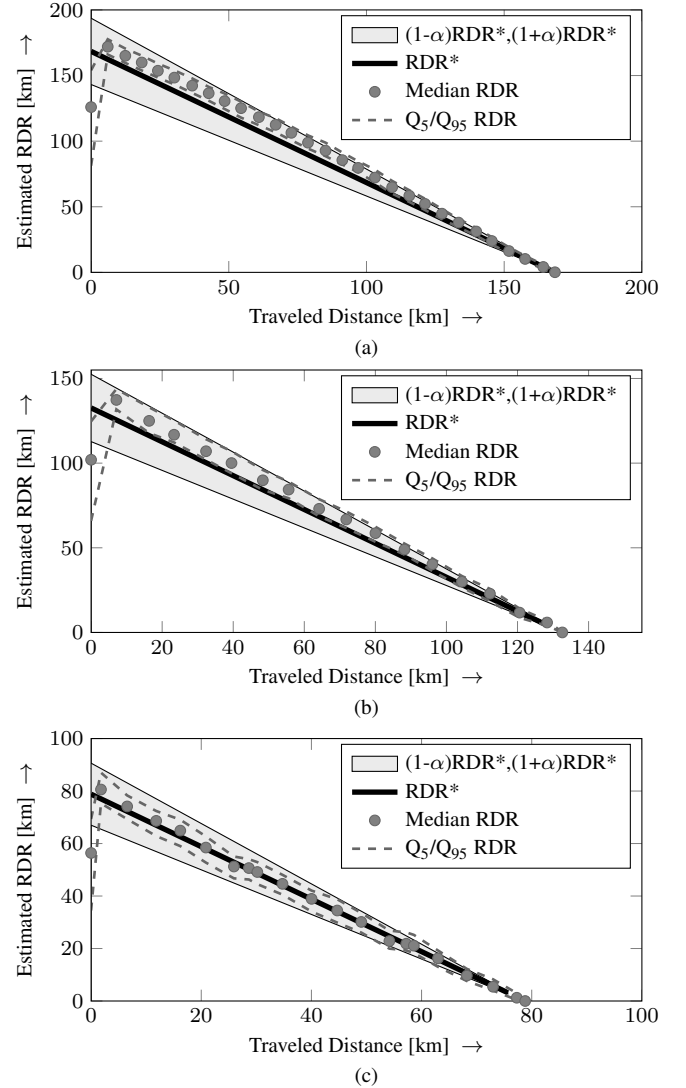

(a)

(b)

(c)

Figure 8. PFPF based RDR estimation in different driving scenarios a) city b) rural areas c) highway.

state estimation. As it is shown in Fig.8, the RDR prediction is carried out under three different driving scenarios, namely in the city and rural areas and on the highway.

The simulation results show that the PFPF is also suitable for estimating the RDR even in situations where the future driving load is unknown and that it reduces the computational complexity of the entire prognostics process. In table 2 both the RA and the t_{cpu} for all scenarios is presented. As it can be observed, even though the PF employ more particles than the PFPF, the accuracy in the RDR prediction is in general not better. Furthermore, a noticeable improvement in the computational performance is appreciated in respect to the experimental results. Although, the PF uses just half of the particles as before, the t_{cpu} is now 4 to 5 times larger than the t_{cpu} required by the PFPF in all scenarios.

Table 2. RDR prediction performance under different driving scenarios.

	Driving scenario											
	Urban				Rural				Highway			
	RA [%]		t_{cpu} [s]		RA [%]		t_{cpu} [s]		RA [%]		t_{cpu} [s]	
k_p	PFPF	PF	PFPF	PF	PFPF	PF	PFPF	PF	PFPF	PF	PFPF	PF
1	74.78	88.48	18.21	88.19	76.91	89.00	6.83	23.06	71.56	89.82	3.19	20.50
3	94.19	79.07	16.46	79.83	92.52	87.32	4.06	19.12	97.39	88.00	3.28	17.04
5	93.42	90.54	15.41	72.53	93.42	89.44	3.68	17.03	96.30	87.75	3.10	14.91
7	91.97	90.19	14.64	68.39	93.37	87.13	3.22	15.51	96.91	91.76	2.83	13.79
9	91.80	89.93	13.38	63.40	93.29	88.44	2.88	13.50	98.93	86.91	2.79	13.63
11	89.96	90.37	12.52	57.09	88.52	88.27	2.50	11.46	99.64	94.96	2.64	12.68
13	88.87	91.22	11.37	50.06	89.85	92.74	2.09	9.70	98.73	91.97	2.53	12.18
15	88.60	90.46	10.27	44.70	87.76	77.08	1.81	7.54	98.90	81.45	2.26	11.85
17	88.56	90.59	9.00	38.98	63.20	81.49	1.33	5.39	97.27	74.13	2.11	10.82
21	89.40	89.43	6.65	28.37	–	–	–	–	93.16	81.11	1.88	8.16
25	95.75	89.43	3.88	16.27	–	–	–	–	–	–	–	–

6. CONCLUSIONS AND FUTURE WORK

In this work a methodology for enhancing the computational performance of a particle-filtering-based prognostics approach is presented. The reduction in computational complexity is achieved by reducing the number of particles needed in the state estimation and thereby reducing the number of simulations needed to determine the RUL of the system. The reduction of particles is carried out by applying a deterministic flow, which migrates the particles through the state space in an optimal manner from the prior to the posterior state estimate. The advantage of such a migration allows us to employ less particles in contrast to the standard particle filter, since the particles are moved to the correct location obeying to the Bayes's rule. Such a particle reduction is highlighted during the prediction step, due to less simulations are needed for determining the distribution of the RUL.

The proposed methodology is afterwards illustrated and validated by means of the RDR estimation problem, in which is desired to determine the distance that can be driven by an electric vehicle with the energy stored in the battery pack at given points in time. Both experimental and simulation results show that the particle flow particle filter successfully reduces the computational burden associated with the estimation of the RUL in nonlinear systems.

Even though the presented approach exhibits both good computational performance and estimation accuracy, it is worth mentioning that the experiments carried out are based just on state estimation. That is, no joint or dual state/parameter estimation is done. This is justified by the assumption that the parameters of the battery model degrade very slow within the time span of a trip. However, a more proper implementation of the RDR estimation problem requires estimating the parameters together with the states in order to account for the aging effect of the battery. We therefore aim to investigate in the future the applicability and performance of the particle flow particle filter for a joint state/parameter estimation.

ACKNOWLEDGMENT

REFERENCES

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

Daigle, M., Saxena, A., & Goebel, K. (2012). An efficient deterministic approach to model-based prediction uncertainty estimation. In *Annual conference of the prognostics and health management society 2012.*

Daum, F., & Huang, J. (2008). Particle flow for nonlinear filters with log-homotopy. In *Proceedings of spie conference* (Vol. 6969).

Daum, F., & Huang, J. (2010). Exact particle flow for nonlinear filters: Seventeen dubious solutions to a first order linear underdetermined PDE. In *Signal processing, sensor fusion, and target recognition XXII* (p. 64-71).

Daum, F., & Huang, J. (2011). Particle degeneracy: root cause and solution. In *Proceedings of spie conference* (Vol. 8050).

Daum, F., & Huang, J. (2013). Particle flow with non-zero diffusion for nonlinear filters. In *Proceedings of spie conference* (Vol. 8745).

George, A., & Powell, W. (2006). Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming. In *Journal of machine learning* (Vol. 65, p. 167-198). Kluwer Academic Publishers.

Jongerden, M., & Haverkort, B. (2009). Which battery model to use? In *Software, IET* (Vol. 15, p. 445-457).

Julier, S., & Uhlmann, J. (2004). Unscented filtering and

nonlinear estimation. In *Proceedings of the IEEE*.

Manwell, J. F., & McGowan, J. G. (1994). Extension fo the kinetic battery model for wind-hybrid power systems. In *Proceedings of EWEC*.

Oliva, J. A., Weihrauch, C., & Bertram, T. (2013). A model-based approach for predicting the remaining driving range in electric vehicles. In *Annual conference of the prognostics and health management society 2013.* (p. 438-448).

Orchard, M., & Vachtsevanos, G. (2010). A particle-filtering approach for on-line fault diagnosis and failure prognosis. In *Transactions of the institute of measurement and control* (Vol. 31, p. 221-246).

Restaino, R., & Zamboni, W. (2013). Rao-blackwellised particle filter for battery state-of-charge and parameters estimation. In *Industrial electronics society, iecon 2013 - 39th annual conference of the ieee* (p. 6783-6788).

Saxena, A., Celaya, J., Saha, B., Saha, S., & Goebel, K. (2009). On applying the prognostics performance metrics. In *Annual conference of the prognostics and health management society 2009.*

Tsiligkaridis, T., & Hero, A. (2013). Covariance estimation in high dimensions via kronecker product expansions. In *Signal processing, ieee transactions on* (Vol. 61, p. 5347-5360).

BIOGRAPHIES

Javier A. Oliva received his B.S. degree in Mechanical Engineering from the University Landivar in Guatemala in 2006 and his M.S. degree in Automation and Robotics from the Technische Universität Dortmund in 2010. His research interests include probabilistic methods, diagnosis and prognostics applied to electric vehicles. He is currently working as researcher at the Institute of Control Theory and Systems Engineering from the TU Dortmund in the area of driver assistance systems for electric vehicles.

Torsten Bertram is Professor at the Technische Universität Dortmund and he directs the Institute of Control Theory and Systems Engineering. He has carried out applied research in the areas of drive systems, service robotics and development methodology.