Recursive Hidden Input Estimation in Nonlinear Dynamic Systems with Varying Amounts of *a priori* Knowledge

Ulaş Güntürkün, James P. Reilly, *Member, IEEE,* Hubert deBruin *Member, IEEE,* and Thia Kirubarajan *Member, IEEE*

***Abstract*—In this article we address the estimation of additive driving–forces (e.g. hidden inputs) in nonlinear dynamic systems. We assume varying amounts of *a priori* knowledge on the underlying system and focus on three typical scenarios that are frequently faced in practice: 1) there is no sufficient prior knowledge to build a mathematical model of the underlying system; 2) the system is partially described by an analytical model; 3) a complete and accurate model of the underlying system is available.**

Arguably, the first scenario possesses the most common and the most challenging aplications in practice, where we have to retrieve the information of interest using only a set of output measurements. Motivated by this fact, our emphasis in this paper is on building and expanding upon the adaptive driving–force estimator (ADFE), which has been proposed in [1, 2] for the retrieval of driving–forces from everywhere–differentiable nonlinear systems with unknown dynamics.

The more optimistic second and the third estimation scenarios are addressed using some state–of–the–art techniques that are well–suited for each scenario respectively. In particular, the expectation–maximization (EM) algorithm is combined with a particle filter and used for estimating the missing data in the second sceniaro, and the Rao–Blackwellized particle filter is used for exploiting all the prior knowledge in an approximately optimum fashion for the third scenario.

In order to explore the usefulness of the ADFE, all techniques are compared on a nonlinear dynamic system in a unified manner. An equivalent amount of computational resources is provided for each technique for a fair comparison, and the qualities of the estimators are judged by the Posterior Cramer–Rao Lower Bound (PCRB). The experimental results reveal that the adaptive driving–force estimator is a favorable method for the estimation of rapidly–varying hidden inputs *unless* a complete and accurate model of the underlying system is available.

***Index Terms*—Driving–force estimation, hidden input estimation, reservoir computation, echo state network, expectation maximization, Rao–Blackwellized particle filter, floating–point operations, posterior Cramer–Rao lower bound (PCRB).**

I. Introduction

Many real–life dynamic systems generate nonstationary outputs due to the inevitable presence of measurement noise, dynamic noise and changes in the internal system parameters or environmental conditions in the course of time. These effects are likely to cause the reconstructions of the hidden states, driving–forces (e.g. hidden inputs, external perturbations) or some parameters of a system to be ill–posed in nature. As a consequence, finding regular solutions to such inverse problems is a common need for many applications in physical, engineering, and vital systems. The availability of sufficient prior knowledge on the underlying phenomenon can greatly facilitate solving such inverse problems. Specifically, the Bayesian inference regularizes the solution of inverse problems by virtue of exploiting all the prior knowledge on the system in a probabilistically optimum fashion. However, in many other cases, (i.e. voice production, brain activity, sea dynamics, an unknown multipath wireless medium, solar system, evolution of seismic waves, etc.) there may be too little or no prior knowledge and evidence to build a specific and accurate model of the underlying system for use of the Bayesian methods. The requirement in such cases is to reconstruct the quantities of interest using only a finite set of time series data without the availability of an analytic model.

Our interest is in the estimation of the driving–forces that perturb nonlinear dynamic systems. Those perturbation signals can be viewed as unknown inputs to the unperturbed dynamics, which are hidden from the observer most of the time if not always. To name a few examples of how the driving–forces in many physical phenomena effect the system dynamics, we can mention that —the mobility of transmit and/or receive terminals in a wireless channel; the effect of wind or small random elements on the sea surface for the evolution of the sea clutter; blinking of eye for the measurement of brain activity; and the development of abrupt or incipient faults in a nonlinear control system— play the role of perturbation inputs. The development of faults in nonlinear dynamic systems possibly deserves special attention since it is a good example of how the internal complexity of some dynamic systems could make it virtually impossible to devise an accurate model of the system. We will have more to say on the fault diagnosis problem as a future application in the concluding section.

In this article, we build and expand upon the two previous contributions [1, 2] where we proposed the adaptive driving–force estimator (ADFE). We showed in [1] that the ADFE provides a much better immunity against the measurement noise than a similar kind of estimation scheme presented in [3]. It is also shown in [1] that the ADFE is capable of reconstructing both slowly and rapidly varying forces that perturb the system in additive, multiplicative or even exponential forms up to a similar degree of accuracy. In [2],

U. Güntürkün is currently with the Telecommunications and Remote Sensing Laboratory at Catholic University of Louvain. James P. Reilly, Hubert deBruin and Thia Kirubarajan are with the Department of Electrical and Computer Engineering, McMaster University, Hamilton, Ontario, Canada. e-mail of the corresponding author: (gunturu@mcmaster.ca)

we demonstrated the performance of the ADFE on the radar scene analysis problem for the extraction of the signature of a small random target using texture modeling of sea clutter. Herein, we present deeper insight on the performance and the limitations of the ADFE and look at the hidden input estimation problem under a broader scope, studying three different estimation scenarios. Specifically, we raise and seek to answer the following questions: How accurately could the driving–forces be estimated if an analytic model of the underlying system were

1) fully available (i.e. the state–space pair and an evolutionary model of the driving–force are all known)?
2) partially available (i.e. only the state–space equations are known)?
3) not available at all?

The question (1) is addressed using the Rao–Blackwellized Particle Filter (RBPF), which provides a benchmark for the other techniques. The RBPF algorithm has been shown to be a favorable technique providing significantly better performance than a sampling importance resampling particle filter whenever it is possible to describe the evolution of the hidden quantity by a linear regression model [4–6]. Since any band–limited process can be represented as an autoregressive (AR) model, the RBPF is well–suited for the estimation of band–limited hidden forces. The RBPF assumes the coefficients of such an AR model as well as the state–space equations are known a priori.

The question (2) is explored by interpreting the problem from a missing–data perspective. To this end, the expectation maximization (EM) algorithm is combined with a particle filter (PF) to yield a method which we abbreviate as the EM–PF algorithm. The objective of the EM algorithm is to estimate the coefficients of the AR model mentioned above using a Kalman smoother (KS). As a simpler alternative to the EM algorithm, we also propose a random walk model to mimic the missing part of the model.

The adaptive driving–force estimator has been shown to be an adequate technique to address the (3)rd question given that the unknown system is governed by an everywhere–differentiable mapping and the signal–to–noise ratio is medium to high. Herein, we test the performance of the ADFE for the estimation of an abruptly–varying hidden input under the effect of undesired outliers.

The rest of this article is organized in the following order. In Section II, the adaptive driving–force algorithm is derived briefly, and its computational complexity is described in terms of *flops* (floating point operations). In Section III and Section IV, the RBPF and the EM–PF algorithms are specified respectively, and their computational complexities are also given in flop counts. Section V is devoted to the experiments conducted on a nonlinear dynamic system, in which the driving–force is an additive term exhibiting rapid variations. Section VI finalizes the paper with the concluding remarks with notes on the direction of the future research.

A. Problem Statement

In mathematical terms, we can state our problem as follows. Consider a (possibly nonlinear) dynamic system as described in (1),

$$\begin{aligned} x_{n+1} &= g\left(x_n, u_n\right) + \omega_{n+1} \\ y_{n+1} &= h(x_{n+1}) + \upsilon_{n+1}, \end{aligned} \tag{1}$$

where the continuous and measurable mapping $g : \mathbb{R}^2 \to \mathbb{R}$ defines the state transition, and $h : \mathbb{R} \to \mathbb{R}$ defines the evolution of the observables respectively. $x_n \in \mathbb{R}$ denotes the state, $u_n \in \mathbb{R}$ is the unknown input signal (driving–force), $y_n \in \mathbb{R}$ denotes the observable all at discrete time $n \in [0, \mathcal{T}-1]$, $\mathcal{T} \in \mathbb{N}$. $\omega_n \in \mathbb{R}$ is the dynamic noise, and $\upsilon_n \in \mathbb{R}$ is the measurement noise, both with zero mean and the variances σ_ω^2 and σ_υ^2 respectively. $\mathbb{R}$ denotes the real space. Note that (1) can be extended to cover the cases where the state is multidimensional without loss of generality.

In general terms, the problem that we are posing is similar to the state estimation problem, where the requirement is to estimate the state x_n. Our problem, however, is different: We wish to estimate the driving–force u_n without knowledge of $g(.)$ and $h(.)$, given that both mappings are differentiable everywhere, and $h(.)$ is invertible.

II. The Adaptive Driving–Force Estimator

A. A Brief Derivation

The basic idea underlying the adaptive driving–force estimator is to predict the observables one-step into the future using a bank of echo state networks (ESN)a reference is needed here in an online fashion, and then exploit the fact that the driving–force is not explicitly modeled by the predictor. Based on this simple idea, we relate the driving–force to the prediction error of the ESNs.

A predictive model of the unknown environment can be built by transforming the evolution of the observables y_n from the unknown system in (1) into a time series model. In such a model, the requirement for knowledge of the equations in (1), and the dependence of the observables on the unknown state is eliminated. For derivation, let us begin with re-arranging the second line of (1),

$$x_n = h^{-1}\left(y_n - \upsilon_n\right). \tag{2}$$

Now let us substitute (2) in the state equation in the first line of (1):

$$x_{n+1} = g\left(h^{-1}\left(y_n - \upsilon_n\right), u_n\right) + \omega_{n+1}. \tag{3}$$

If we substitute (3) in the second line of (1), we obtain the observable as

$$y_{n+1} = h\left(g\left(h^{-1}\left(y_n - \upsilon_n\right), u_n\right) + \omega_{n+1}\right) + \upsilon_{n+1}. \tag{4}$$

Let us denote the nested combination of mappings by $f(\alpha,\beta) \triangleq h\left(g\left(h^{-1}(\alpha),\beta\right)\right)$ with the aid of two dummy variables, α and β. Then, we can write

$$y_{n+1} \approx f\left(y_n, u_n\right) + v_{n+1}, \tag{5}$$

where the overall effect of the transformed and the additive noise processes is denoted by v_{n+1}. In (5), the observable y_{n+1} is expressed as a function of the previous observable y_n, and the unknown driving–force, u_n.

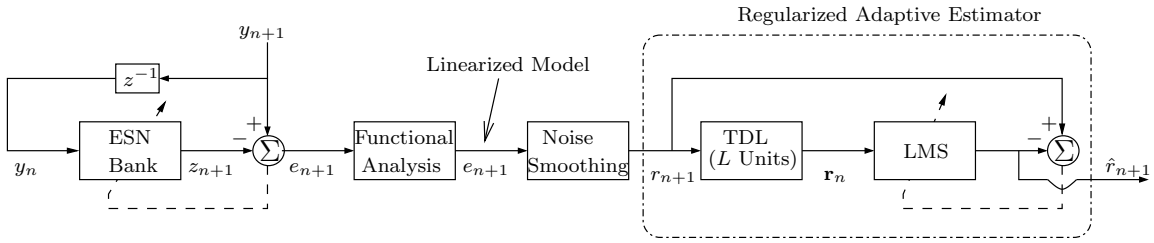

Fig. 1. Block diagram of the ADFE. The task of the functional analysis block is to provide a model in which the prediction error e_{n+1} is linearly dependent on the driving–force. z^{-1} denotes the unit delay operator. "TDL" denotes a tapped–delay line of L shift registers. "LMS" stands for the least mean squares algorithm. $\mathbf{r}_n \triangleq [r_n, \ldots, r_{n-L+1}]^T$.

Next, consider that we design an online nonlinear predictor (i.e. an ESN bank), which receives only the previous observable in its input and provides one-step approximations to the unknown system. Let z_{n+1} denote the output of the nonlinear predictor. Following from the universal approximation property of the recurrent neural networks, z_{n+1} can be described as

$$z_{n+1} = \widetilde{f}\,(y_n, 0)\,, \tag{6}$$

where y_n is input to the predictor, and z_{n+1} is obtained at its output. Note that the notation $\widetilde{f}\,(y_n, 0)$ implies that the driving–force is not provided to the neural network. Therefore, the absence of u_n at the neural network input is denoted by 0 as a second argument. The predictor's trainable parameters are updated at each time instant based on the arrival of a new observable. Then the objective is to extract the driving–force signal from the online prediction error $e_{n+1} = y_{n+1} - z_{n+1}$ based on the finite–difference approximation, given that the mapping f is differentiable everywhere:

$$\begin{aligned} e_{n+1} &= y_{n+1} - z_{n+1} && (7)\\ &= f\,(y\,(n-1)\,, u\,(n)) - \widetilde{f}\,(y_n, 0) + v_{n+1} && (8)\\ &\approx \frac{\partial f\,(y\,(n-1)\,, u\,(n))}{\partial u_n} u_n + \epsilon_{n+1}. && (9) \end{aligned}$$

In (9), ϵ_{n+1} denotes the overall approximation error. The final step of the algorithm is to obtain the refined driving–force estimates using a regularized adaptive filter based on (9). The resulting estimation algorithm encompasses the following three main operations:

1) Predictive modeling of the unknown system using ESNs,
2) Linearization of the 1-step prediction error as in (9),
3) Noise smoothing and regularized adaptive estimation of the hidden input from the linearized prediction error.

One iteration of the algorithm is depicted in the block diagram in Fig.1. The derivation of the ADFE is be elaborated in a supporting document [7], which is available at http://soma.ece.mcmaster.ca/~ulas/eng/studies/pub.htm.

B. Remarks

The basis of the ADFE algorithm is to approximate the observables in an online fashion using a bank of ESNs. An underlying assumption for accomplishing this objective is the smoothness of the hybrid mapping, $f(.)$ given in (4). Ulas–change this if you wish: The smoothness assumption implies that the driving force varies quickly relative to $\partial f(y_n, u_n)/\partial u_n$. The determination of the driving force when the system is unknown is an ill–posed problem. In our case, the resulting ambiguity is resolved by forcing a solution for which the corresponding derivative is slowly varying. Note however, that by observing (4), the noise processes can be greatly amplified by the nonlinear functions $g(.)$, $h(.)$, or $h^{-1}(.)$. This may harm the smoothness of $f(.)$, in which case the predictive modeling of the observables may fail.

The "noise smoothing" block in Fig. 1 is implemented as a recursive first–order low–pass filter, the purpose of which is to reduce the variance of the noise component of e_{n+1}. The "regularized adaptive estimator" realizes an adaptive forward prediction error filter, which adaptively tracks the desired slowly–varying component of the input r_{n+1} and smoothes out the noise component.

Consider the partial derivative term $\partial f(y_n, u_n)/\partial u_n$ in (9). Clearly, this term is model–dependent, and its smoothness can be also violated in case the underlying dynamics of the nonlinear system exhibit an irregular behavior. As an example, let us consider the system in (29). In this case, $\partial f(y_n, u_n)/\partial u_n = x_{n+1}/10$, and the process noise may have a harmful effect on the performance of the estimator.

In order to put the algorithm to test against these challenges, we will measure the algorithm's performance on varying levels of process and measurement noises which represent the undesired outliers in the experiments section. Then we will refer to the PCRB as a mathematically rigorous tool to assess the algorithm's performance under varying environmental conditions.

C. Computational Complexity of the ADFE

The ADFE algorithm is summarized in Algorithm 1, where the operations between the lines 1 and 6 pertain to the ESN bank. The bold face and capital letters denote $(N \times N)$ matrices, and bold face and small letters denote $(N \times 1)$ column vectors, N being the number of hidden units in one ESN. c denotes the connectivity rate within the hidden layer of the ESN, and Q denotes the number of ESNs in the bank. Line 7 denotes the averaging of the prediction error over all ESNs. Line 8 represents the noise–smoothing operation by an exponentially–decaying filter. The commands between the lines 9 and 14 embody the regularized adaptive predictor which is composed of L unit delay elements.

The corresponding computational complexity of the algorithm is calculated in terms of flops in the supporting

Algorithm 1 Adaptive Driving–Force Estimator

Require: Observables from the environment; $y_n, n = 0, ..., T$
Ensure: Driving–force estimates; $\hat{r}_n, t = 1, ..., T$

$for\ (n = 0; n < T; n++)\ \{$
$\quad for\ (i = 1; i \leq Q; i++)\ \{$

$$\mathbf{s}_{n+1}^{(i)} = \tanh\left(\mathbf{W}^{(i)}\mathbf{s}_n^{(i)} + \mathbf{w}^{in,(i)}y_n + \phi_{n+1}^{(i)}\right)$$
$$z_{n+1}^{(i)} = \left(\mathbf{w}_n^{out,(i)}\right)^T \mathbf{s}_{n+1}^{(i)}$$
$$e_{n+1}^{(i)} = y_{n+1} - z_{n+1}^{(i)}$$
$$\mathbf{k}_{n+1}^{(i)} = \frac{\tau^{-1}\mathbf{P}_n^{(i)}\mathbf{s}_{n+1}^{(i)}}{1 + \tau^{-1}\left(\mathbf{s}_{n+1}^{(i)}\right)^T \mathbf{P}_n^{(i)}\mathbf{s}_{n+1}^{(i)}}$$
$$\mathbf{w}_{n+1}^{out,(i)} = \mathbf{w}_n^{out,(i)} + \mathbf{k}_{n+1}^{(i)}e_{n+1}^{(i)}$$
$$\mathbf{P}_{n+1}^{(i)} = \tau^{-1}\mathbf{P}_n^{(i)} - \tau^{-1}\mathbf{k}_{n+1}^{(i)}\left(\mathbf{s}_{n+1}^{(i)}\right)^T \mathbf{P}_n^{(i)}$$

$\quad \}$

$$e_{n+1} = (1/Q)\sum_{i=1}^{Q} e_{n+1}^{(i)}$$
$$r_{n+1} = (1 - \Gamma)\, e_{n+1} + \Gamma r_n$$
$$\hat{r}_{n+1} = \mathbf{a}_n^T\mathbf{r}_n \quad \{\text{where } \mathbf{a}_n \triangleq \left[a_n^{(1)}, \ldots, a_n^{(L)}\right]^T \text{ and } \mathbf{r}_n \triangleq [r_n, \ldots, r_{n-L+1}]^T\}$$
$$\alpha_{n+1} = r_{n+1} - \hat{r}_{n+1}$$
$$\beta_{n+1} = \hat{r}_{n+1} - \hat{r}_n$$
$$\mathbf{a}_{n+1} = \mathbf{a}_n + \mu_n\left[\alpha_{n+1} - \lambda\beta_{n+1}\right]\mathbf{r}_n$$
$$\mu_{n+1} = \mu_n + \gamma\left[\alpha_{n+1} - \lambda\beta_{n+1}\right]\psi_n^T\mathbf{r}_n$$
$$\psi_{n+1} = \left[\mathbf{I} - (1 + \lambda)\mu_n\mathbf{r}_n\mathbf{r}_n^T\right]\psi_n + \left[\alpha_{n+1} - \lambda\beta_{n+1}\right]\mathbf{r}_n$$

$\}$

document, [7]. For the sake of brevity, only the final expression is given in (10). A flop is defined as one addition, subtraction, multiplication or division of two floating–point numbers. In (10), κ_1 and κ_2 indicate the number of flops per division and exponential operations respectively. A neutral flop count is estimated to be $\kappa_1 = 8$ for the division [1] and $\kappa_2 = 20$ for the exp function using the C source code [8]. The order of instructions in the derivation of the Algorithm 1 is optimized in such a way that the computationally least expensive solution is obtained. Even without this optimization, the rapidity of growth of the algorithm's complexity would not exceed cN^2 for a single ESN. Then we can conclude that the total complexity of the ADFE is $O(cQN^2)$ taking into account that $cQN^2 >> L$.

$$C_{ADFE}(c, Q, N, L) = \left[(2c + 5)N^2 + (2\kappa_1 + \kappa_2 + 12 - c)\, N + 1\right] Q + 11L + 10 + \kappa_1. \tag{10}$$

III. Rao–Blackwellized Particle Filter

In this section, we further specify the model in (1) to study the estimation of driving-forces that can be modeled as the additive components in the nonlinear state equations as given in (11)

$$\begin{aligned} x_{n+1} &= g(x_n) + u_{n+1} + \omega_{n+1} \\ y_{n+1} &= h(x_{n+1}) + \upsilon_{n+1}. \end{aligned} \tag{11}$$

[1] Since most processors can do an addition, comparison, or multiplication in a single cycle, those are all counted as one flop. But division always takes more than one cycle.

Since any band-limited discrete time stochastic process can be described by an autoregressive (AR) model of arbitrary order [9], a linear submodel can be devised to describe the dynamic evolution of u_n such that,

$$u_{n+1} = c_1 u_n + \ldots + c_{\mathsf{P}} u_{n-\mathsf{P}+1} + \zeta_{n+1} \tag{12}$$

where ζ_n is white Gaussian noise (WGN) that accounts for the model input, whose variance is σ_ζ^2. The AR(P) model in (12) can be expressed with a Markovian transition in the vector form. For this transformation, let us collect the P driving-force samples from time n to $n - \mathsf{P} + 1$ in a vector and define this vector as,

$$\mathbf{u}_n \triangleq [u_n, \ldots, u_{n-\mathsf{P}+1}]^T \in \mathbb{R}^{\mathsf{P}}. \tag{13}$$

We refer to the vector $\mathbf{u}_n$ as the linear state vector. $\mathbf{u}_n$ can be regressed on its previous state by a $\mathsf{P} \times \mathsf{P}$ regression matrix, which holds the model coefficients $\{c_{\mathsf{p}} | \mathsf{p} = 1, \ldots, \mathsf{P}\}$ in the following form

$$\mathbf{B} = \begin{bmatrix} c_1 & c_2 & \cdots & c_{\mathsf{P}-1} & c_{\mathsf{P}} \\ 1 & 0 & \ldots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}. \tag{14}$$

This enables us to re-arrange (11) as given in (15), in which both the linear and the nonlinear state variables are represented in dynamic Markovian sub-structures,

$$\begin{aligned} \mathbf{u}_{n+1} &= \mathbf{B}\mathbf{u}_n + \xi_{n+1} \\ x_{n+1} &= g(x_n) + \mathbf{b}\mathbf{u}_n + \omega_{n+1} \\ y_{n+1} &= h(x_{n+1}) + \upsilon_{n+1}. \end{aligned} \tag{15}$$

The $(1 \times \mathsf{P})$ vector $\mathbf{b}$ in the second line of (15) is defined as $\mathbf{b} = [1, 0, \ldots, 0]$. The $(\mathsf{P} \times 1)$ noise vector is given by $\xi_n \triangleq [\zeta_n, 0 \ldots, 0]^T$.

The authors of [4–6] demonstrate that the Rao-Blackwellized particle filter is a favorable technique for the estimation of linear state variables, whenever it is possible to partition the state equation into a linear and a nonlinear sub-model as in (15). The RBPF exploits the linear sub-structure by virtue of Rao-Blackwellization (or marginalization). In particular, the nonlinear state x_n is estimated with a particle filter, and the linear state $\mathbf{u}_n$ is then marginalized from the nonlinear state. This marginalization is operated by a Kalman filter (KF). Due to the linearity of the first and second lines of (15), and the Gaussianity of ξ_n and ω_n, the KF estimates of the linear states $\mathbf{u}_n$ are optimum. Therefore, the RBPF is an approximation to an optimum solution.. This explains why the RBPF provides superior performance over the standard particle filters by [4, 6].

For a more formal description of the RBPF, let $p(\mathbf{u}_n, x_n | \mathbf{y}_{1:n})$ denote the joint posterior of the linear and nonlinear state, which can be written in the form,

$$p(\mathbf{u}_n, x_n | \mathbf{y}_{1:n}) = p(\mathbf{u}_n | x_n)\, p(x_n | \mathbf{y}_{1:n}) \tag{16}$$

where we have made use of the fact that the measurements are independent of the linear state. Then the idea is to marginalize

the posterior density of the linear state from the joint posterior in (16) as follows,

$$p(\mathbf{u}_n|\mathbf{y}_{1:n}) = \int p(\mathbf{u}_n, x_n|\mathbf{y}_{1:n})\, dx_n \qquad (17)$$
$$= \int p(\mathbf{u}_n|x_n)\, p(x_n|\mathbf{y}_{1:n})\, dx_n.$$

$p(x_n|\mathbf{y}_{1:n})$ in the second line of (17) can be approximated by a particle filter. Specifically, let $\hat{p}(x_n|\mathbf{y}_{1:n})$ denote the particle filter approximation to $p(x_n|\mathbf{y}_{1:n})$, which can be described as given in (18),

$$\hat{p}(x_n|\mathbf{y}_{1:n}) = \sum_{i=1}^{M} w^{(i)} \delta\left(x_n - x_n^{(i)}\right) \qquad (18)$$

where M denotes the number of particles. Substituting (18) in the second line of (17), the approximate posterior density for the linear state is obtained as

$$\hat{p}(\mathbf{u}_n|\mathbf{y}_{1:n}) = \sum_{i=1}^{M} w^{(i)} p\left(\mathbf{u}_n|x_n^{(i)}\right). \qquad (19)$$

The conditional mean estimates of the linear state variable are approximated from (19) in the following form:

$$\mathbb{E}[\mathbf{u}_n|\mathbf{y}_{1:n}] \approx \int \mathbf{u}_n \hat{p}(\mathbf{u}_n|\mathbf{y}_{1:n})\, d\mathbf{u}_n \qquad (20)$$
$$= \sum_{i=1}^{M} w^{(i)} \mathbb{E}\left[\mathbf{u}_n|x_n^{(i)}\right],$$

where $\mathbb{E}[.]$ denotes the statistical expectation operation. $\mathbb{E}\left[\mathbf{u}_n|x_n^{(i)}\right]$ is computed using a Kalman filter exploiting the linear Gaussian substructure encompassing the first and the second lines of (15). The overall implementation of the Rao-Blackwellized particle filter is provided in a pseudo-code in Algorithm 2, where the nonlinear state is estimated with a standard particle filter. Hence, the importance function is set equal to the transition prior, and the systematic resampling is used at each iteration. The filter's computational complexity is derived from the Algorithm 2 as elaborated in the supporting material, [7].

Before explaining the steps of Algorithm 2, let us introduce the following notation for the estimated quantities, which will be commonly used in the sequel:

$\hat{\mathbf{u}}_{n|n}$: filtered estimates of the vector $\mathbf{u}_n$, and $\mathbf{C}_{n|n}$: filtering covariance for $\hat{\mathbf{u}}_{n|n}$,

$\hat{\mathbf{u}}_{n+1|n}$: one-step predicted estimates of the vector $\mathbf{u}_n$, and $\mathbf{C}_{n+1|n}$: prediction covariance for $\hat{\mathbf{u}}_{n+1|n}$,

Now let us have a closer look at the steps of the Algorithm 2. The first and the second tabs in the line 3 in Algorithm 2 are the Kalman prediction equations, where $\mathbf{R}$ stands for the $\mathsf{P}\times\mathsf{P}$ covariance matrix for the evolution uncertainty for the linear state vector, whose only nonzero entry is its first element, σ_ζ^2. For instance, letting $\mathsf{P} = 2$, we have

$$\mathbf{R} = \begin{bmatrix} \sigma_\zeta^2 & 0 \\ 0 & 0 \end{bmatrix}. \qquad (21)$$

The first tab of the line 4 denotes the sampling of particles from the transition prior, whereas the second tab of line 4 explains the derivation of the pseudo-measurements obtained from the nonlinear state equation.

The Kalman gain vector at time n is denoted Υ_n at line 6, which is a $\mathsf{P}\times 1$ column vector.

Line 7 describes the Kalman measurement update equations. The estimate of the linear state vector associated with the (i)th particle is denoted $\hat{\mathbf{u}}_{n|n}^{(i)}$ as given in the first tab of line 7. $\mathbf{C}_{n|n}$ in the second tab of line 7 denotes the $\mathsf{P}\times\mathsf{P}$ covariance matrix for the estimation error for $\hat{\mathbf{u}}_{n|n}^{(i)}$. Note that each particle is associated with a Kalman filter. However, the same covariance matrix is propagated for all Kalman filters, since the measurement equation is independent of the linear state vector. This greatly reduces the complexity of the RBPF.

Finally, the desired conditional mean estimates of the linear state vector are obtained by averaging over all particles as observed in line 8 since the importance weights have already been normalized. The filtered driving-force estimate, $\hat{u}_n$ equals the first element of the vector $\hat{\mathbf{u}}_{n|n}$.

Algorithm 2 Rao–Blackwellized Particle Filter

Require: Observables from the environment $\{y_n|n = 1,\ldots,\mathcal{T}\}$ and an analytical description of the underlying mechanism (e.g. (15))

Ensure: Conditional mean estimates of the linear state vector; $\hat{\mathbf{u}}_{n|n}$ for $n = 1,\ldots,\mathcal{T}$

Initialize particles, $i = 1,\ldots,M$; $x_{0|-1}^{(i)} \sim p(x_0)$ and set $\mathbf{u}_{0|-1}^{(i)} = \mathbf{u}_0^{(i)}$, and $\mathbf{C}_{0|-1}^{(i)} = \mathbf{C}_0$

for $(n = 0; n < \mathcal{T}; n++)$ {

for $(i = 1; i \le M; i++)$ {

1: Evaluate the importance weights $w_n^{(i)} = p\left(y_n|x_n^{(i)}\right)$ and normalize $\tilde{w}^{(i)} = w^{(i)} / \sum_{j=1}^{M} w_n^{(j)}$

2: Resample M particles with replacement, $\mathbb{P}\left(x_{n|n}^{(i)} = x_{n|n-1}^{(j)}\right) = \tilde{w}_n^{(j)}$

3: State prediction and state prediction covariance:

- $\hat{\mathbf{u}}_{n+1|n}^{(i)} = \mathbf{B}\hat{\mathbf{u}}_{n|n}^{(i)}$
- $\mathbf{C}_{n+1|n} = \mathbf{B}\mathbf{C}_{n|n}\mathbf{B}^T + \mathbf{R}$

4: For $i = 1,\ldots,M$,

- New particle prediction $x_{n+1|n}^{(i)} \sim p\left(x_{n+1|n}|x_n^{(i)}\right) = \mathcal{N}\left(g\left(x_n^{(i)}\right) + \mathbf{b}\hat{\mathbf{u}}_{n|n}^{(i)}, \mathbf{b}\mathbf{C}_{n|n}\mathbf{b}^T + \sigma_\omega^2\right)$
- Measurement prediction $m_{n+1}^{(i)} = x_{n+1}^{(i)} - g\left(x_n^{(i)}\right)$

5: $\Omega_{n+1} = \mathbf{b}\mathbf{C}_{n+1|n}\mathbf{b}^T + \sigma_\omega^2$ {Innovation covariance}

6: $\Upsilon_{n+1} = \mathbf{C}_{n+1|n}\mathbf{b}^T\Omega_{n+1}^{-1}$ {Kalman gain}

7: Updated state estimate and updated state covariance:

- $\hat{\mathbf{u}}_{n+1|n+1}^{(i)} = \hat{\mathbf{u}}_{n+1|n}^{(i)} + \Upsilon_{n+1}\left(m_{n+1}^{(i)} - \mathbf{b}\hat{\mathbf{u}}_{n+1|n}^{(i)}\right)$
- $\mathbf{C}_{n+1|n+1} = \mathbf{C}_{n+1|n} - \Upsilon_{n+1}\Omega_{n+1}\Upsilon_{n+1}^T$

}

8: Desired estimates: $\hat{\mathbf{u}}_{n|n} = (1/M)\sum_{i=1}^{M} \hat{\mathbf{u}}_{n|n}^{(i)}$ shouldn't this be $(1/M)\sum_{i=1}^{M} w_n^{(i)}\hat{\mathbf{u}}_{n|n}^{(i)}$? (to agree with (19) and (20)

}

What about mentioning an explicit flop count for RBPF?

IV. Expectation Maximization – Particle Filter (EM–PF)

The Rao-Blackwellized particle filter is intended for such applications where the exact knowledge of the system equa-

tions (e.g. (15)) is available. However, in some applications, a model of the underlying dynamic system may be only partially available [4, 10, 11]. In this section, we consider such cases where the nonlinear state equation and the measurement equation are available, yet the evolution of the linear state is hidden from the observer. We propose to use the expectation maximization (EM) [12] algorithm for such estimation scenarios, where the hidden states have to be inferred from incomplete data. The objective of the EM algorithm is to parameterize the missing information, and obtain maximum likelihood (ML) estimates of those parameters.

The driving-force estimator based on the EM algorithm is embodied by the following two main operations:

1) Particle Filtering: using the readily available system equations (i.e. the second and the third lines of (15)), and the set of measurements $\mathbf{y}_{1:n}$, the nonlinear state x_n in (15) is estimated with a particle filter. The current estimates of u_n provided by the EM algorithm are also substituted in (15).
2) Expectation Maximization: The EM algorithm operates on a sub– state-space model, which consists of
   - a state equation, (i.e. the first line of (15)), whose parameters are missing. This means that the first line of (15) is treated as an unknown equation by the EM, whose parameters are to be estimated;
   - a measurement equation, (i.e. the second line of (15)), in which the estimates of x_n provided by the particle filter are treated as the measurements.

Following from this brief description, we abbreviate the resulting method as the "EM–PF". The operation of the EM–PF method is illustrated on a block diagram as given in Fig.2. The EM–PF algorithm is proposed a doubly–iterative method that encompasses an inner and an outer loop of iterations. The inner loop consists of K EM iterations over the entire dataset. After the completion of one inner loop, the outer loop is re-initiated by running the particle filter all over again to exploit the updated estimates of u_n provided by the EM algorithm. Thus, the available data vector $\mathbf{m}$ is updated by the particle filter, and provided to the E-step of the EM algorithm.

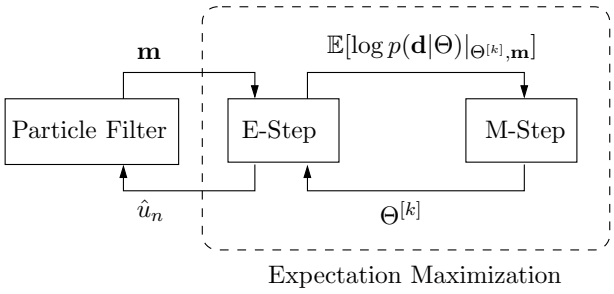

Fig. 2. Graphical representation of the EM–PF method. The outer iteration loop includes the exchange of data between the particle filter and the E-step of the EM algorithm. The inner iteration loop represents the exchange of data between the E and the M-steps of the EM algorithm.

For the particle filtering operations, we use a standard particle filter where systematic resampling is used at each iteration and the importance density is set to the prior. Now, let us elaborate on the EM method by starting with the construction of the sub– state-space pair given in (22), which is built on the first and the second lines of (15):

$$\begin{aligned} \mathbf{u}_n &= \mathbf{B}\mathbf{u}_{n-1} + \xi_n \\ m_n &= \mathbf{b}\mathbf{u}_n + \omega_n. \end{aligned} \tag{22}$$

In (22), we have defined $m_n \triangleq x_n - g(x_{n-1})$, which are treated as the pseudo-measurements for the EM algorithm. Note that the model in (22) is suitable for the operation of a Kalman smoother (KS), which is needed to perform the expectation (E) step of the EM algorithm as will be explained soon. In the M-step, the log-likelihood of the complete data is maximized with respect to the unknown coefficients. Since the unknown regression coefficients $\{c_\mathsf{p}|\mathsf{p} = 1, \ldots, \mathsf{P}\}$ occupy only the first row of the matrix $\mathbf{B}$, performing the M-step over the model in (22) would be wasteful of computational resources. As an alternative, we simplify (22) to obtain (23), where we deal with the $1\times\mathsf{P}$ vector of unknown coefficients ($\mathbf{c} \triangleq [c_1, \ldots, c_\mathsf{P}]$) instead of the $\mathsf{P}\times\mathsf{P}$ matrix, $\mathbf{B}$:

$$\begin{aligned} u_n &= \mathbf{c}\mathbf{u}_{n-1} + \zeta_n \\ m_n &= u_n + \omega_n. \end{aligned} \tag{23}$$

Consequently, the E-step of the EM algorithm operates on (22), whereas the M-step exploits (23).

Imagine that the conditional mean estimates of the nonlinear state x_n are obtained with a particle filter from time $n = 1$ to $n = \mathcal{T}$. For the sake of brevity, let us assume that the particle filter estimates are arbitrarily close to the true states. Then let us collect the pseudo-measurements in a $\mathcal{T}\times 1$ vector such that $\mathbf{m} \triangleq [m_1, \ldots, m_\mathcal{T}]^T$, which represents the available data. Let us also concatenate the entire history of the driving-force samples in another $\mathcal{T}\times 1$ vector, $\mathbf{U} \triangleq [u_1, \ldots, u_\mathcal{T}]^T$. The vector $\mathbf{U}$ represents the missing data which are associated with the hidden parameters, $\Theta \triangleq [c_1, \ldots, c_\mathsf{P}, \sigma_\zeta^2]^T$. The complete data are represented as a concatenation of the missing and the available data. Let the vector $\mathbf{d}$ denote the complete data, which is defined as

$$\mathbf{d} \triangleq [\mathbf{U}^T, \mathbf{m}^T]^T. \tag{24}$$

The E and the M-steps of the EM algorithm are iterated until convergence. In the E-step, the expectation of the log-likelihood of complete data is simulated given the available data $\mathbf{m}$, and the current estimates of the hidden parameters, $\Theta^{[k]}$. In the M-step, the expected log-likelihood of the complete data is maximized with respect to the hidden parameter vector. This maximization results in an updated ML estimate of Θ.

Let us describe the log-likelihood of the complete data as

$$\begin{aligned} \log p\left(\mathbf{d}|\Theta\right) &= \log p\left(\mathbf{U}, \mathbf{m}|\Theta\right) \\ &= \log p\left(\mathbf{U}|\Theta\right) + \log p\left(\mathbf{m}|\mathbf{U}, \Theta\right). \end{aligned} \tag{25}$$

Exploiting the linearity of our sub-model in (23) and using the whiteness and Gaussianity of ζ_n and ω_n, the second line

of (25) can be expressed as

$$\log p\left(\mathbf{d}|\Theta\right) = \log p\left(u_0\right) - \frac{\mathcal{T}}{2}\log\sigma_\zeta^2 - \mathcal{T}\log 2\pi - \frac{\mathcal{T}}{2}\log\sigma_\omega^2 - \frac{1}{2\sigma_\zeta^2}\sum_{n=1}^{\mathcal{T}}\left(u_n - \mathbf{c}^T\mathbf{u}_{n-1}\right)^2 - \frac{1}{2\sigma_\omega^2}\sum_{n=1}^{\mathcal{T}}\left(m_n - u_n\right)^2. \tag{26}$$

Using (26), we are ready to elaborate on the E and the M-steps:

- E-step: Let $\Theta^{[k]}$ be an estimate of the parameters at the kth EM iteration. Then for the E-step, we compute the expectation of the log-likelihood in (26) given the available data $\mathbf{m}$, and our current parameter estimate, $\Theta^{[k]}$:

$$\begin{aligned} E\left(\Theta|\Theta^{[k]}\right) &\triangleq \mathbb{E}\left[\log p(\mathbf{d}|\Theta)|_{\mathbf{m},\Theta^{[k]}}\right] \\ &= o_1 - \frac{\mathcal{T}}{2}\log\sigma_\zeta^2 - \frac{1}{2\sigma_\zeta^2}\sum_{n=1}^{\mathcal{T}}\mathbb{E}\left[(u_n)^2|_{\mathbf{m},\Theta^{[k]}}\right] - 2\mathbf{c}^T\mathbb{E}\left[u_n\mathbf{u}_{n-1}|_{\mathbf{m},\Theta^{[k]}}\right] \\ &\quad + \mathbf{c}^T\mathbb{E}\left[\mathbf{u}_{n-1}\mathbf{u}_{n-1}^T|_{\mathbf{m},\Theta^{[k]}}\right]\mathbf{c} - \frac{1}{2\sigma_\omega^2}\sum_{n=1}^{\mathcal{T}}\mathbb{E}\left[(m_n - u_n)^2|_{\mathbf{m},\Theta^{[k]}}\right] \end{aligned} \tag{27}$$

  where we have defined the constant term as $o_1 \triangleq \log p\left(u_0\right) - \mathcal{T}\log 2\pi - \frac{\mathcal{T}}{2}\log\sigma_\omega^2$. As noted before, the E-step operates on the model in (22), and the indicated expectations in (26) are approximated using a KS. For instance, in order to compute the term $\mathbb{E}\left[(u_n)^2|_{\mathbf{m},\Theta^{[k]}}\right]$, we first compute $\mathbb{E}[\mathbf{u}_n\mathbf{u}_n^T|_{\mathbf{m},\Theta^{[k]}}]$ using a KS. Then $\mathbb{E}\left[(u_n)^2|_{\mathbf{m},\Theta^{[k]}}\right]$ is retrieved from the first element of $\mathbb{E}[\mathbf{u}_n\mathbf{u}_n^T|_{\mathbf{m},\Theta^{[k]}}]$. The operation of the KS is described in full detail in line 3 of the Algorithm 3.
- M-step: We find the value of Θ that maximizes $E\left(\Theta|\Theta^{[k]}\right)$, which becomes the next estimate of the missing parameters,

$$\Theta^{[k+1]} = \arg\max_{\Theta} E\left(\Theta|\Theta^{[k]}\right) \tag{28}$$

  where $E\left(\Theta|\Theta^{[k]}\right)$ is as given in (27). The details and the outcome of the maximization operation are provided in the line 4 of the Algorithm 3.

As opposed to the Adaptive Driving–Force Estimator and the Rao–Blackwelllized particle filter, the EM–PF method is based on batch processing. This is due to the use of a Kalman smoother in the E-step for approximating the second-order quantities indicated in the expectations in (27). It should be noted that replacing a Kalman smoother by a Kalman filter yields a significant drop in the algorithm's performance. As a side product of the E-step, the smoothed driving-force estimates can also be retrieved from the KS, which is the case in our application. That being said, the following notation is introduced for the smoothed estimates:

$\hat{\mathbf{u}}_{n|\mathcal{T}}$: fixed-interval smoothed estimates of the vector $\mathbf{u}_n$ and $\mathbf{C}_{n+1|\mathcal{T}}$: smoothing covariance for $\hat{\mathbf{u}}_{n|\mathcal{T}}$. The fixed-interval smoothing implies that all the data available from time 1 to $\mathcal{T}$ are utilized to estimate the vector $\mathbf{u}_n$.

The entire realization of the EM–PF is given in a pseudo-code in Algorithm 3. The total number of EM iterations in Algorithm 3 is denoted by K. Although there does not exist an analytical way of finding the optimum value of K, in practice, we have observed that more than $4-6$ EM iterations do not lead to a significant improvement in the estimates of u_n. These results match with the observations of other researchers in the literature [13, 14].

include explicit flop count for EMPF?

V. Experiments

A. Experimental Setup

We consider the nonlinear stochastic model in (29) for the experiments, which was originally studied for Bayesian sequential state estimation in a severely nonlinear system in [15]. Our objective in this section is to estimate u_n in (29).

$$\begin{aligned} x_{n+1} &= 0.5x_n + \frac{25x_n}{1+x_n^2} + u_n + \omega_{n+1} \\ y_{n+1} &= \frac{x_{n+1}^2}{20} + \upsilon_{n+1}. \end{aligned} \tag{29}$$

The estimators are specified for the varying observability conditions in the following manner:

1) Full prior knowledge: For the application of the RBPF within this case, we construct u_n in (12) to be a 2nd–order autoregressive process (i.e. $\mathsf{P} = 2$ in (13)). Then the regression coefficients $[c_1, c_2]^T$ in (12) are calculated using the covariance method so as to minimize the forward prediction error in the least squares sense.[2] Setting $\mathsf{P} = 2$ is sufficient since it leads to a negligibly small variance of modeling error σ_ξ^2. Using $\mathsf{P} > 2$ does not lead to a significant decrease in σ_ξ^2. This application will be referred to as RBPF-AR(2), for which the model and all parameters are assumed known.
2) Partial prior knowledge:
   - In this case, the first method we propose is to apply the RBPF without the availability of the model coefficients, $[c_1, \ldots, c_\mathsf{P}]^T$. Then a natural way to proceed with the RBPF is to use a random walk model [10] for the evolution of the driving-force. This filter is abbreviated as the RBPF-RW.
   - The second method we propose for this estimation situation is the EM-PF algorithm. The objective of the EM algorithm is to estimate the coefficient vector $[c_1, \ldots, c_\mathsf{P}]^T$.
3) Finally, for such cases where there is no prior knowledge on the system dynamics, the proposed ADFE is invoked without the knowledge of (29).

The availability of *a priori* knowledge for each filter is summarized in Table I.

Performances of all estimators mentioned above will be evaluated with a reference to the PCRB, which is calculated in a recursive manner following from [16].

B. Performance Results at Minimal Complexity

Our objective in the first part of the experiments is to realize all of the methods mentioned above at the minimal computational cost. To this end, we first determine the minimum

[2]The command `arcov` is readily available in Matlab for this operation.

Algorithm 3 The Expectation Maximization – Particle Filter

Require: Observables from the environment $\{y_n | n = 1, \ldots, \mathcal{T}\}$ and an analytical description of the system (e.g. (22))

Ensure: Estimates of the missing parameters Θ and the smoothed conditional mean estimates of the driving-force; $\hat{u}_n$ for $n = 1, \ldots, \mathcal{T}$

Initialize particles, $i = 1, \ldots, M$; $x_{0|-1}^{(i)} \sim p(x_0)$ and set $\mathbf{u}_{0|-1}^{(i)} = \mathbf{u}_0^{(i)}$, and $\mathbf{C}_{0|-1}^{(i)} = \mathbf{C}_0$

Initialize the estimate for the missing parameters: $\Theta^{[0]}$

$for\ (k = 0; k < K; k++)$ {

$for\ (n = 0; n < \mathcal{T}; t++)$ {

$for\ (i = 1; i \leq M; i++)$ {

1: Evaluate the importance weights $w_n^{(i)} = p\left(y_n | x_n^{(i)}\right)$ and normalize $\tilde{w}^{(i)} = w^{(i)} / \sum_{j=1}^{M} w_n^{(j)}$

2: Res. M particles with replacement, $\mathbb{P}\left(x_{n|n}^{(i)} = x_{n|n-1}^{(j)}\right) = \tilde{w}_n^{(j)}$

3: E-STEP (Kalman smoother):

1) Forward Recursions:

State prediction and state prediction covariance:

- $\hat{\mathbf{u}}_{n+1|n}^{(i)} = \mathbf{B}^{[k]} \hat{\mathbf{u}}_{n|n}^{(i)}$
- $\mathbf{C}_{n+1|n} = \mathbf{B}^{[k]} \mathbf{C}_{n|n} \left(\mathbf{B}^{[k]}\right)^T + \mathbf{R}^{[k]}$
- Particle prediction $x_{n+1|n}^{(i)} \sim p\left(x_{n+1|n} | x_n^{(i)}\right) = \mathcal{N}\left(g\left(x_n^{(i)}\right) + \mathbf{b}\hat{\mathbf{u}}_{n|n}^{(i)}, \mathbf{b}\mathbf{C}_{n|n}\mathbf{b}^T + \sigma_\omega^2\right)$
- Measurement prediction $m_{n+1}^{(i)} = x_{n+1}^{(i)} - g\left(x_n^{(i)}\right)$
- $\zeta_{n+1} = \mathbf{b}\mathbf{C}_{n+1|n}\mathbf{b}^T + \sigma_\omega^2$ {Innovation covariance}
- $\Omega_{n+1} = \mathbf{C}_{n+1|n}\mathbf{b}^T \zeta_{n+1}^{-1}$ {Kalman gain}

Updated state estimate and state covariance:

- $\hat{\mathbf{u}}_{n+1|n+1}^{(i)} = \hat{\mathbf{u}}_{n+1|n}^{(i)} + \Omega_{n+1}\left(m_{n+1}^{(i)} - \mathbf{b}\hat{\mathbf{u}}_{n+1|n}^{(i)}\right)$
- $\mathbf{C}_{n+1|n+1} = \mathbf{C}_{n+1|n} - \Omega_{n+1}\zeta_{n+1}\Omega_{n+1}^T$

} }

2) Backward Recursions:

$for\ (n = \mathcal{T}; n \geq 0; n--)$ {

$for\ (i = 1; i \leq M; i++)$ {

- $\mathbf{P}_n = \mathbf{C}_{n|n}\left(\mathbf{B}^{[k]}\right)^T \mathbf{C}_{n+1|n}^{-1}$
- $\hat{\mathbf{u}}_{n|\mathcal{T}}^{(i)} = \hat{\mathbf{u}}_{n|n}^{(i)} + \mathbf{P}_n\left(\hat{\mathbf{u}}_{n+1|\mathcal{T}}^{(i)} - \hat{\mathbf{u}}_{n+1|n}^{(i)}\right)$
- $\mathbf{C}_{n|\mathcal{T}} = \mathbf{C}_{n|n} + \mathbf{P}_n\left(\mathbf{C}_{n+1|\mathcal{T}} - \mathbf{C}_{n+1|n}\right)$

}

- $\hat{\mathbf{u}}_{n|\mathcal{T}} = (1/M)\sum_{i=1}^{M} \hat{\mathbf{u}}_{n|\mathcal{T}}^{(i)}$ {Desired estimates}
- $\hat{\mathbf{u}}_{n-1|\mathcal{T}} = (1/M)\sum_{i=1}^{M} \hat{\mathbf{u}}_{n-1|\mathcal{T}}^{(i)}$
- $\mathbb{E}\left[\hat{\mathbf{u}}_n (\hat{\mathbf{u}}_n)^T\right] = \mathbf{C}_{n|\mathcal{T}} + \hat{\mathbf{u}}_{n|\mathcal{T}} (\hat{\mathbf{u}}_{n|\mathcal{T}})^T$
- $\mathbb{E}\left[\hat{\mathbf{u}}_{n-1} (\hat{\mathbf{u}}_{n-1})^T\right] = \mathbf{C}_{n-1|\mathcal{T}} + \hat{\mathbf{u}}_{n-1|\mathcal{T}} (\hat{\mathbf{u}}_{n-1|\mathcal{T}})^T$
- $\mathbb{E}\left[\hat{\mathbf{u}}_n (\hat{\mathbf{u}}_{n-1})^T\right] = \mathbf{P}_{n-1}\mathbf{C}_{n|\mathcal{T}} + \hat{\mathbf{u}}_{n|\mathcal{T}} (\hat{\mathbf{u}}_{n-1|\mathcal{T}})^T$

}

4: M-STEP:

$$\left(\mathbf{c}^{[k+1]}\right)^T = \sum_{n=1}^{\mathcal{T}} \mathbb{E}\left[u_n (\hat{\mathbf{u}}_{n-1})^T |_{\mathbf{m},\Theta^{[k]}}\right] \left(\sum_{n=1}^{\mathcal{T}} \mathbb{E}\left[\hat{\mathbf{u}}_{n-1} (\hat{\mathbf{u}}_{n-1})^T |_{\mathbf{m},\Theta^{[k]}}\right]\right)^{-1}$$

$$\sigma_\xi^{2,[k+1]} = \frac{1}{\mathcal{T}} \sum_{n=1}^{\mathcal{T}} \left(\mathbb{E}\left[(u_n)^2\right] - 2\left(\mathbf{c}^{[k]}\right)^T \left(\mathbb{E}\left[u_n (\hat{\mathbf{u}}_{n-1})^T\right]\right)^T + \left(\mathbf{c}^{[k]}\right)^T \mathbb{E}\left[\hat{\mathbf{u}}_{n-1} (\hat{\mathbf{u}}_{n-1})^T\right] \mathbf{c}^{[k]} |_{\mathbf{m},\Theta^{[k]}}\right)$$

- Allocate $\left(\mathbf{c}^{[k+1]}\right)^T$ into the first row of $\mathbf{B}^{[k+1]}$ and $\sigma_\xi^{2,[k+1]}$ into the first element of $\mathbf{R}^{[k+1]}$

}

TABLE I

Availability of the system equations to the estimation methods. The ✓ and × signs indicate the availability and unavailability of a particular line of the system equations respectively to the corresponding filters.

Augmented System	RBPF-AR(P)	EM-PF& RBPF-RW	ADFE
$\mathbf{u}_{n+1} = \mathbf{B}\mathbf{u}_n + \xi_{n+1}$	✓	×	×
$x_{n+1} = 0.5x_n + \frac{25x_n}{1+x_n^2} + \mathbf{b}\mathbf{u}_n + \omega_{n+1}$	✓	✓	×
$y_{n+1} = \frac{x_{n+1}^2}{20} + \upsilon_{n+1}$	✓	✓	×

complexity of the EM–PF method since it is computationally the most demanding filter among the others. Specifically, using 50 particles and 5 EM iterations leads to a reasonable performance for the EM–PF with a model order of $\mathsf{P} = 2$. Increasing the computational resources does not lead to a remarkable increase in the performance for the EM–PF. With this setting, the EM–PF algorithm is realized at approximately 37.10^3 flops. In Table II, this setup is represented by $C_{EMPF}(50, 2, 5)$. Then we realize the other estimators at a similar level of computational cost as presented in Table II, where the theoretical and simulated flop counts[3] are given for each estimator. The discrepancy between the theoretical and simulated flop counts in Table II is less than 3% for all estimators. We proceed to the experiments with the settings

TABLE II

Theoretical and simulated computational complexity of the driving–force estimation methods. The EM–PF is realized by 50 particles with a model order of $\mathsf{P} = 2$ and 5 EM iterations. The RBPF–AR(2) is realized with 275 particles with $\mathsf{P} = 2$. The RBPF–RW is realized with 300 particles, with $\mathsf{P} = 1$ (i.e. the random walk). The ADFE has 34 ESNs in the ESN bank, each having 30% connected dynamic reservoirs with 10 neurons. The regularized LMS estimator has 100 tap–weights.

Filter	Realization	Flop Count	
		Theoretical	Simulated
EM–PF	$C_{EMPF}(50, 2, 5)$	37125	37594
RBPF–AR(2)	$C_{RBPF}(275, 2)$	36983	37812
RBPF–RW	$C_{RBPF}(300, 1)$	36980	38183
ADFE	$C_{ADFE}(0.3, 34, 10, 100)$	36410	37337

given in the second column of Table II.

An example run of the ADFE algorithm with the complexity $C_{ADFE}(0.3, 34, 10, 100)$ is illustrated in Fig.3, which demonstrates the accuracy of the proposed method for the reconstruction of a rapidly varying non-smooth driving–force. For the sake of visuality, only the last 50 samples are shown in Fig.3.

As stated in Section II-B, we conduct our experiments for varying environmental conditions which are represented by the changes in the model uncertainty and sensor noise in the model (29). Let P_u denote the power of the driving–force signal. We vary the ratio of P_u to the variance of the dynamic noise in the range $P_u/\sigma_\omega^2 \in \{-10, 10\}$ dB with 5 dB increments in order to test the performance of the estimators for varying level of unstructured outliers or model uncertainty. We consider two sensor noise levels. For the first case, we consider a low–noise sensor, and set $\sigma_\upsilon^2 = 0.05$. Then we repeat our experiments for

[3]The flops are simulated using the `lightspeed toolbox` [8], which leads to much more accurate flop estimates than the Matlab 5.3.

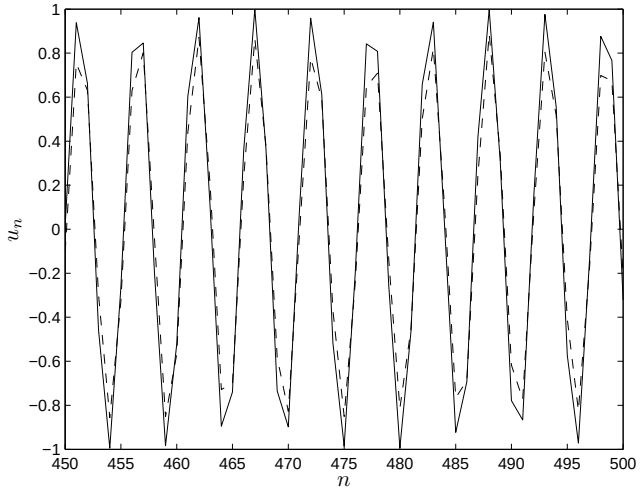

Fig. 3. An example run of the ADFE. The solid curve denotes the actual driving-force, u_n. The dashed curve is the estimate provided by the ADFE, which are obtained by a single example run of the ADFE algorithm. The ADFE is realized with the values and the corresponding complexity given in Table II. $\sigma_\omega^2 = 1.6$ (i.e. $P_u/\sigma_\omega^2 = -5$ dB) and $\sigma_\upsilon^2 = 0.05$.

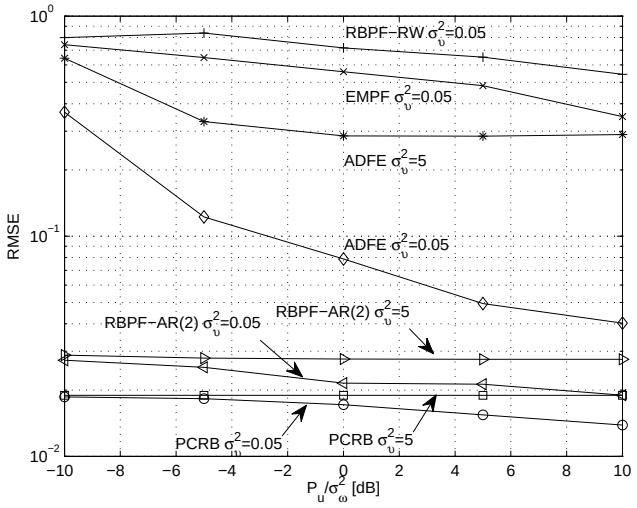

Fig. 4. Performances of the Rao–Blackwellized Particle Filter, Expectation–Maximization, and the Adaptive Driving–Force Estimator on the estimation of hidden inputs. All filters are realized with a computational cost at approximately $C = 37.10^3$ flops. The y–axis is the root mean square error (RMSE), while the x–axis is the ratio of the power of the driving–force to the variance of dynamic noise in dB.

the same range of P_u/σ_ω^2 with a high–noise sensor model by setting $\sigma_\upsilon^2 = 5$.

The results are shown in Fig.4, where the Root Mean–Square Error (RMSE) is plotted with respect to $P_u/\sigma_\omega^2 \in \{-10, 10\}$ dB for all estimators and the Posterior Cramer–Rao Lower Bound (PCRB) for two different measurement noise levels. The EM algorithm has been simulated by 100 independent initializations of the regression coefficients such that $[c_1, c_2]^T \in [-1, 1]$ and the average of those trials is calculated. Comparing the performances of RBPF–RW and EM–PF with the ADFE, it is clear that the ADFE provides a considerably better performance for all noise levels. The RBPF–AR(2) on the other hand greatly outperforms all other filters for all noise levels, follows a similar trend with the PCRB, and gets closest to the PCRB as expected. Inspecting the curves for the ADFE $\sigma_\upsilon^2 = 5$ and ADFE $\sigma_\upsilon^2 = 0.05$ reveals that the ADFE is relatively sensitive to both the effect of outliers and the measurement noise for the low–complexity scheme considered in Fig.4. Note that the RBPF–RW and EM–PF are simulated only for low measurement noise since both of these filters exhibit a very poor performance (i.e. RMSE> 0.71) for $\sigma_\upsilon^2 = 5$. The estimation errors for all estimators in Fig.4 are averaged over 100 independent runs of the respective algorithms. The outputs of the RBPF–RW, EM–PF and the ADFE are normalized by the largest absolute values of their respective estimates so as to compensate the effect of arbitrary scaling. For all particle filtering operations, we use the systematic resampling scheme [17, 18], due to its ease of implementation and low complexity [19].

C. Effect of Computational Resources

Based on the results presented in Fig.4, the following question can be raised: How much better could the ADFE and the other estimators perform if a larger amount of computational resources were available? To address this question, we setup another experiment in which the noise levels are fixed at $\sigma_\omega^2 = 5$ and $\sigma_\upsilon^2 = 0.5$. Then we calculate the RMSE for $C \approx \{37, 100, 200, 400\} \times 10^3$ flops. The results are presented in Table III. Clearly, for the ADFE, the performance is remarkably improved with more and more computational power becoming available. All the other estimators however perform almost the same accuracy regardless of how much more computational power is available.

TABLE III
PERFORMANCE OF ALL ESTIMATORS VS THE INCREASING COMPUTATIONAL COMPLEXITY.

Flop Count	RBPF–RW	EM–PF	ADFE	RBPF–AR(2)
	RMSE			
37.10^3	0.82	0.74	0.62	0.029
100.10^3	0.82	0.73	0.54	0.029
200.10^3	0.83	0.73	0.49	0.029
400.10^3	0.83	0.73	0.43	0.029

Before concluding this subsection, our final set of experiments are designed to specify the conditions under which the ADFE could perform almost at the same level of accuracy as the RBPF. Specifically, we test the performance of the ADFE for $P_u/\sigma_\omega^2 \in \{-10, 10\}$ dB, $C \approx \{37, 100, 200, 400\} \times 10^3$ flops, and compare the results with the RBPF–AR(2).

Note that the RMSE curves in Fig.4 were obtained by averaging over 100 runs, since it is feasible to estimate the point variance over a large number of independent trials empirically (e.g. 100) exploiting the LLN. [4] However, for such algorithms that require very large computational complexity, it is more practical to average the results over smaller ensembles (e.g. 10 independent trials) and then refer to some specified tolerance limits for performance assessment. To this end,

[4]LLN is an abbreviation for the law of large numbers.

an interval estimate of the error standard deviation can be obtained in terms of 95% confidence area. This is how we proceed with the ADFE for $C_{ADFE} \approx \{100, 200, 400\} \times 10^3$ flops. Specifically, in Fig.5, we illustrate the point variance estimates for the RBPF–AR(2) with $C_{RBPF} = 37.10^3$ flops. For $C_{ADFE} \approx \{100, 200, 400\} \times 10^3$ flops however, we illustrate the 95% confidence intervals (See Appendix A). Then a practically meaningful way to judge the performance of the ADFE is to address under what conditions the standard deviation of the RBPF lies within the 95% confidence interval for one of the ADFE realizations. Inspecting Fig.5, we observe that only for $P_u/\sigma_\omega^2 \geq 5$ dB and $C_{ADFE} \geq 400.10^3$ flops the standard deviation of the RBPF lies within the tolerance limits provided by the ADFE.

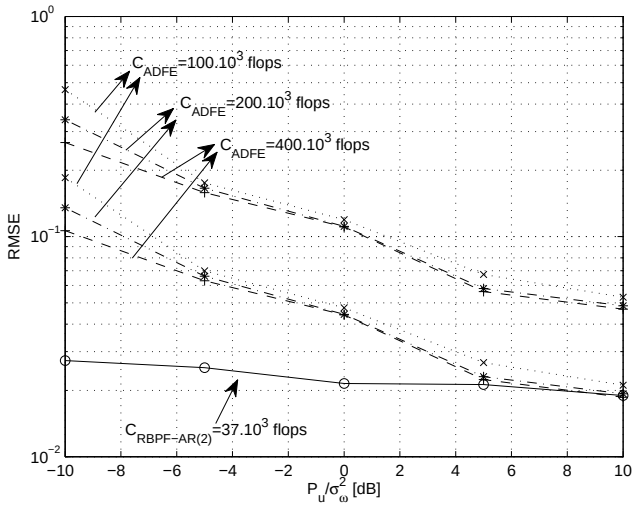

Fig. 5. 95% Confidence intervals for the standard deviation of estimation error for the ADFE with $C_{ADFE} \approx \{100, 200, 400\} \times 10^3$ flops, and the point estimate of the standard deviation of the estimation error for the RBPF with $C_{RBPF} \approx 37.10^3$ flops.

D. Discussion

The results of the experiments studied in this section reveals some interesting observations:

1) The essence of the Bayesian inference is to incorporate all the prior knowledge on the underlying phenomena in a probabilistically optimum fashion to solve the inverse problems. Hence, the Bayesian estimators heavily rely on the accuracy of the prior information that is fused in the estimator. As observed in Fig.3, the driving–force under study is a dynamic force that exhibits rapid variations in time. Therefore, a random walk model is a very poor approximation to describe the evolution of the driving–force. This explains why the performance of the RBPF–RW is greatly inferior to all the other estimators under all noise conditions.

The EM algorithm on the other hand performs better than the random walk model for identifying the driving–force evolution. Specifically, consider the lowest noise levels, ($\sigma_\omega^2 = \sigma_\nu^2 = 0.05$). In this case, the EM algorithm produces after the 5th iteration an estimate of the coefficient vector in (14), $\hat{\theta} = [0.79, -0.49]^T$ averaged over 100 Monte Carlo trials. Further iterations do not lead to any remarkable change in the estimate of θ. The actual coefficient vector however is $\theta = [0.72, -1]$. This observation suggests that a slight discrepancy between the actual dynamics and the model on which the Bayesian estimator operates can lead to a substantial decline in the precision of the estimates. Note that the EM algorithm is very sensitive to the process noise. Therefore, $\sigma_\omega^2 < 0.05$, the results can be further improved, which however is beyond the scope of this work.

When the exact prior information is integrated in the solution of the problem, the results are shown to be greatly improved as illustrated by the RBPF–AR(2) curves in Fig.4.

As a result, we conclude that unless the full prior knowledge about both the governing dynamics of the underlying system and the evolution of the driving–force is available, the proposed ADFE is evidently a favorable approach for the driving–force estimation problem.

2) A shortcoming of the ADFE is its sensitivity to the increasing noise levels in the system. Note that the ADFE was shown to be considerably less sensitive to the process noise when the driving–force is a smooth signal in multiplicative and/or exponent forms in [1]. Following from the results presented in [1] and those obtained in this work, we can suggest that the ADFE provides a close performance to the best available estimator for the additive, non-smooth forces only for low noise levels at the expense of more than 10 times larger computational resources (see Fig.5).

VI. Conclusion

We have provided a deeper analysis of the adaptive driving–force estimator (ADFE) presented in [1, 2], which is designed for estimating the hidden inputs (driving–forces) from nonlinear system with unknown dynamics. The emphasis of the current article is on addressing the conditions under which the ADFE is a good choice for hidden input estimation problem. By virtue of experiments conducted on a general nonlinear system, we have demonstrated that if the underlying system is not fully specified by a mathematical model that describes both the governing system dynamics (state–space equations) and the evolution of the hidden force, then the ADFE is indeed a favorable method. It is also illustrated by a statistical analysis that if very large computational resources are available, then the accuracy of the ADFE gets closer to the best available fully coherent Bayesian estimator for low noise levels. The ADFE owes its good performance to the selection of two robust recursive nonlinear filters that are capable of adapting to the environment: A bank of Echo State Networks for predictive modeling of the observables from the unknown system; and the regularized Least Mean Squares filter for the refinement of the raw driving–force estimates.

Following from the results reported in the current article, the ADFE can be considered to be a good candidate for the diagnosis of abruptly or slowly developing faults in nonlinear dynamic systems for such cases where the system model is not available. In the typical applications of the neural networks to the fault diagnosis problem, system outputs are predicted by

the neural network, and the unexpected deviations in the output residual are used as a fault indicator [20, Ch. 9]. However, our preliminary experiments on the application of the ADFE to this problem show that significantly better performance can be obtained if the input estimation residual is used as the fault indicator rather than the output residual. The rationale behind this observation lies in the fact that the input signal in control applications is typically known and deterministic, whereas the system outputs are mostly corrupted by noise. Then the ADFE can be used for estimating the control input under the normal operating conditions, and the nominal values of the estimation error can be calculated. Hence, the deviation of the input estimation error from the nominal values can be used as a fault indicator. The final results will be declared in a future article once the experiments have been completed.

Acknowledgments

We greatly acknowledge Dr. M. Grasselli of McMaster University for his support and contributions to this work.

Appendix A
Confidence Intervals for Parameter Estimation

To determine the 95% confidence limits, let $\mathcal{E}$ denote the number of independent trials. Let η denote the sample variance obtained by averaging over $\mathcal{E}$ independent trials. Let us also denote the actual variance that we are looking for by σ^2. The random variable $\mathcal{E}\eta/\sigma^2$ is known to have a $\mathcal{X}^2(\mathcal{E})$ density [21]. Then the 95% confidence interval for σ^2 is given by

$$\frac{\mathcal{E}\eta}{\mathcal{X}^2_{0.975}(\mathcal{E})} < \sigma^2 < \frac{\mathcal{E}\eta}{\mathcal{X}^2_{0.025}(\mathcal{E})}. \tag{30}$$

The supporting document reference should contain the authors.

References

[1] U. Güntürkün, "Sequential Reconstruction of Driving-Forces From Nonlinear Nonstationary Dynamics," *Physica D: Nonlinear Phenomena*, vol. 239, no. 13, pp. 1095 – 1107, Jul. 2010, issn: 0167-2789, DOI: 10.1016/j.physd.2010.02.014.

[2] ——, "Toward The Development of Radar Scene Analyzer for Cognitive Radar," *IEEE J. Oceanic Eng., Special Issue on Non-Rayleigh Reverberation and Clutter*, vol. 35, no. 2, pp. 303 – 313, Apr. 2010, issn: 0364-9059, DOI: 10.1109/JOE.2010.2043378.

[3] P. Verdes, P. Granitto, and H. Ceccatto, "Overembedding method for modeling nonstationary systems," *Physical Review Letters*, vol. 118701, pp. 1–4, Mar. 2006.

[4] C. Andrieu and A. Doucet, "Particle filtering for partially observed gaussian state space models," *J. R. Statist. Soc. B*, vol. 64, pp. 827–836, 2000.

[5] G. Hendeby, R. Karlsson, and F. Gustafsson, "A new formulation of the rao-blackwellized particle filter," in *Statistical Signal Processing, 2007. SSP '07. IEEE/SP 14th Workshop on*, 26-29 2007, pp. 84 –88.

[6] T. Schön, F. Gustafsson, and P. Nordlund, "Marginalized particle filters for mixed linear/nonlinear state-space models," *IEEE Trans. Signal Processing*, vol. 53, pp. 2279–2289, Jul. 2005.

[7] (2011) The Supporting Document for – Recursive Hidden Input Estimation in Nonlinear Dynamic Systems with Varying Amounts of *a priori* Knowledge–. [Online]. Available: http://soma.ece.mcmaster.ca/~ulas/eng/studies/pub.htm

[8] T. Minka, "The lightspeed matlab toolbox," 2009, http://research.microsoft.com/en-us/um/people/minka/software/lightspeed/.

[9] A. Papoulis, "Predictable processes and wold's decomposition: A review," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 33, pp. 933–938, Aug. 1985.

[10] J. Liu and M. West, *Combined parameter and state estimation in simulation-based filtering*, ser. Sequential Monte Carlo in Practice, A. Doucet, N. de Freitas, and N. Gordon, Eds. New York: Springer-Verlag, 2001.

[11] C. Andrieu and A. Doucet, "Online expectation-maximization type algorithms for parameter estimation in general state space models," *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '03)*, vol. 6, pp. 69–72, Apr. 2003.

[12] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, pp. 1–38, 1977.

[13] A. Zia, T. Kirubarajan, J. Reilly, D. Yee, K. Punithakumar, and S. Shirani, "An em algorithm for nonlinear state estimation with model uncertainties," *IEEE Trans. Signal Processing*, vol. 56, pp. 921–936, Mar. 2008.

[14] A. Zia, J. Reilly, J. Manton, and S. Shirani, "An information geometric approach to ml estimation with incomplete data," *IEEE Trans. Signal Processing*, vol. 55, pp. 3975–3986, Aug. 2007.

[15] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," *IEE Proc.-Radar, Sonar Navig.*, vol. 140, pp. 107–113, Apr. 1993.

[16] P. Tichavsky, C. Muravchik, and A. Nehorai, "Posterior cramer-rao lower bounds for discrete-time nonlinear filtering," *IEEE Trans. Signal Processing*, vol. 46, pp. 1386–1396, May 1998.

[17] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Proceedings on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[18] G. Kitagawa, "Monte carlo filter and smoother for non-gaussian non-linear state space models," *J. Comput. Graph. Statist.*, vol. 5, no. 1, pp. 1–25, 1996.

[19] J. Hol, T. Schön, and F. Gustafsson, "On resampling algorithms for particle filters," in *Proc. IEEE Nonlinear Statistical Signal Processing Workshop*, Cambridge, UK, Sep. 13–15, 2006.

[20] J. Chen and R. Patton, *Robust Model–Based Fault Diagnosis For Dynamic Systems*. Norwell, Massachusetts, USA: Kluwer Academic Publishers, 1999.

[21] A. Papoulis and S. Pillai, *Probability, Random Variables and Stochastic Processes*, 4th ed. McGraw-Hill, 2002,

ch. 8.