Proceedings of the 45th IEEE Conference on Decision & Control
Manchester Grand Hyatt Hotel
San Diego, CA, USA, December 13-15, 2006

Nonlinear Stochastic Differential-Algebraic Equations with Application to Particle Filtering

Markus Gerdin and Johan Sjöberg
Division of Automatic Control, Department of Electrical Engineering
Linköping University, SE-581 83 Linköping, Sweden
(gerdin,johans)@isy.liu.se

Abstract— **Differential-algebraic equation (DAE) models naturally arise when modeling physical systems from first principles. To be able to use such models for state estimation procedures such as particle filtering, it is desirable to include a noise model. This paper discusses well-posedness of differential-algebraic equations with noise models, here denoted stochastic differential-algebraic equations. Since the exact conditions are rather involved, approximate implementation methods are also discussed. It is also discussed how a particle filter can be implemented for DAE models, and how the approximate implementation methods can be used for particle filtering. Finally, the particle filtering methods are exemplified by implementation of a particle filter for a DAE model.**

I. Introduction

The increased use of modeling in many fields of engineering has led to development of new techniques to make modeling of complex dynamic systems easier. An increased use of model libraries and so-called object-oriented modeling is one result of this trend. The output from these modeling tools is usually not a state-space model but a model containing a mixture of equations, some of which contain variables differentiated with respect to time. These models are usually called descriptor models or DAE (differential-algebraic equation) models. There is a vast recent literature about these models but we have not found a complete treatment of continuous-time noise modeling problems for nonlinear systems. The problem centers around the fact that careless introduction of noise in DAE models might lead to derivatives of the noise appearing or physical variables being equal to a noise process.

The noise modeling problem for linear DAE systems has been treated in [1], and also in [2], [3] where the system is assumed to be *index 1* (see, e.g., [4]). Some classes index 1 nonlinear systems have also been treated in the literature [5], [6]. Here we will examine how noise can be included in nonlinear differential-algebraic equations in the general form

$$F\big(\dot{x}(t), x(t), t\big) = 0 \tag{1}$$

and no assumptions will be made about the index of the DAE. We will also discuss how the system then can be interpreted as a stochastic differential equation, so that stochastic simulation methods and filtering methods such as particle filtering can be used.

This work has been supported by the Swedish Foundation for Strategic Research (SSF) through VISIMOD and EXCEL and by the Swedish Research Council (VR) which are gratefully acknowledged.

First, we will present an example that illustrates the kind of problems that can arise when modeling noise in nonlinear DAE models.

Example 1: Consider the nonlinear DAE

$$\dot{x}_1(t) - x_2^2(t) = 0 \tag{2a}$$

$$x_2(t) - v(t) = 0 \tag{2b}$$

where $v(t)$ is white noise. The second equation states that $x_2(t)$ is equal to a time-continuous white noise process. Since such processes have infinite variance, this is questionable if $x_2(t)$ represents a physical quantity. The first equation states that

$$\dot{x}_1(t) = v^2(t) \tag{3}$$

which also is questionable since nonlinear operations on white noise processes are not well defined.

This paper discusses how noise can be included in DAE models without introducing problems such as those discussed in the example and how particle filters can be implemented for DAE models.

II. Preliminaries

This section presents background material on differential-algebraic equations and stochastic differential equations.

A. Differential-Algebraic Equations

One method to analyze DAE systems is presented in [7], and these results are summarized in this section. For proofs and a complete discussion, the reader is referred to [7]. A slightly modified presentation of these results is also available in [8]. These results are based on rank tests and the implicit function theorem, and are therefore only valid locally. For example, all ranks discussed below may depend on the current value of $x(t)$. The DAE systems treated are of the general form

$$F\big(\dot{x}(t), x(t), t\big) = 0 \tag{4}$$

where $x \in \mathbb{R}^n$, $F \in \mathbb{R}^m$ and $t \in \mathbb{I}$ where $\mathbb{I} \subseteq \mathbb{R}$ is a compact interval.

As with many other methods, this one is based on successive differentiations of the DAE. Therefore define a nonlinear derivative array

$$F_l(t, x, \dot{x}, \ldots, x^{l+1}) = 0 \tag{5}$$

which stacks the original equations and all their derivatives up to level l:

$$F_l(t,x,\dot{x},\dots,x^{l+1}) = \begin{pmatrix} F(\dot{x},x,t) \\ \frac{d}{dt}F(\dot{x},x,t) \\ \vdots \\ \frac{d^l}{dt^l}F(\dot{x},x,t) \end{pmatrix} \tag{6}$$

Partial derivatives of F_l with respect to selected variables p from $(t,x,\dot{x},\dots,x^{l+1})$ are denoted by $F_{l;p}$, e.g.,

$$F_{l;\dot{x},\dots,x^{l+1}} = \begin{pmatrix} \frac{\partial}{\partial \dot{x}}F_l & \cdots & \frac{\partial^{l+1}}{\partial x^{l+1}}F_l \end{pmatrix}. \tag{7}$$

A corresponding notation is used for partial derivatives of other functions.

The solution of the derivative array F_μ for some integer μ is denoted

$$\mathbb{L}_\mu = \{z_\mu \in \mathbb{I}\times\mathbb{R}^n\times\cdots\times\mathbb{R}^n | F_\mu(z_\mu)=0\}. \tag{8}$$

The following hypothesis, Hypothesis 1 by [7], which describes the basic requirements on DAEs handled by the theory can now be formulated.

Hypothesis 1: Consider the general system of nonlinear DAEs (4). There exist integers μ, r, a, d, and v such that the following properties hold:

1) The set $\mathbb{L}_\mu \subseteq \mathbb{R}^{(\mu+2)n+1}$ forms a manifold of dimension $(\mu+2)n+1-r$.
2) We have
$$\operatorname{rank} F_{\mu;x,\dot{x},\dots,x^{(\mu+1)}} = r \tag{9}$$
on $\mathbb{L}_\mu$.
3) We have
$$\operatorname{corank} F_{\mu;x,\dot{x},\dots,x^{(\mu+1)}} - \operatorname{corank} F_{\mu-1;x,\dot{x},\dots,x^{(\mu)}} = v \tag{10}$$
on $\mathbb{L}_\mu$. (Note by current author: The corank is the rank deficiency with respect to rows, i.e, the number of rows minus the rank. For example, if a matrix has 5 rows and rank 3, the corank is 2.) The convention that $\operatorname{corank} F_{-1;x} = 0$ is used.
4) We have
$$\operatorname{rank} F_{\mu;\dot{x},\dots,x^{(\mu+1)}} = r - a \tag{11}$$
on $\mathbb{L}_\mu$ such that there are smooth full rank matrix functions Z_2 and T_2 defined on $\mathbb{L}_\mu$ of size $\big((\mu+1)m,a\big)$ and $(n,n-a)$, respectively, satisfying
$$Z_2^T F_{\mu;\dot{x},\dots,x^{(\mu+1)}} = 0 \tag{12a}$$
$$\operatorname{rank} Z_2^T F_{\mu;x} = a \tag{12b}$$
$$Z_2^T F_{\mu;x} T_2 = 0 \tag{12c}$$
on $\mathbb{L}_\mu$.
5) We have
$$\operatorname{rank} F_{\dot{x}} T_2 = d = m - a - v \tag{13}$$
on $\mathbb{L}_\mu$.

For DAE models that satisfy the hypothesis, it is possible to divide the equations into one part that (locally) forms a state-space system for a part of the variables (denoted x_1), and one part that locally forms static equations for another part of the variables (denoted x_3). If any variables still are undetermined, then they are parameters that can be chosen freely. These variables are denoted x_2. This transformation is performed by first defining the matrix function Z_1 of size (m,d) (which can be chosen constant) such that

$$\operatorname{rank} Z_1^T F_{\dot{x}} T_2 = d \tag{14}$$

and then forming the equations

$$\hat{F}_1 = Z_1^T F = 0 \tag{15a}$$
$$\hat{F}_2 = Z_2^T F_\mu = 0. \tag{15b}$$

The equation (15b) can (locally) be solved for x_3 to give the equations

$$x_3 = \mathcal{R}(t,x_1,x_2) \tag{16}$$

where $\mathcal{R} \in \mathbb{R}^a$. After using (16) to eliminate x_3 and $\dot{x}_3$ in (15a), i.e., in

$$\hat{F}_1(t,x_1,x_2,x_3,\dot{x}_1,\dot{x}_2,\dot{x}_3) = 0,$$

(15a) can be solved for $\dot{x}_1$ to give

$$\dot{x}_1 = \mathcal{L}(t,x_1,x_2,\dot{x}_2) \tag{17}$$

where $\mathcal{L} \in \mathbb{R}^d$. Hence, from Hypothesis 1, it follows that the DAE model (locally) can be casted as

$$\dot{x}_1 = \mathcal{L}(t,x_1,x_2,\dot{x}_2) \tag{18a}$$
$$x_3 = \mathcal{R}(t,x_1,x_2). \tag{18b}$$

Note however that it is typically not possible to solve for $\dot{x}_1$ and x_3 analytically. Instead it is usually necessary to work with $\hat{F}_1$ and $\hat{F}_2$ and solve for $\dot{x}_1$ and x_3 numerically.

The forms discussed here will be the main tools when examining how noise can be added to a nonlinear DAE model. The transformation is typically not unique, for example there may be different possible choices of state variables x_1. Note that the function x_2 is not determined by the equations and can be chosen freely. In many cases the DAE is well determined so that x_2 has size zero.

When using the method discussed in this section, it is usually necessary to successively increase μ until the hypothesis is true. The hypothesis could for example be verified by numeric rank tests at a certain value of $x(t)$, see further Remark 1 in [7]. The practical implementation of methods related to the hypothesis is also discussed in the more recent paper [9].

B. Nonlinear Stochastic Models

The theory of nonlinear stochastic state models is based on the case when white noise enters affinely into the equations,

$$\dot{x}(t) = f\big(x(t),t\big) + \sigma\big(x(t),t\big)w(t). \tag{19}$$

This should be interpreted as a stochastic integral. To point this out, the notion

$$dx = f\big(x(t),t\big)dt + \sigma\big(x(t),t\big)dw \tag{20}$$

is used. Here $w(t)$ is a Wiener process with incremental covariance Idt. It is essential for the existence of a well defined solution that the noise process w enters affinely into the equations. See e.g., [10].

Stochastic state models can be used for example for simulation and for state estimation using extended Kalman filters and particle filters.

III. Main Results

The main theoretical result of the paper describes how white noise can be added to a DAE system

$$F\big(\dot{x}(t), x(t), t\big) = 0 \tag{21}$$

in a way that makes it is possible to interpret it as a stochastic differential equation. It is assumed that the DAE fulfills Hypothesis 1. In general, a DAE with noise can be written as

$$F\big(\dot{x}(t), x(t), t, v(t)\big) = 0. \tag{22}$$

where $v(t)$ ($\dim v = n_v$) is a white noise process. The main result describes how noise can be added. We will then discuss how this result can be used in practice. We will use the notation (7) for partial derivatives and $\hat{F}_{2,x_3}^{-1}$ for the inverse of the matrix $\hat{F}_{2,x_3}$.

Theorem 1: Assume that (21) fulfills Hypothesis 1 so that $\hat{F}_1$ and $\hat{F}_2$ in (15) can be calculated and that x_2 is a known signal. If white noise is added to the equations as in (22), there exists a well defined solution x in terms of stochastic differential equations provided that when repeating the transformation to $\hat{F}_1$ and $\hat{F}_2$ with $v(t)$ included, this gives the system

$$\begin{aligned}\hat{F}_1\big(t, x_1, x_2, x_3, \dot{x}_1 - \sigma(x_1, x_2, x_3)v, \dot{x}_2,&\\ \dot{x}_3 + \hat{F}_{2,x_3}^{-1}\hat{F}_{2,x_1}\sigma(x_1, x_2, x_3)v\big) &= 0 \qquad (23a)\\ \hat{F}_2\big(t, x_1, x_2, x_3\big) &= 0 \qquad (23b)\end{aligned}$$

for some matrix function $\sigma(x_1, x_2, x_3)$ of dimension (d, n_v).

Proof: Since x_2 is a known signal, we include it in the general time dependency t. Differentiating (23b) with respect to time yields

$$\hat{F}_{2;t} + \hat{F}_{2;x_1}\dot{x}_1 + \hat{F}_{2;x_3}\dot{x}_3 = 0. \tag{24}$$

Since $\hat{F}_2$ is locally solvable for x_3, $F_{2;x_3}$ is invertible. This means that $\dot{x}_3$ can be written as

$$\dot{x}_3 = -\hat{F}_{2;x_3}^{-1}(\hat{F}_{2;t} + \hat{F}_{2;x_1}\dot{x}_1). \tag{25}$$

(23b) can also be locally solved for x_3 to give

$$x_3 = \mathcal{R}(t, x_1) \tag{26}$$

for some function $\mathcal{R}$. Inserting this into (23a) gives

$$\begin{aligned}&\hat{F}_1\big(t, x_1, \mathcal{R}, \dot{x}_1 - \sigma(x_1, \mathcal{R})v,\\ &\quad - \hat{F}_{2;x_3}^{-1}(\hat{F}_{2;t} + \hat{F}_{2;x_1}\dot{x}_1) + \hat{F}_{2;x_3}^{-1}\hat{F}_{2;x_1}\sigma(x_1, \mathcal{R})v\big).\end{aligned} \tag{27}$$

The equation $\hat{F}_1 = 0$ now takes the form

$$\begin{aligned}&\hat{F}_1\big(t, x_1, \mathcal{R}, \dot{x}_1 - \sigma(x_1, \mathcal{R})v,\\ &\quad - \hat{F}_{2;x_3}^{-1}\hat{F}_{2;t} - \hat{F}_{2;x_3}^{-1}\hat{F}_{2;x_1}(\dot{x}_1 - \sigma(x_1, \mathcal{R})v)\big) = 0.\end{aligned} \tag{28}$$

Since Hypothesis 1 is fulfilled, this equation can be solved for $\dot{x}_1$. Since $-\sigma(x_1, \mathcal{R})v$ enters the equations in the same way as $\dot{x}_1$, the solution takes the form

$$\dot{x}_1 - \sigma(x_1, \mathcal{R})v = \mathcal{L}(t, x_1) \tag{29}$$

for some function $\mathcal{L}$. This can be interpreted as the stochastic differential equation

$$dx_1 = \mathcal{L}(t, x_1)dt + \sigma(x_1, \mathcal{R})dv \tag{30}$$

so x_1 has a well defined solution. A solution for x_3 is then defined through (26).

■

If noise has been added to a DAE model, the theorem above gives conditions for the system to be well-posed using a transformed version of the system. It may also be interesting to be able to see if the SDAE is well-posed already in the original equations. As discussed in the theorem above, the SDAE is well-posed if the equations $\hat{F}_1 = 0$ and $\hat{F}_2 = 0$ take the form (23). In the original equations, this can typically be seen as adding noise according to

$$F\left(\begin{pmatrix}\dot{x}_1 - \sigma(x_1, x_2, x_3)v\\ \dot{x}_2\\ \dot{x}_3 + \hat{F}_{2;x_3}^{-1}\hat{F}_{2;x_1}\sigma(x_1, x_2, x_3)v\end{pmatrix}, \begin{pmatrix}x_1\\ x_2\\ x_3\end{pmatrix}, t\right) = 0. \tag{31}$$

One common situation, when it is easy to see how the noise can be added is the index 1 semi-explicit DAE

$$\begin{aligned}\dot{x}_a &= f(x_a, x_b) \qquad (32a)\\ 0 &= g(x_a, x_b). \qquad (32b)\end{aligned}$$

Locally, x_b can be solved from (32b), so we have the form (18). Noise can thus be added according to

$$\begin{aligned}\dot{x}_a &= f(x_a, x_b) + \sigma(x_a, x_b)v \qquad (33a)\\ 0 &= g(x_a, x_b). \qquad (33b)\end{aligned}$$

A. Filtered Noise

In practical situations it may not be a good model of the system at hand to include a white noise signal $v(t)$ into the equations. Instead it may be more reasonable to add for example noise with low frequency content. Noise with low frequency content $\bar{x}(t)$ can be realized as white noise $v(t)$ filtered through a linear filter,

$$\dot{\bar{x}}(t) = A\bar{x}(t) + Bv(t). \tag{34}$$

Including $\bar{x}(t)$ in the nonlinear DAE model would then give a contribution of low frequency noise,

$$\begin{aligned}F\big(\dot{x}(t), x(t), t, \bar{x}(t)\big) &= 0 \qquad (35)\\ \dot{\bar{x}}(t) &= A\bar{x}(t) + Bv(t). \qquad (36)\end{aligned}$$

To examine if this problem is well-posed, we can apply Theorem 1.

IV. Particle Filtering

An important aspect of noise modeling in DAE equations is to implement state estimation, and a method to estimate states in nonlinear systems is the particle filter [11], [12], [13]. In this section we will propose a method to implement the particle filter for DAE systems. We will use the following particle filtering algorithm from [14].

1) Initialize the M particles, $\{x_{0|-1}^{(i)}\}_{i=1}^{M} \sim p_{x_0}(x_0)$ and set $t := 0$.
2) Measurement update: calculate importance weights $\{q_t^{(i)}\}_{i=1}^{M}$ according to
$$q_t^{(i)} = p\left(y_t | x_{t|t-1}^{(i)}\right), \quad i = 1, \dots, M, \tag{37}$$
and normalize $\tilde{q}_t^{(i)} = q_t^{(i)} / \sum_{j=1}^{M} q_t^{(j)}$.
3) Resampling: draw M particles, with replacement, according to
$$\Pr\left(x_{t|t}^{(i)} = x_{t|t-1}^{(j)}\right) = \tilde{q}_t^{(j)} \quad i = 1, ..., M. \tag{38}$$
4) Time update: predict new particles according to
$$x_{t+1|t}^{(i)} \sim p\left(x_{t+1|t} | x_{t|t}^{(i)}\right), \quad i = 1, \dots, M. \tag{39}$$
5) Set $t := t + 1$ and iterate from step 2.

This algorithm is typically implemented for state-space systems, and the x variables are estimates of the states. To examine how the implementation for DAE systems should be done, we consider the reduced form
$$\hat{F}_1(t, x_1, x_2, x_3, \dot{x}_1 - \sigma(x_1, x_2, x_3)v, \dot{x}_2, \dot{x}_3) = 0 \tag{40a}$$
$$\hat{F}_2(t, x_1, x_2, x_3) = 0 \tag{40b}$$
to which a well-posed stochastic nonlinear DAE can be transformed. A function of the internal variables is measured at time instances t_k,
$$y(t_k) = h\big(x_1(t_k), x_2(t_k), x_3(t_k)\big) + e(t_k) \tag{41}$$
where $e(t_k)$ is a sequence of stochastic variables. The x_2 vector is not restricted by the equations, so we assume that this is a known function.

Equation (40) can locally be solved for $\dot{x}_1$ and x_3. It is typically not possible to solve for these variables explicitly, but implicitly the equations together with the known function x_2 define the system
$$\dot{x}_1 = \mathcal{L}(t, x_1) + \sigma(x_1, t)v \tag{42a}$$
$$y(t_k) = h(x_1, t) + e(t_k). \tag{42b}$$
Since it is not possible to calculate this system explicitly, we will discuss numerical implementation methods in the following section.

The state-space system (42) can be used to implement a particle filter for estimation of x_1. The state equation should be discretized. (A more thorough discussion of this falls out of the scope of this paper.) This can be done using for example a numerical solver. The state update in step 4 in the particle filtering algorithm is thus performed by solving (42a) for one time step. The measurement update in step 2 of the particle filtering algorithm is performed using the measurement equation (42b). After estimating x_1, estimates of x_3 can be computed using (40b).

V. Implementation Issues

The exact transformation in the result by [7] may be difficult to calculate, and therefore approximate implementations may be considered. One way to do this is to use the type of DAE solver that is included in modeling environments for object-oriented modeling such as Dymola [15].

Purely numeric solvers for DAE only handle a limited class of DAE, usually index 1 systems, or limited classes of higher index systems. One common numerical solver is DASSL [4]. The index of a DAE is basically the number of times the equations must be differentiated to solve $\dot{x}$ as a function of x. For a more thorough discussion on this, see e.g., [4].

For object-oriented models, it is not sufficient to treat lower index problems. Solvers for these models typically reduce the index to 1 by differentiating equations that are chosen using Pantelides's algorithm [16] and structure the equations so that large DAE systems can be simulated efficiently. This means that a transformation into approximations of $\hat{F}_1$ and $\hat{F}_2$ is made. From these equations, $\dot{x}_1$ and x_3 can be solved numerically. After the transformation, a numerical DAE solver is used. A detailed discussion on the construction of such solvers is not presented here, but a discussion about them can be found in, e.g., [17]. During the index reduction process, some of the variables $x(t)$ are selected as states. For the user, this means that initial values of these variables can be selected independently of each other, while the initial values of the remaining variables are computed from the initial values of the states. It is possible for the user to influence the state selection process by indicating that some variables are preferred as states. During the integration process to solve the DAE, one method is to solve for the derivatives of the states and then use a solver for ordinary differential equations (ODE). Another is to use an index 1 solver on the reduced system.

As discussed in Section III, a user should select how the states are affected by noise when modeling a stochastic DAE. The information about which variables that are states can be delivered by a DAE solver. This can then be implemented in two ways. Either a variable representing the noise is added to the original equations according to the term σv in (31). If this signal appears in the approximate $\hat{F}_2$ or in incorrect positions in the approximate $\hat{F}_1$, it is removed from these locations and assumed that noise is added to the original equations so that this is achieved. Another way is to add the noise to the approximate $\hat{F}_1$ computed by the solver. The equations can then be used for simulation or state estimation.

The solvers can also be used for approximate implementation of particle filters for DAE systems. The idea behind this is that the transformation to the form
$$\dot{x}_1 = \mathcal{L}(t, x_1, x_2, \dot{x}_2) + \sigma(x_1, x_2)v \tag{43a}$$
$$x_3 = \mathcal{R}(t, x_1, x_2) \tag{43b}$$
can be made by solving $\hat{F}_1$ and $\hat{F}_2$ numerically at each time step. This means that given values of x_1, x_2, and v the solver can give $\dot{x}_1$ and x_3. This means that the state equation (43a) can be used to estimate x_1, and x_3 can then be computed from (43b).

VI. Example

In this section we examine a DAE model of a pendulum. First noise is added, and then a particle filter is implemented

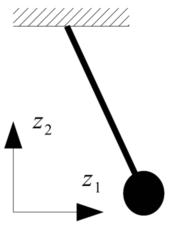

Fig. 1. A pendulum

to estimate the state of the pendulum. The equations describing the pendulum are

$$\dot{z}_1 = z_3 \tag{44a}$$
$$\dot{z}_2 = z_4 \tag{44b}$$
$$\dot{z}_3 = -\lambda \cdot z_1 - b \cdot z_3^2 \tag{44c}$$
$$\dot{z}_4 = -\lambda \cdot z_2 - b \cdot z_4^2 - g \tag{44d}$$
$$0 = z_1^2 + z_2^2 - L^2. \tag{44e}$$

This is a modified example from [4]. As shown in Figure 1, z_1 and z_2 are the horizontal and vertical position of the pendulum. Furthermore, z_3 and z_4 are the respective velocities, λ is the tension in the pendulum, b represents resistance caused by the air, g is the gravity, and L is the length of the pendulum.

We will use the approximate methods discussed in Section V, so the equations are entered into the DAE solver in Dymola. There are several possible ways to select states for these equations, but here z_1 and z_3 are selected. The solver could also be instructed to make other selections. This means that noise could be added to equation (44a) and (44c). But since (44a) is a relation that should hold exactly, noise is only added to (44c). This results in

$$\dot{z}_1 = z_3 \tag{45a}$$
$$\dot{z}_3 = -\lambda \cdot z_1 - b \cdot z_3^2 + w \tag{45b}$$

where w is white noise. Also the remaining equations,

$$\dot{z}_2 = z_4 \tag{46a}$$
$$\dot{z}_4 = -\lambda \cdot z_2 - b \cdot z_4^2 - g \tag{46b}$$
$$0 = z_1^2 + z_2^2 - L^2, \tag{46c}$$

may have to include the noise signal to make the estimation problem well-posed. As discussed earlier, a user does not need to consider the exact form of this. Instead the equations in the form above are compiled using Dymola. In the generated equations, which are available as C code, any occurrence of w in $\hat{F}_2$ are deleted and the model is compiled. In this way, the exact way the noise enters (46) does not have to be considered. (For illustration we will discuss this in Section VI-A, but this is not necessary for implementation of the particle filter.)

To generate data for a state estimation experiment, the model is inserted into the Simulink environment using the Dymola-Simulink interface available with Dymola. The purpose of this experiment is not to demonstrate the performance of a filtering algorithm, but rather to show how DAE models can be used in a direct way when constructing particle filters. Therefore it is sufficient to use simulated data for the experiment. The constants were chosen as $L = 1$, $b = 0.05$ and $g = 9.81$. Process noise was generated with the *Band-Limited White Noise* block in Simulink with noise power 0.01. The initial values of the states were $z_1 = 0.5$ and $z_3 = -0.1$. The measured variable is the tension in the pendulum λ,

$$y(t_k) = \lambda(t_k) + e(t_k). \tag{47}$$

Measurements with noise variance 0.1 was collected at the sampling interval 0.05 s.

The particle filter was implemented in Matlab with the time updates being performed by simulating the model using the Dymola-Simulink interface. The initial particles were spread between $z_1 = 0.1$ and $z_1 = 0.6$ and between $z_3 = -0.2$ and $z_3 = 0.2$. Only positive values of z_1 were used since the symmetry in the system makes it impossible to distinguish between positive and negative z_1 using only measurements of λ. The particle filter was tuned to use noise power 0.1 for the process noise and variance 0.2 for the measurement noise to simulate the situation were the noise characteristics are not exactly known. A typical result of an estimation is shown in Figure 2.

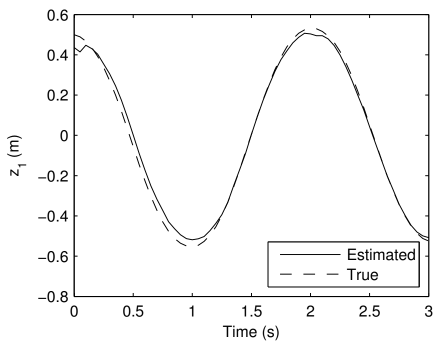

Fig. 2. Typical result of particle filtering.

To examine the reliability of the filtering algorithm, 100 Monte Carlo runs were made. Then the RMSE value was calculated according to

$$\mathrm{RMSE}(t) = \sqrt{\frac{1}{M}\sum_{j=1}^{M}\big(x(t) - \hat{x}_j(t)\big)^2} \tag{48}$$

where M is the number of runs, here $M = 100$, $x(t)$ is the true state value and $\hat{x}_j(t)$ is the estimated state value in run j. The result is shown in Figure 3. The estimation error in the velocity z_3 is larger when the pendulum changes direction, which could mean that it is more difficult to estimate the velocity there. It can be noted that the Monte Carlo runs took a rather long time to perform because of the overhead involved in using the Simulink-Dymola connection for each time update in the particle filter. It would be more efficient for example to use a pure C code implementation.

A. Noise modeling details

To show that the noise term in 45b formally also requires noise to be added to some of the equations (46) we will

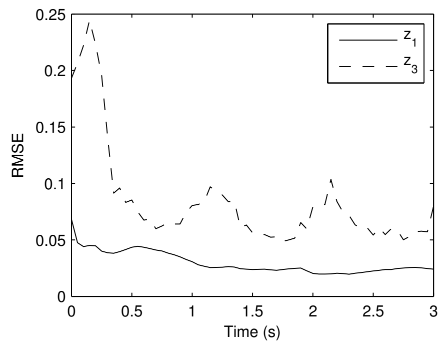

Fig. 3. RMSE for 100 Monte Carlo runs.

perform some manipulations of the equations. As discussed earlier, it is not necessary to make these computations in practice, but we include them here to illustrate that when removing the noise signal from $\hat{F}_2$ (or its approximation), it is possible to interpret this as adding a certain noise signal in the original equations.

Consider (46c)

$$0 = z_1^2 + z_2^2 - L^2. \tag{49}$$

Differentiating this equation with respect to time gives

$$0 = 2z_1\dot{z}_1 + 2z_2\dot{z}_2. \tag{50}$$

Inserting (45a) and (46a) gives

$$0 = 2z_1z_3 + 2z_2z_4 \tag{51}$$

which after differentiation gives

$$0 = 2\dot{z}_1z_3 + 2z_1\dot{z}_3 + 2\dot{z}_2z_4 + 2z_2\dot{z}_4. \tag{52}$$

Inserting the expressions for the derivatives gives

$$0 = 2z_3^2+2z_1(-\lambda\cdot z_1-bz_3^2+w)+2z_4^2+2z_2(-\lambda\cdot z_2-bz_4^2-g). \tag{53}$$

Solving for λ in this expression gives

$$\lambda = \frac{1}{L}\big(z_3^2 + z_4^2 - b(z_1z_3^2 + z_2z_4^2) + z_1w - z_2g\big) \tag{54}$$

where we have used $z_1^2 + z_2^2 = L$. This means that the tension λ is directly depending on the noise w, which is unacceptable. This dependence is eliminated if noise is added to (46b) according to

$$\dot{z}_4 = -\lambda \cdot z_2 - b \cdot z_4^2 - g - \frac{z_1}{z_2}w \tag{55}$$

since w then would be eliminated from (53).

VII. Conclusions

We have presented a theoretical base for introduction of noise processes in DAE models. The exact conditions that this gives can be hard to use in practice, for example since it requires rank tests. Therefore an approximate solution was proposed. This solution uses the kind of DAE solvers that are included in modeling environments for object-oriented modeling. These solvers produce an approximation of the transformation that is necessary to include noise in an appropriate way.

It was also discussed how particle filtering can be implemented for DAE models. This can be useful since particle filters using these methods could be implemented automatically from modeling environments, only requiring the user to select a noise model.

An example which shows that it is possible to implement a particle filter using a DAE model was presented. The results were similar to what could be expected from an implementation using a regular state-space model.

Further research issues include to examine if it is possible to implement other estimation methods such as extended Kalman filtering for DAE models. It is also necessary to make the implementation of the particle filter more efficient by implementing it for example in C code.

References

[1] M. Gerdin, T. Glad, and L. Ljung, “Well-posedness of filtering problems for stochastic linear DAE models,” in *Proceedings of 44th IEEE Conference on Decision and Control and European Control Conference ECC 2005*, Seville, Spain, Dec. 2005, pp. 350–355.

[2] O. Schein and G. Denk, “Numerical solution of stochastic differential-algebraic equations with applications to transient noise simulation of microelectronic circuits,” *Journal of Computational and Applied Mathematics*, vol. 100, no. 1, pp. 77–92, Nov. 1998.

[3] M. Darouach, M. Boutayeb, and M. Zasadzinski, “Kalman filtering for continuous descriptor systems,” in *Proceedings of the American Control Conference*. Albuquerque, New Mexico: AACC, June 1997, pp. 2108–2112.

[4] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, ser. Classics In Applied Mathematics. Philadelphia: SIAM, 1996.

[5] V. M. Becerra, P. D. Roberts, and G. W. Griffiths, “Applying the extended Kalman filter to systems described by nonlinear differential-algebraic equations,” *Control Engineering Practice*, vol. 9, pp. 267–281, 2001.

[6] R. Winkler, “Stochastic differential algebraic equations of index 1 and applications in circuit simulation,” *Journal of Computational and Applied Mathematics*, vol. 163, no. 2, pp. 435–463, Feb. 2004.

[7] P. Kunkel and V. Mehrmann, “Analysis of over- and underdetermined nonlinear differential-algebraic systems with application to nonlinear control problems,” *Mathematics of Control, Signals, and Systems*, vol. 14, no. 3, pp. 233–256, 2001.

[8] ——, *Differential-Algebraic Equations: Analysis and Numerical Solution*. Zürich: European Mathematical Society, 2006.

[9] ——, “Index reduction for differential-algebraic equations by minimal extension,” *Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 84, no. 9, pp. 579–597, July 2004.

[10] K. J. Åström, *Introduction to Stochastic Control Theory*, ser. Mathematics in Science and Engineering. New York and London: Academic Press, 1970.

[11] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, “Novel approach to nonlinear/non-Gaussian Bayesian state estimation,” in *Radar and Signal Processing, IEE Proceedings F*, vol. 140, Apr. 1993, pp. 107–113.

[12] A. Doucet, N. de Freitas, and N. Gordon, Eds., *Sequential Monte Carlo methods in Practice*. New York: Springer-Verlag, 2001.

[13] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter: particle filters for tracking applications*. Boston, Mass., London: Artech House, 2004.

[14] T. B. Schön, “Estimation of nonlinear systems: Theory and applications,” Ph.D. dissertation, Linköping University, Feb. 2006.

[15] S. E. Mattsson, H. Elmqvist, and M. Otter, “Physical system modeling with Modelica,” *Control Engineering Practice*, vol. 6, pp. 501–510, 1998.

[16] C. C. Pantelides, “The consistent initialization of differential-algebraic systems,” *SIAM Journal on Scientific and Statistical Computing*, vol. 9, no. 2, pp. 213–231, Mar. 1988.

[17] P. Fritzson, *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. New York: Wiley-IEEE, 2004.