Sampling Jitter Mitigation in Latency-Critical State-Estimation Applications using Particle Filters

Conference Paper · March 2017

2 authors:

Viktorio El Hakim
University of Twente
1 PUBLICATION 0 CITATIONS

Marco Bekooij
University of Twente
55 PUBLICATIONS 584 CITATIONS

Some of the authors of this publication are also working on these related projects:

ASSUME View project

All content following this page was uploaded by Viktorio El Hakim on 24 April 2017.
The user has requested enhancement of the downloaded file.
Sampling Jitter Mitigation in Latency-Critical State-Estimation Applications using Particle Filters

Viktorio S. El Hakim†† Marco J. G. Bekooij†, 2
(E-mail: v.s.elhakim@utwente.nl) (E-mail: marco.bekooij@nxp.nl)

1Department of Computer Architectures for Embedded Systems, University of Twente, Enschede, The Netherlands
2Department of Embedded Software and Signal Processing, NXP Semiconductors, Eindhoven, The Netherlands

Abstract: State estimation algorithms, such as the Kalman filter, are applied for conditioning and sensor fusion in digital control loops. It is desirable that these algorithms can be executed on embedded multiprocessor systems. However this results in large worst-case execution times with a consequence that a large sampling period must selected, which degrades the estimation and control performance.

In this paper, we propose a free-running state-estimation approach in which the next sample is taken as soon as the current iteration completes. The approach utilizes the particle filter algorithm to mitigate the effects of sampling jitter, introduced by the variation in the execution times of tasks. As a result of the reduced interval between subsequent sampling moments the estimation accuracy is improved. The delay introduced by the estimator in a control loop is reduced by enabling execution of the prediction step in parallel with other control tasks.

We compare simulation results obtained for our approach with a Kalman filter based approach, by estimating the state of a DC motor. These results show that our approach minimizes the estimation error, as a result of sampling jitter, by up to a factor of 10. Additionally we show that the approach does not require precise knowledge of the distribution of the execution times of the tasks.

Keywords: Sampling jitter, Particle filtering

1. INTRODUCTION

State estimation algorithms, such as the Kalman Filter (KF) and the Particle Filter (PF), are often used as part of multi-sensory digital control systems to extract and fuse relevant information about the plant from raw and noisy measurement data. As such the accuracy of the estimate is of critical importance. However the variable execution time per iteration introduces sampling jitter, which can deteriorate the accuracy and the performance of the controller. By execution time per iteration we refer to the combined computational times of the controller, and the prediction and update steps of a typical estimation algorithm shown in Fig. 1. In this figure, $z_k$ and $u_k$ are the measurement and actuation input vectors respectively, $\hat{x}_k$ is an output estimate vector, $D$ is a one sample delay block and $x_{k+1|k}$ is a state prediction vector.

Specifically, performance deterioration results from incorrectly selecting the sampling period of the sensors, which is assumed equal to the time step in the prediction model of the estimator. On one hand, selecting a period smaller than the Worst-Case Execution Time (WCET) of an iteration increases the probability of mismatch error between the predicted state and the sampled state due to jitter. On the other hand, selecting a very large sampling period increases the latency in control loops, amplifies the aliasing effects and can potentially destabilize the controller. Selection of a sampling period larger than the WCET and enforcing periodic execution avoids sampling jitter completely.

However the WCET estimate is usually much larger than the average execution time in case the estimation algorithm is executed on an a processor with caches [1]. Execution on an embedded Symmetric Multiprocessor System (SMP) with a multi-layer cache hierarchy and an SDRAM, on which preemptive task scheduling is applied and also other applications are executed, increases the WCET estimate even more [2]. In reality the estimated WCET is usually much larger than the average execution time and also larger than the actual upper bound on the execution time.

In this paper we propose a state-estimation approach based on PFs, which increases the robustness against sampling jitter. This is done by utilizing multiple state predictions based on Monte Carlo (MC) simulation using an iteration time Probability Density Function (PDF). The resulting robustness to jitter allows aperiodic execution of our estimation algorithm where sampling of the state of the plant, and the next iteration begin immediately as soon as the current iteration cycle ends. The main benefit of this approach is that determining a suitable sampling period becomes unnecessary, since the algorithm adapts the time between subsequent samplings automatically. Additionally the approach allows the prediction task to execute in parallel with other control tasks. This minimizes the delay that the estimator introduces in a control loop and improves control performance. We show that the performance of our estimator is significantly better than that of a periodically sampling estimator based on a WCET estimate.

We demonstrate the benefits of our approach for a continuous-time Linear Time Invariant (LTI) plant model, by comparing our approach with the optimal KF estimator. Specifically we show that the estimation error for our approach, obtained by simulation, is smaller than that of the KF estimator. Furthermore, we show that our approach does not require accurate knowledge about the PDF of the execution times.

The paper is organized as follows. In Section 2 we describe the basic idea of our approach. In Section 3 we describe related work. In Section 4 we compare the iteration times of our approach and an alternative one. In Section 5 we introduce the reader to preliminary state-estimation theory. In Section 6 we present our approach.

---

† Viktorio S. El Hakim is the presenter of this paper.
In Section 7 we present the results of our case study. Finally we conclude the paper in Section 8.

2. BASIC IDEA

In this section we describe the basic idea behind our approach using schedules of the tasks in a control loop with state estimation.

2.1. Overview

State estimation algorithms such as the KF, its derivations, and the PF are based on the Bayesian framework. The “prediction” step in the Bayesian framework performs a time update of the state PDF prior to receiving a measurement. This step utilizes a discretized model of the plant that describes its dynamic behavior as a function of the previous state, a time step and the process noise PDF. The “update” step uses a newly sampled measurement to improve the predicted state estimate using Bayes’ rule.

When mapped as tasks on a multiprocessor system the execution schedules of these steps in a control loop have different implications on the functional behavior and the sampling times of the control loop. Consider the block diagram of a control loop in Fig. 2 where the blue blocks represent discrete time processes executed as individual tasks on a computer, while red blocks – external continuous time processes. The processes are separated using hybrid sample and hold blocks marked in green. Three relevant task schedules are considered as seen in Fig. 3. Here each task execution is represented as a block along a timeline. The measurement, control, prediction and update tasks are abbreviated as “Meas.” “Ctrl.” “Pr.” and “Upd.” respectively. The arrows pointing downwards mark the sampling moment of the system during iteration $k$. The arrows pointing upwards mark the production moments of the estimated state at $t_k$. The arrows pointing to the right mark the actuation moments $u_k$.

In Fig. 3a a schedule is shown when periodic sampling is used, and each task is executed strictly within an iteration time upper bound $L$. This restriction is enforced on the execution time of the tasks to ensure that the sampling period stays constant. However ensuring that all tasks finish their execution within the specified iteration time interval requires, that the sampling period $T = L$ is selected as large as possible to avoid sampling jitter. As discussed earlier finding $L$ is difficult and a WCET estimate is used instead which typically results in a very large sampling period. This is undesirable since the latency is increased and the control loop may destabilize. Note that the prediction and control tasks may execute in parallel to improve the latency of the control loop. However this does not necessarily result in decreased WCET.

In Fig. 3b one can see the schedule of the same tasks when the sampling times in the control loop are not fixed, and instead are completely determined by the duration $L_k$ of iteration $k$. A new iteration also starts immediately after the termination of the previous one. As such sampling jitter is immediately introduced in the system. However since the prediction step is executed after receiving a measurement in iteration $k$, it knows the duration of the previous iteration $L_{k-1}$ and uses it as time step to compute $x_{k|k-1}$, thereby eliminating the effects of sampling jitter. A drawback of this alternative approach is that each task is restricted to sequential execution, thus increasing the iteration time $L_k$. We formally prove this later, and show that our approach minimizes the iteration times while still providing robustness to jitter.

In Fig. 3c a schedule is shown that is based on our approach. Like the schedule in Fig. 3b each sampling and new iteration start immediately after the previous iteration ends. However our approach allows execution of the prediction task in parallel with the control and measurement tasks, and thus minimizes the duration $L_k$ of iteration $k$. As a result, the control loop is sampling much faster on average than a periodically sampling system based on Fig. 3a or an aperiodically sampling system based on Fig. 3b. However standard estimation algorithms such as the KF do not work here, since the $k$-th execution of the prediction step lacks information about the next sampling moment $t_{k+1}$. Thus the predicted state may greatly differ from the sampled state due to time step mismatch.

We address this issue in our approach by utilizing the PF algorithm to perform multiple state predictions corresponding to different time steps drawn from an iteration time probability distribution. We thus assume that the termination of each iteration is a stochastic event with a known PDF $\pi(L_k)$ from which samples can be drawn. There are various methods to estimate this PDF [3, 4] as accurate as possible, however we show in our case study that the influence of the shape of the PDF on the estimation accuracy is small. Finally the update step of the PF estimates the state from the predictions when the measurement arrives.

However the same issue is faced by the controller, if it also utilizes a fixed time step, and therefore requires the plant to be measured periodically. This issue has been actively addressed by the control systems community, by introducing predictor based controllers [5–7]. Analysis of the stability of free-running controllers has been considered in [8]. The increase in robustness against sampling jitter by making use of a PF comes at the cost of a higher computational load. Fortunately, a PF is easy to parallelize, such that computation of particles can be distributed among multiple processors [9]. Therefore, despite the additional computation load, the delay introduced by the PF in a control loop is not necessarily larger than alternative estimation approaches.

3. RELATED WORK

In this section we relate our state-estimation approach to existing state-estimation approaches that address the sampling jitter problem.

The problem of sampling time jitter, caused by randomly delayed and/or irregularly sampled observations of a dynamical system from multiple sensors, is considered in [10, 11]. Here, the authors consider the uncertainty in the sampling moments and the measurement delay. They provide an analytical solution based on KFs, given that the arrival times of the observations are known as is the case in the schedule in Figure 3b. However, they do not analyze the consequences of observation jitter as a result of varying execution times of the KF and do not compare the results with a PF based approach. Furthermore, they do not propose techniques that minimize the delay introduced by the estimator by exploiting parallel execution of tasks. Parallel execution of tasks is explicit in the schedule in Figure 3c.

A similar but more recent work that addresses sam-
In this section we use a formal temporal model to derive algebraic expressions for the iteration time. With these expressions the iteration times of the schedule in Fig. 3b can be compared with the iteration time of the schedule in Fig. 3c for every possible execution time of the tasks.

As discussed earlier when the tasks are executed according to the schedule in Fig. 3b, the estimation algorithm knows the sampling moments and can sample periodically. However, since parallel execution of tasks is restricted the iteration period $L_k$ is increased. Our estimator on the other hand lifts this restriction and reduces the iteration time $L_k$. A disadvantage of the estimator is that it is potentially less accurate than the alternative one. However we show later in our case study that our estimator can outperform the alternative, because its iteration times are smaller on average.

To show that, we compare $L_k$ with $L_k$ using an Homogeneous Synchronous Data-Flow (HSDF) [15, 16] temporal analysis model. An HSDF graph consists of nodes called “actors” interconnected by edges which represent FIFO buffers with an unbounded capacity. An actor $a_i$ is a self-timed entity which fires the $k$-th time as soon as a data token becomes available on all of its input edges at start-time $s_k(a_i)$. When a firing finishes after its firing duration $\rho_k(a_i) \geq 0$, the actor produces tokens on its output edges at an end-time $f_k(a_i)$. This can be formalized as:

$$f_k(a_i) = s_k(a_i) + \rho_k(a_i)$$  \hfill (1)

In an HSDF graph an actor consumes and produces exactly one token per edge on each firing. Initial tokens are also available on edges, and are indicated as black dots.

Since an actor starts firing immediately once there is at least one token produced on all of its input edges, we have for the starting times:

$$s_k(a_i) = \max \{ f_k - \delta_j (a_j) \} , \quad a_j \in I_i ,$$  \hfill (2)

where $\delta_j$ is the initial token amount on the input edge from actor $a_j$ to $a_i$, and $I_i$ is the set of all actors that produce tokens for $a_i$.

Before going further with our analysis we first establish the correspondence between the block diagram in Fig. 2 and the corresponding HSDF models in Figs. 4b and 4a. The models are based on the schedules in Figs. 3b and 3c respectively. Only the blue blocks are represented as actors in the HSDF models because continuous time blocks do not propagate or delay discrete events. The actors with subscripts $P$, $U$, $M$ and $C$ abstract the prediction, update, measurement and controller tasks respectively. Note that $b_i$ represents an actor from the HSDF in Fig. 4b. The sample delays $D$ correspond to initial tokens in the HSDF graphs. Note that the summing junction between the reference signal and the estimator’s output is assumed to be part of the controller task and thus omitted from the HSDF model. The red dashed edge, indicates that it carries a discrete event trigger signal, and it corresponds to a normal edge in the HSDF model.

Given the HSDF model in Fig. 4a, we can derive an expression for the iteration time $L_k$. We can also derive an expression for the iteration time $L_k$ when the prediction step is executed after the completion of the subsequent measurement task. This scheduling constraint is indicated by the additional edge buffer and the computational overhead of the algorithm. The authors do not propose a method to minimize the latency.

4. ITERATION TIME ANALYSIS

A sampling jitter mitigation approach using an Extended Kalman Filter (EKF) is considered in [13] for the case of periodically sampled measurements with additive jitter caused by uncertainty in delay. The approach utilizes an augmented state, which consists of the current state at iteration $k$, and an additional amount of delayed states depending on the maximum delay. The approach is similar to ours in the sense that multiple state predictions for different delay values are used. However, they do not consider aperiodic sampling as a result of varying execution times, nor do they propose techniques to minimize the delay.

An approach, which like ours utilizes a free-running estimator, is proposed in [14]. Here the authors present a periodically sampling “Any-time” KF approach, where the measurements are queued in a First In, First Out (FIFO) buffer. The estimator selects at each sampling moment a portion of the buffered measurements to optimally estimate the state using an optimization algorithm. The unneeded measurements are flushed from the buffer. This way the prediction and update steps can execute with a smaller sampling period. However, the authors do not consider parallel execution of the estimator. Furthermore, the approach might introduce a large latency in a control loop due to the queuing of measurements in the FIFO
in Fig. 4b, on which there are -1 initial tokens. This negative number of initial token modeling construct has been introduced in [17]. We also assume, that \( \rho_k(a_i) = \rho_k(b_i) \) for all \( i \) and \( k \), under the condition that the actors in both graphs represent the same tasks in the control loop, independently of the task schedule.

Using Eq. (2), we then derive the following start-times for the actors in Fig. 4a:

\[
s_k(a_M) = f_{k-1}(a_C),
\]

\[
s_k(a_U) = \max\{f_{k-1}(a_P), f_k(a_M)\},
\]

\[
s_k(a_P) = \max\{f_{k-1}(a_C), f_k(a_U)\},
\]

\[
s_k(a_C) = f_k(a_U).
\]  

The start-times of the actors in Fig. 4b are the same as in Fig. 4a, with the exception that:

\[
s_k(b_P) = \max\{f_{k-1}(b_C), f_k(b_U), f_{k+1}(b_M)\}.
\]  

We then define \( \rho_k = f_k(a_C) - f_{k-1}(a_C) \) as the execution time of iteration \( k \). Using Eq. (1) and (3), and noting that \( f_k(a_U) \geq f_{k-1}(a_C) \), it can be derived that:

\[
L_k = \max\{\rho_{k-1}(a_P) - \rho_{k-1}(a_C), \rho_k(a_M)\} + \rho_k(a_U) + \rho_k(a_C).
\]  

Similarly for Fig. 4b, the execution time of iteration \( k \) is defined as \( L_k = f_k(b_C) - f_{k-1}(b_C) \). Given that \( f_{k+1}(b_M) \geq f_k(b_U) \), it can be derived that:

\[
L_k = \rho_{k-1}(b_P) + \rho_k(b_M) + \rho_k(b_U) + \rho_k(b_C).
\]  

Because \( L_k \leq L_k \), we conclude that allowing execution of the prediction task in parallel with the control and measurement tasks, reduces the iteration time. We can also conclude from these equations that \( L_k \) can be as large as \( 2L_k \), because the prediction task cannot execute in parallel with the control and measurement task, as shown in the schedule from Fig. 3c. Therefore, our estimation approach can halve the delay introduced in the control loop and can result in a sampling rate which is twice as high compared to the alternative approach.

5. STATE ESTIMATION WITH SAMPLING JITTER

In this section we introduce our discrete time model of the plant under aperiodic sampling, i.e. sampling with jitter. Then we explain why the standard KF algorithm cannot function correctly under these conditions.

5.1. Plant dynamics and discretization

We consider a continuous-time model of a plant described using a first order state-space process function \( f : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_w} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x} \), which may be non-linear, such that:

\[
x(t) = f(t, x(t), w(t), u(t)),
\]  

where \( x(t) \in \mathbb{R}^{n_x} \) is the plant’s state vector, \( w(t) \in \mathbb{R}^{n_w} \) is the process noise vector with known PDF \( \pi_w(w) \), and \( u(t) \) is the actuation input. Here we denote with \( n_x \) the dimension of a vector \( x \).

The state is hidden and is observed at discrete sampling moments \( t_k \) through a measurement function \( h_k : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \to \mathbb{R}^{n_x} \) such that:

\[
z_k = h_k(x_k, v_k),
\]  

where \( \{z_k\}_{k = 0}^{n_k} \) is a vector sequence of measurements, \( \{v_k\}_{k = 0}^{n_k} \) is a measurement noise sequence with PDF \( \pi_v(v) \), and \( x_k := z(t_k) \) is the sampled state at time \( t_k \).

After discretization using a numerical approximation method, Eq. (7) becomes:

\[
x_{k+1} = L_k x_k + W_k u_k + w_k,
\]  

where \( L_k = e^{f_k} L_k \) is a state-transition matrix with \( A(t) \) being the system matrix assumed constant for any \( t \), \( w_k \sim \mathcal{N}(0, Q_k) \) with covariance matrix \( Q_k \) and \( L_k = t_{k+1} - t_k \) is the iteration time described earlier in the paper.

Accordingly, the measurement function is defined as:

\[
h_k(x_k, v_k) = H_k x_k + v_k,
\]  

where \( H_k \) is a measurement matrix and \( v_k \sim \mathcal{N}(0, R_k) \) with covariance \( R_k \).

Although the underlying continuous-time model is linear, the discrete model described with Eq. (10) is non-linear with respect to \( L_k \), which we assume to be a stochastic variable of the model. Additionally the model is time-variant, because it depends on the future sampling moment \( t_{k+1} = t_k + L_k \).

In contrast with periodic sampling, the time step \( T = t_{k+1} - t_k \) is assumed to be constant for any \( k \geq 0 \), thus eliminating the non-linearity and time-dependence. However setting the sampling period to \( T = \max(L_k) \) leads to slower responsiveness of the system, and may cause instabilities in the control loop.

5.2. Kalman Filtering

Kalman filtering is an optimal state estimation algorithm for the case that the discrete model is an LTI state-space system corrupted with Gaussian noise.

A state prediction (or time-update) step computes the state using Eq. (10), prior to sampling a measurement at \( t_{k+1} \) such that:

\[
x_{k+1|k} = F_k x_k|k + W_k u_k,
\]

\[
P_{k+1|k} = Q_k + F_k P_{k|k} F_k^T,
\]
where $P_{k+1|k}$ and $\hat{x}_{k+1|k}$ are the covariance and mean of the prior state PDF $p(x_{k+1}|z_k) = N(\hat{x}_{k+1|k}, P_{k+1|k})$, respectively. Here we denote with $X^T$ the matrix transpose of a matrix $X$ in the above equation and the rest of this paper. Note that $L_k$ was omitted from the argument of $P_k$ for brevity.

The update step improves the predicted state after sampling a measurement at $t_k$ using Eq. (11), such that:

$$x_{k|k} = \hat{x}_{k+1|k} + K_k(z_k - H_kx_{k|k-1}),$$

$$P_{k|k} = P_{k+1|k} - K_k H_k P_{k|k-1},$$

where $P_{k|k}$ is the posterior estimation error covariance and $K_k = P_{k|k-1}H_k^T(H_kP_{k|k-1}H_k^T + R_k)^{-1}$ is the Kalman gain.

From Eq. (12) one can see that the KF is incapable of predicting the state, unless it knows when its previous update and the current prediction tasks are going to finish their execution. Because this event is unknown upfront, the only remaining options are to perform the prediction after a measurement $z_k$ is received, or to perform a correction step after these tasks have finished their execution. However this correction step is very similar to the prediction step itself, and results into a schedule and approach similar to the one of the alternative method described in Section 2.

One may also notice that computing the Kalman gain and the posterior covariance are operations that do not depend on the measurement. This means that computing the posterior state estimate is the only operation performed after receiving a measurement, hence the prediction step is the most time consuming.

6. JITTER-ROBUST STATE ESTIMATION

In this section we first introduce the reader to the PF algorithm, and then present our PF based state-estimation approach which takes the execution times of the tasks into account.

6.1. Particle Filtering

The Particle Filter is an approximate recursive Bayesian filtering approach based on the Monte Carlo simulation method. Since an analytic solution to compute the prior and posterior PDFs does not generally exist for non-linear, non-Gaussian systems, the PF represents them as discrete probability mass functions composed of a number of weighted samples called "particles". Its main strength thus lies in its ability to represent arbitrary distributions, which appear in non-linear and/or non-Gaussian state-space systems. However it is a sub-optimal solution for estimating LTI systems with Gaussian distributed noise, and is always outperformed by the KF under ideal conditions.

The most common algorithm which implements a PF is the SIR algorithm, shown in Algorithm 1. First a set $\{x_{k|k}^i| i=1, ..., N\}$ of $N$ state samples at time $t_{k+1}$ are drawn from a proposal distribution $q(x_{k+1}|z_k)$ with the same support as $p(x_{k+1}|z_k)$. This is the prediction step of the PF algorithm. A common choice for the proposal distribution is the state-transition distribution $p(x_{k+1}|x_k)$. We adopt this choice for the SIR algorithm throughout this paper. Then by utilizing the discrete-time model in (9), or (10) for LTI systems, one can compute each sample before time $t_{k+1}$ according to

$$x_{k|k}^i \sim p(x_k) \text{ for } i=1, ..., N,$$

$$x_{k+1}^i = f_k(x_k^i, w_k^i).$$

where $w_k^i \sim \pi_{w_k}(w)$. Here $\sim$ denotes that a sample is drawn from a distribution.

During the update step each sample is weighted when a measurement becomes available at $t_k$ according to

$$w_{k+1}^i = N^{-1} \text{ for } i=1, ..., N,$$

$$w_{k+1}^i = w_{k+1}^i p(z_k|x_k^i),$$

where $p(z_k|x_k^i)$ is a likelihood function that is chosen based on the given application. In this paper we assume $p(z_k|x_k^i) = \exp(\|z_k - H_kx_k^i\|)$ using Eq. (8), or Eq. (11) in case of an LTI plant. Subsequently all weights are normalized so that their sum equals one. The resulting particle set $\{x_{k+1}^i, w_{k+1}^i|i=1, ..., N\}$ with $N$ the number of particles represents a discrete distribution, which tends to the posterior $p(x_k|z_k)$ as $N \to \infty$. One can use the approximated PDF to estimate the state of the plant and its variance.

After the update step an additional resampling step [18] is applied in case the particles start to diverge and all but one particle have negligible weights. This is known as the sample degeneracy problem, and it is a consequence of choosing the proposal distribution equal to state-transition distribution.

6.2. Jitter-robust Particle Filtering

In Section 2 we introduced the notion of iteration time $L_k$, and how it defines the sampling moments in a control loop, i.e. $t_{k+1} = t_k + L_k$. Continuing on this notion and considering the definition in eq. (9), we assume that $L_k$ is a random variable independent of the state with some probability distribution $\pi_L(L_k)$. Our approach uses an approximation $\pi_L^\prime(L_k)$ of the PDF derived to a certain degree of accuracy. Thus its shape highly depends on the target architecture on which the algorithm is executed, and the amount of interference caused by other tasks executed on the same multiprocessor system.

We make use of $\pi_L^\prime(L_k)$ to estimate the state, given that the time step of the state-transition model is equal to the iteration time $L_k$. This is a consequence of the fact that each sampling moment depends on the total execution time of the control loop tasks. The state estimation problem then amounts to estimating the posterior PDF $p(x_k|z_k, L_{k-1})$. As discussed previously, the dependence of the prediction model and the sampling moments on $L_k$ renders methods based on the KF unsuitable,
because propagation of the state mean and covariance requires upfront knowledge of the time step $L_k$.

In contrast, the Particle Filter allows multiple time predictions of the state by drawing time steps from the estimated iteration time distribution $\pi_L(L_k)$. Suppose that a sample set $\{L_k|i=1,\ldots,N\}$ is drawn from the iteration time PDF $\pi_L(L_k)$, where $N$ is the number of particles. One can then perform the usual prediction step in a PF, by propagating the state for each drawn time step. The proposal distribution is hence selected as $p(x_{k+1}|x_k, L_k)$ based on the plant’s discretized model.

In other words, using Eq. (10) and Eq. (14) for LTI plant models the prediction step becomes:

$$x_{k} \sim p(x_{0}) \quad \text{for} \quad i = 1, \ldots, N,$$

$$L_k \sim \pi_L(L_k),$$

$$x_{k+1} = F_k(L_k) x_k + W_k u_k + w_k,$$

where $F_k(L_k) = e^{A_L(L_k)}$ and $W_k = A^{-1}(F_k + I)B$ given Zero-Order Hold (ZOH) on the input, with $A$ and $B$ being the fixed system and input matrices of the continuous LTI state-space model of the plant respectively. The update step of the algorithm remains unchanged. Thus, line 6 in Algorithm 1 is changed to use Eq. (16).

7. CASE STUDY

In this section we evaluate using simulation the performance of our PF based estimator with the KF based estimator. As an example a plant with a linear state-space representation is selected for which it is known that a KF based approach produces optimal estimation results in case that periodic sampling is applied. It should be noticed that our PF based approach is also suitable for non-linear plants.

7.1. Plant definition

In this case study we compare the performance of the KF with our algorithm, by estimating the state of a Brushed DC Motor (BDCM) from measurements with sampling jitter. We test the KF with periodic sampling and execution as depicted in Fig. 3a, and the alternative KF with aperiodic sampling and execution as depicted in Fig. 3b. Note that we exclude the controller from our evaluations and focus on comparing the estimator performance only. The BDCM is a plant with a continuous time LTI state-space model, which is simple to evaluate and frequently encountered in motor control applications.

An equivalent electrical circuit of the motor and external components used in the simulation setup is illustrated in Fig. 5. The BDCM is driven by an external voltage $V_t$. The armature current of the motor is measured by an external resistance, which is then buffered with an amplifier. There is no mechanical load attached to the shaft of the motor. Thus, the state space vector is $x(t) = [\omega(t)\ t(t)]^T$, where $\omega(t)$ is the angular velocity of the shaft and $i(t)$ the armature current. The input vector is $u(t) = [V_t(t)\ w(t)]$, where $V_t(t)$ is the actuation voltage and $w(t)$ is a disturbance. The system matrix is $A = \begin{bmatrix} \frac{b}{L} & \frac{k}{L} \\ \frac{R}{L} & -\frac{R}{L} \end{bmatrix}$, and the input matrix is $B = \begin{bmatrix} 0 \\ \frac{1}{T} \end{bmatrix}$. The parameters and their respective values are depicted in Table 1.

A sawtooth and square wave signals with fundamental frequency $f_s$ were considered for the actuation disturbance $w(t)$, because they are harmonically rich and aliasing effects caused due to large sampling periods become easier to illustrate. Furthermore such periodic signals are useful to determine the responsiveness of the estimators to varying input, which is not necessarily due to noise. Several values for $f_s$ have been tested, and are also described in Tab. 1.

The amplifier’s output voltage $V_o(t)$ is sampled by an A/D converter. Thus, the measurement vector is $z_k = [V_o(t_k)]$. The measurement noise sequence $v_k \sim \mathcal{N}(0, \sigma_v^2)$ in (11) is thermal noise from the measurement resistor, with $\sigma_v^2 = C_v^2 2(ek_BT^2) R_m$ [19]. Here $T_o$ is the room temperature in Kelvin, $k_B$ is the Boltzmann constant, and $h$ is the Planck constant. The measurement matrix is $H_k = [0 \ G_a R_m]$.

Table 1: Table of parameters for the BDCM model and our simulation setup

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value(s)</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B_a$</td>
<td>Armature resistance</td>
<td>15</td>
<td>mH</td>
</tr>
<tr>
<td>$L$</td>
<td>Armature inductance</td>
<td>3</td>
<td>mH</td>
</tr>
<tr>
<td>$b$</td>
<td>EMF constant</td>
<td>0.0082</td>
<td>V</td>
</tr>
<tr>
<td>$J$</td>
<td>Inertial moment</td>
<td>0.015</td>
<td>N m²/s</td>
</tr>
<tr>
<td>$R_m$</td>
<td>Vacuum friction coef.</td>
<td>0.426</td>
<td>N m/s</td>
</tr>
<tr>
<td>$G_a$</td>
<td>Output voltage amplification</td>
<td>10000</td>
<td></td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Number of iterations per sim.</td>
<td>0.1, 1, 8</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>Operation time mean</td>
<td>0.0100</td>
<td>ms</td>
</tr>
<tr>
<td>$\mu_{ET}$</td>
<td>Execution variance</td>
<td>0.0100</td>
<td>var</td>
</tr>
<tr>
<td>$\sigma_{ET}$</td>
<td>WCET estimation parameter</td>
<td>0.1, 1, k</td>
<td></td>
</tr>
</tbody>
</table>

7.2. Simulation method

In this section we first evaluate the estimation performance under the assumption that the iteration time PDF $\pi_{T_k}(L_k)$ is known at the design time. Then we evaluate whether a mismatch of the shape between $\pi_{T_k}(L_k)$ and its approximation $\pi_{T^*_k}(L_k)$ utilized by the PF significantly affects the estimation results. We perform a Monte Carlo simulation using $\pi_{T_k}(L_k)$ to generate the sequence $\{L_k|k \geq 1\}$. The distribution is truncated below to avoid unrealistic time steps, and has a mean $\mu_{ET} = 1$ms. The standard deviation $\sigma_{ET}$ is increased gradually with each simulation run and thus increasing the sampling jitter. We test a Gaussian and uniform distribution for $L_k$.

To evaluate the performance of the KF and our PF algorithm we generate two sets of sampling times $T_{KF}$ and $T_{PF}$ respectively. The periodically sampling KF has a period $T = \mu_{ET} + \sigma_{ET}$. The algorithm therefore waits for the next sampling moment, in case the tasks in the current iteration $\hat{k}$ finish execution early, i.e. $L_k < T$. Otherwise the next iteration of the algorithm starts immediately. Thus, each sampling time $t_k \in T_{KF}$ is generated such that $t_{k+1} = \max\{T, L_k\} + t_k$, with $t_0 = 0$. The parameter $\alpha$ is used to select the period based on an estimation of the upper execution time bound from $\pi_{T^*_k}$. A high value assumes a pessimistic estimation, while a low value assumes an optimistic one, as depicted in Fig. 6.

The PF and alternative KF assume an aperiodic sampling scheme as described in Section 2. Thus, each sampling time $t_k \in T_{PF}$ is generated such that $t_{k+1} = L_k + t_k$. The iteration intervals $L_k$ of the aperiodic KF are assumed larger such that $L_k = \beta L_k$.
Fig. 6: Selection of the sampling period $T$ for the KF, based on WCET estimation and $\pi_L(L_k)$.

We then use Eqs. (10) and (11) to compute the real state $x_k$, as well as measurements for the KF and PF. To minimize the integration error of the generated measurements (due to ZOH on the input) we have chosen a small constant time step $T_{sim} = 0.05\mu s$ for the simulation model. Furthermore to simplify our simulation setup, the sampling times in $T_{KF}$ and $T_{PF}$ are truncated to the nearest integer multiple of the time step $T_{sim}$.

We use the Root-Mean-Square Error (RMSE) as a performance metric, where $M$ is the total number of iterations, $x_k$ is the true state and $\hat{x}_k$ – the estimated state. As mentioned, the sampling jitter’s standard deviation is increased gradually. We therefore perform a simulation for each new jitter value, starting with $\sigma_{ET} = 0$ (no jitter). We simulated the PF for different number of particles $N$ to show how a larger number of particles increases the robustness of our algorithm to sampling jitter. We also test different values of $\alpha$.

7.3. Results

The simulations have been performed for several values of the parameters as indicated in Table 1. In Fig. 7 the armature current’s estimation RMSE $\varepsilon_2$ for both estimators is plotted against the jitter. Here the simulations are performed given an input sawtooth shaped signal. We also simulated the system given a uniform iteration time distribution which lead to similar results. Finally we test a case when the iteration time is utilized in the update step of the PF, but it does not seem to improve the results substantially.

A key observation is that the periodic KF performs worst for $\alpha = 8$ where the RMSE rapidly increases, even when the input signal is slowly changing. Here the effects of aliasing become apparent and the results confirm that a large sampling period deteriorates the performance. The peaks are also attributed to aliasing. Selecting a period close to the execution time mean on the other hand with $\alpha = 0.1$ deteriorates the performance of the KF for slowly changing input, but seems to improve for a rapidly changing input. A reversed effect is observed for $\alpha = 1$.

Another key observation is that the periodic KF’s RMSE is higher than the RMSE of the KF when no sampling jitter is introduced, but as the jitter increases the performance of the KF deteriorates quickly up to a factor of 10 if we compare the purple and teal curves. In the case of Fig. 7b the PF does not seem to fully outperform the KF for small number of particles, as indicated by the red and green curves. For the rest of the cases, the RMSE of the PF is clearly lower compared to the RMSE of the KF. This so because the PF exploits that smaller execution times result in earlier sampling, which reduces the effect of aliasing and the integration error in the prediction step. As expected for a larger number of particles the result of the PF improves even further.

The final key observation is the aperiodic KF is also outperformed by the PF. Here the iteration times of the algorithm are extended by a factor of $\beta = 1.2$ w.r.t. the PF. Even though the algorithm has perfect knowledge of the iteration intervals it still produces a large error because of larger average sampling period.

We also compare the performance of the PF for the case that the approximated iteration time distribution $\pi_L'(L_k)$ is the same as the simulation model’s $\pi_L(L_k)$, to the case that the distributions are mismatched. For both cases, $\pi_L(L_k)$ is selected to be Gaussian. Thus we compare the PF’s performance in the cases that $\pi_L'(L_k)$ is Gaussian or uniform. The results are shown in Fig. 8, where the RMSE of the PF is again plotted against the jitter. We tried different support intervals for the uniform distribution. In all cases, incorrectly estimating the distribution slightly degrades the performance. It is worth noting however, that for a uniform distribution selecting a small support interval, increases the error for large jitter, while a larger interval increases the error for small jitter.

From these results we can conclude that for this linear motor controller example the PF clearly outperforms the KF in terms of robustness to jitter even if the execution time distribution used by the particle filter differs from the one that occurs in the system.

8. CONCLUSION

In this paper we presented a free-running PF based state-estimation method which is robust to measurement sampling jitter caused by variation in the execution times of its tasks. The variation in the execution times is caused by caches and other shared hardware resources.
in multi processor systems. On these systems estimation and control using periodic sampling is usually not an option due to the large worst-case execution times of the tasks.

Furthermore, the proposed approach minimizes the delay that this estimator introduces if applied in a control loop, which improves the performance of the controller. The delay is reduced, up to a factor 2, by exploiting parallel execution of the tasks in the estimator. Additional parallelism is introduced by allowing the prediction task to start its execution before the next sampling moment. Thus the prediction task must also predict the time between subsequent sampling moments.

The performance of the proposed PF based state-estimation approach is compared with the control performance of a KF based approach using a brushed DC motor example. For this example, which has a linear state-space model, the KF based approach is known to be optimal in case periodic sampling applied. However, simulation results show that our PF based state-estimation approach can outperform the KF based approach, up to a factor 10 in the root-means square error, in case of aperiodic sampling, even if a relative small number of particles is used.

Our PF based state-estimation approach makes use of the PDFs of the execution times of tasks. Usually only a coarse estimate of these PDFs is known at design time. However, simulation results show that the shape of these PDFs hardly affects the estimation error.

The use of a PF instead of a KF makes estimation more robust against sampling jitter, but increases the computational load in the system. This additional load does not result in an additional delay in the control loop, however, because particles can be computed in parallel on a multi processor system.

Our next step is that we will evaluate our approach for non-linear systems for which it is known that a KF based approach is suboptimal even if periodic sampling is applied. Furthermore, our plan is to evaluate the approach using an experimental hardware setup and evaluate the improvement in control performance as a result of the decreased average delay of the estimator.

REFERENCES


