# SYNCHRONIZATION OF INPUT AND OUTPUT MEASUREMENTS USING A KALMAN FILTER

Joachim Fox, Laboratory of Process Automation (LPA), Saarland University, Germany, e-mail: j.fox@lpa.uni-saarland.de†

**ABSTRACT**
Kalman filters are used when data from different sensors is combined to obtain a suboptimal estimation of a dynamic system's state. In most applications, the sensor data enters the filter in two places: some data is fed to the inputs of the dynamic system while other data is used as reference measurements of the system's outputs. In order to yield the best possible estimations, both types of sensors have to be well synchronized, but a hardware synchronization mechanism is not always available. In this paper, the Kalman filter is modified to estimate both the system's state and the time delay between input and output measurements. Simulations show that an accurate software synchronization can be achieved by using this method and that the state estimates improve largely.

**KEY WORDS**
Kalman filter, sigma-point Kalman filter, latency estimation, asynchronous measurements

## 1 Introduction

When different kinds of sensor data must be combined to estimate the state of a discrete dynamic system which is subject to process and measurement noise, the Kalman filter is the method of choice in a wide range of applications [1, 2]. For example in inertial navigation, Kalman filters and variants of it are used frequently to estimate the orientation, position and velocity of the moving object by using inertial measurements (accelerations and turn rates) on the one hand and external reference measurements (provided by the Global Positioning System, GPS, for example) on the other [3, 4, 5].

One aspect which is rarely treated in the literature is how to deal with data that is not perfectly synchronized. However, this problem has been recognized as it arises in wireless networks [6], driver assistance systems [7], and inertial navigation. In some applications, a hardware synchronization is provided: many GPS receivers can output a one-pulse-per-second signal which can be used to trigger the inertial sensors [8]. But not in all applications where Kalman filters are used is such a hardware synchronization possible.

Therefore, in this paper a method is introduced which includes the latency time between two sets of measurements in the estimation process. It is assumed that a coarse synchronization has already been done with other methods (correlation-based for example) so that the Kalman filter estimates latency times not greater than one sample time of the system. Here, extended Kalman filter (EKF) approaches will be compared to methods that use so-called sigma-point Kalman filters (SPKF), a new and more accurate variant of Kalman filtering. Simulations will show that the modified filters are able to estimate the latency very accuarately and that the estimation of the system's states will improve largely over the standard Kalman filter without latency estimation.

The paper is structured as follows: first, a mathematical formulation of the synchronization problem is given in section 2. Section 3 describes the newly derived Kalman latency estimators. In section 4, simplified variants of these filters which use interpolation techniques are proposed. Simulation results are presented in section 5. The paper closes with conclusions and an outlook.

## 2 Problem statement

Let $\mathbf{x}_k$ be the state of a discrete dynamic system at the time $k$ with the inputs $\mathbf{u}$, the outputs $\mathbf{y}$, the process noise $\mathbf{n}^{\mathrm{s}}$, the measurement noise $\mathbf{n}^{\mathrm{m}}$, the system function $\mathbf{f}$, and the measurement function $\mathbf{g}$. The system is sampled with the step size $\Delta t$ (vectors stand out in bold face):

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{n}^{\mathrm{s}}_k, \Delta t) \quad (1)$$

$$\mathbf{y}_k = \mathbf{g}(\mathbf{x}_k, \mathbf{n}^{\mathrm{m}}_k) \quad (2)$$

A Kalman-like filter shall be applied to calculate an estimate $\hat{\mathbf{x}}_k$ of the system's state at time $k$ by using all measurements of $\mathbf{y}$ that are available up to the time $k$. When using a standard extended Kalman filter, one needs to know the Jacobians of the system and measurement functions with respect to the state and the noise. These are denoted by $\mathbf{A} = \partial\mathbf{f}/\partial\mathbf{x}$, $\mathbf{B}^{\mathrm{n}} = \partial\mathbf{f}/\partial\mathbf{n}^{\mathrm{s}}$, $\mathbf{C} = \partial\mathbf{g}/\partial\mathbf{x}$ and $\mathbf{D}^{\mathrm{n}} = \partial\mathbf{f}/\partial\mathbf{n}^{\mathrm{m}}$ (these matrices usually depend on $k$; however, for a clearer notation, $k$ is omitted wherever this cannot lead to misunderstandings). Then it is well known [1, 2] that the EKF equations can be written as:

$$\hat{\mathbf{x}}^-_{k+1} = \mathbf{f}(\hat{\mathbf{x}}^+_k, \mathbf{u}_k, \mathbf{0}, \Delta t) \quad (3)$$

$$\hat{\mathbf{P}}^-_{k+1} = \mathbf{A}_k\hat{\mathbf{P}}^+_k\mathbf{A}^T_k + \mathbf{B}^{\mathrm{n}}_k\mathbf{Z}_k(\mathbf{B}^{\mathrm{n}}_k)^T \quad (4)$$

$$\mathbf{K}_k = \hat{\mathbf{P}}^-_{xy,k}(\hat{\mathbf{P}}^-_{yy,k})^{-1} = \hat{\mathbf{P}}^-_k\mathbf{C}^T_k\left(\mathbf{C}_k\hat{\mathbf{P}}^-_k\mathbf{C}^T_k + \mathbf{D}^{\mathrm{n}}_k\mathbf{W}_k\mathbf{D}^{\mathrm{n}T}_k\right)^{-1} \quad (5)$$

†State estimation in robotics and inertial navigation is a research topic of the LPA. The author would like to thank Prof. H. Janocha, head of the LPA, for his scientific and financial support of this work.

$$\hat{\mathbf{y}}_k^- = \mathbf{g}(\hat{\mathbf{x}}_k^-, \mathbf{0}) \quad (6)$$

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\left(\mathbf{y}_k - \hat{\mathbf{y}}_k^-\right) \quad (7)$$

$$\hat{\mathbf{P}}_k^+ = \hat{\mathbf{P}}_k^- - \mathbf{K}_k\mathbf{C}_k\hat{\mathbf{P}}_k^- \quad (8)$$

The superscript '-' denotes a-priori estimates (without the knowledge of the latest measurement $\mathbf{y}_k$), '+' denotes a-posteriori estimates (with knowledge of $\mathbf{y}_k$). The process and measurement noises are gaussian, zero-mean and white with covariance matrices $\mathbf{Z}$ and $\mathbf{W}$ respectively[1]. $\mathbf{n}_k^{\mathrm{s}}$, $\mathbf{n}_k^{\mathrm{m}}$ and $\mathbf{x}_k$ are independent of each other. $\hat{\mathbf{P}}$ is an estimate of the state's covariance matrix. (3) and (4) are referred to as the prediction step, (5)–(8) are called update.

It can be seen that measurements of real physical quantities enter the algorithm at two points: firstly, the system's inputs $\mathbf{u}$ are usually based on sensor data, and secondly, the update measurements $\mathbf{y}$ are provided by the measurement equipment. Both sets of data can come from distinct hardware devices, so one has to take care that the measurements are well synchronized.

Let us assume that both measurements have the sample time $\Delta t$ but that (the real, continuous input signal) $\mathbf{u}(t)$ is sampled at the times $k\Delta t$ while $\mathbf{y}(t)$ is sampled at the times $k\Delta t + \tau$ with $\tau$ being a latency time. We shall further assume that $|\tau| < \Delta t$, i.e. a coarse synchronization has already been performed by other techniques, e.g. correlation analysis.

The task to be tackled in this paper can now be formulated as follows: *Given is the discrete model* (1), (2) *of a system and the measurements* $\mathbf{u}(k\Delta t), \mathbf{y}(k\Delta t + \tau)$*; estimate the state of the system as well as the latency time* $\tau$ *using a Kalman-like filter.*

## Sigma-point filter variants

The term 'Kalman-like filter' needs further explaination. In recent years, variants of the Kalman filter have emerged which are based on the so-called sigma-point approach [9, 10, 11]. Instead of using Jacobians, they implicitely compute numerical derivatives by using weighted means and weighted covariances. There are two basic advantages of this: firstly, the filter design is eased because there is no need to analytically derive the derivatives, and secondly, their approximation accuracy is higher because the system and measurement functions are evaluated at a number of points instead of at the current best estimate only.

The Laboratory of Process Automation's recent research has covered some topics of sigma-point Kalman filtering (SPKF) [3]. Therefore, the latency estimation has been developed both for the EKF and for the SPKF to allow for comparisons. Since the basic structure of the SPKF follows the same lines as the EKF, the latency estimation with both types of filters is very similar and is treated in this paper.

A very brief overview over sigma-point filtering is given here: before each prediction and before each update step, choose a set $\{\mathcal{X}_{i,k}^{+/-}\}$ of 'representative' points in the state space so that the mean and the covariance of $\mathbf{x}_k$ are reflected exactly in this set (most variants use $i = 0 \ldots 2n$ of these 'sigma-points' whereby $n$ is the dimension of the state vector). Now compute $\mathcal{X}_{i,k+1} = \mathbf{f}(\mathcal{X}_{i,k}^+)$ and $\mathcal{Y}_{i,k} = \mathbf{g}(\mathcal{X}_{i,k}^-)$. Use weighted means of the sets $\{\mathcal{X}_{i,k+1}\}$ and $\{\mathcal{Y}_{i,k}\}$ to compute $\hat{\mathbf{x}}_{k+1}^-$ and $\hat{\mathbf{y}}_k^-$. By calculating weighted covariances of these sets, you get $\hat{\mathbf{P}}^-$, $\hat{\mathbf{P}}_{xy}^-$ and $\hat{\mathbf{P}}_{yy}^-$. Depending on the SPKF variant, different representative points, and corresponding weights must be chosen. For details, refer to [3, 11].

A non-additive system or measurement noise requires special treatment in the SPKFs. Sigma-points for the noise have to be chosen in addition to the state sigma-points. Since the noise is usually assumed to be independent of the state, the noise sigma points $\{\mathcal{N}_i\}$ can also be chosen indepenently and the two sets of points can be combined: $\{(\mathcal{X}_i, \mathbf{0})\} \cup \{(\hat{\mathbf{x}}, \mathcal{N}_i)\}$. As you will see below, the latency estimation needs to include 'noise states' in the state vector anyway, so no special care has to be taken when using the SPKF variant of the estimators.

## 3 The new latency estimation filter

To derive a Kalman filter which estimates both the states and the latency time, we make use of the explicit inclusion of the sample time $\Delta t$ in the system function $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{n}_k^{\mathrm{s}}, \Delta t)$. We augment the original state vector by the latency $\tau$ and by a system noise state $\mathbf{x}^{\mathrm{ns}}$. The latter is necessary because the system's output is now influenced by both the original measurement noise and the system noise (as will be seen from (10)) – therefore the (augmented) output noise is no longer independent from the system noise which has to be modeled. Under the additional assumption that $\tau \geq 0$, we can then give an augmented system (marked by $'$) whose outputs are delayed compared to the inputs:

$$\mathbf{x}'_{k+1} = \begin{pmatrix} \mathbf{x}_{k+1} \\ \tau_{k+1} \\ \mathbf{x}_{k+1}^{\mathrm{ns}} \end{pmatrix} = \begin{pmatrix} \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_k^{\mathrm{ns}}, \Delta t) \\ \tau_k \\ \mathbf{n}_k^{\mathrm{s}} \end{pmatrix} \quad (9)$$

$$\mathbf{y}'_k = \mathbf{y}(k\Delta t + \tau) = \mathbf{g}(\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_k^{\mathrm{ns}}, \tau), \mathbf{n}_k^{\mathrm{m}}) \quad (10)$$

Now, an EKF can be applied to the augmented system. It needs to know the new Jacobians which are given by

$$\mathbf{A}' = \begin{pmatrix} \mathbf{A} & \mathbf{0} & \mathbf{B}^{\mathrm{n}} \\ \mathbf{0} & 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}, \quad \mathbf{B}' = \begin{pmatrix} \mathbf{B} \\ 0 \\ \mathbf{0} \end{pmatrix} \quad (11)$$

$$\mathbf{B}^{\mathrm{n}\,\prime} = \begin{pmatrix} \mathbf{0} & 0 & \mathbf{I} \end{pmatrix}^T, \quad \mathbf{D}^{\mathrm{n}\prime} = \mathbf{D}^{\mathrm{n}} \quad (12)$$

$$\mathbf{C}' = \begin{pmatrix} \mathbf{C}\mathbf{A} & \mathbf{C}\partial\mathbf{f}/\partial\Delta t & \mathbf{C}\mathbf{B}^{\mathrm{n}} \end{pmatrix} \quad (13)$$

$\mathbf{I}$ is the identity matrix. Note that $\mathbf{C}'$ depends on $\hat{\tau}^-$ because the derivatives are evaluated at the current best esti-

[1] Note that Gaussianess is more restrictive than necessary for Kalman filters, but it is a sufficient condition.

mates. The noises and their covariances are the same as in the standard EKF.

In the case of a linear system model, $\mathbf{C}'$ and thus the Kalman gain $\mathbf{K}'$ read:

$$\mathbf{C}' = \mathbf{C}\left(\mathbf{A} \quad \frac{\partial \mathbf{A}}{\partial \Delta t}\hat{\mathbf{x}}^- + \frac{\partial \mathbf{B}}{\partial \Delta t}\mathbf{u} \quad \mathbf{B}^{\mathrm{n}}\right) \quad (14)$$

$$\mathbf{K}' = \hat{\mathbf{P}}^{-'}\mathbf{C}'^T\Big(\mathbf{C}'\hat{\mathbf{P}}^{-'}\mathbf{C}'^T + \mathbf{D}^{\mathrm{n}}\mathbf{W}\mathbf{D}^{\mathrm{n}T}\Big)^{-1} \quad (15)$$

Here, $\hat{\mathbf{P}}^{-'}$ is the augmented state's covariance matrix. At time 0, the part of it reflecting $\tau$ must be initialized. $\Delta t^2$ seems a reasonable choice because we assumed that $|\tau| < \Delta t$ (the variance of a uniformly distributed $\tau$ would be $\Delta t^2/12$, but we are using a Gaussian model of the initial state; however, the choice of the initial variance is not critical as long as it is not too small which is guaranteed by setting it to $\Delta t^2$).

Generally, even if the original system is linear, an extended Kalman filter has to be used because the system function $\mathbf{f}$ will introduce nonlinearities in $\Delta t$ almost for sure. The derivation of $\mathbf{f}$ with respect to $\Delta t$ is the only additional information required by the new filter (called SyncKF from now on). In most cases, however, it is easily available as $\mathbf{f}$ is usually a polynomial in $\Delta t$, being derived from the continuous system's transition matrix.

Since SPKFs are based on numerical derivatives, $\partial \mathbf{f}/\partial \Delta t$ has not to be calculated explicitely when using them as latency estimators (called SyncSPKF from now on). As explained above, they are based on multiple evaluations of the system and measurement equations (in order to compute weighted means and covariances) and can therefore easily be implemented from (9) and (10). Looking at (10), one sees that now in the update step, not only $\mathbf{g}$ has to be multiply evaluated but also $\mathbf{f}$, therefore the computation time will increase. In many systems, $\mathbf{f}$ is more complex than $\mathbf{g}$, thus the rise may be significant. This disadvantage will be evaded with the SyncSPKF- presented in section 4.2.

As it can be seen from the simulation results in section 5, a very good estimation accuracy can be achieved with both the SyncKF and the SyncSPKF if $0 \leq \tau < \Delta t$. Let us now take a closer look at this restriction and discuss options how to overcome it.

# 4 Approximate latency estimation

Before we turn to approximate solutions of the above described problem, we first answer the question why we had to restrict the latency to $0 \leq \tau < \Delta t$. The system function $\mathbf{f}$ is usually a discrete-time approximation of a continuous physical system. This function is only valid for $\Delta t \geq 0$ because otherwise states and inputs of the previous timestep would have to be taken as arguments of $\mathbf{f}$, or the output would be non-causal. The second restriction, $\tau < \Delta t$, is not a theoretical necessity; but $\tau > \Delta t$ would impose an unnecessary negligence of available data (inputs at $k+1$) on the system.

Both restrictions could be overcome by setting $\hat{\tau} := \hat{\tau} - \Delta t$ as soon as the estimated latency is larger than a step size (or $\hat{\tau} := \Delta t - \hat{\tau}$ if $\tau$ is smaller than 0), and by delaying the output measurements by one step (or one 'negative' step). This would result in a filter of variable structure. While the stability and convergence of extended Kalman filters cannot be proven generally (and therefore, usually simulations serve as 'proofs'), the attempt to grant stability of such a switching estimator would be even harder. That is why we will now present approximations of the above given results. These approximations have the advantage that explicit knowledge of the dependency of $\mathbf{f}$ on $\Delta t$ is not required anymore. This simplifies the filter design and additionally requires less computation time. Moreover, the approximations are valid for unrestricted $\tau$ (however the first one performs better for positive $\tau$ while the second one shows better results for $\tau \leq 0$).

## 4.1 Approximate latency estimator for $\tau \geq 0$

The basic idea behind the simplifications is the following: assuming that $\Delta t$ is sufficiently small, the trajectories of the physical system's output can be approximated by a straight line between the two discrete states $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$:

$$\begin{aligned}
&\mathbf{y}(k\Delta t + \tau) \approx \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k^{\mathrm{m}}) + \\
&\quad \Big(\mathbf{g}(\mathbf{x}_{k+1}, \mathbf{n}_{k+1}^{\mathrm{m}}) - \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k^{\mathrm{m}})\Big)\frac{\tau}{\Delta t} \\
&= \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k^{\mathrm{m}}) + \\
&\quad \Big(\mathbf{g}(\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{n}_k^{\mathrm{s}}, \Delta t), \mathbf{n}_{k+1}^{\mathrm{m}}) - \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k^{\mathrm{m}})\Big)\frac{\tau}{\Delta t}
\end{aligned} \quad (16)$$

In the case of a linear system, this can be written as

$$\begin{aligned}
\mathbf{y}(k\Delta t + \tau) &\approx \mathbf{C}_k\mathbf{x}_k + (\mathbf{C}_{k+1}\mathbf{x}_{k+1} - \mathbf{C}_k\mathbf{x}_k)\frac{\tau}{\Delta t} \\
&\quad + \mathbf{D}_k^{\mathrm{n}}\mathbf{n}_k^{\mathrm{m}}(1 - \frac{\tau}{\Delta t}) + \mathbf{D}_{k+1}^{\mathrm{n}}\mathbf{n}_{k+1}^{\mathrm{m}}\frac{\tau}{\Delta t} \\
&= \Big(\mathbf{C}_k + (\mathbf{C}_{k+1}\mathbf{A}_k - \mathbf{C}_k)\frac{\tau}{\Delta t}\Big)\mathbf{x}_k \\
&\quad + \mathbf{C}_{k+1}(\mathbf{B}_k\mathbf{u}_k + \mathbf{B}^{\mathrm{n}}\mathbf{n}_k^{\mathrm{s}})\frac{\tau}{\Delta t} \\
&\quad + \mathbf{D}_k^{\mathrm{n}}\mathbf{n}_k^{\mathrm{m}}(1 - \frac{\tau}{\Delta t}) + \mathbf{D}_{k+1}^{\mathrm{n}}\mathbf{n}_{k+1}^{\mathrm{m}}\frac{\tau}{\Delta t}.
\end{aligned} \quad (17)$$

We can see that the measurement is now influenced by $\mathbf{n}_k^{\mathrm{s}}$, $\mathbf{n}_k^{\mathrm{m}}$, and $\mathbf{n}_{k+1}^{\mathrm{m}}$. Therefore the state vector of the new system (marked with $''$) must include all three noise terms (where the superscript 'nmf' stands for 'future measurement noise'):

$$\mathbf{x}_{k+1}'' = \begin{pmatrix} \mathbf{x}_{k+1} \\ \tau_{k+1} \\ \mathbf{x}_{k+1}^{\mathrm{ns}} \\ \mathbf{x}_{k+1}^{\mathrm{nm}} \\ \mathbf{x}_{k+1}^{\mathrm{nmf}} \end{pmatrix} = \begin{pmatrix} \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_k^{\mathrm{ns}}, \Delta t) \\ \tau_k \\ \mathbf{n}_k^{\mathrm{s}} \\ \mathbf{x}_k^{\mathrm{nmf}} \\ \mathbf{n}_k^{\mathrm{m}} \end{pmatrix} \quad (18)$$

The new Kalman matrices then are given by

$$\mathbf{A}'' = \begin{pmatrix} \mathbf{A}' & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}, \quad \mathbf{B}^{\mathrm{n}''} = \begin{pmatrix} \mathbf{B}^{\mathrm{n}'} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \quad (19)$$

$$
\begin{aligned}
\mathbf{B}'' &= \begin{pmatrix} \mathbf{B}'^T & \mathbf{0} & \mathbf{0} \end{pmatrix}^T \\
\mathbf{C}''_k &= \Big( \mathbf{C}_k + (\mathbf{C}_{k+1}\mathbf{A}_k - \mathbf{C}_k)\frac{\hat{\tau}_k^-}{\Delta t}, \\
&\qquad (\mathbf{g}(\mathbf{f}(\hat{\mathbf{x}}_k^-, \mathbf{u}_k, \mathbf{0}, \Delta t), \mathbf{0}) - \mathbf{g}(\hat{\mathbf{x}}_k^-, \hat{\mathbf{x}}_k^{\mathrm{nm}-}))\frac{1}{\Delta t}, \\
&\qquad \mathbf{C}_{k+1}\mathbf{B}_k^{\mathrm{n}}\tfrac{\hat{\tau}_k^-}{\Delta t}, \quad \mathbf{D}_k^{\mathrm{n}}(1 - \tfrac{\hat{\tau}_k^-}{\Delta t}), \quad \mathbf{D}_{k+1}^{\mathrm{n}}\tfrac{\hat{\tau}_k^-}{\Delta t} \Big).
\end{aligned} \tag{20}
$$

Note that $\hat{\mathbf{x}}_k^{\mathrm{ns}-} = \mathbf{0}$ and $\hat{\mathbf{x}}_k^{\mathrm{nmf}-} = \mathbf{0}$. This estimator has no measurement noise anymore because it is part of the system's state. Therefore, it also has no matrices $\mathbf{W}$ and $\mathbf{D}^{\mathrm{n}}$, i.e. $\hat{\mathbf{P}}_{yy}^- = \mathbf{C}''\hat{\mathbf{P}}^-\mathbf{C}''^T$. For $0 \leq \tau \leq \Delta t$, this approximation is based on a linear interpolation of the output. If $\tau$ is outside this interval, (16) forms an extrapolation. Assuming a smooth trajectory, the approximation will still be valid outside (and close to) the interval, but the best results are to be expected within the interval. In the following, this approximation is referred to as SyncKF+ . As in the original latency estimator, replacing the EKF with an SPKF is possible by using a straightforward implementation of the augmented system model (18) and the modified measurement equation (16). This SPKF variant is called SyncSPKF+.

This first approximation needs to know $\mathbf{u}_k$ to compute $\hat{\mathbf{y}}_k$ and $\hat{\mathbf{x}}_k^+$. While the algorithm is still causal, data is needed earlier than in the original Kalman filter which must not know $\mathbf{u}_k$ before the computation of $\hat{\mathbf{x}}_{k+1}^-$. Also, when $\mathbf{n}_k^{\mathrm{m}}$ is not stationary, $\mathbf{W}$ has to be known one step in advance, but this usually bears no problem to the system designer. With the second approximation introduced in the following section, these minor disadvantages can be circumvented.

## 4.2 Approximate latency estimator for $\tau \leq 0$

As already mentioned, in the original estimator as well as in the SyncSPKF+, multiple function evaluations of $\mathbf{f}$ and $\mathbf{g}$ are required to calculate $\hat{\mathbf{y}}^-$. If the evaluation of $\mathbf{f}$ takes much more time than that of $\mathbf{g}$, one would wish to avoid these extra computations. To do so, one can use the interpolation technique described above on the interval $(k-1, k)$ inspite of $(k, k+1)$:

$$
\begin{aligned}
&\mathbf{y}(k\Delta t + \tau) \approx \\
&\quad \mathbf{g}(\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{n}_{k-1}^{\mathrm{s}}, \Delta t), \mathbf{n}_k^{\mathrm{m}})(1 + \tfrac{\tau}{\Delta t}) \\
&\quad -\mathbf{g}(\mathbf{x}_{k-1}, \mathbf{n}_{k-1}^{\mathrm{m}})\tfrac{\tau}{\Delta t}
\end{aligned} \tag{21}
$$

Again, we augment the state vector as in the SycnKF+, i.e. $\mathbf{x}''' = \mathbf{x}''$. In order to compute $\hat{\mathbf{y}}_k^-$, we have to take into account all available information, i.e. we will use $\hat{\tau}_k^-$ and $\hat{\mathbf{x}}_{k-1}^+$ when evaluating (21)[2]. $\hat{\mathbf{P}}_{xy,k}^{-'''}$ is the covariance between the estimation error of $\hat{\mathbf{x}}_k^-$ and the estimation error of $\hat{\mathbf{y}}_k^-$, however the variable used in (21) is $\hat{\mathbf{x}}_{k-1}^+$. Therefore, the computations of $\hat{\mathbf{P}}_{xy,k}^{-'''}$ and $\hat{\mathbf{P}}_{yy,k}^{-'''}$ have to be rewritten:

$$
\begin{aligned}
\mathbf{C}'''_k &= \Big( \mathbf{C}_k\mathbf{A}_{k-1}(1 + \tfrac{\hat{\tau}_{k-1}^+}{\Delta t}) - \mathbf{C}_{k-1}\tfrac{\hat{\tau}_{k-1}^+}{\Delta t}, \\
&\qquad (\mathbf{g}(\mathbf{f}(\hat{\mathbf{x}}_{k-1}^+, \mathbf{u}_{k-1}, \hat{\mathbf{x}}_{k-1}^{\mathrm{ns}+}, \Delta t), \mathbf{0}) \\
&\qquad -\mathbf{g}(\hat{\mathbf{x}}_{k-1}^+, \hat{\mathbf{x}}_{k-1}^{\mathrm{nm}+}))/\Delta t, \\
&\qquad \mathbf{C}_k\mathbf{B}_{k-1}^{\mathrm{n}}(1 + \tfrac{\hat{\tau}_{k-1}^+}{\Delta t}), \quad \mathbf{D}_k^{\mathrm{n}}(1 + \tfrac{\hat{\tau}_{k-1}^+}{\Delta t}), \\
&\qquad -\mathbf{D}_{k-1}^{\mathrm{n}}\tfrac{\hat{\tau}_{k-1}^+}{\Delta t} \Big) \\
\hat{\mathbf{P}}_{xy,k}^{-'''} &= \mathbf{A}''_{k-1}\hat{\mathbf{P}}_{k-1}^{+'''}\mathbf{C}_k'''^T \\
\hat{\mathbf{P}}_{yy,k}^{-'''} &= \mathbf{C}'''_k\hat{\mathbf{P}}_{k-1}^{+'''}\mathbf{C}_k'''^T
\end{aligned} \tag{22}
$$

This formulation now states an interpolation for $-\Delta t \leq \tau \leq 0$, which is why the new approach will be termed SyncKF- (or SyncSPKF- resp.).

It may not be obvious at first sight why the SPKF variant of this approximation requires less function evaluations than the SyncSPKF+: (21) still contains the system function. But remember that $\mathbf{f}(\mathcal{X}_{i,k-1}^-)$ has already been evaluated during the prediction step. Since $\hat{\tau}_k^- = \hat{\tau}_k^+$, we can reuse these computations in the update equations and do not have to evaluate $\mathbf{f}$ anew. To state it more clearly, the full SyncSPKF- algorithm is given now. In order to make the algorithm more readable, the $'''$ superscripts have been dropped from the state variables and the sigma-points. $\mathrm{diag}(\cdot)$ is a block-diagonal matrix built from the matrices '$\cdot$'.

$$
\begin{aligned}
\mathcal{X}_{i,k} &= \mathbf{f}(\mathcal{X}_{i,k-1}^+, \mathbf{u}_k, \Delta t) \\
\mathcal{Y}_{i,k} &= \mathbf{g}(\mathcal{X}_{i,k})(1 + \frac{\tau_{i,k}}{\Delta t}) - \mathbf{g}(\mathcal{X}_{i,k-1}^+)\frac{\tau_{i,k}}{\Delta t} \\
\hat{\mathbf{x}}_k^- &= \sum_{i=0}^{r-1} w_i^m \mathcal{X}_{i,k}\,, \quad \hat{\mathbf{y}}_k^- = \sum_{i=0}^{r-1} w_i^m \mathcal{Y}_{i,k} \\
\hat{\mathbf{P}}_k^- &= \sum_{i=0}^{r-1}\sum_{j=0}^{r-1} w_{ij}^c \mathcal{X}_{i,k}\mathcal{X}_{j,k}^T + \mathrm{diag}(\mathbf{0}, 0, \mathbf{Z}_k, \mathbf{0}, \mathbf{W}_k) \\
\hat{\mathbf{P}}_{xy,k}^- &= \sum_{i=0}^{r-1}\sum_{j=0}^{r-1} w_{ij}^{cc} \mathcal{X}_{i,k}\mathcal{Y}_{j,k}^T \\
\hat{\mathbf{P}}_{yy,k}^- &= \sum_{i=0}^{r-1}\sum_{j=0}^{r-1} w_{ij}^c \mathcal{Y}_{i,k}\mathcal{Y}_{j,k}^T \\
\hat{\mathbf{P}}_k^- &= \sum_{i=0}^{r-1}\sum_{j=0}^{r-1} w_{ij}^c \mathcal{X}_{i,k}\mathcal{X}_{j,k}^T \\
\hat{\mathbf{x}}_k^+ &= \hat{\mathbf{x}}_k^- + \hat{\mathbf{P}}_{xy,k}^-(\hat{\mathbf{P}}_{yy,k}^-)^{-1}(\mathbf{y}_k - \hat{\mathbf{y}}_k^-) \\
\hat{\mathbf{P}}_k^+ &= \hat{\mathbf{P}}_k^- - \hat{\mathbf{P}}_{xy,k}^-(\hat{\mathbf{P}}_{yy,k}^-)^{-1}\hat{\mathbf{P}}_{xy,k}^{-T}
\end{aligned} \tag{23}
$$

($r$ is the number of sigma points; the choice of $\mathcal{X}_i^+$ and the weights $w_i^{\mathrm{m}}$, $w_i^{\mathrm{c}}$ and $w_i^{\mathrm{cc}}$ is dependent on the SPKF variant; see [3, 11] for details). Readers familiar with the SPKF will note that the recalculation of sigma-points before the update step cannot be found in this algorithm anymore. The sigma-point calculation uses a Cholesky factorization and takes most of the computing time of the algorithm – there-

[2] To be precise, we calculate $\hat{\mathbf{y}}_k^- = \mathrm{E}\{\mathbf{g}(\mathbf{x}_k, \mathbf{n}_k^{\mathrm{m}})(1 + \frac{\tau}{\Delta t}) - \mathbf{g}(\mathbf{x}_{k-1}, \mathbf{n}_{k-1}^{\mathrm{m}})\frac{\tau}{\Delta t} | \mathbf{y}_k, \mathbf{y}_{k-1}, \ldots\}$ as the expectation of (21) conditioned on the past measurements, see [2].

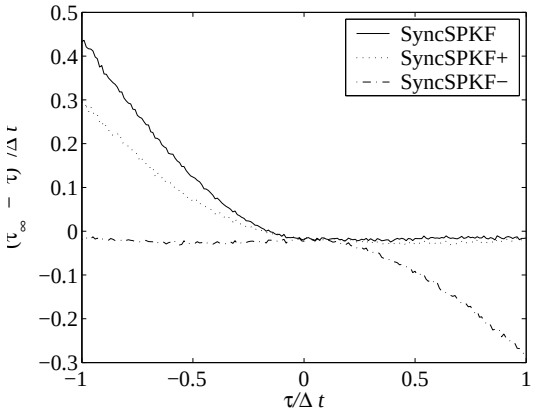


Figure 1. Error of the end value ($k$=500) of the latency estimation $\hat{\tau}_k/\Delta t$

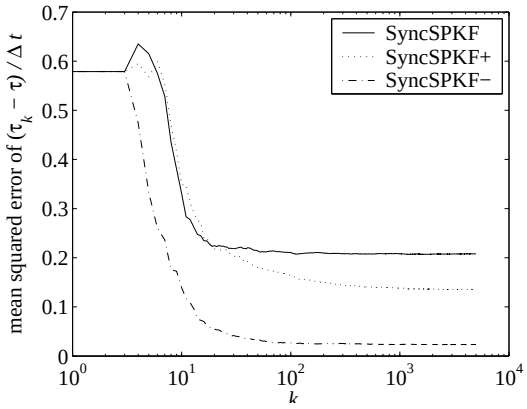


Figure 2. Root of the mean squared latency estimation error (100 runs, $\tau < 0$).

fore one might think that the new latency estimator is actually faster than the original SPKF. Of course, this is not the case, because the new algorithm works on an augmented state vector!

## 5 Simulation results

The performance of the filters proposed here has been studied on a simulated linear second order system. Therefore, the algorithms have been implemented in Matlab/Simulink and tested on a springg-mass-damper system with the spring constant $c$, the mass $m$ and the damping factor $d$:

$$\dot{\mathbf{x}}(t) = \begin{pmatrix} 0 & 1 \\ -\frac{c}{m} & -\frac{d}{m} \end{pmatrix} \mathbf{x}(t) + \begin{pmatrix} 0 \\ -\frac{1}{m} \end{pmatrix} (u(t) + n^{\mathrm{s}}(t)) \quad (24)$$

$$y(t) = (1 \quad 0)\, \mathbf{x}(t+\tau) + \mathbf{n}^{\mathrm{m}}(t+\tau) \quad (25)$$

The state's first element $x_1$ is the position, the second element $x_2$ the velocity of the mass. The scalar input $u$ corresponds to an external force which is measured and disturbed by a noise $n^{\mathrm{s}}$. A second-order discretization of this system results in

$$\mathbf{x}_{k+1} = \begin{pmatrix} -\frac{\Delta t^2}{2m} \\ -\frac{\Delta t}{m} + \frac{d}{2m^2}\Delta t^2 \end{pmatrix} (u_k + n_k^{\mathrm{s}}) + \begin{pmatrix} 1 - \frac{c}{2m}\Delta t^2 & \Delta t - \frac{d}{2m}\Delta t^2 \\ -\frac{c}{m}\Delta t + \frac{cd}{2m^2}\Delta t^2 & 1 - \frac{d}{m}\Delta t + (\frac{d^2}{2m^2} - \frac{c}{2m})\Delta t^2 \end{pmatrix} \mathbf{x}_k$$

$$y_k = (1 \quad 0)\, \mathbf{x}(k\Delta t + \tau) + \mathbf{n}^{\mathrm{m}}(k\Delta t + \tau)$$

The sample time of the simulation was $\Delta t = 0,1\,\mathrm{s}$. The latency $\tau$ has been varied in 200 steps from $-\Delta t$ to $\Delta t$. $u_k$ has been chosen to be time discrete white noise of variance 1; for the continuous system, the value of the input had been held constant between two samples. $n^{\mathrm{s}}$ was white with variance $10^{-2}$, $n^{\mathrm{m}}$ had the variance $10^{-5}$. $c$, $m$ and $d$ were set to 1.

Since the original system with latency 0 is linear, a Kalman filter is the optimal estimator – no EKF or SPKF is needed. As explained above, $\tau \neq 0$ introduces nonlinearities into the system so that a standard Kalman filter isn't applicable anymore. However, the system is still mostly linear (the system equation is truly linear while the output equation contains the $\tau$-nonlinearity). Because of this, the estimation results of the SPKF are not expected to be much better than the ones of the EKF – the SPKF is known to deal better with heavy nonlinearities and to show the same performance as the KF on linear systems. Indeed the simulations show that both the EKF and the SPKF variants operate on the same level of estimation accuracy. Therefore, in the plots, only the SPKF results are shown. The EKF results can be seen in table 1.

Figure 1 shows the error $\hat{\tau} - \tau$ after 500 s for the SPKF latency estimators. They are plotted over $\tau$. Not surprisingly, the SyncSPKF and the SyncSPKF+ show a better performance for $\tau > 0$ and the SyncSPKF- for $\tau < 0$. As already mentioned, the SyncSPKF is not defined except in the interval $0 \leq \tau \leq \Delta t$ but nevertheless it can be applied outside the interval just for comparison.

To get a better impression of the performance of the estimators, let us take a look on the separate intervals. Figure 2 shows $\sqrt{\sum_{i=1}^{N}(\hat{\tau}_{i,k} - \tau_i)^2/N}$, i.e. the root mean square of the estimation error at the time $k$ over the $N = 100$ estimations for $\tau < 0$. One can see that this error decreases rapidly over time (note the logarithmic scale of $k$). If the 'wrong' filters (SyncSPKF or SyncSPKF+) are used, an estimation bias remains while for the SyncSPKF-, the mean squared errors approaches 0.

The user of this filter is not mainly interested in the correct estimation of the latency. It is much more important to see whether the estimation of the original states improves. Table 1 provides the mean squared errors (for all filter runs $i$ and all time steps $k$) $\overline{e}_\diamond = \sqrt{\sum_{i=1}^{N}\sum_{k=1}^{5000}(\hat{\diamond}_{i,k} - \diamond_{i,k})^2/(5000N)}$ of the states and the latency time for all filter variants and for a standard

| | $\overline{e}_{x_1}/10^{-2}$ | $\overline{e}_{x_2}/10^{-2}$ | $\overline{e}_{\tau}/\Delta t$ |
|---|---|---|---|
| $\tau < 0$ | | | |
| KF | 3.8764 | 3.1292 | – |
| SyncSPKF | 1.5556 | 2.4207 | 0.2086 |
| SyncKF | 1.5852 | 2.3981 | 0.2135 |
| SyncSPKF+ | 1.4049 | 2.3085 | 0.1388 |
| SyncKF+ | 1.4168 | 2.2699 | 0.1424 |
| SyncSPKF- | 0.4753 | 1.6776 | 0.0246 |
| SyncKF- | 0.4764 | 1.6725 | 0.0251 |
| $\tau > 0$ | | | |
| KF | 3.8662 | 3.2287 | – |
| SyncSPKF | 0.3003 | 1.4119 | 0.0183 |
| SyncKF | 0.3120 | 1.4180 | 0.0210 |
| SyncSPKF+ | 0.3722 | 1.5514 | 0.0254 |
| SyncKF+ | 0.3705 | 1.5597 | 0.0252 |
| SyncSPKF- | 1.2668 | 2.2949 | 0.1423 |
| SyncKF- | 1.2742 | 2.2729 | 0.1447 |

Table 1. Mean squared errors of the states and the latency estimation.

Kalman filter without latency estimation. One can see clearly that all the latency estimators' performances are far better than that of the standard KF and that the position estimation improves by up to the factor 10 when the correct filter is used.

The very best results are obtained when the SyncKF or the SyncSPKF are used and $0 \leq \tau \leq \Delta t$. This is due to the fact that here, an exact model of the latency is used while all other variants use approximations. However, on can see that in this interval, also the SyncKF+ and the SyncSPKF+ perform very good. These two variants have also the advantage that their estimation errors for $\tau < 0$ are better than those of the SyncKF/SyncSPKF because the latter are not defined on this interval. As predicted, the '-' variants perform better for negative $\tau$.

The difference between the KF and the SPKF variants is not very large and the results don't allow to draw clear conclusions here. As mentioned above, the SPKF variants are expected to perform better on models with stronger nonlinearities (the original system here is linear, only the latency estimation introduces nonlinearities).

These results have been confirmed with different choices of $c$, $m$ and $d$ as well as different noise levels. The performance gain varies but the research has shown that the use of the latency estimators is always worth an examiation if exact synchronization of input and output data cannot be guaranteed.

## 6 Conclusions

In this paper, we have dealed with the problem of state estimation when using two sets of data that are not perfectly synchronized. It has been shown how an extended Kalman filter (or, alternatively, a so-called sigma-point Kalman filter) can be modified to estimate both the states of the original system and the latency between the input and output data of the system. Using these filters, the estimation accuracy can be improved considerably because the inherent modeling error (neglecting the latency) is removed. Filters have been developed for positive and negative latencies. While both filters can be also applied when the latency has a different sign, their performance decreases (however, it is still better than that of an estimator neglecting the latency).

Current research focuses on applying the filters to more complex, nonlinear system models like the calibration filter for inertial measurement systems [3, 12]. In this system, perfect synchronization is very difficult to obtain because the inertial measurement system provides the input data (acceleration and rate) while an industrial robot which has no real-time data interface provides position information.

## References


[1] M. S. Grewal and A. P. Andrews, *Kalman Filtering: Theory and Practice Using MATLAB*: John Wiley & Sons, New York, Chichester, 2001

[2] R. E. Kalman. A new approach to linear filtering and prediction problems, *Transactions of the ASME-Journal of Basic Engineering*: 82(Series D):35–45, 1960

[3] J. Fox and H. Janocha, A rare-update sigma-point Kalman filter as parameter estimator: In *Proceedings of the IASTED International Conference on Modelling, Identification, and Control (MIC'05)*, pages 190–195, Innsbruck, Austria, February 2005

[4] D. H. Titterton and J. L. Weston, *Strapdown inertial navigation technology*: IEE Radar, Sonar, Navigation and Avionics Series 5. Peter Peregrinus Ltd., London, UK, 1997

[5] E. v. Hinüber and H. Janocha, Inertial measurement system for calibration and control of robots: *Industrial Robot*, 20(2):20–27, 1993

[6] J. Elson and K. Römer, Wireless sensor networks: A new regime for time synchronization: *ACM SIGCOMM Computer Communications Review*, 33(1):149–154, January 2003

[7] N. Kaempchen and K. Dietmayer, Data synchroniziation strategies for multi-sensor fusion: In *Proc. of the 10th World Congress of Intelligent Transportation Systems (ITS 2003)*, Madrid, Spain, November 2003

[8] B. Li, A cost effective synchronization system for multisensor integration: In *Proc. of the 17th International Technical Meeting of the Satellite Division of the U.S. Institute of Navigation*, pages 1627–1635, Long Beach, CA, September 2004

[9] S. J. Julier and J. K. Uhlmann, Unscented filtering and nonlinear estimation: *Proceedings of the IEEE*, 92(3):401–421, March 2004

[10] M. Nørgaard, N. K. Poulsen, and O. Ravn, New developments in state estimation for nonlinear systems: *Automatica*, 36:1627–1638, 2000

[11] R. v. d. Merwe, *Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models*: PhD thesis, Oregon Health & Science University, Portland, USA, 2004 *http://www.cslu.ogi.edu/publications/*

[12] J. Fox, Forward and Inverse Stochastic Filtering for Inertial Sensor Calibration: In *Proceedings of the IASTED International Conference on Modelling, Identification, and Control (MIC'06)*, Lanzarote, Spain, February 2006