Multisensor Out of Sequence Data Fusion for Estimating the State of Discrete Control Systems

Eva Besada-Portas, Jose A. Lopez-Orozco, Member, IEEE,
Juan A. Besada, and Jesus M. de la Cruz, Member, IEEE

Abstract—The fusion center of a complex control system estimates the state of the system, $x_t$, the measurement of sensor $s$ at time $t$, $w_t$, the control signal for that sampling period, $F_{t-1}$, and $R_t$, the transition and measurement matrices, and $v_{t-1}$ and $w_{t-1}$, Gaussian variables with zero mean and covariances $Q_{t-1}$ and $R_{t-1}$. The most difficult scenario for the fusion center occurs when the delays and the sequence of arrival of the data are not fixed, constituting the named Out-Of-Sequence Problem (OOSP) [1].

For sequential fusion methods such as the KF [2], the OOSP can be exactly tackled discarding the data or buffering all the information relevant to the current state before estimating it. However, those options are not appropriate for networked real-time control systems with many OOS measurements, because they increase the uncertainty of the state and can affect the stability of the feedback loops [3], [4]. Another option consists in (1) storing the estimates of the state, control signals, and sensor data, (2) rolling back to the time stamp associated with the new measurement, and (3) restarting the fusion process from that point. This lets the fusion center obtain the same estimates as if it had received the data without delays, increasing its memory and computational needs. To reduce these needs, several OOS estimators have been recently developed, such as [1] and [5]–[12] for the KF, and [14]–[16] for Particle Filters (PF).

We present two memory and computational efficient centralized algorithms to estimate the state of MIMO discrete control systems with additive non-correlated Gaussian noise in the transition and measurement models with all the already received data with random delays. The first, designed for linear systems and originally proposed in [17], [18], finds the same solutions as the KF or Information Filter (IF) when the measurements are not delayed. It is computationally more efficient, simple to implement and general in scope than [5]–[12]. The second, for nonlinear systems, is a novel approach inside the EKF and Extended IF (EIF), significantly different to the OOS PF [14]–[16] or the use of linear OOS algorithms for linearized systems [19]. It can either find exactly the same solution as the EKF with non delayed measurements or a quicker approximation.

II. ESTIMATING THE STATE OF A MIMO LINEAR CONTROL SYSTEM WITH OOS MEASUREMENTS

A. IFAsyn: An Algorithm for the OOSP Based on the IF

The behavior of a MIMO linear discrete control system with additive white noise and $S$ sensors is modeled by (1), where $x_t$ is the state of the system, $z_{t,s}$ the measurement of sensor $s$ at time $t$, $w_{t,s-1}$ the control signal for that sampling period, $F_{t-1}$, and $R_s$, the transition and measurement matrices, and $v_{t-1}$ and $w_{t-1}$, Gaussian variables with zero mean and covariances $Q_{t-1}$ and $R_{t-1}$.

\[
\begin{align*}
    x_t & = F_{t-1} x_{t-1} + w_{t-1}, \\
    z_{t,s} & = H_{s,t} x_t + v_{t-1},
\end{align*}
\]

Our objective is to estimate the current system state and covariance $(\hat{x}_t, P_t)$ given their original values $(x_0, P_0)$, the model parameters and control signals $\{F, q_{t-1}, Q_{t-1}, H_s, R_s, w_{t-1}\}$, and all the data $\{z_{t,s}, s \in 1:S, k < 1\}$ measured at time $t$, $\alpha < t$.

For assimilating delayed measurements $(z_{t,k} \in k > t)$, we propose IFAsyn (IF for Asynchronous measurements) that obtains the same results as the EIF or IF with the same data without delays. It combines the KF and IF because the KF prediction, carried out in the working spaces $\{x_t, P_t\}$, is equivalent to other optimal solutions but more general, efficient and easy to implement. EIFAsyn, the nonlinear one, is a new solution of the OOS problem in the Extended KF (EKF) framework.
IFAsyn stores for each step time \( j \) the set \( \{ \hat{y}_{j|j-1}, Y_{j|j-1}, \} \) for the information state with all the data previous to \( j \), \((i, I, I)\) for accumulating the sensorial information of instant \( j \) already gathered and projected into the information space, and \((u, j_{t-1})\) for the control signal applied in each time step. For the current time, it also stores \((\hat{x}_{i|j}, \hat{P}_{i|j})\). To reduce the memory needs, IFAsyn only stores the last \( W \) group of variables, which lets it only assimilate the measurements that arrive with a maximum delay \( M_{a} = W - 1 \).

The prediction step (4) is the KF prediction (2) plus the initialization (5) of \((i, I, I, \hat{y}_{j|j-1}, Y_{j|j-1})\). The assimilation step (5) consists of two different parts. The first (6) projects the new measurement into the information space \((i, i, I, I)\) and adds it to \((i, I, I)\). The second is a loop that restarts the assimilation with the already projected sensorial information posterior to \( k \) and \( (L) \) are the projections between the KF and IF spaces.

\[
\text{(4)} \quad \hat{x}_{i|j-1} = F_{i|j-1} \hat{x}_{i|j-1} + u_{i|j-1} + \hat{w}_{i|j-1} \\
\hat{P}_{i|j-1} = F_{i|j-1} \hat{P}_{i|j-1} F_{i|j-1}^{T} + Q_{i|j-1} \quad \left( \text{P} \right) \\
\left\{ i_{0} = 0, \quad I_{1} = 0 \right\} \\
\text{(5)} \quad \hat{y}_{i|j} = H_{i}^{T} R_{i|j}^{-1} \hat{x}_{i|j} \quad \left( 1 \right) \\
i_{t-1} = i_{t} + 1, \quad I_{t} = I_{t-1} + I_{t,k} \\
\{ \hat{y}_{j|j-1} = Y_{j|j-1} = \hat{y}_{j|j-1} + I_{j} \} \quad \left( A \right) \\
\{ \hat{x}_{j|j} = Y_{j|j} = Y_{j|j-1} + I_{j} \} \quad \left( L \right) \\
\hat{P}_{j|j} = F_{j|j} \hat{P}_{j|j} F_{j|j}^{T} + Q_{j|j} \quad \left( P \right) \\
\left\{ \hat{y}_{j+1|j} = \hat{P}_{j+1|j} \hat{x}_{j+1|j} + \hat{w}_{j+1|j} \right\} + Q_{j+1|j} \quad \left( L \right) \\
\text{end} \\
\]

### B. IFAsyn Versus Other Algorithms for the OOSP

This section compares IFAsyn with other OOS algorithms for solving (1) whose optimality holds when the delays are bigger than one time lag \( (\xi_{t}, \alpha > t + 1) \). We skip [12], optimal for any delay too, because it restarts the fusion process from the time stamp of the incoming \( \xi_{t} \). We also consider [13]-Al, although its degree of suboptimality is a few percent of MSE and is less general because \((F_{k}, Q_{k}, k < 1)\) has to be available for arbitrary \( k \) and \( l \).

Table I compares the algorithms (column 1) according to their optimality (column 2), the algorithm they are based on (column 3) and their supporting assumptions (columns 4, 5 and 6). Those that invert \( F \) are only valid for discrete systems with non-singular \( F \) (property that holds for all discretized continuous systems). Many doesn’t consider \( u, t-1 \), although its inclusion is straightforward in [5], [10], [11] and more difficult in [7], [8], [13]. Not all of them are designed for multi-sensor systems; extending [5] is direct [7], [8], [10] and [11] need considering if there is already a measurement with the same timestamp as the new one, and [13] could be used without changes because it is already an approximated algorithm. Finally, the simplicity of IFAsyn, only exceeded by [5], and its capacity to deal with multiple sensors comes from its IF support. [9] also shares those properties, but it is less general because it inverts \( F_{t}, r-1 \).

To make a fair memory and computational comparison of the algorithms of Table I, we have modified [5]-[11] and [13] to include \( u, t-1 \) and use multiple sensors. Besides, all of them work with the same basic time step and with a window with \( W \) time steps. For [13] we also assume that arbitrary \((F_{k}, k, Q_{k}, k < 1)\) are available.

Table II shows their memory needs (column 2), when \( n_{s} \) and \( u_{t-1} \) have \( n_{t} \) elements. It also compares them (column 3) when \( W > 2 \) (for delays bigger than one time lag). IFAsyn needs more memory than [9]-[11] and [13] because IFAsyn stores \((\hat{y}_{j|j-1}, Y_{j|j-1}, i_{j}, I_{j})\) while [9] stores \((\hat{y}_{j|j}, Y_{j|j})\), and [10], [11] and [13] store \((\hat{x}_{j|j}, \hat{P}_{j|j})\). However, the extra memory of IFAsyn makes it more accurate.

To compare the computational cost of each algorithm, we have computed analytically the number of Floating Point Operations (FLOPs) of a prediction step and assimilation step of a \( \xi_{t} \), delayed \( r \) time steps \((r = t - k) \). IFAsyn, [9]-[11] and [13] have a linear dependency on \( r \), while [5]-[8] depend on \( W^{2} \) \([5] \approx W^{3}, [6], [7], [8] \approx W^{4}\). The asymptotic behavior of their number of FLOPs is shown in Figs. 1, 2 and 3 when we increment the \( r \) at \( W = 1 \), and the maximum delay \( r = W - 1 \). As the changing parameter grows, the most efficient algorithms are IFAsyn, [6], [11] and [13]. [13], which is approximated and not valid for any discrete system, is worst than the others for small delays and better when they grow (Figs. 1 and 2). IFAsyn is always better than [11]. Although [6] can outperform IFAsyn (Fig. 3), its number of FLOPs depends on \( W \) instead of \( r \) (Fig. 1), and it degrades quicker than IFAsyn as the maximum permitted delays grows (Fig. 2). So, for systems which can have random big delays, we consider IFAsyn computational better than [6]. Additionally, IFAsyn is easier to implement.
The memory and computation comparison show that the best optimal algorithms are [11] and IFAsyn. [11] minimizes the memory while only incrementing the computational cost of IFAsyn and [6]. IFAsyn usually minimizes the computation needs of the rest at the expenses of almost doubling the memory of [11]. That is, IFAsyn extra memory is paying for a reduction of the computational needs.

Finally, an analysis of the stability of a control system that uses IFAsyn (or [5], [7]–[11]) is out of the scope of this paper, but its equivalency with [6] makes it share its stability properties. Although they use the OOS data, the magnitude of the delays is important because while the data is not available the control system is working with intermittent observations what can instantaneously affect its stability [3], [4]. The problem is not posed by IFAsyn and [5]–[11], as they update the state as quickly as its computational requirements let them do it, but it has to be considered when they are used inside real-time control loops.

III. ESTIMATING THE STATE OF A MIMO NON-LINEAR CONTROL SYSTEM WITH OOS MEASUREMENTS

A. EIFAsyn: An Algorithm for OOSP Based on the EIF

The behavior of a MIMO nonlinear discrete control system with additive white noise and $S$ sensors is modeled with (6), where $(x_t, z_{s,t}, u_{t-1}, v_{t-1}, Q_{t-1}, R_{s,t})$ have the same meaning as in Section II-A, and $f(\cdot)$ and $h_{s}(\cdot)$ are the transition and measurements functions. They can be linear too, and so the general problem can be completely linear, completely nonlinear, or a mixture of both

\[ x_t = f(x_{t-1}, u_{t-1}, t, t-1) + v_{t-1} \]
\[ z_{s,t} = h_{s}(x_t, t) + v_{s,t} \quad \text{with } s = 1 : S. \]  

(6)

The objective of the fusion center is the same as in the linear case, but with the new models $f(\cdot)$ and $h_{s}(\cdot)$. When the data arrives without delays ($x_{t-1}, z_{s,t}, u_{t-1}, v_{t-1}$) can be approximately obtained by the prediction and assimilation steps of EKF and EIF [2]. The KF prediction and IF assimilation are presented in (7) and (8), where $(J_{g})_{g}(\cdot)$ is the Jacobian of function $g(\cdot)$

\[ F_{t-1,t} = (J_{f})_{g}(\xi_{t-1,t}, u_{t-1}, t, t-1) \]
\[ x_{t-1,t} = f(x_{t-1,t-1}, u_{t-1}, t, t-1) \]
\[ P_{t-1,t} = F_{t-1,t}P_{t-1,t-1}F_{t-1,t}^T + Q_{t-1,t-1} \]  

(7)

\[ H_{s,t} = (J_{h_{s}})_{g}(\xi_{t-1,t}) \]
\[ z_{s,t} = H_{s,t}x_{t-1,t} + v_{s,t} \]
\[ i_{k,t} = H_{s,t}^T R_{s,t}^{-1} e_{s,t} \]
\[ I_{k,t} = H_{s,t}^T R_{s,t}^{-1} H_{s,t} \]
\[ i_{k} = \sum_{s=1}^{S} i_{k,t} \]
\[ [\tilde{y}_{t,k} = y_{t,k} - i_{k} + I_{t,k} X_{k,t} = Y_{k,t} \]  

(8)

To assimilate delayed measurements ($x_{t-1,k}, a > k$) we propose EIFAsyn, an adaptation of IFAsyn which is based on the differences which exist between the expressions (2) and (7), and (3) and (8).

The only significant change of the EKF prediction (2) is a new operation to obtain $F_{t-1,t-1}$. Besides, the predictions ($P$) that appear in IFAsyn ([4] and [5]) use the whole KF prediction (2). So, substituting in IFAsyn the KF prediction by the EKF
prediction makes EIFAsyn suitable for general systems with nonlinear transition models and linear measurements.

The discrepancies of the assimilation steps of IF (3) and EIF (8) are more important. The EIF calculates \( \mathbf{H}_{s,t} \) and projects a corrected measurement \( \mathbf{H}_{s,t} \), that considers the discrepancy between the linearized measurement model \( \mathbf{H}_{s,t} \) and the nonlinear one \( \mathbf{h}_s(x_{t-1}, t) \), into the information space. So, \( i_{a,s}, I_{a,s} \) depend only on \( x_{t-1} \). However, the IFA syn assimilation (5) projects and accumulates the measurements outside its loop because those operations don’t depend on \( x_{t-1} \). So, if for EIFAsyn we maintain them outside the loop, including previously the calculi of \( \mathbf{H}_{s,t} \), and \( \mathbf{C}_{s,t} \), we won’t obtain the same values as the EIF when the measurements are not delayed. To make EIFAsyn behave as EIF, the whole EIF assimilation has to be included inside the loop, losing the memory and computational benefits of EIFAsyn. However, the EKF and EIF are already approximated solutions of the nonlinear problem. So, EIFAsyn assimilation supports both options. The first approximates the estimates reducing the computational and memory cost related with some sensors. The second recalculates the projected measurements for the sensors where \( \mathbf{H}_{s,t} \) changes significantly when \( x_{t-1} \) is updated.

The EIFAsyn prediction and assimilation steps are presented in (9) and (10). The variable \( \text{rec} \) defines the set of nonlinear sensors whose values need to be recalculated. Obviously, the linear sensors don’t belong to \( \text{rec} \). Besides, EIFAsyn stores the IFA syn variables \( \{ \mathbf{h}_s, y_{i,t-1}, Y_{i,t}, \mathbf{P}_{i,t-1} \} \) and all received \( \mathbf{C}_{i,s,t} \), of the sensors belonging to \( \text{rec} \) (i.e., \( I_{a,s} \)). IFA syn uses to accumulate the projected information of the sensors not belonging to \( \text{rec} \), while \( \mathbf{C}_{s,t} \) are used for the remaining ones. However, \( \mathbf{C}_{s,t} \) are optional. They only make EIFAsyn avoid to recalculate the projected measurement for the timestamp \( k \) of the \( \mathbf{C}_{s,t} \) which triggered the assimilation step. For the linear sensors EIFAsyn doesn’t calculate \( \mathbf{J}_{h_{s,t}} \) or \( \mathbf{C}_{s,t} \), and when the transition model is linear \( \mathbf{J}_{f_{s,t}} \) is not used.

If all the sensors belong to \( \text{rec} \), EIFAsyn has to recalculate everything from scratch from the time stamp of each new measurement. Therefore, the memory and computational benefits of EIFAsyn depend on the quantity of sensors not belonging to \( \text{rec} \). So, they should be carefully determined, considering which nonlinear sensors will be affected by older measurements and how small changes in \( x_{t-1} \) affect \( \mathbf{C}_{s,t} \) and \( \mathbf{H}_{s,t} \). This analysis is extremely important, because when a sensor which must belong to \( \text{rec} \) doesn’t, EIFAsyn can return erroneous estimates (see example in Section III-B).

Finally, EIFAsyn is a suboptimal solution to the OOSP due to its EKF and EIF origin, and in the case of not recalculating the measurements of all the nonlinear sensors, it doesn’t obtain the same solutions as the EKF with non-delayed measurements:

\[
\begin{align*}
F_{t+1, t} &= (J_{f_{s,t}})(x_{t-1}, u_{t-1}, t, t-1) \\
X_{t+1, t} &= f(x_{t-1}, u_{t-1}, t, t-1) \\
P_{t+1, t} &= F_{t+1, t} \cdot P_{t-1, t} + Q_{t-1, t} \\
\{ \hat{y}_{t+1} = P_{t+1, t-1} \cdot x_{t-1}, Y_{t+1} = P_{t+1, t-1} \} \\
\{ i_{t=0}, I_{t=0}, \hat{I}_{t=0}, \hat{I}_{t=0} \}
\end{align*}
\]

(E)

\[
\begin{align*}
\{ \text{end} \}
\end{align*}
\]

(B) Example

EIFAsyn is tested with the system of Table III to show the importance of selecting which sensors belong to \( \text{rec} \). The system has a non-linear transition (T) model, and one linear (S1) and two non-linear (S2, S3) sensors. All the sensors measure every 0.1 second and the measurements of S2 arrive delayed to the fusion center 10 time steps. Fig. 4 shows an example where the non-linearities make necessary to re-calculate the information of S3. The real state of the system \( \{ \text{Real} \} \) and the state estimated by EIFAsyn re-projecting S3 measurements \( \{ \text{rec} = [S3] \} \) coincide, while the state estimated without the re-projection \( \{ \text{rec} = [] \} \) differs. However, other tests with the same system
don’t show those differences. So, the properties of each system must be carefully analyzed, paying special attention to how \((\mathbf{H}_{j+1j}^{x+1} - h_i(\mathbf{z}_{j+1j}; j + 1))\) is affected by \(\mathbf{z}_{j+1j}\).

C. EIFAsyn versus Other Algorithms for the OOSP

The extension of EIFAsyn for the nonlinear transition model is direct, and without a substantial computational overload, it lets EIFAsyn obtain the same solution as the EKF for systems with non-linear transitions and linear measurements. However, many linear algorithms can’t be extended for that case: [5], [6] depend on the linearity of the transition which won’t hold true. and [7], [8] pre-calculate variables which won’t be useful due to the non-linearities.

In contrast, the linear algorithms are sometimes used in systems with linearized non-linear models [19]. This can be done when \((\mathbf{H}_{j+1j}^{x+1} - h_i(\mathbf{z}_{j+1j}; j + 1))\) doesn’t change significantly, but in other case, the nonlinear measurements need to be re-projected.

The EIF support of EIFAsyn make it not directly comparable with the OOSPF [14]–[16].

Finally, the stability problem of a control system that includes EIFAsyn for estimating the state will be caused by the EIF properties and the magnitude of the delays.

IV. CONCLUSION

We have presented two new algorithms to estimate the state of dynamic control systems for the OOSP. EIFAsyn is optimal for linear systems and avoids to recalculate the information of the already received measurements, although it re-propagates it. It is computational better rect, and without a substantial computational overload, it lets EIFAsyn obtain the same solution as the EKF for systems with non-linear transitions and linear measurements. However, many linear algorithms can’t be extended for that case: [5], [6] depend on the linearity of the transition which won’t hold true. and [7], [8] pre-calculate variables which won’t be useful due to the non-linearities.

In contrast, the linear algorithms are sometimes used in systems with linearized non-linear models [19]. This can be done when \((\mathbf{H}_{j+1j}^{x+1} - h_i(\mathbf{z}_{j+1j}; j + 1))\) doesn’t change significantly, but in other case, the nonlinear measurements need to be re-projected.

The EIF support of EIFAsyn make it not directly comparable with the OOSPF [14]–[16].

Finally, the stability problem of a control system that includes EIFAsyn for estimating the state will be caused by the EIF properties and the magnitude of the delays.

REFERENCES


