Embedding Kalman Techniques in the One-Shot Task Model when Non-uniform Samples Are Corrupted by Noise.

Conference Paper · July 2010

DOI: 10.1109/MED.2010.5547787 · Source: IEEE Xplore

Camilo Lozoya
Tecnológico de Monterrey
40 PUBLICATIONS **208** CITATIONS
SEE PROFILE

Julio Ariel Romero Pérez
Universitat Jaume I
54 PUBLICATIONS **218** CITATIONS
SEE PROFILE

Pau Martí
Universitat Politècnica de Catalunya
123 PUBLICATIONS **1,587** CITATIONS
SEE PROFILE

Manel Velasco
Universitat Politècnica de Catalunya
92 PUBLICATIONS **991** CITATIONS
SEE PROFILE

18th Mediterranean Conference on Control & Automation
Congress Palace Hotel, Marrakech, Morocco
June 23-25, 2010

Embedding Kalman Techniques in the One-Shot Task Model when Non-uniform Samples Are Corrupted by Noise.

Camilo Lozoya, Julio Romero, Pau Martí, Manel Velasco and Josep M. Fuertes

Abstract— **The performance of several closed-loop systems whose controllers concurrently execute in a multitasking real-time system may be deteriorated due to timing uncertainties in tasksézecutions, problem known as scheduling jitters. Recently, the one-shot task model, that combines irregular sampling, a predictor observer, and strictly periodic actuation, was presented in order to remove the negative effects of jitters. However, its successful application required noise-free samples.**

In this paper we extend the one-shot task model to the case of noisy measurements. In particular, we embed a Kalman filter into the model taking into account that the available measurements are not periodic. This poses the problem of adapting the standard discrete-time Kalman filter to the case under study, and decide when to apply the prediction and the correction phase. Two different strategies are presented, and their control performance and computation demand are analyzed through real experiments.

I. INTRODUCTION

Computer-controlled control systems are often implemented in small microprocessors enabled with real-time technology. In this scenario, the standard approach for real-time control systems considers the implementation of each control algorithm as hard real-time periodic task [1], where sampling and actuation occurs at the beginning and end of each job execution. If only one task is executed, the strict sampling and actuation periodicity mandated by discrete-time control theory [2] is accomplished.

However the implementation of multiple control tasks in a single microprocessor generates job executions prone to violate the periodic control demands due to the introduced timing uncertainties in jobs execution times, phenomena known as scheduling jitters. It has been shown that scheduling jitters deteriorate control performance [1].

To overcome this limitation several solutions can be found in the literature. In particular, a novel control task model, named "one-shot" task model, was presented with the objective of removing the degrading effects that jitters have in control performance [3]. It is built upon control theoretical results that indicate that standard linear discrete time control laws can be implemented considering only periodic actuation. Hence samples are not required to be periodic, and control signals are computed by applying a state feedback control law to the predicted state at the actuation time.

This work was partially supported by C3DE Spanish CICYT DPI2007-61527 and by ArtistDesign NoE IST-2008-214373.

C. Lozoya, P. Martí, M. Velasco and J.M. Fuertes are with the Automatic Control Department, Technical University of Catalonia, Pau Gargallo 5, 08028 Barcelona, Spain {camilo.lozoya, pau.marti, manel.velasco, josep.m.fuertes}@upc.edu

J. Romero is with the Department of Industrial Systems Engineering and Design, University of Jaume I, Castelló, Spain romeroj@esid.uji.es

From the scheduling point of view, the new task model can be seamlessly integrated into existing real-time scheduling theory and practice. However, its operation relies on predictions computed from each sample. If samples are corrupted by noise, its operation quickly deteriorates.

In this paper we extend the one-shot task model to the case of noisy measurements. It is well known that systems are noise corrupted and that sensors in a control loop do not provide exact readings of desired quantities [4]. In these cases, filtering is desirable since it removes the noise from signals while retaining the valuable information. The Kalman filter [5] has been proved to be a useful tool for inferring the missing information from indirect and noisy measurements.

The contribution of this paper is to embed a Kalman filter into the one-shot task model. The standard approach for the implementation of a discrete-time Kalman filter assumes strict periodic sampling and actuation. However, in the one-shot task model, the available measurements are not periodic. This poses the problem of adapting the standard Kalman filter to the case of irregular sampling, and decide when to apply the prediction and the correction phase. For this case, an asynchronous Kalman filter is required. Two different strategies are presented, and their control performance and computation demand are analyzed through real experiments.

The application of Kalman techniques for systems with diverse type of non-periodic sampling can be found in the literature, such as for multirate control systems, e.g. [6], [7], or event-based control systems, e.g. [8], [9]. However, non of them applies to the problem tackled in this paper.

The rest of this paper is structured as follows. Section II introduces the theoretical aspects of the one-shot task model and the Kalman filter. Section III presents the novel strategies for implementing the Kalman filter in the one-shot task model. Sections IV and V describe the experimental setup and results, respectively. Section VI concludes the paper.

II. PRELIMINARIES

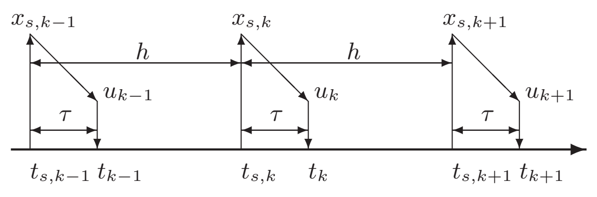
II-A. *Standard control task model*

Consider the state-space model of a linear time-invariant discrete-time system with sampling period h [2]

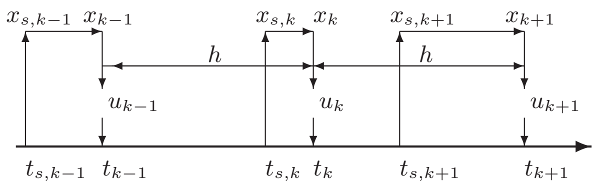
$$\begin{aligned} x_{k+1} &= \Phi(h)x_k + \Gamma(h)u_k \\ y_k &= Cx_k, \end{aligned} \tag{1}$$

where x_k is the plant state, u_k and y_k are the inputs and outputs of the plant, matrix $C \in \mathbb{R}^{p\times n}$ is the output matrix, and matrices $\Phi(t)$ and $\Gamma(t)$ are obtained using

$$\Phi(t) = e^{At}, \quad \Gamma(t) = \textstyle\int_0^t e^{As}Bds, \tag{2}$$



(a) Standard task model



(b) One-shot task model

Fig. 1. Control task models

with $t = h$, where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ are the system and input matrices of the continuous-time form

$$\begin{array}{rcl} \frac{dx(t)}{dt} & = & Ax(t) + Bu(t) \\ y(t) & = & Cx(t). \end{array} \quad (3)$$

For standard closed-loop operation of (1), the control signal u_k is given by

$$u_k = Lx_k \quad \text{with } L \in \mathbb{R}^{1\times n}, \quad (4)$$

where L is the state feedback gain obtained using standard control design methods from matrices $\Phi(h)$ and $\Gamma(h)$.

Model (1) can be augmented to cope with a time delay modelling an input/output latency that appears due to the computation of the control algorithm. The standard model that incorporates a time delay τ, with $\tau \leq h$, is [2]

$$x_{s,k+1} = \Phi(h)x_{s,k} + \Phi(h-\tau)\Gamma(\tau)u_{k-1} + \Gamma(h-\tau)u_k. \quad (5)$$

Model (5) has been often taken as the underlying standard control task model for design and analysis of real-time control systems. This model assumes a time reference given by the sampling instants with a fixed time delay from sampling to actuation (see Fig. 1-a). It mandates to periodically sample and actuate.

II-B. *One-shot task model for real-time control systems*

In a multitasking real-time control system, the timing demanded by the control model (5) is often violated due to the irregular sampling and actuation that scheduling jitters introduce. A solution to this problem, from the task model perspective, can consist on providing synchronization at actuation instants rather than at the sampling instants, as facilitated by the one-shot task model.

In the one-shot task model the time elapsed between consecutive actuation instants, named t_{k-1} and t_k, is periodic, and $h = t_k - t_{k-1}$ is defined as the actuation period. Within this time interval, the system state is sampled, named $x_{s,k} \in (t_{k-1}, t_k)$, and the sampling time recorded, $t_{s,k}$. The difference between this time and the next actuation time

$$\tau_k = t_k - t_{s,k} \quad (6)$$

is used to estimate the state at the actuation instant as

$$\hat{x}_k = \Phi(\tau_k)x_{s,k} + \Gamma(\tau_k)u_{k-1}. \quad (7)$$

Then, using the estimated state, the control signal is

$$u_k = L\hat{x}_k \quad \text{with } L \in \mathbb{R}^{1\times n} \quad (8)$$

where L is the original controller gain as in (4). The control signal u_k is held constant within actuation instants.

A control strategy using (6)-(8) relies on the time reference given by the actuation instants, if u_k is applied to the plant by hardware interrupts. In addition, samples are not required to be periodic because τ_k in (6) can vary at each closed-loop operation, as illustrated in Fig. 1-b. The interested reader is referred to [3] for further reading on this task model.

II-C. *Kalman filter for noisy signals*

The discrete-time Kalman filter addresses the general problem of trying to estimate the system state of a discrete-time controlled plant. Therefore, for the filter implementation we can enhance the model (1) by adding process and measurement noise (w_k and v_k respectively) as in

$$\begin{array}{rcl} x_{k+1} & = & \Phi x_k + \Gamma u_k + w_k \\ y_k & = & Cx_k + v_k. \end{array} \quad (9)$$

The algorithm for implementing the Kalman filter is divided in two phases: time update (predictor) and measurement update (corrector). The predictor phase uses the previous estimation to produce the *a priori* estimation of the system state (equations (10) and (11)). In the corrector phase, measurement information from the system output is used to refine the prediction and obtain the *a posteriori* estimation (equations (12), (13) and (14)). The *a posteriori* estimation is used in the next predictor phase.

In the predictor phase, if we consider that we want to estimate the next system state as in (9), then in the predictor phase, the *a priori* estimation of the system state is

$$\hat{x}^-_{(k+1)} = \Phi\hat{x}_{(k)} + \Gamma u_{(k)} \quad (10)$$

where Φ and Γ represent the system dynamics from (9), $\hat{x}_{(k)}$ defines the current *a posteriori* estimate of the process state, and $u_{(k)}$ represents the current input. The *a priori* estimation of the covariance error is

$$P^-_{(k+1)} = \Phi P_{(k)}\Phi^T + Q \quad (11)$$

where $P_{(k)}$ is the current *a posteriori* estimate of the covariance error, and Q is the constant covariance value of the process noise.

In the corrector phase, the next Kalman gain value

$$K_{(k+1)} = \frac{CP^-_{(k+1)}}{CP^-_{(k+1)}C^T + R} \quad (12)$$

is obtained prior to the calculation of the *a posteriori* estimation, where $K_{(k+1)}$ is the Kalman gain, C defines the constant measurement gain as in (9), and R is the covariance value of the measurement noise. Then, the *a posteriori* estimation of the next state is

$$\hat{x}_{(k+1)} = \hat{x}^-_{(k+1)} + K_{(k+1)}(y_{(k)} - C\hat{x}^-_{(k+1)}) \quad (13)$$

where $y_{(k)}$ is the measured output of the system as in (9). The *a posteriori* estimation of the covariance error is

$$P_{(k+1)} = (I - CK_{(k+1)})P^-_{(k+1)} \quad (14)$$

where I is the identity matrix.

III. DESIGN STRATEGY

The implementation of a discrete-time Kalman filter is straightforward if strictly periodic sampling is ensured. However, integrating a Kalman filter with the one-shot task model raises some problems that require a detailed analysis.

The Kalman filter algorithm has two phases which are prediction and correction. The correction must take place at the sampling instant, since we require a process measurement in order to execute the correction. However the one-shot task model makes the synchronization at the actuation instants and sampling is accepted to be non-periodic. Furthermore the one-shot task model uses a time difference (6) to estimate the state at the actuation instant (7), in addition to the estimations and predictions required by the Kalman filter algorithm.

By considering these aspects, two different approaches to embed a Kalman filter with the one-shot task model were identified.

The first approach implements the Kalman correction just from sampling to actuation instants, and the second approach considers the complete sampling interval to implement the Kalman algorithm. For the rest of this paper, we identify the first approach as the *half* Kalman filter and the second one as the *complete* Kalman filter.

III-A. Half Kalman filter

In this approach the Kalman filter is split into two parts. In the first one, from sampling ($t_{s,k}$) to actuation (t_k), only the predictor phase is used. In the second one, from actuation (t_k) to next sampling ($t_{s,k+1}$), the predictor and the corrector phases are executed, as illustrated in Fig. 2-a. It is important to highlight that, during the first part, the corrector phase cannot be used since process measurements values, used for corrections, are only available at sampling instants and not at actuation instants.

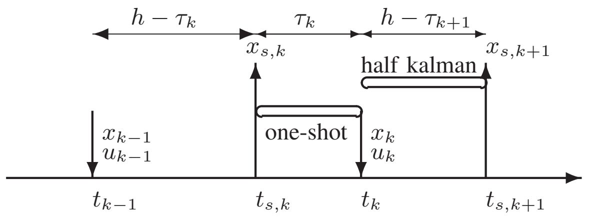
Hence, if only predictor applies from sampling ($t_{s,k}$) to actuation (t_k), equations (10) and (11) transform to

$$\hat{x}^-_k = \Phi(\tau_k)\hat{x}_{s,k} + \Gamma(\tau_k)u_{k-1} \quad (15)$$

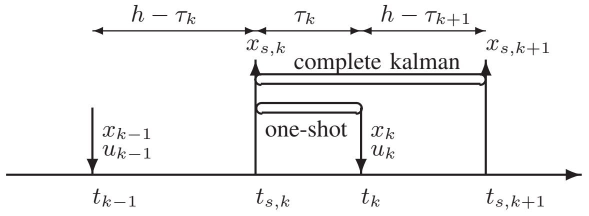
$$P^-_k = \Phi(\tau_k)P_{s,k}\Phi(\tau_k)^T + Q. \quad (16)$$

In the second part, from actuation (t_k) to next sampling ($t_{s,k+1}$), the Kalman predictor and corrector apply. First, the predictor from (10) and (11) is redefined as

$$\hat{x}^-_{s,k+1} = \Phi(h-\tau_{k+1})\hat{x}^-_k + \Gamma(h-\tau_{k+1})u_k \quad (17)$$



(a) Half Kalman filter



(b) Complete Kalman filter

Fig. 2. Kalman filter design approaches

$$P^-_{s,k+1} = \Phi(h-\tau_{k+1})P^-_k\Phi(h-\tau_{k+1})^T + Q, \quad (18)$$

and then from (12), (13) and (14), the corrector phase is formulated in this strategy as

$$K_{s,k+1} = \frac{CP^-_{s,k+1}}{(CP^-_{s,k+1}C^T + R)} \quad (19)$$

$$\hat{x}_{s,k+1} = \hat{x}^-_{s,k+1} + K_{s,k+1}(y_{s,k+1} - C\hat{x}^-_{s,k+1}) \quad (20)$$

$$P_{s,k+1} = (I - CK_{s,k+1})P^-_{s,k+1}, \quad (21)$$

Then, the one-shot task model (7) and (8) can be implemented. Notice that the estimation of the state at the actuation instant has been already obtained in (15). Hence the control signal is calculated by

$$u_k = L\hat{x}^-_k. \quad (22)$$

III-B. Complete Kalman filter

This approach uses a Kalman filter to predict and correct from current sampling ($t_{s,k}$) to next sampling ($t_{s,k+1}$). In addition, the one-shot task model requires an estimation from sampling ($t_{s,k}$) to actuation (t_k), as illustrated in Fig. 2-b.

If the complete sampling interval is considered, the Kalman *a priori* estimation can be obtained by substituting (15),(16) into (17),(18) respectively, then the following predictor phase equations are obtained

$$\begin{aligned}\hat{x}^-_{s,k+1} &= \Phi(h-\tau_{k+1}+\tau_k)\hat{x}_{s,k} \\ &\quad +\Phi(h-\tau_{k+1})\Gamma(\tau_k)u_{k-1} \\ &\quad +\Gamma(h-\tau_{k+1})u_k \quad (23)\\ P^-_{s,k+1} &= \Phi(h-\tau_{k+1}+\tau_k)P_{s,k}\Phi(h-\tau_{k+1}+\tau_k)^T \\ &\quad +\Phi(h-\tau_{k+1})Q\Phi(h-\tau_{k+1})^T + Q. \quad (24)\end{aligned}$$

Notice that u is not constant over the sampling inverval. Hence, eq. (23) considers u_{k-1} and u_k. Also, the sampling interval is not constant, and it varies according to $h-\tau_{k+1}+\tau_k$ at each closed-loop operation.

From (12), (13) and (14), the corrector phase is formulated in this strategy as

$$K_{s,k+1} = \frac{CP^-_{s,k+1}}{(CP^-_{s,k+1}C^T + R)} \quad (25)$$

$$\hat{x}_{s,k+1} = \hat{x}^-_{s,k+1} + K_{s,k+1}(y_{s,k+1} - C\hat{x}^-_{s,k+1}) \quad (26)$$

$$P_{s,k+1} = (I - CK_{s,k+1})P^-_{s,k+1}. \quad (27)$$

According to the one-shot task model, the control signal is calculated from the estimation of the state at the actuation instant, which is taken from the *a posteriori* state estimation at sampling instance. Therefore, equations (7) and (8) of the task model are redefined as

$$\hat{x}_k = \Phi(\tau_k)\hat{x}_{s,k} + \Gamma(\tau_k)u_{k-1} \quad (28)$$

$$u_k = L\hat{x}_k. \quad (29)$$

III-C. Discussion

At first sight both approaches are similar and it is expected that both will produce similar results. However, in a deeper analysis, there are some differences that may affect the computational demand of their implementation.

The implementation of the complete Kalman filter requires to calculate $\Phi(\cdot)$ as a function of three different time values, i.e., $\Phi(\tau_k)$, $\Phi(h-\tau_{k+1})$, $\Phi(h-\tau_{k+1}+\tau_k)$. Meanwhile, in the half approach only two $\Phi(\cdot)$ values are required, i.e., $\Phi(\tau_k)$, $\Phi(h-\tau_{k+1})$. This may increase the computational demand for the complete approach. On the other hand, the half Kalman filter requires to obtain $\hat{x}^-_k$ previous to $\hat{x}^-_{s,k+1}$, which may imply an additional operation. However $\hat{x}^-_k$ is required anyway by the one-shot model.

IV. EXPERIMENTAL SET-UP

An unstable plant in the form of a double integrator electronic circuit is controlled by a control task executing on the Erika real-time kernel [10]. The executing platform is the full Flex board [10] equipped with a dsPIC micro-processor (Fig. 3-bottom).

IV-A. Controlled plant

The electronic double integrator is illustrated in Fig. 3-top. In this circuit, according to a specific set-point, the PWM, acting as actuator, adjusts the duty cycle to provide the proper output voltage (V_1), which is read through the analog-to-digital converter. The controller objective is to have the circuit output voltage V_1 tracking random set point changes.

Considering the component values $R_{1/2} = 1\text{k}\Omega$, $R_1 = R_2 = 100\text{k}\Omega$, and $C_1 = C_2 = 420\text{nF}$, the following model for the plant can be obtained,

$$\begin{bmatrix} \dot{v}_1 \\ \dot{v}_2 \end{bmatrix} = \begin{bmatrix} 0 & -23{,}8 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -23{,}8 \end{bmatrix} u$$
$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}. \quad (30)$$

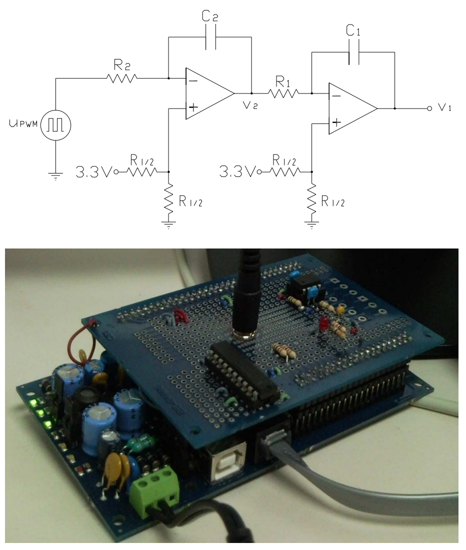


Fig. 3. Double integrator circuit scheme and implementation set-up

IV-B. Controller design

The controller gain L corresponds to the discrete Linear Quadratic Regulator for (30), which minimizes a discrete cost function equivalent to the continuous cost function

$$J = \int_0^\infty (x^T Q_{1c} x + u^T Q_{2c} u)dt \quad (31)$$

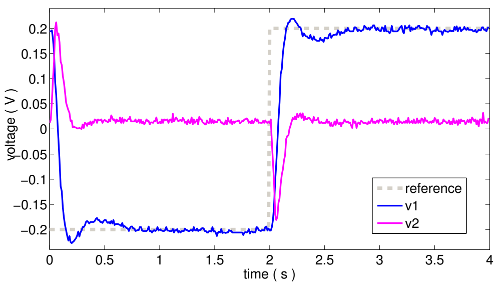
where the weighting matrices Q_{1c} and Q_{2c} are the identity. Considering a sampling period of $h = 50$ms, the optimal controller gain is $L = \begin{bmatrix} 0{,}3951 & -0{,}9728 \end{bmatrix}$.

The Kalman filter was designed taking into account the noise covariances $Q_n = E(w \cdot w^T) = 2 \cdot 10^{-7}$ and $R_n = E(v \cdot v^T) = 8 \cdot 10^{-5}$ extracted from the electronic circuit of the experimental setup, where w and v are the plant noise and the measurement noise, respectively. Off-line sample measurements data, using the dsPIC and considering sampling periods of $h = 50$ms, were taken in order to determine the measurement noise covariance. Plant noise covariance was calculated from data obtained from direct plant measurements with calibrated instruments. In both cases, the data obtained corroborate the presence of white noise.

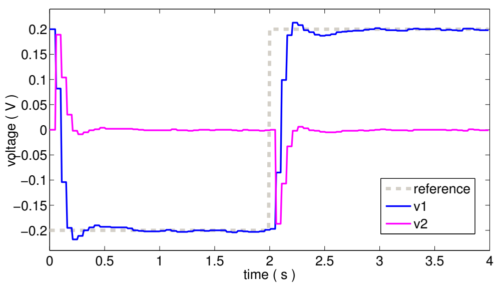
IV-C. Kalman algorithm implementation

The implementation of the half Kalman algorithm and the complete Kalman algorithm into the dsPIC processor, requires to calculate $\Phi(\cdot)$ and $\Gamma(\cdot)$ as function of different time values. These calculations represent the most time consuming operations for the processor. Two strategies were implemented.

The first one, called "generic function", allows computing Φ and any Γ for any time value. The second one, called "specific function", uses a look-up table to obtain the Φ and Γ for specific time values. The generic function spends more processor time, but with the advantage of accepting any time value. The specific function is faster, but it requires memory for storing the table.

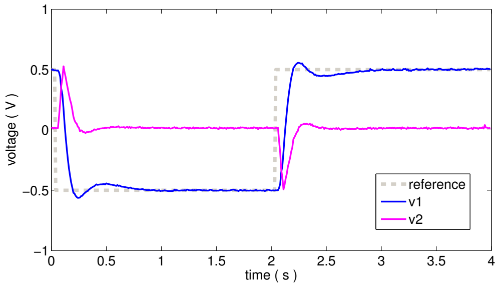


(a) Noisy signals from plant

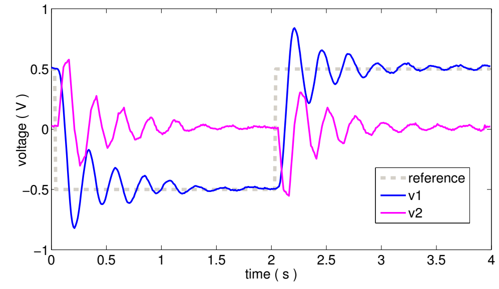


(b) Kalman estimated state values

Fig. 4. Removing noise with the Kalman filter



(a) One-shot controller



(b) Standard controller

Fig. 5. Controllers response with jitters

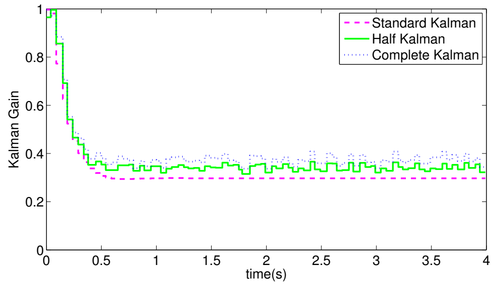


Fig. 6. Kalman gain evolution

V. EXPERIMENTAL RESULTS

The experimental results are divided in three groups. The first two experiments show that the Kalman filter and the one-shot task model preserve their benefits when both are integrated in a control loop. In the third group of experiments, the two different implementations of the Kalman filter (half and complete) with the one-shot task model are compared each other and with other Kalman filter implementations in terms of control performance. Finally, the resource demand of the half and complete Kalman implementation is also analyzed.

V-A. Kalman filter results

The objective of this experiment is to validate that the Kalman filter implementation integrated with the one-shot task model is able to effectively estimate the system states from a noisy signal. The half Kalman one-shot controller was used in this evaluation. Similar results are found with the complete Kalman. Fig. 4 compares the noisy captured data from the plant (top) with the estimated states obtained with the half Kalman filter (bottom). As expected, the Kalman filter effectively removes the noise from the signal. Note that reference changes use small values ($-0{,}2$volts to $0{,}2$volts) to appreciate the noisy signal.

V-B. One-shot controller results

For this experiment, the controller task has the presence of random timing variations in the form of scheduling jitters. Jitters produce irregular sampling periods ranging from 0 to 20ms. The system response of the half Kalman one-shot controller is compared with a standard controller (Fig. 5) in order to assess whether embedding the Kalman filter jeopardizes the benefits of the one-shot task model in removing the jitters effects. As it can be seen in the figure, the control performance of the standard controller is considerably degraded while the one-shot controller achieves the same performance than the case without jitters.

V-C. Kalman gain evolution

The Kalman gain values during the previous experiment (with jitters) were obtained in order to certificate the correct implementation of the Kalman filter. Fig. (6) shows the evolution of the first element of the Kalman gain for the half and complete approaches compared with the Kalman filter's gain using a standard controller with no jitters. It can be

TABLE I

EXPERIMENTAL CONTROL PERFORMANCE

Implementation Approach	Control Performance
(A) Kalman with standard controller (no jitters)	4.4517
(B) Half Kalman with one-shot controller	4.4523
(C) Complete Kalman with one-shot controller	4.4538
(D) Kalman with standard controller	5.0645
(E) No Kalman with standard controller	7.5359

noticed that the values are similar despite of small variations for the half and complete approaches.

V-D. *Control performance evaluation*

A total of five different implementation approaches, including the half and the complete Kalman filter, were evaluated in terms of control performance. For each approach, performance was measured with a discrete-time control cost equivalent to the continuous cost defined in (31).

A set of ten different experimental scenarios were elaborated in order to cover a wide variety of system conditions. Each scenario considers different jitters values, and different setpoints (reference) amplitudes and frequencies. The same set of scenarios was applied to each approach, with the exception of the first approach (A) where no jitters were applied, since this implementation approach serves as a reference (ideal case) for the experimental evaluation. Average values of the ten scenarios (smaller values means better performance) are presented on Table I.

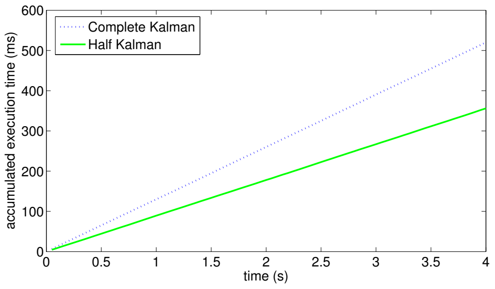
The results shows that the half Kalman (B) and the complete Kalman (C) implementations using the one-shot controller has no meaningful differences in their performance. And both approaches have practically the same performance as the ideal case (A), even when (B) and (C) includes jitters. The Kalman filter implementation in the standard controller (D) has a worse performance compared with (B) and (C) as expected because jitters affects its performance. Finally, it is interesting to notice that if a standard controller is used without Kalman (E), the jitters degrading effect is greater than the one obtained with the use of Kalman (D).

V-E. *Resource demand evaluation*

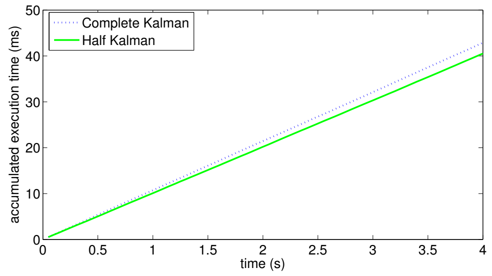
Fig. 7 shows the accumulated execution time during a period of 4 seconds for the half and complete Kalman implementations using the generic or the specific function. In both cases the half Kalman algorithm consume less execution time due less operations are required, as discussed in subsection III-C. Now, when the specific function is used the difference is reduced considerably but at the expenses of increasing the memory demand. In this example, considering a range of timing variations from 0 to 50ms, and a time granularity of 1ms, the memory required for storing the loop-up table is less than 1Kb.

VI. CONCLUSIONS

This paper has presented the integration of Kalman filter techniques with the one-shot task model. Experimental results over a noisy plant have demonstrated that their



(a) Using the generic function



(b) Using the look-up table (specific function)

Fig. 7. Accumulated execution time

integration preserves their own benefits: noise removal and jitters effects elimination. Two different Kalman implementation approaches have been presented with similar control performance results but slightly different resource demands. Future work will focus on extending this integration in the context of networked control systems.

REFERENCES

[1] K.-E. Årzén, A. Cervin, J. Eker and L. Sha, "An introduction to control and scheduling co-design," *39th IEEE Conference on Decision and Control,*, 2000.

[2] K.J. Åström and B. Wittenmark, *Computer-Controlled Systems*, third ed, Prentice-Hall, 1997.

[3] C. Lozoya, P. Martí, and M. Velasco, "The one-shot task model for robust real-time embedded control systems," *IEEE Transactions on Industrial Informatics,* 2008.

[4] P.S. Maybeck, *Stochastic models, estimation, and control*, Vol. 1, Academic Press, 1979.

[5] R.E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME - Journal of Basic Engineering*, Vol. 82: pp. 35-45, 1960.

[6] W. Lia, S.L. Shaha, and D. Xiao, "Kalman filters in non-uniformly sampled multirate systems: For FDI and beyond," *Automatica*, Vol. 44, Is. 1, pp. 199-208, Jan 2008

[7] R. Piza, J. Salt, A. Cuenca, V. Casanova, "Kalman filtering applied to Profibus-DP systems. Multirate control systems with delayed signals," *34th Annual Conf. of IEEE Industrial Electronics Society*, Nov. 2008

[8] A. Lem, R. McCann, "Event-Based Measurement Updating Kalman Filter in Network Control Systems," *2007 IEEE Region 5 Technical Conference*, April 2007.

[9] Y.S. Suh, V.H. Nguyena, and Y. S. Roa, "Modified Kalman filter for networked monitoring systems employing a send-on-delta, " *Automatica*, Vol. 43, Is. 2, pp. 332-338, Feb. 2007.

[10] Evidence Srl., `http://www.evidence.eu.com/`