A Tutorial on Implementing Kalman Filters with Commonly Used Blocks

Tiago Davi Curi Busarello
Depart. Control, Automation and Computation Engineering
Federal University of Santa Catarina - UFSC
Blumenau, Brazil
ORCID 0000-0002-5406-3811

Marcelo Godoy Simões
Electrical Engineering Department
Colorado School of Mines
Golden, USA
ORCID 0000-0003-4124-061X

***Abstract*—Kalman filters are a mature and widely used technology in the field of engineering. However, its implementation is sometimes not trivial and usually not well explained in scientific papers. This paper presents a tutorial on implementing Kalman filters with commonly used blocks, that are sum, product, unit delay and zero order holder. The main objective of this tutorial is to make a clear understanding of the Kalman filter algorithm. A brief review on averaging filter is first presented to support the understanding of the Kalman filters. Later, the Kalman filter algorithm is carefully described, step-by-step, and its block implementation is described. The simulation made by the blocks is then uploaded in a physical digital signal processor, TMS320F28335, through the Matlab/Simulink and simulation and experimental results show the efficacy of this tutorial. Therefore, this tutorial helps engineering designers who need a simple, fast, accurate and an easy-to-use implementation of Kalman filters.**

Keywords—Kalman, Digital Filters, Tutorial, Algorithm, error covariance, varying weight, prediction, estimation.

I. INTRODUCTION

Kalman filters are found in almost every field of engineering. Their applications can make a system to perform tasks that would be impossible with other kind of filters. The most common applications of Kalman filter is in scenarios where the input data varies dynamically and there is a need to estimate accurately the tendency of the input data. Design Thinking is an educational technique in scaffolding Problem Based Learning (PBL) experiences where students get involved in the project, and acquire several practical and interdisciplinary abilities. Kalman filters expand the mathematical and engineering perspectives of engineering students.

A variety of books describe Kalman filters [1]-[7]. They describe the theory behind the Kalman filter, how it was derived and features that require a strong background in math. These books have their legitimacy and efficacy. Due to the maturity of the Kalman filter, a clear understanding of its implementation algorithm may be sufficient to employ such filter in engineering applications. Phil Kim [8] describe in his books how to implement the Kalman filter algorithm without giving much attention to the theory. The book is easy to follow and it contains Matlab scripts with C- code to implement Kalman filters. However, the environment of the applications is only computational.

Scientific papers with engineering applications that employ Kalman filter hardly describe details, particularly because usually those papers focus on the engineering problem and solutions, and not really on the working cogs of the Kalman filter algorithm, making the assumption of a black-box or toolbox, which is possibly true only for the purpose of the scientific application. The objective of our paper is to clarify the implementation of a Kalman filter, using simple mathematical assumptions, a clear block diagram and allow engineering students and practicing professionals to apply Kalman filtering for many other uses.

Applications using Kalman filter, but not giving the desired attention to it, are constantly being presented in the literature [9]-[19].

In [9] the paper discusses a linear-quadratic-Gaussian/loop transfer recovery procedure using reduced-order Kalman filters by extending known exact-recovery result. The state-space realization commonly used for reduced-order observer design is employed. The zero-structure intrinsic to the realization is revealed. In [10] the authors investigate an adaptation of the high-gain Kalman filter for nonlinear continuous-discrete system with multirate sampled outputs under an observability normal form. The paper proved that the global exponential convergence of the observer through the existence of bounds for the Riccati matrix. It is also showed that, under certain conditions on the sampling procedure, the observer's asynchronous continuous-discrete Riccati equation is stable.

In [11] the paper proposes a noninvasive Permanent Magnet Temperature (PMT) estimation approach based on the Permanent Magnet Synchronous Machines steady-state equation and using Kalman filter. First, a linear temperature model, dependent solely on the PMT, is derived from the steady- state equation and the PMT model. Thus, the PMT can be directly estimated from the measurements using the derived linear model. In order to improve the estimation performance, a linear state-space model was developed based on the derived model and the Kalman filter was employed for PMT estimation.

In [12] the authors propose an implementation of generalized regression neural network as an alternative to this traditional received signal strength indicators-based approach, to obtain first location estimates of single target moving in 2-D in wireless sensor networks, which are then further refined using Kalman filtering framework.

There are also some researches that proposed modifications and improvements in the Kalman filter algorithm [20]-[23]. In [20] an inverse free Kalman filter is proposed. The inverse of innovation covariance matrix required in the update step of the Kalman filter is approximated using Taylor series expansion. The approximate inverse has a closed form expression in the elements of the original matrix. In [21] the authors studied the cascade of a global nonlinear observer with the linearized Kalman filter, where the estimate from the nonlinear observer is an exogenous signal only used for generating a linearized model to the Kalman filter. It was shown that the two-stage nonlinear estimator inherits the global stability property of the nonlinear

observer, and simulations indicate that local optimally properties similar to a perfectly linearized Kalman filter was achieved.

All the above researches use Kalman filters supposing that the readers have deep knowledge in this field. This is not true for the majority of readers. In this case, the reader needs to stop to study Kalman filter and return to the researches for a clear understand. However, this interruption usually takes a long time due to the complexity of the books in explaining Kalman filters. This even may discourage the reader to continue reading those researches. The literature lacks of a basic tutorial on implementing Kalman filters.

In this context, this paper presents a tutorial on implementing Kalman filters with commonly used blocks. The main purpose of proposing a tutorial like this is to allow readers clearly understand Kalman filters. Without much knowledge in optimization math and programming, a reader is able to understand and implement Kalman filters through this tutorial. A brief review on averaging filter is first presented to support the understanding of the Kalman filters. Later, the Kalman filter algorithm is carefully described, step-by-step, and its block implementation is explained. The simulation made by the blocks is then uploaded in a physical digital signal processor, TMS320F28335, through the Matlab/Simulink and simulation and experimental results show the efficacy of this tutorial. Therefore, this tutorial helps engineering designers who need a simple, fast, accurate and an easy-to-use implementation of Kalman filters.

II. Averaging Filters

This section presents a short background on averaging filter. Its understanding is necessary to later understand Kalman filters. There are three basic types of averaging filter, as described in the following subsections.

A. Average Filter

The average filter is the most basic structure of a filter which computes the average value of an input signal. The most common applications of average filter are in noise filtering. The average filter consists in computing the average value of a certain amount of input data. The discrete time-domain for the output of the average filter, written in a recursive way is given by:

$$\overline{x}_k = \frac{k-1}{k}\overline{x}_{k-1} + \frac{1}{k}x_k \tag{1}$$

where k is the amount of data in the input signal.

Three points must be highlighted in this kind of filter:

- The output is merely the average value of all k values available in the input signal;
- When a new value of k comes into the filter, the average is computed again taking into account the new amount of values;
- All samples have the same weight, which is equal to $1/k$

B. Moving Average Filter

The average filer described in the previous subsection is not sufficient for dynamic variations in the input signal. On the other hand, the moving average filter is able to track such variations. The moving average filter does not compute the average taking into account all k values of the input signal, but a limited and recent values. The discrete time-domain for the output of the moving average filter for a certain amount of n is given by:

$$\overline{x}_k = \overline{x}_{k-1} + \frac{x_k - x_{k-n}}{n} \tag{2}$$

where n is chosen according to design requirements.

A drawback of this filter is that all samples have the same weight, which is equal to $1/n$.

C. Exponential Average Filter

Having the same weight for all samples is not attractive because the filter may not track dynamic variation of the input signal in a satisfactory manner. The exponential average filter appears to overcome this inconvenience. The discrete time-domain for the output of the exponential average filter is given by:

$$\overline{x}_k = \alpha\overline{x}_{k-1} + (1-\alpha)x_k \tag{3}$$

where α is a constant $0 < \alpha < 1$ and it defines the weight of the samples. The value of α may be chosen randomly. Notice that depending on the value of α, the newest value of the input signal, in this case the x_k, can be higher or lower weight compared to the previous output signal x_{k-1}.

III. Kalman Filter Algorithm

Fig. 1 presents the Linear Kalman Filter Algorithm. The algorithm consists of performing five steps. The first step is just the definition of initial values. The second step is the computation of state prediction and covariance error. The third step is the computation of the Kalman gain while the fourth step is the estimate calculation (calculation of the output). Finally, the fifth step is the calculation of the covariance error. The algorithm has a loop which connects the fifth step to the second.

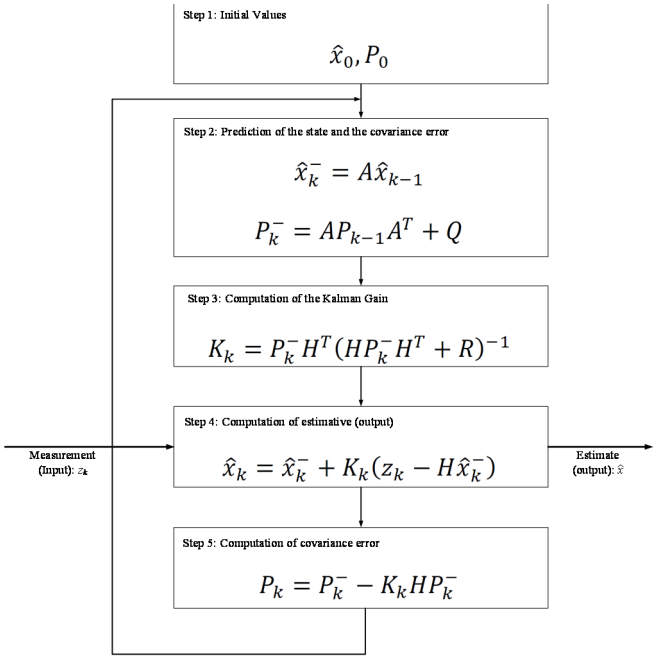

Fig. 1. The Linear Kalman Filter Algorithm.

IV. Understanding the Kalman Filter Algorithm

The linear Kalman filter algorithm has several variable, described as follows:

A. Input z_k (measurement)

The Kalman filter can be simplified drawn as shown in Fig. 2. The reason for presenting the Kalman filter in this way is just to emphasize that the filter has one input and one output. This is valid independently of the order of the filter. However, the input of the Kalman filter is not called input, but measurement. The variable used for the measurement is z_k. By looking at Fig. 1, the measurement z_k is used only in the fourth step.

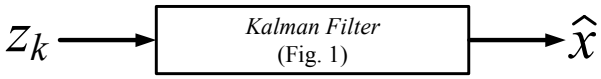

Fig. 2. Simplified diagram of the Kalman filter.

B. Output $\hat{x}_k$ (estimate)

Similar to the input, the output of a Kalman filter is not called output, but estimate. This is due to the fact the output of the filter is actually an estimate of the real output. The variable used for the estimate is $\hat{x}_k$. The estimate is the result of the fourth step. The prior estimate is defined as $\hat{x}_{k-1}$, which is the estimate calculated in the previous loop of the algorithm, and it is used in the second step. Notice that the variable $\hat{x}_k^-$ that appears in the second and fourth step is not the estimate, but the prediction of the estimate. The prediction of the estimate will be described later.

C. Model of the System

The variables ***A***, ***H***, ***Q*** and ***R*** are matrices that describe the model of the input signal. These matrices are used along the algorithm. Discussion about the model of the system will also be described later.

D. Variables for internal calculation

The following variables are used for internal calculation in the algorithm.

- $\hat{x}_k^-$: state prediction
- $\overline{P}_k^-$: covariance error prediction
- P_k: covariance error
- K_k: Kalman gain

where the superscript ⁻ means prediction.

V. Implementing the Kalman Filter Algorithm using Commonly Used Blocks

A. Initial values: Step 1 of the algorithm

Step 1 is just the definition of the initial values for the estimate and for the covariance error, described by:

$$\hat{x}_0, P_0 \tag{4}$$

Fig. 3 presents the implementation of step 1 with commonly used blocks, valid for a second order Kalman filter. For other orders, the implementation would be similar, but with the matrix dimension accordingly. Descriptions about the matrix dimensions are presented in section VI. The initial values for the estimate $\hat{x}_0$ are constant values of a *2x1* matrix. The number between parenthesis right after the variable name indicate the row and the column of the matrix. The $\hat{x}_{0(1,1)}$ indicates the initial value for the first row first column of the estimate matrix and so on. The initial values for the covariance error P_0 are a *2x2* matrix with constant values. The initial values for the matrices are defined by a step source. This source varies from the initial value to zero in a short period of time, usually the time equivalent to a single sampling period. Later, all the initial values are used in the steps 4 and 5. All initial values are drawn with "goto" routing tag.

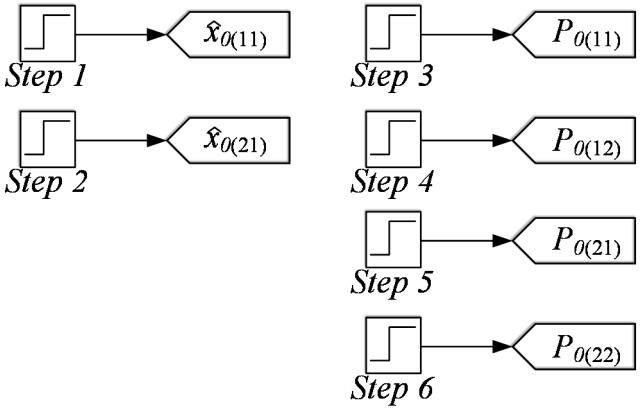

Fig. 3. Implementation of Step 1 with commonly used blocks.

B. Matrices of the system model

Step 2 of the algorithm uses matrix ***A*** and its transposed $\boldsymbol{A}^T$. These matrices are *2x2*. Fig. 4 presents the block implementation of matrix ***A*** and its transposed. The square box are constants. All elements of matrix ***A*** and its transposed are connected to "goto" routing tag. The letter *t* means transposed.

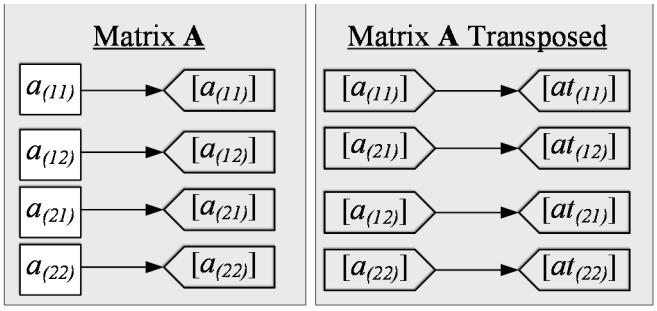

Fig. 4. Block implementation of matrix ***A*** and its transposed $\boldsymbol{A}^T$.

Fig. 5 presents the block implementation of matrix ***H*** and its transposed $\boldsymbol{H}^T$. These matrices are *1x2* and *2x1*, respectively.

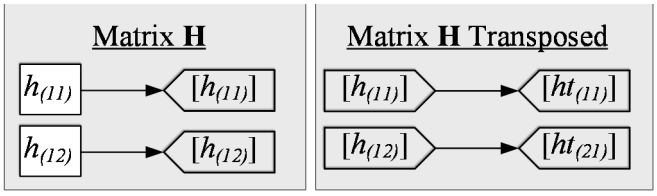

Fig. 5. Block implementation of matrix ***H*** and its transposed $\boldsymbol{H}^T$

Fig. 6 presents the block implementation of matrices ***Q*** and ***R***. The matrix ***Q*** is *2x2* while matrix ***R*** is *1x1*.

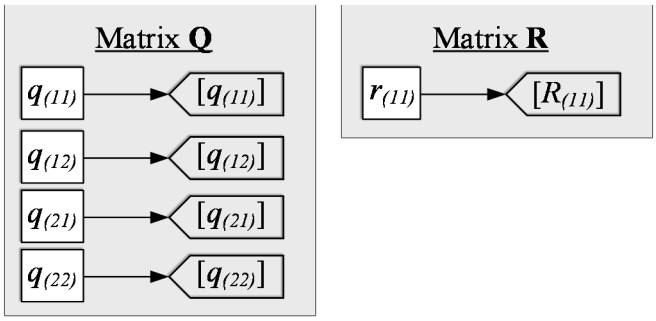

Fig. 6. Block implementation of matrices ***Q*** and ***R.***

C. State and covariance error prediction: Step 2 of the algorithm

Step 2 of the algorithm consist of two parts. The first is the computation of the state prediction, given by:

$$\hat{x}_k^- = \boldsymbol{A}\hat{x}_{k-1}^- \tag{5}$$

The computation of the state prediction is just a product of matrix ***A*** and the previous estimate. Fig. 7 presents the block implementation of the first part of step 2 of the algorithm. The product is implemented with routing tags. Matrix ***A*** tags come from Fig. 4. The previous estimate will be presented later, but it is used in the product and it appears as $x_{(ij)\_prior}$. Matrix ***A*** is *2x2* while the previous estimate matrix $\hat{x}_{k-1}$ is *2x1*. Therefore, the result is a *2x1* matrix, called ***xp***.

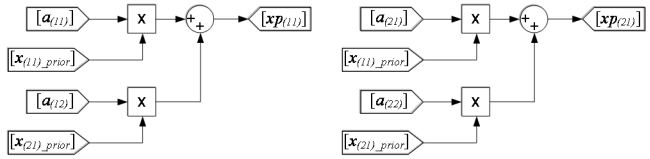

Fig. 7. Block implementation of the first part of step 2 of the algorithm.

The second part of step 2 is the computation of the covariance error prediction, given by:

$$P_k^- = \boldsymbol{A}P_{k-1}\boldsymbol{A}^T + \boldsymbol{Q} \tag{6}$$

The covariance error is implemented with commonly used blocks splitting (6) into three subparts. The first subpart is the product of matrices ***A*** and $\boldsymbol{P}_{k-1}$. Fig. 8 presents the block implementation of the first subpart of the second part of step 2. The result of this subpart is a *2x2* matrix, called ***AP***.

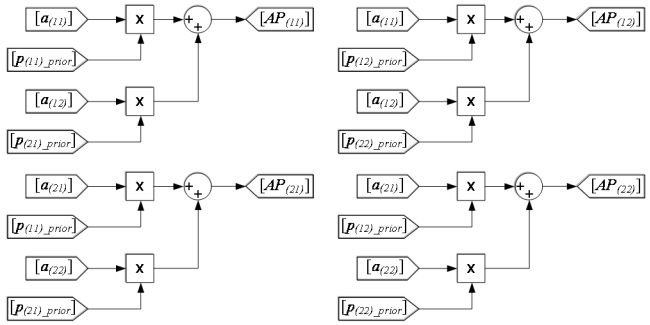

Fig. 8. Block implementation of the first subpart of the second part of step 2.

The second subpart is the product of ***AP*** with matrix ***A*** transposed $\boldsymbol{A}^T$. Both matrices are *2x2*, so the result of the product is also a *2x2* matrix, called ***APAt***. Fig. 9 presents the block implementation of the second subpart of the second part of step 2.

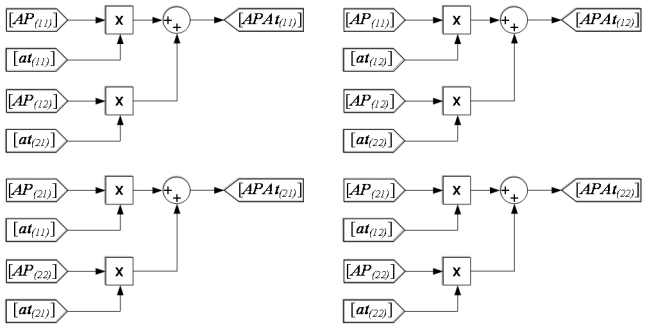

Fig. 9. Block implementation of the second subpart of the second part of step 2.

The third and last subpart consist of adding the ***APAt*** matrix with matrix ***Q***. Both matrices are *2x2*, resulting also in a ***2x2*** matrix, called ***Pp***. Fig. 10 presents the block implementation of the third subpart of the second part of step 2.

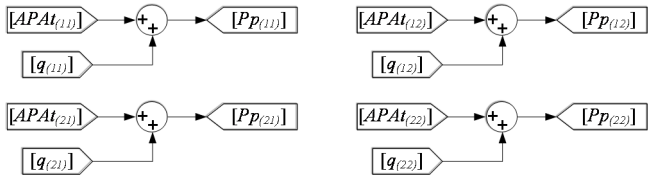

Fig. 10. Block implementation of the third subpart of the second part of step 2.

D. Computation of the Kalman gain: Step 3 of the algorithm

The third step of the algorithm is the computation of the Kalman gain, given by:

$$K_k = P_k^- \boldsymbol{H}^T (\boldsymbol{H}P_k^- \boldsymbol{H}^T + \boldsymbol{R})^{-1} \tag{7}$$

The block implementation of (7) begins with the expression within the parenthesis, which is composed of the sum of two terms. The first term of the sum is the product of three matrices. Fig. 11 presents the block implementation of the first term of the sum within the parenthesis of (7). Notice that the result of this term is a *1x1* matrix, called ***HPpHt***.

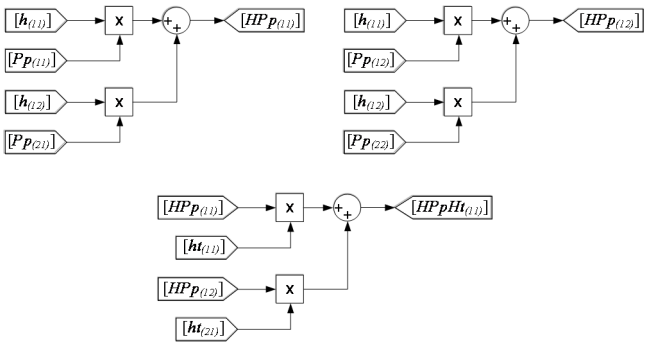

Fig. 11. Block implementation of the first term of the sum within the parenthesis of (7).

Fig. 12 presents the block implementation of the remaining parts of (7). The ***HPpHt*** is summed to matrix ***R***. The result is a *1x1* matrix. Therefore, the inversion of this matrix is just a division, appeared in the product-division

block. The result of step 3 of the algorithm is the *2x1* matrix ***K***.

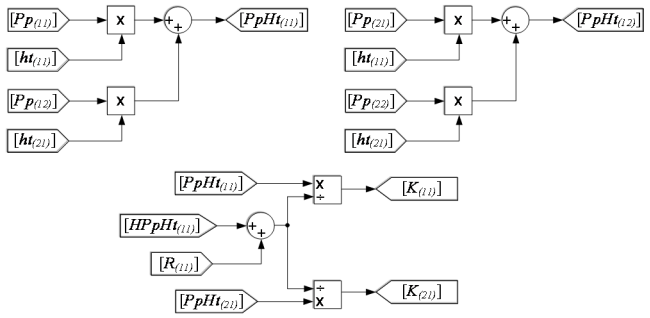

Fig. 12. Block implementation of the remaining parts of (7).

The Kalman gain has an important role in the Kalman filters. Its value defines the weight of the new sample and the previous estimate. Every loop of the algorithm, a new Kalman gain is computed and the weight of the samples are updated. This capability of adjusting the weight of the sample is what makes the Kalman filter so attractive for scenarios where the input signal changes dynamically.

E. Computation of the estimate (output): step 4 of the algorithm

The fourth step of the algorithm is the computation of the estimate. In this step, the measurement (input) is used. The estimate computation is given by (8). The estimate is computed using its prediction, computed in step 2, the Kalman gain and the matrix ***H***.

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - \boldsymbol{H}\hat{x}_k^-) \tag{8}$$

Fig. 13 presents the block implementation of the expression within the parenthesis of (8). The measurement is highlighted for better emphasis of the input. The result of such expression is a *1x1* matrix, called ***zHx.***

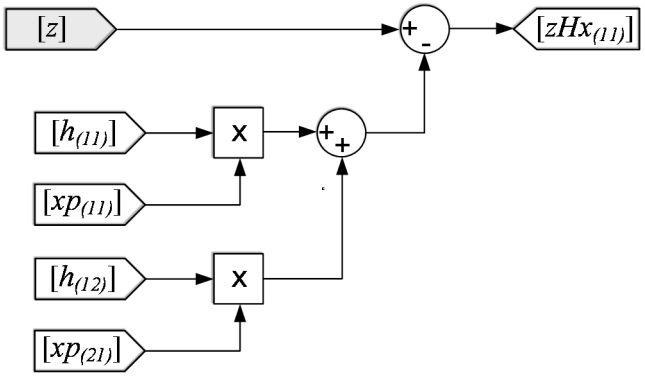

Fig. 13. Block implementation of the expression within the parenthesis of (8).

Fig. 14 presents the block implementation of the remaining parts of (8). The estimate is a *2x1* matrix and it is highlighted to show the output of the Kalman filter. There are two points that deserve attention in this figure: (*i*) the initial values defined in step 1 of the algorithm are used here. The initial values are summed to the current signal. This is the reason the initial values are defined as step sources varying from the defined initial value to zero in a sampling period. So, the initial value has effect only in the first loop, as expected and (*ii*) the estimate passes through a unit delay block, creating the previous estimate which is used in step 2 of the algorithm.

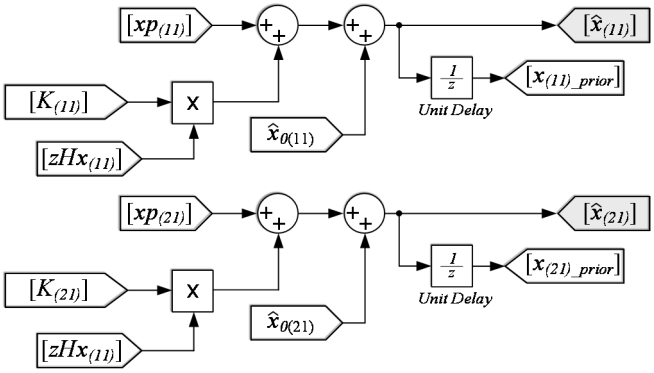

Fig. 14. Block implementation of the remaining parts of (8).

Before presenting the step 5, it is interesting to take a different look at (8). Eq. (8) can be written as showed in (9).

$$\hat{x}_k = (\boldsymbol{I} - K_k\boldsymbol{H})\hat{x}_k^- + K_k z_k \tag{9}$$

where ***I*** is the identity matrix. The reason for presenting the estimate in this way is to show the similarity with the exponential moving average filter, given by (3) and repeated here as (10).

$$\overline{x}_k = \alpha\overline{x}_{k-1} + (1-\alpha)x_k \tag{10}$$

While the exponential moving average filter has a fixed weight of the new sample and the previous output and chosen by empirical manner, the Kalman filter has the weight of the new sample and the previous output decided mainly by the Kalman gain, computed in every loop of the algorithm. This shows the superiority of the Kalman filters over moving average filters.

F. Computing the covariance error: step 5 of the algorithm

The last step of the algorithm is to compute the covariance error. This is required to check if the prediction of the covariance error, computed in step 2, was correctly calculated, making the Kalman filter reaches a satisfactory behavior with minimum error.

The covariance error is given by:

$$P_k = P_k^- - K_k\boldsymbol{H}P_k^- \tag{11}$$

Notice that the computation of the covariance error uses the Kalman gain as well as the matrix ***H*** and the prediction of the covariance error. Fig. 15 presents the block implementation of the product of the Kalman gain and the ***H*** matrices. The result of this product is a *2x2* matrix, called ***KH***.

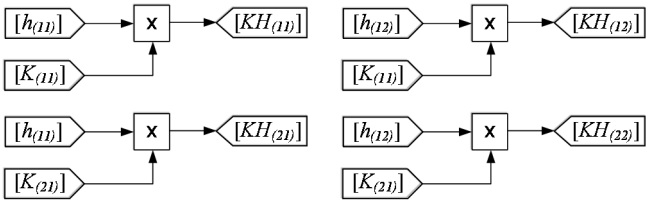

Fig. 15. Block implementation of the product of the Kalman gain and the ***H*** matrices.

Fig. 16 presents the block implementation of the ***KH*** matrix with the prediction of the covariance error. The result of this product is also a *2x2* matrix, called ***KHPp***.

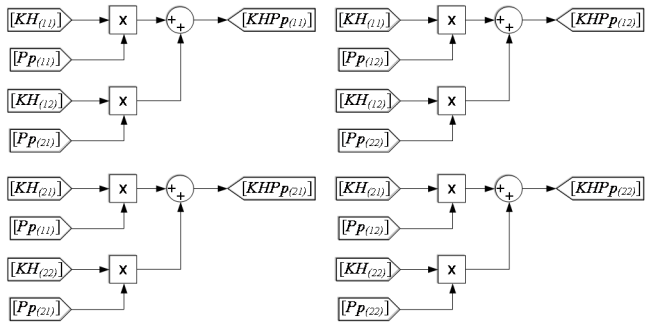

Fig. 16. Block implementation of the ***KH*** matrix with the prediction of the covariance error.

Fig. 17 presents the block implementation of the remaining parts of (10). Notice that the two previous calculated matrices are used in this part. Additionally, the initial values for the covariance error are also used. Similar to the initial values of the estimate, the initial values of the covariance error are summed to the current signal. The result of this step is a *2x2* matrix, which is the previous covariance error, used in step 2 of the algorithm.

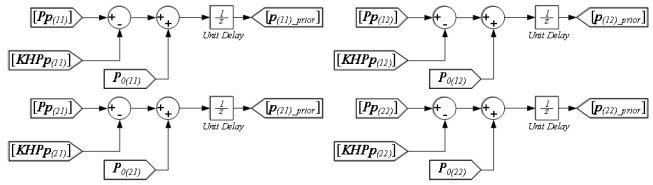

Fig. 17. Block implementation of the remaining parts of (10).

VI. System Model and Matrix Dimension

The Kalman filter algorithm uses matrices that belong to the system model, i.e. it is necessary a mathematical modeling for the process to have their estimation and forecasting. Therefore, the behavior of the Kalman filter is strongly dependent of the system model. The model of the system must describe the input signal accurately, otherwise the Kalman filter may not work as expected. Actually, the main reason for an unsuccessful implementation of Kalman filter is due to inaccuracy of the model system. This section describes how the system should be described in order to correctly use the Kalman filter.

The system model must be described as:

$$\begin{cases} \dot{x}_k = \boldsymbol{A}x_k + w_k \\ z_k = \boldsymbol{A}x_k + v_k \end{cases} \tag{12}$$

The matrices ***Q*** and ***R*** are defined as:

$$Q = covariance\ matrix\ of\ w_k, diagonal\ matrix\ (n\ x\ n)$$

$$R = covariance\ matrix\ of\ v_k, diagonal\ matrix\ (m\ x\ m$$

The definition of each variable of (11) and (12) and its dimension is given as follows:

$$x_k = state\ variable, (n\ x\ 1)$$

$$z_k = measurment\ (m\ x\ 1)$$

$$A = state\ transition\ matrix\ (n\ x\ n)$$

$$H = state\ to\ measurment\ matrix\ (m\ x\ n)$$

$$w_k = state\ transition\ noise\ (n\ x\ 1)$$

$$v_k = measurment\ noise\ (m\ x\ 1)$$

VII. Simulation and Experimental Results

A second order Kalman filter implemented with commonly used block was verified through simulation and experimentally in the Digital Signal Processor (DSP) TMS320F28335. The DSP was programmed through automatic code generation in Simulink. The automatic code generation means that the Simulink build, load and run the DSP converting the simulation made by blocks into code. The simulation and experimental results are placed side-by-side for the sake of comparison and also to show the efficacy of this tutorial for both simulated and experimental environments. The Digital-to-Analog Converter (DAC) MCP4922 was used to measure digital variables in the oscilloscope. Due to the range of the input voltage in the DAC, the signal amplitudes and offset had to be adjusted to be visualized in the DAC. Table I presents the system parameters used in the simulation and in the experimental verification.

TABLE I. System Parameter used in the Simulation and in the Experimental Verification

Parameter	**Value**
Matrix A	$A = \begin{bmatrix} 1 & 0.001 \\ 0 & 1 \end{bmatrix}$
Matrix H	$H = [1 \quad 0]$
Matrix Q	$Q = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}$
Matrix R	$R = [10]$
Matrix $\hat{x}_0$	$\hat{x}_0 = \begin{bmatrix} 0 \\ 20 \end{bmatrix}$
Matrix P_0	$P_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
Sampling frequency	10 kHz

The input signal of the Kalman filter is a sawtooth waveform varying from 0 to 1000 with period of 10 seconds and polluted with noise. Fig. 18 presents the measurement z_k (input) and the estimate $\hat{x}$ (output) of the Kalman filter. This result shows the ability of the Kalman filter to follow the input signal.

Fig. 19 presents the measurement and the estimate in details during the transition of the measurement signal. The estimate rapidly tracks the measurement signal, taking approximately 2 ms to reach the new value.

In order to show the efficacy of the Kalman filter implemented with commonly used blocks, the measurement signal was applied as input signal of an IIR and a FIR low-pass filter. This comparison shows the behavior of these three types of filter.

Figure 20 shows the measurement signal and the output signal of an IIR filter. The response of the IIR filter is oscillatory and takes approximately 40 ms to reach the final value.

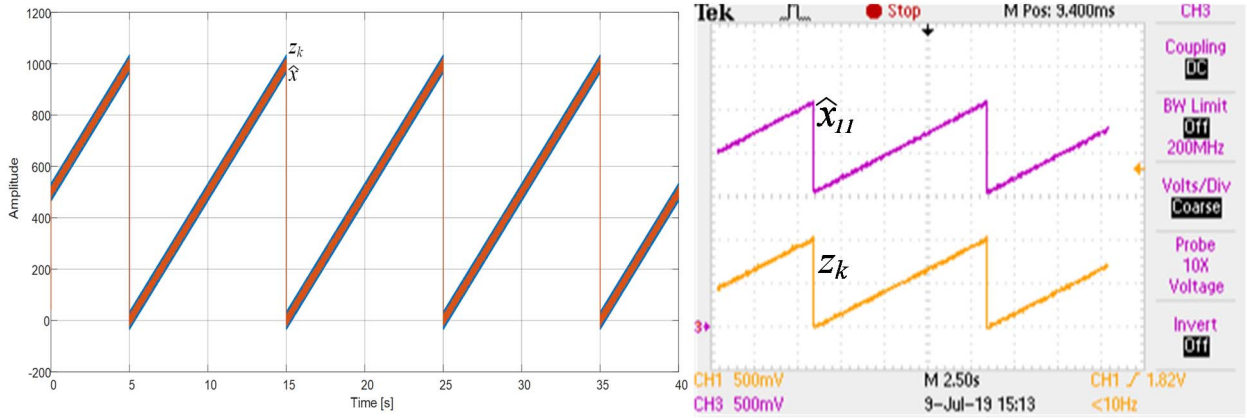

Fig. 18. The measurement z_k (input) and the estimate $\hat{x}$ (output) of the Kalman filter.

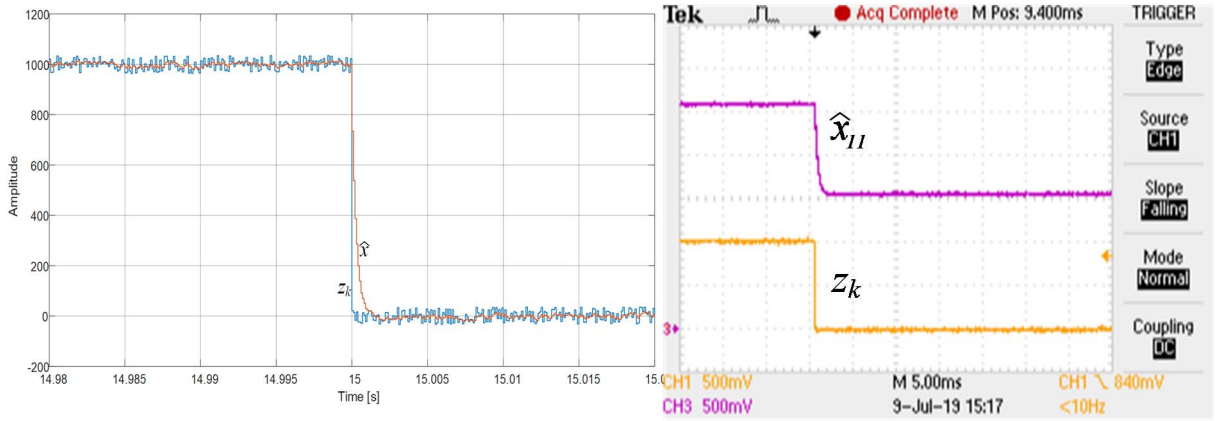

Fig. 19. The measurement and the estimate in details during the transition of the measurement signal.

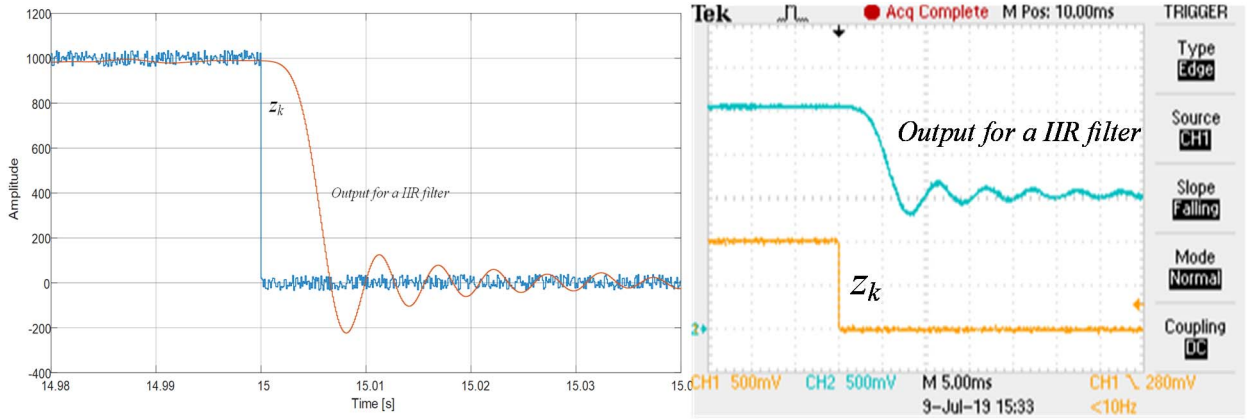

Fig. 20. The measurement signal and the output signal of a IIR filter.

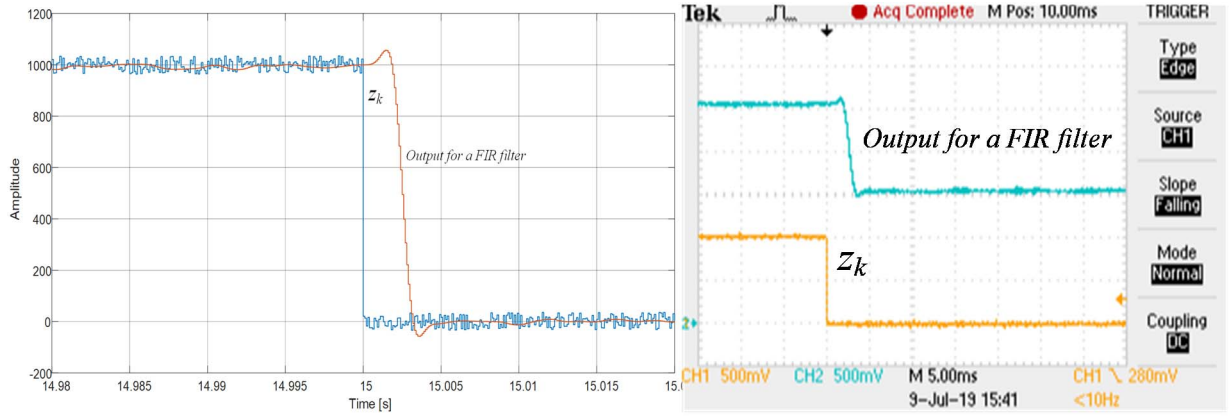

Fig. 21. Measurement signal and the output signal of a FIR filter.

Fig. 21 presents the measurement signal and the output signal of a FIR filter. The response of the FIR filter has some oscillation and it is slow, taking approximately 5 ms to reach the final value.

Fig. 22 presents the two estimates (outputs) of the Kalman filter. The upper signal is the derivative of the signal showed in the central.

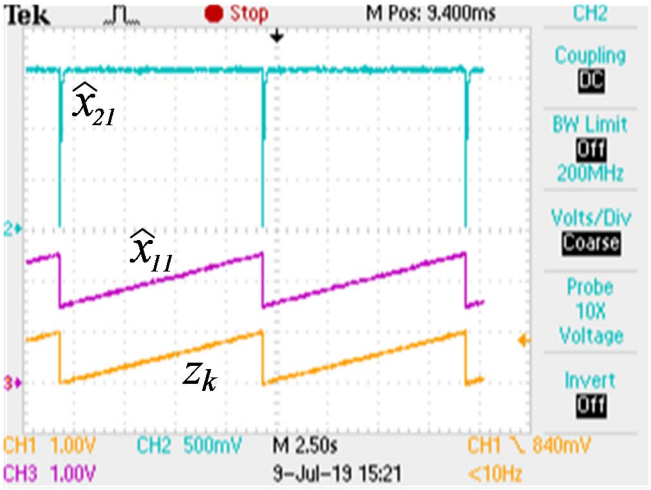

Fig. 22. Measurement signal and the two estimates (outputs) signals of a Kalman filter.

VIII. Conclusions

This paper presented a tutorial on implementing Kalman filters with commonly used block. A second order Kalman filter was implemented with blocks of sum, product, unit delay and zero order holder. The main objective of this tutorial was to make a clear understanding of the Kalman filter algorithm. A brief review on averaging filter was first presented to support the understanding of the Kalman filters. Later, the Kalman filter algorithm was carefully described, step-by-step, and its block implementation is described.

The block implementation of the Kalman filter was verified experimentally in the DSP TMS320F28335. The automatic code generation of Simulink was used to build, load and run the blocks in the DSP. Experimental results showed the efficacy of this tutorial. Therefore, this tutorial helps engineering designers who need a simple, fast, accurate and an easy-to-use implementation of Kalman filters. The simulation files used in this paper will be freely available on the author's webpage http://busarello.prof.ufsc.br.

References

[1] S. Haykin and S. S. Haykin, *Adaptive Filter Theory*, 2014.

[2] A. D. Poularikas and Z. M. Ramadan, *Adaptive Filtering Primer with MATLAB*, 2017.

[3] S. Haykin, *Kalman Filtering and Neural Networks*, 2004.

[4] D. E. Catlin, *Estimation, Control, and the Discrete Kalman Filter*, 2012. [5]A. C. Harvey, *Forecasting, Structural Time Series Models and the Kalman Filter*, 1990.

[5] R. L. Eubank, *A Kalman Filter Primer*, 2005.

[6] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications*, 2003.

[7] P. Kim, *Kalman Filter for Beginners: With MATLAB Examples*, 2011.

[8] T. Ishihara and L. A. Zheng, "LQG/LTR procedure using reduced-order Kalman filters," *International Journal of Control*, vol. 0, no. 0, pp. 1–15, 2017.

[9] A. Feddaoui, N. Boizot, E. Busvelle, and V. Hugel, "High-gain extended Kalman filter for continuous-discrete systems with asynchronous measurements," *International Journal of Control*, vol. 0, no. 0, pp. 1–14, 2018, C:et al. – 2018.

[10] G. Feng, C. Lai, J. Tjong, and N. C. Kar, "Noninvasive Kalman Filter Based Permanent Magnet Temperature Estimation for Permanent Magnet Synchronous Machines," *IEEE Transactions on Power Electronics*, vol. 33, no. 12, pp. 10 673–10 682, 2018, C:8UN27et al. – 2018.

[11] S. R. Jondhale and R. S. Deshpande, "Kalman Filtering Framework-Based Real Time Target Tracking in Wireless Sensor Networks Using Generalized Regression Neural Networks," *IEEE Sensors Journal*, vol. 19, no. 1, pp. 224–233, 2019.

[12] Z. Lin, H. H. T. Liu, and M. Wotton, "Kalman Filter-Based Large-Scale Wildfire Monitoring With a System of UAVs," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 1, pp. 606–615, 2019.

[13] W. Yan, B. Zhang, G. Zhao, S. Tang, G. Niu, and X. Wang, "A Battery Management System With a Lebesgue-Sampling-Based Extended Kalman Filter," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 4, pp. 3227–3236, 2019.

[14] S. P. Talebi and S. Werner, "Distributed Kalman Filtering in Presence of Unknown Outer Network Actuations," *IEEE Control Systems Letters*, vol. 3, no. 1, pp. 186–191, 2019.

[15] B. Ristic, J. Houssineau, and S. Arulampalam, "Robust target motion analysis using the possibility particle filter," *Sonar Navigation IET Radar*, vol. 13, no. 1, pp. 18–22, 2019.

[16] H. Yu, Z. Li, J. Wang, and H. Han, "Data fusion for a GPS/INS tightly coupled positioning system with equality and inequality constraints using an aggregate constraint unscented Kalman filter," *Journal of Spatial Science*, vol. 0, no. 0, pp. 1–22, 2018.

[17] V. Mahboub, S. Ebrahimzadeh, M. Saadatseresht, and M. Faramarzi, "On robust constrained Kalman filter for dynamic errors-in-variables model," *Survey Review*, vol. 0, no. 0, pp. 1–8, 2018.

[18] H. Yang, P. J. Jin, B. Ran, D. Yang, Z. Duan, and L. He, "Freeway traffic state estimation: A Lagrangian-space Kalman filter approach," *Journal of Intelligent Transportation Systems*, vol. 0, no. 0, pp. 1–16, 2018.

[19] K. S. Babu and K. Detroja, "Inverse Free Kalman Filter Using Approximate Inverse of Diagonally Dominant Matrices," *IEEE Control Systems Letters*, vol. 3, no. 1, pp. 120–125, 2019, C:e Detroja – 2019.

[20] T. A. Johansen and T. I. Fossen, "The eXogenous Kalman Filter (XKF)," *International Journal of Control*, vol. 90, no. 2, pp. 161–167, 2017.

[21] Z. Gao, "Fractional-order Kalman filters for continuous-time linear and nonlinear fractional-order systems using Tustin generating function," *International Journal of Control*, vol. 0, no. 0, pp. 1–15, 2017.

[22]]M. Skliar and W. F. Ramirez, "Implicit Kalman filtering," *International Journal of Control*, vol. 66, no. 3, pp. 393–412, 1997.