

Implementing a Kalman Filter on FPGA Embedded Processor for Speed Control of a DC Motor Using Low Resolution Incremental Encoders

Herman I. Veriñaz Jadan, Caril R. Martinez Vera,
Ronald Ponguillo Intriago, *Member, IAENG*, Vladimir Sanchez Padilla, *Member, IAENG*

Abstract— Incremental rotary encoders are used to measure the angular position of a motor shaft. It is required to calculate the first difference of the angular position in order to measure angular velocity. The operation could cause noisy signals, mainly because the encoder output is from digital nature, plus sensor resolution is limited. To avoid this scenario, filters are used before doing operations with this output signal. This paper presents a practical method for implementing a steady state Kalman filter, using engine's mathematical model algorithm designed with MATLAB. The filter was used for speed control of a DC motor, and the controller performance used the filtering algorithm resulting in a considerable improvement compared to the same controller without using the filter. The algorithm presented was used as part of the system control of a sprinter robot and implemented in an embedded system building with a NIOS II soft microprocessor inside of an FPGA of Altera Corporation.

Index Terms—Digital filter, FPGA, Kalman filter, NIOS II processor, speed measurement, velocity control.

I. INTRODUCTION

IT is common at sprinter robot tournaments to implement velocity controllers to improve robot performance on the track. Rotary encoders used in these applications generally have low resolution, making an unexpected behavior of the controller. To solve this issue, it is usually implemented filtering algorithms [2].

This paper presents a Kalman filter designed for a sprinter robot. The algorithm presented was designed and implemented with MATLAB in a NIOS II processor [1][3] over an Altera FPGA [5]. To reduce the use of computational resources, the time variant algorithm is not implemented on the Kalman filter, but only the steady-state filter [4]. For the system identification, MATLAB's Ident toolbox was used. The mathematical model identified was considered for the design of both the controller and the filter. The identified system represented as matrix of state is:

$$\begin{aligned}\dot{x} &= Ax + Bu \\ y &= Cx + Du\end{aligned}\quad (1)$$

Where:

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad A = \begin{pmatrix} k_1 & k_2 \\ 1 & 0 \end{pmatrix} \quad B = \begin{pmatrix} k_3 \\ 0 \end{pmatrix}\quad (2)$$

$$C = (1 \quad 0) \quad D = 0\quad (3)$$

$$k_1 = -5760 \quad k_2 = -2.116e5 \quad k_3 = 6663\quad (4)$$

The representations are:

x_2 , angular velocity of the motor shaft.

x_1 , represents the angular acceleration.

y , represents the output system, the angular velocity.

Matrixes A, B, C, and D are obtained from the identification in MATLAB.

II. MATHEMATICAL MODEL

When the transfer function in MATLAB is identified, the model does not include any external perturbation into the system. In order to improve the filter prediction, this effect is added to the model. The following equation is obtained after applying Newton's second law for rotational dynamics over the system:

$$\sum \tau = I\alpha\quad (5)$$

Where τ represents the magnitude of each torque acting on the motor shaft; I represents the inertia moment on the motor shaft; and α represents the angular acceleration of the system. This sum of torques can be separated from the external torque:

$$\sum \tau_{m1} + \tau_{ex} = I\alpha\quad (6)$$

Where τ_{m1} represents the magnitude of each torque acting on the motor shaft (not including the external torque to the system) and τ_{ex} is the external torque of the system.

In addition to equation 7, x_1 is the acceleration of the system that does not include the external torque:

Manuscript received July 17th, 2016; revised August 2nd, 2016

The authors are with the Escuela Superior Politécnica del Litoral, ESPOL, Faculty of Electrical and Computer Engineering, Campus Gustavo Galindo, Km 30.5 Via Perimetral, P.O. Box 09-01-5863, Guayaquil, Ecuador (email: {hverinaz, crmartin, rponguil, vladsanc}@espol.edu.ec), website: www.espol.edu.ec

$$\sum \tau_{m1} = Ix_1 \quad (7)$$

Then:

$$Ix_1 + \tau_{ex} = I\alpha \quad (8)$$

$$\alpha = x_1 + \tau_{ex}/I$$

Therefore, the angular acceleration of the new model is just the angular acceleration model identified in MATLAB plus τ_{ex}/I . The new angular acceleration is \dot{x}_3 , and x_4 is the value of τ_{ex}/I . Then, the system equation adding the new equation will be:

$$\dot{x}_1 = k_1x_1 + k_2x_2 + k_3u \quad (9)$$

$$\dot{x}_2 = x_1 \quad (10)$$

$$\dot{x}_3 = x_1 + x_4 \quad (11)$$

An additional equation that describes the dynamics of x_4 is needed, but this variable is related to the introduced external torque without information about it, whereby a constant torque model is used:

$$\dot{x}_1 = k_1x_1 + k_2x_2 + k_3u \quad (12)$$

$$\dot{x}_2 = x_1 \quad (13)$$

$$\dot{x}_3 = x_1 + x_4 \quad (14)$$

$$\dot{x}_4 = 0 \quad (15)$$

$$y = x_3 \quad (16)$$

Matrixes system is obtained:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{pmatrix} = \begin{pmatrix} k_1 & k_2 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} k_3 \\ 0 \\ 0 \\ 0 \end{pmatrix} u \quad (17)$$

$$y = (0 \quad 0 \quad 1 \quad 0) \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \quad (18)$$

The sampling time used for the implementation in discrete time is 0.5ms. The discrete time model equation is:

$$x_{k+1} = Ax_k + Bu_k \quad (19)$$

$$y_k = Cx_k + Du_k$$

The matrixes can be obtained using MATLAB from the continuous time model [7]:

$$A = \begin{pmatrix} 0.051 & -34.4020 & 0 & 0 \\ 0.0002 & 0.9877 & 0 & 0 \\ 0.0002 & -0.0123 & 1 & -0.0005 \\ 0 & 0 & 0 & 1 \end{pmatrix} B = \begin{pmatrix} 1.08 \\ 3.87e-4 \\ 3.87e-4 \\ 0 \end{pmatrix} \quad (20)$$

$$C = (0 \quad 0 \quad 1 \quad 0) \quad D = 0 \quad (21)$$

III. KALMAN FILTER

The implemented algorithm is the steady state Kalman filter. A MATLAB script was developed to design the filter, where the complete algorithm is implemented with matrix equations, both prediction and upgrading. The equations are:

Prediction:

$$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + B_k u_k \quad (22)$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \quad (23)$$

Upgrading:

$$K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1} \quad (24)$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \quad (25)$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (z_k - H_k \hat{x}_{k|k-1}) \quad (26)$$

The script uses real data from the measured angular velocity and the motor's measured input voltage in percent of the PWM signal duty cycle [7] (Fig. 1).

Besides these information, it is needed to define the values of the covariance matrixes Q and R. There is no standard method to obtain the value of these matrixes, and they were obtained manually until getting the desired result of the filtering. Both filters (input and output, Fig. 2) with the matrixes Q and R set correctly are:

$$R = 25 \quad (27)$$

$$Q = \begin{pmatrix} 0.001 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 10000 \end{pmatrix} \quad (28)$$

Furthermore, once obtained the values for the matrixes Q and R, it can be analyzed the elements behavior of the gain matrix. Values taken by these elements in time are shown in Figure 3. Each element converges to a specific value. The group of values form the steady state gain matrix:

$$K = \begin{pmatrix} k_1 \\ k_2 \\ k_3 \\ k_4 \end{pmatrix} = \begin{pmatrix} 0.0024118 \\ -6.5265e-05 \\ 0.096626 \\ 6.0112 \end{pmatrix} \quad (29)$$

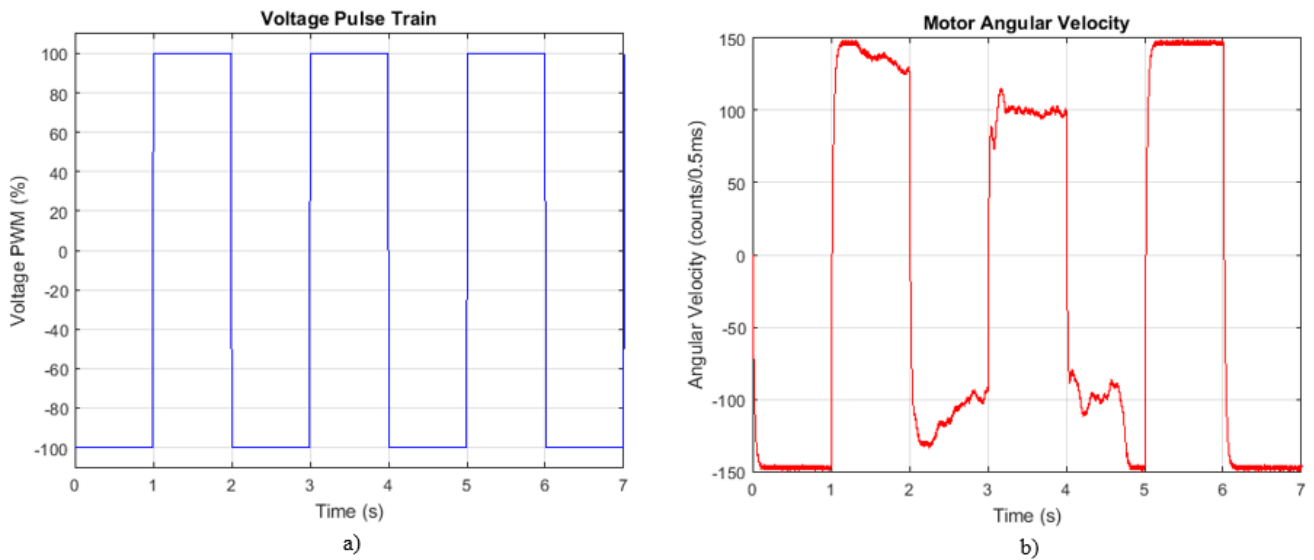


Fig. 1. Input signals for the simulation of the filtering system:
 a) Voltage motor input measured in % of PWM; b) Angular velocity measured in counts / 0.5ms.

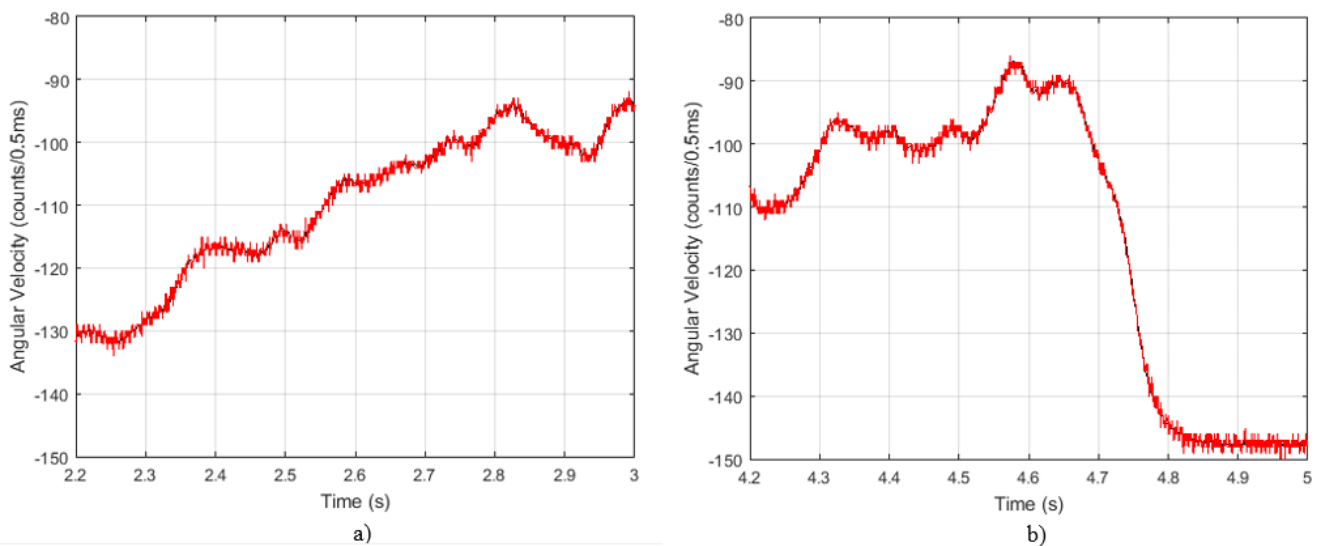


Fig. 2. Filter response with matrixes Q and R set correctly:
 a) Signal measured between 2 to 3 seconds; b) Signal measured between 4 to 5 seconds.

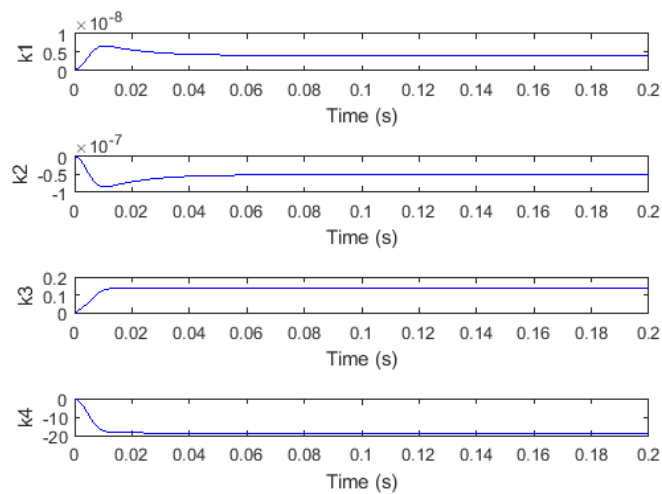


Fig. 3. Convergence of the Kalman filter's gain matrix elements.

IV. IMPLEMENTATION AND RESULTS

The filter was implemented in a NIOS II [1][3] single core processor over the FPGA EP4CE22F17C6N, Altera's Cyclone IV family, using a development board DE0-Nano [6]. The system works with a sampling time of 0.5 ms, and the used motor has coupled a magnetic quadrature encoder of 512 pulses per revolution. The motor is a FAULHABER 2224SR06 with an IE2-512 encoder. The sensor used was not in an optimal state, decreasing the lectures quality. The final system was implemented in FPGA and the resulting matrixes are:

Prediction:

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + Bu_k \quad (30)$$

Upgrading:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K(z_k - C\hat{x}_{k|k-1}) \quad (31)$$

Where:

$$A = \begin{pmatrix} 0.051 & -34.4020 & 0 & 0 \\ 0.0002 & 0.9877 & 0 & 0 \\ 0.0002 & -0.0123 & 1 & -0.0005 \\ 0 & 0 & 0 & 1 \end{pmatrix} B = \begin{pmatrix} 1.08 \\ 3.87e-4 \\ 3.87e-4 \\ 0 \end{pmatrix} \quad (32)$$

$$C = (0 \ 0 \ 1 \ 0) \quad K = \begin{pmatrix} 0.0024118 \\ -6.5265e-05 \\ 0.096626 \\ 6.0112 \end{pmatrix} \quad (33)$$

With the mathematical model presented in the introduction, the controller was designed through the MATLAB's Sisotool (single input, single output tool). The controller is represented by the equation:

$$u_M(k) = 0.2545u_M(k-1) + 2217err_M(k) - 1167err_M(k-1) \quad (34)$$

Where $err_M(k)$ is the feedback error in the control system and $u_M(k)$ is the controller output.

A step input of size 80 was used to test the controller. This value means that 80 counts in the encoder occur every 0.5 ms (equivalent to 4687.5 RPM) [7]. The change of the step signal occurs at 0.1 s after the signal starts, as shown in Fig. 4, where in a) the encoder output signal is used directly to measure the system error; and, b) represents the controller implementation using the designed Kalman filter. The system has a steady time of approximately 25 ms, and the error steady state is 0.7 counts/0.5 ms.

V. CONCLUSIONS

The controller failed to put the system in steady state when it was used directly the signal read by the rotary encoders. The signal oscillates around the reference but does not stabilize due to the digital nature of the signal and also because the encoders were not in an optimal state.

The use of a Kalman filter allows to soften the control, with a response signal resulting in less noise than the signal

without filtering. The system achieves stabilization in a response time and with a suitable steady state [7]. Despite the use of a filtering algorithm, the system does not present any important delay in the response of the controller. This could happen if filters that cut the frequency spectrum are used [2].

MATLAB script for simulation allows to work in the filtered code improvements before making the final implementation, i.e., it can be implemented the algorithm on a script to be used as data entry in order to observe the result in graphic outputs to perform setting parameter. Once the desired performance is achieved, the final design in the processor is implemented.

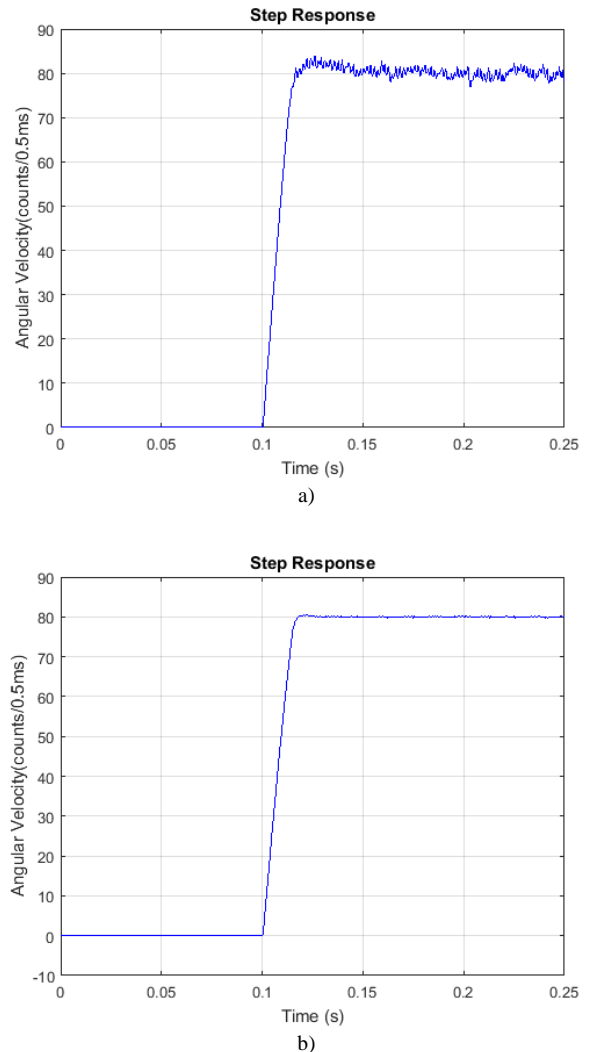


Fig. 4. Controller response: a) Without filtered algorithm; b) With filtered algorithm.

REFERENCES

- [1] Altera Corporation, "Introduction to the Altera Nios II Soft Processor", San Jose, CA, 2008.
- [2] Bellini, A., Bifaretti, S., & Costantini, S. "A Digital Speed Filter for Motion Control Drives with a Low Resolution Position Encoder", *AUTOMATIKA*, 67-74, 2003.
- [3] Chu, P. P., "Embedded SOPC Design with NIOS II Processor and VHDL Examples", Hoboken, NJ: John Wiley & Sons, Inc., 2011.
- [4] Fadali, M. S., & Visioli, A., "Digital Control Engineering Analysis and Design", Second Edition. Elsevier Inc., 2013.
- [5] Moore, A., "FPGAs For Dummies", Altera Special Edition. Hoboken: John Wiley & Sons, Inc., 2014
- [6] Terasic Technologies Inc., DE0-Nano User Manual, 2013.

- [7] Veriñaz, H., Martínez, R., & Ponguillo, R., "Diseño e Implementación de un Robot Velocista de Competencia sobre Plataforma FPGA". Guayaquil-Ecuador: Dspace ESPOL, Thesis for engineering degree, 2015