An optimal algorithm for estimating angular speed using incremental encoders

Un algoritmo óptimo para la aproximación de la velocidad angular utilizando encoders

J. D. Rairán Antolines[1]

ABSTRACT

This paper proposes a new algorithm using signals from an incremental encoder for estimating a rotating shaft's speed. This algorithm eliminated the oscillations appearing in classical fixed-time and fixed-space algorithms, even when speed was constant. A fixed-time algorithm measures angular displacement at fixed-time intervals, while a fixed-space algorithm measures time every fixed-angular displacement. Time and displacement measurements were used to generate estimations for speed. The new algorithm generated a unique value for estimating speed due to synchronising encoder pulses and a signal formed by impulses every time increase (delta). A first modification of the proposed algorithm was defined, resulting in the harmonic mean between these two proposed alternatives having the smallest relative error possible. This error was always smaller than half the error with fixed-time and fixed-space algorithms. Experimental setup and algorithms are shown, as well as Simulink results using signals acquired from an incremental encoder.

Keywords: Fixed-space, fixed-time, incremental encoder, optimal algorithm, estimating speed, truncation error.

RESUMEN

En este artículo se propone un algoritmo nuevo, el cual utiliza las señales provenientes de un *encoder* incremental para aproximar la velocidad de giro de un eje. Este algoritmo elimina las oscilaciones que aparecen en los algoritmos clásicos, conocidos como a espacio fijo y a tiempo fijo, las que son visibles incluso a velocidad constante. Un algoritmo a tiempo fijo mide desplazamiento angular a intervalos de tiempo fijo, mientras que un algoritmo a espacio fijo mide tiempo a intervalos de espacio fijo. Las medidas de tiempo y desplazamiento se utilizan para estimar la velocidad. El algoritmo nuevo genera una aproximación sin saltos a velocidad constante, dada la sincronización entre los pulsos del encoder y la señal formada a partir de impulsos cada delta de tiempo. Se propone una primera modificación del algoritmo, y el resultado es que el promedio armónico entre las dos propuestas de algoritmo tiene el menor error relativo posible. Ese error siempre es menor que la mitad del error que se produciría con los algoritmos a tiempo fijo o a espacio fijo. Finalmente, se presenta un arreglo experimental para probar los algoritmos, y se muestran los resultados de la ejecución en Simulink, usando señales adquiridas de un encoder incremental.

Palabras clave: Aproximación de la velocidad, error de truncamiento, espacio fijo, tiempo fijo.

Received: May 31st 2012
Accepted: July 19th 2013

Introduction

Estimating speed represents an interesting problem, especially regarding control involving the dynamics of a system based on signals given by sensors; i.e. based on an estimations of what actually happens. The algorithm proposed in this paper was aimed at obtaining maximum information from an incremental encoder, thus establishing a fast and stable measurement of a shaft's angular speed. An incremental encoder is an electromechanical instrument. The mechanical part is an evenly spaced slotted disc, whereas the electric part includes a light beam and detector. A high light level (one) is recorded when the light passes through the slit and reaches the detector, and zero otherwise.

Measuring speed has one physical limitation; estimating speed requires measuring two instants in time and the result necessarily neglects one of them. An estimation of speed has at least two sources of error because it requires two measurements: position and time. Since ω_m is $\Delta\theta/\Delta t$, error regarding estimation ω_m lies in the nature of such deltas (i.e. discrete and digital values). Digital measurement involves an inevitable truncation error. Thus, $\Delta\theta$ is an integer multiple of the pulses emitted by the encoder (*pul*), and *dt* is likewise an integer multiple of clock resolution (*ts*).

The speed at which the time between two consecutive pulses from the encoder lasts exactly *dt* defines the so-called *speed limit* given in revolutions per second (ω_{lim} = (1/*pul*)/*dt*). Speed values less than ω_{lim} are low, whereas greater or equal than this are high. A similar concept (Tsuji *et al.*, 2009) has defined what authors call speed resolution.

[1] José Danilo Rairán Antolines. Ingeniero Electricista, Universidad Nacional de Colombia, MSC Computer Science, Universidad Nacional de Colombia. Affiliation: Universidad Distrital Francisco Jose de Caldas, Colombia.
E-mail: drairan@udistrital.edu.co

How to cite: Rairán, J. D., An optimal algorithm for estimating angular speed using incremental encoders., Ingeniería e Investigación. Vol. 33, No. 2. August 2013, pp. 56 – 62.

Two classical algorithms estimate speed (fixed-time and fixed-space algorithms); other algorithms usually come from these two. If an estimation involved using the fixed-time approach, which means updating the counting of pulses (*Nep*) each *dt*, then the algorithm generates an estimation $\omega_m = Nep\,\omega_{lim}$. On the other hand, the fixed-space approach updates *dt* counts (called *Ndt*) between every two consecutive pulses, so $\omega_m = (1/Ndt)\,\omega_{lim}$. The fixed-space algorithm is recommended for low speeds, whereas the fixed-time algorithm is useful for high speeds. The truncation error makes ω_m oscillate regardless of the approach adopted. The effect of these oscillations requires using low-pass filters to smooth the value of the estimation, thereby implying delay.

Current solutions for avoiding the effect of truncation can be classified mainly into two groups: highly accurate measurements of time and synchronising position and time signal (a third group is also referenced, including other options). Solutions from the first group use the power of computing electronic circuits for high resolution measurement of time, for instance microcontrollers (Petrella and Tursini, 2008), DSP (Boggarpu and Kavanagh, 2010) and FPGAs. A more elaborate solution (Merry, Molengraft and Steinbuch, 2007) detects rising edges from the encoder to make a polynomial fitting, thereby smoothing the estimation. Other work (Lygouras *et al*., 2008) has gone so far as to claim that the truncation error effect disappears when clock resolution reaches 50 ns. The oscillations within the estimation, though small, still remain in this first solution group. On the other hand, the second group looks for synchronising position and time measurementents. An algorithm named the S method (Tsuji *et al*., 2009) has been proposed based on this synchronisation concept, mainly being helpful for high speed. Another proposal (Se-Han, Lasky and Velinsky, 2004) for optimising encoders with few slits uses synchronisation for high speeds, whereas for low speeds it switches the algorithms for another based on its model of the system. The third group of solutions is not exactly focused on the truncation problem, but nonetheless the problem is well referenced and is taken into account in the solution. Some work (Su *et al*., 2005) has focused on very low speeds, estimation being made by absolute position measurement while the average speed of every rotation (Hachiya and Ohmae, 2007) has been used for developing a simple and effective method for eliminating fluctuations, as well as mitigating the effect of some mechanical errors. Another solution consists of using filters and switching from fixed-time to fixed-space methods or vice versa to ensure obtaining the smallest relative error possible (D'arco, Piegari and Rivo, 2003). It has been suggested that some steps can be traced backwards to control the highest relative error (Liu, 2002).

Describing the new algorithm

The new algorithm presented in this section eliminated oscillations in estimating speed when they came from discrete and digital measurements, as in an incremental encoder. The core of the proposal consisted of synchronizing position and time pulses. Since there was no control over the time at which the encoder pulse appeared, the algorithm simply started to count *dt* when a position pulse appeared. As a result, such estimation would have been constant for constant speeds, without the use of state observers or system models as required in other work (Se-Han, Lasky and Velinsky, 2004).

Guaranteeing the mathematical results in this section involved the following assumptions: the encoder did not lose pulses during counting, measuring time was exact and in multiples of clock time (*ts*) and the mechanical distance between pulses remained constant. These assumptions idealised an incremental encoder's true performance, although they were necessary for ensuring error bounds on the estimations.

The first step in transforming the encoder signal detected the rising edge from each pulse. This detection generated a pulse train (*Iep*) as shown in Figure 1. The next step counted these pulses and led to a stair-type signal named (*Cep*). If counting *Cep* went to zero, for instance because the time reached *dt*, then a register saved the last value in the counter into a variable Nep_I. The scheme in the lower part of Figure 1 illustrates an additional series of blocks. These blocks counted *dts*. A pulse *dt* resulted when the accumulation of *ts* reached the value of *dt*. This event reset and started the counting of *ts* over again. The counter block held *dt* counting in a variable *Cdt*. If an *Iep* pulse reset that counter, then a register held the last value of *Cdt* into Ndt_I.

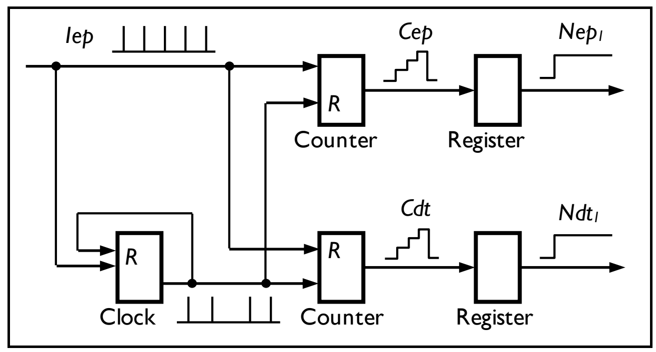

Figure 1. Algorithm scheme and definition of Nep_I and Ndt_I

Speed was estimated as $\omega_{mI} = n_I\,\omega_{lim}$, where n_I was Nep_I/Ndt_I. Counting required that Nep_I should be a whole number (0, 1, 2,…), while Ndt_I had to be a natural number. There were more *Iep* pulses in the count than *dt* pulses for speeds higher than ω_{lim}, so Ndt_I stayed fixed at one. On the contrary, there were more *dt* pulses than *Iep* pulses for speeds lower than ω_{lim}, so Nep_I remained fixed at one. Possible values for n_I, for high speeds were 1, 2, 3, and so on, and ½, ⅓, ¼, and so on for low speeds.

Table 1. Pseudocode for the proposed speed estimation algorithm

	Input: ts, dt, Iep impulses
	Output: Ndt_I, Nep_I
1	**Begin**
	Cep ← 0, Cdt ← 0, Nep_I ← 0, Ndt_I ← 1
2	clock ← 0
3	**Repeat**
4	clock ← clock + ts
5	**if** Iep = 1 & Cdt ≠ 0 **then**
6	Ndt_I ← Cdt
7	clock ← 0
8	**if** clock = dt & Cep ≠ 0 **then**
9	Nep_I ← Cep
10	**if** Iep = 1 **then**
11	Cep ← Cep + 1, Cdt ← 0
12	**if** clock = dt **then**
13	Cdt ← Cdt + 1, Cep ← 0
14	clock ← 0
15	**return**(Ndt_I,Nep_I)
16	**until** stop
17	**End**

Table 1 shows the algorithm's pseudocode. This algorithm generated a new estimation ω_{mI} every *ts* by computing Ndt_I and Nep_I. The core of the algorithm synchronised time and space pulses in line 6 where the clock went to zero. However, this synchronisation did not mean that Ndt_1 and Nep_1 updated their values synchronously: Ndt_1 updated its value in line 6 (synchronised with *Iep* pulses) and Nep_1 updated its value in line 9 (synchronised with *dt* pulses).

Plots a) and b) in Figure 2 illustrate algorithm operation for low and high speeds, respectively. Event *a* started the programme in both examples, and subsequent letters defined other events. For instance, events *d* and *g* updated Nep_1 in both examples. Estimation ω_{m1} remained constant for events later than *f* in plot a. On the other hand, estimation after any event later than *d* in plot b remained constant.

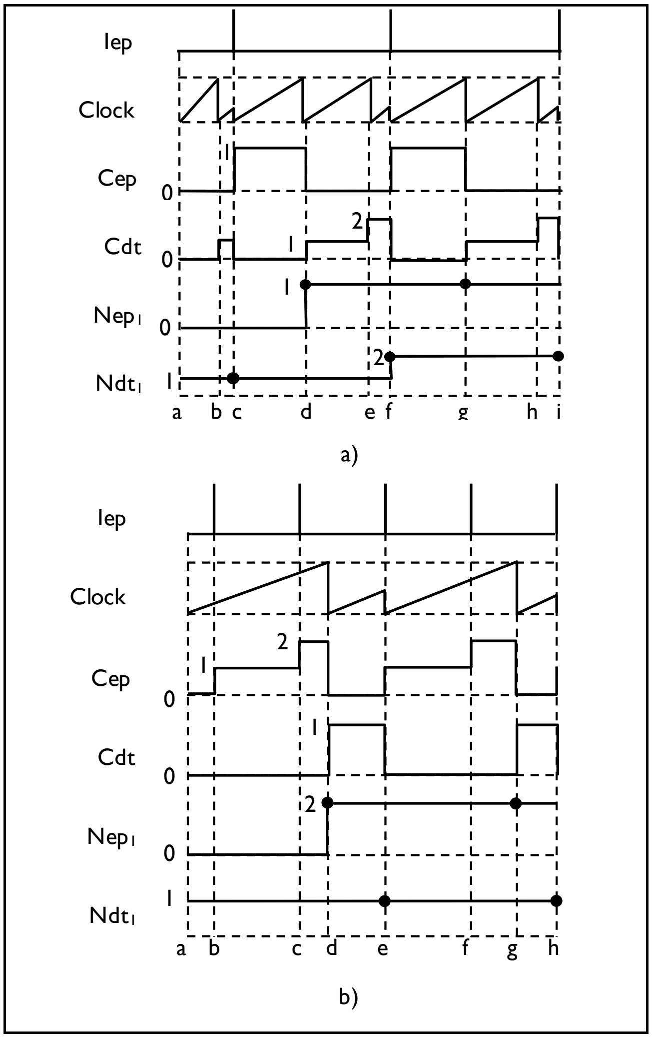

Figure 2. Examples of algorithm use: a) $\omega \approx 0.25\omega_{lim}$, b) $\omega \approx 1.5\omega_{lim}$

Solid points in Figure 2 depicted the time when the algorithm updated Nep_1 or Ndt_1. The time between updates for low speeds was $dt_a = (\omega_{lim}/\omega)\ dt$, and for high speeds dt_a (see Eq. 1). The maximum value of dt_a in Equation 1 was twice *dt* as speed approached ω_{lim}, and delay decreased as ω grew, with *dt* as its limit:

$$dt_a = \left(\frac{-1}{\left\lfloor \frac{\omega}{\omega_{lim}} \right\rfloor} \cdot \frac{\omega}{\omega_{lim}} + 2 + \frac{1}{\left\lfloor \frac{\omega}{\omega_{lim}} \right\rfloor} \right) dt \quad (s) \qquad (1)$$

The value of Nep_1 represented the ceiling for the number of pulses *Iep*, as can be seen in Figure 2; Ndt_1 was the floor for counting *dts*, so estimation n_1, defined as Nep_1/Ndt_1, was always greater than the actual speed, as shown in Figure 3, where the X and Y axes had been scaled by ω_{lim}.

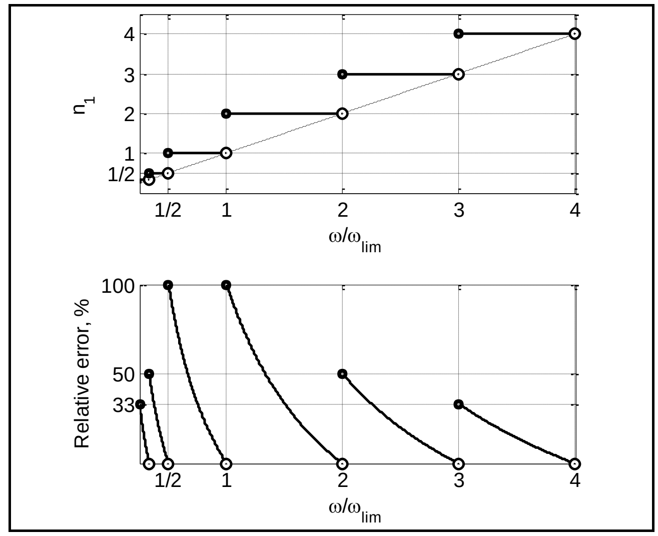

Figure 3. Speed estimation, ω_{m1}, and relative error for the proposed algorithm

The first measurement of quality for estimating speed correlated relative error and speed, as shown in the lower part of Figure 3. A second measurement of quality (maximum relative error per interval) took into account that the value of ω actually was unknown. An interval covered the whole range of speeds having the same estimation. For instance, all speeds estimated as $n_1 = 3$ as estimation defined an interval. This second measurement of quality depended entirely on algorithm output. Even so, Equation 2 showed the definition of relative error for the second measurement of quality to come up with an equation for this indicator:

$$er_1 = \frac{|\omega_{m1} - \omega|}{\omega} 100\% \qquad (2)$$

Since $\omega_{m1} = n_1 \omega_{lim}$ then,

$$er_1(\omega, n_1) = \left(n_1 \frac{\omega_{lim}}{\omega} - 1 \right) 100\% \qquad (3)$$

Relative error er_1 maximised its value at the left of each interval when ratio ω_{lim}/ω was maximum. For high speeds (when n_1 exceeded 1), ratio ω_{lim}/ω was the inverse of $n_1 - 1$, as can be seen by analysing the upper part of Figure 3. For instance, if $n_1 = 3$, maximum ratio would have been $\omega_{lim}/\omega = 1/2$, so maximum relative error reached 50%. For low speeds the ratio was $\omega_{lim}/\omega = (1/n_1) + 1$. For instance, if $n_1 = 1/2$, then $\omega_{lim}/\omega = 3$, so maximum relative error reached 50% again. Previous analysis of ratio ω_{lim}/ω produced the expression for the maximum relative error shown in Equation 4:

$$Er1_{Max}(n_1) = \begin{cases} n_1 \times 100\% & , n_1 \le 1 \\ \dfrac{1}{n_1 - 1} \times 100\% & , n_1 \ge 2 \end{cases} \qquad (4)$$

The relative error in Equation 4 matched the maximum relative error for the traditional fixed-time algorithm if $\omega \ge \omega_{lim}$ and $dt = ts$. $Er1_{Max}$ matched the maximum relative error for the traditional fixed-space algorithm, when $\omega < \omega_{lim}$ and $dt = ts$. The proposed algorithm equated its measurement of quality with relative error from traditional algorithms.

Optimising maximum relative error

The algorithm's optimum minimised error Er_{Max}, such minimisation coming from an observation about where maximum error happened; according to the result shown in Figure 3, such maximum error took place at the left end from each interval. If the estimations shifted down one place, for example from n_1 equal to 3 to a new n_2 equal to 2, or from n_1 = ½ to n_2 = ⅓, then relative error decreased. This decrease occured due to the maximum difference between ω and the new ω_{m2} taking place at the right of each interval, instead of at the left.

The new estimation ($\omega_{m2} = n_2\omega_{lim}$) requires n_2 as given in Eq. 5.

$$n_2 = \begin{cases} \dfrac{1}{\dfrac{1}{n_1}+1} & ,\omega < \omega_{\lim} \\ n_1 - 1 & ,\omega \geq \omega_{\lim} \end{cases} \tag{5}$$

Nep_1 and Ndt_1 provide an alternate means for calculating n_2, as shown in Equation 6:

$$n_2 = \begin{cases} \dfrac{Nep_1}{Ndt_1 + 1} & ,\omega < \omega_{\lim} \\ \dfrac{Nep_1 - 1}{Ndt_1} & ,\omega \geq \omega_{\lim} \end{cases} \tag{6}$$

The relative error for estimation ω_{m2} is shown in Equation 7:

$$er_2(\omega, n_2) = \left(1 - n_2\frac{\omega_{\lim}}{\omega}\right)100\% \tag{7}$$

According to Equation 7, and for high speeds, every integer ratio ω/ω_{lim} = n_2 + 1 maximised relative error. For instance if n_2 equal to 2, then ω_{lim}/ω was 1/3, so the maximum relative error reached 33.3%. For low speeds, integer ratio ω/ω_{lim} was the inverse of $(1/n_2)$ + 1. So, for instance, if n_2 = ½, fraction ω_{lim}/ω = 3. As a result, maximum error reached 33.3% again. Error in terms of n_2 is shown in Equation 8. This equation defined a supremum and not a maximum, because the right end of each interval was open, as shown in Figure 3:

$$Er2_{Sup}(n_2) = \begin{cases} n_2 \times 100\% & ,n_2 < 1 \\ \dfrac{1}{n_2+1} \times 100\% & ,n_2 \geq 1 \end{cases} \tag{8}$$

Computing estimation ω_{m1} and ω_{m2} required the actual value of ω to classify a speed as low or high (Eq. 5 and 6). The value of the speed was unknown, although there were two approaches for determining whether speed passed speed limit, ω_{lim}. Both approaches used an indirect measurement. In the first approach, whereas low speeds had the time between consecutive *Iep's* exceeding *dt*, that time did not exceed *dt* for high speeds. The second approach counted n_1. Whereas that number exceeded $n_1 \geq 2$ for high speeds, for low speeds it did not.

Estimation ω_{m2} reduced maximum error by up to 50% with estimation ω_{m1}, mainly for speeds close to ω_{lim}. On the other hand, $Er1_{Max}$ and $Er2_{Sup}$ became closer and closer as speed went farther from ω_{lim}, or, in other words, when n_1 and n_2 went either to zero or infinity.

Any estimation greater than ω_{m1} generated errors larger than $Er1_{Max}$; any estimation under ω_{m2} resulted in errors over $Er2_{Sup}$. So if there was a way to reduce maximum error per interval, the resultant estimation would have fallen between ω_{m1} and ω_{m2}. See, for instance that estimation ω_{m3} equalled $n_3\omega_{lim}$ in Figure 4. The work then involved finding a value for n_3 that minimised error in region $\omega_{m2} \leq \omega_{m3} \leq \omega_{m1}$:

$$er_L = \frac{n_3 - n_2}{n_2}100\% = \left(\frac{n_3}{n_2} - 1\right)100\% \tag{9}$$

$$er_R = \frac{n_1 - n_3}{n_1}100\% = \left(1 - \frac{n_3}{n_1}\right)100\% \tag{10}$$

The minimum relative error for ω_{m3} could have been on the right or left side at each interval. If estimation ω_{m3} decreased, relative error on the left also decreased. As a result, relative error on the right increased. Thus, errors on the left opposed errors on the right and optimal value levelled the error on both sides, as shown by Equations 9, 10 and 11:

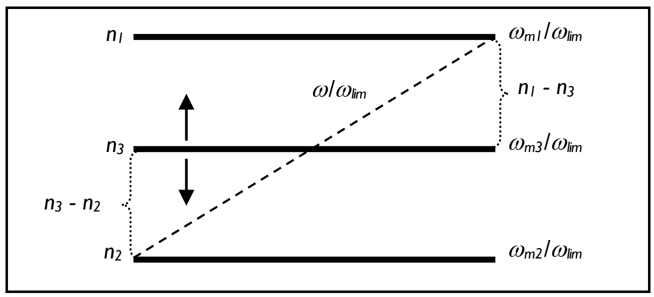

Figure 4. Finding the best estimation, n3

Error er_L reached zero when $n_3 = n_2$ in Equation 9, and according to the representation in Figure 4 the maximum occurred when n_3 equalled n_1. Equation 10 showed that er_R reached zero when n_3 equalled n_1, and the maximum occurred when $n_3 = n_2$. Since er_R and er_L were linear, then the minimum from combining er_R and er_L occurred when er_L equalled er_R, as shown in Equation 11. The value of n_3 in Equation 11 defined optimal estimation for ω_{m3}: the harmonic mean between ω_{m1} and ω_{m2}. No other value of n_i produced a smaller error:

$$\frac{n_3 - n_2}{n_2} = \frac{n_1 - n_3}{n_1} \quad \rightarrow \quad n_3 = \frac{2}{\dfrac{1}{n_1} + \dfrac{1}{n_2}} \tag{11}$$

Error $Er3_{Max}$ was defined in terms of n_3 in Eq. 12, and in terms of n_1 in Eq. 13:

$$Er3_{Max}(n_3) = \begin{cases} \dfrac{n_3}{2} \times 100\% & ,n3 < 1 \\ \dfrac{n_3 - \lfloor n_3 \rfloor}{\lfloor n_3 \rfloor} \times 100\% & ,n_3 > 1 \end{cases} \tag{12}$$

If n_1 = 1 or 2 in Equation 13 then $Er3_{Max}$ equalled one third of the error given by Equation 4. On the other hand, when $n_1 \to \infty$ or $n_1 \to 1/\infty$ then error $Er3_{Max}$ went below half $Er1_{Max}$. Therefore, this third estimation ω_{m3} in fact optimised error by setting the smallest possible bound for the relative error at each interval.

$$Er3_{Max}(n_1)=\begin{cases}\dfrac{1}{1+\dfrac{2}{n_1}}\times 100\% & ,n_1\le 1\\[2ex] \dfrac{1}{2n_1-1}\times 100\% & ,n_1\ge 2\end{cases} \tag{13}$$

The bounds for relative error were calculated for constant speed and also applied to variable speeds, if acceleration did not go beyond a given bound. The method for finding the maximum acceleration began by supposing constant acceleration and high speeds. Under these assumptions, and according to the result in Equation 1, any estimation ω_{mi} updated its value at most every 2*dt*, so maximum acceleration had an increase in speed equal to ω_{lim} in an interval of time equal to 2*dt*. The same analysis could be made for speeds under ω_{lim}, as shown in Equation 14:

$$a_{\max}=\begin{cases}\dfrac{\omega_{\lim}}{2dt}\dfrac{1}{Ndt(Ndt+1)},\left(rev/_{s^2}\right) & \omega<\omega_{\lim}\\[2ex] \dfrac{\omega_{\lim}}{2dt},\left(rev/_{s^2}\right) & \omega\ge\omega_{\lim}\end{cases} \tag{14}$$

The solid lines in Figure 5 illustrate estimations ω_{mi}, and the dashed line represents the actual speed. Estimation ω_{m1} was always greater than ω_{m3}, and ω_{m3} was always greater than ω_{m2}. The simulation parameters in Figure 5 were *dt* = 1 ms, *pul* = 10,000 and *ts* = 0.1 μs. These values made ω_{lim} = 0.1 rev/s, so maximum acceleration was a_{max} = 50 rev/s^2. The ramp with positive acceleration had 8.33 rev/s^2 constant acceleration whereas negative acceleration was -12.5 rev/s^2.

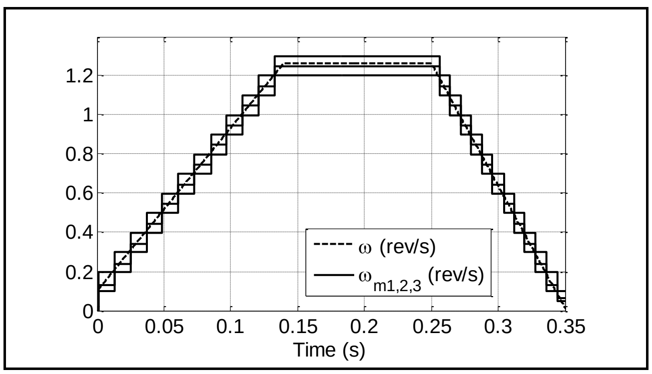

Figure 5. Estimation ω_{mi} for variable speed

The estimation technique tested in this section was also used as an incremental encoder, so the results appeared to be limited to measuring angular speed, even though this technique can be equally applied for estimating any variable's rate of change. There was only a single restriction on the data: sampling rate had to be constant. This restriction, along with the discrete nature of the measurements, generated oscillations on traditional estimation algorithms, even when the rate of change remained constant; however, those oscillations disappeared when using the proposed algorithm.

Experimental results

In this section, the algorithm generated estimations using real signals coming from an encoder. The scheme in Figure 6 illustrates an encoder coupled to a shaft, where the system aimed at estimating the shaft's angular speed. The encoder's output fluctuated from zero to five volts for low and high levels, respectively. A data acquisition card captured this voltage signal, and a computer recorded its values for later use. Data acquisition card PCI-6024E had 100 μs *ts* sampling time. The 160-slot encoder and the DC motor (Minertia 6GFMED) are shown in the lower part of Figure 8.

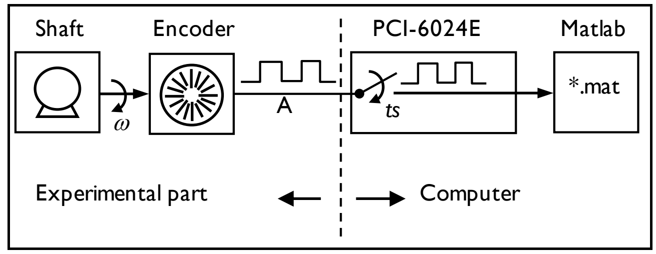

Figure 6. Experimental setup and signal processing scheme

The algorithm's verification required an actual value of ω to compute relative error. Instead of using a highly accurate sensor for obtaining the aforementioned value, the proposed algorithm found estimations using the best parameters for the smallest feasible error. For instance, ideal parameters were *pul* = 160 and *dt* = 2 s. These parameters guaranteed a boundary for relative error = 0.07%, when ω = 2 rev/s, as shown in Equation 12. This error decreased to 0.0065%, if ω equalled 24 rev/s.

The first experiment had the next parameters: *pul* = 160 and 1 ms for *dt*, *dt* thus overcoming *ts* by a factor of ten. In order to capture the data, the motor ran at constant voltage; a computer then saved data for ten seconds starting only when the motor entered its stationary stage. The algorithm obtained estimations using the recorded data as input. The same steps were repeated every 0.5 V for constant voltages from 3 to 24 V.

The solid line in Figure 7 represents the bound for relative error per interval, as indicated in Equations 4, 8 and 13. The dots represent the experimental values for relative errors: e_{r1} for ω_{m1}, e_{r2} for ω_{m2} and e_{r3} for ω_{m3}. Results in Figure 7 agreed with the analysis given in the previous section. Relative errors always fell within bounds, and estimation ω_{m3} consistently had minimum relative error.

Other experiments have used *dt* equal to 2, 5, 10, 20, 50 and 100 ms. If *dt* increased ω_{lim} decreased. For instance if *dt* were greater than 5 ms, ω_{lim} went below experimental shaft speed. The greater the *dt*, the larger the n_i and, as a result, relative error decreased, but updating time increased. So the relationship between *dt* and relative error became a trade-off problem, the solution of which depended on its application.

Relative error also decreased by making ω_{lim} higher than any experimental speed; this change in ω_{lim} value required the use of an encoder having a lower number of slots, or the definition of a smaller *dt*. Thus, instead of changing the encoder to run this experiment, only one of every *k* pulses of the encoder was recorded. The experiment used *k* = 40 and *dt* = 1 ms. So ω_{lim} = 250 rev/s, and since all speeds fell under 30 rev/s, maximum relative error fell under 10%, as was experimentally proven.

The algorithm was tested on-line using an embedded system for testing the algorithm's application in an electronical device (i.e. dsPIC33FJ128MC802). dsPIC output was estimated as ω_{m3}, closing a control loop in Simulink, as shown in the upper part of Figure 8, where *u* was the actuating signal. The filter made the system slow enough to have *ts* = 1 ms and *dt* = 40 ms.

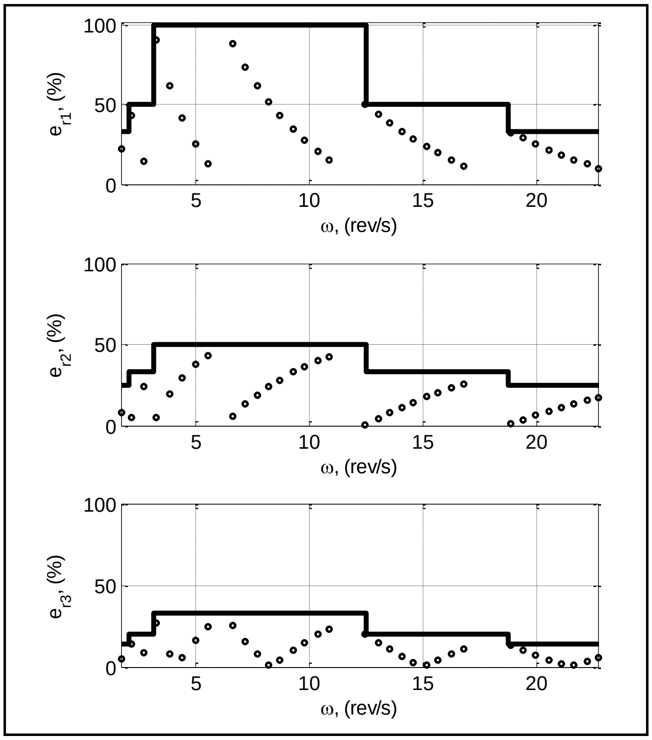

Figure 7. Relative error for *pul* = 160, *dt* = 1 ms, and *ts* = 100 μs

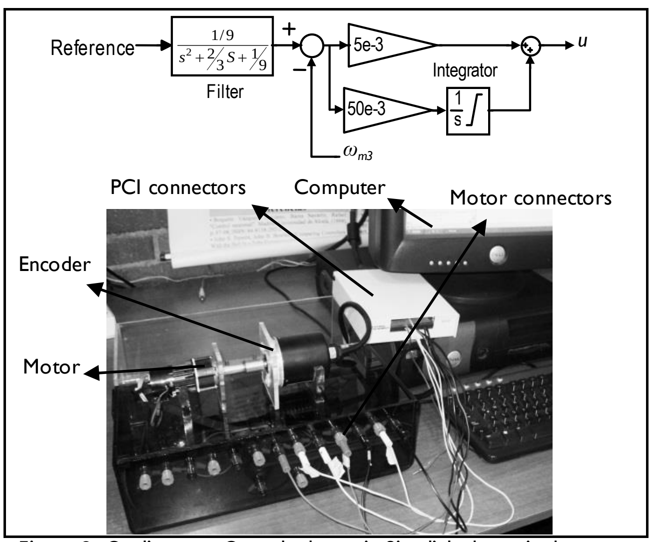

Figure 8. On-line test. Control scheme in Simulink shown in the upper part and experimental setup in the lower part

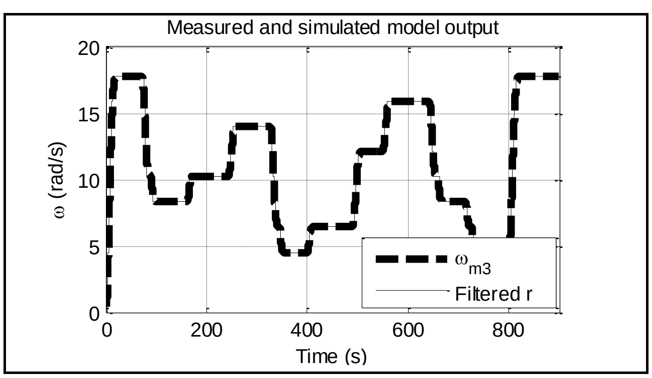

Figure 9. Identification result.

Identification involved using the filtered reference as input and ω_{m3} as ouput. The identification result is shown in Figure 9. The model for the output–input ratio, H(S), was H(*S*) = 1/((1+1.512*S*)(1+1.506*S*)). The model fit 94.41% of the data after 20 iterations, thereby showing the quality of the data and control loop.

Conclusions

A new algorithm for estimating speed has been proposed proven and experimentally tested (using incremental encoders). One of the the algorithm's advantages was that it had constant output when the speed was constant, contrary to fixed-time and fixed-space algorithms producing two values.

Relative error for the stationary case was computed and compared to fixed-time and fixed-space algorithms, showing that maximum relative error per interval was the same, where an interval was defined as the range of speeds havin the same value as estimation. A modification of the algorithm has been proposed, aimed at reducing maximum relative error by interval. It was proved that the optimal value was the harmonic mean between estimation ω_{m1} and ω_{m2}. The maximum relative error was always smaller than half of that produced by fixed-time and fixed-space algorithms.

Even when the mathematical proof was made for constant speed, it was shown that the estimation remained valid for acceleration smaller than a given maximum, depending exclusively on *pul* and *dt* for high speed and *pul*, *dt* and *Ndt* for lower speed.

Observing experimental data showed the effect of changing parameters *dt* and *pul* in the encoder; such parameter changes reduced the relative error of the estimation, and could be done using three approaches: increasing *pul* and *dt* at their highest possible values, reducing *pul* and *dt* and decreasing *ts*.

Some problems of interest remain. For example, it appears that an adaptive algorithm for dynamically changing the value of *dt*, *ts* and *k* could boost algorithm performance, because making ω_{lim} greater or smaller than current speed would reduce error in the estimation. Implementing the algorithm in an embedded system, faster than dsPIC, including additional aspects such as rotation sense (also called sign) would make the algorithm ideal for use in industrial applications.

References

Boggarpu, N., and Kavanagh, R., New Learning Algorithm for High-Quality Speed Measurement and Control When Using Low-Cost Optical Encoders., IEEE Trans. on Instrumentation and Measurement, Vol. 59, No. 3, Mar., 2010, pp. 565–574.

D'arco, S., Piegari, L., and Rivo, R., An Optimized Algorithm for Speed Estimation Method for Motor Drives., Proc. Symposium on Diagnostics for Electric Machines, Power Electronics and Drivers, Atlanta, GA, USA, Aug., 2003, pp. 76–80.

Hachiya, K., and Ohmae, T., Digital speed control system for a motor using two speed detection methods of an incremental encoder., Proc. European Conference Power Electronics and Applications, Aalborg Denmark, 2007, pp. 1–10.

Liu, G., On speed estimation using position measurements., Proc. American Control Conference, Anchorage, Alaska, USA, 2002, pp. 1115–1120.

Lygouras, J., Pachidis, T., Tarchanidis, K., and Kodogiannis, V., Adaptive High-Performance Speed Evaluation Based on a High-Resolution Time-to-Digital Converter., IEEE Trans. on Instrumentation and Measurement, Vol. 57, No. 9, Sep., 2008, pp. 2035–2043.

Merry, R., Molengraft, R., and Steinbuch, M., Error modeling and improved position estimation for optical incremental encoders by means of time stamping., Proc. 2007 American Control Conference, New York City, USA, 2007, pp. 3570–3575.

Petrella, R., and Tursini, M., An Embedded System for Position and Speed Measurement Adopting Incremental Encoders., IEEE Trans. on Industry Applications, Vol. 44, No. 5, Sep., 2008, pp. 1436–1444.

Se-Han, L., Lasky, A., and Velinsky, S., Improved Speed Estimation for Low -Speed and Transient Regimes Using Low -Resolution Encoders., IEEE/ASME Trans. on Mechatronics, Vol. 9, No. 3, Sep., 2004, pp. 553-560.

Su, Y. X., Zheng, C. H., Sun, D., and Duan, B. Y., A Simple Nonlinear Speed Estimator for High-Performance Motion Control., IEEE Trans. on Industrial Electronics, Vol. 52, No. 4, Aug., 2005, pp. 1161-1169.

Tsuji, T., Hashimoto, T., Kobayashi, H., Mizuochi, M., and Ohnishi, K., A Wide-Range Speed Measurement Method for Motion Control., IEEE Trans. on Industrial Electronics, Vol. 56, No. 2, Feb., 2009, pp. 510–519.