High Performance Velocity Estimation for Controllers with Short Processing Time by FPGA

Manuel Nandayapa* Student Member, Chowarit Mitsantisuk* Member
Kiyoshi Ohishi* Senior Member

(Manuscript received April 29, 2011, revised Sep. 27, 2011)

In motion control, the number of controllers based on field programmable gate arrays (FPGAs) has increased in recent years owing to the inherent short sampling time. At present, the most commonly implemented velocity estimation methods are the M and T methods. The proposed velocity estimation method reduces the calculation time by eliminating the division operation and prevents reduction in resolution owing to a short sampling time. In addition, the proposed method shows a high performance across a wide range of velocities. Program implementation in FPGAs has a low computational cost and a high calculation speed. The conventional methods present some difficulties in programming using FPGAs owing to the divider implementation. The validity of the proposed method was tested using a velocity controller based on a disturbance observer.

Keywords: velocity estimation, disturbance observer, motion control, FPGA, robotic, ball screw

1. Introduction

In the past few decades, the development of mechatronic devices and computer science has provided impetus to progress in motion control technology. In addition, field programmable gate arrays (FPGAs) are being used widely, owing to their short sampling time. Combining programmable devices with mechanical systems has been the key to making advances in the motion control technology and mechatronic systems. Almost twenty years ago, controllers based on the disturbance observer were used in sensorless force control applications. Since then, motion control based on the disturbance observer has been applied to controllers using programming platforms such as real-time Linux (RTLinux). In recent years, controllers based on the FPGA have shown an acceptable level of performance because of the processing speed. The short sampling time has led to an increase in the accuracy, efficiency, and speed of FPGA algorithms. Thus, the algorithms have low delay, similar to continuous-time algorithms. Motor velocity is used as a variable in motion control systems. However, velocity sensors are not used widely in these systems, and hence, the velocity must be estimated by some other method. The literature describes many methods for estimating the velocity, several of which have been implemented in systems with long sampling and processing times. For instance, methods based on the Kalman filter or computational algorithms have a long processing time. Conventional velocity estimation methods such as the M and T methods involve the arithmetic division of distance by a given interval of time. This time interval is called the sampling time, and it is often equal to the processing time. For example, the processing time in computers is of the order of microseconds and the processing and sampling time are the same. However, conventional methods have also been programmed into FPGAs with minor adjustments in the use of fixed-point arithmetic operations. Similarly, the hybrid method has been implemented.

The accuracy of the M method depends on the difference between positions over the interval of one sampling time. In contrast, the accuracy of the T method depends on the number of clock pulses in a single encoder pulse. Thus, the M method achieves high performance at fast velocities; the T method, slow velocities.

The method proposed here is based on the M method, but it is reduced to a simple subtraction of positions. Programming with the minimum of arithmetic operations reduces the number of resources required and thus increases the processing speed. The resources and speed are the most important features used to measure an algorithm’s effectiveness. Therefore, the proposed method is suitable for programming in FPGAs.

The remainder of this paper is organized as follows. Section 2 introduces the conventional methods used for velocity estimation, and the proposed method is described in section 3. Section 4 describes velocity control based on the disturbance observer, and in section 5, the programming of FPGAs is discussed. Section 6 presents the simulation and experimental results. Finally, our conclusions are discussed in section 7.

2. Conventional Velocity Estimation Methods

The M method is based on velocity estimation using Euler’s method, and the approximation is actually a backward finite difference. In other words, it estimates the difference
between the present position and the position in a previous sampling time, and this value is then divided by the sampling time. This is the distance over time. In order to obtain the velocity in rad/s, it is calculated using Eq. (1).

\[
\dot{\omega}(k) = \frac{2\pi(\theta(k) - \theta(k-1))}{(eP)(Ts)} \tag{1}
\]

Figure 1 shows Ts as the sampling time for the velocity calculation. \(\theta(k)\) and \(\theta(k-1)\) are the present and previous positions, respectively, and \(eP\) is the encoder resolution (counts per revolution). In the M method, the calculation is executed for each Ts.

In contrast, the T method is calculated from the integration of the number of clock pulses \(nP\) in the duration of a pulse encoder. The distance is the encoder pulse, and the time is the integration of time using the clock pulses. In this case, it can also be considered as distance divided by time. However, the implementation in the FPGA is complicated given that the divider implementation is difficult and it uses more resources. The calculation is executed for every encoder pulse. In order to obtain the velocity in rad/s, it is calculated using Eq. (2),

\[
\dot{\omega}(k) = \frac{2\pi(f_c)}{(eP)(nP)} \tag{2}
\]

where \(f_c\), \(eP\), and \(nP\) are the clock frequency, encoder resolution, and number of clock pulses in an encoder pulse, respectively.

Figure 2 shows a schematic of velocity estimation by the T method, where the calculation is carried out for every encoder pulse.

Conventional methods normally use a low pass filter (LPF) after the velocity calculation. In the literature, these methods use a first-order LPF.

3. Proposed Velocity Estimation Method

The proposed method uses a short sampling time and the fast clock signal (of the order of nanoseconds) of FPGAs. The proposed method is based on the M method, and the positions used to estimate the distance are almost the same. The difference between the methods is the processing time, with the M method using Ts and the proposed method using a shorter time interval \(Tp\) (see Fig. 3). This means that the calculation speed of the proposed method is faster than that of the M method. The number of calculations is determined by the \(m\) value, and the calculation is executed for each \(Tp\). By using a faster calculation, it is possible to obtain many velocity values in a single sampling time used in the conventional methods. The proposed method uses the Eq. (3), which involves the resolution of the encoder \(eP\), the processing time \(Tp\), and the number of processing times \(m\) that are necessary between the present position \(\theta(k)\) and the previous position \(\theta(k-m)\).

\[
\dot{\omega}(k) = \frac{2\pi(2^m)(\theta(k) - \theta(k-m))}{(eP)(Tp)(m)} \tag{3}
\]

3.1 Selection of \(m\) Value

It is possible to execute algorithms in the FPGA at different processing times, using a PLL (Phase-Locked Loop). FPGAs offer feature-rich PLLs that provide robust clock management capabilities and synthesis for device clock management, external system clock management, and high-speed I/O pin interfaces. The PLL can be configured to obtain a diverse range of clock signals, which are multiples of the clock bases. In our case, this means multiples of 20 ns.

The selection of the \(m\) value depends on the processing time and the encoder resolution. First, we select our \(Tp\) value, which depends on the algorithm’s processing time and the FPGA base clock. It is better if we have a short processing time, and the number of arithmetic operations utilized in the algorithm impacts on the processing time. If \(Tp\) is a short time, the \(m\) value and accuracy increase. Secondly, the \(m\) value is multiplied by \(Tp\) to obtain the pulse interval for our velocity calculation; \((Tp)\) is similar to \(Ts\) in the M method. The pulse interval depends on the encoder resolution, but is normally around 100 μs on industrial applications. Thirdly, \(eP\) depends on the encoder in our mechanism. It is possible to use different values of \(eP\). Finally, the \(n\) value depends on the difference between 16 bits and the encoder resolution used.
The value of \( n \) of the encoder used. The multiplication of 2 to the \( n \)th power involves no computational complexity, because it simply shifts a bit. The sign of the \( n \)-value indicates whether the bit shift is to the right or left.

The encoder resolution is important because it means that the time between each pulse generated is not very long. If the resolution is low, the time \( (T_p/m) \) is not long enough to acquire encoder pulses; the same problem is encountered in the M method. Moreover, if the resolution is high, the FPGA will be able to acquire more pulses and the accuracy will increase.

When the processing time is long and the \( m \) value is very small, the proposed method is similar to the M method. The proposed method has a better performance, and is thus more suitable, if the processing time is short.

Considering the aforementioned, Eq. (4) is proposed only for reduction purposes. Equation (3) shows the velocity calculation to get \( \text{rad/s} \), and it considers a short processing time. However, Eq. (3) includes many arithmetic operations. In Eq. (4), the parameters \( n, m, eP, \) and \( T_p \) are chosen to get a closer approximation to 1. Thus, Eq. (3) is reduced to a single subtraction of positions (Eq. (5)). So, by applying Eq. (4) to Eq. (3), the proposed method is reduced to Eq. (5).

\[
\hat{\omega}(k) = \theta(k) - \theta(k - m) \tag{5}
\]

Obtaining the velocity calculated using a subtraction of positions is very appropriate, and it is easy to program in the FPGA.

After the velocity calculation, a LPF is applied in order to get an average. The cut-off frequency is 3000 \( \text{rad/s} \). Considering that the LPF is a Butterworth filter, the transformation to the discrete domain is shown in Eq. (6),

\[
G_f(Z) = \frac{a}{1 - bZ^{-1}} \tag{6}
\]

Transforming Eq. (5) from time to a discrete domain,

\[
G_a(Z) = 1 - Z^{-m} \tag{7}
\]

Using Eqs. (6) and (7), the proposed method is constituted as shown in Eq. (8).

\[
G_{\hat{\omega}}(Z) = (1 - Z^{-m})(2^n) \frac{a}{1 - bZ^{-1}} \tag{8}
\]

The transfer function expressed in Eq. (8) is the complete velocity estimation method that is programmed in the FPGA.

### 4. Velocity Control based on Disturbance Observer

Velocity control based on the disturbance observer is a robust acceleration control \([9,10]\). The disturbance observer is an observer that estimates the disturbance torque added to the actuator. Torque disturbance is composed of friction torque, external torque, Coriolis force, influence of gravity, and parameter variation. However, if the disturbance observer is used in an inner loop with a current feedback, compensation is achieved.

The disturbance torque is obtained from the motor current, torque constant, inertia motor, and the second derivative of the position signal, as shown in Eq. (9). The first term \( K_m I_m \) in Eq. (9) is from the input data, and the second term \( J\ddot{\theta} \) is from output data.

\[
\tau_{\text{dis}} = K_m I_m - J\ddot{\theta} \tag{9}
\]
Velocity Estimation with Short Processing Time by FPGA

Manuel Nandayapa et al.

Fig. 6. Block diagram of velocity control with proposed velocity estimation method

Table 1. Velocity controller parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{tn}$</td>
<td>0.234</td>
<td>$(N-m)/A_{rms}$</td>
</tr>
<tr>
<td>$J_n$</td>
<td>1.89004E-05</td>
<td>kg-m$^2$</td>
</tr>
<tr>
<td>$\theta_{in}$</td>
<td>2000</td>
<td>rad/sec</td>
</tr>
<tr>
<td>$P$</td>
<td>250</td>
<td></td>
</tr>
</tbody>
</table>

where $K_{tn}$ is the nominal force coefficient and $J_n$ is the nominal inertia motor. In order to suppress the high-frequency noise, a LPF is applied. The LPF’s cut-off frequency determines the bandwidth of the estimated torque. The disturbance torque estimated from Eq. (10) can be represented as, Eq. (9).

$$\hat{\tau}_{dis} = g_{dis}(K_{tn}I_a + g_{dis}\dot{\theta}) - g_{dis}J\ddot{\theta} \cdots \cdots (10)$$

where $g_{dis}$ denotes the cut-off frequency of the LPF and the Laplace operator $s$ denotes a derivative calculation. The velocity controller is a proportional (P) controller (see Fig. 6).

Table 1 gives the parameters used in the velocity controller. Figure 7 shows the robust acceleration control that is applied to a velocity controller.

5. FPGA Programming

As mentioned above, in order to carry out the programming of the proposed method, short sampling times are required. Therefore, an Altera Cyclone III FPGA, with a clock frequency of 50 MHz, is used. The block diagram in Fig. 8 shows the implementation of the proposed method into the FPGA. The position is obtained from channels A and B of the encoder, by passing through a digital filter, quadrature encoder, and a pulse counter.

The FPGA uses a base clock of 20 ns, but in the implementation of the system a processing clock of 500 ns by PLL is utilized. From digital filter to pulse encoder counter section, and Digital to Analog (DA) Converter Serial Peripheral Interface (SPI) protocol, the base clock 20 ns is used, while the blocks of the velocity estimation, disturbance observer, and controller are being processed with the processing clock of 500 ns. The processing clock helps to get better accuracy in the coefficient calculation used in the Infinite Impulse Response (IIR) digital filters. If the ratio of the sampling rate and processing rate is very high, the coefficient needs more bits. Moreover, the mechanical system does not respond to high variations, and clock processing is sufficient for our purpose.

The use of FPGA means that the proposed method must use fixed-point operations; this is one of the programming difficulties. In this paper, the fixed-point number performs the control calculations by taking into consideration the maximum and minimum values of the variables measured. The fixed-point number is a representation of a decimal number, and is essentially an integer. Figure 9 shows the number of bits used for the velocity calculation.

In Table 2, the resources used in the FPGA for each method are shown. It can be seen that the proposed method uses the blocks of the velocity estimation, disturbance observer, and controller are being processed with the processing clock of 500 ns. The processing clock helps to get better accuracy in the coefficient calculation used in the Infinite Impulse Response (IIR) digital filters. If the ratio of the sampling rate and processing rate is very high, the coefficient needs more bits. Moreover, the mechanical system does not respond to high variations, and clock processing is sufficient for our purpose.

The use of FPGA means that the proposed method must use fixed-point operations; this is one of the programming difficulties. In this paper, the fixed-point number performs the control calculations by taking into consideration the maximum and minimum values of the variables measured. The fixed-point number is a representation of a decimal number, and is essentially an integer. Figure 9 shows the number of bits used for the velocity calculation.

In Table 2, the resources used in the FPGA for each method are shown. It can be seen that the proposed method uses
fewer resources than the conventional methods. The use of fewer multipliers also indicates that the speed is higher. The delay from the calculation varies due to the processing time of each method.

### 6. Simulation and Experimental Results

In order to prove the wide range of the proposed method, a simulation was performed with a fast velocity of 100 rad/s, a medium velocity of 24 rad/s, and a slow velocity of 10 rad/s. High performance was recorded for the T method at slow velocity, and for the M method at high velocity. Conversely, the M method performs badly at slow velocity, and the T method at high velocity. All three methods present acceptable performances at medium velocity.

To perform the experiments, a setup was implemented using a ball-screw mechanical system. The ball-screw system consists of an amplifier (SGDH-02BE, Yasakawa Company), and a ball-screw (SKR-4320, THK Company) with a SGMAH-02B-1A4E motor. The FPGA model is EP3C120F780C8N, and the DA Converter is a UTL-012 from HuMANDATA Ltd. The experimental setup is shown in Fig. 10.

In order to test the velocity estimation method, a sine wave was applied as the input signal. The sine wave was generated by the COordinate Rotation DIgital Computer (CORDIC) algorithm.

The sine wave is considered adequate because it has different velocities at different points. At the beginning, the velocity command was set at the zero value. On the other hand, at the peak of the velocity it changes in direction. Let us analyze the performance of the velocity estimation methods for the conditions mentioned.

Figure 11 shows the performance of the T and M conventional methods compared with the proposed method. The T method is affected by two factors. Firstly, at high velocity it gives an error. Secondly, when the velocity is close to zero, the divider implementation on the FPGA affects the accuracy due to an arithmetic fixed point. In the M method, the noise affects the accuracy and the bandwidth, because it is produced by a long sampling time in comparison with the proposed method. The difference between the three methods close to 0 rad/s is very small because the velocities change very quickly. The advantages of the proposed method are confirmed in the following experiments.

Step responses were obtained for data analysis. The same parameters as in the previous simulation were applied: 10, 24, and 100 rad/s. The experimental results are shown in Figs. 12, 13 and 14, respectively. Figure 12 shows that the M method has low performance, and the T and proposed methods have high performances, for low velocity. However, the fixed point error in the T method is not present at this velocity. Figure 13 shows that the three methods give similar results for medium velocity. Figure 14 shows that for high velocity, the T method has low performance and the M and proposed methods have high performances.

In Tables 3 and 4 the maximum, minimum, and mean values obtained from the simulation and experimental results are
Velocity Estimation with Short Processing Time by FPGA

Manuel Nandayapa et al.

Fig. 13. Experimental results of step response of velocity controller with amplitude at 24 rad/s

Fig. 14. Experimental results of step response of velocity controller with amplitude at 100 rad/s

Fig. 15. Simulation results of three methods with constant velocity using low, medium, and high values presented. The three cases show that the proposed method presents less deviation from the constant value. The comparison is also shown in Figures 15 and 16. Figure 15 shows the simulation, and Fig. 16 shows the experimental results.

Fig. 16. Experimental results of three methods with constant velocity using low, medium, and high values

Table 3. Maximum, Minimum and Mean values of the three methods with constant velocity from simulation

<table>
<thead>
<tr>
<th>Method</th>
<th>Measure</th>
<th>10 rad/s</th>
<th>24 rad/s</th>
<th>100 rad/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>Max</td>
<td>9.9354</td>
<td>24.0242</td>
<td>101.9195</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>9.9354</td>
<td>24.0242</td>
<td>101.9195</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>9.9354</td>
<td>24.0242</td>
<td>101.9195</td>
</tr>
<tr>
<td>M</td>
<td>Max</td>
<td>10.1658</td>
<td>23.9684</td>
<td>100.0740</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>9.8228</td>
<td>23.9684</td>
<td>99.7088</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>9.9870</td>
<td>23.9684</td>
<td>99.8684</td>
</tr>
<tr>
<td>Proposed</td>
<td>Max</td>
<td>9.9991</td>
<td>23.9979</td>
<td>99.9919</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>9.9991</td>
<td>23.9979</td>
<td>99.9919</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>9.9991</td>
<td>23.9979</td>
<td>99.9919</td>
</tr>
</tbody>
</table>

Table 4. Maximum, minimum, and mean values of the three methods with constant velocity from experiments

<table>
<thead>
<tr>
<th>Method</th>
<th>Measure</th>
<th>10 rad/s</th>
<th>24 rad/s</th>
<th>100 rad/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>Max</td>
<td>10.4194</td>
<td>24.6125</td>
<td>102.9046</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>9.5640</td>
<td>23.6184</td>
<td>101.0301</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>9.9878</td>
<td>24.1007</td>
<td>102.2760</td>
</tr>
<tr>
<td>M</td>
<td>Max</td>
<td>10.7851</td>
<td>24.7187</td>
<td>100.8896</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>9.2815</td>
<td>23.2807</td>
<td>99.0462</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>9.9851</td>
<td>23.9638</td>
<td>99.8826</td>
</tr>
<tr>
<td>Proposed</td>
<td>Max</td>
<td>10.4039</td>
<td>24.4953</td>
<td>100.4066</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>9.5845</td>
<td>23.5587</td>
<td>99.0920</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>9.9973</td>
<td>23.9938</td>
<td>100.0058</td>
</tr>
</tbody>
</table>

7. Conclusion

In this paper, we have proposed a velocity estimation method that achieves a good response at low and high velocities. The short processing time of FPGAs is used to calculate the velocity with more detail than the conventional methods. The proposed method involves only a position subtraction, and the velocity is obtained in rad/s. By performing a velocity control experiment, we confirmed that the proposed method shows a high-velocity estimation performance with disturbance torque compensation by using the disturbance observer. Moreover, the proposed method is suitable for programming implementation in FPGAs.
Velocity Estimation with Short Processing Time by FPGA (Manuel Nandayapa et al.)

References


Manuel Nandayapa (Student Member) received a B.S. degree in Electronics Engineering from Institute of Technology of Tuxtla Gutierrez, Chiapas, Mexico in 1997, M.S. degree in Mechatronics Engineering from CENIDET, Morelos, Mexico in 2003. At the moment he is a Ph.D. student at Nagaoka University of Technology, Niigata, Japan. He is a student member of the IEEE Industrial Electronics Society and Robotics Automation Society, the Institute of Electrical Engineers of Japan (IEEE).

Chowarit Mitsantisuk (Member) received the B.S. degree in Electrical Engineering from Thammasat University, Pathumthani, Thailand, in 2003, and the M.S. degree in Electrical Engineering and D.Eng degree in Energy and Environmental Science from Nagaoka University of Technology, Japan, in 2007 and 2010, respectively. At present, he is a Postdoctoral Fellow of the Japan Society for the Promotion of Science (JSPS). His research interests include motion control, force control, human-robot interaction, sensor fusion and real-world haptics. Dr. Chowarit is a Member of the IEEE Industrial Electronics Society. He received the IECON’09 IEEE Student Scholarship Award from the IEEE Industrial Electronics Society.

Kiyoshi Ohishi (Senior Member) received the B.S., M.S. and Ph.D. degrees, all in electrical engineering, from Keio University, Yokohama, Japan, in 1981, 1983 and 1986, respectively. From 1986 to 1993, he was an Associate Professor with Osaka Institute of Technology, Osaka Japan. From 1993 to 2003 he was an Associate Professor with Nagaoka University of Technology, Niigata, Japan. Since August 2003 he has been Professor at the same university. He is an administration committee member of the IEEE Industrial Electronics Society, the Institute of Electrical Engineers of Japan (IEEE), the Japan Society of Mechanical Engineers (JSME), the Society of Instrument and Control Engineers (SICE), and Robotics Society of Japan (RSJ).