IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 29, 2018, accepted August 24, 2018, date of publication August 28, 2018, date of current version September 21, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2867510

A Novel Divisionless MT-Type Velocity Estimation Algorithm for Efficient FPGA Implementation

ALEŠ HACE, (Member, IEEE), AND MILAN ČURKOVIČ
Faculty of Electrical Engineering and Computer Science, University of Maribor, SI-2000 Maribor, Slovenia
Corresponding author: Aleš Hace (ales.hace@um.si)

This work was supported by the Slovenian National Science Foundation under Grant P2-0028.

ABSTRACT This paper deals with incremental encoder velocity estimation. More specifically, it addresses the computation issues of the well-known MT-method, used widely in advanced motion control applications. The MT-method calculus involves arithmetical division, which is a very unsuitable operation for digital implementation on real hardware, such as an FPGA. Thus, we propose a novel divisionless algorithm for velocity estimation. The proposed method possesses the advantages of both frequency count and period count methods as the well-known MT-method to produce reasonably accurate and smooth velocity signals in a wide speed range. However, its advantages in terms of simpler calculus make it significantly more suitable for a single-chip FPGA cost-effective solution. This paper presents the method and shows the results with simulation data and real encoder data acquired by practical digital circuit hardware. The shown results fully verify the proposed algorithm.

INDEX TERMS Incremental encoder, velocity estimation, MT-method, divisionless calculus, FPGA.

I. INTRODUCTION

Incremental position encoders have already been used for position measurement in digital control systems of servo drives for several decades. They generate a digital output signal of a pulse train while its shaft is rotating or moving linearly, by which we can implement not only position measurement, but it can also provide velocity information. However, due to the discrete nature of the incremental encoder operating as a pulse generator based measurement method, a straightforward approach for velocity estimation results in highly noisy data that cannot be applied directly in advanced motion applications with velocity feedback. Though the noise component can be filtered out, this approach causes a phase delay that is undesired in high-performance closed-loop control. Thus, there has been extensive research in order to provide accurate and delay-less smooth velocity information over a wide speed range from a digital incremental encoder.

A widely used common straightforward method for velocity estimation, which may be suitable for low performance motion control applications, is called the M-method. In the M-method, the encoder pulses, which normally occur asynchronously with sampling instants, are counted over a prescribed constant sampling time interval. However, especially in the low-speed region where only a small number of pulses are generated, low-resolution data generate considerable noise in this method due to the spatial quantization. The spatial quantization inherent in the operation of a shaft encoder limits severely the resolution and accuracy of any digital-differentiator-based velocity calculation or pulse-counting technique incorporated in a control or measurement application. Therefore, an alternative T-method can be utilized in which the time interval between two adjacent encoder pulses should be measured by counting clock pulses. The velocity information is then obtained by the reciprocal of the measured time interval, i.e. by arithmetic division. This method can perform well in the case of rare encoder pulses; however, the resolution may be poor at high speed. Ohmae [1] therefore proposed the so-called MT method for velocity estimation that combines both: The number of pulses that occur in a sampling period are counted and the time interval between boundary pulses in the sampling periods is provided as well. Then the velocity information is computed by division of the number of pulses over the time interval. Such method can perform well in a wide speed range, from low-speed to high speed. Analyses have been presented that illustrate a very high potential accuracy of the velocity measurement [2].

The MT-method is a high-performance velocity estimation approach with a hardware and a software part, which can easily be incorporated into digital processing systems with a fixed sample time. The computation algorithm of the MT-method (or a variant of this method) can easily be

implemented in real-time on DSP processors and, thus, the method is used widely in high-accuracy motion control applications since proposed, although it may experience some problems with practical incremental encoders due to their mechanical inaccuracies and other non-idealities of their hardware structure [3]–[10]. Therefore, such approach, which enables accurate and timely high-bandwidth digital velocity information, is, nowadays, incorporated into high performance mechatronic, robotic and precision production systems. However, it is worth mentioning that an arithmetic division is a much more demanding operation for processing hardware than, e.g., an arithmetic multiplication [11].

Digital programmable circuits such as FPGAs are, nowadays, used widely for many industrial applications [12], [13]. FPGAs offer possibilities for integrating complex hardware systems and their interface into a single FPGA chip and, thus, they are highly suitable for performing the high frequency counting and timing operations required for improved performance of incremental electronics, and yet can be easy to reprogram, which reduces the development time greatly [14]. Thus, common implementations of the advanced velocity measurement methods may conveniently combine an FPGA for the specific conditional circuit for counting encoder pulses and clock pulses, and a computation engine such as a DSP processor [4], [5], [8], [15], [16], which enables the execution of complex control algorithms in real-time. Thus, it is employed for implementation of the software part, i.e. the computation algorithm of the velocity estimation methods that inherently involve arithmetical division operation. However, though advanced DSP processors allow implementation with the arithmetic division with no significant effort in the design phase, the system design complexity may increase significantly in the case of advanced motion control applications with high sampling rates based on the platforms with an FPGA. Furthermore, though some performance-line DSP controllers can also incorporate high-frequency counters for encoder pulses and clock pulses, their conditional circuits lack a proper hardware support for advanced MT-type methods for velocity estimation [17]. On the other hand, FPGAs allow for economical implementation of complex circuitry, featuring parallel operation, low power consumption, and small occupied spaces. Besides that, they can also ensure high computational power. Furthermore, compared with the processor-based controller, the FPGA-based controller allows more accurate, as well as shorter sampling periods, which is a highly important issue for modern advanced motion control systems requiring an accurate wide-range velocity measurement [18], [19] such as, e.g., robotics and haptic interfaces [20]–[23]. Although FPGAs are improving in size as well as in performance, an excessive hardware resources demand related to the implementation of complex computation algorithms may prevent their further applications. Yet, the current technology development trends direct smaller design solutions, lowering consumption and costs.

The low cost of recent FPGAs and the capacity of these devices to implement complex circuitry have introduced the possibility of developing new high-performance practical applications [24]. However, the main obstacle for implementation of an MT-type velocity estimation method as a whole, i.e. including the computational part, into an FPGA, is the inclusion of arithmetical division. While being easy for a DSP computing system, the division operation is particularly difficult for FPGA-based embedded electronics to perform [25], [26], characterized with high computation latency and high resources' consumption [27], [28]. On the other hand, arithmetic multiplication can be implemented easily in an efficient way, since an FPGA can embed dedicated digital multipliers that recently evolved into more complex DSP blocks, eliminating the need for using FPGA distributed logic cells [24], [29]. Thus, there is a need to eliminate the division operation from the MT-type velocity estimation algorithm.

Zhu [30] has developed a solution for a velocity estimation that combines the M-method and the T-method. He showed that the arithmetic division calculus in the latter can be replaced by a sophisticated mechanism that is comprised of two counters and an accumulator. The developed solution is based on the Bresenham-like algorithm [31] that seeks for the solution incrementally from a known initial value, requiring only integer addition and subtraction. It yields a result that is mathematically equivalent to the computation of the reciprocal of the measured time interval between the adjacent encoder pulses. Thus, the proposed velocity estimation method, which shifts seamlessly between the M-method and the T-method, can be easily implemented on an FPGA. However, despite the simplicity of the method, it requires execution of the algorithm continuously in every clock tick that increases power and resources consumption of the digital circuit.

In this paper, we introduce a variant of the MT method algorithm, in which the arithmetic division is eliminated. Instead, the algorithm involves multiplication and addition that are the only required arithmetic operations in the real-time calculation. The minimum of arithmetic operations reduces the number of resources required and, thus, increases the processing speed. Since consumed resources and execution speed are the most important aspects of an algorithm used to measure its effectiveness, the proposed algorithm is suitable for implementation on an FPGA. It is given in a recursive form, thus stability and convergence are discussed theoretically and shown by numerical examples too. The main contribution of this paper is the proposed algorithm for the divisionless MT-type velocity estimation method, which is verified in simulations by generated data as well as by off-line processing of real data that have been obtained by a practical encoder.

The rest of the paper is organized as follows: In Section II firstly we present the basics of the well-known MT-method, that is continued with the description of the novel divisionless velocity estimation algorithm. The dynamics' analysis of the introduced novel method is presented in Section III, which includes convergence and stability considerations along with some numerical examples. Then, in Section IV, we provide

simulation results obtained by generated input data and pre-recorded real-encoder data, respectively. This section also involves a description of the models used for generating data in the simulations. Finally, conclusions are made in Section V.

II. THE VELOCITY ESTIMATION METHOD

A. THE PRINCIPLE OF THE BASIC MT-METHOD

The MT-method for velocity measurement can be described simply by the following formula [5]:

$$\hat{v}_k^{MT} = \frac{x_k - x_{k-1}}{T_s + \delta t_{k-1} - \delta t_k} \tag{1}$$

where $\hat{v}_k^{MT}$, and x_k are MT-method estimated velocity in the k-th sampling interval and encoder position sampled in the k-th sampling instant, respectively. T_s is the sampling period, and δt_k is the time elapsed between the time instant of the most recent encoder pulse in the current sampling interval (t_k^p) and the current sampling instant (t_k) such that it can be calculated as (for details see Fig. 1)

$$\delta t_k = t_k - t_k^p \tag{2}$$

where $t_k = kT_s$. Note that $x_k = x(t = t_k^p)$ and $x(t)$ is the actual position trajectory.

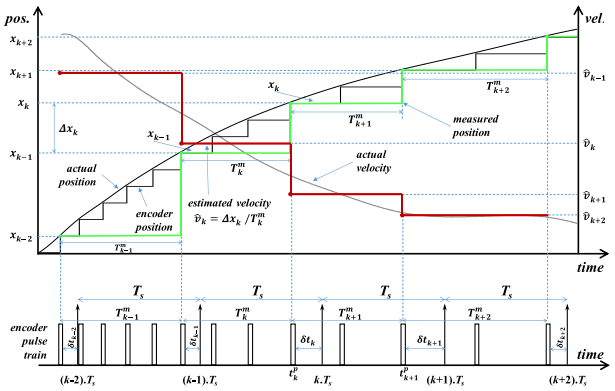


FIGURE 1. The timing diagram of the conventional MT-method.

In order to implement the MT velocity measurement method in practice, it is required to acquire the position pulse train from incremental encoder and clock pulses that enable determination of the measurement time interval. Fig. 1 shows the timing diagram of the conventional MT-method with real position, encoder position, encoder pulse train, sampling intervals with the constant period T_s, and variable measurement intervals T_k^m

$$T_k^m = T_s + \delta t_{k-1} - \delta t_k, \tag{3}$$

which enable velocity detection more accurately than is possible by the simple M-method. The latter only counts position pulses occurring within a fixed single sampling interval, and the velocity information is obtained by

$$v_k^M = \frac{\Delta x_k}{T_s} \tag{4}$$

where $\Delta x_k = x_k - x_{k-1}$ is the encoder position change in two consecutive sampling periods, that is available at every sampling instant. It is well-known that the M-method produces significant noise in the velocity data that should be filtered out in advanced motion control applications.

The measurement interval in the MT-method is split into three parts: The first part is the constant sampling interval in the middle, and the next two parts are the initial and final variable parts of the measurement interval respectively, the durations of which depend on the occurrence of the position pulse closest to the next sampling instant. The velocity update may be possible at every current sampling instant for the recent past measurement interval, which is adjusted automatically with reference to the position pulse occurrence; such velocity estimation procedure can provide information on average velocity over the most recent measurement interval, the duration of which is related not only to the duration of the sampling period, but also to the frequency of the position pulses. If the position pulses occur at every sampling interval, then the duration of the measurement interval cannot exceed two sampling intervals. In order to provide regular updates, the method requires at least one position pulse to occur within the sampling interval. Accurate velocity measurement at relatively low frequency of position pulses is thus possible. However, in order to obtain the velocity information, it is necessary to perform an arithmetic division in the processing hardware, since the denominator in (1) is a variable time interval that can be obtained by dedicated hardware for real-time measurement of the time interval that has been elapsed since the most recent position pulse in the sample interval. One can note that a division operation can be avoided in the case of the M-method, since the division with the constant sampling period can be replaced by multiplication with the constant sampling frequency $F_s = 1/T_s$ (see (4)). Thus, the arithmetic algorithm of the M-method simplifies computation complexity significantly in comparison with the algorithm of the MT-method. However, the M-method cannot provide an accurate velocity information in a wide speed range, thus an alternative solution should be derived which could combine the good features of both the MT-method and the M-method.

B. THE DIVISIONLESS ALGORITHM

The basic concept of the proposed algorithm for divisionless implementation of the MT-method (abbreviated as the DLMT-method), which *simplifies* the computation complexity, is shown by Fig. 2. Here, the main assumption is that the actual position information is available at the sampling instants. However, the position can be acquired in practice by the encoder only, and it yet has only a limited resolution, as shown by Fig. 1, and the true actual position information at the sampling instants cannot be available by means of a digital encoder. Thus, we propose to replace the actual position at the sampling instants, i.e. $x_k^a = x^a(t = t_k)$, by its estimated value $\hat{x}_k^a$, which may be computed from the encoder position

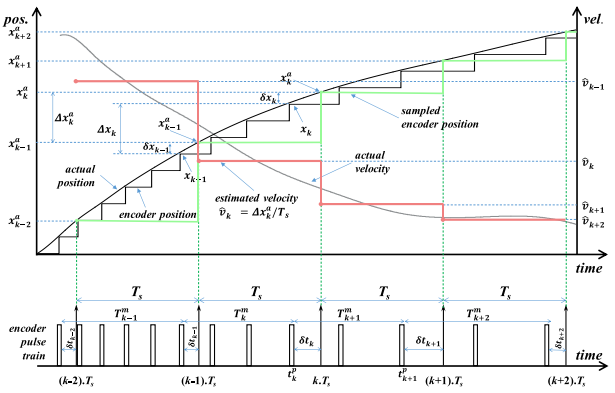


FIGURE 2. The timing diagram of the proposed DLMT-method.

that should be corrected slightly as

$$\hat{x}^a_k = x_k + \delta\hat{x}_k. \tag{5}$$

Though the required position correction is not available in practice, it can be estimated in the case of a relatively short sampling period in which the velocity could be considered quasi constant. Then the position correction can be approximated by linear interpolation $\delta\hat{x}_k = v_k\delta t_k$. However, unfortunately, in a causal estimation algorithm v_k is not available in the time when it is needed, and, therefore, it must be replaced by the estimated value of v_{k-1}, i.e. the velocity value in the previous sampling interval. The proposed divisionless velocity estimation algorithm is, thus, defined by

$$\hat{x}^a_k = x_k + \hat{v}_{k-1}\delta t_k, \tag{6}$$

$$\hat{v}_k = \frac{\Delta\hat{x}^a_k}{T_s}, \tag{7}$$

where $\Delta\hat{x}^a_k = \hat{x}^a_k - \hat{x}^a_{k-1}$, and $\hat{x}^a_k$ is the estimation of the actual position at the current sampling instant.

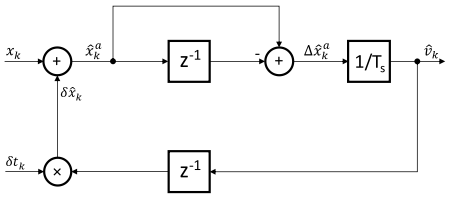


FIGURE 3. The block scheme of the proposed DLMT-method algorithm.

The proposed divisionless approximation algorithm of the MT-method (6)-(7) can be described graphically by the block scheme in Fig. 3. Obviously, the inputs in the algorithm are the sampled encoder position $\hat{x}_k$ and the sampled recent position pulse elapsed time δt_k, whereas the algorithm outputs the estimated velocity $\hat{v}_k$.

III. THE DYNAMICS ANALYSIS

In this section we analyze the proposed DLMT algorithm.

A. THE INPUT-OUTPUT DYNAMICS

By simple manipulation of (6)-(7) we can derive the difference equation that yields linear time-variant discrete filter of the 2nd order (8), which describes the input-output dynamics of the proposed algorithm.

$$\hat{v}_k - \frac{\delta t_k}{T_s}\hat{v}_{k-1} + \frac{\delta t_{k-1}}{T_s}\hat{v}_{k-2} = \frac{\Delta x_k}{T_s} \tag{8}$$

One can note that the right-hand side expression is the same as in (4), i.e. the encoder position change in two consecutive samplings Δx_k over the sampling period T_s; however, the coefficients on the left-hand side expression are time-variant, since their values depend on the time instants of the encoder position pulses' occurrence. It is worth noting that, in the state of the *dynamic equilibrium* (when $\hat{v}_k = \hat{v}_{k-1} = \hat{v}_{k-2}$), the difference equation of the filter (8) results in the algebraic equation

$$\hat{v}_k(1 - \frac{\delta t_k}{T_s} + \frac{\delta t_{k-1}}{T_s}) = \frac{\Delta x_k}{T_s}, \tag{9}$$

which can be resolved yielding the equilibrium velocity estimation $\hat{v}^{eq}_k$

$$\hat{v}^{eq}_k = \frac{T_s}{T_s - \delta t_k + \delta t_{k-1}}\frac{\Delta x_k}{T_s} = \frac{x_k - x_{k-1}}{T_s + \delta t_{k-1} - \delta t_k}. \tag{10}$$

The right-hand side of the equilibrium equation equals the output of the algebraic MT-method (1).

B. THE CONVERGENCE DYNAMICS

If we consider the dynamic equilibrium as the final desired value for the recursive estimation algorithm (6)-(7), then we can define the convergence error ε as

$$\varepsilon_k = \hat{v}^{eq}_k - \hat{v}_k, \tag{11}$$

and in the sequence it is possible to derive the error dynamics. The equation (10) can be rewritten as

$$T_s\hat{v}^{eq}_k - \delta t_k\hat{v}^{eq}_k + \delta t_{k-1}\hat{v}^{eq}_k = \Delta x_k,$$

and its final form can be written as

$$\hat{v}^{eq}_k - \frac{\delta t_k}{T_s}\hat{v}^{eq}_{k-1} + \frac{\delta t_{k-1}}{T_s}\hat{v}^{eq}_{k-2} = \frac{\Delta x_k}{T_s} + \frac{\Delta\delta t_k}{T_s}\Delta\hat{v}^{eq}_k - \frac{\delta t_{k-1}}{T_s}\Delta\hat{v}^{eq}_{k-1}, \tag{12}$$

where $\Delta\hat{v}^{eq}_k = \hat{v}^{eq}_k - \hat{v}^{eq}_{k-1}$ and $\Delta\delta t_k = \delta t_k - \delta t_{k-1}$. If we subtract (8) from (12), it yields the convergence error dynamics

$$\varepsilon_k - \frac{\delta t_k}{T_s}\varepsilon_{k-1} + \frac{\delta t_{k-1}}{T_s}\varepsilon_{k-2} = \frac{\Delta\delta t_k}{T_s}\Delta\hat{v}^{eq}_k - \frac{\delta t_{k-1}}{T_s}\Delta\hat{v}^{eq}_{k-1}. \tag{13}$$

It is obvious that the convergence error dynamics are driven by the acceleration, which is manifested by velocity change $\Delta\hat{v}^e_k$. If we assume relatively fast sampling intervals, then the velocity change in a single sampling interval can be considered negligible, and the right-hand side of (13) can be zeroed in practical advanced motion control applications.

C. THE ESTIMATION DYNAMICS

Let us define the average velocity $\bar{v}_k$ on a sampling interval such that

$$\bar{v}_k = \frac{x_k^a - x_{k-1}^a}{T_s}. \tag{14}$$

We assume that average velocity $\bar{v}_k$ could be the best possible velocity estimation if only accurate position measurements at the sampling instants would be available. Thus, it is considered as the reference velocity, and we then define the velocity estimation error as

$$e_k = \bar{v}_k - \hat{v}_k. \tag{15}$$

In the following, we derive the estimation error dynamics. Equation (14) can be rewritten as

$$x_k^a = x_{k-1}^a + T_s \bar{v}_k. \tag{16}$$

Furthermore, we can find a relation that links the time-stamped encoder position x_k (available at the current sampling instant) with the actual position x_{k-1}^a (measured at the previous sampling instant) and the average velocity $\bar{v}_k$ (valid in the current sampling interval) such that we obtain

$$x_k = x_{k-1}^a + (T_s - \delta t_k)\bar{v}_k. \tag{17}$$

We can also write the similar relations

$$x_k = x_k^a - \delta t_k \bar{v}_k, \tag{18}$$

and

$$x_{k-1} = x_{k-1}^a - \delta t_{k-1} \bar{v}_{k-1}, \tag{19}$$

where x_k^a is the actual position at the time-instant $t = t_k$, i.e. $x_k^a = x^a(t = t_k)$, and x_k is the encoder position sampled at the time-instant $t = t_k$ with a recent encoder position pulse occurrence at the time instant $t = t_k - \delta t_k$ within the current sampling time interval $t_{k-1} < t \leq t_k$, i.e. $x_k = x(t = t_k - \delta t_k)$. From the equations above we can further derive

$$\Delta x_k = x_k - x_{k-1} = (T_s - \delta t_k)\bar{v}_k + \delta t_{k-1}\bar{v}_{k-1}, \tag{20}$$

which can be rewritten as

$$T_s \bar{v}_k - \delta t_k \bar{v}_k + \delta t_{k-1}\bar{v}_{k-1} = \Delta x_k. \tag{21}$$

Finally, we can come to the equation of the form:

$$\bar{v}_k - \frac{\delta t_k}{T_s}\bar{v}_{k-1} + \frac{\delta t_{k-1}}{T_s}\bar{v}_{k-2} = \frac{\Delta x_k}{T_s} + \frac{\delta t_k}{T_s}\Delta\bar{v}_k - \frac{\delta t_{k-1}}{T_s}\Delta\bar{v}_{k-1}. \tag{22}$$

The difference equation above that outputs $\bar{v}_k$ is driven by the encoder position change Δx_k and the average velocity change $\Delta\bar{v}_k$ in a single sampling interval. This equation describes the dynamic relation that also links the sampled velocity $\bar{v}_k$ with the time interval δt_k.

The next step in the derivation procedure of the estimation error dynamics is subtraction of (8) from (22). Thus, we obtain the final result of the velocity error dynamics

$$e_k - \frac{\delta t_k}{T_s}e_{k-1} + \frac{\delta t_{k-1}}{T_s}e_{k-2} = \frac{\delta t_k}{T_s}\Delta\bar{v}_k - \frac{\delta t_{k-1}}{T_s}\Delta\bar{v}_{k-1}. \tag{23}$$

The velocity error dynamics are time-variant with the difference equation coefficients, which are defined by variable δt_k. This difference equation is driven by velocity change $\Delta\bar{v}_k$ in a single sampling period, which can be considered negligible if we consider very short sampling intervals. It is easy to see that the error is zeroed ($e_k^{eq} = 0$) in the system equilibrium. However, in order to eventually provide zero error estimation, asymptotical stable convergence must be guaranteed.

D. STABILITY ANALYSIS

In order to simplify the notation we denote variable coefficient $\tau_k = \delta t_k / T_s$ and rewrite (8) as:

$$y_k - \tau_k y_{k-1} + \tau_{k-1} y_{k-2} = u_k. \tag{24}$$

where $u_k = \Delta x_k / T_s$, and $y_k = \hat{v}_k$ are the filter input, and the filter output, respectively. Please note that $0 \leq \tau_k < 1$. It is clear that the filter (24) that includes variable coefficients within the difference equation, can be described as a discrete linear time-variant system. Thus, the filter (8) can be transformed to the canonical controllable state-space system representation

$$\begin{aligned} z_{k+1} &= A_k \cdot z_k + B_k \cdot u_k \\ y_k &= C_k \cdot z_k + D_k \cdot u_k \end{aligned} \tag{25}$$

where u_k, y_k, z_k are system input, output, and system state, respectively, for some $k = 0, 1, 2 \ldots$. The state-space vector is defined as $z = [z_1, z_2]^T$ with $z_1 = y_{k-2}$, and $z_2 = y_{k-1}$. A_k, B_k, C_k, and D_k are the system matrix, the control matrix, the output matrix, and the feed-forward matrix, respectively:

$$A_k = \begin{bmatrix} 0 & 1 \\ -\tau_{k-1} & \tau_k \end{bmatrix}, \quad B_k = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad C_k = \begin{bmatrix} -\tau_{k-1} & \tau_k \end{bmatrix}, \quad D_k = 1 \tag{26}$$

1) PRELIMINARIES

The system response can be derived as a unique solution of (25) such that the application of the superposition law yields two components, i.e. zero-input response and zero-state response [32]:

$$z_k = \Phi_k(0) z_0 + \sum_{j=0}^{k-1} \Phi_k(j+1) B_j u_j \tag{27}$$

where Φ_k is the state-transition matrix that is defined as

$$\Phi_k(j) = \begin{cases} A_{k-1}A_{k-2}\ldots A_j, & k > j \geq 0 \\ I & k = j. \end{cases} \tag{28}$$

Please note that $\Phi_k(j) = A_{k-1}\Phi_{k-1}(j)$ for some $k > j$. The default is to leave $\Phi_k(j)$ undefined for k<j. However, under the additional hypothesis that A_k is invertible for every k, we can set:

$$\Phi_k(j) = A_k^{-1}A_{k+1}^{-1}\ldots A_{j-1}^{-1}, \quad k < j. \tag{29}$$

The driven filter output response can subsequently be derived as:

$$y_k = C_k \Phi_k(0) z_0 + \sum_{j=0}^{k-1} C_k \Phi_k(j+1) B_j u_j + D_k u_k \quad (30)$$

Stability analysis of a discrete linear time-variant system is a hard problem as can be concluded from the extensive theoretical treatment in the literature [32]–[34]. However, the basic stability conditions of such systems can be explained as follows.

The un-driven system $z_{k+1} = A_k z_k$ is uniformly internally stable if, and only if, there exists a finite positive constant $\gamma > 0$ such that (see [32, Th. 22.4])

$$\|\Phi_k(j)\| \leq \gamma, \quad \forall k, j: \ k \geq j, \quad (31)$$

where $\|.\|$ stands for a vector induced norm. Additionally, a sufficient condition for internal asymptotic stability of the system can be guaranteed if, and only if, [33], [35]

$$\lim_{k \to \infty} \|\Phi_k(j)\| = 0, \quad \forall k, j: \ k \geq j. \quad (32)$$

Furthermore, the system is uniformly exponentially stable if, and only if, there exist a finite positive constant $\gamma > 0$ and a constant $0 \leq \lambda < 1$ such that (see [32, Th. 22.7])

$$\|\Phi_k(j)\| \leq \gamma \lambda^{k-j}, \quad \forall k, j: \ k \geq j. \quad (33)$$

Additionally, the system is uniformly exponentially stable if, and only if, there exists a finite positive constant $\gamma^s > 0$ such that (see [34, Th. 22.8])

$$\sum_{i=j+1}^{k} \|\Phi_k(i)\| \leq \gamma^s, \quad \forall k, j: \ k \geq j+1. \quad (34)$$

However, it can also be shown that the linear system under consideration is uniformly asymptotically stable if, and only if, it is uniformly exponentially stable (see Theorem 22.14 in [32]). It means that uniform exponential stability and uniform asymptotic stability are equivalent.

Further analysis treats the input-output driven system. The system (25) is uniformly BIBO (Bounded-Input Bounded-Output) stable if, and only if, there exists a positive constant $\gamma > 0$ such that [32]

$$\sum_{i=j}^{k-1} \|g_k(i)\| \leq \gamma, \quad \forall k, j: \ k > j \geq 0, \quad (35)$$

where g is the unit-pulse response

$$g_k(i) = C_k \Phi_k(i+1) B_i, \quad k > j \geq 0. \quad (36)$$

In the latter, we assume that D_k is bounded and, therefore, it does not affect the treatment, thus it can be omitted from the stability consideration without loss of generality. However, we can relate BIBO stability as a property of the zero-state response, with internal asymptotic stability as a property of a zero-input response. Suppose that the system (25) is uniformly exponentially stable, and there exist finite positive constants β and μ such that

$$\|B_k\| \leq \beta, \quad \|C_k\| \leq \mu, \quad \forall k \geq k_0. \quad (37)$$

Then the system is also uniformly BIBO stable [32]. Hence, asymptotic stability implies BIBO stability, but vice-versa is not always true.

Another theoretical tool for stability analysis is the Lyapunov stability criteria [32]–[34]. However, unfortunately, by its application it is not much easier to prove stability, since a quadratic Lyapunov function that proves uniform (asymptotic) stability for a given linear state equation can also be rather complicated to construct [32].

Though the stability conditions above are precisely definite, they are rather inconvenient in general on the other hand, because of the difficulty of computing explicit expressions, possible in closed-form, for $\Phi_k(j)$. Thus, thorough stability analysis of a practical linear time-variant system is often rather cumbersome and, thus, it is not always straightforward to provide a general theoretical stability proof for a certain system. However, slowly varying systems deserve special treatment [32], [36]–[39]. In the case of slowly-varying systems it is possible to simplify the problem [40] and to apply [32, Th. 24.8], which links uniform exponential stability with a pointwise eigenvalue λ_k of A_k and $\|\Delta A_k\|$. The theorem states that the system $z_{k+1} = A_k z_k$ is uniformly exponentially stable (and thus asymptotically stable) if there exist finite positive constants $\alpha > 0$, $\delta > 0$ such that

$$\alpha > 0, \quad \delta > 0: \ \|A_k\| \leq \alpha, \ \|\Delta A_k\| \leq \delta, \quad \forall k \geq 0, \quad (38)$$

where $\Delta A_k = A_k - A_{k-1}$, and every pointwise eigenvalue λ_k of A_k satisfies

$$|\lambda(A_k)| < 1, \quad \forall k \geq 0. \quad (39)$$

The proof of the theorem is based on the discrete Lyapunov equation (for details see [32]).

2) PROOF

Since a stability proof for discrete linear time-variant (26)-(27) is a hard problem, we relax the theoretical treatment such that we consider the system (25)-(26) as a slowly varying linear discrete-time system, which is a reasonable assumption, since the advanced motion control applications can provide relatively high sampling rates. Then, we apply the stability conditions (38)-(39) to test for uniform internal exponential stability. We need to show the existence and boundness of the norms $\|A_k\|$, and $\|A_k - A_{k-1}\|$, and we need to show the location of every pointwise eigenvalue inside the unit circle, such that $|\lambda(A_k)| < 1$, $\forall k \geq 0$, respectively. So, compute the vector induced norm $\|A_k\|$. It can be computed as

$$\|A_k\| = \sqrt{\lambda_{\max}(A_k^T A_k)} = \sqrt{\frac{(1 + \tau_{k-1}^2 + \tau_k^2) + \sqrt{(1 + \tau_{k-1}^2 + \tau_k^2)^2 - 4\tau_{k-1}^2}}{2}}. \quad (40)$$

By the analysis of the right-hand side expression in (40) it can quickly be verified that there exists a finite positive constant

$\alpha > 0$, such that

$$\sqrt{\frac{(1+\tau_{k-1}^2+\tau_k^2)+\sqrt{(1+\tau_{k-1}^2+\tau_k^2)^2-4\tau_{k-1}^2}}{2}} \leq \alpha \tag{41}$$

on the bounded interval $0 \leq \tau_k < 1$. In fact, the minimum and maximum values of the norm $\|A_k(\tau_k, \tau_{k-1})\|$ on this interval are

$$\min \|A_k(\tau_k, \tau_{k-1})\| = \|A_k(0, \tau_{k-1})\| = 1 \tag{42}$$

and

$$\alpha = \sup \|A_k(\tau_k, \tau_{k-1})\| = \|A_k(1, 1)\| = \sqrt{\frac{3+\sqrt{5}}{2}} \approx 1.62, \tag{43}$$

respectively. Thus, the norm $\|A_k\|$ cannot exceed α

$$\|A_k\| < \alpha \tag{44}$$

on the bounded interval $0 \leq \tau_k < 1$.

Next, it is necessary to evaluate the norm of ΔA_k

$$\begin{aligned} \Delta A_k &= A_k - A_{k-1} \\ &= \begin{bmatrix} 0 & 1 \\ -\tau_{k-1} & \tau_k \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ -\tau_{k-2} & \tau_{k-1} \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 \\ -\Delta\tau_{k-1} & \Delta\tau_k \end{bmatrix} \end{aligned} \tag{45}$$

where $\Delta\tau_k = \tau_k - \tau_{k-1}$. The vector induced norm $\|\Delta A_k\|$ can be computed as

$$\|\Delta A_k\| = \sqrt{\lambda_{\max}(\Delta A_k^T \Delta A_k)} = \sqrt{\Delta\tau_{k-1}^2 + \Delta\tau_k^2}. \tag{46}$$

For slowly varying parameters we can assume $\Delta\tau_k^2 \approx 0$ and $\Delta\tau_{k-1}^2 \approx 0$ that yields

$$\|\Delta A_k\| \approx 0. \tag{47}$$

However, even if we consider a general case of the bounded interval $0 \leq \tau_k \leq 1$, then the parameter change is bounded such that $0 \leq \Delta\tau_k^2 \leq 1$ and the maximum of the norm $\|\Delta A_k\|$ on the bounded interval $0 \leq \tau_k < 1$cannot exceed $\delta = \sqrt{2}$

$$\|\Delta A_k\| < \delta. \tag{48}$$

The pointwise eigenvalues for the time-variant system matrix A_k are:

$$\lambda_{1,2}(A_k) = \frac{1}{2}\left(\tau_k \pm \sqrt{\tau_k^2 - 4\tau_{k-1}}\right) \tag{49}$$

If $\tau_k^2 = 4\tau_{k-1}$ then we have the double real eigenvalue

$$\lambda_{1,2}(A_k) = \frac{\tau_k}{2} \tag{50}$$

that is obviously bounded as $\left|\lambda_{1,2}(A_k)\right| < 0.5$ on the bounded interval $0 \leq \tau_k < 1$. If $\tau_k^2 < 4\tau_{k-1}$ then we have the complex conjugate pair of eigenvalues

$$\lambda_{1,2}(A_k) = \frac{1}{2}\left(\tau_k \pm i\sqrt{4\tau_{k-1} - \tau_k^2}\right) \tag{51}$$

with an absolute value $\left|\lambda_{1,2}(A_k)\right| = \sqrt{\tau_{k-1}}$ that are obviously bounded by $\left|\lambda_{1,2}(A_k)\right| < 1$ on the interval $0 \leq \tau_k < 1$. If $\tau_k^2 > 4\tau_{k-1}$ then we have two positive real eigenvalues, such that they are bounded with a lower bound and an upper bound, such as

$$\begin{aligned} \frac{\tau_k}{2} < \lambda_1(A_k) &= \frac{1}{2}\left(\tau_k + \sqrt{\tau_k^2 - 4\tau_{k-1}}\right) \leq \frac{1}{2}\left(\tau_k + \sqrt{\tau_k^2}\right) \\ &= \tau_k, \end{aligned} \tag{52}$$

and

$$\begin{aligned} \frac{\tau_k}{2} > \lambda_2(A_k) &= \frac{1}{2}\left(\tau_k - \sqrt{\tau_k^2 - 4\tau_{k-1}}\right) \geq \frac{1}{2}\left(\tau_k - \sqrt{\tau_k^2}\right) \\ &= 0, \end{aligned} \tag{53}$$

respectively, and, thus, their maximum value is bounded, such as defined by the inequality $\max(\lambda_{1,2}(A_k)) < \tau_k$. The pair of real eigenvalues are, therefore, obviously bounded such that $\left|\lambda_{1,2}(A_k)\right| < 1$ on the bounded interval $0 \leq \tau_k < 1$. We can conclude that the pointwise eigenvalues given by (49) are absolutely bounded, such that

$$|\lambda(A_k)| < 1 \tag{54}$$

on the bounded interval $0 \leq \tau_k < 1$. Hence, the slowly varying linear discrete-time system system (25)-(26) satisfying (44) and (54) is asymptotically and uniformly exponentially stable.

Finally, the BIBO stability can be confirmed further by testing condition (37), that requires to show the existence of the upper bound μ such that the output matrix satisfies the inequality $\|C_k\| \leq \mu$ (the input matrix and the feedforward matrix of the system (25)-(26) are bounded by default, respectively). We can compute the norm $\|C_k\|$

$$\|C_k\| = \sqrt{\tau_{k-1}^2 + \tau_k^2} \tag{55}$$

It is easy to see that, on the bounded interval, $0 \leq \tau_k < 1$ the norm $\|C_k\|$ is bounded as well, and it cannot exceed $\mu = \sqrt{2}$ such that:

$$\|C_k\| < \mu. \tag{56}$$

Hence, the slowly varying linear discrete-time system (25)-(26) satisfying (44), (48), (54) and (56) is BIBO stable. This completes the proof.

3) NUMERICAL EXAMPLES

Though the proof is valid for slowly varying systems (that is a reasonable assumption in our case), we illustrate the stability properties of the proposed algorithm by numerical examples. For this purpose, we introduce the following stability indices ϕ and G,

$$\phi_k = \max_{j=0\ldots k-1} \|\Phi_k(j)\| \tag{57}$$

$$G_k = \max_{j=0\ldots k-1} \left(\sum_{i=j}^{k-1} \|g_k(i)\|\right) \tag{58}$$

which are derived from the stability conditions (31) and (35), respectively. $\Phi_k(j)$ is defined by (28), and $g_k(i)$ is defined by (36). In the case of asymptotic stability of a general linear time-varying system, the stability indices should stay bounded for $\forall k, j{:}k > j \geq 0$. The stability indices ϕ_k and G_k can, strictly, show the asymptotically stable sequence evaluation of any discrete linear time-variant system. We further consider stability indices α_k and δ_k that are related to a linear slowly time-varying system derived from (38)

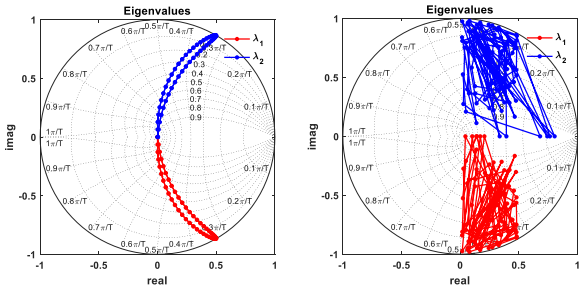
$$\alpha_k = \|A_k\|, \quad \forall k \geq 0 \tag{59}$$

$$\delta_k = \|\Delta A_k\|, \quad \forall k > 0 \tag{60}$$

which should stay bounded for $\forall k \geq 0$ in the case of an asymptotically stable system with every pointwise set of eigenvalues λ_k of A_k within the unit circle in a complex hyperplane ($|\lambda(A_k)| < 1$).

We checked stability indices for two examples:

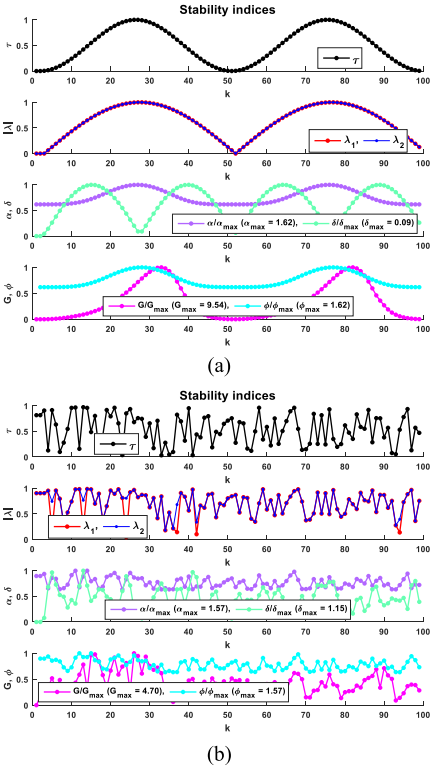
- Example (a) the variable coefficient τ_k is generated by sinusoidal waveform such that it varies within the range from 0 to 1 (τ_k is changing gradually), and
- Example (b) the variable coefficient τ_k is generated randomly within the range from 0 to 1 (τ_k is changing rapidly).



(a) (b)

FIGURE 4. The pointwise eigenvalues in complex plane. (a) Sinusoidal τ; (b) Random τ.

The pointwise eigenvalues in a complex plane are shown by Fig. 4, whereas Fig. 5 depicts stability indices. The plots on the left show example a), and the plots on the right show example b). Fig. 5 shows traces of the variable coefficient τ_k on the top diagrams, absolute values of the eigenvalues $\lambda_1(A_k)$ and $\lambda_2(A_k)$ on the upper middle diagrams, the normalized stability indices α_k and δ_k on the lower middle diagrams, and the normalized stability indices G_k and ϕ_k on the bottom diagrams, respectively. It is shown very clearly that the stability indices α_k and δ_k remain bounded at all pointwise eigenvalues within the unit circle for every τ_k within the range from 0 to 1. Similarly, the stability indices G_k and ϕ_k remain bounded too. Thus, asymptotically stable sequence evaluation of the considered system can be confirmed clearly by the numerical examples above in both cases: In the case of slowly varying coefficient τ_k, and even in the case of rapidly varying coefficient τ_k, respectively. One can notice



(a) (b)

FIGURE 5. The stability indices. (a) Sinusoidal τ; (b) Random τ.

that ϕ_k actually follows α_k, or more precisely $\phi_k = \alpha_{k-1}$. If $\phi_k = \alpha_{k-1}$ then

$$\max_{j=0\ldots k-1} \|\Phi_k(j)\|$$

is achieved at $j = k-1$. The evaluation of the pointwise eigenvalues and the stability indices depicted by Fig. 4 and Fig. 5, respectively, represent solid evidence that the system under consideration (24)-(26) with the equilibrium point given by (9) exhibits an asymptotically stable convergence and BIBO stability on the bounded interval of the variable coefficient $0 \leq \tau_k < 1$.

IV. THE SIMULATION RESULTS

We verified the proposed method by simulations with the generated motion data, and by the real data of an incremental position encoder. In the latter case, the data were processed in the same velocity estimation algorithm as in the simulations. In order to provide the simulation results, we developed the models of an incremental encoder and of the enhanced quadrature decoder, which are presented in the next sections. The models were designed in such a way to provide an effective simulation performance and verification of the proposed method.

A. THE MODEL OF AN INCREMENTAL ENCODER

Without loss of generality, we can assume a rotary Incremental Encoder (IE) in our study. Additionally, we can neglect its implementation technology (electromechanical, magnetic,

optical), and we assume an ideal encoder, so there are no imperfections present.

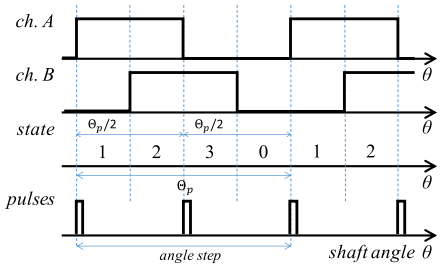


FIGURE 6. The waveform and the state transitions of an incremental position encoder.

The basic principle of operation can be described as the rotary IE, which transforms mechanical rotary motion into electrical output, provides digital pulses while its shaft is rotating, allowing measurement of relative displacement of the shaft [41], [42]. The number of the generated output pulses is proportional to its relative angular position of the shaft, while the frequency of them is proportional to the angular velocity. Fig. 6 illustrates the symmetric repeating waveforms of the IE A&B output square wave signals, and the state transitions, along with the corresponding pulses at the state changes, respectively. Here, Θ_p denotes angular step graduation such that

$$\Theta_p = (2\pi)/N, \tag{61}$$

where N stands for the number of channel pulses (A or B) generated by the encoder per one revolution of the shaft. It equals the number of the angular position steps in a single shaft turn and, thus, determines the angular position resolution of the encoder. In a single encoder pitch (Θ_p) the encoder generates four pulse transitions in both A and B channels. Thus, the minimal detectable angle increment is Θ_p /4.

The A&B output signals' generation can be described by the following equations:

$$A(\theta) = \begin{cases} 1, & \text{if } 1 \le \text{mod}(\dfrac{4\theta}{\Theta_p}, 4) < 3 \\ 0, & \text{else,} \end{cases} \tag{62}$$

$$B(\theta) = \begin{cases} 1, & \text{if } 2 \le \text{mod}(\dfrac{4\theta}{\Theta_p}, 4) < 4 \\ 0, & \text{else,} \end{cases} \tag{63}$$

where the function "a modulo n", i.e. mod(a, n), defines the operation that finds the remainder after division a/n, such that $a = n.q + r$, where q is the integer *quotient* and r is the remainder, such that $|r| < |n|$. This model for the IE output signals' generation, which is used for later simulation, can be described by the block scheme depicted by Fig. 7. Here, the shaft angle input θ is given in radian units (rad). The input block maps the shaft angle to the quarters of the angle step θ(qs), and the next block computes the shaft angle within a single encoder pitch in the same units of quarter-steps θ_p(qs). The pitch angle θ_p is quantized further in order to

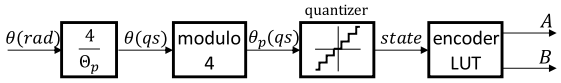


FIGURE 7. The model of an incremental position encoder.

obtain integer encoder pitch states. Finally, the output block generates the A&B output signals by the encoder Look-Up Table (LUT).

In the simulation (with enabled zero-crossing detection), every occurrence of a state transition can be time stamped accurately, however, all, or only selected events, are then stored for further processing.

B. THE MODEL OF THE ENHANCED QUADRATURE DECODER

In order to read pulses which are generated on the output of the incremental encoder, and to measure the relative angular position of the shaft, it is necessary to connect the encoder A&B channels to a dedicated hardware electronic circuit that provides proper conditioning of the external digital signals, A&B pulses decoding and bidirectional counting [14]. Such hardware component is usually called a quadrature decoder [43].

A quadrature decoder is applied for decoding the output of a quadrature encoder and, thus, it counts the edges of the A&B pulses, incrementing or decrementing, as determined by the decoded motion direction. The clock source for this high-speed high-resolution counter is usually provided by the XOR function of the A&B input signals. The process, by which edge counts are converted to digital position, depends on the mode of decoding used: X1, X2, or X4. In X1 decoding mode the counted position corresponds to the number of a single channel pulses. In X2 decoding mode doubles (x2) the number of pulses per revolution, whereas X4 decoding mode will quadruple (x4) the number of pulses per revolution and, thus, we can achieve four times better resolution of position measurement. In the latter case, that results in a pulse train as pictured by the bottom diagram in Fig. 6. However, whether the counter increments or decrements depends on which channel leads the other. An enhanced quadrature decoder can also provide a sort of event time stamping that enables measurement of elapsed time since the recent position pulse. The functional model of the enhanced quadrature position decoder with the outputs' sampling, which was designed in our study, is depicted by Fig. 8.

The position encoder module, which executes on each edge of channels A and B, is split mainly into the decoding part and the counting part. It outputs quantized angular position θ given in units of position pulses (pp), and, furthermore, the enhanced event-driven position decoder can also provide time stamping of each A&B edge (of each position pulse counted), i.e. t^p given in units of time pulses (tp). The logic of the quadrature decoder block (qdec) can be described by the Finite-State Machine (FSM) depicted by Fig. 9.

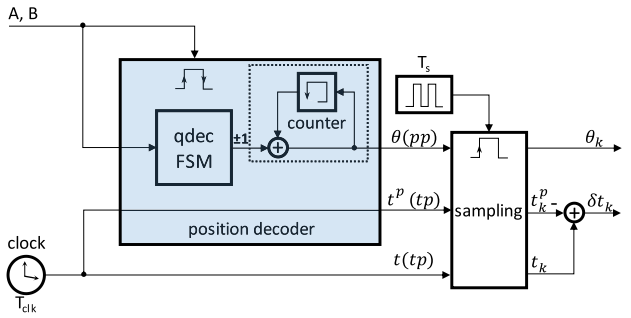


FIGURE 8. The model of the enhanced quadrature position decoder with output sampling.

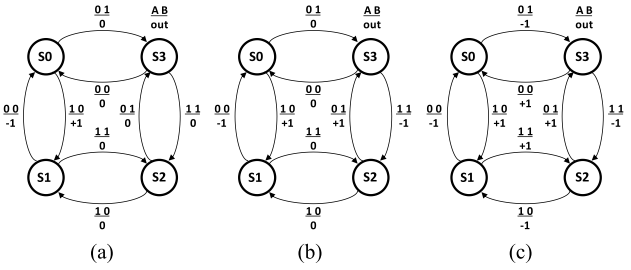


FIGURE 9. The FSM of the quadrature position decoder. (a) X1 decoding mode; (b) X2 decoding mode; (c) X4 decoding mode.

In general, it outputs increment (+1), decrement (−1) or zero (0) position pulse at the corresponding edge of the A&B inputs in order to update the output position.

The important feature of the enhanced quadrature encoder is providing an interval time measurement. Thus, we provide a clock source for absolute time t. The time t is latched at each edge of the A&B pulses by the position decoder and, thus, we obtain the pulse time t^p. Then, the digital signals of the position θ, the absolute time t and the pulse time stamp t^p are passed through the output time-driven block, respectively, which samples the data by a constant sampling period T_s and thus provides the corresponding outputs at regulator time instants $t_k = k.T_s$, where $k = 0, 1, 2, 3, \ldots$. The final operation in the enhanced quadrature encoder is a subtraction of the sampled time instant t^p from the sampled absolute time t. Thus, we obtain the time interval elapsed since the recent position pulse occurrence, i.e. $\delta t = t - t^p$ that is computed regularly at every sampling instant t_k. Since highly accurate information is required for this purpose, a fine clock grain should be provided given by minimal clock period T_{clk}. The enhanced quadrature position decoder finally outputs position $\theta_k = \theta(k.T_s)$ and elapsed time interval $\delta t_k = \delta t(k.T_s)$, which can be applied further for advanced velocity estimation. In real motion control applications there is normally a large ratio between T_s and T_{clk} in order to meet the required resolution demanded for the interval time measurement. Also, note that the architecture of a real hardware decoder with enhanced timing features may be different from the model developed for the effective simulation represented by the block scheme in Fig. 8.

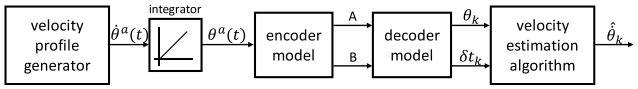


FIGURE 10. The simulation block scheme with generated data.

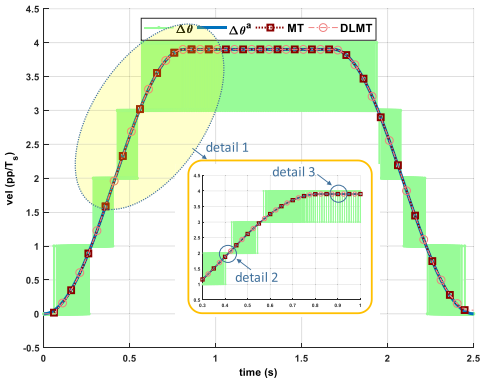
C. RESULTS WITH SIMULATED DATA

The simulation results were obtained by the simulation block scheme illustrated by Fig. 10. The simulation was performed by a desktop computer and the Matlab/Simulink simulation package. The actual velocity $\dot{\theta}(t)$ was generated by s-profile with maximum acceleration A_{max} and maximum velocity V_{max}, and integrated in time to produce actual position $\theta(t)$ that was further inputted into the encoder model (t is the independent time variable). The encoder model then outputted A&B channel pulses employed in the decoder model with output sampling. The decoder model produced sampled positions θ_k and sampled elapsed time interval since recent position pulse δt_k. The final block that implements either the conventional MT-method or the proposed DLMT-method (the simplified, divisionless algorithm) produced the velocity estimation $\hat{\dot{\theta}}_k$ at every $t_k = k.T_s$, where $k = 0, 1, 2, 3, \ldots$.

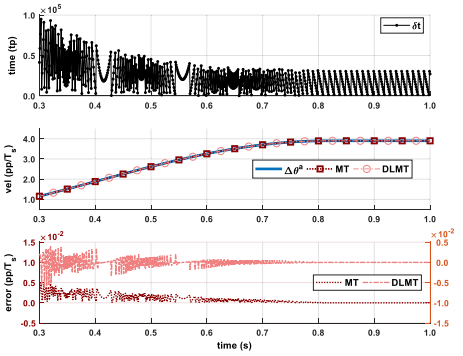
We have performed extensive simulation at various decoder parameters values, however, the results shown in this paper were obtained with the setup of hardware parameters shown in Table 1. The velocity profile generator parameters were $V_{max} = 1.56$ (rev/s) and $A_{max} = 3.00$ (rev/s^2). The results are shown by Fig. 11 on the plots (a)-(d).

The plot (a) shows the estimated velocity obtained by the M-method (denoted by $\Delta\theta$ – solid light green line), the true average velocity on the sampling interval obtained by reading actual position as described by (14), which may be available in simulation only (denoted by $\Delta\theta^a$, solid blue line), the estimated velocity calculated by the conventional MT-method (dotted dark red line with square markers), and the estimated velocity computed by the proposed DLMT- method (light red dash-dot line with circle markers), respectively, on the full time interval. The velocity signals are given in the units of the encoder position pulses in the single sampling period (pp/T_s). The Detail 1 that is marked on the plot is shown in the subplot. Well-known alternation of the M-type velocity ($\Delta\theta$) is shown clearly, while the true velocity ($\Delta\theta^a$), the MT-type velocity and the DLMT-type velocity overlap on the big plot, as well as on the subplot of the Detail 1.

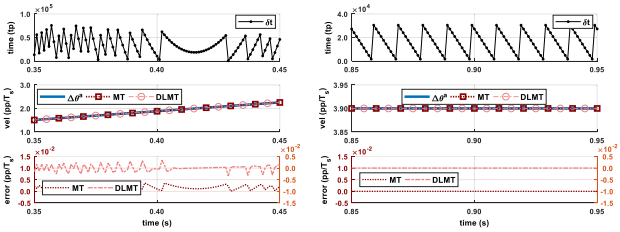
The next plots (b), (c), and (d) reveal more data related to the MT and DLMT velocities of the Details 1, 2 and 3, respectively. Note that Details 2 and 3 are focused on the acceleration phase and on the constant velocity phase, respectively, while Detail 1 covers a larger area, with the transition from the accelerated velocity to the constant velocity. The top diagram shows the variable time interval δt in units of the time pulses (tp), e.g. in the single sampling period



(a)



(b)



(c) (d)

FIGURE 11. The results with simulated data. (a) Velocities on full time interval including interval of detail 1; (b), (c), and (d) Elapsed time interval δt, velocities, and errors of detail 1, detail 2, and detail 3, respectively.

of 1 (ms) we can count up to 125,000 time pulses. The middle diagram of plot (b) shows the true velocity ($\Delta\theta^{\mathrm{a}}$, blue curve), the MT-method velocity (dotted dark red line with square markers), and the DLMT-method velocity (dash-dotted light red line with circle markers), respectively. It is evident that the velocity traces overlap. The bottom diagram plots errors w.r.t. the true average velocity ($\Delta\theta^{\mathrm{a}}$) as described by the equation (15) for the MT-velocity and DLMT-velocity, respectively. The MT velocity error scale is on the left y-axis, whereas the DLMT velocity error scale is on the right y-axis. It is very clear that the DLMT velocity error is larger, however, this is not significant, since both are less than 0.01 (pp/$\mathrm{T_s}$). Furthermore, they converge to zero as the velocity is passing from the acceleration phase to the constant phase. The DLMT velocity error during the acceleration phase is correlated with the variation of δt shown on the top diagram. Here, the error peaks are caused by abrupt changes in δt however, afterwards, the stable error convergence to zero is preserved. During the constant velocity phase, neither gradual changes nor abrupt changes of δt have any significant impact on the error, which remains close to zero.

The simulation results shown by Fig. 11 verify fully the proposed DLMT velocity estimation algorithm given by (6)-(7), as well as the error dynamics described by (23) and the stability analysis described in section on stability analysis (section III.D).

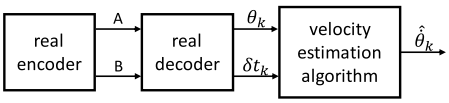


FIGURE 12. The simulation block scheme with measured data.

D. RESULTS WITH REAL DATA

A practical rotary encoder was connected to the real decoder digital circuit as shown by Fig. 12. In the experiments, the encoder was driven manually or by a gravitational load on a string wounded on the encoder shaft with the other end in order to eliminate the possible undesired effects of an automatic drive, such as an electrical servoactuator, which could add undesired disturbance to the smooth motion of the encoder shaft and, thus, complicate the analysis. The signals were captured by a suitably enhanced quadrature encoder hardware, available on the experimental hardware. The digital decoder produced θ_k and δt_k on-line. The data were logged into a memory in real-time, and then transferred in batch and stored on the simulation computer. Finally, these data were then processed by the simulation package, i.e. θ_k and δt_k were inputted to the velocity estimation block with either the conventional MT-method or the proposed DLMT-method, which produced the final output of velocity estimation $\hat{\theta}_k$. The parameters of the hardware used in the experiment are included in Table 1.

TABLE 1. The hardware parameters.

Encoder		
Type	rotary	
Model	Heidenhain ROD420	
N	2500	(pulses/rev)
Decoder		
clock	125	(MHz)
$\mathrm{T_{clk}}$	8	(ns)
$\mathrm{T_s}$	1	(ms)

The results with real encoder data are shown by Fig. 13 in the plots (a)-(e). The plot (a) shows the estimated velocity obtained by the M-method (denoted by $\Delta\theta$ – solid light green line), the estimated velocity calculated by the conventional MT-method (solid dark red line with square markers), and the estimated velocity computed by the proposed DLMT- method (light red dotted line with circle markers), respectively, in the units of encoder position pulses over sampling period (pp/$\mathrm{T_s}$), e.g. the speed of 10 (rev/s) converts to 2.5 (pp/$\mathrm{T_s}$).

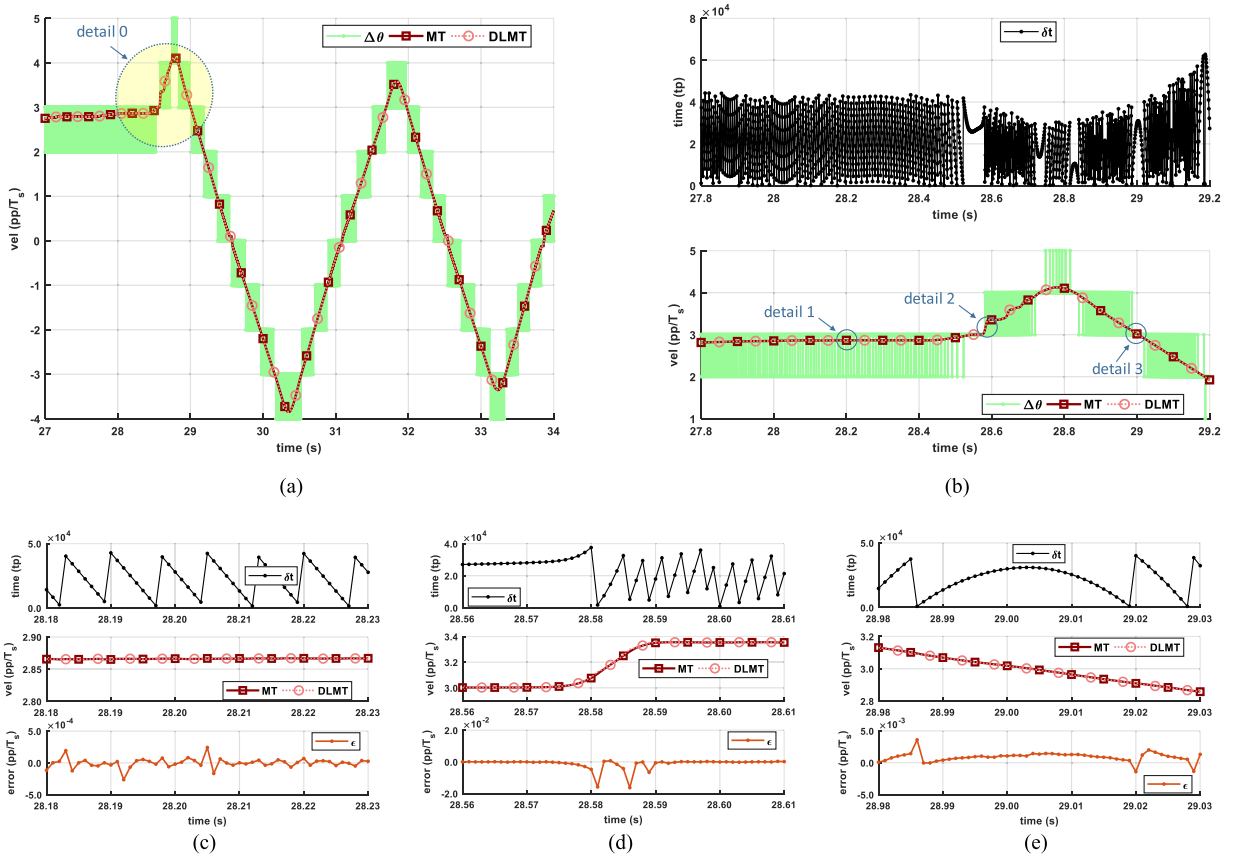


FIGURE 13. The results with measured data. (a) Velocities on full time interval; (b) Elapsed time interval δt and velocities of Detail 0; (b), (c), and (d) Elapsed time interval δt, velocities, and errors of detail 1, detail 2, and detail 3, respectively.

The traces are shown for a rather long time interval in which the velocity is changing closely to a triangular waveform. The plot (b) depicts the Detail 0. Here, the top diagram shows the variable time interval δt in the units of time pulses (tp). The bottom diagram shows the estimated velocity by the M-method, the MT-method, and the velocity by the proposed DLMT-method, respectively. Similarly, the middle plots on (c)-(e) show the estimated velocity by the MT-method, and by the proposed DLMT-method, respectively. Additionally, the bottom plots on (c)-(e) show velocity error $\varepsilon = \hat{v}^{MT} - \hat{v}$ that is related to (13). The M-velocity is highly alternating, with a value change of one position pulse over sampling interval (pp/T_s). It is well-known that the velocity obtained by the M-method may be extremely noisy, as confirmed in our case. The MT-method can estimate velocity smoothly, even in practical applications as shown by Fig. 13. However, it also shows clearly that DLMT-velocity can produce a smooth output as well; the MT-velocity and the DLMT-velocity practically overlap. The plots (c)-(e) further show some details in which we can observe the velocity error as well. The Detail 1 from plot (b) is shown on plot (c). This detail is related to the constant velocity phase; however, the time interval δt is varying gradually with occasional abrupt changes. Such trace plot is characteristic for a constant frequency encoder pulse train that is asynchronous with reference to the equidistant sampling of the decoder. In our case, three pulses are normally counted in a single sampling interval, with the recent pulse gradually approaching the sampling instant in sequential sampling intervals, thus δt is decreasing linearly. It is crossing the sampling time instant in almost regular intervals. On these occasions, only two position pulses are counted in the single sampling period and δt is increased instantly, since it records the elapsed time interval since the second position pulse of the triple. However, the velocity error in this case is less than 0.0005 (pp/T_s), which is extremely low. Detail 3 on plot (e) shows a linearly decreasing velocity case. In this case, we can observe gradual increasing and decreasing of the δt in the large middle of the diagram. Additionally, there are some abrupt changes in the first and in the last phase of the record, which are due to the change of counted position pulses in a single sampling period. Consequently, these abrupt changes of δt cause the error peaks evident in the bottom diagram. However, the velocity error stays within 0.005 (pp/T_s), which is still extremely low, though significantly more than in Detail 1. However, as expected, while the velocity is changing at a constant rate, by increasing δt, the velocity error is also increasing slightly. Detail 2 on plot (d) is focused on more rapid velocity change in the middle of the diagram. Again, the change of counted position pulses in a single sampling period is reflected in the rapid changes of δt. Few velocity

error peaks appear, limited by 0.02 (pp/T_s). In the rest of the plot we can observe a constant velocity with negligible velocity error in the scale shown. It is interesting that the latter holds for the first phase with constant velocity, where δt is practically constant, as well as for the last phase with constant velocity, where we can observe rapid changes in δt. However, the asymptotically stable convergence to zero after the error peaks is also evident. Thus, the results with real data clearly confirm the derived error dynamics (13).

V. CONCLUSION

In this paper, we proposed a novel computation algorithm for velocity estimation by incremental encoders, which relates to the well-known MT-method. The proposed algorithm avoids the use of arithmetic division; instead, it requires only more simple arithmetic operations such as addition and multiplication. Thus, it is named as DLMT (Division Less MT) method. This is a significant advantage over the conventional MT-method and, thus, it allows for cost-effective implementation on a digital platform that lacks a powerful computation unit. Thus, the proposed algorithm is suitable for efficient implementation on an FPGA circuit. However, the proposed algorithm converts the MT-method from the algebraic form to a recursive form. The input-output relation is resolved into a difference equation of second order with variable coefficients. Thus, though the proposed algorithm is simplified from the computational point of a view, it is, on the other hand, more complex, since it is governed by the dynamic input-output relationship. Therefore, the stability and convergence have been discussed. It has been shown that the proposed algorithm is inherently asymptotically stable and converges to the conventional MT-method output value.

The proposed algorithm has been verified off-line on a desktop computer by simulated data and real data obtained by a practical rotary encoder and a proper reading digital hardware circuit. The model of the encoder and the reading electronics applied in the simulation have been developed and are presented in the paper as well. We have conducted extensive simulations in which we have examined the impact of practical parameters such as sampling time and clock frequency on the velocity estimation. It has been shown in all cases with simulated data, respectively, that there are no significant differences between the conventional MT-method and the proposed DLMT-method in terms of the output velocity estimation. However, the results of velocity estimation may improve with higher sampling rates and clock frequency as expected. Similarly has also been confirmed in the case of the real data. In future, we plan to implement the proposed algorithm on an FPGA to validate it by a practical implementation on a targeted hardware in this paper.

REFERENCES

[1] T. Ohmae, T. Matsuda, K. Kamiyama, and M. Tachikawa, "A microprocessor-controlled high-accuracy wide-range speed regulator for motor drives," *IEEE Trans. Ind. Electron.*, vol. IE-29, no. 3, pp. 207–211, Aug. 1982.

[2] M. Prokin, "Double buffered wide-range frequency measurement method for digital tachometers," *IEEE Trans. Instrum. Meas.*, vol. 40, no. 3, pp. 606–610, Jun. 1991.

[3] M. Prokin, "Extremely wide-range speed measurement using a double-buffered method," *IEEE Trans. Ind. Electron.*, vol. 41, no. 5, pp. 550–559, Oct. 1994.

[4] R. C. Kavanagh, "Improved digital tachometer with reduced sensitivity to sensor nonideality," *IEEE Trans. Ind. Electron.*, vol. 47, no. 4, pp. 890–897, Aug. 2000.

[5] R. C. Kavanagh, "Performance analysis and compensation of M/T-type digital tachometers," *IEEE Trans. Instrum. Meas.*, vol. 50, no. 4, pp. 965–970, Aug. 2001.

[6] R. C. Kavanagh, "An enhanced constant sample-time digital tachometer through oversampling," *Trans. Inst. Meas. Control*, vol. 26, no. 2, pp. 83–98, 2004.

[7] K. Hachiya and T. Ohmae, "Digital speed control system for a motor using two speed detection methods of an incremental encoder," in *Proc. Eur. Conf. Power Electron. Appl.*, Aalborg, Denmark, 2007, pp. 1–10.

[8] N. K. Boggarpu and R. C. Kavanagh, "New learning algorithm for high-quality velocity measurement and control when using low-cost optical encoders," *IEEE Trans. Instrum. Meas.*, vol. 59, no. 3, pp. 565–574, Mar. 2010.

[9] J.-T. Pu and H. Wang, "A novel variable M/T method for speed measurement with high precision in a wide speed range," in *Proc. 2nd Int. Conf. Electron. Mech. Eng. Inf. Technol. (EMEIT)*, Liaoning, China, 2012, pp. 1855–1858.

[10] Y. Chen, M. Yang, J. Long, D. Xu, and F. Blaabjerg, "M/T method based incremental encoder velocity measurement error analysis and self-adaptive error elimination algorithm," in *Proc. 43rd Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Beijing, China, Oct./Nov. 2017, pp. 2085–2090.

[11] I. Koren, *Computer Arithmetic Algorithms*, 2nd ed. Natick, MA, USA: A K Peters, 2002.

[12] E. Monmasson, L. Idkhajine, M. Cirstea, I. Bahri, A. Tisan, and M. W. Naouar, "FPGAs in industrial control applications," *IEEE Trans. Ind. Informat.*, vol. 7, no. 2, pp. 224–243, May 2011.

[13] J. J. Rodríguez-Andina, M. D. Valdés-Peña, and M. J. Moure, "Advanced features and industrial applications of FPGAs—A review," *IEEE Trans. Ind. Informat.*, vol. 11, no. 4, pp. 853–864, Aug. 2015.

[14] C. Quintáns, J. Fariña, and J. Marcos-Acevedo, "Improving the performance of incremental encoders with conditioning circuits based on FPGA," *Measurement*, vol. 90, pp. 1–3, Aug. 2016.

[15] P. Bhatti and B. Hannaford, "Single-chip velocity measurement system for incremental optical encoders," *IEEE Trans. Control Syst. Technol.*, vol. 5, no. 6, pp. 654–661, Nov. 1997.

[16] J. N. Lygouras, V. Kodogiannis, T. P. Pachidis, and G. C. Sirakoulis, "A new method for digital encoder adaptive velocity/acceleration evaluation using a TDC with picosecond accuracy," *Microprocess. Microsyst.*, vol. 33, nos. 7–8, pp. 453–460, Oct./Nov. 2009.

[17] Texas Instruments. (Dec. 2008). *TMS320x2833x, 2823x Enhanced Quadrature Encoder Pulse (EQEP) Module—Reference Guide*. Accessed: May 12, 2018. [Online]. Available: http://www.ti.com/lit/ug/sprug05a/sprug05a.pdf

[18] K. Ohnishi, M. Shibata, and T. Murakami, "Motion control for advanced mechatronics," *IEEE/ASME Trans. Mechatronics*, vol. 1, no. 1, pp. 56–67, Mar. 1996.

[19] T. Tsuji, T. Hashimoto, H. Kobayashi, M. Mizuochi, and K. Ohnishi, "A wide-range velocity measurement method for motion control," *IEEE Trans. Ind. Electron.*, vol. 56, no. 2, pp. 510–519, Feb. 2009.

[20] A. Hace and M. Franc, "FPGA implementation of sliding-mode-control algorithm for scaled bilateral teleoperation," *IEEE Trans. Ind. Informat.*, vol. 9, no. 3, pp. 1291–1300, Aug. 2013.

[21] A. Hace and M. Franc, "Pseudo-sensorless high-performance bilateral teleoperation by sliding-mode control and FPGA," *IEEE/ASME Trans. Mechatronics*, vol. 19, no. 1, pp. 384–393, Feb. 2014.

[22] W.-H. Zhu, T. Lamarche, E. Dupuis, D. Jameux, P. Barnard, and G. Liu, "Precision control of modular robot manipulators: The VDC approach with embedded FPGA," *IEEE Trans. Robot.*, vol. 29, no. 5, pp. 1162–1179, Oct. 2013.

[23] V. Chawda, O. Celik, and M. K. O'Malley, "Evaluation of velocity estimation methods based on their effect on haptic device performance," *IEEE/ASME Trans. Mechatronics*, vol. 23, no. 2, pp. 604–613, Apr. 2018.

[24] J. J. R. Andina, E. de la Torre Arnanz, and M. D. V. Peña, *FPGAs: Fundamentals, Advanced Features, and Applications in Industrial Electronics*. Boca Raton, FL, USA: CRC Press, 2017.

[25] W.-H. Zhu, "FPGA logic devices for precision control: An application to large friction actuators with payloads," *IEEE Control Syst. Mag.*, vol. 34, no. 3, pp. 54–75, Jun. 2014.
[26] B. Jovanovic, R. Jevtic, and C. Carreras, "Binary division power models for high-level power estimation of FPGA-based DSP circuits," *IEEE Trans. Ind. Informat.*, vol. 10, no. 1, pp. 393–398, Feb. 2014.
[27] G. Sutter and J. P. Deschamps, "High speed fixed point dividers for FPGAs," in *Proc. Int. Conf. Field Program. Logic Appl.*, Prague, Czech Republic, 2009, pp. 448–452.
[28] G. Sutter and J.-P. Deschamps, "Fast radix 2^k dividers for FPGAs," in *Proc. 5th Southern Conf. Program. Logic (SPL)*, Sao Carlos, Brazil, 2009, pp. 115–122.
[29] J.-P. Deschamps, G. J. A. Bioul, and G. D. Sutter, *Synthesis of Arithmetic Circuits: FPGA, ASIC and Embedded Systems*. Hoboken, NJ, USA: Wiley, 2006.
[30] W.-H. Zhu, "FPGA-based velocity estimation for control of harmonic drives," in *Proc. IEEE Int. Conf. Mechatronics Automat.*, Xi'an, China, Aug. 2010, pp. 1069–1074.
[31] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Syst. J.*, vol. 4, no. 1, pp. 25–30, Mar. 1965.
[32] W. J. Rugh, *Linear System Theory*, vol. 2. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.
[33] R. P. Agarwal, *Difference Equations and Inequalities: Theory, Methods, and Applications*. Boca Raton, FL, USA: CRC Press, 2000.
[34] A. Halanay and V. Ionescu, *Time-Varying Discrete Linear Systems: Input-Output Operators. Riccati Equations. Disturbance Attenuation*. Berlin, Germany: Birkhäuser, 1994.
[35] B. Zhou and T. Zhao, "On asymptotic stability of discrete-time linear time-varying systems," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 4274–4281, Aug. 2017.
[36] C. A. Desoer, "Slowly varying discrete system $x_{i+1} = A_i x_i$," *Electron. Lett.*, vol. 6, no. 11, pp. 339–340, May 1970.
[37] F. Amato, G. Celentano, and F. Garofalo, "New sufficient conditions for the stability of slowly varying linear systems," *IEEE Trans. Autom. Control*, vol. 38, no. 9, pp. 1409–1411, Sep. 1993.
[38] E. W. Kamen, P. P. Khargonekar, and A. Tannenbaum, "Control of slowly-varying linear systems," *IEEE Trans. Autom. Control*, vol. 34, no. 12, pp. 1283–1285, Dec. 1989.
[39] J. J. DaCunha, "Stability for time varying linear dynamic systems on time scales," *J. Comput. Appl. Math.*, vol. 176, no. 2, pp. 381–410, 2005.
[40] J. M. Davis, I. A. Gravagne, R. J. Marks, and A. A. Ramos, "Algebraic and dynamic Lyapunov equations on time scales," in *Proc. 42nd Southeastern Symp. Syst. Theory*, Tyler, TX, USA, 2010, pp. 329–334.
[41] I. I. Incze, C. Szabó, and M. Imecs, "Incremental encoder in electrical drives: Modeling and simulation," in *Computational Intelligence in Engineering* (Studies in Computational Intelligence), vol. 313, I. J. Rudas, J. Fodor, and J. Kacprzyk, Eds. Berlin, Germany: Springer, 2010, pp. 287–300.
[42] Jiss Mohan K. and N. Johnson, "Devising simulink optical encoder pulse manipulation and its evaluation," in *Proc. 12th Int. Conf. Intell. Syst. Design Appl. (ISDA)*, Kochi, India, 2012, pp. 640–644, doi: 10.1109/ISDA.2012.6416612.
[43] M.-F. Tsai and C.-P. Chen, "Design of a quadrature decoder/counter interface IC for motor control using CPLD," in *Proc. IEEE 28th Annu. Conf. Ind. Electron. Soc. (IECON)*, Sevilla, Spain, Nov. 2002, pp. 1936–1941.

ALEŠ HACE received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Maribor, Maribor, Slovenia, in 1994, 1998, and 2001, respectively. In 1994, he joined the Institute of Robotics, Faculty of Electrical and Computer Science, University of Maribor, where he has been an Assistant Professor for automation and robotics since 2006. In 1999, he was a Visiting Research Fellow with Loughborough University, Leicestershire, U.K. In 2010, he has been elected Associate Professor. He is currently the Head of the Laboratory for Industrial Robotics. The main areas of his research are related to the areas of mechatronics, robotics and motion control, haptic interfaces and bilateral teleoperation, and sliding-mode control applications. He was a recipient of the Bedjanic Slovenian National Research Award in 1998. He is an IEEE member of its Industrial Electronics Society and Robotics and Automation Society and a member of the National Automatic Control Society of Slovenia.

MILAN ČURKOVIČ received the B.Sc., M.Sc., and Ph.D. degrees in electrical engineering from the University of Maribor, Maribor, Slovenia, in 1983, 2010, and 2014, respectively. Since 1983, he has been with the Institute of Robotics, University of Maribor, where he currently serves as a Researcher. His research interests include control of electrical drives, driving, and measurement interfaces in programmable logic devices.