

Article

# Accurate FPGA-Based Velocity Measurement with an Incremental Encoder by a Fast Generalized Divisionless MT-Type Algorithm

Aleš Hace \*  and Milan Čurkovič

Institute of Robotics, Faculty of Electrical Engineering and Computer Science, University of Maribor, Koroška cesta 46, SI-2000 Maribor, Slovenia; milan.curkovic@um.si

\* Correspondence: ales.hace@um.si; Tel.: +386-2-220-7301

Received: 22 August 2018; Accepted: 26 September 2018; Published: 27 September 2018



**Abstract:** Velocity measurement by an incremental encoder is an important issue for advanced motion control applications such as robotics. In this paper, we deal with a kind of MT-type velocity estimation method. Though the conventional MT method is well known and has been well proven in practice, it requires execution of an arithmetic division operation that prevents an efficient implementation on low-cost FPGA-based control platforms. Thus, we propose a divisionless MT-type algorithm, which can provide a similar performance in velocity estimation accuracy as the conventional method, but requiring significantly less FPGA resources, since it implements only simple arithmetic operations such as addition, subtraction, and multiplication, that can be implemented more easily on the processing hardware. Furthermore, the algorithm is fast in execution, thus, it provides the output in only a few clock cycles. Though the proposed algorithm can be described in a recursive form, the stability of the estimation process is not jeopardized, although it is an important issue in this case. Hence, the algorithm is introduced in a form which assures stability in a wide speed range. We show the implementation of the algorithm on the experimental FPGA platform. The experimental results validated the proposed divisionless MT-type algorithm fully for accurate velocity estimation.

**Keywords:** incremental encoders; velocity measurement; MT method; FPGA; motion control

## 1. Introduction

Advanced motion control applications require highly accurate wide-range velocity information with high-bandwidth [1–4]. A typical motion control system consists of a motorized drive, a feedback device and a controller. The latter should generate the necessary control decisions in order to achieve the demanded system performance; however, to be able to do so, it requires proper feedback information. In motion control applications the system stability is of primary concern. Though there are numerous control architectures in use, the velocity feedback plays a crucial role in assuring the stability of the system and in the disturbance rejection. Velocity estimation is also essential for precision and performance of the motion control, which requires high-accuracy and high bandwidth velocity feedback information. Typical feedback devices in such systems are incremental position encoders, which have been used widely for position measurement in digital control systems of servo drives for several decades [5]. Usually, the feedback signal for the velocity controller is calculated from the measured position obtained by the incremental encoder [6]. However, due to the discrete nature of the incremental encoder operating as a pulse generator, a straightforward approach for velocity estimation results in highly noisy data that cannot usually be applied directly in the control loop. Though the noise component can be filtered out, this approach causes a phase lag that is undesired in high-performance closed-loop control in the applications of robotics or haptic interfaces [3,4,7].

There has been extensive research over the past few decades in order to provide accurate and delay-less smooth velocity information over a wide speed range from a digital incremental encoder. An excellent survey is provided in [8,9]. The simplest velocity estimation approach is based on counting encoder position pulses within a sampling period, as is the case in the well-known M-method. However, the M-method produces highly noisy output, especially at high sampling rates, due to the spatial position quantization inherent to incremental encoders. The alternative T-method can be utilized instead, in which the time interval between two adjacent encoder pulses should be measured by counting high-frequency clock pulses. The velocity information is then obtained by the reciprocal of the measured time interval, i.e., by arithmetic division. Though the method can provide fine velocity estimation at low speed, it is prone to errors at high speed. Thus, the main stream of the velocity measurement methods combine the M-method and the T-method. Since the pioneering work by Ohmae [10], the MT-method has been applied widely, because it works well in wide speed ranges and also has a high accuracy in the low speed range. Some variations of the MT-method appeared in [11–13]. They can be classified as the MT-type methods which may provide highly smooth velocity with no phase lag in a wide speed range. Further research deals with performance improvement of the MT-method and its enhanced robustness to the hardware inaccuracies [2,14–16]. Time stamping of encoder pulses presents a generalization of the MT method [17]. Here, the velocity is estimated by a polynomial fitting through a number of time-stamped encoder counts, which increases computation complexity significantly. On the other hand, Zhu [18] proposed an MT-type method with an efficient processing algorithm in order to simplify the required real time computation effort.

The MT-method is a high-performance velocity estimation approach that requires dedicated hardware, which should provide counting encoder pulses and measurement of elapsed time intervals related to the encoder pulses. It can be incorporated easily into digital processing systems with a fixed sample period. However, processing requires execution of some arithmetic operations, including multiplication and division. The latter is a much more demanding operation for processing hardware than multiplication [19]. Though advanced DSP processors allow implementation with the arithmetic division with no significant effort [20], the system design complexity may increase significantly in the case of advanced motion control applications with high sampling rates, which require special hardware interfaces usually unavailable in commercial DSP controllers. On the other hand, digital programmable circuits such as FPGAs, which are, nowadays, used widely for many industrial applications [21,22], offer possibilities for integrating complex hardware systems and their interfaces into a single FPGA chip. FPGA devices have reached a high level of development that puts them in competition with the traditional Application Specific Integrated Circuits (ASICs) in terms of performance, power consumption and cost, which make them most suitable for smart, intelligent and reconfigurable low-power sensor systems in different application fields [23–26]. Thus, they can also easily be designed for custom conditioning circuits required for improved velocity estimation methods [27] in advanced motion control applications that require not only highly accurate and short sampling periods, but also a short processing time of control algorithms. Furthermore, the low cost of recent FPGAs and the capacity of these devices enable economical implementation of a custom complex circuitry featuring parallel operation and low power consumption in a small space. This allows development of new high-performance applications; however, complex computation algorithms still present a design challenge [28].

In our research, we are concerned with the implementation of the MT-method on an FPGA. Our focus is on the data processing part that involves a computation with an arithmetic division, which is particularly difficult for FPGA-based embedded electronics to perform [29,30], and is characterized with high computation latency and high consumption of FPGA resources [31]. Therefore, for efficient implementation of the MT-method on an FPGA, it is necessary to circumvent the division operation inherent in the conventional algorithm. On the other hand, multiplication can be implemented easily in an efficient way, since some FPGAs embed dedicated multipliers, thus eliminating the need for using FPGA distributed logic cells [32]. Zhu [18] proposed a solution which combines the M-method

and the T-method seamlessly, in which the arithmetic division is performed by the sophisticated mechanism with two counters and an accumulator. However, despite the simplicity of the method, it requires execution of the algorithm continuously in every clock cycle that increases resources and power consumption of the digital circuit, which is undesired. Especially, the latter has become a first order concern for VLSI design engineers [30]. Hacı [33] introduced a novel MT-type method, which eliminates the division from the algorithm. Hence, it is called a DivisionLess MT method (DLMT). It can be performed by simpler arithmetic operations, such as addition and multiplication, yet produces an output very close to the conventional MT-method.

In this paper, we redesign the DLMT method algorithm, which is described in a recursive form. Due to the recursive nature of the algorithm, stability is an issue and, thus, it has already been investigated in [33]. Asymptotic stability has been shown clearly; however, the case of blank intervals (sampling intervals with no encoder pulses' occurrence), which may appear sporadically or as a series of contiguous blank sampling intervals, has been not covered. As is explained later in Section 2 of the current paper, the blank intervals can cause instability of the recursive DLMT algorithm. Therefore, we modified it and thus now we introduce the Generalized DivisionLess MT algorithm (GDLMT) that assures asymptotic stability in all speed ranges, including the case when encoder pulses are widespread. The proposed algorithm, which has been designed for efficient FPGA implementation while preserving utmost accuracy of the conventional MT-method, presents the main contribution of the paper. It has been implemented on the FPGA-based platform and validated experimentally with an industrial optical rotary incremental encoder. The experimental results show excellent matching with the conventional MT-method. Thus, the latter can be replaced successfully by the proposed GDLMT algorithm, which provides an efficient FPGA implementation, since it requires significantly less resources and provides fast execution that is highly desirable in advanced motion control applications with short sampling periods.

## 2. Materials and Methods

### 2.1. Basics of Incremental Encoders

Incremental encoders are used mainly for position measurement by counting the digital pulses which they produce. Without loss of generality, we can assume a rotary incremental encoder in our study. The basic principle of its operation can be described such that it transforms mechanical motion into electrical output, thus providing digital pulses while its shaft is rotating, allowing measurement of the relative displacement of the shaft [34]. The number of generated output pulses is proportional to its relative angular position of the shaft, while the frequency of them is proportional to the angular velocity. In a single turn, the encoder generates a certain number of pulses, such that the angular step graduation given by the encoder pitch  $\Theta_p$  can be calculated as:

$$\Theta_p = (2\pi)/N, \quad (1)$$

where  $N$  stands for the number of pulses generated by the encoder per one revolution of the shaft. In our study, we consider quadrature incremental encoders that output two channels of pulses, i.e., A and B channels, with the same number of digital pulses generated. However, the pulses in channel B should be shifted in phase by 90 degrees w.r.t. the pulses in channel A. The phase shift between the A and B channel pulses enables us to discriminate the direction of movement. In order to accurately read the pulses, which are generated by the incremental encoder, and to measure the relative angular position of the shaft, it is necessary to connect the encoder electrical lines to a dedicated hardware electronic circuit that provides proper conditioning of the digital signals, pulses decoding and bidirectional counting [27]. Such hardware component is usually called a quadrature decoder [35]. A quadrature decoder is applied for decoding the output of a quadrature encoder and, thus, it counts the pulses, incrementing or decrementing, as determined by the decoded motion direction. An enhanced quadrature decoder can

be supplemented in such a way that it also provides measurement of the elapsed time interval since the recent encoder pulse was acquired.

### 2.2. The Simple M-Method

The simplest way to measure velocity by an incremental encoder is to count encoder pulses generated within a single sampling time interval. If we assume that the position can be obtained by accumulating the counted pulses, then the velocity is obtained as:

$$v_k^M = \frac{x_k - x_{k-1}}{T_s}, \quad (2)$$

where  $x_k$ , and  $v_k^M$  are the acquired encoder positions sampled in the  $k$ -th sampling instant, and the corresponding velocity estimation, respectively.  $T_s$  is the sampling time period. This method works as a frequency counter, producing an output alternating between a discrete number of the counted encoder pulses, which occur asynchronously with the sampling instants. Thus, especially in the low-speed region where only a small number of pulses are generated, the method produces considerable noise in the output due to the spatial quantization given by the encoder pitch (1). However, one can note that the division with the constant sampling period  $T_s$  can be replaced by multiplication with the constant sampling frequency  $F_s = 1/T_s$ . Thus, the M-method is rather simple in terms of computation complexity.

### 2.3. The Conventional MT-Method

The conventional MT-method for velocity measurement can be described as [8]:

$$v_k^{MT} = \frac{x_k - x_{k-1}}{T_s + \delta t_{k-1} - \delta t_k}, \quad (3)$$

where  $x_k$ ,  $\delta t_k$  and  $v_k^{MT}$  are the acquired encoder positions, the elapsed time interval since the recent encoder pulse (sampled in the  $k$ -th sampling instant), and the estimated velocity calculated by the formula, respectively (please see Figure 1 for details).

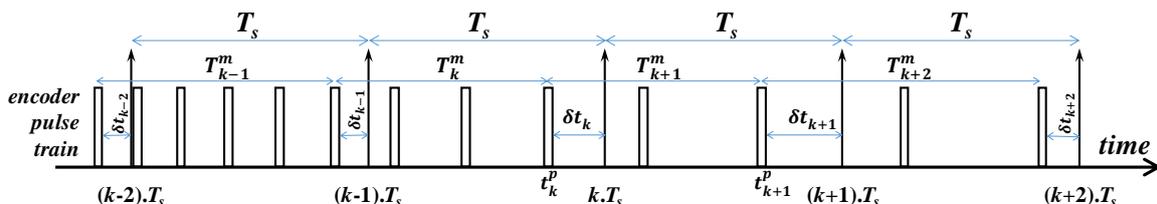


Figure 1. The timing diagram of encoder pulse train with sampling periods.

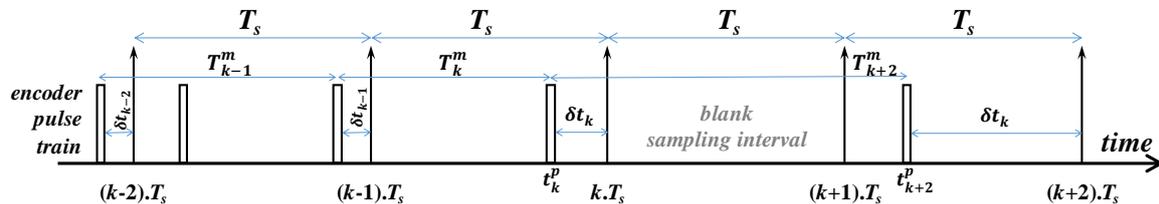
Though the sampling period is normally fixed, the measurement time interval  $T^m$ , which extends between the edges of the first and the last position pulse captured in the measurement window, varies accordingly. Thus, smooth velocity output can be obtained that actually represents accurate average velocity on the measurement interval. However, at low velocity, it may happen that encoder pulses may not occur in every sampling interval, but a single blank sampled period or a series of blank sampling periods may appear before the next encoder pulse occurrence. Then, the previous formula can be rewritten as:

$$v_{k_i}^{MT} = \frac{x_{k_i} - x_{k_i - m_{i-1}}}{m_{i-1} T_s + \delta t_{k_i - m_{i-1}} - \delta t_{k_i}}, \quad (4)$$

where  $m_i = 1 + n_i$ , and  $n_i$  determines the number of blank sampling intervals. It means that if some encoder pulses appear in the  $k_i$ -th sampling interval, then this is followed by  $n_i$  blank sampling intervals, such that:

$$k_{i+1} = k_i + n_i + 1, \quad (5)$$

Thus, between the sampling intervals with encoder pulse occurrence there are  $n_i$  blank sampling intervals. The measurement interval is then extended for the same number of intervals. Note that the velocity estimation formula is calculated at the non-blank sampling intervals only. The timing diagram with rare encoder pulses and a blank sampling interval are illustrated by Figure 2.



**Figure 2.** The timing diagram of encoder pulses widespread with a blank sampling interval.

#### 2.4. The Generalized Divisionless Algorithm for the MT-Type Velocity Estimation Method.

The DivisionLess MT-type (DLMT) velocity estimation algorithm was introduced in [33]. The basic idea of the method is to calculate the velocity in fixed sampling periods, having available exact actual position readings at the sampling time instants. This would result in the average velocity within the sampling period that can be considered as the optimal velocity information. However, incremental encoders cannot provide the actual position at the sampling instants due to the asynchronous nature of the encoder pulses; the correct position evidence can be provided just for the time instant of the encoder pulse occurrence that mismatches the real position at the sampling instants. Therefore, alternative sampling instant position information should be provided. The authors in [33] propose linear extrapolation based on the read encoder position, the velocity and the measured time interval elapsed since the recent encoder pulse was acquired. However, the real velocity information, accurate in time, is not available. Instead, only the velocity estimation can be applied. In advanced motion control applications it is reasonable to assume fast sampling periods such that the velocity change in a single sampling period is close to zero. Therefore, we can use the velocity estimation from the previous sampling period. The algorithm can be described by (6) and (7):

$$\hat{x}_k^a = x_k + \hat{v}_{k-1} \delta t_k, \quad (6)$$

$$\hat{v}_k = \frac{\hat{x}_k^a - \hat{x}_{k-1}^a}{T_s}, \quad (7)$$

where  $\hat{x}_k^a$  is the estimated actual position and the time instant  $t_k = kT_s$ , and  $\hat{v}_k$  is the corresponding estimated velocity. The first equation of the algorithm (6) approximates the real position at the sampling instant, and the second Equation (7) outputs the velocity estimation, which should approximate the average velocity in the recent sampling interval. The algorithm bypasses the arithmetic division efficiently by estimating the actual position at the sampling instant. Then, the pure position difference (similar to the simple M-method) leads to the smooth velocity estimation output, which is close to the conventional MT-method velocity output. However, the algorithm is recursive, since the actual velocity estimation is defined by the previous values, as described by:

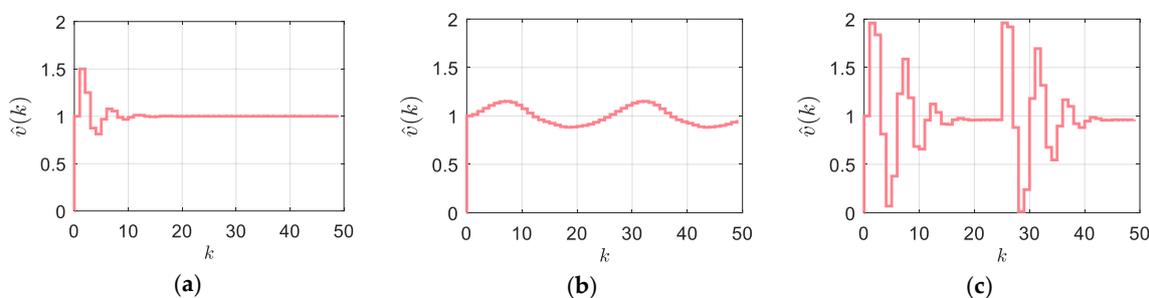
$$\hat{v}_k - \frac{\delta t_k}{T_s} \hat{v}_{k-1} + \frac{\delta t_{k-1}}{T_s} \hat{v}_{k-2} = \frac{\Delta x_k}{T_s}, \quad (8)$$

where the input  $\Delta x_k = x_k - x_{k-1}$  stands for the number of encoder pulses counted in a single sample interval. One can note that the dynamic discrete Equation (8) actually represents a linear discrete filter of a second order with time-varying coefficients, and with its right-hand side that equals the M-method (2). Due to the recursive nature of the algorithm, stability becomes an issue. It has been shown that the

algorithm is asymptotically stable under the condition that the left-hand side coefficients are bounded within interval  $[0,1]$ . The dynamic equilibrium  $\hat{v}_k^{eq}$  of (8), given by:

$$\hat{v}_k^{eq} = \frac{x_k - x_{k-1}}{T_s + \delta t_{k-1} - \delta t_k}, \quad (9)$$

which equals the MT-method output (3), is, therefore, attractive. It means that the DLMT algorithm output converges to the accurate velocity estimation. The step response of the filter dynamics (8) is illustrated by Figure 3 with three cases of  $\delta t_k/T_s$  such that it is: (a) Constant and equals 0.5, (b) Variable by sinus waveform of two periods with the amplitude of 1 and offset of 0.5, and (c) Variable by two periods of the sawtooth waveform that varies in the interval  $[0,1]$ . The asymptotically stable response is evident in the case of (a). In the case of (b), the filter generates a sinusoidal output to provide stable convergence to the dynamic equilibrium defined by (9). In the case of (c), we can observe high velocity peaks when the sawtooth waveform generates the abrupt changes from 0 to 1 (sudden change from  $\delta t_k = 0$  to  $\delta t_{k+1} = T_s$ ), which are followed by the asymptotically stable convergence, although it is oscillatory. However, it should be noted that this scenario is highly unrealistic in practice.



**Figure 3.** Step response of the DLMT filter: (a) constant  $\delta t_k/T_s = 0.5$ ; (b) sinusoidal variation of  $\delta t_k/T_s$ ; (c) sawtooth variation of  $\delta t_k/T_s$ .

Though the theoretical treatment of the DLMT algorithm in [33], along with the numerical cases, shows stable convergence, it does not address a scenario with blank sampling intervals, which appear at low velocity. Let encoder pulses appear in the  $k_i$ -th sampling interval, which is then followed by  $n_i$  blank sampling intervals. If we consider that the velocity estimation is updated only in non-blank sampling intervals, and if we derive the equation of the estimated actual position at the  $(k_i - 1)$ -th sampling interval as:

$$\hat{x}_{k_i-1}^a = \hat{x}_{k_i-1}^a + \hat{v}_{k_i-1}(n_{i-1}T_s), \quad (10)$$

where  $\hat{x}_{k_i-1}^a = x_{k_i-1} + \hat{v}_{k_i-2}\delta t_{k_i-1}$ , then the output velocity estimation (as in (7)) can be derived as:

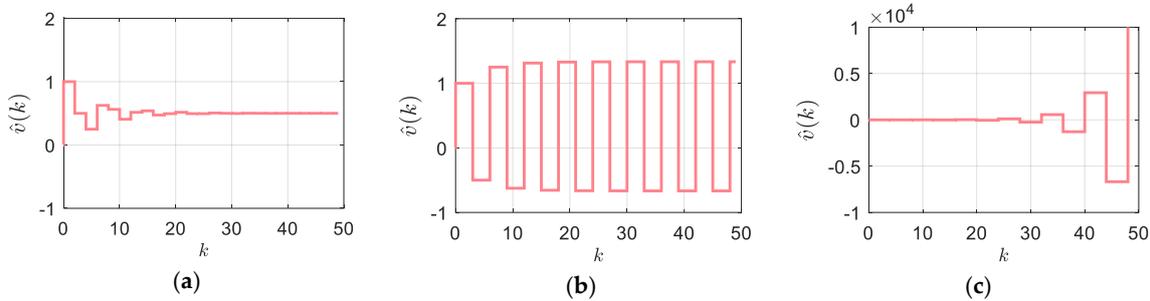
$$\hat{v}_{k_i} = \frac{\hat{x}_{k_i}^a - \hat{x}_{k_i-1}^a}{T_s} = \frac{(x_{k_i} - x_{k_i-1}) + \hat{v}_{k_i-1}(\delta t_{k_i} - n_{i-1}T_s) - \hat{v}_{k_i-2}\delta t_{k_i-1}}{T_s}, \quad (11)$$

If we rearrange the upper final equation and, furthermore, simplify the notation such that we index by  $(\cdot)_i$  instead of  $(\cdot)_{k_i}$ , then the discrete dynamics Equation (8) in case of some blank sampling intervals transforms into the recursive discrete equation of the second order (12) that updates in irregular intervals at  $t_i = t_{k_i} = k_i T_s$ , where  $k_i = k_{i-1} + n_{i-1} + 1$ .

$$\hat{v}_i - \left(\frac{\delta t_i}{T_s} - n_{i-1}\right)\hat{v}_{i-1} + \frac{\delta t_{i-1}}{T_s}\hat{v}_{i-2} = \frac{\Delta x_i}{T_s} \quad (12)$$

A comparison of (12) with (8) yields the difference in the second coefficient on the left-hand side of the latter equation. While the coefficients on the left-hand side of (8) are bounded by the interval  $[0,1]$  in the absolute sense, this is not the case in (12), since  $n_{i-1} = [0|1|2|\dots]$  appears in the coefficient of  $\hat{v}_{i-1}$ . It can be shown that this seriously violates the stability conditions, which are presented in [33].

Thus, the algorithms (6), (7) cannot provide asymptotically stable velocity estimation in a general case, and therefore, must be appropriately modified. Instability of the DLMT filter (12) in the case with blank sampling intervals is shown by Figure 4, with the asymptotically stable response at  $n_i = 1$  (a), the limited stability at  $n_i = 2$  (b), and the unstable response at  $n_i = 3$  (c).



**Figure 4.** Step response of the DLMT filter in the case of blank sampling intervals (constant  $\delta t_k/T_s = 0.5$ ): (a)  $n_i = 1$ ; (b)  $n_i = 2$ ; (c)  $n_i = 3$ .

In this paper, we introduce an adaptation of the DLMT algorithm to provide an asymptotically stable output in the generalized case that involves blank sampling intervals. The Generalized DivisionLess algorithm for MT-type (GDLMT) velocity estimation method is given by (13) and (14):

$$\hat{x}_i^a = x_i + \hat{v}_{i-1}\delta t_i, \quad (13)$$

$$\hat{v}_i = \frac{\hat{x}_i^a - \hat{x}_{i-1}^a}{m_{i-1}T_s}, \quad (14)$$

It can be shown easily that the proposed GDLMT algorithm cancels the  $n_{i-1}$  term in the second coefficients of the left-hand side of (12). Thus, the dynamics equation can be rewritten such that it yields:

$$\hat{v}_i - \frac{\delta t_i}{m_{i-1}T_s}\hat{v}_{i-1} + \frac{\delta t_{i-1}}{m_{i-1}T_s}\hat{v}_{i-2} = \frac{\Delta x_i}{m_{i-1}T_s}, \quad (15)$$

which satisfies the conditions for asymptotically stable convergence to the dynamic equilibrium:

$$\hat{v}_i^{eq} = \frac{1}{1 - \frac{\delta t_i}{m_{i-1}T_s} + \frac{\delta t_{i-1}}{m_{i-1}T_s}} \frac{\Delta x_i}{m_{i-1}T_s} = \frac{x_i - x_{i-1}}{m_{i-1}T_s - \delta t_i + \delta t_{i-1}}. \quad (16)$$

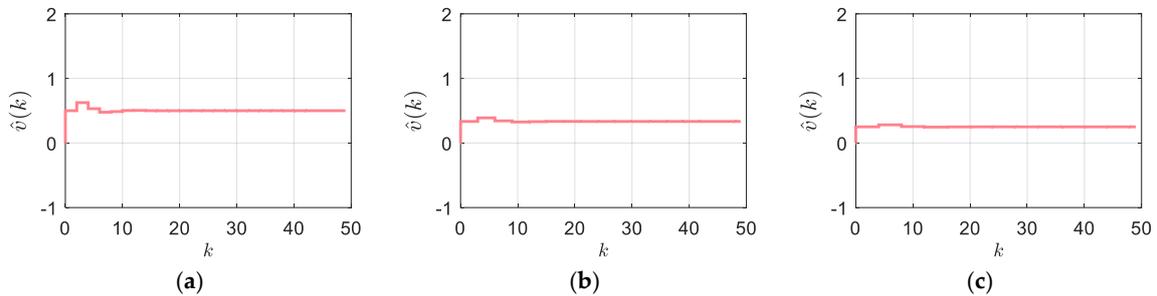
By comparison of (16) and (4) we can conclude that the dynamic equilibrium of the GDLMT method is equivalent to the output of the MT-method. The step response of the GDLMT filter (15) is depicted by Figure 5 with the asymptotically stable response in all cases of a number of blank intervals ( $n_i = 1$  (a),  $n_i = 2$  (b), and  $n_i = 3$ (c)).

One can note that the GDLMT algorithm involves the variable term  $m_{i-1}T_s$  in the denominator of the right-hand side of the output Equation (14). It implies on-line arithmetic division, which cannot be substituted simply by multiplication with a single fixed inverse value. However, for a bounded set of  $m_i = [1|2| \dots |m_{\max}]$  we can calculate the inverse values off-line, since all the possible values of  $m_i$  represent a set of constants known in advance. It is easy to calculate the inverse values of  $(m_iT_s)^{-1}$  for a limited number of  $m = 1, 2, \dots, m_{\max}$  off-line and store the results into a memory, e.g., as a zero-indexed array (during the on-line computation, the values of  $n_i$  can be used as the index). Then the output formula (14) can be read as:

$$\hat{v}_i = (m_{i-1}T_s)^{-1}\Delta\hat{x}_i^a, \quad (17)$$

where  $\Delta\hat{x}_i^a = \hat{x}_i^a - \hat{x}_{i-1}^a$  and  $(m_iT_s)^{-1}$  can be recalled from the memory during the on-line execution of the algorithm. At the bottom line, the arithmetic division involved in the GDLMT algorithm can be

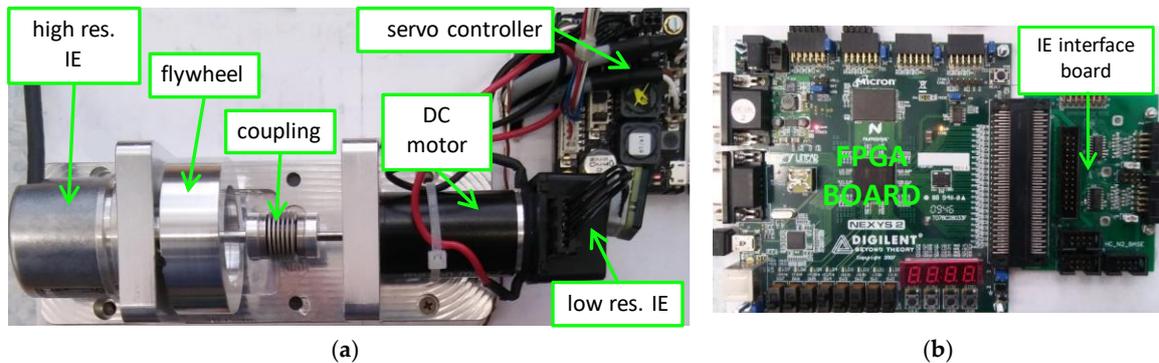
replaced by the arithmetic multiplication in on-line execution with real-time performance. However, note, that  $m_{\max}$  determines the minimal measurable velocity. If overrun of  $m$  happens (such that  $m > m_{\max}$ ), then the estimated velocity can be set to zero.



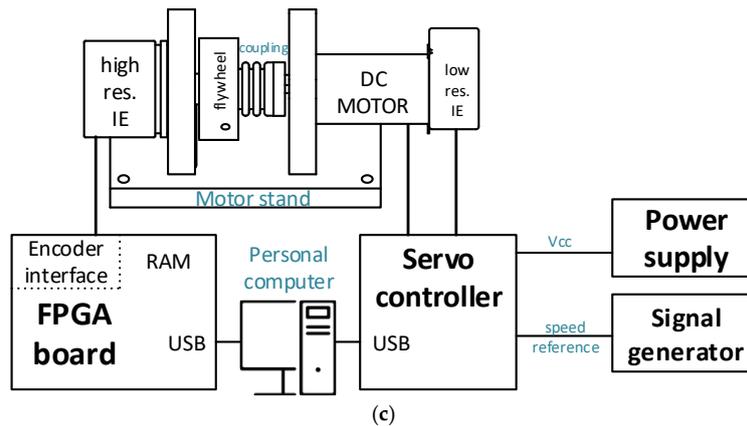
**Figure 5.** Step response of the GDLMT filter in the case of blank sampling intervals (constant  $\delta t_k/T_s = 0.5$ ): (a)  $n_i = 1$ ; (b)  $n_i = 2$ ; (c)  $n_i = 3$ .

### 2.5. The Experimental System

The experimental setup consists mainly of a mechanical drive system with a rotary incremental encoder (IE), and the necessary processing hardware electronics. It is shown in Figure 6. For the processing hardware, we employed a Digilent Nexys2 Board [36] with a cost optimized Xilinx Spartan XC3S1200E FPGA that is designed as a high-performance digital circuit for high-volume, consumer-oriented applications. Besides that, our board also includes lots of memory (16 MB Micron Cellular PSRAM, 16 MB Flash ROM, and additional Flash ROM for nonvolatile FPGA configurations), oscillators (50 MHz in-circuit oscillator and an additional oscillator of 125 MHz with 50 ppm in the socket for custom crystal oscillator), a high-speed Hirose FX2 expansion connector, four PMOD connectors with 8 I/O lines each, and a USB 2.0 high-speed port for the communication link to the development PC, among others. The FPGA digital circuit features numerous logic cells, distributed RAM bits, blocks of RAM, hardware multipliers with 18-bit inputs, digital clock managers, etc.



**Figure 6.** Cont.



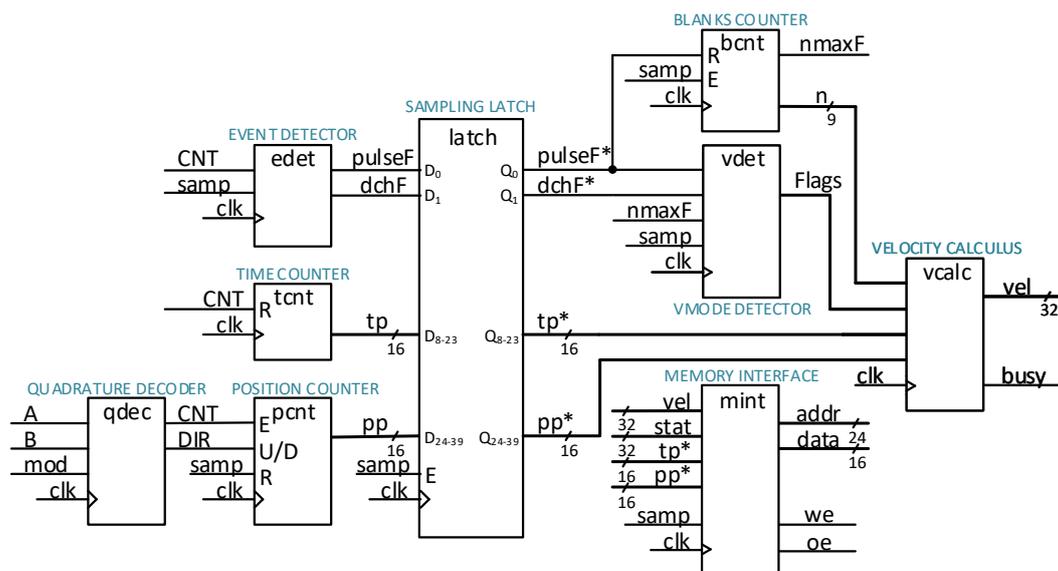
**Figure 6.** The experimental setup: (a) the mechanical drive system with the IE; (b) the processing hardware electronics with the FPGA; (c) the principal interconnection block scheme.

The mechanical drive system consists of a DC motor Maxon RE 25 with low-resolution incremental encoder HEDS 5540 (with 500 line counts per turn), mounted on the rear side of the motor shaft, and a high resolution Heidenhain ROD 1020 rotary incremental encoder (with 3600 line counts per turn) is linked to the front-side of the motor shaft. The latter was used for the experimental evaluation and validation of the proposed method for velocity estimation. The motor shaft and the high-resolution incremental encoder shaft are interconnected mechanically by a metal bellows coupling. Additionally, we attached a rotational flywheel on the encoder shaft in order to increase its rotational inertia, which can, consequently, reduce mechanical oscillations. The flywheel, along with the testing incremental encoder, represents the motor dynamical load. In order to damp further the mechanical disturbances from the motor side, which were demonstrated during our experimental tests by higher frequency motion oscillations, we replaced the metal coupling with a soft silicon tube in some experiments, which, on the other hand, cannot provide the proper stiffness required for highly dynamical motion. Thus, the mechanical system transformed into an elastic two-mass system and, therefore, an additional low frequency fluctuating motion occurred on the load side. The DC motor was driven electrically by a Maxon ESCON 36/2 DC Servo Controller configured in speed control mode with the low-resolution encoder in the loop. The driving signal of reference speed, connected to its analog input, was provided by the Rigol DG 1022 function generator.

The electrical lines of the high-resolution incremental encoder were connected to the FPGA board by a custom interface board on the high-speed expansion connector. The interface board contains standard RS 422 line receivers, proper analog filters and line termination. The enhanced quadrature decoder, with a clock frequency of 125 MHz, was implemented in the FPGA circuit. Thus, capture of encoder pulses and relatively accurate measurement of time intervals were performed with a resolution of 8 ns. The sampling frequency, by which we read encoder pulses and the elapsed time interval since the recent encoder pulse, was 10 kHz. The update of the velocity estimation output was performed synchronously with the sampling. The data were stored to the on-board RAM every sampling period in real-time. Thus, after an experiment was completed, the data were available for transfer in batch via the USB port to the desktop PC and stored for further off-line post processing.

The calculus of the velocity estimation algorithm was designed in a fixed-point format, and implemented by integer arithmetic operations. The inputs in the algorithm sampled by the period of  $T_s = 0.1$  ms are: (a) Integer number of encoder position pulses (position pulses abbreviated as pp), and (b) Integer number of clock ticks counted from the recent encoder pulse (time pulses abbreviated as tp). The longest measurable time interval in the current implementation is limited by the 16-bit register and the clock frequency of 125 MHz, that yield around 0.5 ms. Additionally, we can count up to 511 blank sampling periods. The velocity output is provided in a 32-bit signed Q-format with 18 bits for the integer part (including the sign bit) and 14 bits for the fractional part. It is given in

units of the number of encoder position pulses per sampling period ( $pp/T_s$ ). The algorithm was coded in VHDL. The calculation of the algorithm was organized in sequential steps such that it performs in only a few clock cycles. For the multiplication arithmetic operations, we employed the hardware multipliers embedded in the FPGA, while the addition and subtraction operations were performed by the general FPGA logic cells. Normally, the calculus is performed every sampling period if at least a single encoder pulse occurs. Otherwise, the velocity is not updated and an old value is preserved at the output. If the count of blank sampling periods exceeds a maximum number, or in a case of detected direction change, then the output is forced to zero. The calculus of the velocity can be re-enabled when two sampling periods with encoder pulses in the same direction appear. The principal core framework logic circuit design implemented on the FPGA, is shown by Figure 7.



**Figure 7.** The block scheme of the velocity measurement system logic design on the FPGA.

All the modules featured by the block scheme operate synchronously by the basic clock signal 'clk', and 'samp' is the trigger signal generated at the sampling instants in the duration of a single clock cycle. The modules left from the 'latch' represent the capturing section with the encoder signals as the inputs 'A' and 'B', whereas the modules on the right side process the signals sampled by the 'latch'. The sampled signals (and the processed signals as well) are logged promptly to RAM. The result of the processing section is the estimated velocity 'vel'. The 'QUADRATURE DECODER' reads encoder signals every clock cycle, and generates 'CNT', which reflects a position pulse to be counted, and 'DIR' that determines the counting direction (incrementing or decrementing). We can select two modes of decoding, i.e., in the 'X1' mode, the number of position pulses generated on the decoder output in one revolution equals the encoder line counts, whereas in the mode 'X4' the number of the position pulses are quadruple. The 'POSITION COUNTER' then counts the position pulses. The 'TIME COUNTER' measures the elapsed time interval since the recent position pulse by counting the clock cycles. The 'EVENT DETECTOR' module generates the event flags for "motion direction change" (the 'dchF' output) and for "position pulse occurred" (the 'pulseF' output) in a recent sampling period. The output data from the above-mentioned modules are captured by the 'latch' module at the sampling instants. The latched signals are denoted with a supplemented asterisk (\*). The 'BLANKS COUNTER' counts blank sampling periods (a contiguous series of sampling periods with no position pulse occurrence), which is provided at the 'n' output. The 'nmaxF' is the output flag, which signals that the 'n' has reached a maximum value. The latched pulse flag 'pulseF\*' reset the value of the counted blanks. Then the 'VMODE DETECTOR' module generates velocity mode flags that are required for the control logic of the arithmetic module 'VELOCITY CALCULUS'. The latter performs calculation of the velocity estimation using counted position pulses ('pp\*'), counted time

pulses ('tp\*'), and counted blank sampling periods ('n\*'). The 'busy' output flag indicates the duration of the calculus. The sequence of positive integers' reciprocals ( $1/1, 1/2, 1/3, \dots, 1/511$ ) was calculated off-line, and organized as a look-up table in the embedded block RAM. The size of the look-up table limits the maximum number of counted blank periods.

The timing diagram, which includes the interval required for the calculus and writing the data to the RAM (by the 'MEMORY INTERFACE' module), is shown by Figure 8. The logic signals from the figure were captured by a Tektronix MSO2014 oscilloscope (1 Gsps). The velocity estimation algorithm starts soon after the fresh readings are available for the processing. It provides the result in 64 ns. Thus, the output is available in less than 100 ns since the positive edge of the sampling instant (the 'samp' trigger pulse). Writing to the RAM, which is enabled by the low level of the write-enable ('we') signal, starts already during the computation phase. It transfers the data word-by-word in the following sequence of 8 words: a single word of the 'pp\*' data, a single word of the 'tp\*' data, two reserved words, a double word of 'stat' data, and a double word of 'vel' data.

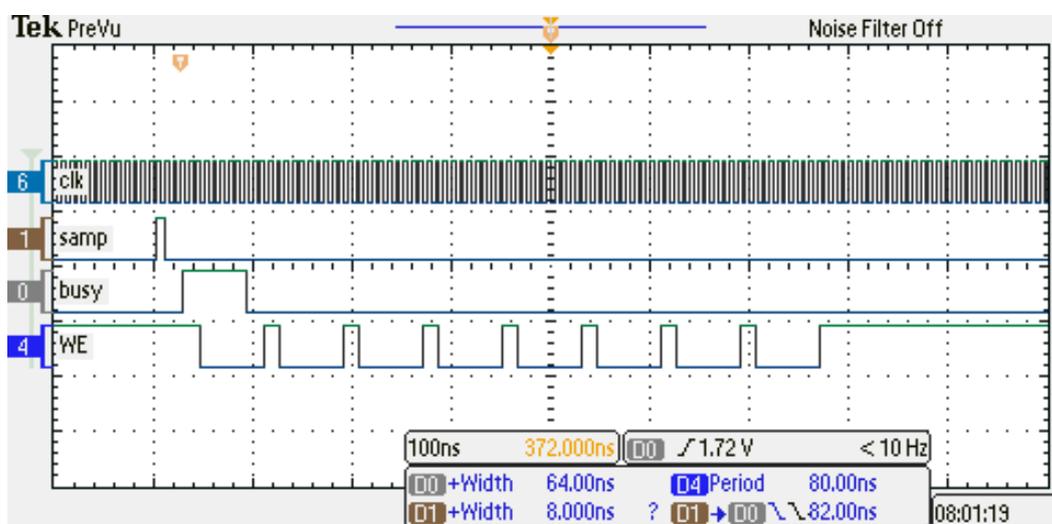


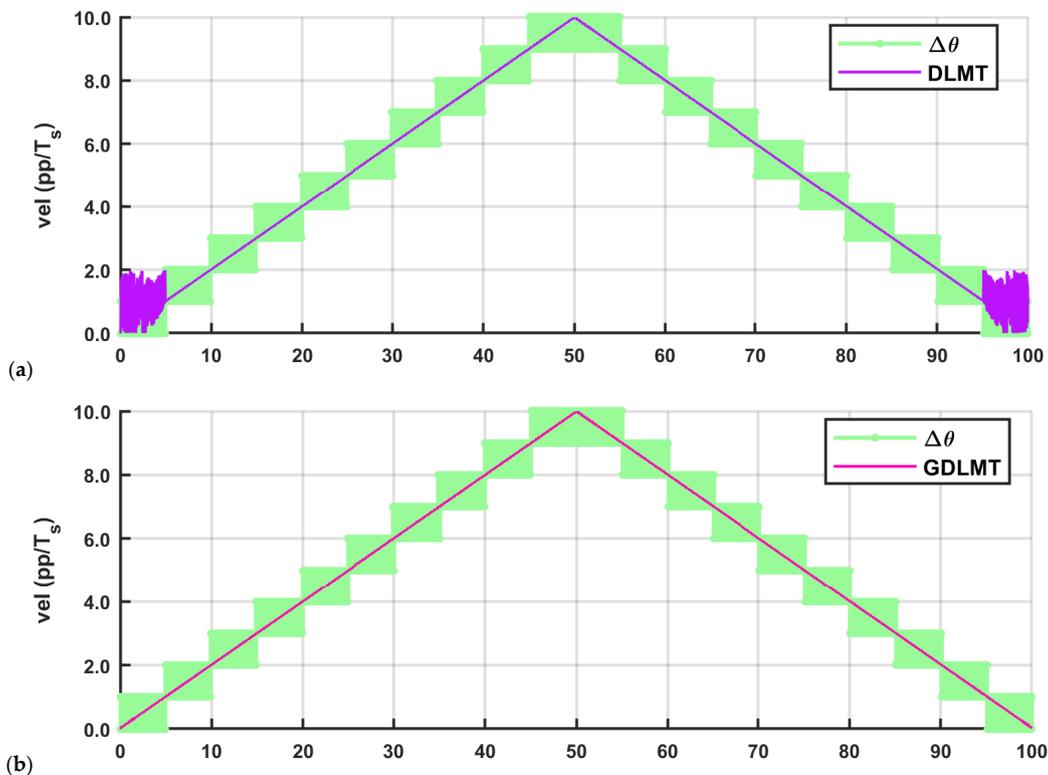
Figure 8. The timing diagram of the core FPGA circuit.

The consumption of the FPGA resources for the experimental design solution presented above is relatively small. For the calculation process, we used 5 dedicated  $18 \times 18$  hardware multipliers, two blocks of RAM, and the total number of occupied logic cells was less than 5%.

### 3. Experimental Results

We performed various experiments in a wide speed range in order to test and validate the velocity measurement by the proposed GDLMT algorithm implemented on the FPGA board. The experiments covered the range from high speed to low speed. We consider high speed in the case of a large number of encoder pulses per sampling period, e.g., 50 pulses and more, and low speed is about 1 pulse per sampling period or below. In order to achieve high-speed motion, we employed the motorized drive by a ramp or a trapezoidal speed profile with a constant acceleration/deceleration phase, respectively. The motor shaft and the encoder shaft with the flywheel were linked by a metal bellows coupling. However, due to considerable system disturbance from the motor side that appears as undesired high-frequency oscillations in motion, we replaced the stiff coupling with a soft silicon tube in some experiments. Though it resulted in smoother motion, low-frequency fluctuations appeared due to the soft coupling in the two-mass system with the motor rotor on one side and the flywheel on the other side. In spite of the disturbing motion oscillations and fluctuations, the velocity estimation could provide proper velocity information in all circumstances. Nevertheless, we performed low-speed experiments driving the encoder only by hand, i.e., manually, or by a gravitational load, in order to reduce disturbances even further and to achieve a smooth motion.

Figure 9 depicts the experimental results with a ramp speed profile (motion oscillations are not shown due to the relatively large scale). We compared the performance of the pure DLMT algorithm, and the proposed GDLMT algorithm with the adaptation for stable performance at low-speed. The GDLMT algorithm was computed on-line on the FPGA platform by integer arithmetic operations with data numbers in an appropriate Q-format. The DLMT velocity was computed on the desktop computer for off-line post-processing by single floating-point arithmetic. In both cases, the input data were the same. The velocity results are given in “position pulses per sampling period” ( $pp/T_s$ ).  $\Delta\theta$  denotes the integer number of encoder pulses read in a single sampling period, i.e., the simple M-method (the light green curve trace). It is characterized by stepping and alternating, as expected. In both diagrams of Figure 9, the MT-type velocity estimations show smooth performance in the whole speed range except at low speed below  $1 pp/T_s$ . In this range, the DLMT method (the purple curve trace) demonstrated highly oscillating behavior with an erroneous result, whereas the proposed GDLMT algorithm (the pink curve trace) shows stable performance clearly, similar as in the speed range beyond  $1 pp/T_s$ .



**Figure 9.** Comparison of the DLMT method and the GDLMT method at ramp speed: (a) pure DLMT method; (b) GDLMT method.

The next three figures show the experimental results obtained by a trapezoidal profile at high speed of  $50 pp/T_s$  (with the stiff metal bellows coupling, Figure 10, at medium speed of  $5 pp/T_s$  (with the soft silicon coupling, Figure 11, and at low-speed of  $0.5 pp/T_s$  (with the soft silicon coupling, Figure 12, respectively. With these figures we provide mainly a comparison with the conventional MT-method (the dark red curve trace), which was computed on the desktop computer for off-line post-processing with the same input data as in the case of the GDLMT algorithm (the pink curve trace) implemented on the FPGA platform. The upper diagrams show the velocity traces including the simple M-method (the light green curve trace), while the bottom diagrams provide the velocity estimation error, which is computed such that the MT-method is considered as a reference (i.e., the difference between the MT-velocity trace and the GDLMT-velocity trace). On every Figure we added some characteristic details in zoom: (i) Detail 1 with a velocity overshoot (peak velocity), (ii) Detail 2 with

a velocity zero crossing, (iii) Detail 3 with a ramp velocity at constant acceleration, and (iv) Detail 4 with a constant velocity. Figure 10 shows the results at high speed. The estimation error was below  $0.001 \text{ pp}/T_s$ , except around the velocity reversals, with error peaks that could exceed  $0.01 \text{ pp}/T_s$ . The lowest error was in the case of the constant velocity (Figure 10e). Figure 11 shows the results at medium speed. The estimation error was very low, not exceeding only a few ten-thousandths  $\text{pp}/T_s$  during the whole experiment interval. Within the selected details, the error stayed below  $0.001 \text{ pp}/T_s$  practically in all cases (with a slight exception in the case of zero crossing). The lowest error was again in the case of the constant velocity (Figure 11e). In the low speed experiment (Figure 12), the error remained even below  $0.0005 \text{ pp}/T_s$ . However, the velocity traces show slowly fluctuating motion that arose due to the soft shaft coupling.

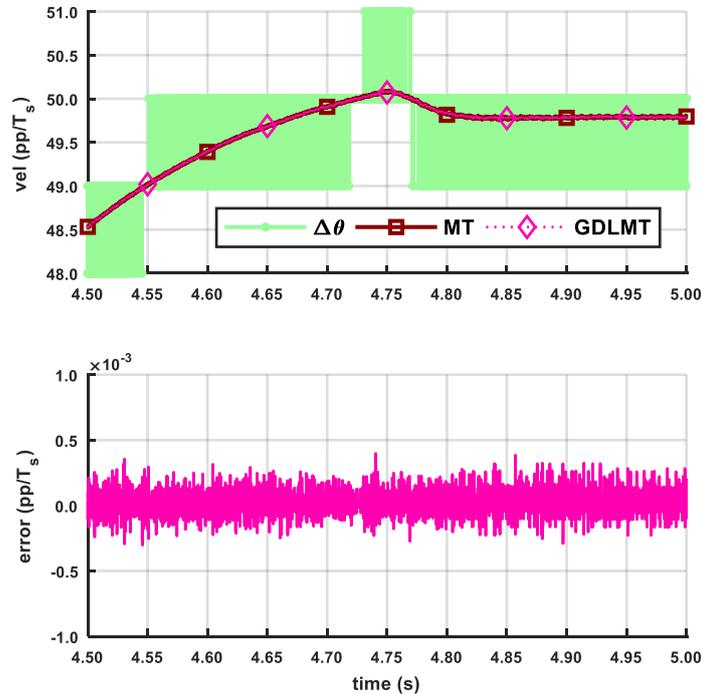
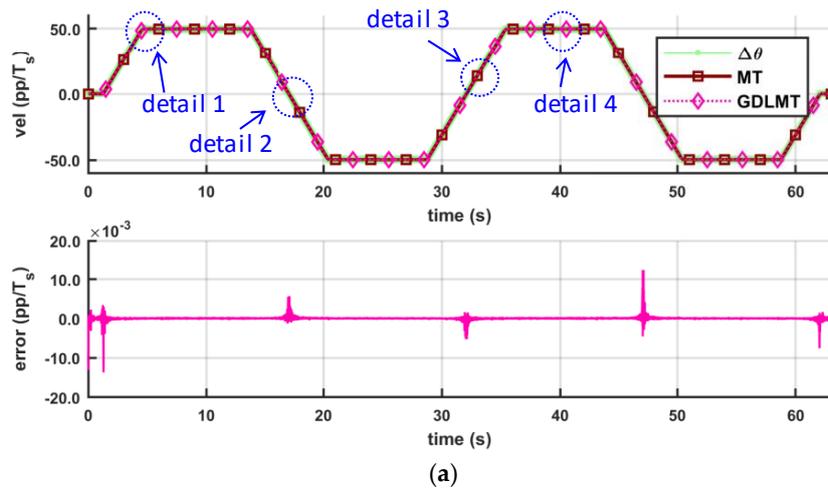
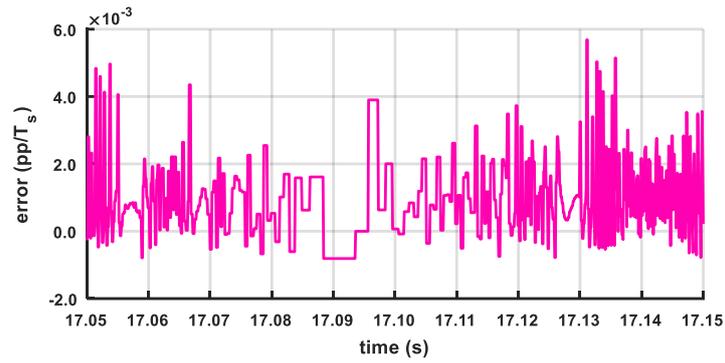
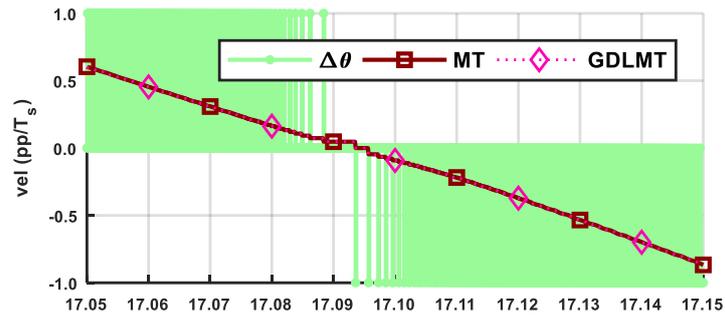
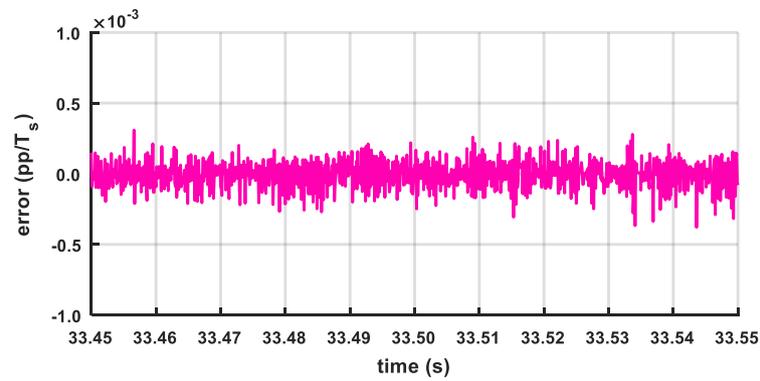
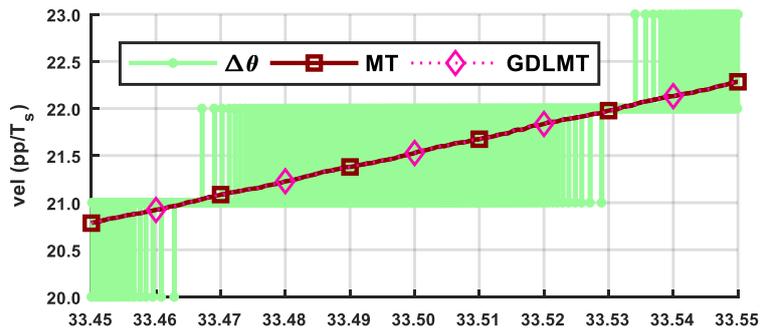


Figure 10. Cont.

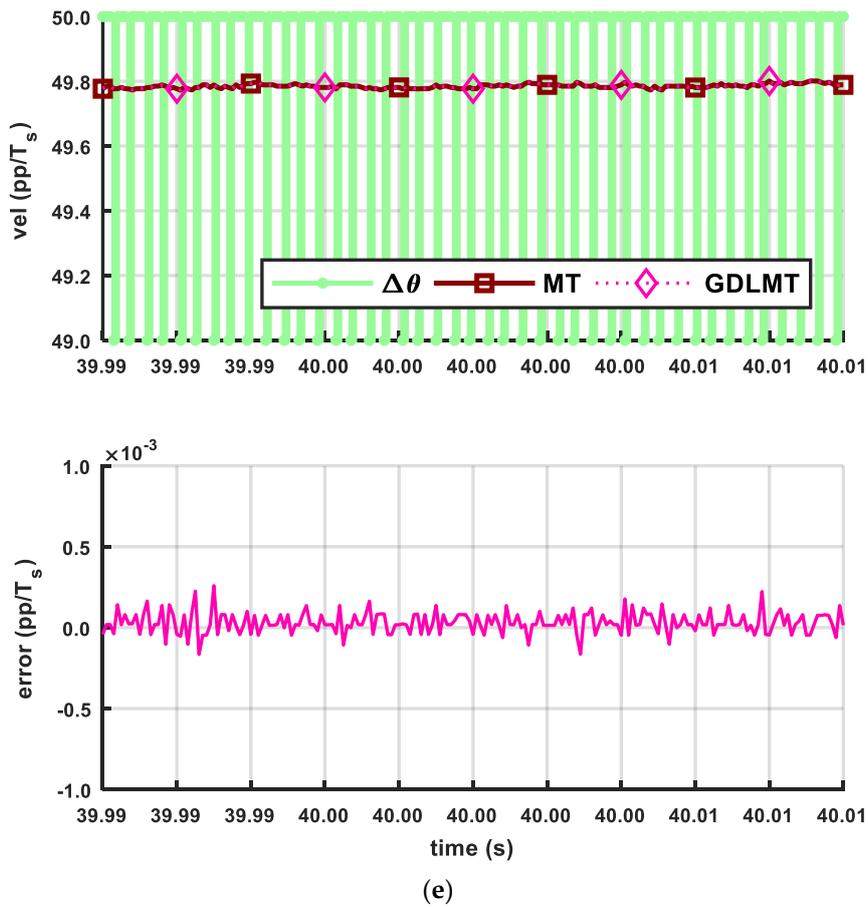


(c)

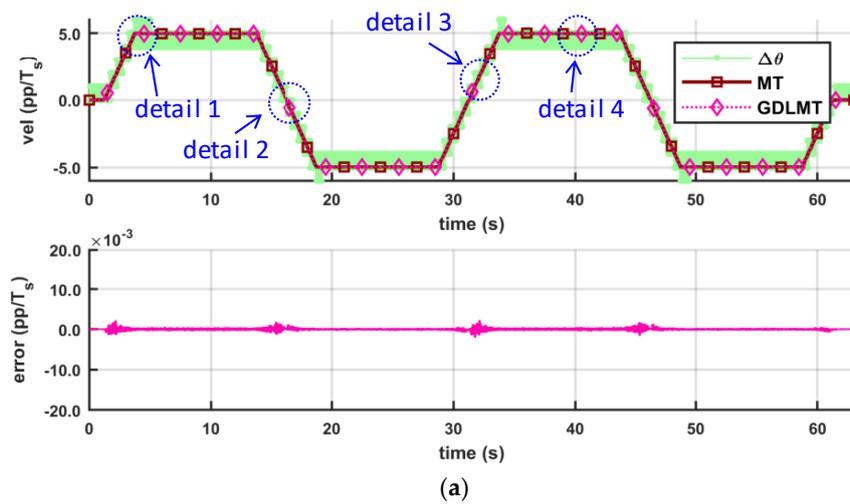


(d)

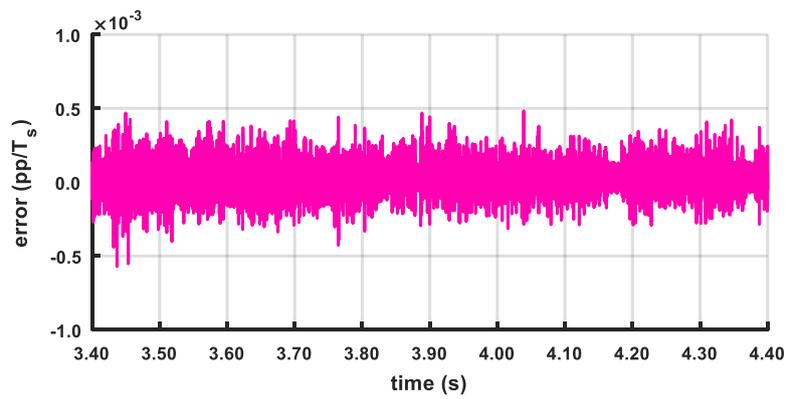
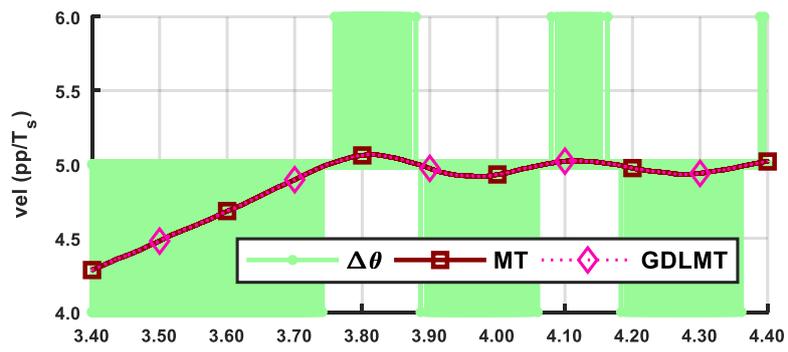
Figure 10. Cont.



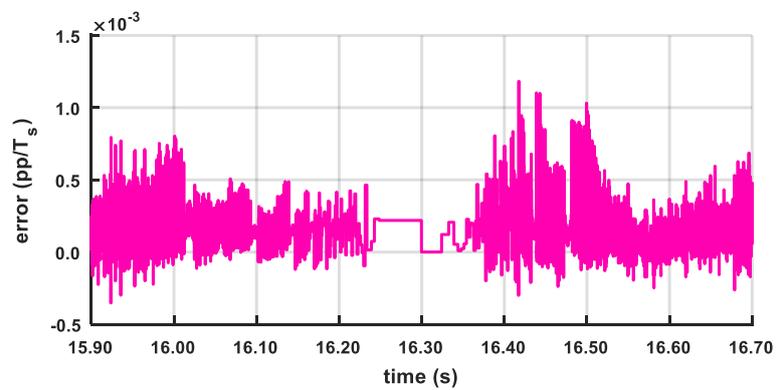
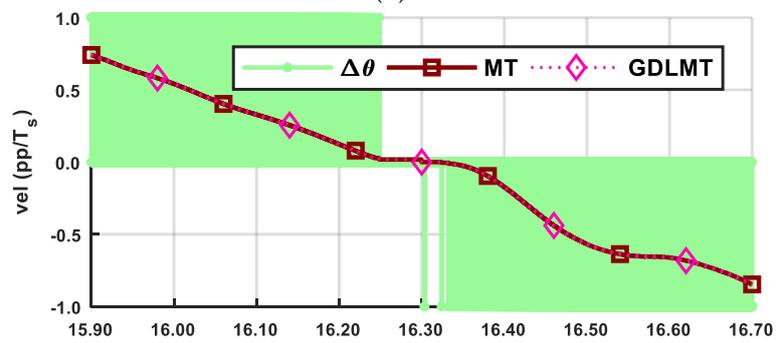
**Figure 10.** Comparison of the MT method and the GDLMT method at high speed: (a) full time interval; (b) detail 1 with velocity overshoot (peak velocity); (c) detail 2 with velocity zero crossing; (d) detail 3 with ramp velocity (constant acceleration), and (e) detail 4 with constant velocity.



**Figure 11.** Cont.

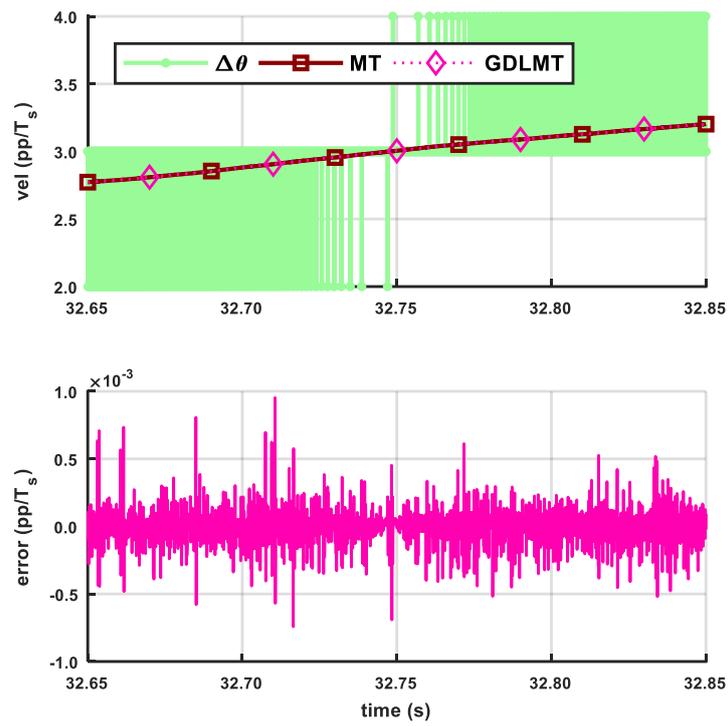


(b)

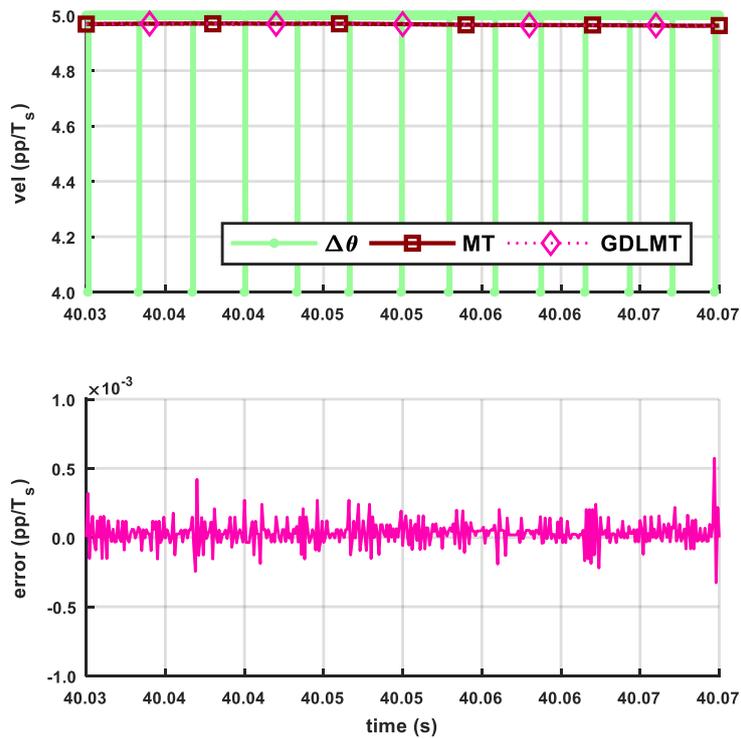


(c)

Figure 11. Cont.



(d)



(e)

**Figure 11.** Comparison of the MT method and the GDLMT method at medium speed: (a) full time interval; (b) detail 1 with velocity overshoot (peak velocity); (c) detail 2 with velocity zero crossing; (d) detail 3 with ramp velocity (constant acceleration), and (e) detail 4 with constant velocity.

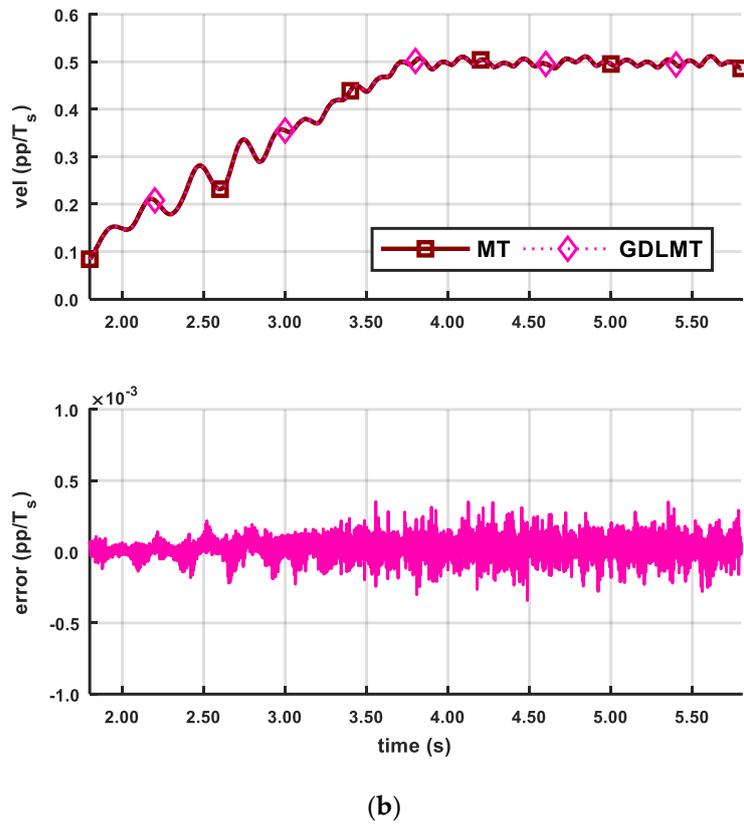
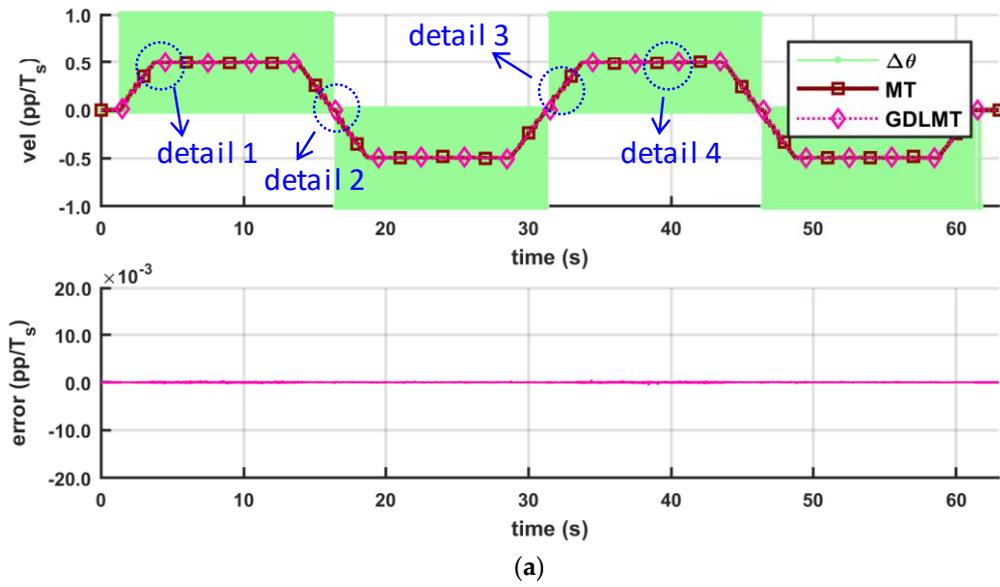
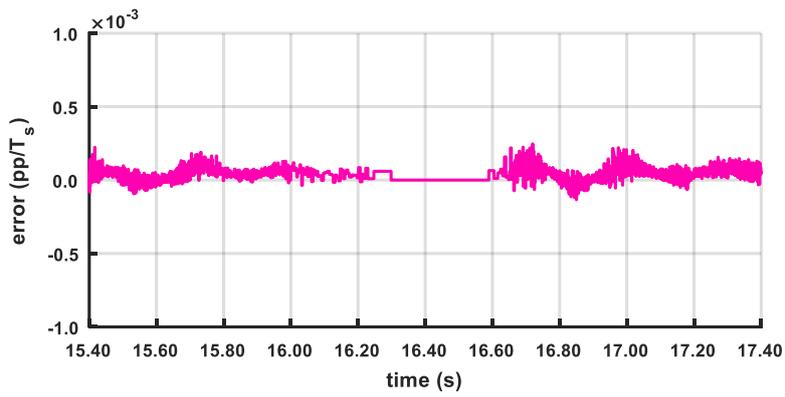
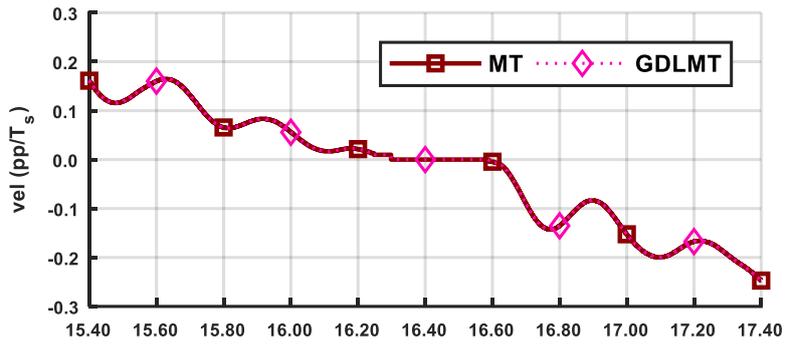
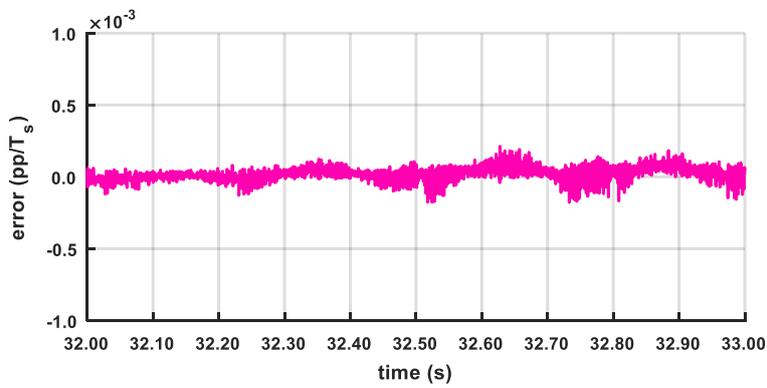
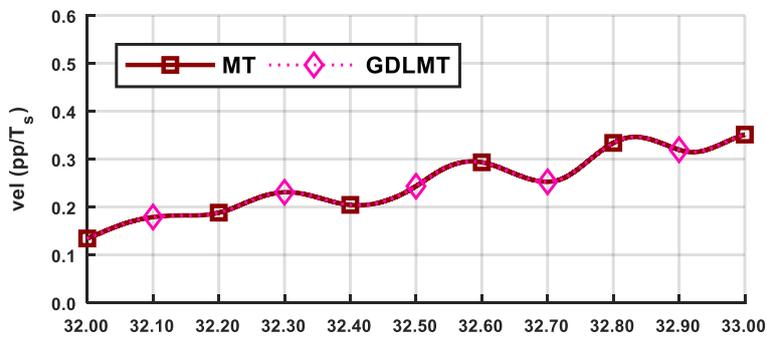


Figure 12. Cont.

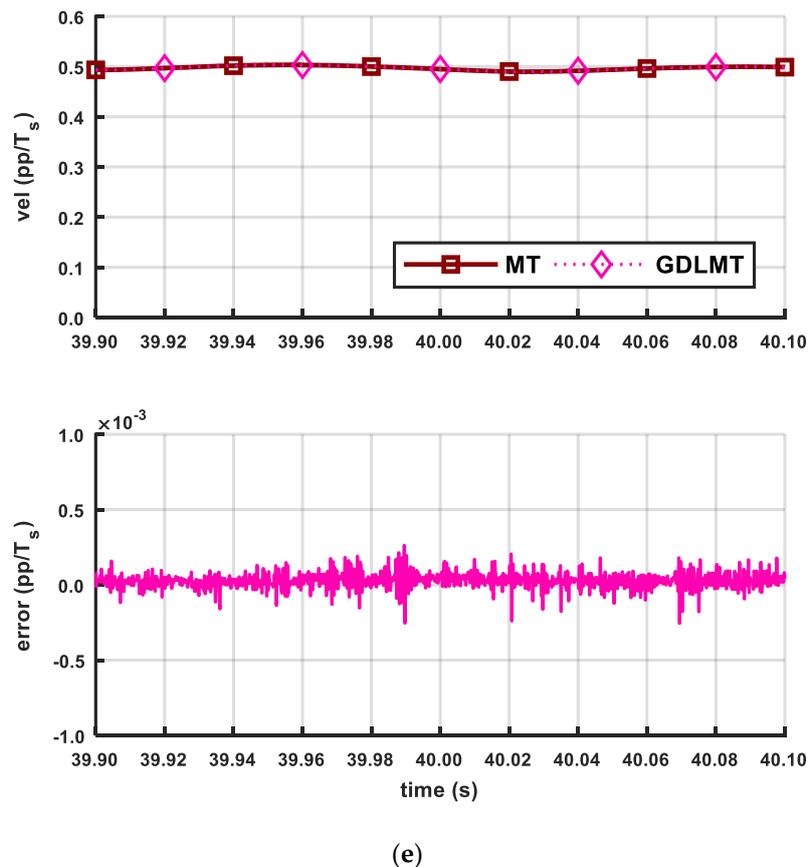


(c)



(d)

Figure 12. Cont.



**Figure 12.** Comparison of the MT method and the GDLMT method at low speed: (a) full time interval; (b) detail 1 with velocity overshoot (peak velocity); (c) detail 2 with velocity zero crossing; (d) detail 3 with ramp velocity (constant acceleration), and (e) detail 4 with constant velocity.

The velocity curve traces of the MT-method and of the GDLMT-method practically overlap in all cases, thus, relatively high estimation accuracy was preserved. Nevertheless, the stability of the GDLMT algorithm performance was not jeopardized in any case.

Further experiments were performed driving the encoder shaft with the flywheel manually or by gravitational load. Thus, we removed the main source of the disturbance and achieved highly smooth motion. The results shown by Figure 13 are organized by a  $3 \times 3$  matrix. In the three columns we show the experiments as follows: (i) The first column shows the experiment with abrupt manual start up to the medium speed, which is then followed by self-slowdown at moderate deceleration (Figure 13a,d,g); (ii) The second column shows the experiment with manual driving of the encoder shaft obtaining sinusoidal-like motion at low speed (Figure 13b,e,h); and (iii), The third column shows the near-constant very low speed experiment with a vertically hanging gravitational load on a string, which linked the load over the pulley to the encoder shaft (Figure 13c,f,i). In the three rows, we show: (i) The comparison of the GDLMT algorithm (computed on-line on the FPGA platform) with the conventional MT-method (computed off-line on the desktop computer); (ii) The comparison of the FPGA implementation of the GDLMT algorithm (GDLMT(fpga)) with the GDLMT-method computed in a single floating-point format off-line on the desktop computer (GDLMT(comp)); and (iii) Comparison with the so-called actual velocity obtained by a non-causal discrete zero-phase low-pass filter (similar to the approach presented in [12]), and the velocity estimation, which is normally provided in conventional motion control applications by real-time low-pass filtering of encoder pulses read in a single sampling period (filtering the M-method). The upper diagrams show velocity traces, whereas the bottom diagrams feature error traces. In the case of comparison (iii), the filtered traces were obtained by off-line post-processing the stored data on the desktop computer, and both filters

performed on the recorded data of the M-type velocity were designed by Butterworth low-pass filter of 2nd order: (a) The non-causal zero-phase filter with the cut-off frequency of 50 Hz, and (b) The causal phase-lag filter with the cut-off frequency of 20 Hz, respectively. The error traces were computed: (i) in the first row as the difference between the MT-velocity trace and the GDLMT-velocity trace; (ii) in the second row as the difference between the GDLMT (comp)-velocity trace and the GDLMT (fpga)-velocity trace; and (iii) in the third row as the difference between the so called actual velocity trace and the GDLMT-velocity trace.

In all the experimental results which are depicted by Figure 13a–c in the first row, the MT-velocity (dark red curve trace) and the GDLMT-velocity (pink curve trace) practically overlap (thus, in the third column experiment with the near-constant low speed, the GDLMT velocity trace is plotted by an inverse sign in order to provide better transparency of the plots). We can observe extremely low velocity estimation error, which increased only in the case of high acceleration in Figure 13a or in the case of high deceleration in Figure 13c.

In the second row of Figure 13 we can observe a comparison of the FPGA-computed GDLMT-velocity (pink curve trace) with the GDLMT-velocity computed off-line in a single floating-pointing format (light red curve trace). The error traces were computed off-line as well, such that the computer GDLMT-velocity was considered as a reference. Again, the velocity traces practically overlap (as shown by Figure 13d–f, and the error is kept close to zero (below  $0.0005 \text{ pp}/T_s$ ) in all the experiments.

In the last row, (Figure 13g,h,i), we compare the GDLMT-velocity (pink trace curve) with the so-called actual velocity (denoted as “vel”, and plotted with a dark green curve trace) and the filtered M-velocity (denoted as “ $\Delta\theta_{\text{flt}}$ ”, and plotted with a normal green curve trace). We also provide the error computed by the so-called actual velocity as a reference used in the comparison with the on-line computed GDLMT velocity.

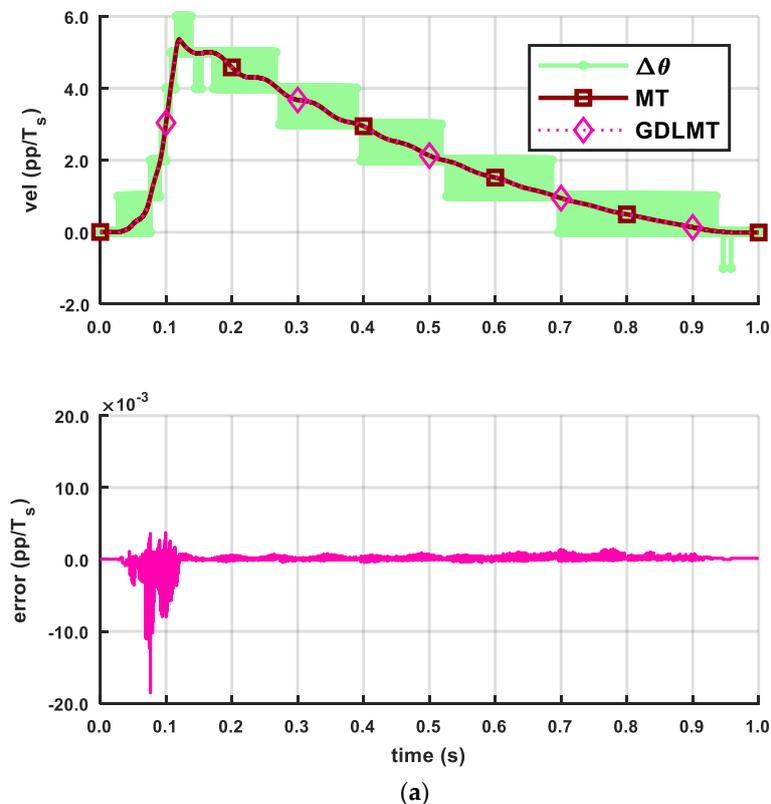
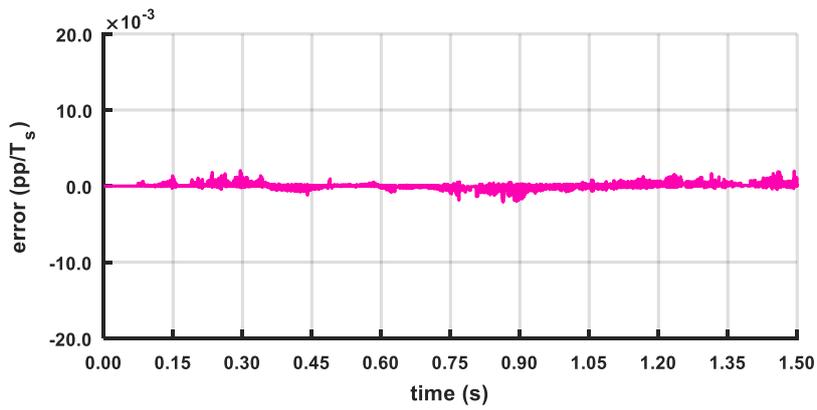
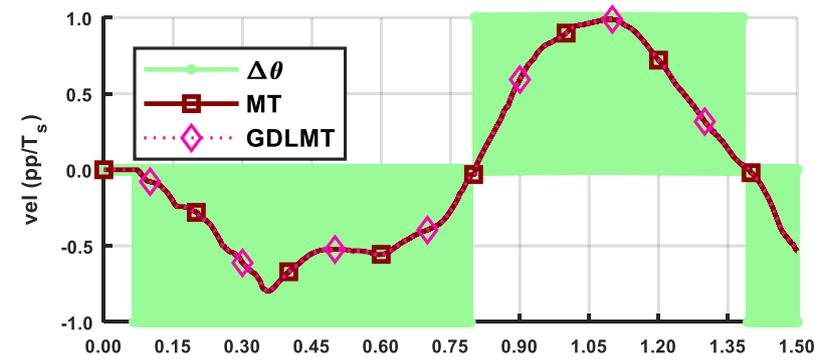
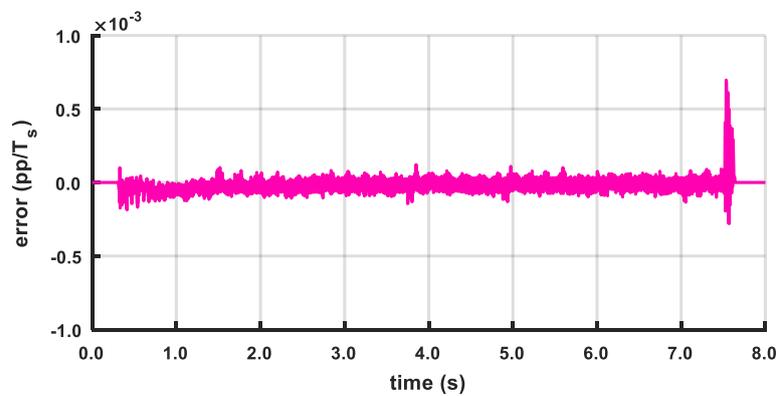
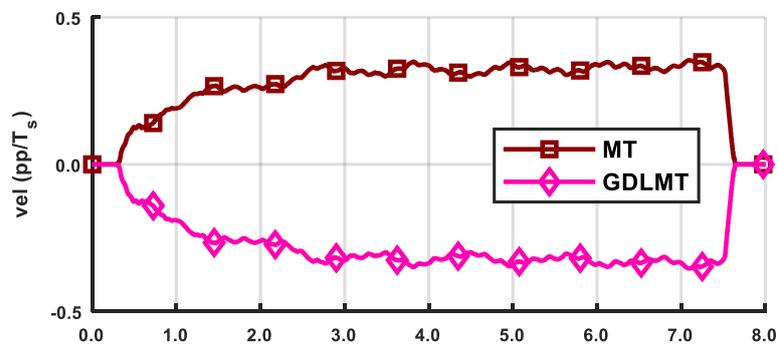


Figure 13. Cont.

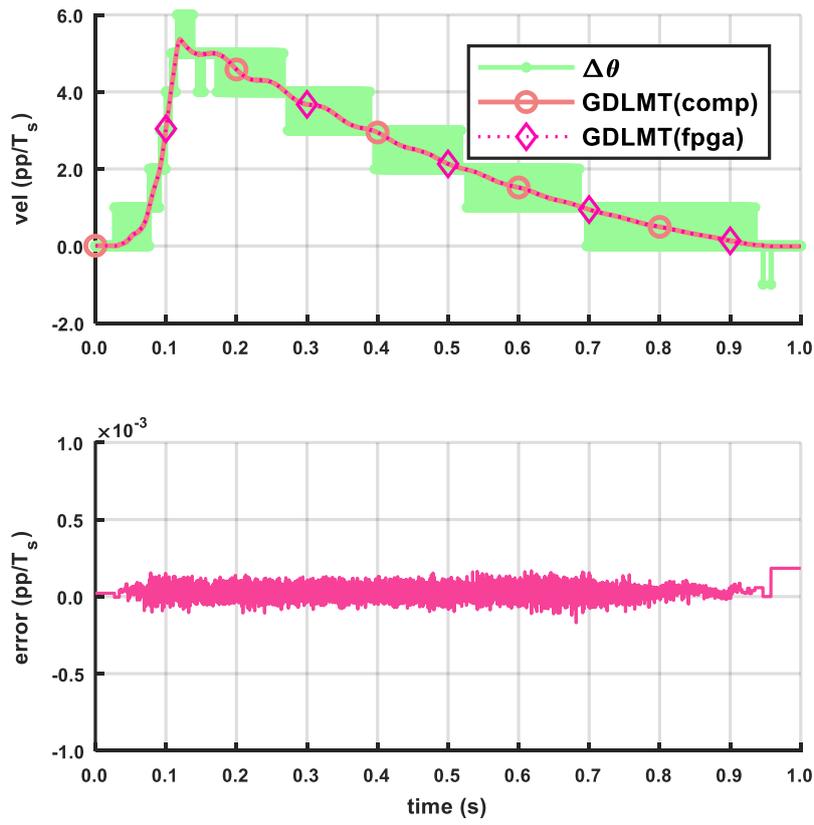


(b)

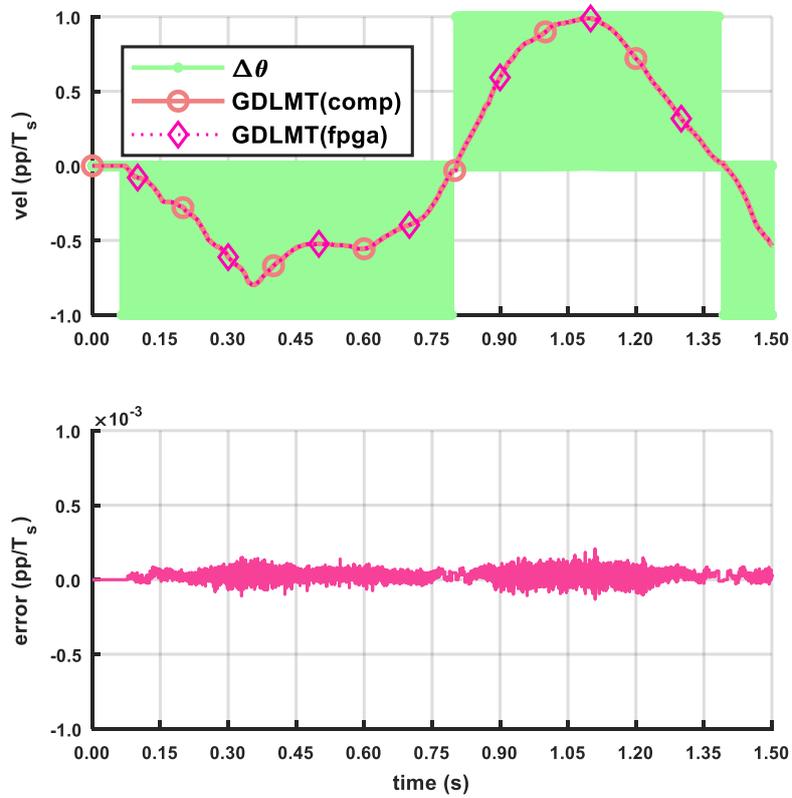


(c)

Figure 13. Cont.

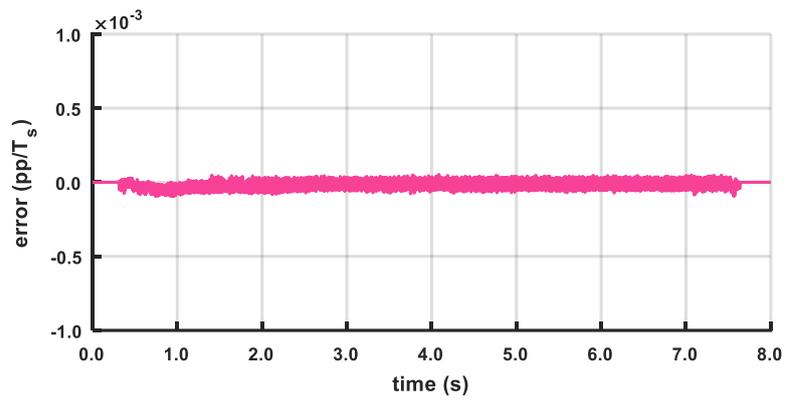
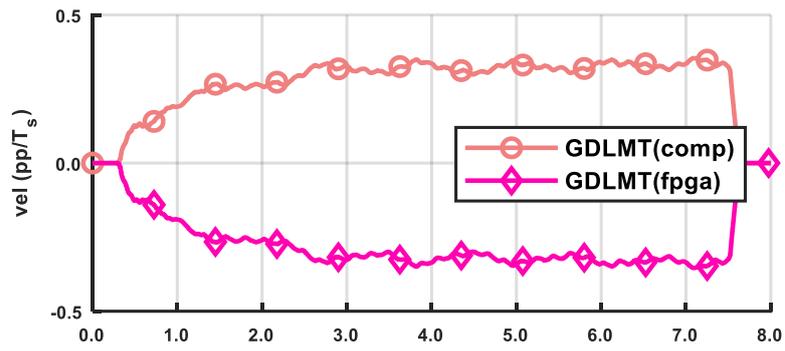


(d)

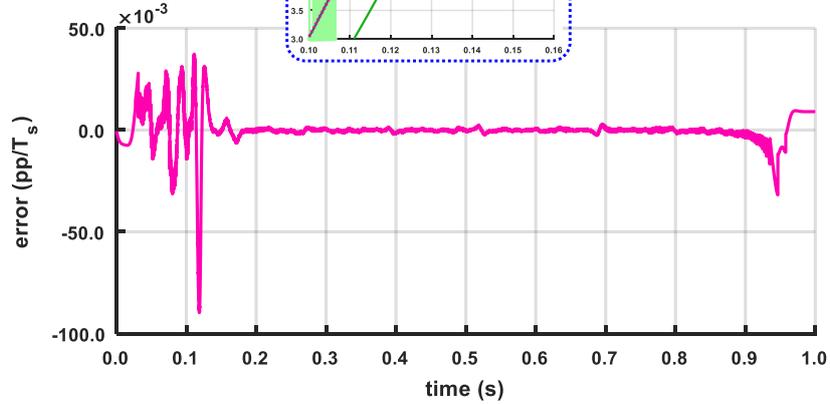
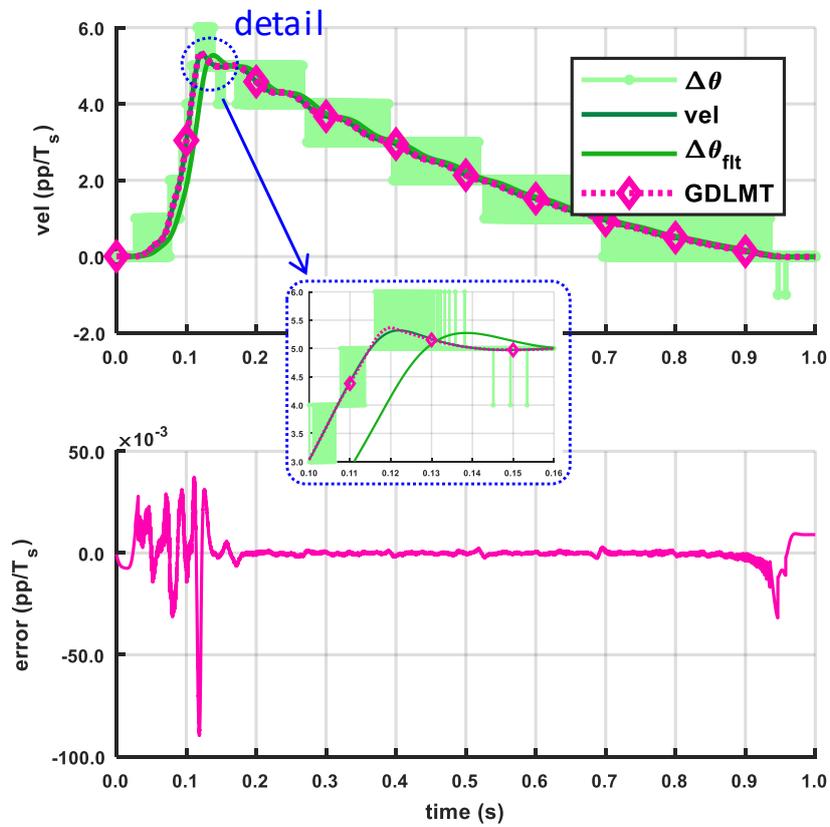


(e)

Figure 13. Cont.

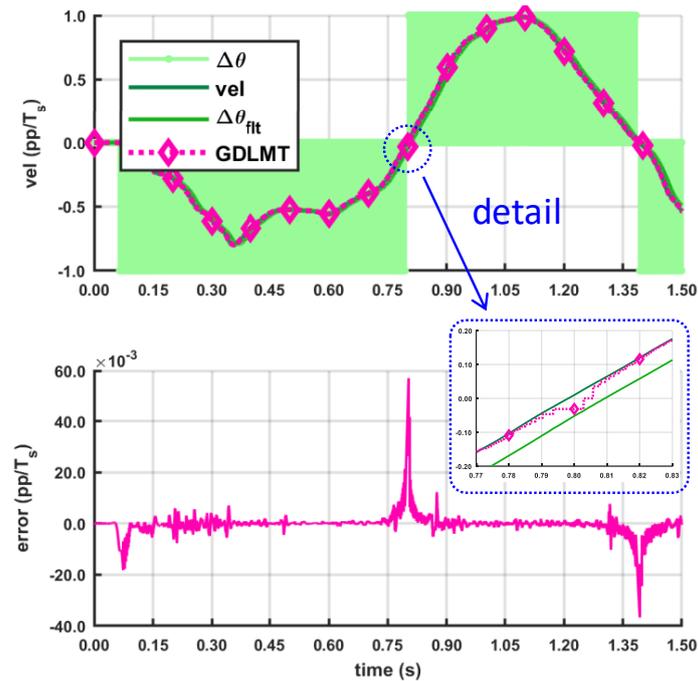


(f)

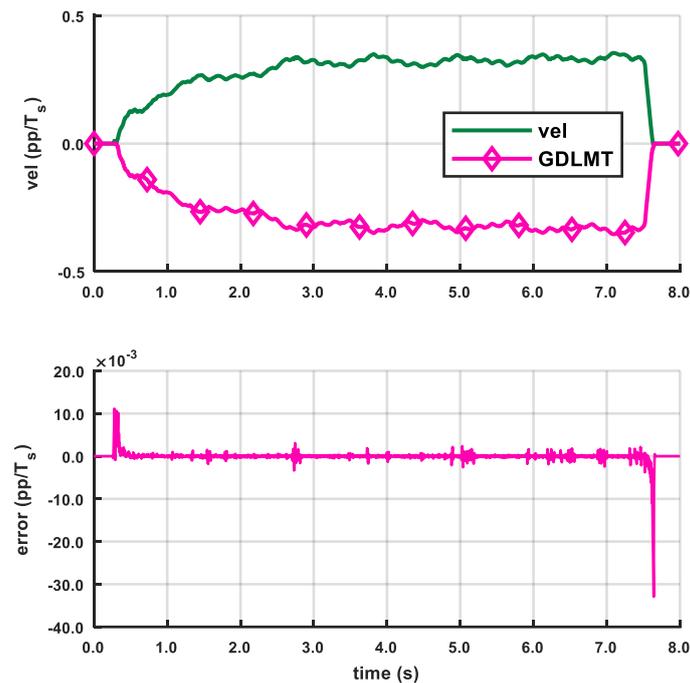


(g)

Figure 13. Cont.



(h)



(i)

**Figure 13.** Experiments with non-motorized encoder driving at medium and low speed: (a,d,g) abrupt manual start up followed by self-slowdown; (b,e,h) manual sinusoidal-like motion; (c,f,i) driving by gravitational load at near-constant very low speed.

The actual velocity and the GDLMT-velocity practically overlap, however, at high acceleration, deceleration and the zero crossings, the error increased (see the details in Figure 13g,h). The main causes for the increased error are related to the process of generating the so-called actual velocity trace, i.e., (a) The low-pass filtering nature, and (b) Disregarding the timing information in the occurrence of the encoder pulses. Furthermore, the zero-crossing error is inherent to the conventional

MT-type estimation approach, which produces output updates only in the case of position pulse occurrence within a recent sampling period, i.e., without new position pulses the algorithm has no new information to perform the estimation. In order to deal with this problem, a special approach is required, which is out of the scope of this paper. However, on the other hand, we can observe a considerable phase-lag in the case of the causal low-pass filter, which presents a common real-time solution in motion control applications.

#### 4. Discussion

The experimental results of the GDLMT-algorithm for velocity estimation by an incremental encoder show stable performance and a very high degree of matching with the conventional MT-method at various operating conditions. Nevertheless, the stability of the GDLMT algorithm performance was not jeopardized in any case, moreover, the asymptotical convergence was also clearly shown in the experiments. A typical error was below  $0.1/100$  (pp/ $T_s$ ), which is extremely low. However, good accuracy of the time interval measurement and high sampling rate are very important in order to keep the error low. Though at high acceleration and at velocity reversals the error peaks were significantly larger, i.e.,  $1/100$  (pp/ $T_s$ ), the velocity estimation method performed well in a wide speed range with a motor driven encoder shaft, as well as in the case of a manually driven encoder shaft. In the latter case, we could achieve highly smooth motion, while, in the motorized case, the motion was fluctuating and oscillating. However, it did not affect the method performance in any way. Hence, the results validated the proposed algorithm.

The proposed algorithm is simple by purpose, since it is meant for implementation on a cost optimized FPGA platform. Hence, the required resources for the implementation are an issue. The lowest detected velocity in an MT-type velocity estimation method is related to the longest time interval that can be measured by the hardware. However, it is linked with the counter of blank sampling periods that comes into effect in the case of widely spread encoder pulses. The counter itself is not a problem in the FPGA implementation; however, the method requires multiplication with the inverse of the counted blanks in order to provide stable behavior. Since the values are calculated in advance off-line and stored into the FPGA memory, the maximum length of a contiguous blank interval determines the size of the required memory. Thus, the minimum detected velocity and required FPGA resources are a design trade-off, though it is not a problematic issue.

As shown in the paper, we have proven that the algorithm can be implemented efficiently on the FPGA-based platform and provide accurate velocity estimation simultaneously (though the presented experimental setup is not fully optimized yet). It should be noted that the applied experimental FPGA-platform offers overwhelming resources which are not all required to such an extent, however, they make the experimental work easier. However, further optimization in terms of the FPGA implementation is possible towards smart incremental encoders with a position and a smooth velocity output.

#### 5. Conclusions

The paper deals with FPGA implementation of the MT method for velocity estimation by an incremental encoder. The conventional method involves arithmetic division, which presents a serious obstacle for implementation in a digital circuit such as an FPGA. Thus, we modified the method and introduced an algorithm which bypasses the arithmetic division efficiently. The whole algorithm can be implemented by simpler arithmetic operations such as multiplication, addition and subtraction only. We demonstrated the proof of the concept, i.e., that the implementation of the proposed divisionless MT algorithm on a real cost optimized FPGA is possible. In our experimental FPGA configuration the resources consumption is minimal and the result is available in only a few clock cycles, yet providing similar velocity estimation performance as the conventional MT-method. Thus, in the future, the proposed solution can be supported on a small size low-cost FPGA (with a cost of only few dollars), which offers only moderate resources and thus cannot provide a high performance computation

engine required for complex algorithms of the arithmetical division in real-time processing of advanced motion control applications.

**Author Contributions:** The presented work was carried out in collaboration and with significant contributions of both authors. A.H. developed the main concept of the measurement method, performed the theoretical analysis and simulations, supervised the work, and wrote the manuscript, whereas M.Č. provided important technical advice, implemented the method on the hardware, performed the experiments, examined the manuscript draft and provided valuable comments.

**Funding:** The authors acknowledge the financial support from the Slovenian Research Agency (Research Core Funding No. P2-0028).

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Ohnishi, K.; Shibata, M.; Murakami, T. Motion Control for Advanced Mechatronics. *IEEE/ASME Trans. Mechatron.* **1996**, *1*, 56–67. [[CrossRef](#)]
2. Tsuji, T.; Hashimoto, T.; Kobayashi, H.; Mariko, M.; Ohnishi, K. A Wide-Range Velocity Measurement Method for Motion Control. *IEEE Trans. Ind. Electron.* **2009**, *56*, 510–519. [[CrossRef](#)]
3. Zhu, W.-H.; Lamarche, T.; Dupuis, E.; Jameux, D.; Barnard, P.; Liu, G. Precision Control of Modular Robot Manipulators: The VDC Approach With Embedded FPGA. *IEEE Trans. Robot.* **2013**, *29*, 1162–1179.
4. Chawda, V.; Celik, O.; O'Malley, M.K. Evaluation of Velocity Estimation Methods Based on Their Effect on Haptic Device Performance. *IEEE/ASME Trans. Mechatron.* **2018**, *23*, 604–613. [[CrossRef](#)]
5. Kennel, R.M. Why Do Incremental Encoders Do a Reasonably Good Job in Electrical Drives with Digital Control? In Proceedings of the Conference Record of the 2006 IEEE Industry Applications Conference Forty-First IAS Annual Meeting, Tampa, FL, USA, 8–12 October 2006.
6. Kennel, R.M. Encoders for simultaneous sensing of position and speed in electrical drives with digital control. In Proceedings of the Conference Record of the 2005 Industry Applications Conference Fourtieth IAS Annual Meeting, Kowloon, Hong Kong, China, 2–6 October 2005.
7. Hace, A.; Franc, M. Pseudo-Sensorless High-Performance Bilateral Teleoperation by Sliding-Mode Control and FPGA. *IEEE/ASME Trans. Mechatron.* **2014**, *19*, 384–393. [[CrossRef](#)]
8. Kavanagh, R.C. Performance Analysis and Compensation of M/T-Type Digital Tachometers. *IEEE Trans. Instrum. Meas.* **2001**, *50*, 965–970. [[CrossRef](#)]
9. Petrella, R.; Tursini, M.; Peretti, L.; Zigliotto, M. Speed Measurement Algorithms for Low-Resolution Incremental Encoder Equipped Drives: A Comparative Analysis. In Proceedings of the 2007 International Aegean Conference on Electrical Machines and Power Electronics, Bodrum, Turkey, 10–12 September 2007.
10. Ohmae, T.; Matsuda, T.; Kamiyama, K.; Tachikawa, M. A Microprocessor-Controlled High-Accuracy Wide-Range Speed Regulator for Motor Drives. *IEEE Trans. Ind. Electron.* **1982**, *IE-29*, 207–211. [[CrossRef](#)]
11. Prokin, M. Extremely Wide-Range Speed Measurement Using a Double-Buffered Method. *IEEE Trans. Ind. Electron.* **1994**, *41*, 550–559. [[CrossRef](#)]
12. Kavanagh, R.C. An Enhanced Constant Sample-Time Digital Tachometer Through Oversampling. *Trans. Inst. Meas. Control.* **2004**, *26*, 83–98. [[CrossRef](#)]
13. Pu, J.-T.; Wang, H. A Novel Variable M/T Method for Speed Measurement with High Precision. In Proceedings of the 2nd International Conference on Electronic & Mechanical Engineering and Information Technology (EMEIT-2012), Liaoning, China, 7–12 September 2012.
14. Hachiya, K.; Ohmae, T. Digital Speed Control System for a Motor Using Two Speed Detection Methods of an Incremental Encoder. In Proceedings of the 2007 European Conference on Power Electronics and Applications, Aalborg, Denmark, 2–5 September 2007.
15. Lygouras, J.N.; Kodogiannis, V.; Pachidis, T.P.; Sirakoulis, G.C. A New Method for Digital Encoder Adaptive Velocity/Acceleration Evaluation Using a TDC with Picosecond Accuracy. *Microprocess. Microsyst.* **2009**, *33*, 453–460. [[CrossRef](#)]
16. Chen, Y.; Yang, M.; Long, J.; Xu, D.; Blaabjerg, F. M/T Method Based Incremental Encoder Velocity Measurement Error Analysis and Self-Adaptive Error Elimination Algorithm. In Proceedings of the IECON 2017—43rd Annual Conference of the IEEE Industrial Electronics Society, Beijing, China, 29 October–1 November 2017.

17. Merry, R.J.; Van de Molengraft, M.J.; Steinbuch, M. Velocity and Acceleration Estimation for Optical Incremental Encoders. *Mechatronics* **2010**, *20*, 20–26. [CrossRef]
18. Zhu, W.-H. FPGA-Based Velocity Estimation for Control of Harmonic Drives. In Proceedings of the 2010 IEEE International Conference on Mechatronics and Automation, Xi'an, China, 4–7 August 2010.
19. Koren, I. *Computer Arithmetic Algorithms*, 2nd ed.; A K Peters, Ltd.: Natick, MA, USA, 2002.
20. Hahn, T.; Humphreys, J.; Fritsch, A.; Greenstreet, D. Demystifying Digital Signal Processing (DSP) Programming: The Ease in Realizing Implementations with TI DSPs. Texas Instruments, 2015. Available online: <http://www.ti.com/lit/wp/spry281/spry281.pdf> (accessed on 26 July 2018).
21. Monmasson, E.; Idkhajine, L.; Cirstea, M.N.; Bahri, I.; Tisan, A.; Naouar, M.W. FPGAs in Industrial Control Applications. *IEEE Trans. Ind. Inform.* **2011**, *7*, 224–243. [CrossRef]
22. Rodríguez-Andina, J.J.; Valdés-Peña, M.D.; Moure, M.J. Advanced Features and Industrial Applications of FPGAs—A Review. *IEEE Trans. Ind. Inform.* **2015**, *11*, 853–864. [CrossRef]
23. Moreno-Tapia, S.V.; Vera-Salas, L.A.; Osornio-Rios, R.A.; Dominguez-Gonzalez, A.; Stiharu, I.; De Jesus Romero-Troncoso, R. A Field Programmable Gate Array-Based Reconfigurable Smart-Sensor Network for Wireless Monitoring of New Generation Computer Numerically Controlled Machines. *Sensors* **2010**, *10*, 7263–7286. [CrossRef] [PubMed]
24. De la Piedra, A.; Braeken, A.; Touhafi, A. Sensor Systems Based on FPGAs and Their Applications: A Survey. *Sensors* **2012**, *12*, 12235–12264. [CrossRef]
25. García, G.J.; Jara, C.A.; Pomares, J.; Alabdo, A.; Poggi, L.M.; Torres, F. A Survey on FPGA-Based Sensor Systems: Towards Intelligent and Reconfigurable Low-Power Sensors for Computer Vision, Control and Signal Processing. *Sensors* **2014**, *14*, 6247–6278. [CrossRef] [PubMed]
26. Rodríguez-Donate, C.; Osornio-Rios, R.A.; Rivera-Guillen, J.R.; Romero-Troncoso, R.D.J. Fused Smart Sensor Network for Multi-Axis Forward Kinematics Estimation in Industrial Robots. *Sensors* **2011**, *11*, 4335–4357. [CrossRef] [PubMed]
27. Quintáns, C.; Fariña, J.; Marcos-Acevedo, J. Improving the Performance of Incremental Encoders with Conditioning Circuits based on FPGA. *Measurement* **2016**, *90*, 1–3.
28. Rodríguez Andina, J.J.; De la Torre Arnanz, E.; Valdés Peña, M.D. *FPGAs—Fundamentals, Advanced Features and Applications in Industrial Electronics*; CRC Press: Boca Raton, FL, USA, 2017.
29. Zhu, W.-H. FPGA Logic Devices for Precision Control—An Application to Large Friction Actuators with Payloads. *IEEE Control. Syst. Mag.* **2014**, *34*, 54–75.
30. Jovanovic, B.; Jevtic, R.; Carreras, C. Binary Division Power Models for High-Level Power Estimation of FPGA-Based DSP Circuits. *IEEE Trans. Ind. Inform.* **2014**, *10*, 393–398. [CrossRef]
31. Sutter, G.; Deschamps, J.P. High speed fixed point dividers for FPGAs. In Proceedings of the 2009 International Conference on Field Programmable Logic and Applications, Prague, Czech Republic, 31 August–2 September 2009.
32. Deschamps, J.-P.; Antoine Bioul, G.J.; Sutter, G.D. *Synthesis of Arithmetic Circuits*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2006.
33. Hace, A.; Čurkovič, M. A Novel Divisionless MT-type Velocity Estimation Algorithm for Efficient FPGA Implementation. *IEEE Access* **2018**. [CrossRef]
34. Incze, I.I.; Szabó, C.; Imecs, M. Incremental Encoder in Electrical Drives: Modeling and Simulation. In *Computational Intelligence in Engineering. Studies in Computational Intelligence*; Rudas, I.J., Fodor, J., Kacprzyk, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; Volume 313, pp. 287–300.
35. Tsai, M.-F.; Chen, C.-P. Design of a Quadrature Decoder/Counter Interface IC for Motor Control Using CPLD. In Proceedings of the IEEE 28th Annual Conference of the Industrial Electronics Society (IECON '02), Sevilla, Spain, 5–8 November 2002.
36. Digilent Nexys 2 Reference Manual. Digilent Inc. Available online: <https://reference.digilentinc.com/reference/programmable-logic/nexys-2/reference-manual> (accessed on 31 July 2018).

