

Article

# A New Software-Based Optimization Technique for Embedded Latency Improvement of a Constrained MIMO MPC

David Sotelo <sup>1</sup>, Antonio Favela-Contreras <sup>1,\*</sup>, Alfonso Avila <sup>1</sup>, Arturo Pinto <sup>1</sup>,  
Francisco Beltran-Carbajal <sup>2</sup> and Carlos Sotelo <sup>1</sup>

<sup>1</sup> Tecnológico de Monterrey, School of Engineering and Sciences, Ave. Eugenio Garza Sada 2501, Monterrey 64849, Mexico; david.sotelo@tec.mx (D.S.); aavila@tec.mx (A.A.); apinto@gmail.com (A.P.); carlos.sotelo@tec.mx (C.S.)

<sup>2</sup> Departamento de Energía, Universidad Autónoma Metropolitana, Unidad Azcapotzalco, Av. San Pablo No. 180, Col. Reynosa Tamaulipas, Mexico City 02200, Mexico; fbeltran.git@gmail.com

\* Correspondence: antonio.favela@tec.mx

**Abstract:** Embedded controllers for multivariable processes have become a powerful tool in industrial implementations. Here, the Model Predictive Control offers higher performances than standard control methods. However, they face low computational resources, which reduces their processing capabilities. Based on pipelining concept, this paper presents a new embedded software-based implementation for a constrained Multi-Input-Multi-Output predictive control algorithm. The main goal of this work focuses on improving the timing performance and the resource usage of the control algorithm. Therefore, a profiling study of the baseline algorithm is developed, and the performance bottlenecks are identified. The functionality and effectiveness of the proposed implementation are validated in the NI myRIO 1900 platform using the simulation of a jet transport aircraft during cruise flight and a tape transport system. Numerical results for the study cases show that the latency and the processor usage are substantially reduced compared with the baseline algorithm,  $4.6\times$  and  $3.17\times$  respectively. Thus, efficient program execution is obtained which makes the proposed software-based implementation mainly suitable for embedded control systems.

**Keywords:** model predictive control; embedded systems; MIMO systems; system-on-chip; NI myRIO

**MSC:** 9304



**Citation:** Sotelo, D.;

Favela-Contreras, A.; Avila, A.; Pinto, A.; Beltran-Carbajal, F.; Sotelo, C. A New Software-Based Optimization Technique for Embedded Latency Improvement of a Constrained MIMO MPC. *Mathematics* **2022**, *10*, 2571. <https://doi.org/10.3390/math10152571>

Academic Editors: Róbert Szabolcsi and Péter Gáspár

Received: 9 June 2022

Accepted: 22 July 2022

Published: 24 July 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Most industrial processes are multivariable systems which often present strong interactions between the manipulated and the controlled variables. Here, the definition of a suitable control structure to regulate the system is not often trivial [1], especially when dealing with multiple-objective problems and dynamics complexity. In this scenario, even though diverse strategies are possible [2], Model Predictive Control (MPC) turned out to be among the most effective ones in fulfilling the aforementioned tasks [3–5]. Studies found in the recent literature exploit algorithms based on MPC methods applied to several process, in which online optimization problems are involved [6–10]. However, it is well known that MPC is more challenging than other control strategies [11,12]. The challenges and differences arise from the following main aspects:

1. The control action must be calculated in a receding horizon manner. MPC solves an optimal control problem at each sampling instant, where a sequence of control moves are optimized and only the first control move is injected into the system [13]. This process is repeated, which translate into a greater computational demand on control platforms [14].
2. MPC has been known as powerful tool to deal with system constraints [15]. This control strategy addresses the state constraints and input saturations to obtain the

manipulation that minimizes the proposed cost function [16]. Here, heavy execution load and communication resources are needed [17].

3. Use of weighting matrices in the cost function. In MPC approaches the value of the pondering matrices are set a priori, to penalize the large excursion of the tracking error and the deviation of the manipulation. This, to obtain the control action; thus, the optimization process that derives the best results can not usually sustain a low stable computation time [18].

Henceforth, these aspects limits the algorithms that can be used to solve the MPC strategy and has motivated the search for computationally efficient optimization algorithms for practical applications where the sampling period tends to be short [12]. In addition, embedded systems are a powerful tool in industrial implementations. In contrast to personal computer applications, they address low computational resources, but provides portability, low cost and low jitter because of the reduction of software interruptions from the operative systems. Thus; MPC is desirable for embedded applications with fast control cycles, such as bioengineering, aerospace, automotive, etc. These embedded applications require small and power-efficient controllers, capable to solve optimization problems at high sampling rates [19]. The first embedded implementation of an MPC algorithm is reported in [20], which consists of a constrained control strategy for a single-input single-output (SISO) system on a Field-Programmable Gate Array (FPGA). Nowadays, research efforts are focused on the development of embedded MPC implementations with low latency. Nevertheless, the computation complexity makes the multivariable MPC poorly efficient for high speed applications where the controller iteration must be executed in a few milliseconds or even in microseconds.

For this reason, recent research works address the computational load by software or hardware improvement. In [21], for multivariable processes, a relaxed performance index with the constraints implicitly defined in the weighting matrices is developed, which contributes to reduce the execution time. Moreover, in [22], the online latency is reduced, by converting the quadratic programming problem into an equivalent nonnegative least-squares problem. On the other hand, in [23–25], the authors implement a constrained MIMO MPC on high resources processors of 1.6 GHz, 2.2 GHz and 1.25 GHz respectively. These implementations reach low latency due to the amount of resources on their embedded devices. However, the processors are not affordable. Furthermore, in [9], a constrained MIMO MPC is well implemented on a NI myRIO 677 MHz processor. Nevertheless, as it is mentioned, it is not recommended for fast applications. In [26], an MPC algorithm is implemented on a microprocessor of 216 MHz reaching low latency, but constraints are not considered. Additionally, a constrained MIMO MPC is implemented on a microprocessor of 96 MHz in [27], nevertheless, high latency is reported. In contrast to research efforts found in the literature, this work presents an embedded implementation of a constrained MIMO MPC algorithm (“Baseline”) and a new software-based optimization for the embedded control strategy (“Propose”). Thus, without affecting the tracking performance of the Baseline, the execution time and the processor usage are substantially reduced in the Propose. This makes the proposed software-based implementation mainly suitable for embedded control systems.

The paper is organized as follows. Section 2 presents the constrained MIMO predictive control algorithm. Section 3 describes the foundations of computational architecture used in this work. Section 5 presents a computational analysis of the MIMO predictive control algorithm and the proposed software-based implementation. Additionally, a jet transport aircraft as study case and the selected embedded platform are presented. Section 6 shows the performance, the execution time and the processor usage, of both implementations: the Baseline and the Propose. Finally, Section 7 discusses the conclusions.

## 2. Model Predictive Control Algorithm

This section presents a brief review of the MIMO predictive control algorithm. This MPC is based on discrete-time state space model, and it is proposed by Wang in [10]. In

this previous work, considering the predictions of the states, the control action is obtained through the solution of a constrained optimization problem by using a cost function with a weighting matrix. At each sampling time, an optimal control problem is solved which demands high computational resource usage. The system dynamics is denoted by the discrete time Linear Time Invariant (LTI) State-Space Model taking the following structure:

$$\begin{aligned} x_m(k+1) &= A_m x_m(k) + B_m u(k), \\ y(k) &= C_m x_m(k) \end{aligned} \tag{1}$$

where  $x_m(k) \in \mathbb{R}^{n_x}$  is the state vector,  $u(k) \in \mathbb{R}^{n_u}$  is the controlled input vector,  $y(k) \in \mathbb{R}^{n_y}$  is the output vector,  $A_m \in \mathbb{R}^{n_x \times n_x}$  is the state matrix,  $B_m \in \mathbb{R}^{n_x \times n_u}$  is called the input matrix,  $C_m \in \mathbb{R}^{n_y \times n_x}$  is called the output matrix and  $k \in \mathbb{N}$  denotes the sampling instant number.

The MPC strategy uses the Velocity Form Model (VFM) as offset-free method [10,28–30]. The VFM method is based on an augmented State-Space Model. Hence, from (1),

$$\begin{aligned} \Delta x_m(k+1) &= A_m \Delta x_m(k) + B_m \Delta u(k) \\ \Delta y(k+1) &= C_m A_m \Delta x_m(k) + C_m B_m \Delta u(k) \end{aligned} \tag{2}$$

where  $\Delta x_m(k+1) = [x_m(k+1) - x_m(k)]$ ,  $\Delta x_m(k) = [x_m(k) - x_m(k-1)]$ ,  $\Delta u(k) = [u(k) - u(k-1)]$  and  $\Delta y(k+1) = y(k+1) - y(k)$ , the VFM takes the following structure:

$$\begin{aligned} x(k+1) &= Ax(k) + B\Delta u(k) \\ y(k) &= Cx(k) \end{aligned} \tag{3}$$

where

$$\begin{aligned} A &= \begin{bmatrix} A_m & 0_{n_x \times n_y} \\ C_m A_m & I_{n_y \times n_y} \end{bmatrix} \\ B &= \begin{bmatrix} B_m \\ C_m B_m \end{bmatrix} \\ C &= [0_{n_y \times n_x} \quad I_{n_y \times n_y}] \end{aligned} \tag{4}$$

and  $x(k) \in \mathbb{R}^{n_x+n_y}$  is the new state vector:

$$x(k) = \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} \tag{5}$$

As in [10], the state predictions for the consecutive sampling instants are:

$$\begin{aligned} x(k+1) &= Ax(k) + B\Delta u(k) \\ x(k+2) &= Ax(k+1) + B\Delta u(k+1) \\ &= A^2x(k) + AB\Delta u(k) + B\Delta u(k+1) \\ &\vdots \\ x(k+N_p) &= A^{N_p}x(k) + A^{N_p-1}B\Delta u(k) + \dots \\ &\quad + A^{N_p-N_c}B\Delta u(k+N_p-1) \end{aligned} \tag{6}$$

where  $N_p$  stands for prediction horizon and  $N_c$  corresponds to control horizon. Then, from (3) and (6), the predicted outputs can be formulated using the following vector-matrix notation:

$$Y = \Psi x(k) + \Phi \Delta U \tag{7}$$

where

$$\Psi = \begin{bmatrix} (CA)^T & (CA^2)^T & \dots & (CA^{N_p})^T \end{bmatrix}^T$$

$$\Phi = \begin{bmatrix} CB & 0_{n_y \times n_u} & \dots & 0_{n_y \times n_u} \\ CAB & CB & \dots & 0_{n_y \times n_u} \\ \vdots & \vdots & \ddots & \vdots \\ CA^{N_p-1}B & CA^{N_p-2}B & \dots & CA^{N_p-N_c}B \end{bmatrix} \tag{8}$$

and  $Y \in \mathbb{R}^{N_p \cdot n_y}$  is the whole predicted outputs of  $y = [y_1 \ y_2 \ \dots \ y_{n_y}]^T$ , and  $\Delta U \in \mathbb{R}^{N_c \cdot n_u}$  concatenates the future control trajectory of  $\Delta u = [\Delta u_1 \ \Delta u_2 \ \dots \ \Delta u_{n_u}]^T$ :

$$Y = \begin{bmatrix} y(k+1)^T & y(k+2)^T & \dots & y(k+N_p)^T \end{bmatrix}^T$$

$$\Delta U = \begin{bmatrix} \Delta u(k)^T & \Delta u(k+1)^T & \dots & \Delta u(k+N_c-1)^T \end{bmatrix}^T \tag{9}$$

The definition of the cost function is one of the key elements in MPC strategy to define the desired system behavior [11,12]. The cost function  $J(\Delta U|Y, R_s)$  used to find the best sequence of control action over the prediction horizon  $[k, k+N_p]$  is defined as follows:

$$J = (R_s - Y)^T (R_s - Y) + \Delta U^T R_u \Delta U \tag{10}$$

where, the first element considers the error  $e \in \mathbb{R}^{N_p \cdot n_y}$  between the whole predicted outputs denoted by  $Y \in \mathbb{R}^{N_p \cdot n_y}$  and the reference  $R_s \in \mathbb{R}^{N_p \cdot n_y}$  of the vector  $r_s = [r_1 \ r_2 \ \dots \ r_{n_y}]^T$ . Furthermore, as in [31], the second term stands to penalize the rate excursion of the control vector; here, the weighting matrix  $R_u$  represents a direct influence on the performance of the system [11]. Hence, when a low pondering matrix  $R_u \in \mathbb{R}^{N_c \cdot n_u \times N_c \cdot n_u}$  value is used, the first element of the cost function is interpreted as the primary situation to be minimized. The reference  $R_s$  vector is expressed as follows:

$$R_s = \begin{bmatrix} r_s(k+1)^T & r_s(k+2)^T & \dots & r_s(k+N_p)^T \end{bmatrix}^T \tag{11}$$

Substituting (7) in the first element of (10), the cost function is expressed in state-space representation:

$$J = (R_s - \Psi x(k) - \Phi \Delta U)^T (R_s - \Psi x(k) - \Phi \Delta U) + \Delta U^T R_u \Delta U \tag{12}$$

Thus, to obtain the control law, the performance index (12) is rewritten in the following compact form:

$$J = \frac{1}{2} \Delta U^T E \Delta U + [F_1 x(k) + F_2 R_s]^T \Delta U + \text{Cst} \tag{13}$$

where

$$\begin{aligned} E &= 2[\Phi^T \Phi + R_u] \\ F_1 &= 2\Phi^T \Psi \\ F_2 &= -2\Phi^T \\ \text{Cst} &= (R_s - \Psi x(k))^T (R_s - \Psi x(k)) \end{aligned} \tag{14}$$

subject to the following inequality constraints:

$$M \Delta U \leq \gamma(k) \tag{15}$$

where

$$\begin{aligned}
 M &= \begin{bmatrix} L_u^T & -L_u^T & I_u & -I_u & \Phi^T & -\Phi^T \end{bmatrix}^T \\
 \gamma(k) &= \begin{bmatrix} U^{\max} - I_1 u(k-1) \\ -U^{\min} + I_1 u(k-1) \\ \Delta U^{\max} \\ -\Delta U^{\min} \\ \Upsilon^{\max} - \Psi x(k) \\ -\Upsilon^{\min} + \Psi x(k) \end{bmatrix}
 \end{aligned} \tag{16}$$

then,  $M \in \mathbb{R}^{(4 \cdot N_c \cdot n_u + 2 \cdot N_p \cdot n_y) \times N_c \cdot n_u}$  is a matrix reflecting the constraints, and  $\gamma(k) \in \mathbb{R}^{4 \cdot N_c \cdot n_u + 2 \cdot N_p \cdot n_y}$  contains the maximum and minimum values allowed in  $u$ ,  $\Delta u$ , and  $y$ . Moreover,  $I_u \in \mathbb{R}^{N_c \cdot n_u \times N_c \cdot n_u}$  is a identity matrix, additionally  $L_u \in \mathbb{R}^{N_c \cdot n_u \times N_c \cdot n_u}$  is a lower triangular matrix and  $I_1 \in \mathbb{R}^{N_c \cdot n_u}$  is a vector-matrix whose elements are  $I \in \mathbb{R}^{n_u \times n_u}$ .

As in [32], the sequence of the future actions  $\Delta U$  cannot be freely chosen in  $\mathbb{R}^{N_c \cdot n_u}$ . The constraints have to be taken into account giving the following optimization problem  $P(x(k))$ :

$$P(x(k)) : \min_{\Delta U \in \mathbb{R}^{N_c \cdot n_u}} J(\Delta U|x(k)) \quad \text{s.t.} \quad g(\Delta U|x(k)) \leq 0 \tag{17}$$

The cost function  $J(\Delta U|x(k))$  is minimized considering the constraints arranged in  $g(\Delta U|x(k))$ , and the first action of the best sequence control action  $\Delta U$  is used. Nevertheless, solving the optimization problem of the baseline algorithm involves a high number of decision variables and high number of constraints which demands high computational resources.

### 3. Computer Optimization

An algorithm is a set of sequential calculations where tasks execution may depend on a different task product. Due to these dependencies, bottlenecks are carried out. This section introduces two concepts to break down the bottlenecks: software pipelining and one-step ahead prediction.

#### 3.1. Software Pipelining Technique

The set of tasks executed in one iteration is called Single-Cycle Iteration (SCI). Figure 1a shows the SCIs carried out in sequence, according to  $N$  instructions. The instructions are labeled as  $I_i^n$ , where  $n \in \{1, 2, \dots, N\}$  is the instruction number and  $i \in \{0, 1, \dots, \infty\}$  is the SCI number. Each instruction has its own latency, and the sum of the  $N$  latencies  $\sum_{n=1}^N \ell_n$  from the same SCI corresponds to the SCI latency  $\ell_{SCI}$ .

The pipelining technique is used to divide the instructions of an SCI in multiple steps. Thus, some dependencies are broken and instructions from different SCIs are overlapped during runtime. Figure 1b shows the pipeline technique. The instructions from a SCI are divided, hence, there are not dependencies inside the new-bundled iteration. This new iteration is called Pipelined Iteration (PI) with a new latency equal to  $\max\{\ell_1, \ell_2, \dots, \ell_N\}$ .

The implementation of the pipelining technique provides two main advantages: Parallelization and delivery time. The parallelization can be reached due to the broken dependencies that allows the execution of more than one instruction at the same time. As it is shown in Figure 1b, the instructions of the same column can be executed in parallel as long as hardware capability. The delivery time with nonpipelining technique of a SCI is  $\sum_{n=1}^N \ell_n$  while the execution time with pipelining implementation is  $\ell_{PI}$ , here, only the first SCI execution time takes  $N \cdot \ell_{PI}$  to be delivered.

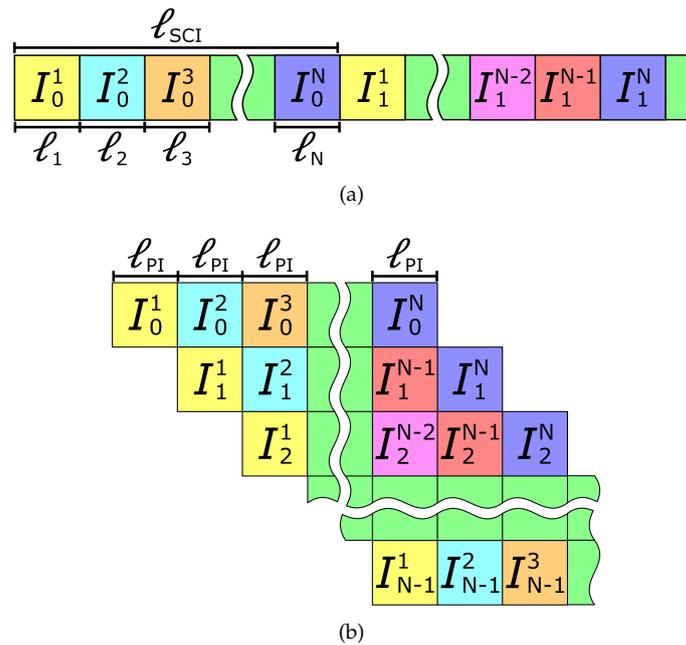


Figure 1. Execution techniques: (a) Nonpipelining. (b) Pipelining.

### 3.2. One-Step Ahead Prediction

The MPC approach assumes that with the system information been collected at each sampling instant  $y(k)$ ,  $\gamma(k)$ ,  $x(k)$ ,  $u(k - 1)$ , the control action  $u(k)$  can be calculated instantaneously. However, the real-time implementation of this approach is mainly limited due to the optimization problem to be solved.

In the proposed algorithm, the controller parameters optimized from the previous sampling interval is used to calculate the current control action  $u(k - 1)$ . By minimizing (10),  $\Delta U(k)$ , (9), is obtained subject to the constraints. Hence, the first vector corresponds to  $\Delta u(k)$ ; thus, the current manipulation  $u(k)$  can be computed as follows:  $u(k) = \Delta u(k) + u(k - 1)$ . This breaks the bottleneck due to the dependencies carried out by the variables computed at the same iteration.

## 4. Study Cases

The present work and the MPC strategy in [10] are developed using the same computational platform. The study cases consider the following state-space models in continuous time, where  $A_c \in \mathbb{R}^{n_x \times n_x}$  is the state matrix,  $B_c \in \mathbb{R}^{n_x \times n_u}$  stands for the input matrix and  $C_c \in \mathbb{R}^{n_x \times n_y}$  corresponds to the output matrix. Hence, the systems are discretized to be represented according to (1).

### 4.1. Study Case 1— Jet Transport Aircraft

From [21,33,34], a Jet Transport Aircraft Boeing 747 is obtained. In high-lift configuration, it addresses complex geometries and physical phenomena that make the controller design a difficult process. Figure 2 illustrates the Jet Transport Aircraft with its components and variables involved such as the angles  $\beta$  and  $\phi$  and the angular velocities  $\psi$  and  $\theta$ .

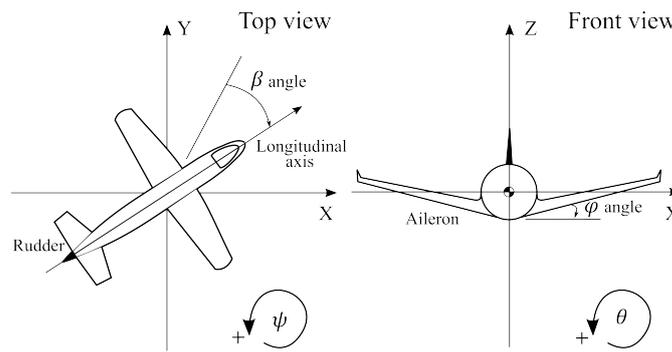


Figure 2. Jet transport aircraft.

Although the physical model of the Boeing 747 is lengthy, in (18), the simplified state-space model during cruise flight at  $MATCH = 0.8$  and  $H = 40,000$  ft is presented in continuous time:

$$\begin{aligned} \dot{x}(t) &= A_c x(t) + B_c u(t), \\ y(t) &= C_c x(t), \end{aligned} \tag{18}$$

where

$$\begin{aligned} A_c &= \begin{bmatrix} -0.0558 & -0.9968 & 0.0802 & 0.0415 \\ 0.598 & -0.115 & -0.0318 & 0 \\ -3.05 & 0.388 & -0.465 & 0 \\ 0 & 0.0805 & 1 & 0 \end{bmatrix} \\ B_c &= \begin{bmatrix} 0.0729 & 0 \\ -4.75 & 0.775 \\ 0.153 & 14.3 \\ 0 & 0 \end{bmatrix} \\ C_c &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

The model has four states,  $x = [\beta \ \psi \ \theta \ \phi]^T$ , where  $\beta$  is the sideslip angle,  $\phi$  stands for the bank angle, and meanwhile  $\psi$  and  $\theta$  represent the yaw and roll rate, respectively. Herein, all the angles are in rad and the angular velocities in rad/s. The system has two inputs  $u = [u_1 \ u_2]^T$ : the rudder and the aileron deflections, and two outputs  $y = [y_1 \ y_2]^T$  the yaw rate  $\psi$  and the bank angle  $\phi$ . Then, using a sampling time  $\tau = 0.2$  s, the system is discretized and executed under the constraints described in Table 1.

Table 1. Jet transport aircraft constraints.

	Output		Input		Input Increment	
	Yaw Rate	Bank Angle	Rudder Deflection	Aileron Deflection	Incremental Rudder Deflection	Incremental Aileron Deflection
Maximum	1	1	4	4	1	1
Minimum	-1	-1	-4	-4	-1	-1

#### 4.2. Study Case 2—Tape Transport System

The tape drive system consists of two reels to supply and file data. Here, the data transfer rate is proportional to the tape transport speed. Thus, the tape drive mechanism must be able to rapidly transport a fragile tape with an accurate tension regulation. Figure 3 shows the schematic of a tape transport system where its components and variables involved are the tape stiffness and the damping denoted by  $K$  and  $D$ , the reel radii and the inertia

represented as  $r$  and  $J$ , the motor torque constant  $K_t$ , and the viscous friction coefficient denoted by  $\beta$ .

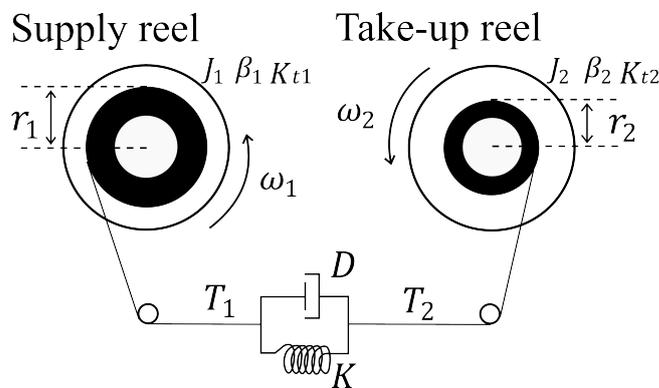


Figure 3. Tape transport system.

Assuming there is no force loss across the head, the tape tension  $T = T_1 = T_2$  [35]. Although the physical model of the process contains nonlinearities, in (19), a simplified state-space model is presented in continuous time [21,35–37]:

$$\begin{aligned} \dot{x}(t) &= A_c x(t) + B_c u(t), \\ y(t) &= C_c x(t), \end{aligned} \tag{19}$$

where

$$\begin{aligned} A_c &= \begin{bmatrix} -D\left(\frac{r_1^2}{J_1} + \frac{r_2^2}{J_2}\right) & D\frac{\beta r_1}{J_1} - \frac{K}{r_1} & \frac{K}{r_2} - D\frac{\beta r_2}{J_2} \\ \frac{r_1}{J_1} & -\frac{\beta}{J_1} & 0 \\ -\frac{r_2}{J_2} & 0 & -\frac{\beta}{J_2} \end{bmatrix} \\ B_c &= \begin{bmatrix} -DK_t\frac{r_1}{J_1} & DK_t\frac{r_2}{J_2} \\ \frac{K_t}{J_1} & 0 \\ 0 & \frac{K_t}{J_2} \end{bmatrix} \\ C_c &= \begin{bmatrix} 0 & \frac{r_1}{2} & \frac{r_2}{2} \\ 1 & 0 & 0 \end{bmatrix} \end{aligned}$$

The model has three states,  $x = [ T \ \omega_1 \ \omega_2 ]^T$ , where  $T$  is the tension tape in  $N$ ; meanwhile,  $\omega_1$  and  $\omega_2$  represent the supply and take-up reel in  $\text{rad/s}$ , respectively. Moreover, the system has two inputs  $u = [ u_1 \ u_2 ]^T$  that represent the voltages applied to the reel motors in volts, and two outputs  $y = [ y_1 \ y_2 ]^T$  which stand for the tape speed  $v_{rw}$  at the read-write head in  $\text{m/s}$  and the tape tension  $T$ , respectively. The control strategy described in the present work is simulated using parameters from the tested tape system described in [35,38,39], whose parameters are summarized in Table 2.

Table 2. Tape transport system parameters.

Symbol	Parameter	Value
$K$	Tape stiffness	$2 \times 10^3 \text{ N/m}$
$D$	Damping	$2 \text{ N s/m}^2$
$r_1$	Radius of supply reel	$21.2 \times 10^{-3} \text{ m}$
$r_2$	Radius of take-up reel	$9.75 \times 10^{-3} \text{ m}$
$J_1$	Moment of inertia of the supply reel	$14.2 \times 10^{-6} \text{ kg m}^2$
$J_2$	Moment of inertia of take-up reel	$10.35 \times 10^{-6} \text{ kg m}^2$
$K_t$	Motor torque constant	$24.8 \times 10^{-3} \text{ N m/V}$
$\beta$	Viscous friction coefficient	$1.03 \times 10^{-4} \text{ N m s/rad}$

Considering that the motors are nominally identical, for both motors, it is used as the same motor torque constant  $K_t$  and viscous friction coefficient  $\beta$  [38,39]. Thus, using a sampling time  $\tau = 0.1$  s, the system (19) is discretized and executed under the constraints expressed in Table 3.

**Table 3.** Tape transport system constraints.

	Output		Input		Input Increment	
	Tape Speed	Tape Tension	Motor Voltage	Motor Voltage	Incremental Motor Voltage	Incremental Motor Voltage
Maximum	3	0.8	2	0.4	1	1
Minimum	0	0	−1	−0.1	−1	−1

## 5. Proposed Software-Based Implementation

### 5.1. Profile Analysis

The profiling analysis is performed on a NI myRIO 1900, illustrated in Figure 4, from National Instruments [40]. The NI myRIO is a portable reconfigurable I/O (RIO) device that can be used to design control, robotics, and mechatronics systems. The NI myRIO contains a Xilinx Z-7010 System-on-Chip (SoC), working at a frequency of 650 MHz.



**Figure 4.** NI myRIO 1900.

The Xilinx Z-7010 SoC has an architecture divided in two parts: a processing system (PS) and a programmable logic. The PS includes 256 KB on-chip memory, 8 Direct Memory Access (DMA) channels, external memory interfaces, and multiple I/O peripherals. The main component of the PS is the application processor unit (APU) which contains a dual-core ARM Cortex-A9 for running the application software. The ARM cores use the ARMv7 architecture.

Although the NI myRIO can be considered a high-speed processing device compared to the microprocessor in [22,26,27]. In contrast with those with acceptable implementation for high speed applications [23–25], it can still be considered as a low resources processor.

The process flow diagram of the baseline MPC algorithm is shown in Figure 5. As it can be seen, it is composed of two types of functions: offline and online. The offline functions are one-time executed, while the online functions are executed each sampling time. Tables 4 and 5 shows the description of each function and the latency according to the study case 1.

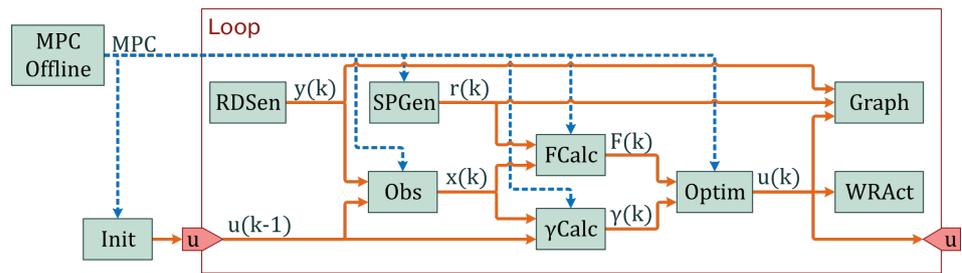


Figure 5. Process flow diagram of the MPC algorithm.

Table 4. Offline functions.

Function	Description	Latency
MPC Offline	It contains the matrices of the State-Space Model and the VFM, the sampling time, the prediction and control horizon, the constraints and the matrices $E, F_1, F_2, M, I_1, \Psi$ and $\Phi$ .	4183.580 $\mu\text{s}$
Init	It initializes the control vector $u_k$	86,718.800 $\mu\text{s}$

Table 5. Online functions.

Function	Description	Latency
RDSen	It reads the controlled variable $y(k)$ .	22.532 $\mu\text{s}$
SPGen	It generates the setpoint $r_s(k)$ .	14.509 $\mu\text{s}$
Obs	It measures the current $x(k)$ vector.	52.378 $\mu\text{s}$
FCalc	It computes the vector $F(k) = F_1x(k) + F_2R_s$ .	80.698 $\mu\text{s}$
$\gamma$ Calc	It computes the vector $\gamma(k)$ .	63.821 $\mu\text{s}$
Optim	It computes the optimal manipulated variable $u(k)$ .	1290.780 $\mu\text{s}$
WRAct	It applies the manipulated variable $u(k)$ .	39.202 $\mu\text{s}$
Graph	It plots the variables $y(k), u(k), \Delta u$ and the constraints.	37.468 $\mu\text{s}$

### 5.2. Software-Based Implementation Development

To improve the computing performance, a proposed software-based implementation is developed based on the profiling analysis of the online functions (Table 5) and the computer architecture concepts described in Section 3. Thus, the following improvements are implemented: efficient calculation of the optimization algorithm, implementing pipelining. This work apply three different methods to achieve a better performance. In Table 5, it can be seen that the Optim function has the highest latency. Thus, the present research work first proposes to speed it up by modifying the Hildreth’s algorithm. Then, the concept of pipelining is applied, and the optim function is parallelized with the rest of the functions. However, a problem shows up through the implementation. Therefore, the prediction idea is carried out to solve this problem.

#### 5.2.1. Optimization Algorithm

The baseline algorithm uses the Hildreth’s Quadratic Programming Solver (qpSolver), shown in Algorithm 1, to minimized the objective function. Nevertheless, there are two weaknesses that increase the execution time of the Optim function: the operations recurrence and the operation of array elements. The operational recurrence was solved by reusing the previously calculated data. The calculation of the inverse of the  $E$  matrix should be highlighted, as it is mentioned in [41]. The  $E$  matrix is not modified during the execution, following this, the inverse operation can be calculated once. For this reason, the new algorithm requires the  $E^{-1}$  as an input, and the inversion operation is calculated in the MPC Offline function. The operation of arrays elements was boosted using optimized functions that works directly with arrays instead of the elements. The modified algorithm can be seen in Algorithm 2. A new profile analysis was performed, with these modifications, thus,

the latency of the Optim function is reduced from 1290.78  $\mu\text{s}$  to 226.75  $\mu\text{s}$  for the study case 1 and from 3491  $\mu\text{s}$  to 394  $\mu\text{s}$  for the study case 2.

---

**Algorithm 1:** Hildreth's QP Solver (Original)
 

---

```

Input :  $E, F, M, g$                                 /* Current data information  $E, F, M, \gamma$  */
Output:  $DU$                                         /* Optimal trajectory  $\Delta U(k)$  */
begin
1   $n \leftarrow$  rows of  $M$ 
2   $DU \leftarrow -\text{inverse}(E) \cdot F$ 
3   $e \leftarrow 0$ 
4  for  $i$  from 0 to  $n - 1$  do
5  |   if  $M(i, \text{all}) \cdot DU > g(i)$  then
6  |   |    $e \leftarrow e + 1$ 
7  |   else
8  |   |    $e \leftarrow e + 0$ 
9  if  $e = 0$  then
10 |   Return  $DU$ 
11  $P \leftarrow M \cdot \text{inverse}(E) \cdot \text{transpose}(M)$ 
12  $d \leftarrow M \cdot \text{inverse}(E) \cdot F + g$ 
13  $n \leftarrow$  size of  $d$ 
14  $l \leftarrow$  vector of zeros of dimension  $n$           /*  $l = \lambda$  */
15 for  $a$  from 0 to 37 do
16 |    $lp \leftarrow l$ 
17 |   for  $i$  from 0 to  $n - 1$  do
18 |   |    $w \leftarrow P(i, \text{all}) \cdot l - P(i, i) \cdot l(i)$ 
19 |   |    $w \leftarrow w + d(i)$ 
20 |   |    $la \leftarrow -w / P(i, i)$ 
21 |   |    $l(i) \leftarrow \max(0, la)$ 
22 |    $e \leftarrow \text{transpose}(l - lp) \cdot (l - lp)$ 
23 |   if  $e < 100 \times 10^{-9}$  then
24 |   |   BREAK
25  $DU \leftarrow -\text{inverse}(E) \times F - \text{inverse}(E) \cdot \text{transpose}(M) \cdot l$ 
26 Return  $DU$ 

```

---

**Algorithm 2:** Hildreth's QP Solver (Improved)

---

```

Input :  $E, F, M, g$  /* Current data information  $E^{-1}, F, M, \gamma$  */
Output:  $DU$  /* Optimal trajectory  $\Delta U$  */
begin
1 |  $n \leftarrow$  rows of  $M$ 
2 |  $DU \leftarrow -Ei \cdot F$ 
3 |  $e \leftarrow \text{OR}(M \cdot DU > g)$ 
4 | if  $e = 1$  then
5 | |  $P \leftarrow M \cdot Ei \cdot \text{transpose}(M)$ 
6 | |  $d \leftarrow M \cdot Ei \cdot F + g$ 
7 | |  $l \leftarrow$  vector of zeros of dimension  $n$ 
8 | | for  $a$  from 0 to 37 do
9 | | |  $lp \leftarrow l$ 
10 | | | for  $i$  from 0 to  $n - 1$  do
11 | | | |  $w \leftarrow P(i, \text{all}) \cdot l - P(i, i) \cdot l(i) + d(i)$ 
12 | | | |  $la \leftarrow -w / P(i, i)$ 
13 | | | |  $l(i) \leftarrow \max(0, la)$ 
14 | | |  $e \leftarrow \text{transpose}(l - lp) \cdot (l - lp)$ 
15 | | | if  $e < 100 \times 10^{-9}$  then
16 | | | | BREAK
17 | |  $DU \leftarrow DU - Ei \cdot \text{transpose}(M) \cdot l$ 
18 | Return  $DU$ 

```

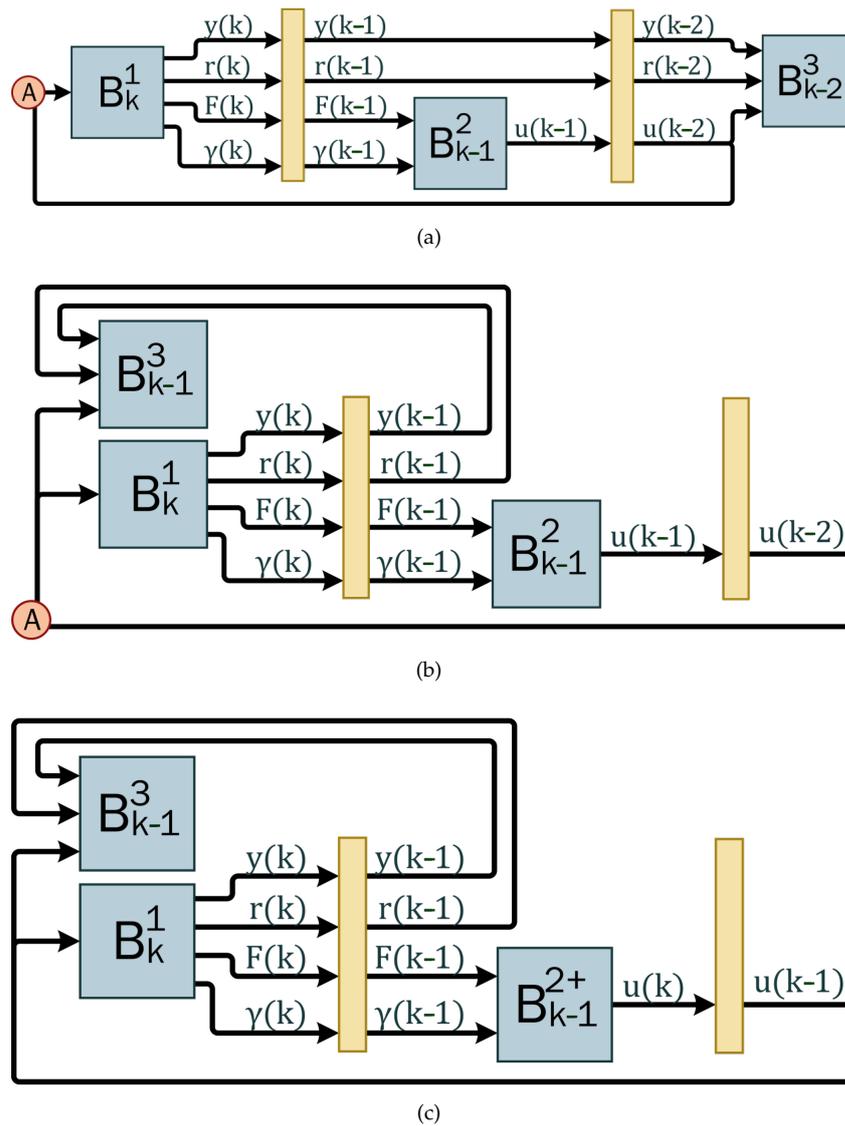
---

## 5.2.2. Pipelining

The Online functions from every SCI, shown in Table 5, can be arranged in 3 bundles. Bundle 1 (Latency: 233.938  $\mu\text{s}$ ) packs functions RDSen, SPGen, Obs, Fcalc, and  $\gamma$ Calc. Bundle 2 (Latency: 226.75  $\mu\text{s}$ ) include the modified Optim function. Bundle 3 (Latency: 76.670  $\mu\text{s}$ ) packs functions WRAct and Graph. The concept of pipelining, Section 3.1 can be extrapolated to work with bundles instead of instructions for both study cases. Figure 6 uses the notation  $B_k^n$  to indicate the bundle  $n$  of the SCI  $k$ . As show in Figure 6a, the execution is performed running a bundle per step. In Figure 6b, Bundle 1 and 3 are executed in the same step, in order to balance the time execution between step 1 (310.608  $\mu\text{s}$ ) and 2 (226.750  $\mu\text{s}$ ). Currently, the bundles  $B_k^1, B_{k-1}^2, B_{k-1}^3$  are run within the same PI.

## 5.2.3. Synchronization Error Troubleshooting

Using the Figure 5 as reference, a synchronization error can be noticed in node  $A$  of the Figure 6a,b. In Figure 6b,  $B_k^1$  (Function Obs and  $\gamma$ Calc) and  $B_{k-1}^3$  (Function WRAct and Graph) needs  $u(k-1)$  as input, but they are receiving  $u(k-2)$ . The bundle  $B_k^2$  represent the Optim function that can be seen in Algorithm 3. The Optim function calculates  $u(k)$  based on the iteration data  $F(k), \gamma(k)$ . A modification of the algorithm Optim can take place. The Algorithm 4, shows a new Optim function called Optim+. The Optim+ function takes advantage of the QP Solver definition to compute the optimal  $u(k)$  based on the previous iteration data  $F(k-1)$  and  $\gamma(k-1)$ . In Figure 6c, the  $B_k^{2+}$  (Optim+ function) is implemented to solve the synchronization error.



**Figure 6.** Optimization methodology: (a) Executing a bundle per step. (b) Executing Bundle 1 and Bundle 3 in the same step. (c) Executing Bundle 1 and Bundle 3 in the same step, and implementing Optim+Function.

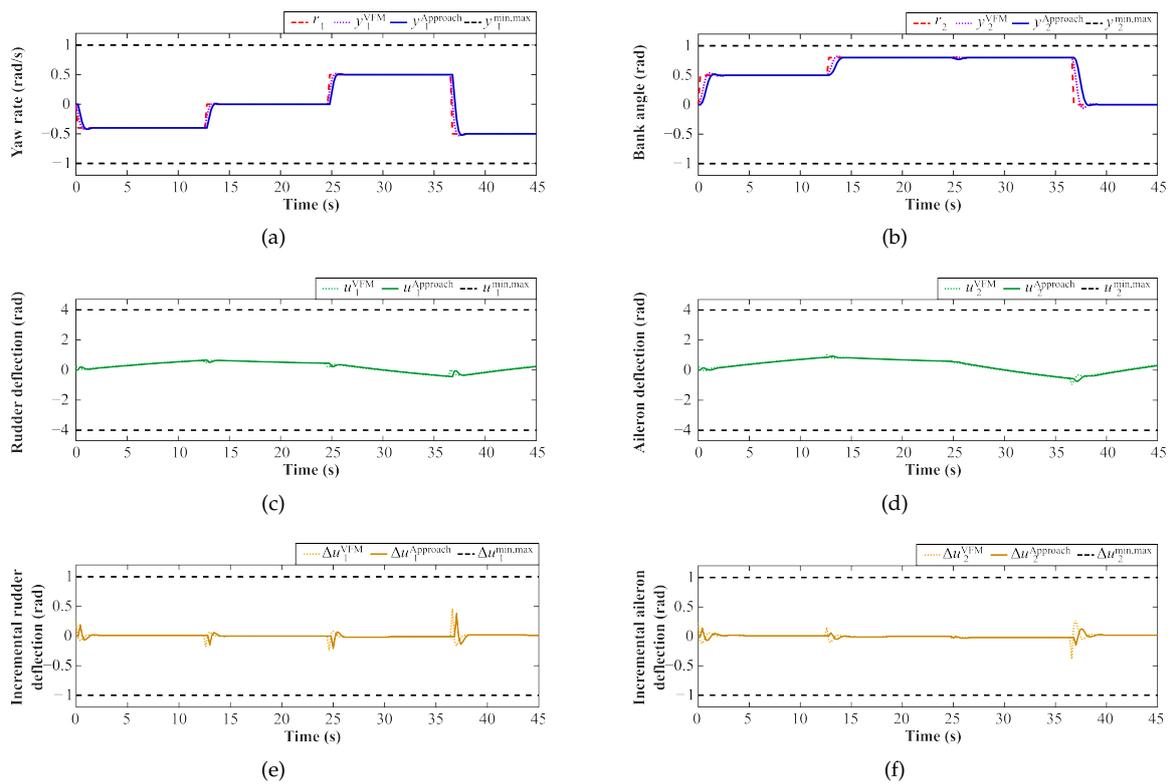
**Algorithm 3:** Optim function

```

Input :  $F_k, g_k$  /* Iteration data  $F_k, \gamma_k$  */
Output:  $u$  /* Manipulation variable  $u_k$  */
begin
1  $Du_k \leftarrow \mathbf{Hild}(E_i, F_k, M, g_k)$ 
2  $u \leftarrow u + Du_k(0 : nu - 1)$ 

```





**Figure 8.** Results for the jet transport aircraft under VFM and Approach: (a) Yaw rate. (b) Bank angle. (c) Rudder deflection. (d) Aileron deflection. (e) Incremental rudder deflection. (f) Incremental aileron deflection.

First, an enhancement is obtained by the implementation of the improvements to the Hildreth’s Algorithm. This reduces its latency from 1290.78  $\mu\text{s}$  to 226.75  $\mu\text{s}$ . Reducing the execution time by 1064  $\mu\text{s}$ . Then, the strategy of pipelining is implemented making an additional reduction of 1037  $\mu\text{s}$ . Thus, the total execution time is reduced 2101  $\mu\text{s}$ . Additionally, the speedup factor is calculated according to the execution times average presented in Table 6. Here, the speedup factor is  $4.6\times$ , (50.64% by the Hildreth’s modification and 49.35% by the pipeline implementation).

Furthermore, considering that both implementations are run in multicore execution, the processor utilization is evaluated, Table 7.

**Table 7.** Processor usage comparison.

	VFM	Approach
CPU 0	79.22%	43.29%
CPU 1	56.07%	40.48%

As it can be seen, with the present research work, the processor usage was released 35.93% for CPU 0 (from 79.22% to 43.29%) and 15.59% for CPU 1 (from 56.07% to 40.48%).

6.2. Study Case 2—Results

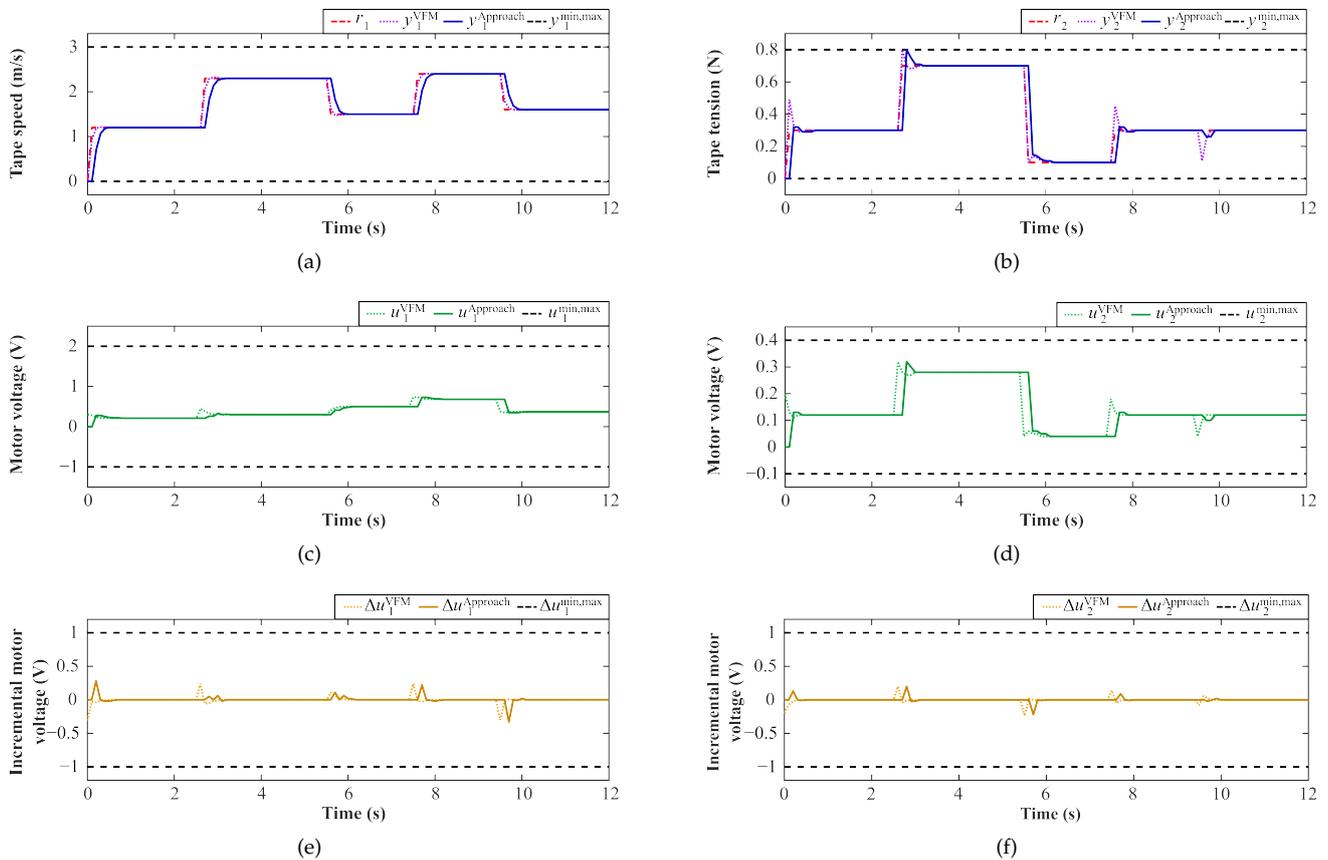
Considering a prediction horizon  $N_p = 20$ ,  $N_c = 15$  and a  $R_u = I_{30 \times 30}$ , the simulation results are shown in Figure 9, the controlled variable, the manipulated variables, and the incremental variable are plotted for both: the present work and the baseline algorithm.

Both implementations execute 1000 iterations, and a profile analysis is executed. The results from the profile analysis is shown in Table 8.

**Table 8.** Execution time comparison.

	VFM	Approach
Performance	170.736 Hz	541.7118 Hz
Average	5857 $\mu$ s	1846 $\mu$ s
Variance ( $s^2$ )	$5.46 \times 10^{-20}$	$2.08 \times 10^{-19}$
Maximum	39,660 $\mu$ s	27,093 $\mu$ s

According to the implementation of the modified Hildreth’s Algorithm, latency is considerably reduced from 3491  $\mu$ s to 394  $\mu$ s. Thus, the execution time is reduced 3097  $\mu$ s. Additionally, when the strategy of pipelining is implemented making an additional reduction of 914.1  $\mu$ s. Thus, the total execution time is reduced 4011 $\mu$ s. Additionally, the speedup factor is calculated according to the execution times average presented in Table 8. Here, the speedup factor is  $3.17\times$ , (72.21% by the Hildreth’s modification and 22.79% by the pipeline implementation).



**Figure 9.** Results for the tape transport system under VFM and Approach: (a) Tape speed. (b) Tape tension. (c) Motor voltage supply reel. (d) Motor voltage take-up reel. (e) Incremental motor voltage supply reel. (f) Incremental motor voltage take-up reel.

Furthermore, considering that both implementations are run in multicore execution, the processor utilization is evaluated, Table 9.

**Table 9.** Processor usage comparison.

	VFM	Approach
CPU 0	82.50%	53.49%
CPU 1	70.40%	56.84%

As it can be seen, with the present research work, the processor usage was released 29.01% for CPU 0 (from 82.50% to 53.49%) and 13.56% for CPU 1 (from 70.40% to 56.84%).

## 7. Conclusions

This paper proposes a new software-based implementation for a constrained Multi-Input-Multi-Output predictive control algorithm embedded in the NI myRIO 1900 using NI LabVIEW Real-Time Module. Based on the computer architecture concepts, a software-based implementation is proposed. Thus, as an evaluation case, the proposed implementation is used to control a jet transport aircraft during cruise flight.

Experimental results shows that the new software-based implementation has a good performance regardless the computational modification effected. Additionally, compared with the baseline algorithm implementations, there is a significant improvement on the execution time without affecting the tracking performance for both study cases. The proposed software-based implementation reaches a speedup factor up to  $4.6\times$  for the jet transport aircraft and  $3.17\times$  for the tape transport system with respect to the baseline implementation. In addition, the processor usage was released for both study cases, 35.93% and 29.01% in CPU 0 and 15.59% and 13.56% in CPU 1, respectively. Thus, the obtained results do benefit the implementation of MIMO MPC techniques by reducing the execution time and the computational load.

**Author Contributions:** Conceptualization, A.F.-C. and A.A.; methodology, A.F.-C., A.A. and A.P.; validation, A.F.-C., D.S., C.S., A.A. and F.B.-C.; investigation, A.F.-C., A.A., A.P., F.B.-C., C.S. and D.S.; resources, A.F.-C., C.S., D.S. and A.P.; writing—original draft, A.P., D.S. and C.S.; writing—review and editing, A.F.-C., D.S., C.S., A.A. and F.B.-C.; supervision, A.F.-C. and A.A.; project administration, A.F.-C.; funding acquisition, C.S. and D.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Data is contained within the article.

**Acknowledgments:** The authors would like to thank Consejo Nacional de Ciencia y Tecnología (CONACyT) and Tecnológico de Monterrey for the financial support for conducting the present research. Thanks also to the Nanosensors and Devices and Robotics Research Groups from the School of Engineering and Sciences of Tecnológico de Monterrey for the support given for the development of this work. Finally, thank students Juan Carlos Triana Vela and Ian De La Garza for their fruitful discussions, figures and tables editing, and especially for the analysis of the simulation results.

**Conflicts of Interest:** The author declares no conflict of interest.

## References

1. Jeng, J.C.; Chang, Y.J.; Lee, M.W. Novel design of dynamic matrix control with enhanced decoupling control performance. In *Computer Aided Chemical Engineering*; Elsevier B.V.: Amsterdam, The Netherlands, 2018; Volume 44, pp. 541–546. [[CrossRef](#)]
2. Abro, G.E.M.; Zulkifli, S.A.B.; Ali, Z.A.; Asirvadam, V.S.; Chowdhry, B.S. Fuzzy Based Backstepping Control Design for Stabilizing an Underactuated Quadrotor Craft under Unmodelled Dynamic Factors. *Electronics* **2022**, *11*, 999. [[CrossRef](#)]
3. Musa, A.; Pipicelli, M.; Spano, M.; Tufano, F.; De Nola, F.; Di Blasio, G.; Gimelli, A.; Misul, D.A.; Toscano, G. A Review of Model Predictive Controls Applied to Advanced Driver-Assistance Systems. *Energies* **2021**, *14*, 7974. [[CrossRef](#)]
4. Liu, F.; Li, H.; Liu, L.; Zou, R.; Liu, K. A control method for IPMSM based on active disturbance rejection control and model predictive control. *Mathematics* **2021**, *9*, 760. [[CrossRef](#)]
5. Bécsi, T. Quasi-Linear Parameter Varying Modeling and Control of an Electromechanical Clutch Actuator. *Mathematics* **2022**, *10*, 1473. [[CrossRef](#)]
6. Ławryńczuk, M.; Marusak, P.M.; Chaber, P.; Seredyński, D. Initialisation of Optimisation Solvers for Nonlinear Model Predictive Control: Classical vs. Hybrid Methods. *Energies* **2022**, *15*, 2483. [[CrossRef](#)]
7. Zavitsanou, S.; Chakrabarty, A.; Dassau, E.; Doyle, F.J. Embedded control in wearable medical devices: Application to the artificial pancreas. *Processes* **2016**, *4*, 35. [[CrossRef](#)]

8. Baca, T.; Loianno, G.; Saska, M. Embedded Model Predictive Control of Unmanned Micro Aerial Vehicles. In Proceedings of the 2016 21st International Conference on Methods and Models in Automation and Robotics (MMAR), Miedzyzdroje, Poland, 29 August–1 September 2016; pp. 992–997. [[CrossRef](#)]
9. Ibañez, C.; Ocampo-Martinez, C.; Gonzalez, B. Embedded optimization-based controllers for industrial processes. In Proceedings of the 2017 IEEE 3rd Colombian Conference on Automatic Control (CCAC), Cartagena, Colombia, 18–20 October 2017; pp. 1–6. [[CrossRef](#)]
10. Wang, L. *Model Predictive Control System Design and Implementation Using MATLAB*, 1st ed.; Springer: Perth, Australia, 2009. [[CrossRef](#)]
11. Vazquez, S.; Rodriguez, J.; Rivera, M.; Franquelo, L.G.; Norambuena, M. Model predictive control for power converters and drives: Advances and trends. *IEEE Trans. Ind. Electron.* **2016**, *64*, 935–947. [[CrossRef](#)]
12. Karamanakos, P.; Geyer, T. Guidelines for the design of finite control set model predictive controllers. *IEEE Trans. Power Electron.* **2019**, *35*, 7434–7450. [[CrossRef](#)]
13. Hu, J.; Ding, B. One-step ahead robust MPC for LPV model with bounded disturbance. *Eur. J. Control* **2020**, *52*, 59–66. [[CrossRef](#)]
14. Zafra, E.; Vazquez, S.; Guzman Miranda, H.; Sanchez, J.A.; Marquez, A.; Leon, J.I.; Franquelo, L.G. Efficient FPSoc prototyping of FCS-MPC for three-phase voltage source inverters. *Energies* **2020**, *13*, 1074. [[CrossRef](#)]
15. Luo, Q.; Nguyen, A.T.; Fleming, J.; Zhang, H. Unknown input observer based approach for distributed tube-based model predictive control of heterogeneous vehicle platoons. *IEEE Trans. Veh. Technol.* **2021**, *70*, 2930–2944. [[CrossRef](#)]
16. Ju, Z.; Zhang, H.; Tan, Y. Distributed stochastic model predictive control for heterogeneous vehicle platoons subject to modeling uncertainties. *IEEE Intell. Transp. Syst. Mag.* **2021**, *14*, 25–40. [[CrossRef](#)]
17. Pang, H.; Liu, M.; Hu, C.; Liu, N. Practical Nonlinear Model Predictive Controller Design for Trajectory Tracking of Unmanned Vehicles. *Electronics* **2022**, *11*, 1110. [[CrossRef](#)]
18. Lee, T.; Kang, Y. Performance Analysis of Deep Neural Network Controller for Autonomous Driving Learning from a Nonlinear Model Predictive Control Method. *Electronics* **2021**, *10*, 767. [[CrossRef](#)]
19. Richter, S.; Jones, C.N.; Morari, M. Real-time input-constrained MPC using fast gradient methods. In Proceedings of the IEEE Conference on Decision and Control, Shanghai, China, 15–18 December 2009. [[CrossRef](#)]
20. He, M.; Ling, K.V. Model Predictive Control On A Chip. In Proceedings of the 2005 International Conference on Control and Automation, Budapest, Hungary, 26–29 June 2005; pp. 528–532. [[CrossRef](#)]
21. Sotelo, D.; Favela-Contreras, A.; Kalashnikov, V.V.; Sotelo, C. Model Predictive Control with a Relaxed Cost Function for Constrained Linear Systems. *Math. Probl. Eng.* **2020**, *2020*, 7485865. [[CrossRef](#)]
22. Bang, H.; Lee, Y.S. Embedded Model Predictive Control for Enhancing Tracking Performance of a Ball-and-Plate System. *IEEE Access* **2019**, *7*, 39652–39659. [[CrossRef](#)]
23. Hýl, R.; Wagnerová, R. Design and Realization of Embedded Model Predictive Controller with Software Support. In Proceedings of the 2016 17th International Carpathian Control Conference (ICCC), Tatranska Lomnica, Slovakia, 29 May 2016–1 June 2016; pp. 259–264. [[CrossRef](#)]
24. Yu, L.; Goldsmith, A.; Di Cairano, S. Efficient convex optimization on GPUs for embedded model predictive control. In Proceedings of the General Purpose GPUs, GPGPU-10 2017, Austin, TX, USA, 4–8 February 2017; pp. 12–21. [[CrossRef](#)]
25. Cimini, G.; Bernardini, D.; Levijoki, S.; Bemporad, A. Embedded Model Predictive Control With Certified Real-Time Optimization for Synchronous Motors. *IEEE Trans. Control Syst. Technol.* **2020**. [[CrossRef](#)]
26. Boshkovski, G.; Stojanovski, G.; Stankovski, M. Development of embedded model predictive controller. In Proceedings of the 2017 13th IEEE International Conference on Control & Automation (ICCA), Ohrid, Macedonia, 3–6 July 2017; pp. 76–81. [[CrossRef](#)]
27. Nermin, H.; Prljaca, N. An Implementation and Evaluation of Fast Embedded Model Predictive Control. In Proceedings of the 2020 19th International Symposium INFOTEH-JAHORINA (INFOTEH), East Sarajevo, Bosnia and Herzegovina, 18–20 March 2020.
28. Wang, W.C.; Liu, T.H.; Syaifudin, Y. Model Predictive Controller for a Micro-PMSM-Based Five-Finger Control System. *IEEE Trans. Ind. Electron.* **2016**, *63*, 3666–3676. [[CrossRef](#)]
29. Guzman, R.; De Vicuna, L.G.; Camacho, A.; Miret, J.; Rey, J.M. Receding-Horizon Model-Predictive Control for a Three-Phase VSI with an LCL Filter. *IEEE Trans. Ind. Electron.* **2019**, *66*, 6671–6680. [[CrossRef](#)]
30. Pannocchia, G.; Gabiccini, M.; Artoni, A. Offset-free MPC explained: Novelties, subtleties, and applications. *IFAC-PapersOnLine* **2015**, *48*, 342–351. [[CrossRef](#)]
31. Sotelo, C.; Favela-Contreras, A.; Beltrán-Carbajal, F.; Dieck-Assad, G.; Rodríguez-Cañedo, P.; Sotelo, D. A Novel Discrete-time Nonlinear Model Predictive Control Based on State Space Model. *Int. J. Control. Autom. Syst.* **2018**, *16*, 2688–2696. [[CrossRef](#)]
32. Alamir, M. *A Pragmatic Story of Model Predictive Control: Self-Contained Algorithms and Case-Studies*; CreateSpace Independent Publishing Platform: Scotts Valley, CA, USA, 2013.
33. Singh, S.; Murthy, T.V.R. Simulation of sensor failure accommodation in flight control system of transport aircraft a modular approach. *World J. Model. Simul.* **2015**, *11*, 55–68.
34. Singh, S.; Rama Murthy, T. Design of an optimal yaw damper for 747 jet aircraft model. In *Emerging Research in Electronics, Computer Science and Technology*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 801–810.

35. Mathur, P.D.; Messner, W.C. Controller development for a prototype high-speed low-tension tape transport. *IEEE Trans. Control Syst. Technol.* **1998**, *6*, 534–542. [[CrossRef](#)]
36. Lu, Y.; Messner, W.C. Robust servo design for tape transport. In Proceedings of the 2001 IEEE International Conference on Control Applications (CCA'01) (Cat. No. 01CH37204), Mexico City, Mexico, 7 September 2001; pp. 1014–1019.
37. Tenne, D.; Singh, T. Robust feed-forward/feedback design for tape transport. In Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit, Providence, RI, USA, 16–19 August 2004; p. 5119.
38. Baumgart, M.D.; Pao, L.Y. Robust control of nonlinear tape transport systems with and without tension sensors. *J. Dyn. Sys. Meas. Control.* **2007**, *129*, 41–55. [[CrossRef](#)]
39. Baumgart, M.D.; Pao, L.Y. Robust control of tape transport systems with no tension sensor. In Proceedings of the 2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No. 04CH37601), Nassau, Bahamas, 14–17 December 2004; Volume 4, pp. 4342–4349.
40. National Instruments. *NI myRIO-1900. User Guide and Specifications*; Technical report; National Instruments: Austin, TX, USA, 2013.
41. Zhu, Q.; Onori, S.; Prucka, R. Pattern recognition technique based active set QP strategy applied to MPC for a driving cycle test. In Proceedings of the American Control Conference, Chicago, IL, USA, 1–3 July 2015; pp. 4935–4940. [[CrossRef](#)]