Article

Fast Real-Time Model Predictive Control for a Ball-on-Plate Process

Krzysztof Zarzycki * and Maciej Ławryńczuk

Faculty of Electronics and Information Technology, Institute of Control and Computation Engineering, Warsaw University of Technology, ul. Nowowiejska 15/19, 00-665 Warsaw, Poland; M.Lawrynczuk@ia.pw.edu.pl
* Correspondence: Krzysztof.Zarzycki@pw.edu.pl

Abstract: This work is concerned with an original ball-on-plate laboratory process. First, a simplified process model based on state–space process description is derived. Next, a fast state–space MPC algorithm is discussed. Its main advantage is computational simplicity: the manipulated variables are found on-line using explicit formulas with parameters calculated off-line; no real-time optimization is necessary. Software and hardware implementation details of the considered MPC algorithm using the STM32 microcontroller are presented. Tuning of the fast MPC algorithm is discussed. To show the efficacy of the MPC algorithm, it is compared with the classical PID and LQR controllers.

Keywords: ball-on-plate process; model predictive control; PID controller; LQR controller; modeling

check for updates

Citation: Zarzycki, K.; Ławryńczuk, M. Fast Real-Time Model Predictive Control for a Ball-on-Plate Process. *Sensors* **2021**, *21*, 3959. https://doi.org/10.3390/s21123959

Academic Editors: Haruo Kobayashi and Ramón Vilanova Arbós

Received: 17 March 2021
Accepted: 7 June 2021
Published: 8 June 2021

1. Introduction

Laboratories have a very important role in the successful education of engineering students. As far as automatic control is concerned, students use the classical laboratory processes such as the water tanks, the inverted pendulum, the magnetic levitation system and the ball-on-plate benchmark. In particular, the ball-on-plate process is very interesting in undergraduate and graduate courses since it requires multivariable stabilizing control [1–3]. Furthermore, it is a fast dynamical system that requires short sampling times of the controller. Because control of fast, unstable and multivariable systems is very important in different practical applications, the ball-on-plate process may be successfully used in control courses.

Since the ball-on-plate process may be described by state equations, Linear Quadratic Regulator (LQR) is frequently used [4–6]. Similarly, applications of the Sliding Mode Control (SMC) [5,7–9] and adaptive SMC [8,9] are reported. Furthermore, one may also try to use simple PD [2] or PID controllers [3,5,9,10]. More advanced solutions include fuzzy controllers [5,9,10], neuro-controllers [11], feedback linearization controllers [12] and disturbance-observer-based friction compensation schemes [13]. Finally, some works consider advanced Model Predictive Control (MPC) [14]. Historically, MPC algorithms have been used for industrial process control; example processes are chemical reactors [15], distillation columns [16], pasteurization plants [17] and fermentation systems [18]. Currently, MPC algorithms are also used for numerous other processes; example applications are: heating, ventilation and air conditioning systems [19], robotic manipulators [20], electromagnetic mills [21], servomotors [22], quadrotors [23], autonomous vehicles [24], unmanned aerial vehicles [25] and stochastic systems [26]. MPC algorithms have also been used for the considered ball-on-plate process. Unfortunately, the solutions presented in the literature are computationally demanding as they require on-line optimization carried out at each sampling instant. The MPC algorithm in which a nonlinear model is used for prediction and a nonlinear optimization task must be solved at each sampling instant on-line is presented in [27] (only simulations are shown). The MPC method based on a linearized model and quadratic optimization used on-line to calculate the manipulated variables is discussed in [28,29]. The objective of this work is to develop a fast MPC algorithm for the ball-on-plate process which does not need any on-line optimization and compare its performance with the classical PID and LQR controllers very frequently used for the considered process.

When the sampling time is short, e.g., of millisecond order, computational efficiency of MPC is an important issue [30]. The time required to perform calculations at every execution of the MPC algorithm must be shorter than the sampling time. There are a few methods that may be used to obtain short calculation time of MPC algorithms. First, specialized fast on-line optimization methods may be used, especially tailored for MPC applications [31]. Secondly, in the constrained explicit MPC algorithms [32], on-line optimization is not used, but a number of local explicit controllers are used. Successful implementation of the explicit MPC is reported [33], even when the available memory is limited [34]. Thirdly, the numerical optimization procedure used in the MPC algorithm may be replaced by a specially designed neural network which acts as a neural optimizer [35]. Fast explicit (analytical) MPC is possible in which the manipulated variables are calculated analytically using explicit formulas and next projected onto the admissible set determined by the constraints [36]. As a result, on-line optimization is not required. A practical application of this approach is described in [37], but only for processes described by simple step-response models and by discrete transfer functions (i.e., difference equations). This work follows the idea presented in [36,37] for state–space models. Finally, some specialized methods have been developed to handle constraints in on-line MPC optimization that make it possible to use sampling times of the order of milliseconds [16,38].

This work is concerned with an original ball-on-plate laboratory process. The contribution of this work is three-fold:

1. Simplified process modeling based on state–space process description is derived.
2. A fast state–space MPC algorithm is discussed and next applied to the considered ball-on-plate system. Its main advantage is computational simplicity: the manipulated variables are computed on-line using explicit formulas with parameters calculated off-line, no on-line optimization is necessary (nonlinear or quadratic). The presented state–space MPC algorithm uses a simple, yet very efficient state and output disturbance estimation necessary for prediction in state–space MPC [39].
3. Software and hardware implementation details of the MPC algorithm are presented.

The article is organized in the following way. Section 2 shortly describes the laboratory ball-on-plate process and introduces its model. The MPC algorithm is detailed in Section 3 while Section 4 deals with software and hardware implementation of the MPC algorithm using the STM32 microcontroller. Section 5 reports tuning of MPC and discusses results of experiments in which the discussed MPC scheme is compared with the classical PID and LQR controllers. Finally, Section 6 summarizes the whole article.

2. Ball-on-Plate Process

This work is concerned with an original ball-on-plate laboratory system built at the Warsaw University of Technology and shown in Figure 1. Its construction details are given in [40]. In short, it includes: the touch pad, two digital servomotors, two servo arms and two servo rods, the microcontroller development board and some additional elements.

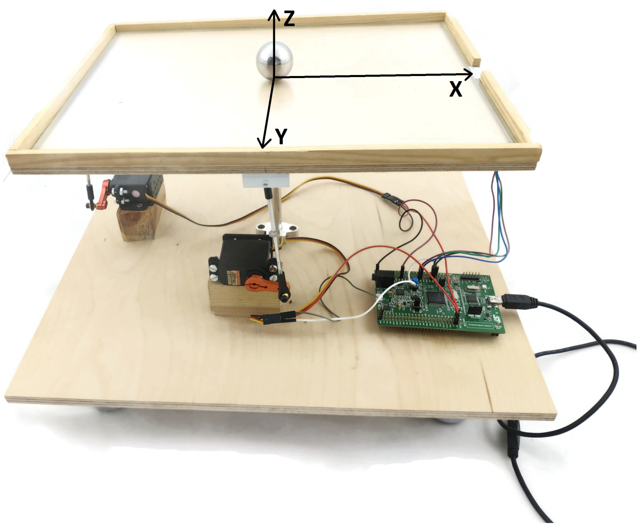


Figure 1. The ball-on-plate process

The process has two manipulated and two controlled variables. The Pulse Width Modulation (PWM) signals are used to control the digital servomotors. The plate rotates around the X and Y axes. Two signals that define the ball position on the top panel are the controlled variables. The actual ball position on the resistive touch panel is determined by two Analog to Digital Converters (ADCs). The STM32 F401C Disco microcontroller development board is used for implementation of control algorithms and communication with the PC computer (it is only used for saving the data and plotting the results).

Provided that friction and air resistance forces are neglected as well as angular velocities of the surface plate are low, the balance of the forces on the X axis is

$$F_{\mathrm{tx}}(t) + F_{\mathrm{rx}}(t) = F_{\mathrm{gx}}(t). \tag{1}$$

Translation, rotation and gravity forces for the X axis are

$$F_{\mathrm{tx}}(t) = m\ddot{x}_{\mathrm{ball}}(t), \tag{2}$$
$$F_{\mathrm{rx}}(t) = \tfrac{2}{5}m\ddot{x}_{\mathrm{ball}}(t), \tag{3}$$
$$F_{\mathrm{gx}}(t) = mg\sin\Phi(t), \tag{4}$$

where x_{ball} denotes the ball position along the X axis and Φ denotes the tilt angle of the plate in relation to the X axis. Similarly, for the Y axis, we have

$$F_{\mathrm{ty}}(t) + F_{\mathrm{ry}}(t) = F_{\mathrm{gy}}(t), \tag{5}$$

where the forces are

$$F_{\mathrm{ty}}(t) = m\ddot{y}_{\mathrm{ball}}(t), \tag{6}$$
$$F_{\mathrm{ry}}(t) = \tfrac{2}{5}m\ddot{y}_{\mathrm{ball}}(t), \tag{7}$$
$$F_{\mathrm{gy}}(t) = mg\sin\Theta(t), \tag{8}$$

where y_{ball} denotes the ball position along the Y axis and Θ denotes the tilt angle of the plate in relation to the Y axis. Using the forces (2)–(4) and (6)–(8), the model Equations (1) and (5) become

$$\ddot{x}_{\mathrm{ball}}(t) = \tfrac{5}{7}g\sin\Phi(t), \tag{9}$$
$$\ddot{y}_{\mathrm{ball}}(t) = \tfrac{5}{7}g\sin\Theta(t). \tag{10}$$

Let ϕ and θ denote rotation angles of the servo arm related to the Φ and Θ plate tilts, respectively. The relations are

$$\sin\Phi(t) = \tfrac{d}{L_\mathrm{x}}\sin\phi(t), \tag{11}$$
$$\sin\Theta(t) = \tfrac{d}{L_\mathrm{y}}\sin\theta(t), \tag{12}$$

where $d = 0.024$ m is the servo arm length, $L_\mathrm{x} = 0.165$ m and $L_\mathrm{y} = 0.135$ m stand for distances between the point of attachment of the servo rod to the plate and the cross joint for X and Y axes, respectively. The values of d, L_x and L_y are constant and do not change in time. To remove nonlinearity, we use the approximations $\sin\phi \approx \phi$, $\sin\theta \approx \theta$. It may be easily verified that for the angles $-20^\circ \leq \theta \leq 20^\circ$ and $-20^\circ \leq \phi \leq 20^\circ$, accuracy of the approximation used is very good. Using Equations (11) and (12), the model (9) and (10) becomes

$$\ddot{x}_\mathrm{ball}(t) = C_\mathrm{x}\phi(t), \tag{13}$$
$$\ddot{y}_\mathrm{ball}(t) = C_\mathrm{y}\theta(t), \tag{14}$$

where $C_\mathrm{x} = \frac{5}{7}g\frac{d}{L_\mathrm{x}} = 1.0189$, $C_\mathrm{y} = \frac{5}{7}g\frac{d}{L_\mathrm{y}} = 1.2453$. Equations (13) and (14) are expressed in the form of the classical linear continuous-time state–space model

$$\dot{x}(t) = \boldsymbol{A}_\mathrm{c}x(t) + \boldsymbol{B}_\mathrm{c}u(t), \tag{15}$$
$$y(t) = \boldsymbol{C}_\mathrm{c}x(t), \tag{16}$$

where the state, input and output vectors are

$$x = \begin{bmatrix} x_\mathrm{ball} \\ \dot{x}_\mathrm{ball} \\ y_\mathrm{ball} \\ \dot{y}_\mathrm{ball} \end{bmatrix}, \; u = \begin{bmatrix} \phi \\ \theta \end{bmatrix}, \; y = \begin{bmatrix} x_\mathrm{ball} \\ y_\mathrm{ball} \end{bmatrix}. \tag{17}$$

The model matrices are

$$\boldsymbol{A}_\mathrm{c} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \; \boldsymbol{B}_\mathrm{c} = \begin{bmatrix} 0 & 0 \\ C_\mathrm{x} & 0 \\ 0 & 0 \\ 0 & C_\mathrm{y} \end{bmatrix}, \; \boldsymbol{C}_\mathrm{c} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{18}$$

The discrete-time version of the model (15) and (16) is

$$x(k+1) = \boldsymbol{A}x(k) + \boldsymbol{B}u(k), \tag{19}$$
$$y(k) = \boldsymbol{C}x(k), \tag{20}$$

where the model matrices are

$$\boldsymbol{A} = \begin{bmatrix} 1 & T_\mathrm{s} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T_\mathrm{s} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \; \boldsymbol{B} = \begin{bmatrix} 0 & 0 \\ T_\mathrm{s}C_\mathrm{x} & 0 \\ 0 & 0 \\ 0 & T_\mathrm{s}C_\mathrm{y} \end{bmatrix}, \; \boldsymbol{C} = \boldsymbol{C}_\mathrm{c}, \tag{21}$$

and T_s denotes the sampling time.

3. Fast Model Predictive Control

Let $u = [u_1 \ \dots u_{n_\mathrm{u}}]^\mathrm{T}$, $x = [x_1 \ \dots x_{n_\mathrm{x}}]^\mathrm{T}$ and $y = \left[y_1 \ \dots y_{n_\mathrm{y}}\right]^\mathrm{T}$ be the vector of process inputs (manipulated variables), the vector of states and the vector of outputs (controlled variables), respectively. For the considered process, $n_\mathrm{u} = n_\mathrm{y} = 2$, $n_\mathrm{x} = 4$. At each sampling instant k of MPC, the following vector of increments is calculated [14]

$$\triangle \boldsymbol{u}(k) = \begin{bmatrix} \triangle u(k|k) \\ \vdots \\ \triangle u(k+N_\mathrm{u}-1|k) \end{bmatrix}, \tag{22}$$

where N_u is named the control horizon. The increments in Equation (22) are: $\triangle u(k|k) = u(k|k) - u(k-1)$ and $\triangle u(k+p|k) = u(k+p|k) - u(k+p-1|k)$ for $p = 1, \dots, N_\mathrm{u}-1$. The decision variables of MPC (22) are computed on-line as the result of minimization of the performance cost-function

$$J(k) = \sum_{p=1}^{N} \| y^{\mathrm{sp}}(k+p|k) - \hat{y}(k+p|k) \|^2_{\boldsymbol{\Psi}_p} + \sum_{p=0}^{N_\mathrm{u}-1} \| \triangle u(k+p|k) \|^2_{\boldsymbol{\Lambda}_p}. \tag{23}$$

The first part of the cost-function measures predicted control errors over the prediction horizon N. The vectors $y^{\mathrm{sp}}(k+p|k)$ and $\hat{y}(k+p|k)$ denote the required set-point of the controlled variables and the predicted vector, respectively, both for the sampling instant $k+p$ and calculated at the instant k. The second part of the cost-function measures changes of the manipulated variables, which usually should be penalized. The weighting matrices $\boldsymbol{\Psi}_p = \mathrm{diag}(\psi_{p,1}, \dots, \psi_{p,n_\mathrm{y}})$ and $\boldsymbol{\Lambda}_p = \mathrm{diag}(\lambda_{p,1}, \dots, \lambda_{p,n_\mathrm{u}})$ are of dimensionality $n_\mathrm{y} \times n_\mathrm{y}$ and $n_\mathrm{u} \times n_\mathrm{u}$, respectively. Having computed the vector (22), only the increments for the current instant are applied to the process. The state and output process variables are then measured (or estimated) and the whole computational task is repeated at the next sampling instant.

As proved in [14], when the state–space linear model (19) and (20) is used for prediction, the predicted trajectory of the controlled variables, which is a vector of length $n_\mathrm{y}N$

$$\hat{\boldsymbol{y}}(k) = \begin{bmatrix} \hat{y}(k+1|k) \\ \vdots \\ \hat{y}(k+N|k) \end{bmatrix}, \tag{24}$$

is given by the the following formula

$$\hat{\boldsymbol{y}}(k) = \widetilde{\boldsymbol{C}}\boldsymbol{M}_\mathrm{x}\triangle\boldsymbol{u}(k) + \widetilde{\boldsymbol{I}}_\mathrm{y}d(k) + \widetilde{\boldsymbol{C}}(\widetilde{\boldsymbol{A}}x(k) + \boldsymbol{V}(\boldsymbol{B}u(k-1) + \nu(k))), \tag{25}$$

where $\widetilde{\boldsymbol{C}} = \mathrm{diag}(\boldsymbol{C}, \dots, \boldsymbol{C})$ is the matrix of dimensionality $n_\mathrm{y}N \times n_\mathrm{x}N$, the matrix of dimensionality $n_\mathrm{x}N \times n_\mathrm{x}N_\mathrm{u}$ is

$$\boldsymbol{M}_\mathrm{x} = \begin{bmatrix} \boldsymbol{B} & \dots & \boldsymbol{0}_{n_\mathrm{x} \times n_\mathrm{u}} \\ (\boldsymbol{A} + \boldsymbol{I}_{n_\mathrm{x} \times n_\mathrm{x}})\boldsymbol{B} & \dots & \boldsymbol{0}_{n_\mathrm{x} \times n_\mathrm{u}} \\ \vdots & \ddots & \vdots \\ \left(\sum_{i=1}^{N-1} \boldsymbol{A}^i + \boldsymbol{I}_{n_\mathrm{x} \times n_\mathrm{x}}\right)\boldsymbol{B} & \dots & \left(\sum_{i=1}^{N-N_\mathrm{u}} \boldsymbol{A}^i + \boldsymbol{I}_{n_\mathrm{x} \times n_\mathrm{x}}\right)\boldsymbol{B} \end{bmatrix}, \tag{26}$$

and the matrices of dimensionality $n_\mathrm{x}N \times n_\mathrm{x}$ are

$$\widetilde{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A} \\ \vdots \\ \boldsymbol{A}^N \end{bmatrix}, \ \boldsymbol{V} = \begin{bmatrix} \boldsymbol{I}_{n_\mathrm{x} \times n_\mathrm{x}} \\ \vdots \\ \sum_{i=1}^{N-1} \boldsymbol{A}^i + \boldsymbol{I}_{n_\mathrm{x} \times n_\mathrm{x}} \end{bmatrix}. \tag{27}$$

In this work, a very efficient state and output disturbance model introduced in [14] and thoroughly discussed in [39] is used. Conversely to the classical augmented state approach [41–43], it does not require an extension of the state vector. The state disturbance vector is computed as the difference between the measured or observed state vector, $x(k)$, and the state variables calculated from the state model (19)

$$\nu(k) = x(k) - \boldsymbol{A}(k)x(k-1) - \boldsymbol{B}(k)u(k-1). \tag{28}$$

The output disturbance vector is computed as the difference between the measured output vector, $y(k)$, and the corresponding values calculated from the output model (20)

$$d(k) = y(k) - \boldsymbol{C}x(k). \tag{29}$$

The matrix $\widetilde{\boldsymbol{I}}_{\mathrm{y}} = [\boldsymbol{I}_{\mathrm{y}} \ldots \boldsymbol{I}_{\mathrm{y}}]^{\mathrm{T}}$, where $\boldsymbol{I}_{\mathrm{y}}$ is the matrix of dimensionality $n_{\mathrm{y}}N \times n_{\mathrm{y}}$ whose all entries are 1. Using the predicted trajectory (25), the minimized MPC cost-function (23) may be rewritten in the following vector-matrix form

$$\begin{aligned} J(k) = {} & \|\boldsymbol{y}^{\mathrm{sp}}(k) - \widetilde{\boldsymbol{C}}\boldsymbol{M}_{\mathrm{x}}\triangle\boldsymbol{u}(k) - \widetilde{\boldsymbol{I}}_{\mathrm{y}}d(k) - \widetilde{\boldsymbol{C}}(\widetilde{\boldsymbol{A}}x(k) + \boldsymbol{V}(\boldsymbol{B}u(k-1) + \nu(k)))\|^2_{\boldsymbol{\Psi}} \\ & + \|\triangle\boldsymbol{u}(k)\|^2_{\boldsymbol{\Lambda}}, \end{aligned} \tag{30}$$

where the set-point trajectory is the vector of length $n_{\mathrm{y}}N$

$$\boldsymbol{y}^{\mathrm{sp}}(k) = \begin{bmatrix} y^{\mathrm{sp}}(k+1|k) \\ \vdots \\ y^{\mathrm{sp}}(k+N|k) \end{bmatrix}. \tag{31}$$

Typically, minimization of the MPC cost-function (30) is performed subject to constraints. The most useful ones include magnitude and rate of change of the manipulated variables

$$u^{\min} \le u(k|k) \le u^{\max}, \tag{32}$$

$$\triangle u^{\min} \le \triangle u(k|k) \le \triangle u^{\max}, \tag{33}$$

where $u^{\min}$, $u^{\max}$, $\triangle u^{\min}$ and $\triangle u^{\max}$ are vectors of length n_{u}.

In this work, we derive a fast version of the state–space MPC controller [14]. A similar approach is discussed in [37] for processes described by simple discrete step-response models and difference equations. Minimization is performed without any constraints, but they are taken into account afterwards. Considering that the cost-function (30) is quadratic with respect to $\triangle\boldsymbol{u}(k)$, the optimal increments of the manipulated variables for the current sampling instant are calculated analytically, without on-line optimization

$$\triangle u(k|k) = \boldsymbol{K}_{n_{\mathrm{u}}}(\boldsymbol{y}^{\mathrm{sp}}(k) - \widetilde{\boldsymbol{C}}(\widetilde{\boldsymbol{A}}x(k) + \boldsymbol{V}(\boldsymbol{B}u(k-1) + \nu(k))) - \widetilde{\boldsymbol{I}}_{\mathrm{y}}d(k)), \tag{34}$$

where the matrix $\boldsymbol{K}_{n_{\mathrm{u}}}$ of dimensionality $n_{\mathrm{u}} \times n_{\mathrm{y}}N$ is

$$\boldsymbol{K}_{n_{\mathrm{u}}} = \left[\boldsymbol{I}_{n_{\mathrm{u}} \times n_{\mathrm{u}}} \boldsymbol{0}_{n_{\mathrm{u}} \times (n_{\mathrm{y}}N - n_{\mathrm{u}})}\right](\boldsymbol{M}_{\mathrm{x}}^{\mathrm{T}}\widetilde{\boldsymbol{C}}^{\mathrm{T}}\boldsymbol{\Psi}\widetilde{\boldsymbol{C}}\boldsymbol{M}_{\mathrm{x}} + \boldsymbol{\Lambda})^{-1}\boldsymbol{M}_{\mathrm{x}}^{\mathrm{T}}\widetilde{\boldsymbol{C}}^{\mathrm{T}}\boldsymbol{\Psi}. \tag{35}$$

Having calculated the unconstrained vector $\triangle u(k|k)$ from Equation (34), it is projected on the admissible set determined by the constraints (32) and (33) and the results are applied to the process. The projection procedure is

$$\begin{aligned}
&\text{if } \triangle u(k|k) < \triangle u^{\min}(k|k) \text{ then } \triangle u(k|k) = \triangle u^{\min},\\
&\text{if } \triangle u(k|k) > \triangle u^{\max} \text{ then } u(k|k) = \triangle u^{\max},\\
&u(k|k) = \triangle u(k|k) + u(k-1),\\
&\text{if } u(k|k) < u^{\min} \text{ then } u_n(k|k) = u^{\min},\\
&\text{if } u(k|k) > u^{\max} \text{ then } u_n(k|k) = u^{\max},\\
&u(k) = u(k|k).
\end{aligned} \tag{36}$$

for manipulated variables, i.e., for $n = 1, \ldots, n_{\text{u}}$.

Let us stress the fact that the vector $\boldsymbol{K}_{n_{\text{u}}}$ is calculated off-line for given model parameters and tuning coefficients of MPC. Hence, the explicit control law (34) depends only on precalculated parameters, the current set-point vector, estimations of the state and disturbance variables as well as on the value of the manipulated variables at the previous sampling instant.

4. Real-Time Implementation of MPC for Ball-on-Plate Process Using STM32 Microcontroller

4.1. Hardware Set-Up

In the experiments described below, the STM32 F401C Disco development board is used as the hardware platform for implementation of control algorithms. The considered development board offers all functionality required to develop a microprocessor-based embedded control system. Therefore, no other circuit boards are used in the system. The board uses the Cortex M4F core running at 84 MHz clocking and offers, among others: the Floating Point Unit (FPU), 256 kB flash memory and 64 kB SRAM memory. The microcontroller has a sufficiently large amount of RAM and the hardware FPU. The memory requirement is crucial since the MPC algorithm performs multiplication of matrices of large dimensionality. The STM32 F401C Disco contains all the I/O pins needed for the project.

The STM32 F401C Disco microcontroller development board is also responsible for communication with an external PC computer (using the serial port). This function is used only to send data later used to create plots presented in this work. All the calculations used in control algorithms are performed by the development board. Because of it, the floating point unit and a significant amount of memory are important features of the chosen microcontroller. The considered development board offers all necessary I/O pins, i.e., two PWM outputs to control the servomotors and two ADCs to read the ball's position in two dimensions.

The digital Hi-tech HS 5485 HB [44] servomotors are used as the actuators. The choice has been motivated by the fact that servos do not require the use of an additional control loop for engine rotation. Additionally, their relatively low cost is also an essential advantage. The servomotors are equipped with a proportional controller. The Pulse Width Modulation (PWM) signal is used to control them, the pulse cycle is equal to 20 ms and pulse width is in the 900–2100 µs range. Their rotational range is 60°, the maximum speed is $0.2\ \text{s}/60^{\circ}$.

The Green Touch 15″ 4-wire resistive screen pad is used [45] for detection of the ball location. In general, there are a few methods that may be used for touch detection: capacitive, infrared, surface acoustic wave and resistive. The resistive touch panel has been chosen because of its advantages: the ball may be made of any material, resistance to water and dust pollution and low cost. The main advantage, however, is the fact that the resistance touch screen is very easily controlled via the microcontroller's pins. The ADCs simply read the current X and Y position of the ball. The ball position is determined by measuring horizontal and vertical resistances of the plate. The main drawback of the used sensor is the fact that the touch panel is very sensitive to disturbances. The pressure acting on it must be sufficient. The ball's mass should be not lower than 70 g. If the force is too small, two different effects can be observed: no touch is detected and the sensor returns Not a Number (NaN) signal or the ball is temporarily detected in a different place, other than its actual location. If the ball is lying on the plane without motion, both effects occur very rarely. However, when the ball is constantly rolling on the surface of the plate, measurement errors appear quite often, even for balls that have a weight 100 g or more. It is due to the fact that ball sometimes bounces and breaks away from the plate surface for a short moment. In other moments the tilt of the plate is large enough to reduce the component of the gravity force perpendicular to the plate to a value that makes the pressure force acting on the screen not sufficient. The following filters are used to reduce the effects of the mentioned phenomena:

- A filter that finds out when the sensor stops detecting the ball. The touch panel then returns position values zero or NaN. When the filter detects such a value, it is ignored and the last meaningful number is taken as the current position value.
- A filter that finds out when the sensor detects an incorrect ball position. If the current position change is greater than a certain threshold, it is ignored.
- An arithmetical filter that collects n measurements of the ball position and calculates the ball position as:
$$x = \frac{\sum_i^n x_i}{n}, \quad y = \frac{\sum_i^n y_i}{n}. \tag{37}$$
  The ADCs of the microcontroller operate with much higher frequency than the developed controllers. Therefore, it is possible to collect more than 100 measurements in one control loop. The value $n = 50$ has been chosen experimentally, as the smallest one that is able to eliminate all measurement errors. This filter is necessary for the system to work properly. Without it, the measurement errors destabilize all control algorithms.
- A median filter, which could be used as an alternative to arithmetical filter described above.
- Additionally, a Kalman filter has been implemented in the system. Its main role is to serve as an observer that estimates the unmeasured ball velocity value later used in the MPC algorithm. However, during the experiments it was observed that using Kalman filter's estimates of the ball position instead of the true position measurements, helps to improve the quality of control slightly, due to elimination of small interferences that slipped through the arithmetical/median filter. In other words, the estimated position signal is slightly smoother than the measured signal.

Width of the panel is 322 mm, its height is 247 mm, which determines the size of the whole process.

The pins of the microcontroller used to connect it to sensors and actuators are:

- pin PB10 is configured as TIM3 timer's channel 3 and used to send PWM control signal to the X axis servomotor,
- pin PA1 is configured as TIM3 timer's channel 2 and used to send PWM control signal to the Y axis servomotor,
- pins PA2, PA3, PB0, PB1 are connected to the resistive touch panel, their configuration change in time as described in Table 1.

Table 1. STM32 F401C Disco pins configuration for communication with the touch panel.

Function	PA2	PA3	PB0	PB1
read x	output 1	ADC2	output 0	input no pullup
read y	ADC1	output 1	input no pullup	output 0

4.2. Software System

The software system offers implementation of three control algorithms: PID, LQR, MPC. The code is written in the C programming language. Keli µVision 5.0 is used as a programming environment. Additionally, other tools are used to simplify programming tasks: STMCube MX to configure the microcontroller's clocks frequencies and I/O pins, the STM-Studio to transfer data between the microcontroller and an external PC, MATLAB to find the controllers' settings.

An external PC computer is used only to archive and visualize various signals and data. All calculations are performed using the microcontroller. Matrix and vector multiplications are the most computationally demanding. The size of the matrices in the MPC controller is correlated with the sampling time. Therefore, it is important to choose an adequate sampling period. A short sampling time provides theoretically better control but requires long horizons and a lot of calculations whereas a longer one allows using shorter horizons and results in a less computationally demanding solution, but the control quality may be insufficient. Both prediction and control horizons have an impact on the dimensionality of vectors and matrices, the latter one determines the number of calculated decision variables.

The structure of the real-time software system is depicted in Figure 2. It uses three timers. Discrete-time control algorithms must work in a strict time regime. The sampling time is 50 ms. Timer 3 is therefore set to work with the frequency of 20 Hz. As described earlier, the ball position sensor tends to produce many measurement errors. At each cycle, the timer routine begins with collecting n measurements of ball position from the touch panel via the ADCs of the microcontroller. The converters are sampled with higher frequency; therefore, during the 50 ms of the control loop, many measurements can be collected. Depending on the user's preference, the arithmetic filter or the median filter is then used to generate the filtered current position of the ball as described by Equation (37). The best results are achieved when $n = 50$. For lower values of n, measurement errors are frequent. For larger values, the measurements take too much time and cannot be performed in one control loop cycle.

Sometimes, the sensor is not pressed with force high enough and a false ball position is obtained. An additional function checks if the current measurement does make sense in relation to past measurements. Furthermore, the Kalman filter is used to generate estimates of the ball position and velocity based on the filtered measurement as well as the model (19) and (20). The control algorithm chosen by the user generates new control values. Finally, the ball position measurements are updated.

The servomechanisms' control signals have a specific frequency, typically 50 Hz. Therefore, timer 2 is used. Its only purpose is to send the PWM signals from two of its channels to servomechanisms with duty cycle calculated based upon the values generated by the chosen control algorithm.

Timer 4 changes the set position of the ball. Its frequency varies depending on the reference trajectory chosen by the user. If the timer is disabled, the user can change the set-points by pressing a button.

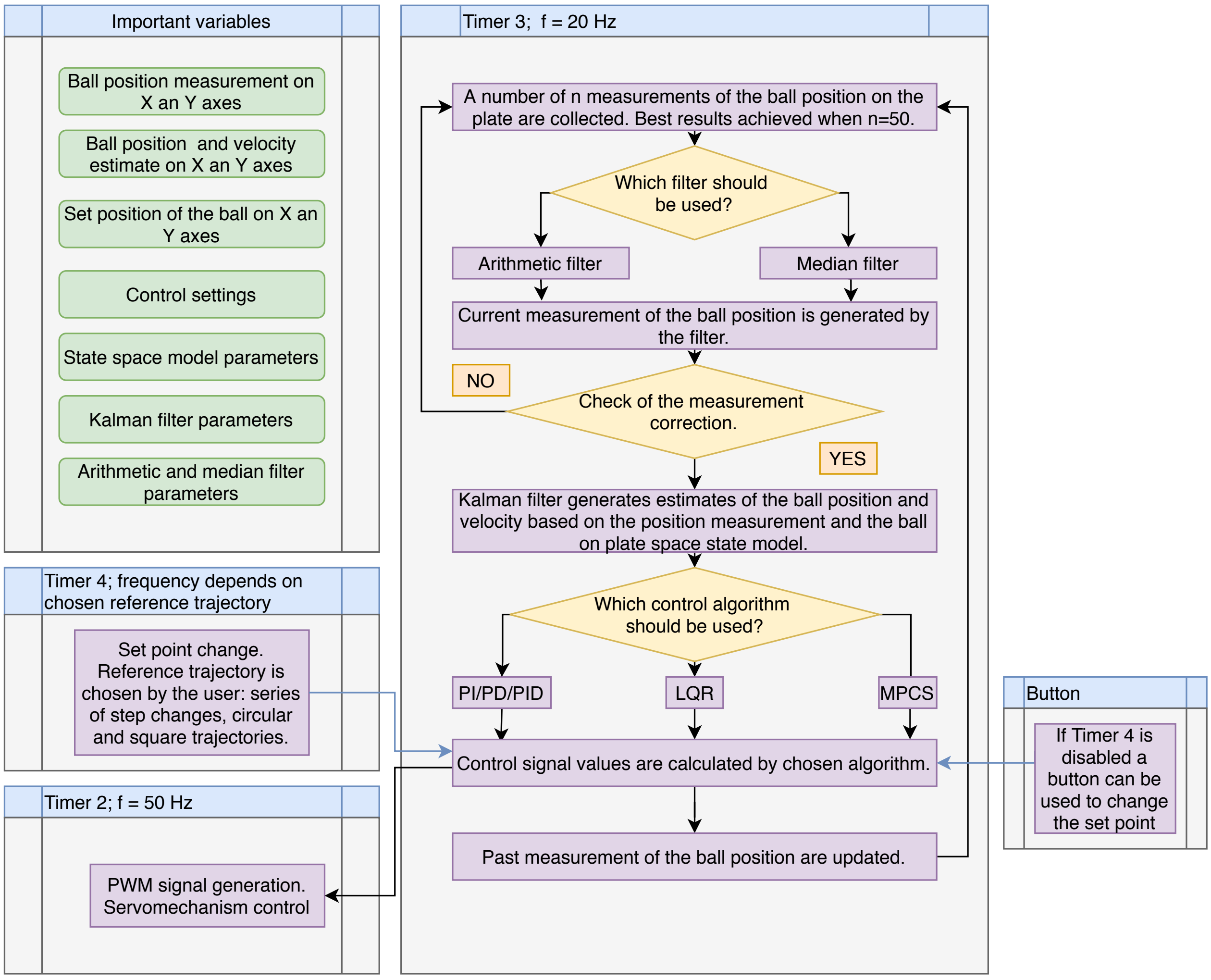

Important variables
Ball position measurement on X an Y axes
Ball position and velocity estimate on X an Y axes
Set position of the ball on X an Y axes
Control settings
State space model parameters
Kalman filter parameters
Arithmetic and median filter parameters
Timer 3; f = 20 Hz
A number of n measurements of the ball position on the plate are collected. Best results achieved when n=50.
Which filter should be used?
Arithmetic filter
Median filter
Current measurement of the ball position is generated by the filter.
NO
Check of the measurement correction.
YES
Kalman filter generates estimates of the ball position and velocity based on the position measurement and the ball on plate space state model.
Which control algorithm should be used?
PI/PD/PID
LQR
MPCS
Control signal values are calculated by chosen algorithm.
Past measurement of the ball position are updated.
Timer 4; frequency depends on chosen reference trajectory
Set point change. Reference trajectory is chosen by the user: series of step changes, circular and square trajectories.
Timer 2; f = 50 Hz
PWM signal generation. Servomechanism control
Button
If Timer 4 is disabled a button can be used to change the set point

Figure 2. The structure of the real-time software system.

Listing 1 shows a fragment of the code in which the values of the manipulated variables for the current sampling instant are calculated (Equation (34)). It is important that all vectors and matrices related to the linear model are constant, they are calculated off-line, i.e., $\boldsymbol{K}_{n_u}$, $\widetilde{\boldsymbol{C}}$, $\widetilde{\boldsymbol{A}}$, $\boldsymbol{V}$ and $\boldsymbol{B}$. Only the set-point vector, $\boldsymbol{y}^{\text{sp}}(k)$, the current state estimation, $x(k)$, the manipulated variables at the previous sampling instant, $u(k-1)$, as well as the output and state disturbance estimations, $d(k)$ and $v(k)$, respectively, are updated on-line. Next, the calculated decision variables are projected onto the admissible set, determined by the constraints. For this purpose the procedure characterized by Equation (36) is used.

Listing 1: Fragment of the code: calculation of the control signal for the MPC controller.

```
//ysp-y0=IyN*ysp(k)-(Ct*At*[x1(k); x2(k)]+Ct*B*V*u(k-1)+
//+Ct*V*v(:,k)+d(k)*IyN)
for(ii=0;ii<80;ii++)
{
Ysp_minus_Y0[ii]=IyN_Yset[ii]-CtAtx[ii]-CtBVu[ii]-CtVv[ii];
}
//du(k)=Kc*(yset(k)-y0(k))
for(ii=0;ii<10;ii++)
{
tmpMPC=0;
for(kk=0;kk<80;kk++)
{
tmpMPC=tmpMPC+Kc[ii][kk]*Ysp_minus_Y0[kk];
}
du[ii]=tmpMPC;
}
//constraints on du
if (du[0]<dumin[0]) du[0]=dumin[0];
if (du[1]<dumin[1]) du[1]=dumin[1];
if (du[0]>dumax[0]) du[0]=dumax[0];
if (du[1]>dumax[1]) du[1]=dumax[1];
// calculate the~control signals for~the~moment k
U[0][0]=U[0][1]+du[0];
U[1][0]=U[1][1]+du[1];
//constraints on u
if (U[0][0]<umin[0]) U[0][0]=umin[0];
if (U[0][0]>umax[0]) U[0][0]=umax[0];
if (U[1][0]<umin[1]) U[1][0]=umin[1];
if (U[1][0]>umax[1]) U[1][0]=umax[1]
// scaling the~signal and sending it to servo
if (U[0][0]*(10.0f)*180.0f/M_PI+1150.0f>1300){
htim2.Instance->CCR3=1300;}
else{
htim2.Instance->CCR3=U[0][0]*(10.0f)*180.0f/M_PI+1150.0f;
}
if (U[1][0]*(10.0f)*180.0f/M_PI+1100.0f>1300){
htim2.Instance->CCR2=1300;
}
else{
htim2.Instance->CCR2=U[1][0]*(10.0f)*180.0f/M_PI+1100.0f;
}
```

5. Results of Experiments

Three controllers have been implemented: MPC, PID and LQR. In all cases, the analytically calculated manipulated variables are projected onto the admissible set determined by the constraints

$$-20^\circ \leq \theta \leq 20^\circ, \tag{38}$$

$$-20^\circ \leq \phi \leq 20^\circ. \tag{39}$$

Tuning of the MPC controller starts with the selection of the prediction horizon. It should be long enough to cover the dynamic behavior of the process. However, if the horizons are too long, the matrices sizes rapidly increase as shown in Table 2, which results in longer computation time and memory requirement. The control horizon cannot be too short since it gives insufficient control quality while its lengthening increases the computational burden. The horizons $N = 40$ and $N_u = 5$ have been found experimentally. MATLAB simulations have been performed and next the horizons have been verified in real process. The process of tuning is the following:

- constant weighting coefficients $\lambda_{p,m} = 1$ are assumed,
- the same prediction horizon N and control horizon N_u are used; if the controller is not working properly, both horizons are lengthened,
- the prediction horizon is gradually shortened, and its length is chosen (with the condition $N_u = N$),
- the effect of the changing the length of the control horizon on the resulting control quality is assessed experimentally (e.g., assume successively $N_u = 1, 2, 3, 4, 5, 10, ..., N$). The shortest possible control horizon is chosen.

Table 2. Size of matrices used in MPC for different prediction and control horizons.

N	N_u	$\widetilde{C}\widetilde{A}$	$\widetilde{C}VB$	V	K_{n_u}
20	3	40 × 4	40 × 2	80 × 4	6 × 40
40	5	80 × 4	80 × 2	160 × 4	10 × 80
80	10	160 × 4	160 × 2	320 × 4	20 × 160

Next, the weighting coefficients must be tuned. Because the objective is precise control in both X and Y axes, $\psi_{p,n} = 1$ for all $p = 1, \ldots, N$, $n = 1, \ldots, n_{\mathrm{y}}$. Hence, the penalties $\lambda_{p,n}$ for all $p = 0, \ldots, N_{\mathrm{u}} - 1$, $n = 1, \ldots, n_{\mathrm{u}}$ must be determined. Let us consider Figure 3 which shows trajectories obtained for the MPC1 algorithm with relatively small values $\lambda_{p,1} = 0.35$ and $\lambda_{p,2} = 1.1$. The process outputs very quickly follow changes of the set-points, but there are some small oscillations of the outputs and the manipulated variables change very rapidly. Let us consider Figure 4 that shows trajectories obtained for the MPC2 algorithm with relatively large values $\lambda_{p,1} = \lambda_{p,2} = 500$. As a result, changes of the manipulated variables are very small, but control accuracy is insufficient, i.e., overshoot is high and the setting time is long. It can be noticed that the ball tends to stop its movement a few millimeters under or over the set-point. The mild changes in control signals are then unable to overcome the static friction force affecting the ball for a few seconds. After that time, the ball moves and stops on the other side of the set-point. This cycle then continues and as a result, the set position is never reached. After tuning, the best results are achieved for $\lambda_{p,1} = 0.8$ and $\lambda_{p,2} = 5$ in the MPC3 algorithm. The resulting process trajectories are depicted in Figure 5. In this case, the setting time is short, there are no oscillations and no overshoot.

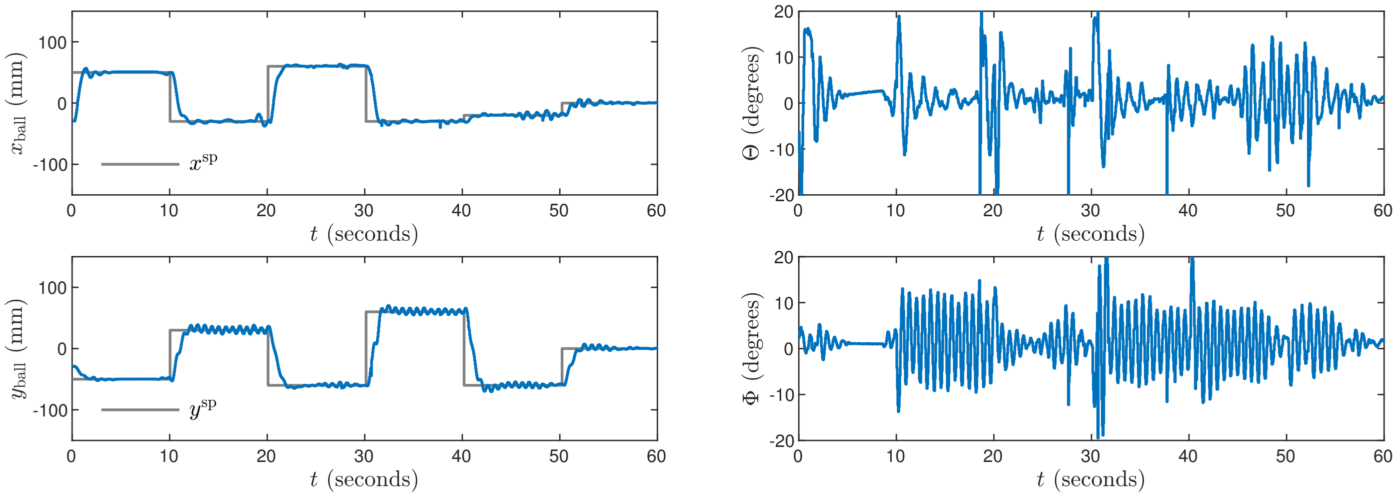


Figure 3. Experimental results of the MPC1 algorithm.

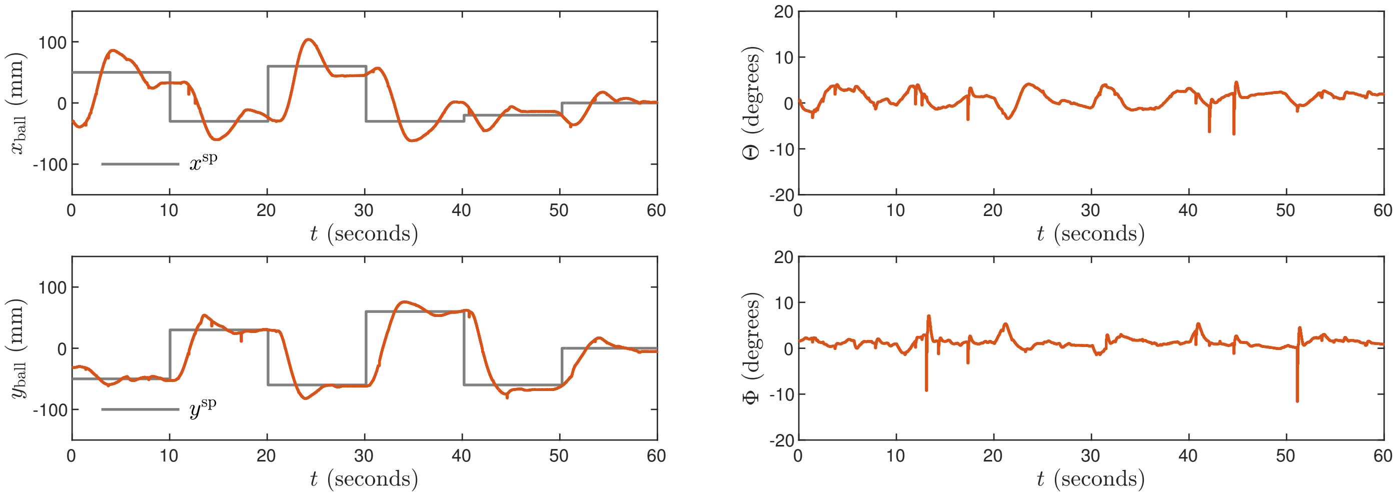


Figure 4. Experimental results of the MPC2 algorithm.

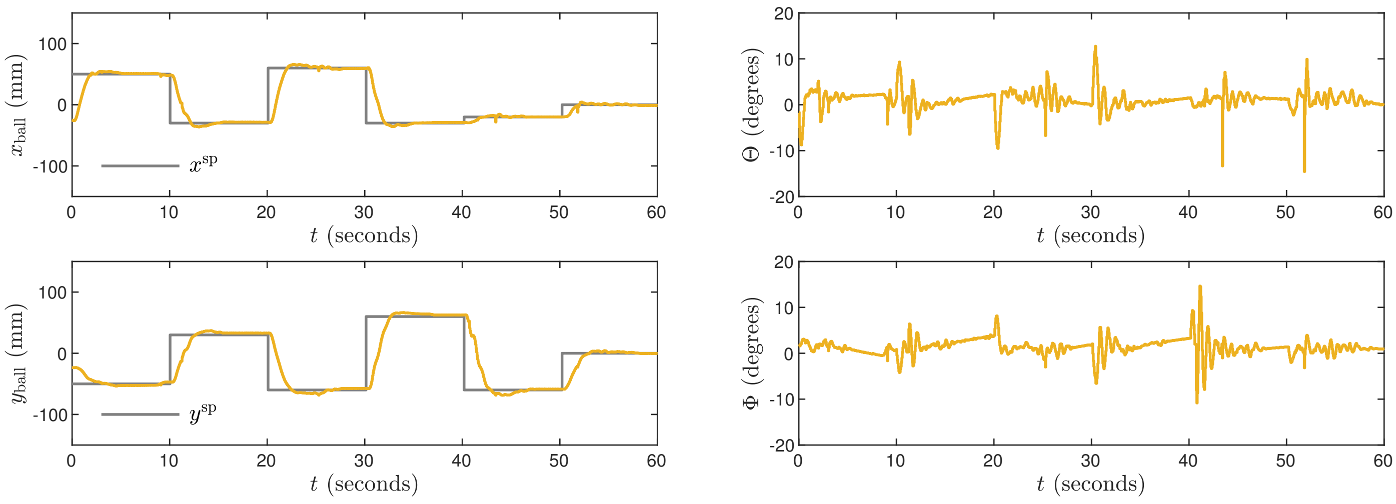
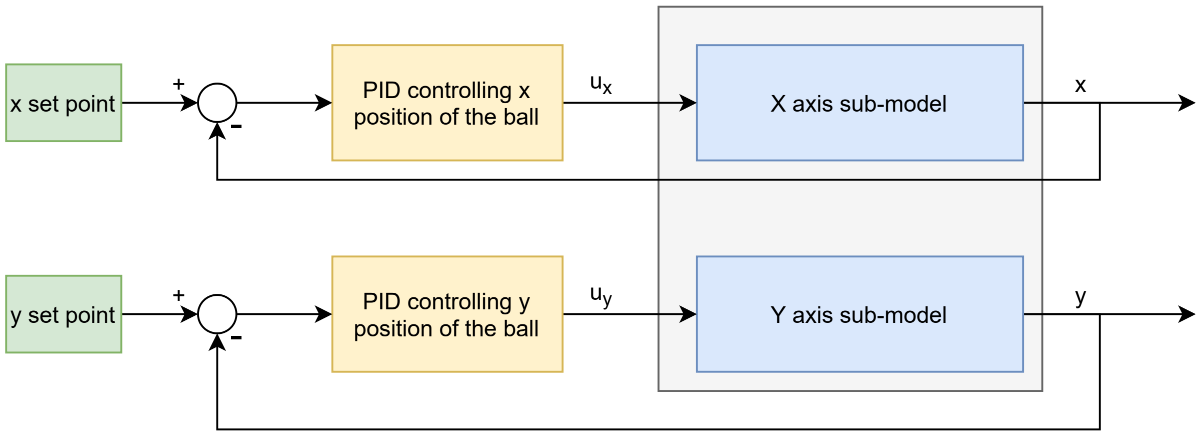


Figure 5. Experimental results of the MPC3 algorithm.

It is an interesting question whether the simple PID controller can give the control quality comparable with that possible when the well-tuned MPC3 scheme is used. The PID control system structure is shown in Figure 6. It consists of two independent classical single-loop PID controllers. The first of them controls the ball's x position by manipulating the PWM signal of the first servo. The second one controls the y position by the PWM signal of the second servo. Two single-loop PID controllers may be used assuming that the interaction between the plate rotation around X axis and ball position y is negligible. The same applies for the rotation around Y axis and ball position x. For each sub-system, an independent PID controller has been tuned using the Ziegler–Nichols method and next readjusted experimentally. Let K, T_{i} and T_{d} denote the proportional gain as well as integration and derivation time constants, respectively.



x set point
+
-
PID controlling x position of the ball
u_x
X axis sub-model
x
y set point
+
-
PID controlling y position of the ball
u_y
Y axis sub-model
y

Figure 6. Diagram of the PID control structure for ball and plate system.

The P controller is unable to stabilize of the system. Regardless of the used parameter K, the ball oscillates endlessly. Results for the PI controller are shown in Figure 7. Its tuning parameters are: $K_{\text{x}} = K_{\text{y}} = 0.06$, $T_{\text{ix}} = T_{\text{iy}} = 5$. It gives strong overshoot and slowly fading huge oscillations. The required set-points are never reached. Results for the PD controller are shown in Figure 8. Its parameters are: $K_{\text{x}} = 0.09$, $K_{\text{y}} = 0.07$, $T_{\text{dx}} = T_{\text{dy}} = 0.6$. In comparison with the PI structure, it makes it possible to obtain smaller oscillations and reduced overshoot, but the steady-state error is inevitable. Finally, the PID controller is considered with the parameters: $K_{\text{x}} = 0.09$, $K_{\text{y}} = 0.08$, $T_{\text{ix}} = T_{\text{iy}} = 1.5$, $T_{\text{dx}} = T_{\text{dy}} = 0.6$. The obtained trajectories are shown in Figure 9. The PID controller makes it possible to eliminate the steady-state error. Unfortunately, let us stress the fact that the PID controller gives much worse control than the MPC3 scheme (Figure 5). In particular, the setting time is much longer, and significant overshoot is present.

PID and PD controllers give very rapidly changing control signals. If the calculations are performed using the ball position measurement, the derivative part causes the whole system to vibrate strongly, even when the ball reaches the set-points. This problem is eliminated using an estimate generated by the Kalman filter instead of the measurements.

The PID controller turns out to be very sensitive to measurement errors. Let us consider Figure 9 and $t = 55$ s. A small measurement error causes the ball to move significantly away from the set-points, in particular for the y_{ball} output. Errors of the same order do not affect the MPC and LQR controllers.

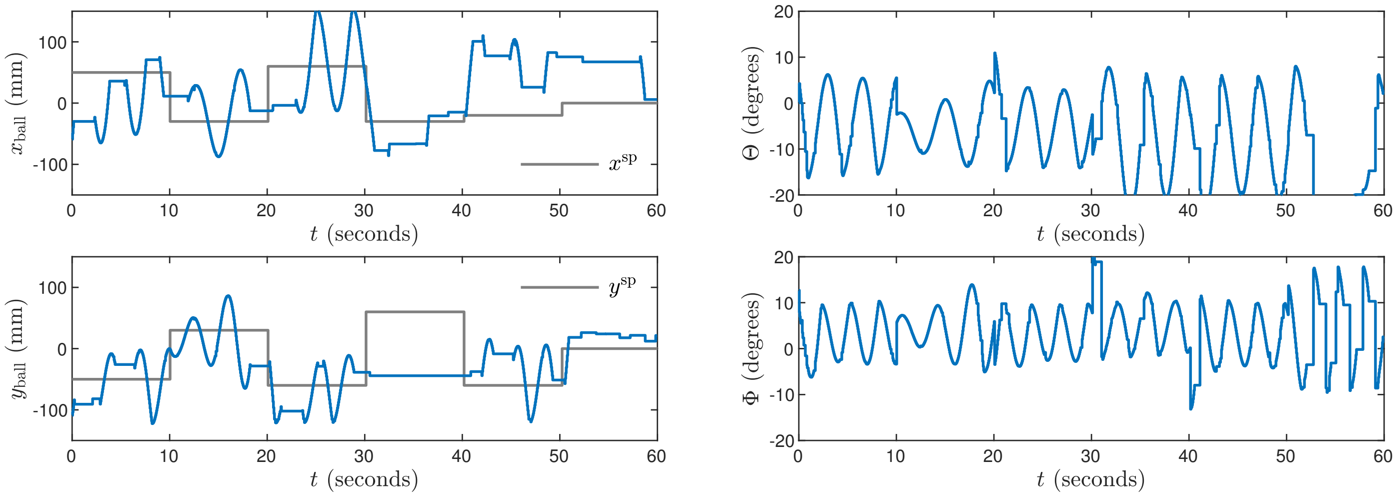


Figure 7. Experimental results of the PI algorithm.

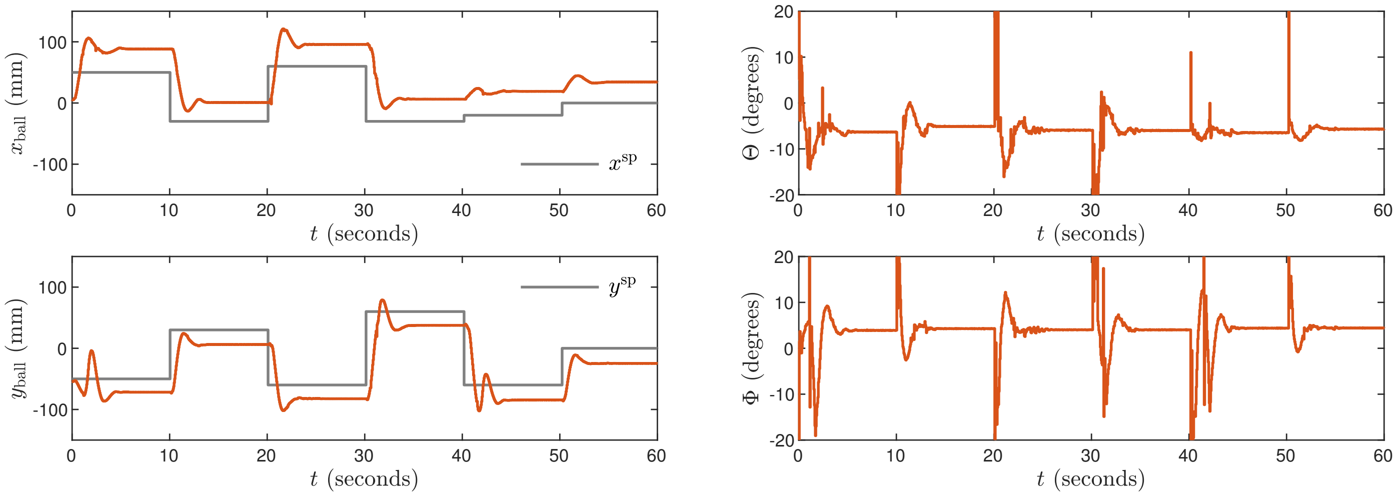


Figure 8. Experimental results of the PD algorithm.

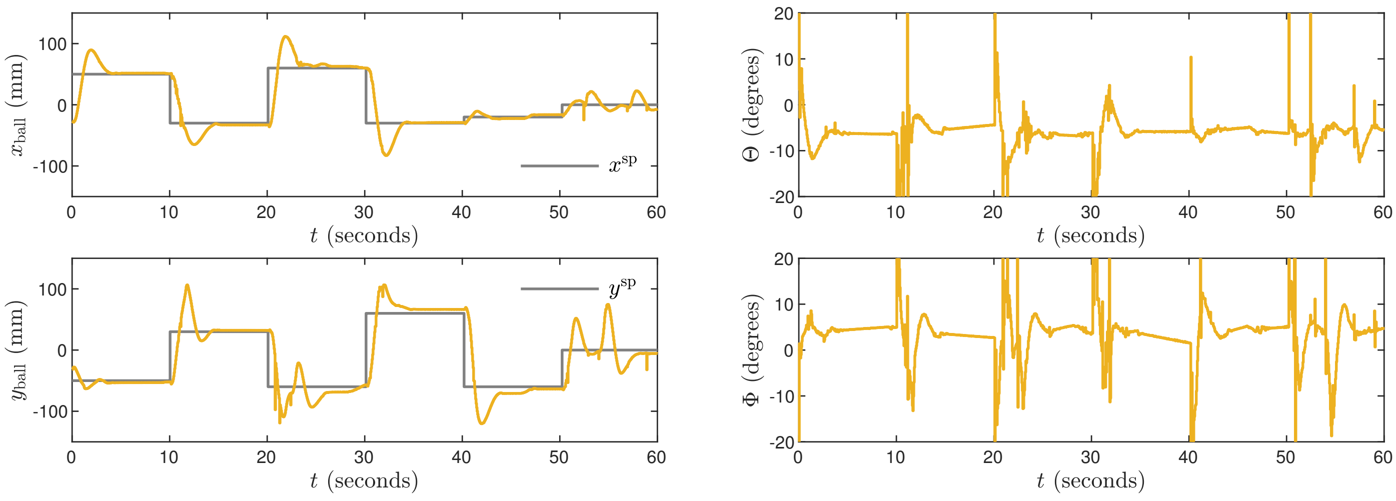


Figure 9. Experimental results of the PID algorithm.

Finally, the LQR controller is evaluated. Its settings are selected in two ways: (a) chosen experimentally using the pole placement method, (b) optimized in MATLAB using the DLGR function. For the poles $0.9404 - 0.0019i$, $0.9404 + 0.0019i$, $0.9404 - 0.0019i$, $0.9404 + 0.0019i$ of the closed-loop system, the controller's (LQR1) gain matrix is

$$\boldsymbol{K}_1 = \begin{bmatrix} 1.3960 & 2.3049 & 0 & 0 \\ 0 & 0 & 1.1422 & 1.8859 \end{bmatrix}. \tag{40}$$

In the first optimization approach (LQR2), the weighting matrices $\boldsymbol{Q} = \text{diag}(7, 0, 7, 0)$, $\boldsymbol{R} = \text{diag}(1, 1)$ give

$$\boldsymbol{K}_2 = \begin{bmatrix} 2.4966 & 2.2137 & 0 & 0 \\ 0 & 0 & 2.4813 & 1.9963 \end{bmatrix}. \tag{41}$$

In the second optimization approach (LQR3), the weighting matrices $\boldsymbol{Q} = \text{diag}(15, 0, 15, 0)$, $\boldsymbol{R} = \text{diag}(1, 1)$ give

$$\boldsymbol{K}_2 = \begin{bmatrix} 3.6104 & 2.6621 & 0 & 0 \\ 0 & 0 & 3.5837 & 2.3991 \end{bmatrix}. \tag{42}$$

The obtained trajectories are given in Figures 10–12 for the controllers LQR1, LQR2 and LQR3, respectively. In general, the manipulated variables change slowly. It is observed especially for the LQR2 and LQR3 controllers. No overshoot in a ball position signal is observed for those settings. However, the setting time is approximately two times longer than for the MPC3 algorithm (Figure 5). The LQR1 controller allows a shortening of that time, but overshoot appears. For the set-point tracking task, the LQR controllers perform much better than P, PD, PI, and PID ones, but still, the results are worse than in the case of the MPC3 scheme.

Let us compare all considered controllers numerically. Table 3 specifies control errors (the sum of squared control errors) [46] for all algorithms. The obtained values confirm observations possible on the basis on the process trajectories, i.e., the MPC3 algorithm is the best one. Table 4 specify the average and maximal setting times for all algorithms. Once again, the MPC3 algorithm gives the best results. Let us stress the fact that in the case of PI, PD, LQR1 and MPC2 controllers, it is impossible to determine the setting times for the assumed set-point changes, i.e., these control algorithms do not result is sufficiently fast control for the used changes of the set-points.

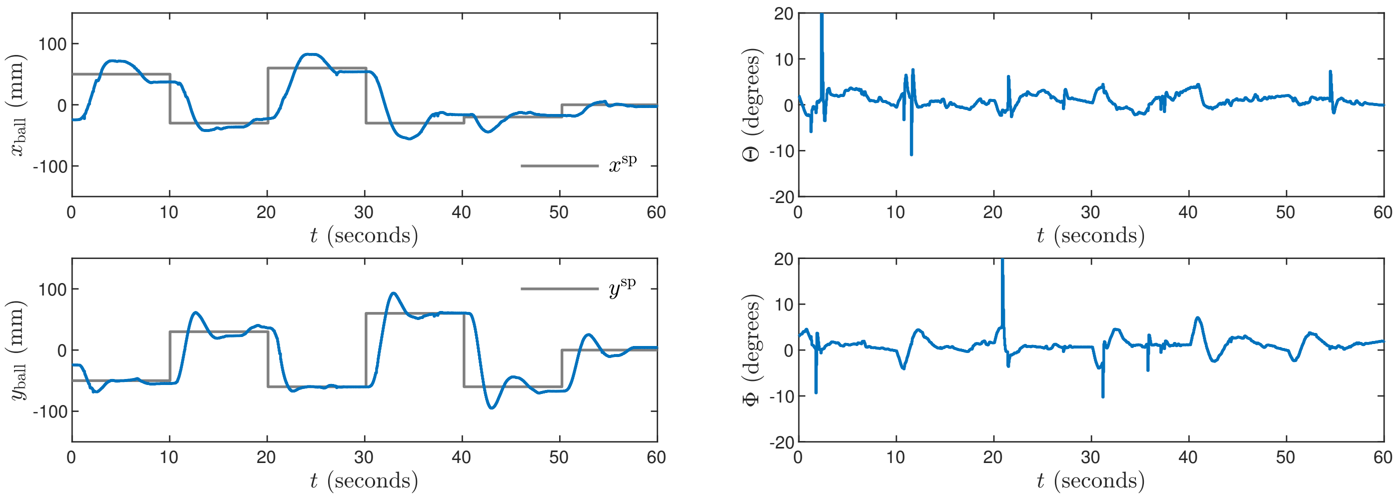


Figure 10. Experimental results of the LQR1 algorithm.

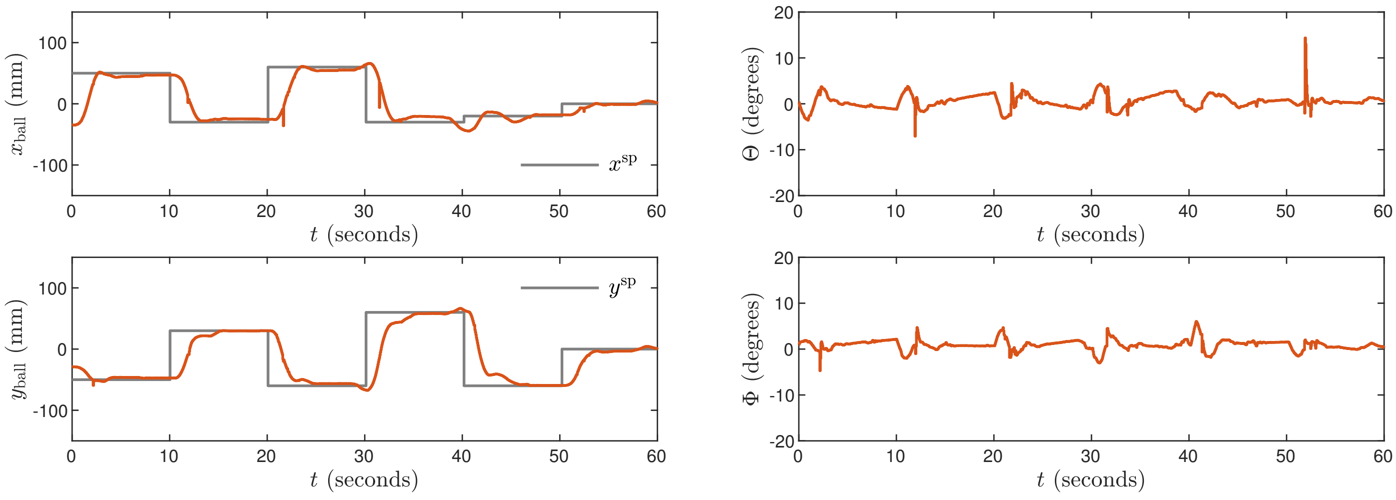


Figure 11. Experimental results of the LQR2 algorithm.

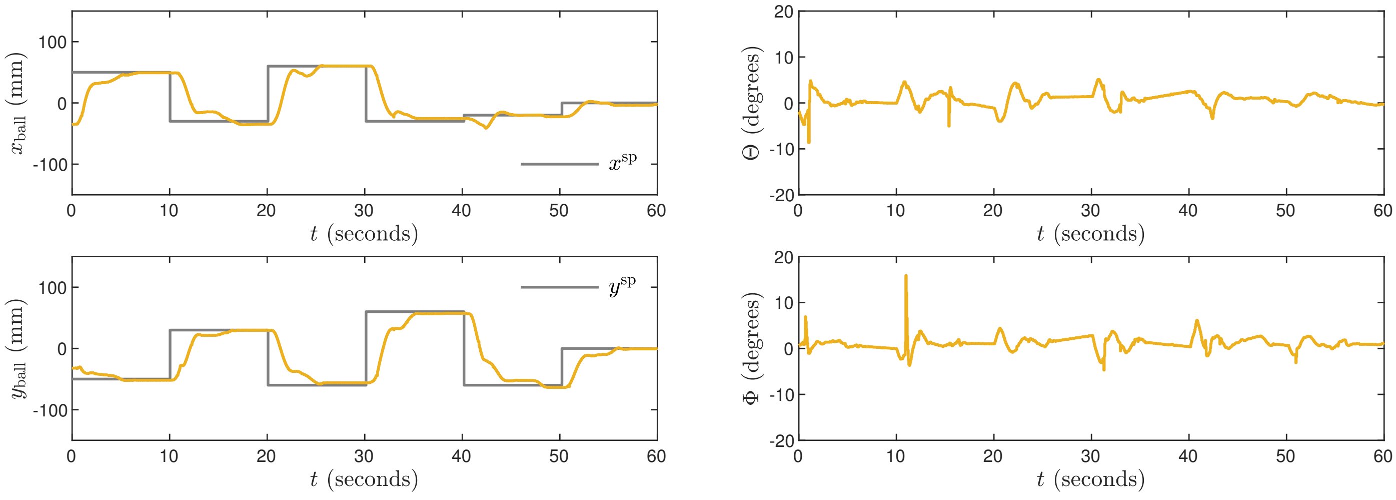


Figure 12. Experimental results of the LQR3 algorithm.

Table 3. Control errors for all compared control algorithms (ISE_x: the error for the X axis, ISE_y: the error for the Y axis).

Controller	ISE_x	ISE_y	$\text{ISE}_\text{x} + \text{ISE}_\text{y}$
PI	7327.9	2759.2	10087.1
PD	1662.7	4420.9	6083.6
PID	483.9	837.9	1321.8
LQR1	692.3	825.6	1517.9
LQR2	684.3	793.7	1478.0
LQR3	676.0	849.9	1525.9
MPC1	881.5	498.0	1379.5
MPC2	1274.5	1378.6	2653.1
MPC3	373.7	760.0	1133.7

Table 4. Setting times for all compared control algorithms (T_x^avg, T_x^max: the average and maximal setting times for the X axis, respectively; T_y^avg, T_y^max: the average and maximal setting times for the Y axis, respectively).

Controller	T_x^avg	T_x^max	T_y^avg	T_y^max
PI	Impossible to determine			
PD	Impossible to determine			
PID	3.75	7	4.5	9
LQR1	Impossible to determine			
LQR2	4	7	5.5	8
LQR3	4.83	6	5.83	8
MPC1	3.17	4	5.17	7
MPC2	Impossible to determine			
MPC3	2.17	3	2	2

The calculation time for the explicit MPC control law (Equation (34)) and the projection procedure (Equation (36)) is 4.86 ms. Although it is longer than the time required by PID and LQR controllers, 0.29 ms and 0.27 ms, respectively, but it is important to point out that the obtained value is very short when compared to the sampling time used (50 ms).

6. Conclusions

This work discusses an application of the fast state–space MPC algorithm to the ball-on-plate laboratory process. Moreover, software and hardware implementation details are discussed; the STM32 microcontroller is used for calculations. The considered MPC algorithm is characterized by a very short calculation time (in comparison to the sampling period). This is because the manipulated variables are found on-line using explicit formulas with parameters calculated off-line, no real-time optimization is necessary. The simplicity of calculations is possible because in MPC an approximate state–space linear model is used, nonlinear terms are neglected. Secondly, the constraints are not taken into account during calculation of the manipulated variables, but afterwards, i.e., the unconstrained optimal solution is projected onto the admissible set determined by the constraints. When applied to the ball-on-plate process, the considered MPC algorithm works much better than the classical PID and LQR controllers. It is necessary to stress that all three controllers compared in this work are linear. The MPC and LQR controllers are multivariable, i.e., the manipulated variables are calculated taking into account all interactions of process variables and the same linear model is used for development of the control law. Conversely, the PID control scheme is comprised of two classical single-loop controllers.

The discussed MPC algorithm uses for prediction a simplified linear model of the process. In future, it is planned to develop nonlinear MPC approaches [47,48] and evaluate their performance.

Author Contributions: Conceptualization, K.Z. and M.Ł.; methodology, K.Z.; software, K.Z.; validation, K.Z. and M.Ł.; formal analysis, K.Z.; investigation, K.Z.; writing–original draft preparation, M.Ł.; writing–review and editing, K.Z. and M.Ł.; visualization, K.Z. and M.Ł.; supervision, M.Ł. 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: The data presented in this study are available on request from the corresponding author.

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

References

1. Bay, C.J.; Rasmussen, B.P. Exploring controls education: A re-configurable ball and plate platform kit. In Proceedings of the 2016 American Control Conference (ACC), Boston, MA, USA, 6–8 July 2016; pp. 6652–6657.
2. Fabregas, E.; Dormido-Canto, S.; Dormido, S. Virtual and Remote Laboratory with the Ball and Plate System. *IFAC-PapersOnLine* **2017**, *50*, 9132–9137. [CrossRef]
3. Stander, D.; Jiménez-Leudo, S.; Quijano, N. Low-Cost "ball and Plate" design and implementation for learning control systems. In Proceedings of the 2017 IEEE 3rd Colombian Conference on Automatic Control (CCAC), Cartagena, Colombia, 18–20 October 2017; pp. 1–6.
4. Dušek, F.; Honc, D.; Sharma, K.R. Modelling of ball and plate system based on first principle model and optimal control. In Proceedings of the 2017 21st International Conference on Process Control (PC), Strbske Pleso, Slovakia, 6–9 June 2017; pp. 216–221.
5. Kassem, A.; Haddad, H.; Albitar, C. Commparison Between Different Methods of Control of Ball and Plate System with 6DOF Stewart Platform. *IFAC-PapersOnLine* **2015**, *48*, 47–52. [CrossRef]
6. Spacek, L.; Bobal, V.; Vojtesek, J. Digital control of Ball & Plate model using LQ controller. In Proceedings of the 2017 21st International Conference on Process Control (PC), Strbske Pleso, Slovakia, 6–9 June 2017; pp. 36–41.
7. Bang, H.; Lee, Y.S. Implementation of a Ball and Plate Control System Using Sliding Mode Control. *IEEE Access* **2018**, *6*, 32401–32408. [CrossRef]
8. Jeon, J.; Hyun, C. Adaptive sliding mode control of ball and plate systems for its practical application. In Proceedings of the 2017 2nd International Conference on Control and Robotics Engineering (ICCRE), Bangkok, Thailand, 1–3 April 2017; pp. 119–123.
9. Morales, L.; Gordón, M.; Camacho, O.; Rosales, A.; Pozo, D. A Comparative Analysis among Different Controllers Applied to the Experimental Ball and Plate System. In Proceedings of the 2017 International Conference on Information Systems and Computer Science (INCISCOS), Quito, Ecuador, 23–25 November 2017; pp. 108–114.
10. Robayo Betancourt, F.I.; Brand Alarcon, S.M.; Aristizabal Velasquez, L.F. Fuzzy and PID controllers applied to ball and plate system. In Proceedings of the 2019 IEEE 4th Colombian Conference on Automatic Control (CCAC), Medellin, Colombia, 15–18 October 2019; pp. 1–6.
11. Moreno-Armendáriz, M.A.; Pérez-Olvera, C.A.; Rodríguez, F.O. Indirect hierarchical FCMAC control for the ball and plate system. *Neurocomputing* **2010**, *73*, 2454–2463. [CrossRef]
12. Huang, W.; Zhao, Y.; Ye, Y.; Xie, W. State Feedback Control for Stabilization of the Ball and Plate System. In Proceedings of the 2019 Chinese Control Conference (CCC), Guangzhou, China, 27–30 July 2019; pp. 687–690.
13. Wang, Y.; Sun, M.; Wang, Z.; Liu, Z.; Chen, Z. A novel disturbance-observer based friction compensation scheme for ball and plate system. *ISA Trans.* **2014**, *53*, 671–678. [CrossRef] [PubMed]
14. Tatjewski, P. Disturbance modeling and state estimation for offset-free predictive control with state-space models. *Int. J. Appl. Math. Comput. Sci.* **2014**, *24*, 313–323. [CrossRef]
15. Nebeluk, R.; Marusak, P. Efficient MPC algorithms with variable trajectories of parameters weighting predicted control errors. *Arch. Control Sci.* **2020**, *30*, 325–363.
16. Huyck, B.; De Brabanter, J.; De Moor, B.; Van Impe, J.F.; Logist, F. Online model predictive control of industrial processes using low level control hardware: A pilot-scale distillation column case study. *Control Eng. Pract.* **2014**, *28*, 34–48. [CrossRef]
17. Pour, F.K.; Puig, V.; Ocampo-Martinez, C. Multi-layer health-aware economic predictive control of a pasteurization pilot plant. *Int. J. Appl. Math. Comput. Sci.* **2018**, *28*, 97–110. [CrossRef]
18. Wang, B.; Shahzad, M.; Zhu, X.; Rehman, K.U.; Uddin, S. A Non-linear Model Predictive Control Based on Grey-Wolf Optimization Using Least-Square Support Vector Machine for Product Concentration Control in l-Lysine Fermentation. *Sensors* **2020**, *20*, 3335. [CrossRef] [PubMed]
19. Carli, R.; Cavone, G.; Ben Othman, S.; Dotoli, M. IoT Based Architecture for Model Predictive Control of HVAC Systems in Smart Buildings. *Sensors* **2020**, *20*, 781. [CrossRef] [PubMed]

20. Rybus, T.; Seweryn, K.; Sąsiadek, J.Z. Application of predictive control for manipulator mounted on a satellite. *Arch. Control Sci.* **2018**, *28*, 105–118.
21. Ogonowski, S.; Bismor, D.; Ogonowski, Z. Control of complex dynamic nonlinear loading process for electromagnetic mill. *Arch. Control Sci.* **2020**, *30*, 471–500.
22. Horla, D. Experimental Results on Actuator/Sensor Failures in Adaptive GPC Position Control. *Actuators* **2021**, *10*, 43. [CrossRef]
23. Eskandarpour, A.; Sharf, I. A constrained error-based MPC for path following of quadrotor with stability analysis. *Nonlinear Dyn.* **2020**, *98*, 899–918. [CrossRef]
24. Ducajú, S.; Salt Llobregat, J.J.; Cuenca, Á.; Tomizuka, M. Autonomous Ground Vehicle Lane-Keeping LPV Model-Based Control: Dual-Rate State Estimation and Comparison of Different Real-Time Control Strategies. *Sensors* **2021**, *21*, 1531. [CrossRef] [PubMed]
25. Bassolillo, S.R.; D'Amato, E.; Notaro, I.; Blasi, L.; Mattei, M. Decentralized Mesh-Based Model Predictive Control for Swarms of UAVs. *Sensors* **2020**, *20*, 4324. [CrossRef]
26. Bania, P. An information based approach to stochastic control problems. *Int. J. Appl. Math. Comput. Sci.* **2020**, *30*, 47–59.
27. Fan, J.; Han, M. Nonliear model predictive control of ball-plate system based on gaussian particle swarm optimization. In Proceedings of the 2012 IEEE Congress on Evolutionary Computation, Brisbane, QLD, Australia, 10–15 June 2012; pp. 1–6.
28. 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]
29. Oravec, M.; Jadlovská, A. Model Predictive Control of a Ball and Plate laboratory model. In Proceedings of the 2015 IEEE 13th International Symposium on Applied Machine Intelligence and Informatics (SAMI), Herl'any, Slovakia, 22–24 January 2015; pp. 165–170.
30. Houska, B.; Ferreau, H.J.; Diehl, M. An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. *Automatica* **2011**, *47*, 2279–2285. [CrossRef]
31. Wang, Y.; Boyd, S. Fast model predictive control using online optimization. *IEEE Trans. Control Syst. Technol.* **2010**, *18*, 267–278. [CrossRef]
32. Bemporad, A.; Morari, M.; Dua, V.; Pistikopoulos, E.N. The explicit linear quadratic regulator for constrained systems. *Automatica* **2002**, *38*, 3–20. [CrossRef]
33. Valencia-Palomo, G.; Rossiter, J.A. AEfficient suboptimal parametric solutions to predictive control for PLC applications. *Control Eng. Pract.* **2011**, *19*, 732–743. [CrossRef]
34. Rauová, I.; Valo, R.; Kvasnica, M.; Fikar, M. Real-Time Model Predictive Control of a Fan Heater via PLC. In Proceedings of the 18th International Conference on Process Control, Slovak University of Technology in Bratislava, Tatranská Lomnica, Slovakia, 14–17 June 2011; pp. 388–393.
35. Liu, S.; Wang, J. A simplified dual neural network for quadratic programming with its KWTA application. *IEEE Trans. Neural Netw.* **2006**, *17*, 1500–1510. [PubMed]
36. Tatjewski, P. *Advanced Control of Industrial Processes, Structures and Algorithms*; Springer: London, UK, 2007.
37. Chaber, P.; Ławryńczuk, M. Fast Analytical Model Predictive Controllers and Their Implementation for STM32 ARM Microcontroller. *IEEE Trans. Ind. Inf.* **2019**, *15*, 4580–4590. [CrossRef]
38. Valencia-Palomo, G.; Rossiter, J.A. Programmable logic controller implementation of an auto-tuned predictive control based on minimal plant information. *ISA Trans.* **2011**, *50*, 92–100. [CrossRef]
39. Tatjewski, P.; Ławryńczuk, M. Algorithms with state estimation in linear and nonlinear model predictive control. *Comput. Chem. Eng.* **2020**, *143*, 107065. [CrossRef]
40. Zarzycki, K.; Ławryńczuk, M. Development and modelling of a laboratory ball on plate process. In *Advanced, Contemporary Control*; Bartoszewicz, A., Kabziński, J., Kacprzyk, J., Eds.; Advances in Intelligent Systems and Computing; Springer: Cham, Switzerland, 2020; Volume 1196, pp. 396–408.
41. Maeder, U.; Morari, M. Offset-free reference tracking with model predictive control. *Automatica* **2010**, *46*, 1469–1476. [CrossRef]
42. Muske, K.; Badgwell, T. Disturbance modeling for offset-free linear model predictive control. *J. Process Control* **2002**, *12*, 617–632. [CrossRef]
43. Pannocchia, G.; Rawlings, J. Disturbance models for offset-free model predictive control. *AIChE J.* **2003**, *49*, 426–437. [CrossRef]
44. HS-5485HB Standard Karbonite Digital Sport Servo. Available online: https://hitecrcd.com/products/servos/sport-servos/digital-sport-servos/hs-5485hb-standard-karbonite-digital-servo/product (accessed on 8 June 2021).
45. 15" 4-Wire Resistive Screen. Available online: http://www.greentouch.com.tw/product/22-inch-four-wire-resistive-screen.html (accessed on 8 June 2021).
46. Domański, P. *Control Performance Assessment: Theoretical Analyses and Industrial Practice*; Studies in Systems, Decision and Control; Springer: Cham, Switzerland, 2020; Volume 245.
47. Ławryńczuk, M. *Computationally Efficient Model Predictive Control Algorithms: A Neural Network Approach*; Studies in Systems, Decision and Control; Springer: Cham, Switzerland, 2014; Volume 3.
48. Marusak, P.M. A numerically efficient fuzzy MPC algorithm with fast generation of the control signal. *Int. J. Appl. Math. Comput. Sci.* **2021**, *31*, 59–71.

Short Biography of Authors

Krzysztof Zarzycki was born in Pruszków, Poland, in 1993. He received his B.Sc. degree in 2017 at the Faculty of Mechatronics, Warsaw University of Technology, and his M.Sc. degree in 2020 at the Faculty of Electronics and Information Technology, the same university, both in automatic control and robotics. He has been with the Institute of Control and Computation Engineering, Warsaw University of Technology, since 2020, where he is currently employed as an assistant. His scientific interests include: algorithms for industrial process control, mainly advanced Model Predictive Control (MPC), and applications of artificial intelligence as a tool in process control and modelling.

Maciej Ławryńczuk was born in Warsaw, Poland, in 1972. He obtained his M.Sc. in 1998, Ph.D. in 2003, D.Sc. in 2013, all in automatic control, from Warsaw University of Technology, Faculty of Electronics and Information Technology. Currently, he is employed at the same university at the Institute of Control and Computation Engineering as an associate professor. He is the author or co-author of six books and more than 100 other publications, including over 40 journal articles. His research interests include advanced control algorithms, in particular, Model Predictive Control (MPC) algorithms, set-point optimisation algorithms, soft computing methods, in particular, neural networks, modelling, and simulation.