IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 16, 2021, accepted January 2, 2022, date of publication January 11, 2022, date of current version January 28, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3142054

Advanced Control With PLC–Code Generator for aMPC Controller Implementation and Cooperation With External Computational Server for Dealing With Multidimensionality, Constraints and LMI Based Robustness

JAROSLAW TARNAWSKI, PIOTR KUDEŁKA, AND MATEUSZ KORZENIOWSKI
Faculty of Electrical and Control Engineering, Gdansk University of Technology, 80-233 Gdansk, Poland
Corresponding author: Jaroslaw Tarnawski (jaroslaw.tarnawski@pg.edu.pl)

ABSTRACT The manufacturers of Programmable Logic Controllers (PLC) usually equip their products with extremely simple control algorithms, such as PID and on-off regulators. However, modern PLCs have much more efficient processors and extensive memory, which enables implementing more sophisticated controllers. The paper discusses issues related to the implementation of matrix operations, time limitations for code execution within one PLC cycle, and memory requirements. The adaptive Model Predictive Controller (aMPC) algorithm is selected for predictive control with on-line adaptation of model parameters. The combination of predictive and adaptive properties in the regulator enables control of many industrial objects for which PID control is ineffective, e.g. nonstationary plants with time-varying delays. The presented generic approach consists in developing a C++ application for desktop PC that generates, based on user provided parameters - such as MPC horizons, the code in Structured Text (ST) language compliant with the IEC-61131 standard for PLCs. Despite the enhanced capabilities of programmable controllers, there are limitations to this platform that cannot be overcome. The implementation of optimization-based control algorithms requires cooperation with an external computing server. In the article, the PLC/external computer cooperation is used to implement the control with constraints taken into account. The robust control using the Linear Matrix Inequalities (LMI) for a multivariable plant is also presented. A number of tests were carried out to verify the correctness of implementation of this control in software-in-the-loop and hardware-in-the-loop structures.

INDEX TERMS Control engineering, control system synthesis, programmable control, programmable logic devices, predictive control, adaptive control, optimal control, application software, system simulations, system verification.

I. INTRODUCTION

Programmable Logic Controllers are a popular industrial platform for the implementation of control algorithms. Due to their numerous advantages, such as high reliability, preparation for operation in industrial conditions, wide range of I/O modules providing the possibility of connecting any control object, large communication possibilities ensuring integration with SCADA systems and other elements of the control system, high scalability, and easy programming, it is difficult to find better equipment to act as a direct control layer. To apply advanced control (AC) methods with a PLC, we have two main approaches: implement a complex algorithm in the PLC, or use an external computing machine. Both these approaches are discussed in the article.

In order to implement advanced control algorithms on a given platform, its possibilities and limitations should be analyzed beforehand. A detailed analysis of CPU performance and memory limitations of selected PLCs is presented. Algorithms described by difference equations written in the

The associate editor coordinating the review of this manuscript and approving it for publication was Jesus Felez.

form of relations between values in the present and past times are especially predestined for implementation in a PLC. Due to the loop-based manner of PLC operation, it is relatively easy to write a program that realizes dynamic with input to output or input and state to output relations. This opens the space for the implementation of discrete control algorithms, estimation, filtration, etc. A separate chapter presents the implementation of the algorithms described by difference equations in a loop environment such as PLC.

Natural limitations of the PLC platform refer to much lower computing power and memory area compared to PC computers. Also, the real-time operating system enforces a finite computation time within one PLC work cycle. Moreover, programming libraries that are available for PCs are not available on the PLC. Most of the advanced control algorithms use matrix notation. PLCs offer matrix data storage but do not have native arithmetic operations on matrices. In the article, attention is paid to the implementation of basic mathematical operations using matrices such as addition, multiplication and inversion in terms of their memory requirements, execution time and accuracy. In addition to the PLC limits, it is not possible to create dynamic matrix sizes on the PLC platform, which makes it difficult to implement algorithms in which the matrix sizes depend on parameters set by the user. Hence the idea that the program for the PLC should be prepared by a specially created generator for a standard PC. The assumption is that the user enters the controller parameters, and then the code generator working on PC prepares the program in the form of a text file in ST language for the PLC. This approach is very flexible, it additionally provides the information on the amount of memory used and the time needed to execute such a code in the PLC. All this allows the user to define the scope of applicability of a given code in relation to a specific PLC CPU.

There are situations where the computation time necessary to complete the entire control algorithm exceeds the allowable cycle time. For these situations, an approach is presented in which the computations are shared between successive PLC cycles. This obviously increases the step at which the controller can work to the sum of times of all cycles taking part in the calculation of all steps of the algorithm.

However, not everything can be calculated directly within the PLC. When we need to use external libraries or an optimizer, or simply when the dimensions of the matrix or the computation time exceed the capabilities of the PLC, the only solution is to use PLC cooperation with an external computing server.

Advanced adaptive-predictive control algorithms were selected for implementation. These are MPC and DMC algorithms with on-line updating of information about the object model. These algorithms use different forms of the model, but are functionally similar to each other. They preserve all advantages of predictive control related to: extension of the horizon, entering the parameters as trajectories, taking into account multidimensionality, the ability to control non-minimal phase objects, and natural handling of delays, including time-varying delays. In addition, due to the adaptive part, they have the ability to control objects in which the occurring parameter changes are sufficiently slow to be captured by the estimation mechanism. To present the potential of the structure employing an external computing server, control algorithms were prepared taking into account constraints, such as QDMC and LMI-based robust control. Non-stationary SISO and MIMO objects were used for verification. The correctness of implementation of the prepared algorithms was verified by simulation only within the PLC, or with the acquisition card and the control object simulated outside.

The article is a significant extension of the work presented in publications [1] and [2]. A large part of the present achievements comes from master's theses [3] and [4].

II. STATE OF THE ART

This section presents, based on literature and Internet references, the most important aspects of the article, including: what we call advanced control, what industrial equipment enables the implementation of advanced control algorithms, and which algorithms are implemented in their devices by leading manufacturers. The next section will give brief descriptions of the IEC-61131 standard defining programming languages for PLC and of the PLCopen organization dealing with the maintenance and development of the PLC programming standard and its close relationship with the issues of data exchange in the automation system using the OPC standard. Another important element directly related to the content of the article is the PLC Coder tool as part of the Matlab package. Finally, a few literature examples of advanced control algorithms implemented in PLCs in various industrial applications are presented.

A. DEFINITIONS OF ADVANCED CONTROL

Since there is no strict definition of the term ''advanced control'', an attempt was made to find its clarification in the industry literature. In [5], we can read ''Classical control techniques are mainly represented by the proportional-integral-derivative (PID) controller which has been employed in a broad range of applications. PID controllers are applied mostly to the processes which are characterized by linear, low-order dynamics. Advanced control techniques might be classified into the following three main categories: model-based control, fuzzy logic control, and artificial neural network-based control''.

In [6], it is written ''Typically, advanced control methods involve more complex calculations than the conventional PID controller algorithm. Advanced control has the following features: Process modelling and parameter identification (off line or on line), Prediction of process behavior using process model, Evaluation of performance criterion, subject to process constraints, Optimisation of performance criterion, Matrix calculations (multivariable control) and feedback control''.

A different approach is proposed in [7]. Instead of simple division into simple and advanced control algorithms, a multi-layer (direct, constraint/set-point, optimization, management) control structure is proposed, with three goals identified as: safety of the control system, keeping product quality, and, at the third place, consideration of economic issues. The least complicated layer located next to the object is the direct layer. In [7] we can read ''Algorithms of direct control should be safe, robust and relatively easy, that is why classic PID algorithms are still dominant''. The author names algorithms that can be applied in the direct layer derived from or cooperating with PID, but also algorithms of unconstrained adaptive and predictive control and concludes: ''However, it should be strongly emphasized that the generic feature distinguishing all direct control algorithms is the direct access to the controlled process (the process manipulated inputs are outputs of the direct (basic) controllers) and high frequency of intervention (small sampling period) – not the kind of control algorithm employed''. Later in the book, the algorithms of model based fuzzy and predictive control and set-point optimization are presented as advanced. A similar approach related to ensuring security in the direct/basic layer is presented in [8]. The advanced algorithms listed in this book include PID tuning methods, linear estimation issues, artificial neural networks, and model-based predictive control.

The latest and very important trend in industrial control systems is ensuring security against network attacks with the application of sophisticated machine learning technology [9], [10].

B. INDUSTRIAL PLATFORMS FOR ADVANCED CONTROL IMPLEMENTATION

There are two main options of advanced control system implementation with industrial hardware and software platform: the DCS system, and the PLC + SCADA system. In chapter IX.B less popular alternatives are presented.

Leading DCS manufacturers, such as Emmerson, Yokogawa, Honeywell, Rockwell, ABB and Siemens, have equipped their products with advanced control algorithms. Let us see what DCS manufacturers consider advanced control and what advanced algorithms are available in their products.

Emerson [11] introduces its advanced control section as follows ''We recognize that control systems are more than just a collection of PID-algorithm control loops. Advanced control can help your plant optimize its performance by increasing throughput, increasing conversion, increasing on spec product, and reducing by-product. Some of the other plant improvements to maximize profit include: Helping site management and engineering focus on optimizing process operations 100% of the time, Determine economic optimum, Determine optimal setpoints, Process issues such as complex multi-input, multi-output relationships, operating constraints, nonlinear dynamics, and frequently changing targets.''

Emerson tools and solutions in AC are:

- Aspen DMC3 - enables rapid deployment and sustains optimal performance with patented adaptive process control technology that enables simultaneous process optimization and testing.
- DeltaV EnTech Toolkit – offers Advanced analysis and tuning technology to improve the performance for difficult control loops, along with easy plant test data collection and model identification, and Optimal tuning to improve the performance. Advanced performance provides insight into the characteristics of loop variability and interactions, and possible root causes of poor performance.
- SmartProcess Optimization Software – optimizes control processes using turnkey, off-the-shelf advanced control and monitoring applications from Emerson
- Model-Predictive / Multivariable Control (MVC)
- DeltaV Predict/Pro - enables the user to obtain greater throughput, reduced variability, and increased safety for his/her model predictive control strategies. DeltaV Predict/Pro is fully embedded in the DeltaV system which makes it easy to configure, validate, test and deploy.

State-Based Control - automated procedures for safer and faster startup.

On Yokogawa's website pages and brochures [12] it can be read: ''Advanced Process Control brings benefits by reducing energy consumption and improving the yields. The platform for Advanced Control and Estimation brings multi-variable control, quality estimation, complex calculation, user interface definition all in one application dramatically reducing deployment time and simplifying maintenance for robust performance. Our all-inclusive Platform for Advanced Control and Estimation brings multi-variable control, quality estimation, complex custom calculations, and operator user interface design all into one application; in doing so, dramatically reducing deployment time and simplifying maintenance for robust performance.''

Yokogawa tools and solutions are: Stable Base-Layer Control with Advanced Regulatory Controls, Integrated Dynamic Modeling Tools and Automatic Step-Tests, Holistic Advanced Process Control Application, and Advanced Monitoring & Diagnostics Tool.

Multivariable Control & Quality Estimation is generally divided into Design Time and Run Time. Design Time provides a single workspace for: process data management, process dynamics modeling, processor and sequence design, and scenario-based simulation - all of which are based on best practices in Advanced Process Control deployment. Seamless sharing of information between Design Time and Run Time empowers control engineers to develop complex applications rapidly.

Design Time tools are: Preliminary Controller Design, Robust Model Identification, and Complex Application Development, while Run Time tasks are: Auto Step Testing, New Human-Machine Interface with Key Performance Indicator Faceplate, and Online Tuning.

Honeywell [13] claims that ''Advanced Process Control (APC) products address all aspects of advanced process

control and optimization from improving regulatory loop control to globally optimizing the entire process using a unique layered approach. This model allows new technologies to be easily added at any time to a common platform that meets optimization objectives without compromising on future opportunities to improve business performance. Delivered through the unified Experion Process Knowledge System (PKS) architecture, Honeywell's APC products improve profitability by increasing throughput, reducing costs, increasing yields and improving product quality.

Solutions tackle complex problems through a unique outcome-based consulting approach that supports better process design, process history and analytics, operations excellence, production management and enterprise collaboration. These solutions allow users to make faster and smarter decisions to improve safety, reliability, efficiency and sustainability''.

Some selected Honeywell modules of advanced software are listed below.

Honeywell Forge for Industrial enables Enterprise Performance Management by using real-time accurate models and visual analytics to deliver intelligent actionable recommendations for sustained peak performance.

Profit – Software for Advanced Control, Optimization and Monitoring. From blending, movements and advanced process control to plant-wide optimization and condition-based monitoring, Profit Technologies enable facilities to achieve and maintain excellent operations.

Software for Oil and Gas Production. Honeywell's software for oil and gas users includes Production Control Center, which automates time consuming tasks and makes information available across the organization, and Well Performance Monitor, which enables users to proactively assess the performance of oil and gas production fields using real-time key performance indicators (KPI).

Software for Operations Excellence. From the control room to off-sites material movement, Operations Excellence runs the gamut to eliminate events caused by sub-optimal control performance, increase production and energy efficiency, enable accurate operations monitoring, and so much more.

UniSim - Software for Process Design and Simulation. Honeywell's UniSim software family substantially improves simulation of online and off-line process unit design and optimization applications and helps determine the workflow, equipment needs and implementation requirements for a particular process.

DCS systems are designed for the largest industrial installations in areas such as oil&gas, chemistry, power plants, pharmaceuticals, pulp and paper mills, etc. Wherever the highest reliability is needed, no downtime of the industrial process which may result in the loss of raw material and blocking of the installation is acceptable, and therefore the DCS system should be considered as base infrastructure. A characteristic feature of DCS is the ability to change hardware configuration and control logic on-line, during operations. DCS is more configurable than programmable, and it usually contains libraries of advanced continuous control algorithms, as these systems are derived from the control of chemical processes, which more often require continuous control algorithms than discrete control. DCS scope of operation is usually the whole installation and the philosophy of action can be described as a top-down fashion.

PLCs have gone a different way to the present state. They come from the automotive industry, from assembly lines where it was necessary to replace control systems with contactor and relay diagrams. Hence, the domain of PLCs for years has been discrete control, timers, counters, comparators and triggers. PLC scope of operation is often part of the process.

These systems also differ in the structure. The DCS system consists of layers: controllers at the process level, and usually redundant computing servers with a proprietary operating system on the system layer where subsequent advanced control modules are run. DCS has central database where all system variables are stored. That is why it is so scalable and manageable even for a huge plant. On the contrary, each PLC has its own data storage space, so there is a need to design custom information exchange between individual controllers and other elements of the control system.

Although the differences between modern DCS and PLC + SCADA based solutions are diminishing and blurring, the approach presented in this article does not mean replacing DCS with a PLC, but rather implementing some of the advanced control DCS functionality in the PLC cooperating with a computational server.

C. IEC 61131 STANDARD FOR PLC PROGRAMMING

The IEC 61131 standard [14] defines basic concepts, general principles, software, and the communication model (data exchange between software components), as well as basic data types and structures for PLC programming. It names two groups of programming languages: text and graphics. The LD graphic language was intended to enable quick and trouble-free programming for electricians who know the principles of the construction of contactor-relay schemes. It is a very transparent language for discrete control applications, but too long-winded and therefore not well readable for the implementation of advanced control algorithms. A similar situation takes place for the FBD language, where the graphic form resembles logic gate diagrams. The text languages, composing the other group of programming languages, can be divided into low-level IL and high-level ST. For the implementation of advanced control algorithms, the ST language (resembling the PASCAL, BASIC, or C languages known from PCs) is the most suitable. Since 2003, when the object-oriented and namespace aspects appeared in the IEC 61131 standard, this language can be said to be similar to object-oriented languages such as C++, JAVA, Python, etc. Certainly, the similarity applies to the syntax, but the specific loop-like manner of PLC operation, with a watchdog system

that supervises the end of all PLC cycle operations, causes a different programming approach for PLC than for classic PC.

D. PLCopen ASSOCIATION

PLCopen [15] is a vendor and product independent association, whose members come from all fields of the industry to deal with topics concerning industrial control software. The areas of PLCopen operation include: faster application development, faster commissioning time, and reduced software life cycle costs. According to [15], current PLCopen topics are: ''Motion Control and Safety functionality, XML data exchange format standardizing the base data of IEC projects in software systems, and mapping to the OPC Unified Architecture for transparent Communication''. The association has undertaken the development of technical specifications related to IEC 61131-3, which resulted in ''standardized libraries for various application fields, harmonized language compatibility levels, and engineering interfaces for interchange and clear communication''. This organization also deals with training, publishing, and certification activities. On their pages you can find a large number of documents that allow you to improve your qualifications and quality, as well as the portability of the produced code for PLC and related industries. Noteworthy is also establishing close cooperation with the OPC UA organization that replicates the communication standards between PLC, DCS, SCADA, industrial databases, measuring devices, actuators, and other elements of automation. Noteworthy is the standard XML-based Exchange Format developed by PLCopen for export and import of IEC 61131-3 projects. Being part of the 61131-10 standard, it can be considered the next step towards a joint description of projects for PLCs irrespective of their manufacturers. Summarizing the PLCopen topic, when the issue of implementing advanced control algorithms on the PLC platform is considered, it is helpful to become familiar with the achievements of this organization.

E. PLC CODER MATLAB TOOLBOX

An important tool for the implementation of advanced control algorithms in PLCs is undoubtedly the Matlab toolbox called PLC Coder [16]. This software is designed to automatically generate the logic developed in the three main Matlab design tools, i.e. Simulink, Stateflow, or Matlab, and save it as m-files in LD and ST PLC languages. The big advantage of PLC Coder is targeting logic to popular platforms for PLC programming such as: CODESYS, Rockwell Automation Studio, Siemens TIA Portal, and OMRON Sysmac Studio. When some hardware platform is not directly supported, you can use the generic path and adapt the received code to your hardware and software. For example, adjusting the code for the PAC GeFanuc controller (which is not originally supported by PLC Coder) used in this article comes down to defining the variables by yourself and dividing the logical part of the program into the initialization part and the rest of the code. The obvious advantage of PLC Coder in Matlab is the ability to use other toolboxes and their contents to generate a code for the PLC. The joint use of the MPC toolbox and PLC Coder allows you to quickly obtain the code of an advanced controller, which can be comprehensively tested in simulated scenarios and environment, and then quickly implemented on the target industrial equipment. It can be called a typical example of the idea of rapid prototyping.

F. SELECTED PAPERS WITH ADVANCED CONTROL IMPLEMENTED IN PLCs

The subject of the implementation of advanced control algorithms in PLC is discussed in scientific and industry articles. The article [17] presents the implementation of a Generalized Predictive Controller (GPC) on Allen Bradley PLC - SCL500 using Function Block Diagram and Structured Text. As the authors claim: ''The results of this work encourage to keep working in developing simple predictive algorithms to be programmed in industrial standard hardware.'' An example used to illustrate the correctness of the implementation was a heating process having the features of a first-order inertial object with delay. The same object was used in [18] to illustrate the operation of a predictive controller with autotuning and constraints on control signals. The comparison of the control quality with the PID controller clearly shows why it is worth making the effort to implement the MPC controller. The third in a series of articles describes MPC implementation with constraints [19], and its authors propose an original approach to ''parameterization of the parametric regions which allows efficiency of definition, effective spanning of the feasible region, and also highly efficient search algorithms''. The whole series of these articles shows that the PLC is an appropriate industrial hardware platform on which both classic and novel advanced control algorithms can be implemented.

Another paper worth mentioning is [20], which describes the implementation of MPC with constraints on input signals using the Nesterov's fast gradient method using [21] approach. The authors successfully implement the MPC algorithm in the Modicon M340 controller from Schneider Electric and test it in a loop with a simulated object having the form of connected tanks. The authors raise the problem of limitations in the available memory and computing power of PLC. The MPC program for the PLC was executed in ST language.

An interesting implementation of an advanced algorithm in PLC is described in [22]. The model predictive control was applied with a projected gradient algorithm designed for controlling nonlinear input constrained systems. The implemented nonlinear MPC was used for a laboratory crane with sampling resolution of 2 ms. A spectacular video documenting successful implementation and handling constraints is provided. In that case, the FESTO PLC was a host platform for implementation, and PLC Coder was used to export the algorithm from Matlab/Simulink to Codesys engineering software.

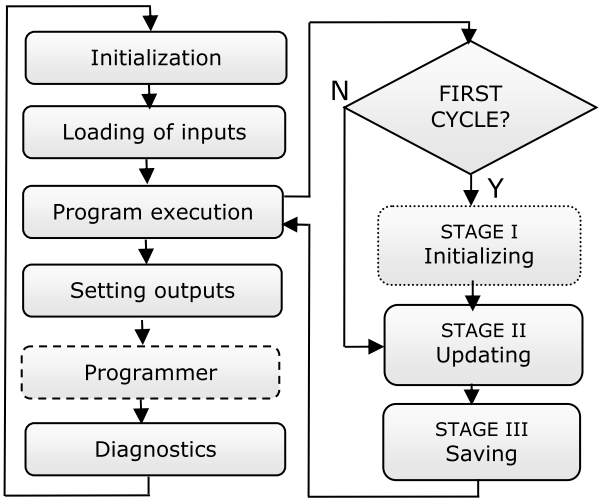

FIGURE 1. PLC scan and 3-stage program framework for implementation formulas given by difference equations.

G. SUMMARY OF THE LITERATURE REVIEW AND GUIDELINE FOR THE ARTICLE

As a result of the query on the literature resources and the largest manufacturers of DCS systems, an approach emerged that advanced algorithms on industrial control platforms are adaptive, predictive, and of fuzzy control type, and they make use of neural networks and broadly understood optimization. The term ''advanced'' also refers to taking into account limitations, multidimensionality, as well as the use of models, self-tuning, and safe control at the limits of the constraints.

Hence, the implementation of predictive control on the PLC platform with on-line adaptation of model parameters was chosen as the main challenge of the paper. The cooperation of PLC with an external solver (of DCS-type) for building the system with optimization was also considered. Two control systems were implemented as an example: constrained multivariable MPC control, and LMI-based robust control.

The article makes use of an idea from the PLC Coder toolkit, i.e. instead of writing programs for PLC directly, it is more convenient and more developmental to create an external application for generating a parametric program for the PLC. The ST language according to IEC 61131 standard and PLCopen materials were used for this purpose.

III. SPECIFICS OF PLC AND OVERCOMING ITS LIMITATIONS

The specificity of the PLC is the use of a simplified real-time operating system implemented as a loop consisting of several operations: reading inputs, executing program logic, setting outputs, diagnostics, and possibly communication with the programmer. All these activities must be completed within the set time, supervised by the watchdog system. Typically, the PLC cycle time without program is several milliseconds, while the maximum time set in the watchdog is of the order of seconds. This means that the time allowed for one iteration of the program cannot exceed seconds. In a natural way, this eliminates the PLC from a certain class of applications, such as solving complex optimization problems. Another limitation is the use of significantly slower CPUs in the PLC than those used in the PC, and limited amount of memory. An important limitation is also the method of PLC programming, in particular the lack of native support for algebraic matrix operations while allowing data saving in the form of a matrix. The specifics of PLC operation are not only limitations - due to the loop-oriented way of operation, it is relatively easy to implement algorithms given by difference equations. This section describes how to overcome the limitations of the PLC and use its properties. It shows how to measure the computing power of the PLC, how to implement algorithms given by difference equations, and how to effectively implement the matrix algebra and divide the algorithm into a number of cycles.

A. PERFORMANCE ANALYSIS – PLC BENCHMARKING

GE Fanuc RX3i was the controller used for testing and implementing advanced control algorithms. Two different CPUs (CPU310 and CPE305) were used for performance testing. The faster of them was then used for further research. 32,640 logical (16-bit) variables can be stored in the available memory, which allows the use of 16320 real variables when operating on real 32-bit numbers [23].

The duration of the cycle depends on the time required to execute the uploaded program and service the connected modules. Based on the information available in the operating instruction manual for a given controller, it is possible to calculate the theoretical cycle time and the program execution time. The times of execution of single instructions are given with an accuracy of 0.01 [μs]. This makes it possible to theoretically determine the computing power of the controller and express it in FLOPS (Floating Point Operation Per Second), which is the number of floating-point operations that the computational unit can perform per second. However, as specified in the instruction manual, the given time values represent the average execution times of given instructions and may vary depending on the inputs. In addition, the impact of high-level ST instructions on the execution time is not described, as the execution times given in the manual apply to the LD language [23]. Therefore, it was decided to test also the performance of the programmable controller in practice.

The most common unit for determining processor performance is FLOPS. The simplest operations such as addition, subtraction, and multiplication were used to evaluate the performance. Performing one of these operations is considered to be equal to one operation on a floating-point number. The processor performance is often determined using Fused Multiply-add (FMA) and its variations, such as Multiply-accumulate (MAC), which take the form (1) [24]:

$$D = AB + C \tag{1}$$

The controller performance evaluation was carried out for two PLC CPUs using 3 programs with different designs,

TABLE 1. Description of methods for determining controller performance.

Descriptive number	Description of method
1	Determined theoretically
2	Program that uses only high-level instructions
3	Program that uses high-level and simple instructions
4	Program that uses only simple instructions

TABLE 2. Test results for MAC operations performed in several ways.

Processor	Method	Performance [FLOPS]	Program size [BYTES]
CPU310	1	467 289	---
	2	467 426	1 139
	3	1 895 743	48 186
	4	2 848 837	608 136
CPE305	1	1 197 604	---
	2	1 243 523	1 139
	3	4 953 560	48 186
	4	7 145 833	608 136

implementing MAC operations. The tests were performed using the ST language, as the use of a high-level language allows for better overall performance measurements than the tests based on low-level languages [25]. The methods used for determining the controller performance are described in Table 1, while the results are given in Table 2.

For three different programs, three different execution times and three different determined computing capacities of the controller were obtained. This is due to the fact that by testing the computational performance with a program, we determine the effectiveness of a given program [26]. Therefore, it can be concluded that the least effective implementation of calculations is the use of high-level instructions, while the most effective is the complete absence of them. A good alternative is to carry out part of operations with loops and part with basic operations. This will significantly reduce the program execution time and, at the same time, keep the code transparent.

It can be concluded that the performance of both CPUs obtained after executing the program using basic operations is the value closest to the actual performance of the controller and close to the maximum possible efficiency that the user program can achieve.

Having the information about the PLC performance, the user can estimate whether his algorithm can be implemented on this platform, and/or eventually what the maximum values of the algorithm's parameters may be.

B. PLC IMPLEMENTATION METHODS OF ALGORITHMS GIVEN WITH DIFFERENCE EQUATIONS

Due to the loop/cyclic operation, PLCs are naturally predisposed to the implementation of estimation, filtration, and control algorithms defined by difference equations. The notation in the form of difference equations is recursive and given by the relations of appropriate quantities at the present time with past times. Three phases: initialization, updating, and saving are sufficient for complete implementation of the algorithm given by a differential equation. The context of their use adapted to the cyclical operation of PLC is shown in Fig. 1. In the initialization phase, which is usually called only once with the first PLC cycle, the initial conditions are given to all quantities. The updating phase includes correct description of the algorithm based on the given difference equation. In the saving phase, usually at the end of the code, the current data is saved in relevant variables, which will be then treated as variables delayed by one step in the next controller cycle.

TABLE 3. 3-Stage PLC implementation of 2nd order oscillation dynamics with delay given by difference equation.

Stage	Operations
Initialization (only in the first PLC cycle)	`y_new = 0; y = 0; y1 = 0;` `u3 = 0; u2 = 0; u1 = 0;`
Update	`y_new =1.469*y-0.6703*y1+` `+0.1077*u2+0.09414*u3;`
Save	`y1 = y; y = y_new;` `u3 = u2; u2 = u1; u1 = u;`

The first example of 3-stage implementation is the dynamics of a 2nd order oscillation model with delay given by (2):

$$G(s) = \frac{\omega^2 e^{-\tau s}}{(s^2 + 2\xi\omega s + \omega^2)} \qquad \tau = 2s, \quad \omega = 0.5, \ \xi = 0.4 \tag{2}$$

After discretization with sampling time T = 1 s we get

$$H(z) = \frac{Y(z)}{U(z)} = z^{-2}\frac{0.1077z + 0.09414}{z^2 - 1.469z + 0.6703} \tag{3}$$

After regrouping, this can be written as

$$y(k)(z^2 - 1.469z + 0.6703) = u(k)z^{-2}(0.1077z + 0.09414) \tag{4}$$

$$y(k+2) - 1.469y(k+1) + 0.6703y(k) = 0.1077u(k-1) + 0.09414u(k-2) \tag{5}$$

and finally:

$$y(k+1) = 1.469y(k) - 0.6703y(k-1) + 0.1077u(k-2) + 0.09414u(k-3) \tag{6}$$

In order to preserve the dynamics of the object modeled in this way, it is necessary to keep the intervals between particular cycles equal to the sampling period. This effect can be achieved programmatically by using timer blocks, or by forcing a fixed PLC cycle time in the CPU. The 3-stage implementation scheme is presented in Table 3.

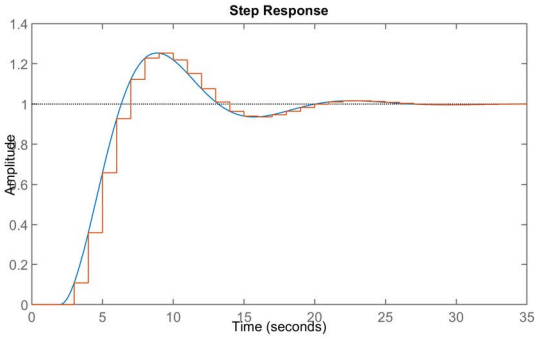

FIGURE 2. Step response of G(s), H(z) and dynamics described by difference equation (6). Time series for H(z) and for dynamics (6) are the same.

TABLE 4. 3-Stage PLC implementation of normalized gradient estimation method given by difference equation.

Stage	Operations
Initialization (only in the first PLC cycle)	`theta = theta_init;`

Update	`eps = theta'*fi-y;`
	`m = sqrt(kappa+fi'*fi);`
	`theta_new = theta-(gamma*fi*eps)/m;`

Save	`theta = theta_new;`

Note the specific reverse order of assigning variables in the save stage.

The time series for (2), (3) and (6) with initial conditions from Table 3 and sampling time of 1s are presented in Fig. 2.

Another example of 3-stage implementation is the normalized gradient estimation method from [27]. The algorithm described in matrix (vector) form is:

$$\theta(t+1) = \theta(t) - \frac{\Gamma\phi(t)\varepsilon(t)}{m^2} \tag{7}$$

$$\theta(t_0) = \theta_0, \quad t \in \{t_0, t_0+1, \ldots\}$$

$$0 < \Gamma = \Gamma^T < 2I$$

$$m(t) = \sqrt{\kappa + \phi(t)^T\phi(t)}$$

$$\kappa > 0 \tag{8}$$

$$\varepsilon(t) = \theta^T(t)\phi(t) - y(t) \tag{9}$$

where:

$\theta(t)$ is the parameter vector to be estimated,

θ_0 is the initial estimate vector at time t_0,

Γ, κ are the design parameters,

I is the identity gain matrix,

$y(t)$ is the measured output of the modeled plant,

$\varepsilon(t)$ is the estimation error,

$\phi(t)$ is the vector with delayed inputs and outputs.

The 3-stage implementation scheme for this case is presented in Table 4.

```
' STAGE 1 - Initialization
if #FST_SCN then
   y_new := 0.0; y := 0.0; y1 := 0.0;
   u := 0.0; u1 := 0.0; u2 := 0.0; u3 := 0.0;
   th1 := 0.0; th2 := 0.0; th3 := 0.0; th4 := 0.0;
   kappa := 0.1;
end_if;

' STAGE 2 - Update
u := 1.0+0.1*noise;
y_new := 1.469*y-0.6703*y1+0.1077*u2+0.09414*u3;

m_sq_rev := 1.0/(kappa+y*y+y1*y1+u2*u2+u3*u3);
eps := (th1 * y + th2 * y1 + th3 * u2 + th4 * u3)-y_new;
th1_new := th1 - y  * eps * m_sq_rev;
th2_new := th2 - y1 * eps * m_sq_rev;
th3_new := th3 - u2 * eps * m_sq_rev;
th4_new := th4 - u3 * eps * m_sq_rev;

' STAGE 3 - Save
y1 := y;    y := y_new;
u3 := u2;   u2 := u1;    u1 := u;
th1 := th1_new; th2 := th2_new;
th3 := th3_new; th4 := th4_new;
```

FIGURE 3. ST-language 3-stage code for PLC with implementation of oscillation dynamics and normalized gradient parameter estimation method.

In order to illustrate the implementation and performance of the estimation algorithm, it was used to determine the parameters of the model from the first example. It was assumed and stated in (10) that the initial values of the estimated parameters are zero, the parameter kappa = 0.1, and the gamma matrix is unitary.

$$\begin{aligned}
\theta^* &= (1.469,\ -0.6703,\ 0.1077,\ 0.09414)\\
\theta(0) &= (0,\ 0,\ 0,\ 0)\\
\theta(t) &= (th1(t), th2(t), th3(t), th4(t))\\
\phi(t) &= (y(k), y(k-1), u(k-2), u(k-3))
\end{aligned} \tag{10}$$

Based on the algorithm described by formulas (7)-(9) and the 3-stage implementation method from Table 4, the program for PLC was developed as shown in Fig. 3. Several ST-specific features can be seen: each line ends with a semicolon, the assignment operator is: =, as well as the high-level language construct if..then..end_if; and distinguishing INT types from REAL in that the latter must be written in full dotted form even if the numbers are zeros.

The initialization phase is based on the #FST_SCN system variable, which is on high state only in the first cycle after PLC start. The input signal is realized as the sum of the constant value 1 and a pseudorandom number generated based on system timers available in the PLC. Fig. 4 shows the trajectories of parameter estimates.

The speed of convergence of the estimates depends on parameters kappa and gamma, but also on the noise level at the input of the simulated object. Fig. 4 shows the ±1 waveform of noise, and a much faster convergence can be obtained by significantly increasing the noise. As it can be seen, the estimates are heading towards the ideal values, which indicates the correctness of the implementation.

For the purposes of the illustration, simple and clear examples were chosen, but many different, more advanced

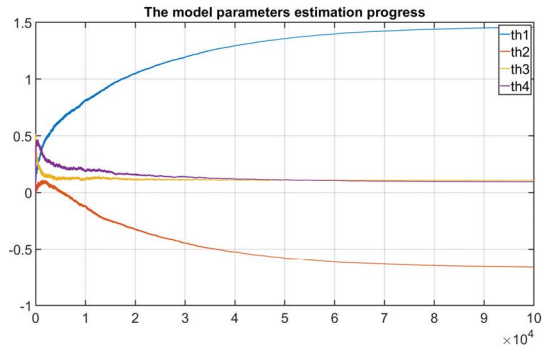

FIGURE 4. The model parameter estimation progress.

algorithms for control, estimation, and filtering of models of dynamic objects defined by differential equations can be implemented in the PLC using the presented three-stage method.

As can be seen in the presented example, matrix operations were performed in a scalar manner due to the lack of native support for matrix operations in the PLC. Proper programming of matrix operations in PLC seems to be crucial, due to the memory management and the speed of execution of algorithms with large matrix sizes.

C. DATA STORAGE IN MATRICES AND MATRIX OPERATIONS

The IEC 61131 standard specifies structures such as matrices, but the manufacturers usually allow working with matrices by referring to individual elements. There are also restrictions on matrix size, for instance the maximum amount of data in the matrix in the GE Fanuc RX3i controller is 9999. In addition, the PLCs do not have built-in instructions for matrix operations, so all the necessary operations need to be implemented. That is why the efficiency analysis of various methods of performing matrix operations was undertaken.

One of the most important features of the computer program is its efficiency, which describes the use of computer hardware resources: computing power and memory. The time consumed by the computer unit to execute the algorithm and the memory usage are critical for evaluating algorithm's effectiveness. Frequently, the order of the number of operations is used to determine the size of the problem, rather than the exact number of operations. This order can be expressed with $O(f(n))$, where n is the measure of the size of the problem, and for $n \to \infty$ the number of operations is $cf(n)$, where c is a constant independent of n. According to this description, the multiplication of matrices with sizes $n \times m$ and $m \times p$ takes $O(nmp)$ number of multiplication and addition operations. In particular, the multiplication of square matrices has the order of $O(n^3)$ operations. Different algorithms can only differ by the coefficient c. The order of the algorithm is the best performance indicator, which tells how it will be able to deal with large problems [28]. Most matrix operations can be done in more than one way. There are $(3! = 6)$ possible ways to do matrix multiplication [29]. For each method, the order of the algorithm is $O(nmp)$ or $O(n^3)$ for a square matrix, but each method reads and writes data differently, which may affect the execution time of the operation. To reduce the number of operations, the Strassen algorithm can be used, which reduces the order of the algorithm by reducing the number of multiplication operations required. It is assumed in the Strassen algorithm that the matrices A and B are square matrices of equal dimensions. This allows the number of multiplication operations to be reduced from 8 to 7 for a 2×2 matrix. However, as the number of multiply operations is decreased, the number of addition operations increases. The addition of two $k \times k$ size matrices is a problem of the order of $O(k^2)$, so for a sufficiently large number of operations performed with the Strassen algorithm, the number of required mathematical operations is smaller than for the traditional multiplication algorithm. Therefore, recursively dividing the multiplied matrices into P matrices if the matrix size n is of the form 2^e makes that the algorithm can be used e times and then the traditional matrix multiplication operation can be used for each submatrix of size $\leq n/2^e$ [29]. The general order of the Strassen algorithm can be described as $O(n^{2.807})$, which is a significant reduction compared to traditional methods of matrix multiplication. However, for present implementation of the multiplication operation in predictive control, it was decided to use the conventional matrix multiplication methods because:

- In the implemented predictive control algorithm, the multiplied matrices (with assumed dimensions $n \times n$) are smaller than the point for which the Strassen algorithm becomes more efficient [29], [30] so the multiplication making use of a traditional method will be more efficient.
- The Strassen algorithm uses more memory than traditional methods [31],[32].
- The matrices on which the operations will be performed are not square matrices, therefore, in order to use the Strassen algorithm, the given matrices should be supplemented with zeros, which additionally increases both memory use and computational complexity [28].
- In part of the operation, the matrix elements can be non-negative and filled with small values. Therefore, in order to be able to use the Strassen algorithm, an error analysis should be additionally performed, because this algorithm does not guarantee the accuracy comparable with that of traditional methods [29].

During the analysis, it was decided to carry out matrix multiplication using several methods with different ways of accessing the data and different number of high-level ST language instructions. The applied methods and the obtained results are presented in Table 5.

The analysis was performed for several matrix sizes, the results given in Table 5 are for the 50×50 matrix.

The best method under consideration is method 2, because it is 2.6 times faster than method 1, without significantly

TABLE 5. Performance ratio of programs.

Descriptive number	Method	Performance ratio of programs
1	Updating all C elements using 3 for loops	45.06
2	Computing each element of matrix C in one operation using 2 for loops	17.12
3	Carrying out matrix multiplication using elementary operations without use of high-level structures	2.92
Classical methods taking into account the multiplication of upper- with lower-triangular matrices occurring in the control law.		
4	Skipping zero values of one matrix	23.52
5	Skipping zero values of both matrices using additional conditional statements.	16.24
6	Skipping zero values of both matrices using only elementary operations.	1

increasing the size of the program. If the fastest execution of the program is needed, method 3 is 5.8 times faster than method 2. The cost of using this method is a very large code size, which makes the program unreadable. For example, to perform matrix multiplication, the code had to be divided into 14 ST blocks, due to the ST block size limitation of 0.131072 megabytes.

Method 2 was used during the implementation of predictive control on the programmable controller.

Matrix inversion can be considered a problem when solving linear equations. The methods for solving linear equations can be divided into two categories, which are:

- Finite methods of solving systems of linear equations,
- Iterative methods.

The iterative methods are used for matrix sizes n of 10^3 or greater. The disadvantage of iterative methods is that for two different matrices of the same size, the method may require a different number of steps, compared to the constant number of steps executed in a classical method. Moreover, the number of steps and the execution time of the algorithm itself significantly depend on the conditions of the system [33]. Therefore, when selecting the algorithm for implementation of matrix inversion in the MPC control, the analysis of finite methods for solving systems of linear equations was performed.

The first method considered was the determinant method using the following ways to calculate the determinant of the matrix and the append matrix:

- Laplace's minor expansion method along a row or column for which the order of the algorithm is $n!$ It is therefore impractical [34].
- More efficient Gaussian elimination method for which the determinant method has the order of computational complexity of $2n^5$.

The implementations of the Gauss-Jordan elimination or LU decomposition methods were considered, they have the same computational complexity order of $2n^3$ [35].

The method with lower computational complexity is the Cholesky decomposition method, which uses only the L or U matrix. The cost of the algorithm's execution is half that of the LU algorithm and its order is n^3 [35], However, for an adaptive algorithm, it should be proved that the inverted matrix will be positively determined for each parameter value that can be derived during the estimation. Applying the Cholesky decomposition to the matrix that is not positively defined makes that the algorithm is not numerically correct, which means that the error value can be arbitrarily large.

Full matrix inversion would be one of the most time-consuming operations performed during the algorithm operation. It is necessary only when all values of the control horizon are to be determined. When the algorithm uses RHC (Receding Horizon Control), it is enough to determine only the first value of the control horizon. Consequently, in that algorithm it is enough to calculate the first row of the inverse matrix.

The implemented predictive control algorithm uses RHC, therefore only the first row of the inverted matrix can be calculated. This approach is the most efficient of all described, it can be implemented using both the determinant method and the Gauss-Jordan elimination method.

The Gauss-Jordan elimination method is more efficient, and for larger sizes of the inverted matrix, $n > 25$, it enables two-fold time reduction compared to full matrix inversion.

Due to the small size of the inverted matrix, the Gauss-Jordan elimination method was used with full matrix inversion, but the modification transforming the algorithm for single-row inversion is very simple. Both methods were built on the basis of [36].

The implementation of single-line inversion is justified when we need to additionally reduce the time and/or length of the control horizon.

D. SHARING CALCULATIONS AMONG CYCLES

The implementation of the predictive control algorithm is associated with the need to perform complex calculations, the execution time of which can be long when implemented on a programmable controller.

In some cases, the computation time may exceed the maximum PLC cycle time and the control system sampling time may be longer than the computation time. In this case, we can divide the math operations into several function blocks and execute them sequentially in successive PLC cycles.

Fig. 5 presents the program for dividing mathematical operations into successive cycles of controller operation. In this program, the calculation execution time for each function block was 1.4842 s, which means that the execution of all calculations takes 5.9368 s. This time is twice as long as the maximum allowable cycle duration in the GE Fanuc Rx3i controller. The program ladder with set and reset coils causes that one function block is calculated in one controller cycle, so that the entire algorithm does not lead to controller crash. This solution is used for systems where the sampling period is longer than the maximum cycle duration.

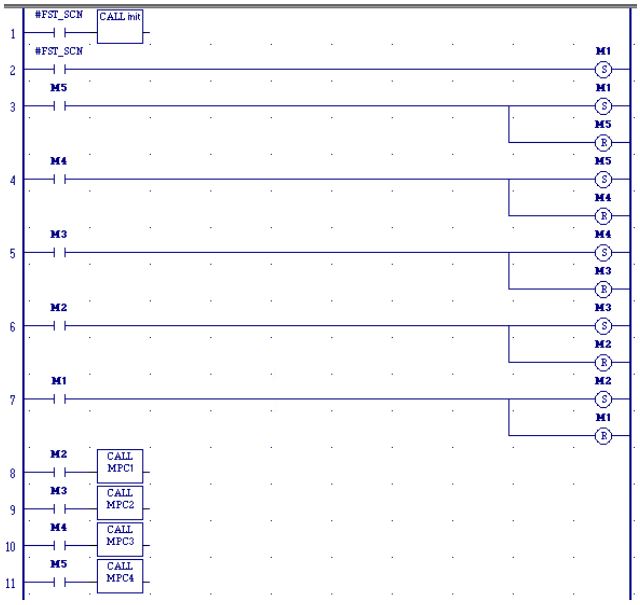

FIGURE 5. Program for dividing mathematical operations into successive cycles of controller operation.

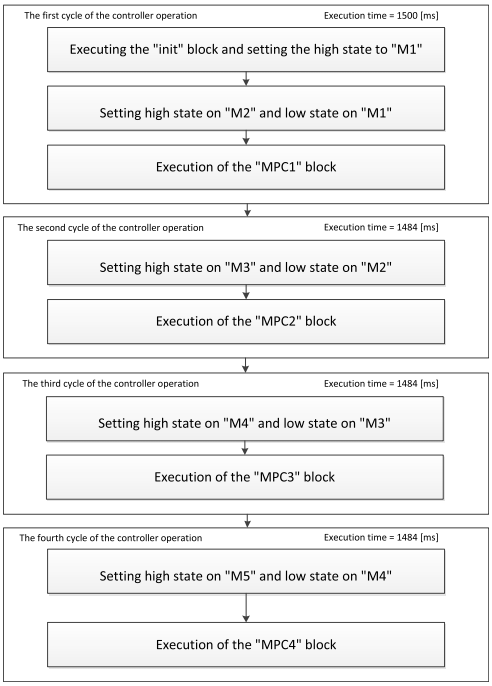

FIGURE 6. Program execution for the first four control cycles.

Fig. 6 presents the program execution for the first four control cycles.

IV. CODE GENERATOR FOR aMPC CONTROLLER

To illustrate the implementation of the advanced control algorithm on the PLC hardware platform, the MPC [37] and DMC [38] algorithms, and their versions with adaptation of model parameters, i.e. aMPC and aDMC, were selected.

The regular approach is to create and enter the code in the environment for configuring and programming PLC controllers. However, it was decided instead to develop an individual, independent, external application in C++ that will generate a text file in ST language for the PLC with the MPC algorithm included, taking into account algorithm parameters set by the user, such as sampling, prediction, and control horizons.

This section presents the MPC algorithm and the parameter adaptation procedure using the least squares method. The implementation is divided into steps. The task of the code generator application is to fill the program content using scalar algebraic operations so as to obtain the result of the matrix operation intended to be implemented in a given step.

In the implemented predictive control systems, the controller was created on the basis of a non-parametric object model and had the form of a step response (DMC) or impulse response (MPC). The main feature of the discussed systems is the assumption that the process is asymptotically stable [38]. Despite this assumption, algorithms based on nonparametric models are often used because limiting only to stable processes is rarely justified [39]. Therefore, a general approach is adopted in which the object is assumed to be one-dimensional and asymptotically stable. On the basis of this assumption, it was possible to develop a program that, based on the impulse or step response and the given prediction and control horizon lengths, would generate the algorithm code in ST language.

A. aMP ALGORITHM DESCRIPTION

The MPC algorithm was created on the basis of the control object dynamics model having the form of a non-parametric impulse response model. A feature of the MPC algorithm is the assumption of object stability, which in the case of impulse response allows assuming that the values of the ignored response parameters are equal to zero. The algorithm was developed on the basis of [37].

For MPC regulation, the model takes the form (11):

$$\begin{aligned} y(k) &= Vu(k-1) \\ V &= v_o + v_1 z^{-1} + \cdots + v_n z^{-nV} \end{aligned} \tag{11}$$

The used model is of the moving average type in which the coefficients of the polynomial V are the parameters of the impulse response and is therefore called the convolutional model. The model assumes that the discrete delay time is equal to 1. Therefore, it is assumed that the first coefficients of the V polynomial may be equal to 0. During the synthesis of the MPC controller, the discrete delay time was not used, so that in the adaptive form the controller is not very sensitive to changes in delay time. The used incremental predictor had the following form (12):

$$\bar{y}(k+j) = \bar{y}(k+j-1) + K^m u(k+j) \tag{12}$$

where K^m is the model transmittance. When applied to (11), Equation (12) takes the form (13):

$$\bar{y}(k+j) = \bar{y}(k+j-1) + V\Delta u(k+j-1) \tag{13}$$

where:

$$\bar{y}(k) = y(k) \tag{14}$$

Since Equation (13) uses future control values, V transforms to the form (15):

$$V = V_j^1 + z^{-j}V_j^2 \tag{15}$$

where:

$$\begin{aligned} V_j^1 &= v_o + v_1 z^{-1} + \cdots + v_n z^{-j+1} \\ V_j^2 &= v_j + v_{j+1} z^{-1} + \cdots + v_n z^{-nV+j} \end{aligned} \tag{16}$$

After substituting (15) into (13) we get Equation (17):

$$\bar{y}(k+j) = \bar{y}(k+j-1) + V_j^1\Delta u(k+j-1) + V_j^2\Delta u(k-1) \tag{17}$$

It follows from Equations (16) and (17) that

$$\bar{y}^0(k+j) = \bar{y}^0(k+j-1) + V_j^2\Delta u(k-1) \tag{18}$$

The vector of control increments is denoted as (19):

$$\delta\bar{u} = [\delta u(i), \delta u(i+1), \cdots, \delta u(i+H-1)]^T \tag{19}$$

Based on (16), (17), and (18), the prediction vector can be written as (20):

$$\bar{y} = Q\delta\bar{u} + \bar{y}^0 \tag{20}$$

where:

$$\bar{y} = [\bar{y}(i+1), \bar{y}(i+2), \cdots, \bar{y}(i+H)]^T \tag{21}$$

$$Q = \begin{bmatrix} v_0 & 0 & \cdots & 0 \\ v_1 & v_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ v_{L-1} & v_{L-2} & \cdots & h_0 \\ \vdots & \vdots & \ddots & \vdots \\ v_{H-1} & v_{H-2} & \cdots & h_{H-L} \end{bmatrix} \tag{22}$$

In Matrix (22), v_j and h_j are, respectively, the parameters of the impulse response and step response of the process.

The main purpose of the MPC controller is to minimize the distance between the predicted trajectory of the output signal and the reference trajectory, taking into account the weight for control deviation from the $u(i-1)$ value. Therefore, the minimized criterion function takes the form:

$$J = \sum_{j=1}^{H} \{[\bar{y}(k+j) - \bar{w}^0(k+j)]^2 + \rho\delta u^2(k+j-1)\} \tag{23}$$

where the value of the variable $\delta\bar{u}$ minimizing Equation (23) can be calculated analytically with formula (24):

$$\delta u_{opt} = (Q^T Q + pI)^{-1} Q^T(\bar{w}^o - \bar{y}^o) \tag{24}$$

where:

$\bar{w}^o$ is the reference trajectory vector

$\bar{y}^o$ is the prediction vector of the control object output. Assuming zero increments of the control signal, the control signal is constantly equal to $u(k-1)$.

The DMC algorithm was developed on the basis of [38].

For the two examined predictive control algorithms it was assumed that the model used for approximation would be a discrete model of finite impulse response, having the form (25):

$$y_m(t) = \sum_{j=1}^{N} v_j u(t-1), \quad t \geq t_o \tag{25}$$

The following relationships were used to change the form of the impulse response model into the step response model (26):

$$\begin{aligned} h_j &= \sum_{j=1}^{i} v_j \\ v_i &= h_i - h_{i-1} \end{aligned} \tag{26}$$

where v_i are the impulse response coefficients, and h_j are the step response coefficients. The purpose of the recursive algorithm based on the normalized least squares method is to minimize the cost function consisting of the sum of squared errors with a penalty for the initial estimate $\theta(t_0) = \theta$ of variable θ^*. The cost function was defined as (27):

$$\begin{aligned} J(\theta) = \frac{1}{2}\sum_{t_0}^{t} \frac{(\theta^T(t)\phi(\tau) - Y(\tau))^2}{m^2(\tau)} d\tau \\ + \frac{1}{2}((\theta(t) - \theta_0)^T P_0^{-1}(\theta(t) - \theta_0)) \end{aligned} \tag{27}$$

where:

$\theta(t)$ is the unknown vector of estimated parameters,

$\phi(t)$ is the vector of previous control input values.

The equations defining the algorithm of the normalized least squares method are (28) [27]:

$$\begin{aligned} \varepsilon &= y_m(t) - y(t) \\ m(t) &= \sqrt{\mathrm{K} + \phi^T(t)P(t-1)\phi(t)} \\ \theta(t+1) &= \theta(t) - \frac{P(t-1)\phi(t)\varepsilon(t)}{m^2(t)} \\ P(t) &= P(t-1) - \frac{P(t-1)\phi(t)\phi^T(t)P(t-1)}{m^2(t)} \end{aligned} \tag{28}$$

where $K > 0$ is the design factor, $P(t)$ is the gain matrix, and $m(t)$ is the normalizing signal. The required initial values are: $P(0)$ - the gain matrix value at initial time, and $\theta(0)$ - parameter estimates at initial time.

B. aMPC ALGORITHM IMPLEMENTATION

For the purpose of implementing DMC/MPC control in ST language, the algorithm has been divided into individual steps listed in Fig. 7.

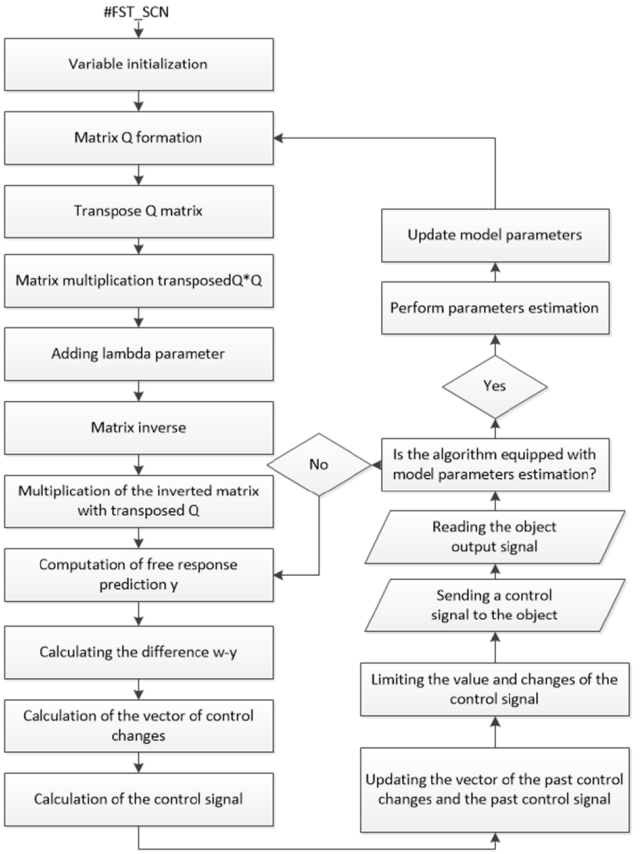

FIGURE 7. Algorithm divided into individual steps.

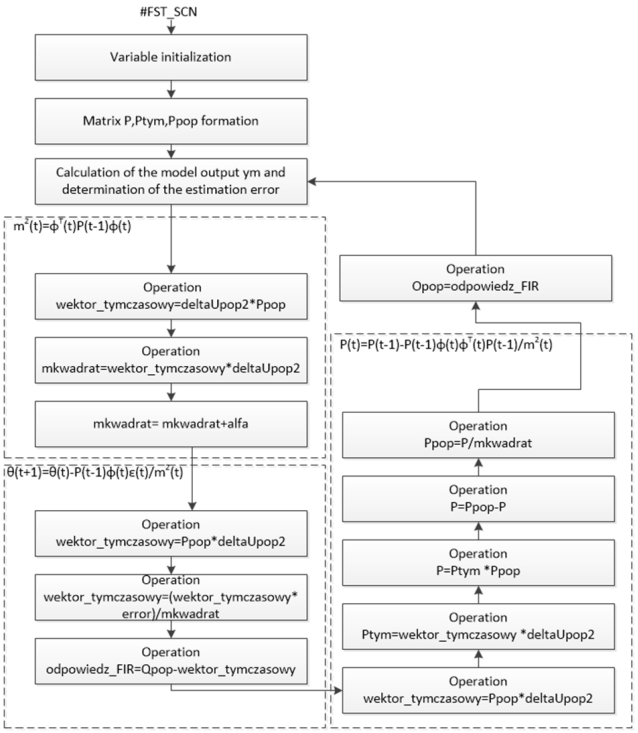

FIGURE 8. Algorithm of the normalized least squares method divided into individual steps.

Due to the lack of built-in matrix and vector operating instructions in the PLC, all necessary operations must be separately performed in a scalar manner. Therefore, the implementation of the MPC algorithm for the PLC described by formula (24) requires significantly more code in relation to matrix-supported environments, such as SciLab, Matlab, or MathCad. To facilitate the implementation and control tests, it was decided to generate the code through application in C++. The developed application uses method 2 of matrix operations (described in previous section), which is a compromise between the performance and size of the program for matrix implementation and vector multiplication. The application user specifies the length of the prediction and control horizon and the object step response vector on the basis of which the regulator code is generated.

The algorithm of the normalized least squares method divided into individual steps is shown in Fig. 8.

C. CODE GENERATOR C++ APPLICATION

The main purpose of the developed application for generating the code for the programmable controller is to write the code in ST language based on the given step response and the given length of the control and prediction horizon. The developed program saves the generated code in properly titled text files. The files generated by the program include:

1. The file ''Initialization'', containing initial values of the predictive algorithm and the variables that should be initialized in the controller,
2. The file ''DMC'' (or ''MPC'') containing the user-selected prediction algorithm,
3. The file ''RLS'' that contains the least squares algorithm.

The code generator program has been extended with functions in which regulator parameters are selected automatically, based on given process parameters or step response. The standardized least squares method, modified according to [40], is used to determine object parameters, assuming inertia with delay dynamics of the object. The scheme of program's operation is shown in Fig. 9.

Fig. 10 shows the graphic interface of the program. The type of algorithm is selected as one of ''DMC, MPC, aDMC, aMPC'' buttons. After selecting the algorithm, the user can enter the controller parameters, or select one of the available automatic parameter selection methods. Step response, or the response to variable signal and set signal need to be given in text files and can be directly copied from the Matlab environment. The developed interface checks the entered parameters and warns the user in case incorrect data is provided. Then, after entering the correct parameters, the button ''Generate code'' should be pressed to save the generated code in appropriate files. The program will also determine the expected cycle duration and the amount of memory required. The estimated time applies to the Rx3i controller with CPE305

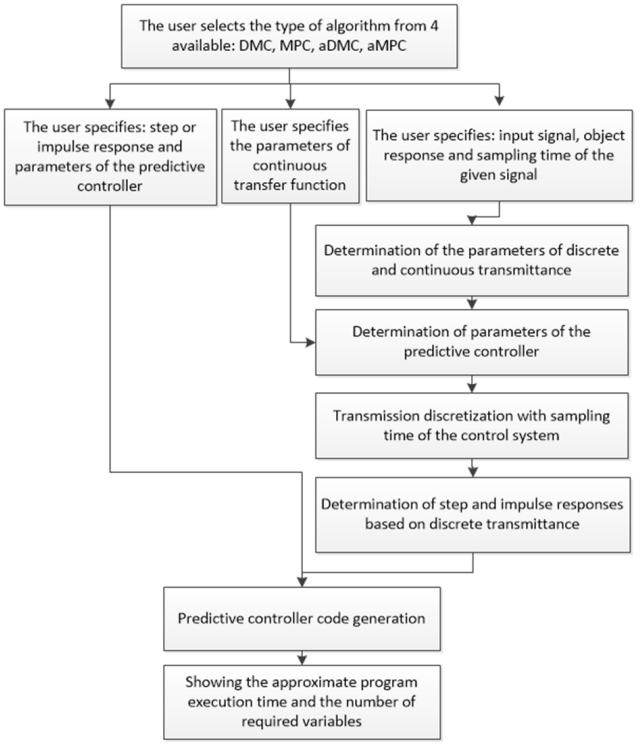

FIGURE 9. Scheme of code generator operation.

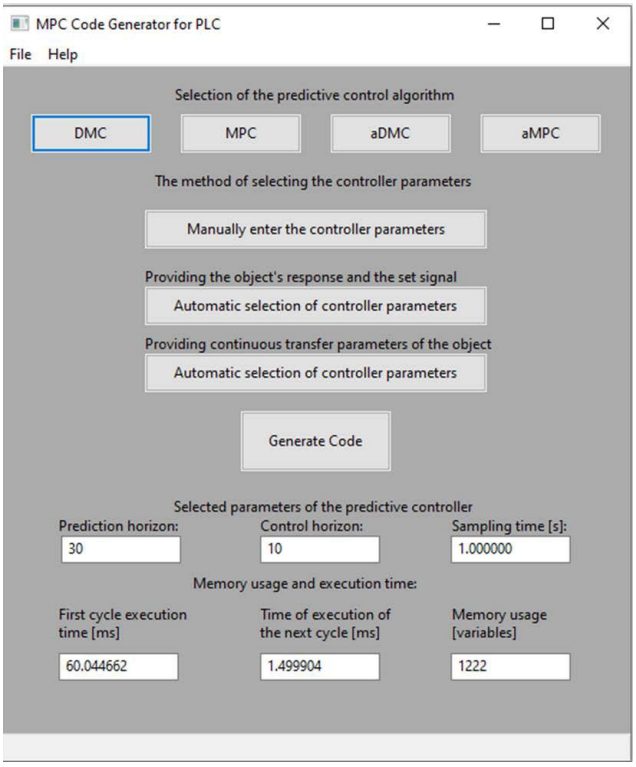

FIGURE 10. Graphic user interface of code generator.

central unit. In order to facilitate the use of the developed applications, programs have been built that install the code generation applications along with appropriate libraries.

D. aMPC VERIFICATION

Being a typical object that often occurs in industrial applications, the first-order inertia with constant delay was selected as the test object. The model transmittance is described by (29):

$$G(s) = \frac{0.497}{2.16s + 1} e^{-7s} \tag{29}$$

1) SOFTWARE-IN-THE-LOOP

Before testing the operation of the system in a hardware loop, a discrete version of the object can be programmed on the PLC and the control system can be tested using the PLC only. In the Proficy Machine Edition Logic Developer PLC environment, the data monitor window allows to view the values of variables on a graph.

Figs. 11-14 show the tracking of the set trajectories.

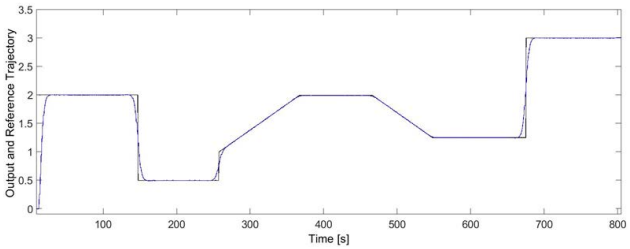

FIGURE 11. Response from DMC control (blue), and reference trajectory (black).

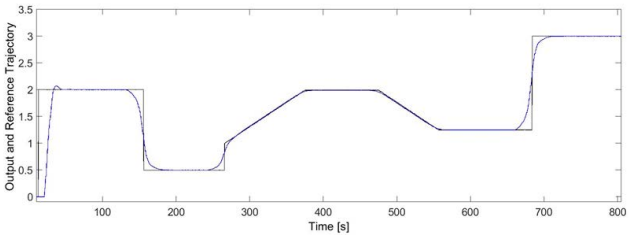

FIGURE 12. Response from aDMC control (blue), and reference trajectory (black).

2) HARDWARE-IN-THE-LOOP

The system located in the Computer Controlled Systems laboratory at the Gdansk University of Technology, Faculty of Electrical and Control Engineering, was used to implement the predictive control in the Hardware-in-the-loop (HIL) system. The diagram of the hardware structure of the system is shown in Fig. 15.

The system consists of a GE Fanuc RX3i controller with CPE 305 main unit equipped with IC694ALG442 module. Using the ALG442 module, the controller was connected via the PLCD9710 connection board to the Advantech PCI1711

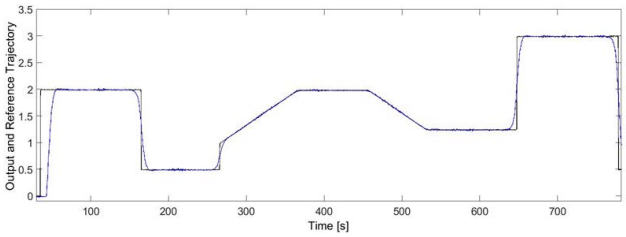

FIGURE 13. Response from MPC control (blue), and reference trajectory (black).

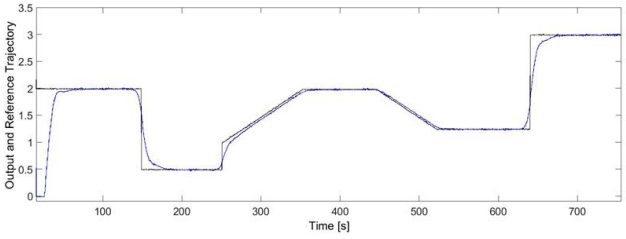

FIGURE 14. Response from aMPC control (blue), and reference trajectory (black).

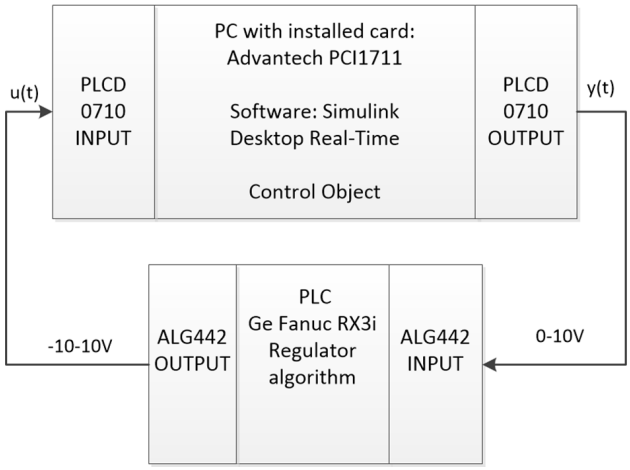

FIGURE 15. Diagram of hardware structure of the system.

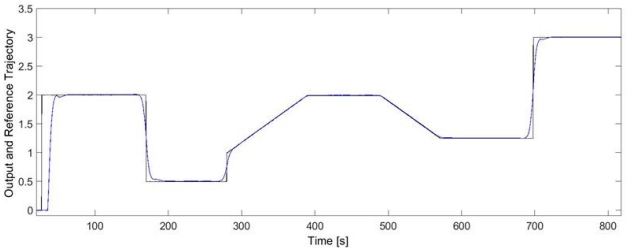

FIGURE 16. Response from DMC control (blue), and reference trajectory (black).

acquisition card. The performance of the control systems is presented in Figs. 16-19.

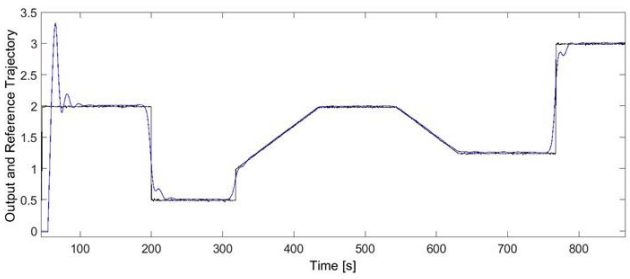

FIGURE 17. Response from aDMC control (blue), and reference trajectory (black): 100% increase of object gain parameter.

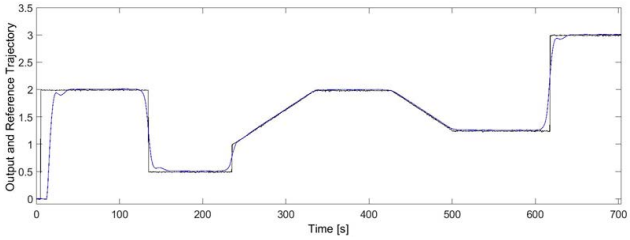

FIGURE 18. Response from MPC control (blue), and reference trajectory (black).

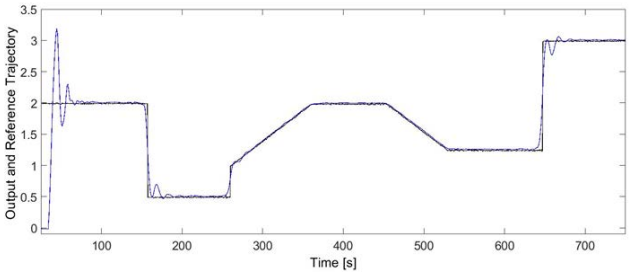

FIGURE 19. Response from aMPC control (blue), and reference trajectory (black): 100% increase of object gain parameter.

All test scenarios ended with accurate tracking of the reference trajectory. For the version with parameter adaptation, more extensive research was carried out after introducing non-stationarity in simulation model parameters. If the parameters changed slowly enough for the estimation procedure to capture them, and the MPC mechanism was able to take these changes into account (the prediction and control horizons were appropriate), the regulation was stable and qualitatively correct for all changes of gain, time constant, and delay. This proves the correctness of the structure of the code generation tool.

Along with natural advantages, such as the possibility to synthesize the algorithm for arbitrary parameters specified by the user, the applied approach has very useful features, such as calculating the memory demand and the execution time of one cycle of the controller program by the code generator application. This means that the user can conclude on the applicability of the controller with given parameters and the

TABLE 6. Brief comparison of digital devices able to perform control algorithms.

	Cycle time	Industrial applications	Flexibility	Scalability	Advanced control	Cost
PLC	ms	H	M	H	L	M
FPGA	ns	M	M	L	M	M
DSP	µs	M	M	L	M	M
IPC	µs	M	H	H	H	M
DCS	ms	H	M	H	H	H
Embedded	µs	M	L	L	M	L
Single board computer	µs	M	H	M	H	L

Legend: H-high, M-medium, L-low

specific PLC CPU. Another huge advantage of the presented approach is potential code portability.

V. COOPERATION WITH EXTERNAL COMPUTATIONAL SERVER

It is assumed in this section that due to PLC drawbacks and limitations, the algorithm used in the control system cannot be implemented in the PLC, even using the guidelines given in the previous section. These may be the cases which require significant computing power or, for instance, the availability of libraries or specialized toolkits.

A. ALTERNATIVE CONTROL DEVICES

The article basically concerns the implementation of advanced control algorithms in a PLC hardware platform, but some of the described algorithms can also be implemented with other devices. A brief comparison is shown in Table 6.

A comparison of the control devices characteristics can be found in [41]. The table has been updated and supplemented on the basis of information gathered in [42]–[44]. A detailed comparison of FPGA and DSP platforms can be found in [45]. The following categories of comparison were presented: single cycle duration, preparation for operation in industrial conditions, flexibility, scalability, predisposition to advanced control algorithms implementation and cost.

The category ''industrial applications'' means not only the DIN rail mounting, heavy-duty chassis, power redundancy, operation in harsh industrial conditions, but also features related to the so-called maintenance, i.e. hot-swap and online redundancy and high availability guarantee. Flexibility is understood as configurability and the possibility of wide application to various purposes and industries. Scalability is the possibility of unrestricted development of the control infrastructure with growing requirements, based on the possibility of supplementing, expansion, without having to replace the main part of the equipment and change the whole concept. The advanced control section addresses the ability and ease of implementing advanced control algorithms.

It should be noted that the three-scale assessments presented in Table 6 are evaluative and are the result of averaging features and opinions. For example, the price of a PLC can range from $100 to several thousand. It should be noted that the target versions of a given device were taken into account, and not rapid development environmentals. For example, a great solution for the implementation of advanced control algorithms can be the National Instruments cRIO system, SpeedGoat or dSpace hardware platforms with LabView or Matlab rapid prototyping software. However, these are not the mainstream target industrial hardware platforms but rather laboratory R&D equipment.

There has been a significant technological progress in the single board computer hardware class (also called IoT - Internet of Things platforms) due to budget Raspberry Pi or Intel Galileo solutions. While DSP and FPGA provide high time resolution and the implementation of specialized control algorithms, it is difficult to use them to solve optimization problems or use C/C++/Python libraries with advanced control algorithms. Basically all-purpose computers are suitable for these tasks, and recently also single-board computers. In the following chapter of the article, a solution based on a PLC (as the basic platform for industrial control) and a computing server is considered. The role of the server, depending on the computational requirements, can be performed by an industrial PC or a single-board computer. The PLC and computing server set has all the advantages of Table 6 while respecting the price and industrial conditions regime.

It is worth emphasizing once again that this is not the only solution, but the one that is considered in the article where PLC is the main control device.

B. PLC-PC CONTROL STRUCTURE AND REAL-TIME OPERATION

The solution to the problem of communication between the PLC and the computing server with the provision of real-time operation is described below, while the next section presents the verification of the proposed control structure using a multidimensional non-linear object and two control methods: QDMC and robust control based on LMI.

The structure of the system with external computation server is presented in Fig. 20.

A properly built control system is expected to work in the real-time regime. This means that after receiving the current information about the state of the object from the measurement, the control system is able to develop and provide to the object a control signal before the sampling time expires. The implementation is relatively simple when performed inside the PLC, due to real-time nature of the PLC operating system. However, when the control signal is calculated outside the PLC, the round-trip communication (to and from the computing server) and the calculation time in the computing server must be taken into account. There are different approaches to implementing the real-time

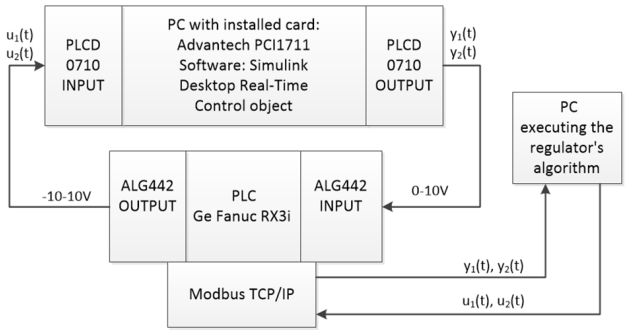

FIGURE 20. Control structure with external computation server.

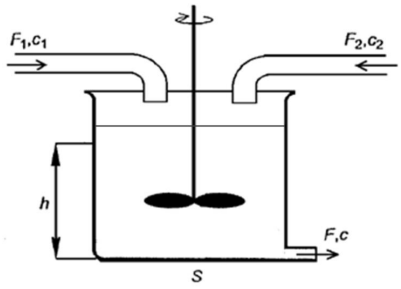

FIGURE 21. Continuous stirred-tank reactor (CSTR).

postulate [46]. The implementation of hard real-time regime means a full guarantee of control signal calculation in a preset time. This is achieved by a fully deterministic approach to both communication and computing. There is also a soft real-time approach, in which incidental overruns of computation times may occur. Soft real-time systems are usually not based on guarantees, but on practical approach and analyses of regular performance of individual elements composing the control system. The specificity of simulation with real-time models is also presented in [47]. The soft real-time approach was used as a prototype solution in this article. The computing server was a computer with OS Windows 10 with Matlab Optimization Toolbox software and YALMIP library [48]. The communication made use of the Modbus TCP protocol and the PLC built-in communication mechanism asynchronous to the cycle based on the Direct Memory Access.

After receiving the new measurement information from the object, the PLC sets a flag in the form of a discrete variable about the availability of new data. The external computing server scans frequently whether this flag is set. If so, the new data is downloaded from the PLC to the server, the flag is cleared, and the data is used to calculate the control signal which is then sent to the PLC after the calculation. Another flag is used to mark a new control signal.

A reasonable supplement to the above solution is to develop a strategy for situations in which an externally generated control signal does not come. This may have a form of maintaining the previous control signal, or a look-up table solution, or using a simpler algorithm executed in the PLC to ensure suboptimal but object-safe control.

For presentation purposes it was assumed that the controlled plant operates with the step of 1 s. The conclusions from the performance analysis of individual system components are as follows: the one-way communication usually takes less than 1 ms, while the calculations in the external server do not exceed 200 ms, and their time duration depends on the control algorithm used. In this way, a real-time working effect is achieved, without a guarantee but with practical confirmation resulting from hours of real tests.

For industrial applications, the above-presented idea is fully applicable, but the hardware and software parts should be replaced with hard-real time solutions. That means that a real-time network should be used, Profinet for instance, the computing server should work with a real-time system, e.g. RT Linux or QNX, and the calculation software should guarantee finite and predictable computation time.

C. THE CONTROL OBJECT-CSTR

A continuous stirred-tank reactor (CSTR) was selected as an example of control object. This plant is non-linear and has two inputs: flow rates F_1 and F_2 of the liquid with concentrations c_1 and c_2, respectively, and two outputs: the volume V of the liquid in the tank and the concentration c of the liquid flowing out of the tank through free outflow. Fig. 21 presents a schematic of the plant.

D. QDMC REGULATOR FOR MULTIVARIABLE CONTROL WITH CONSTRAINTS

The optimization problem in the predictive control algorithm with constraints on inputs, outputs, and input increments for the cost function J cannot be solved directly in PLC. However, the problem can be turned into a quadratic programming problem that will be solved on an external computing server. The constrained optimization problem after transformation is presented as (30):

$$\min_{\Delta u}(\Delta u^T H \Delta u + c^T \Delta u) \tag{30}$$

with constraints $A\Delta u \leq b$ given by(31):

$$\begin{cases} \Delta u_{min} \leq \Delta u(k) \leq \Delta u_{max} \\ u_{min} \leq u(k) \leq u_{max} \\ y_{min} \leq y(\hat{k}) \leq y_{max} \end{cases} \tag{31}$$

Matrix H in (30) is called the Hessian, while c^T is the gradient, and Δu is the control increment.

The above predictive control algorithm with constraints performs the task of tracking the reference trajectory [38].

E. LMI BASED REGULATOR FOR MULTIVARIABLE ROBUST CONTROL

Classic predictive control algorithms use models with constant parameters and structure, which do not take into account the influence of disturbances and uncertainty of parameters. The solution to this problem is robust predictive control.

In [49], the authors present an approach to robust predictive control using linear matrix inequalities (LMI).

The uncertain system is the system whose parameters change over time, which can be described as (32):

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k) \\ y(k) &= Cx(k)[A(k)B(k)] \in \Omega \end{aligned} \tag{32}$$

where $u(k) \in \Re^{n_u}$ is the control, $x(k) \in \Re^{n_x}$ is the state of the object, $y(k) \in \Re^{n_y}$ is the output of the object, Ω is some predefined set, $A(k)$ is the state matrix, and $B(k)$ is the input matrix.

Based on the input/output measurement data recorded at various operating points, it is possible to determine a set of L linearized models [49], which forms the polytopic set Ω (33):

$$\Omega = Co\{[A_1B_1], [A_2B_2], \ldots, [A_LB_L]\} \tag{33}$$

where Co refers to a convex shell and L is the number of its vertices.

The robust predictive control algorithm with an infinite prediction horizon uses the cost function in the form (34):

$$J_\infty(k) = \sum_{i=0}^{\infty} \{x(k+i|k)^T Q_1 x(k+i|k) + u(k+i|k)^T R u(k+i|k)\} \tag{34}$$

where $Q_1 > 0$ and $R > 0$ are symmetric weight matrices. In cases when the prediction horizon $p = \infty$, the Infinite Horizon MPC (IH-MPC) approach is used. The use of an infinite prediction horizon guarantees nominal stability.

The optimization problem is defined as (35):

$$\min_{u(k+i|k), i=0,1,\ldots,m} \max_{[A(k+i)B(k+i)]\in\Omega, i\geq 0} J_\infty(k) \tag{35}$$

The min-max approach consists in determining the controls on the control horizon m as a result of minimizing the cost of the worst case.

The given problem, although convex for a finite m, is computationally insolvable [49]. However, it can be solved by specifying the upper limit of the objective function.

After transformation, the optimization problem can be written as the minimization problem with linear matrix inequalities LMI in the form (36):

$$\min_{\gamma, Q, Y} \gamma \tag{36}$$

and constraints (37) and (38):

$$\begin{bmatrix} 1 & x(k|k)^T \\ x(k|k) & Q \end{bmatrix} \geq 0 \tag{37}$$

$$\begin{bmatrix} Q & QA_j^T + Y^T B_j^T & QQ_1^{1/2} & Y^T R^{1/2} \\ A_jQ + B_jY & Q & 0 & 0 \\ Q_1^{1/2}Q & 0 & \gamma I & 0 \\ R^{1/2}Y & 0 & 0 & \gamma I \end{bmatrix} \geq 0 \tag{38}$$

The controls are determined on the basis of the state feedback control law $u(k|k) = Fx(k|k)$ where the matrix F is determined as (39):

$$F = YQ^{-1} \tag{39}$$

The robust control algorithm with input and output constraints determines the control by solving the optimization problem (40):

$$\min_{\gamma, Q, X, Y, Z} \gamma \tag{40}$$

with constraints (41), (42), (43) and (44):

$$\begin{bmatrix} 1 & x(k|k)^T \\ x(k|k) & Q \end{bmatrix} \geq 0 \tag{41}$$

$$\begin{bmatrix} Q & QA_j^T + Y^T B_j^T & QQ_1^{1/2} & Y^T R^{1/2} \\ A_jQ + B_jY & Q & 0 & 0 \\ Q_1^{1/2}Q & 0 & \gamma I & 0 \\ R^{1/2}Y & 0 & 0 & \gamma I \end{bmatrix} \geq 0 \tag{42}$$

$$\begin{bmatrix} X & Y \\ Y^T & Q \end{bmatrix} \geq 0, \quad X_{rr} \leq u_{r,\max}^2, \; r = 1, 2, \ldots, n_u \tag{43}$$

$$\begin{bmatrix} Z & C(A_jQ + B_jY) \\ (A_jQ + B_jY)^T C^T & Q \end{bmatrix} \geq 0$$

$$Z_{rr} \leq y_{r,\max}^2, \quad r = 1, 2, \ldots n_y, \; j = 1, 2, \ldots, L \tag{44}$$

The aim of the presented algorithm of robust predictive control using linear matrix inequalities is to bring the states to the zero state, and thus bring the object to the selected operating point.

F. VERIFICATION RESULTS

The performance of predictive control with constraints was tested using HIL. The optimization problem was resolved on a PC in the Matlab environment. The calculated controls were sent to the programmable controller, which then controlled the object in the form of a model in the Matlab/Simulink environment on a PC. Figs. 22 and 23 show the results of tracking the set trajectories for the output volume V and concentration c. Similarly, the robust predictive control algorithm with constraints was tested in the HIL loop. The optimization task was performed on a PC in the Matlab environment with YALIMP. The designated control signals were sent to the programmable controller, and then to the object in the form of a model in another Matlab/Simulink. Figs. 24 and 25 show the effects of forcing the outputs to the values appropriate for the selected operating point for the volume V and concentration c, at the base value of concentration c2 = 0.1.

In order to test the robustness of the control system, the tests were repeated for different values of coefficient c_2. Figs. 26 and 27 show controlling the outputs to the values appropriate for the selected operating point set for the output of volume V and concentration c, at the base value of concentration $c_2 = 1$.

The use of the external computing server allowed the implementation of complex predictive control algorithms that

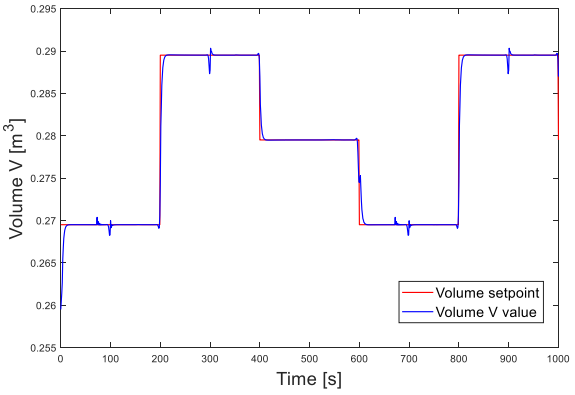

FIGURE 22. Operation of the predictive control system for a nonlinear object using the QDMC method for the task of tracking the trajectory of a given volume. Prediction horizon p = 150, control horizon m = 15.

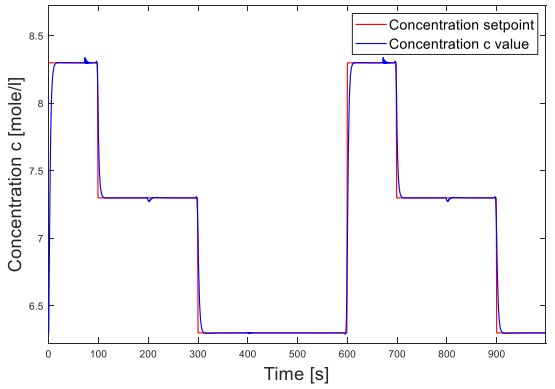

FIGURE 23. Operation of the predictive control system for a nonlinear object using the QDMC method for the task of tracking the trajectory of a given concentration. Prediction horizon p = 150, control horizon m = 15.

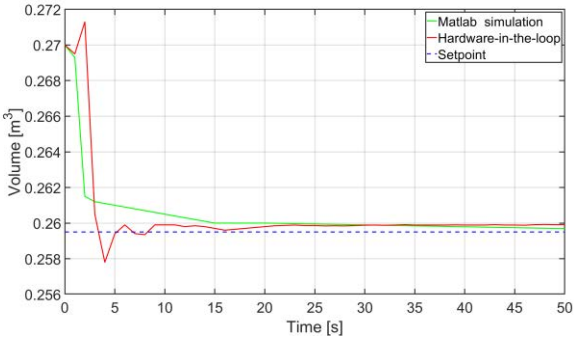

FIGURE 24. The operation of the robust algorithm for concentration $c_2 = 0.1$ and the set reference values of volume $V_{ref} = 0.2595$ and concentration $c_{ref} = 6.3$ for volume V.

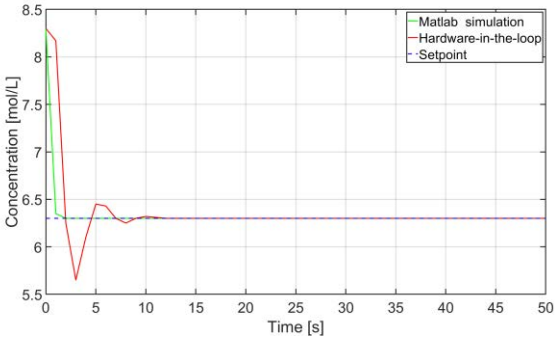

FIGURE 25. The operation of the robust algorithm for concentration $c_2 = 0.1$ and the set reference values of volume $V_{ref} = 0.2595$ and concentration $c_{ref} = 6.3$ for concentration c.

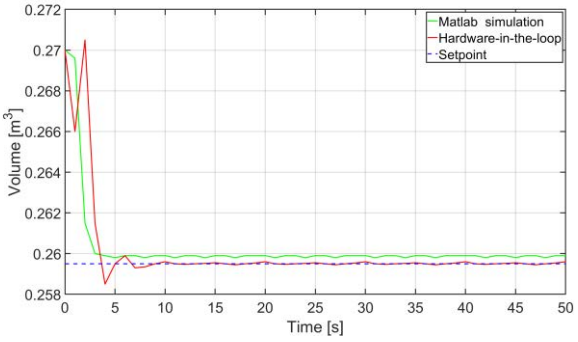

FIGURE 26. The operation of the robust algorithm for concentration $c_2 = 1$ and the set reference values of volume $V_{ref} = 0.2595$ and concentration $c_{ref} = 6.3$ for volume V.

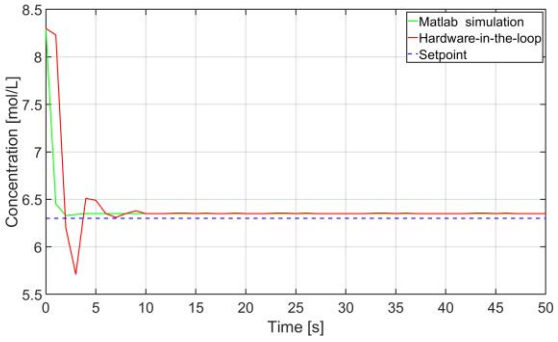

FIGURE 27. The operation of the robust algorithm for concentration $c_2 = 1$ and the set reference values of volume $V_{ref} = 0.2595$ and concentration $c_{ref} = 6.3$ for concentration c.

could not be implemented on the programmable controller. The two presented algorithms worked in soft real-time regime and allowed for controlling a nonlinear and multidimensional object with constraints, as well as robust control. The time of calculating the control signals and communication with the controller was shorter than the sampling time of the object. The algorithms enabled tracking the reference values of object states, as well as the fulfillment of certain constraints on inputs and outputs.

VI. CONCLUSION

PLCs usually work with implemented simplest control algorithms or as part of a hierarchical control system, where the PLC receives from another device a setpoint to be maintained. This article shows that with some preparation effort, it is possible to implement advanced control algorithms on the PLC platform.

PLC limitations concerning cycle times, memory sizes, and the lack of support for matrix arithmetic operations are widely discussed. A general method of implementing algorithms described by difference equations is presented. The properties of individual methods of multiplication and subtraction of key matrices for the implementation of advanced control algorithms in the presence of strong limitations of the PLC platform are carefully considered.

The implementation of aMPC and aDMC algorithms was developed, which allowed for regulation of disturbed, non-stationary objects with large and time-varying delay. The developed application produces the code for the PLC, but after minor changes it can also do it for microcontrollers, single-board computers, or industrial computers. If more complex algorithms need to be used, the concept of dividing calculations between PLC cycles is presented, as a consequence of which the time of one cycle is not a final limitation anymore. The most advanced control methods can be implemented using the presented concept of PLC cooperation with an external computing server. The multi-dimensional predictive control with input and output constraints for a CSTR plant was presented. In order to prove wide possibilities of the proposed solution, the concept of a robust control based on an external solver and libraries for LMI was included. The verification using the hardware-in-the-loop structure as close as possible to real conditions confirmed the correctness of implementation and the usefulness of the presented methods.

REFERENCES

[1] J. Tarnawski, "Realizacja programowa algorytmów filtracji, estymacji i sterowania w PLC/PAC," *Pomiary Automatyka Robotyka*, vol. 17, no. 196, pp. 100–107, May 2013.
[2] J. Tarnawski, "Implementacja algorytmu regulacji predykcyjnej MPC w sterownikach programowalnych," *Pomiary Automatyka Robotyka*, vol. 17, no. 197, pp. 100–107, Jun. 2013.
[3] M. Korzeniowski, "Predictive control of a multidimensional chemical reactor with continuous flow of components," (in Polish), M.S. thesis, Dept. Intell. Control Decis. Support Syst., Fac. Elect. Control Eng., Gdańsk Univ. Technol., Gdańsk, Poland, 2020.
[4] P. Kudełka, "Implementation of the predictive control algorithm in the programmable controller with the analysis of computational complexity," (in Polish), M.S. thesis, Dept. Intell. Control Decis. Support Syst., Fac. Elect. Control Eng., Gdańsk Univ. Technol., Gdańsk, Poland, 2020.
[5] T. Kondakci and W. Zhou, "Recent applications of advanced control techniques in food industry," *Food Bioprocess Technol.*, vol. 10, no. 3, pp. 522–542, Mar. 2017, doi: 10.1007/s11947-016-1831-x.
[6] P. Airikka, "Advanced control methods for industrial process control," *Comput. Control Eng.*, vol. 15, no. 3, pp. 18–23, Jun. 2004, doi: 10.1049/cce:20040303.
[7] P. Tatjewski, *Advanced Control of Industrial Processes: Structures and Algorithms*. London, U.K.: Springer, 2007.
[8] T. L. Blevins, G. K. McMillan, W. K. Wojsznis, and M. W. Brown, *Advanced Control Unleashed: Plant Performance Management for Optimum Benefit*. Research Triangle Park, NC, USA: ISA, 2002.
[9] J. Flaus and J. Georgakis. (2018). *Review of Machine Learning Based Intrusion Detection Approaches for Industrial Control Systems*. Accessed: Jan. 7, 2022. [Online]. Available: https://www.semanticscholar.org/paper/Review-of-machine-learning-based-intrusion-for-Flaus-Georgakis/3165069b0b93c2517745426690170545 9dda7814
[10] M. Conti, D. Donadel, and F. Turrin, "A survey on industrial control system testbeds and datasets for security research," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 4, pp. 2248–2294, 2021, doi: 10.1109/COMST.2021.3094360.
[11] *Emerson—Advanced Control and Optimization*. Accessed: Jan. 7, 2022. [Online]. Available: https://www.emerson.com/pl-pl/automation/operations-business-management/control-operator-performance/advanced-control-and-optimization
[12] *Yokogawa—Platform for Advanced Control and Estimation (Advanced Process Control)*. Accessed: Jan. 7, 2022. [Online]. Available: https://www.yokogawa.com/solutions/products-platforms/solution-based-software/optimization/advanced-process-control-platform-for-advanced-control-and-estimation/
[13] *Honeywell—Advanced Process Control*. Accessed: Jan. 7, 2022. [Online]. Available: https://www.honeywellprocess.com/en-U.S./training/programs/advanced-applications/Pages/advanced-process-control.aspx
[14] *IEC 61131-3:2013*. Accessed: Jan. 7, 2022. [Online]. Available: https://webstore.iec.ch/publication/4552
[15] *PLCopen Association*. Accessed: Jan. 7, 2022. [Online]. Available: https://plcopen.org/
[16] *Mathworks PLC Coder*. Accessed: Jan. 7, 2022. [Online]. Available: https://www.mathworks.com/products/simulink-plc-coder.html
[17] G. Valencia-Palomo, K. R. Hilton, and J. A. Rossiter, "Predictive control implementation in a PLC using the IEC 1131.3 programming standard," in *Proc. Eur. Control Conf. (ECC)*, Aug. 2009, p. 1322, doi: 10.23919/ECC.2009.7074588.
[18] G. Valencia-Palomo and J. A. Rossiter, "Programmable logic controller implementation of an auto-tuned predictive control based on minimal plant information," *ISA Trans.*, vol. 50, no. 1, pp. 92–100, Jan. 2011, doi: 10.1016/j.isatra.2010.10.002.
[19] G. Valencia-Palomo and J. A. Rossiter, "Efficient suboptimal parametric solutions to predictive control for PLC applications," *Control Eng. Pract.*, vol. 19, no. 7, pp. 732–743, Jul. 2011, doi: 10.1016/j.conengprac.2011.04.001.
[20] M. Pereira, D. Limon, D. Munoz de la Pena, and T. Alamo, "MPC implementation in a PLC based on Nesterov's fast gradient method," in *Proc. 23rd Medit. Conf. Control Autom. (MED)*, Jun. 2015, pp. 371–376, doi: 10.1109/MED.2015.7158777.
[21] S. Richter, C. N. Jones, and M. Morari, "Computational complexity certification for real-time MPC with input constraints based on the fast gradient method," *IEEE Trans. Autom. Control*, vol. 57, no. 6, pp. 1391–1403, Jun. 2012, doi: 10.1109/TAC.2011.2176389.
[22] B. Kaepernick and K. Graichen, "PLC implementation of a nonlinear model predictive controller," in *Proc. 19th IFAC World Congr.*, Cape Town, South Africa, 2014, pp. 1892–1897.
[23] *PACSystems CPU Reference Manual, GFK-2222L*. GE Fanuc Intelligent Platforms. Accessed: Jan. 7, 2022. [Online]. Available: https://eia.pg.edu.pl/documents/1113028/0/gfk2222l.pdf
[24] R. Dolbeau, "Theoretical peak FLOPS per instruction set: A tutorial," *J. Supercomput.*, vol. 74, no. 3, pp. 1341–1377, Mar. 2018, doi: 10.1007/s11227-017-2177-5.
[25] H. J. Curnow, "A synthetic benchmark," *Comput. J.*, vol. 19, no. 1, pp. 43–49, Jan. 1976, doi: 10.1093/comjnl/19.1.43.
[26] J. E. Gentle, *Matrix Algebra: Theory, Computations, and Applications in Statistics*. New York, NY, USA: Springer, 2007.
[27] G. Tao, *Adaptive Control Design and Analysis*, 1st ed. Hoboken, NJ, USA: Wiley, 2003.
[28] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 4th ed. Baltimore, MD, USA: Johns Hopkins Univ. Press, 2013.
[29] S. Huss-Lederman, E. M. Jacobson, A. Tsao, T. Turnbull, and J. R. Johnson, "Implementation of Strassen's algorithm for matrix multiplication," in *Proc. ACM/IEEE Conf. Supercomputing (CDROM)*, Pittsburgh, PA, USA, Jan. 1996, p. 32, doi: 10.1145/369028.369096.
[30] P. D'Alberto and A. Nicolau, "Using recursion to boost ATLAS's performance," in *Proc. 6th Int. Symp. High-Performance Comput. 1st Int. Conf. Adv. Low Power Syst.*, Berlin, Heidelberg, 2005, pp. 142–151.
[31] J. Huang, T. M. Smith, G. M. Henry, and R. A. Van De Geijn, "Strassen's algorithm reloaded," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2016, pp. 690–701, doi: 10.1109/SC.2016.58.
[32] B. Boyer, J.-G. Dumas, C. Pernet, and W. Zhou, "Memory efficient scheduling of Strassen-Winograd's matrix multiplication algorithm," in *Proc. Int. Symp. Symbolic Algebr. Comput.*, Séoul, South Korea, Jul.2009, p. 8. Accessed: Jan. 7, 2022. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00163141
[33] Å. Björck, *Numerical Methods in Matrix Computations*. New York, NY, USA: Springer, 2014.

[34] J. D. Hoffman, J. D. Hoffman, and S. Frankel, *Numerical Methods for Engineers and Scientists*, 2nd ed. Boca Raton, FL, USA: CRC Press, 2017, doi: 10.1201/9781315274508.
[35] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed. Philadelphia, PA, USA: SIAM, 2002.
[36] K. D. Dorfman and P. Daoutidis, *Numerical Methods with Chemical Engineering Applications*, 1st ed. Cambridge, U.K: Cambridge Univ. Press, 2017.
[37] A. Niederlinski, J. Moscinski, and Z. Ogonowski, *Regulacja Adaptacyjna*. Warsaw, Poland: Wydawnictwo Naukowe PWN (in Polish), 1995.
[38] E. F. Camacho and C. B. Alba, *Model Predictive Control*. London, U.K.: Springer, 2013.
[39] J. A. Rossiter, *A First Course in Predictive Control*, 2nd ed. Boca Raton, FL, USA: Taylor & Francis, 2018.
[40] P. Kudełka and M. Korzeniowski, "Synthesis of adaptive regulators for a laboratory stand with transport of warm air," (in Polish), M.S. thesis, Dept. Intell. Control Decis. Support Syst., Fac. Elect. Control Eng., M.S. thesis, Gdańsk Univ. Technol., Gdańsk, Poland, 2018.
[41] W. Grega, *Metody i Algorytmy Sterowania Cyfrowego w Ukladach Scentralizowanych i Rozproszonych*. Kraków, Poland: Uczelniane Wydawnictwa Naukowo-Dydaktyczne Akademii Górniczo-Technicznej (in Polish), 2004.
[42] P. Zhang, *Advanced Industrial Control Technology*, 1st ed. Amsterdam, The Netherlands: William Andrew, 2010.
[43] B. R. Mehta and Y. J. Reddy, *Industrial Process Automation systems?: Design and Implementation*. Amsterdam, The Netherlands: Elsevier, 2015. Accessed: Jan. 07, 2022. [Online]. Available: https://www.nlb.gov.sg/biblio/202651517
[44] K. L. S. Sharma, *Overview of Industrial Process Automation*, 2nd ed. Amsterdam, The Netherlands: Elsevier, 2017.
[45] E. Monmasson and M. N. Cirstea, "FPGA design methodology for industrial control systems—A review," *IEEE Trans. Ind. Electron.*, vol. 54, no. 4, pp. 1824–1842, Aug. 2007, doi: 10.1109/TIE.2007.898281.
[46] P. A. Laplante and S. J. Ovaska, *Real-Time Systems Design and Analysis: Tools for the Practitioner*, 4th ed. Hoboken, NJ, USA: Wiley, 2012.
[47] J. Tarnawski and T. Karla, "Real-time simulation in non real-time environment," in *Proc. 21st Int. Conf. Methods Models Automat. Robot. (MMAR)*, Aug. 2016, pp. 577–582, doi: 10.1109/MMAR.2016.7575200.
[48] J. Lofberg, "YALMIP: A toolbox for modeling and optimization in Matlab," in *Proc. IEEE Int. Conf. Robot. Automat.*, Sep. 2004, pp. 284–289, doi: 10.1109/CACSD.2004.1393890.
[49] M. V. Kothare, V. Balakrishnan, and M. Morari, "Robust constrained model predictive control using linear matrix inequalities," *Automatica*, vol. 32, no. 10, pp. 1361–1379, 1996, doi: 10.1016/0005-1098(96)00063-5.

JAROSLAW TARNAWSKI was born in Gdansk, in 1974. He received the M.Sc. and Ph.D. degrees from the Gdansk University of Technology, in 2000 and 2006, respectively. He is currently an Assistant Professor with the Department of Electrical Engineering, Control Systems and Computer Science, Gdańsk University of Technology. He is the Head of the Laboratory of Computer Control Systems with industrial-class infrastructure, including DCS, PLC, SCADA systems, and industrial IT networks. He has experience in the field of control of drinking water supply systems and in nuclear power control systems. His research interests include mathematical modeling, identification, optimization, hierarchical control systems, and adaptive and predictive control for objects with time-varying delays.

PIOTR KUDEŁKA was born in Wejherowo, Poland, in 1996. He received the M.Sc. degree in automatics and robotics from the Gdańsk University of Technology, in 2020. His research interests include adaptive and predictive control, computational complexity, programmable logic controllers, and efficiency and computational performance.

MATEUSZ KORZENIOWSKI was born in Pasłęk, Poland, in 1996. He received the M.Sc. degree in automatics and robotics from the Gdańsk University of Technology, in 2020. His research interests include adaptive and predictive control, robust control, neural networks, optimization, and genetic algorithms.