sensors

MDPI

Article

Bridging the Gap in Technology Transfer for Advanced Process Control with Industrial Applications

Vitali Vansovits [1], Eduard Petlenkov [1], Aleksei Tepljakov [1], Kristina Vassiljeva [1] and Juri Belikov [2,*]

[1] Department of Computer Systems, Tallinn University of Technology, 12618 Tallinn, Estonia; vitali.vansovits@taltech.ee (V.V.); eduard.petlenkov@taltech.ee (E.P.); aleksei.tepljakov@taltech.ee (A.T.); kristina.vassiljeva@taltech.ee (K.V.)
[2] Department of Software Science, Tallinn University of Technology, 12618 Tallinn, Estonia
* Correspondence: juri.belikov@taltech.ee

Abstract: In the present paper, a software framework comprising the implementation of Model Predictive Control—a popular industrial control method—is presented. The framework is versatile and can be run on a variety of target systems including programmable logic controllers and distributed control system implementations. However, the main attractive property of the framework stems from the goal of achieving smooth technology transfer from the academic setting to real industrial applications. Technology transfer is, in general, difficult to achieve, because of the apparent disconnect between academic studies and actual industry. The proposed software framework aims at bridging this gap for model predictive control—a powerful control technique which can result in substantial performance improvement of industrial control loops, thus adhering to modern trends for reducing energy waste and fulfilling sustainable development goals. In the paper, the proposed solution is motivated and described, and experimental evidence of its successful deployment is provided using a real industrial plant.

Keywords: advanced control; model predictive control; industrial process; Industry 4.0; technology transfer

Citation: Vansovits, V.; Petlenkov, E.; Tepljakov, A.; Vassiljeva, K.; Belikov, J. Bridging the Gap in Technology Transfer for Advanced Process Control with Industrial Applications. *Sensors* **2022**, *22*, 4149. https://doi.org/10.3390/s22114149

Academic Editor: Maysam Abbod

Received: 4 May 2022
Accepted: 27 May 2022
Published: 30 May 2022

1. Introduction

Control systems are ubiquitous in industry [1]. At the same time, the way that these control systems are designed, configured, and deployed has been frequently criticized [2]. Indeed, it is commonplace that the deployed controllers underperform significantly in their intended application domain, which leads to considerable resource and energy waste, but can be improved through proper implementation and tuning using optimization [3].

Optimization is, in general, one of the key development trends in many areas including manufacturing and related industries. In part, this is motivated by the vision of the fourth industrial revolution—commonly referred to as Industry 4.0 [4]. From the perspective of optimization, major concerns are related to the efficiency of production processes, consumption of raw materials, utilization of assets, various emissions, etc. Meanwhile, it is not possible to do a full upgrade of existing industrial assets at once; consequently, to achieve the transition to sustainable and efficient modes of industrial operation, the analysis of utilized technologies, development of optimization solutions, and their deployment to existing manufacturing systems must take place. In fact, a vast amount of optimization algorithms has been developed throughout the years to address the mentioned issues at all levels of production processes [5]. New materials, mechanical solutions, and control algorithms can be implemented to mitigate the bottlenecks of a certain process to make production use fewer materials, produce less emissions, and make products of the highest possible quality.

As far as control systems are concerned, the majority of existing systems still use proportional-integral-derivative (PID) controllers. They are easy to set up and deploy,

but they may also be difficult to tune, considering the number of control loops involved in a given production process. Indeed, this is a serious bottleneck that has its roots in general lack of knowledge and technology transfer (KTT) between academia dealing with control system research and the industry which heavily relies on such knowledge to improve all aspects of control system performance [2,3].

There are certain successful trends to improve existing PID loops performance utilizing Industry 4.0 techniques such as big data collection and cloud computing to collect control loop diagnostic data and analyze their performance [6]. On the other hand, other much more efficient control system methods are also readily available based on the *advanced process control* (APC) approach.

It is commonly acknowledged that the application APC in manufacturing can be profitable [7]. In recent years, there have been a number of developments in the area of APC to further confirm this. In [8], a real-time optimal control scheme for a chemical process is developed leading to significant improvements of production scheduling. The scheme is verified with real industrial data. On the other hand, in [9], the state-of-the-art of model predictive control is given. One of the claims in the conclusions states that one of the key issues to be addressed is the gap between the theory and practice of model predictive control. In [10], a comprehensive state-of-the-art of model predictive control in the application to electric machines and systems is provided. The authors again highlight the importance of MPC in industrial applications and predict that it will be more frequently applied to industrial problems in the future due to its favorable qualities. Additionally, in [11], a complete platform supporting the APC approach is presented, tailored to the requirements of Industry 4.0.

Similarly, in this paper, APC is considered as a tool that has the capacity to provide significant improvements in the context of industrial control and which would allow to achieve a smooth technology transfer from theory to practice, i.e., from academia to industry. Specifically, we address the problem of the practical implementation of advanced process control in the industrial space which is still largely dominated by PID control loops. Advanced process control techniques are considered to play a crucial role in achieving the goals of Industry 4.0 [12]. Naturally, many companies already offer APC solutions [13], yet their coverage area is still quite limited in the industry due to two main reasons: one of them has already been mentioned and is related to the widespread use of the PID controllers, and the other one is that the configuration and deployment of an APC solution is complicated and costly. This causes industrial stakeholders to only choose APC in cases where the application of the PID controller is not feasible at all. In this case, the return on investment and the desire to be able to automate a certain process loop can cover the APC implementation costs.

At the same time, the high costs of APC implementation leave out smaller players in the industry who cannot afford it. Furthermore, even with major players, smaller process control loops may also fall outside of the APC implementation scope, whereas Sustainable Development Goals (SDGs) stated by the UN [14] cannot be accomplished excluding smaller participants as their total amount is significant. Boiler houses, small- and medium-sized production facilities, data centers, commercial buildings, households, and a vast amount of other energy producers and consumers experience unnecessary losses due to inefficient energy usage. Leaving them out contradicts with SDG7.3 that assumes to "double the global rate of improvement in energy efficiency". Meanwhile, buildings consume up to 40% of total energy produced, more than industry or transport [15,16].

Furthermore, another key performance indicator nowadays is the carbon footprint. The reduction of carbon emissions is especially important in major contributing industrial entities including energy generation processes and chemical production [17]. Model predictive control has been shown to be very effective in reducing carbon emissions in the industry [18]. On the other hand, it is also crucial to optimize the electricity and heating or cooling consumption in buildings. Towards that end, in [19], a 400% CO_2 saving was

reported for a single building following the introduction of a model predictive control strategy which optimizes the electrical and thermal loads.

Therefore, accelerating the rate of adoption of APC—and more precisely MPC—in various industries is seen as a rather pressing issue toward achieving the above-stated goals. As previously mentioned, one of the key elements of accelerating this rate is the introduction of efficient KTT mechanisms [20].

Academic investigations of optimization are frequently limited to simulations only, and the actual deployment of the solutions rarely takes place, resulting in rather low technology readiness of the whole solution [21]. University–Industry Technology Transfer (UITT) is a complex process, as bringing theoretical results to the industry involves risks that might be unacceptable to many industrial partners. One of the most significant hurdles to technology innovation is the high cost of innovation implementation and the relatively low level of R&D expenditure in commercial firms. This problem was outlined many decades ago when UITT appeared as a separate research field [22,23] and UITT still is a challenging procedure that requires special efforts and administrative support [24–26]. On the other hand, several highly successful examples of UITT can also be found in the literature [27,28].

A scheme of APC solution development and transfer from academia to industry is presented in Figure 1. The data collected from industry are used to develop control and optimization algorithms (research phase) in academia using the academic tools reviewed in this paper. The proposed MPC application is used as an example to solve the problem of transferring the developed algorithm in software from academia to industry and its implementation in a production environment [29].

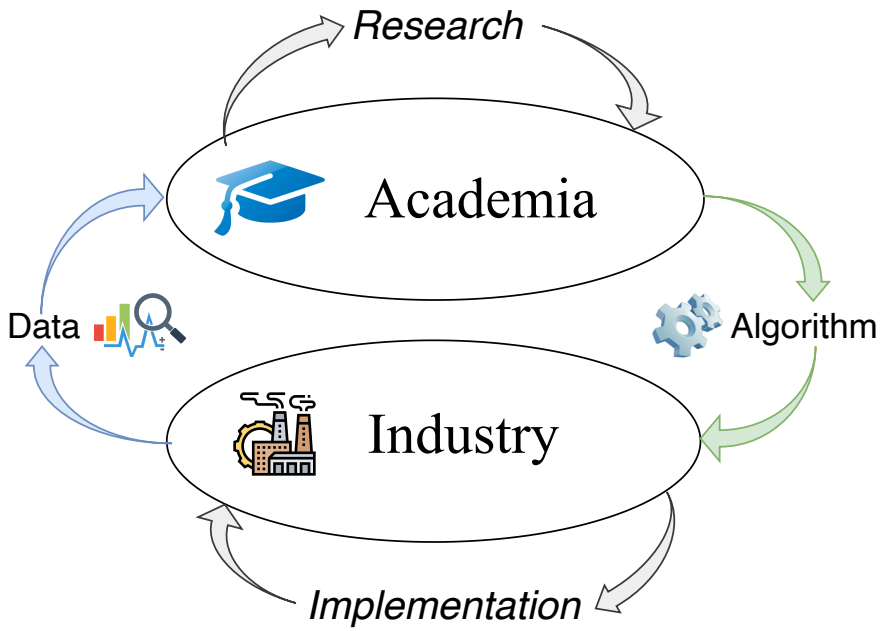


Figure 1. A typical cycle of advanced process control development: data are collected on the factory floor and shared with academic researchers; the data are processed, relevant research is conducted, and a control algorithm is designed and finally transferred back to the industrial plant where it is then implemented.

We now state the contribution and novelty of the work presented in this manuscript. First, the results reported in [30] are extended by developing the general software framework for deploying APC algorithms to production environments. This is the most significant innovation that underpins the technology transfer aspect since it enables for the implementation of any type of controller while maintaining a consistent and smooth user experience with the same user interface. The second part of the contribution is concerned with the study of the implementation process of the MPC algorithm specifically and shows that the novel framework is well suited for facilitating technology transfer from academia to industry. The solution is deployed to an industrial plant the proper operation of which is critical since it serves a city district. The end result is a more stable control of an industrial process output. In fact, the application of the MPC improved the performance of the industrial control loop almost threefold compared to the original PID-based control according to several performance metrics. Other achieved benefits are the reduced number of manual interactions between operators and process control, as well as lower fuel use. Furthermore, as a result of process control optimization, the CO_2 emission is also reduced, thus supporting the sustainable development goals.

The rest of the manuscript is structured as follows. The theoretical and practical aspects of developing and deploying the MPC algorithm are described in Section 2. The essentials concerning the development of the proposed solution are provided in Section 3. Section 4 describes practical aspects of APC technology transfer. Section 5 is dedicated to the application to industrial implementation. Section 5 is dedicated to the implementation of the developed solution to an industrial plant. Section 6 concludes paper with results and future development plans.

2. Model Predictive Control: Theoretical and Practical Aspects

In this work, model predictive control—a well studied and powerful control method—is considered as the advanced control approach. It is, however, also well known that it is far more difficult to transfer it to a production environment compared to, e.g., a PID controller. In the following section, the MPC method is briefly recalled.

2.1. MPC Theory

In MPC, the main aim is to minimize a cost function defined as [31]:

$$V(k) = \|Z(k) - T(k)\|_Q^2 + \|\triangle U(k)\|_R^2, \tag{1}$$

where $Z(k)$ is the outputs' prediction vector within prediction horizon H_p, $T(k)$ is the set points' trajectory within H_p, and $\triangle U(k)$ is the vector of process input moves (changes) within the control horizon H_c.

After simple algebraic manipulation, the cost function can be rewritten as:

$$V(k) = \text{const} - \triangle U(k)^\mathsf{T} G + \triangle U(k)^\mathsf{T} H \triangle U(k), \tag{2}$$

where $\Delta U(k)$ is the only unknown that should be found to minimize $V(k)$. The expressions for G and H are known and are obtained from (1). The constant term const is eventually of no importance as it disappears during the differentiation of $V(k)$ as part of the minimization procedure.

In case of MPC, it is possible to set constraints on input moves $\triangle u$, input value u, and output z as $\triangle u_{\min} \leq \triangle u(k) \leq \triangle u_{\max}$, $u_{\min} \leq u(k) \leq u_{\max}$ and $z_{\min} \leq z(k) \leq z_{\max}$ at any moment, where min and max are the lower and upper bounds of the corresponding variable. As the controller produces optimal input moves, then we are interested to express all equations through $\triangle u$.

The combined inequality has the form:

$$\begin{bmatrix} \Lambda \\ \Phi \\ \Gamma \end{bmatrix} \triangle U(k) \leq \begin{bmatrix} \lambda \\ \phi(u(k-1)) \\ \gamma(x(k), u(k-1)) \end{bmatrix}, \tag{3}$$

where Λ, Φ, Γ are known polynomials and λ, ϕ, γ are known functions, $u(k-1)$ is the process input vector at the previous time sample, $x(k)$ is the model state vector at the current time sample, and $\triangle U(k)$ is the vector of future control moves known at the time moment k, see [31] for details.

Finally, taking into account (2), we get a problem in the quadratic programming form:

$$\min_{\theta} \frac{1}{2} \theta^{\mathsf{T}} H \theta + h\theta \tag{4}$$

subject to $\Omega\theta \leq \omega$.

This type of problems can be solved using standard methods. Most popular are the active set and interior point method. In this work, the interior point method was used as it was developed later then the active set method and is known to require less computational power.

2.2. Interior Point Method for MPC

The interior point or barrier function method is used in the present MPC formulation. It is deeply studied in [31–38] and a brief overview is given next.

The barrier function method brings the problem in Equation (2)—excluding the constant component—to

$$V(k) = \triangle U(k)^{\mathsf{T}} H \triangle U(k) - \triangle U(k)^{\mathsf{T}} G \tag{5}$$

subject to constraints (3) to the form with no explicit constraints. The modified cost function is given by:

$$V(k) = \triangle U(k)^{\mathsf{T}} H \triangle U(k) - \triangle U(k)^{\mathsf{T}} G + \mu B(\triangle U(k)), \tag{6}$$

where

$$B(\triangle U(k)) = -\sum_{i}^{H_u m} \left(\ln(\triangle U_{\max_i} - \triangle U_i(k)) + \ln\left(\triangle U_i(k) - \triangle U_{\min_i}\right) \right). \tag{7}$$

In (7), B is a logarithmic barrier function, i refers to the ith element of vectors $\triangle U_{\max}$, $\triangle U_{\min}$, and $\triangle U(k)$. There are also other types of barrier functions, but the logarithmic function is the most convenient to use with standard optimization techniques. Function (6) is smooth, so it can be solved with a suitable formulation of the Newton's method. As $\mu \to 0$, the solution of (6) tends to the solution of (5), see [33].

With the application of Newton's method, the update of $\triangle U$ results in

$$\triangle U = \triangle U - \alpha \mathcal{H}^{-1} g, \tag{8}$$

where g is the gradient vector of the cost function, $\mathcal{H}$ is the Hessian of the cost function and α is a scalar to ensure reduction of the cost function. Furthermore,

$$g(i) = H(i,:)\triangle U(k) - G(i) + \mu \left(\frac{1}{\triangle U_{\max_i} - \triangle U_i(k)} - \frac{1}{\triangle U_i(k) - \triangle U_{\min_i}} \right), \tag{9}$$

where i refers to the ith element of gradient vector g and vector G, $H(i,:)$ refers to the ith row of the matrix H. The Hessian matrix $\mathcal{H}$ is

$$\mathcal{H} = H + \mu D, \tag{10}$$

where D is a diagonal matrix with the following elements on the diagonal:

$$D(i,i) = \frac{1}{\left(\triangle U_{\max_i} - \triangle U_i(k)\right)^2} + \frac{1}{\left(\triangle U_i(k) - \triangle U_{\min_i}\right)^2}. \tag{11}$$

Gradient and Hessian expressions are only true if $\triangle U(k)$ is feasible. That requires the starting point for the optimization to be within the limits, which can be achieved, if, for example, only input constraints are used:

$$\triangle u(k) = \frac{u_{\max} + u_{\min}}{2} - u(k-1), \tag{12}$$

and

$$\triangle u(k+i) = 0, \quad i = 1, \ldots, H_u - 1. \tag{13}$$

In case of output constraints, a feasible starting point may not exist if these constraints contradict each other. In these circumstances, output constraints should be softened, meaning they should be shifted enough in the search space until a feasible point is reached. The reason for softening output restrictions is due to the physical limitations of input constraints, which makes it impossible to soften these restrictions [31].

For the implementation of the interior point method, one also needs a scaling factor $\nu \in (0,1)$ to prevent violations of the constraints, a positive integer K to define the number of iteration steps, a barrier scaling factor $\mu > 0$, and a barrier-scaling-factor weighting factor ζ. The closer the value of ν is to 1, the closer $\triangle u(k)$ goes to the limit. The scaling factor μ is required to prevent a jump out of the feasible region at the beginning of the algorithm execution. After the initial value is assigned (e.g., to 10), it decreases in each iteration of the algorithm, thus leading to the convergence of the problems defined in (5) and (6). Note that the inversion of the Hessian matrix $\mathcal{H}$ is not needed as $\rho = \mathcal{H}^{-1}g$ can be calculated with a QR-decomposition using the Householder algorithm [39].

If the optimal solution for the unconstrained MPC problem is located beyond the constraints, then in each iteration of the interior point method, one should decrease the value of μ and approach the value of the constraint without crossing it.

The algorithm that is ultimately used in the implementation of the MPC algorithm in the present work is depicted in Figure 2. First, the feasible initial values of the control moves are found using (12) and (13). Then, the iterative process to ensure the fulfillment of the constraints (j-steps) is started: the gradient vector g and Hessian matrix $\mathcal{H}$ of the cost function (5) are found and the next update of the control moves is calculated as ρ with Newton's method. Next, all elements of ρ (i-steps) are tested for the control moves vector $\triangle U$ to be within the constraints. If any element of ρ violates the constraints, then its value is set to bring $\triangle U$ back into the region where the constraints are fulfilled. Meanwhile, the α parameter with $\alpha \in [0,1]$ is introduced to get a better estimation of the $\triangle U$ update. After this, the control moves $\triangle U$ and the barrier weight μ are updated, and the algorithm is repeated until K iterations are reached.

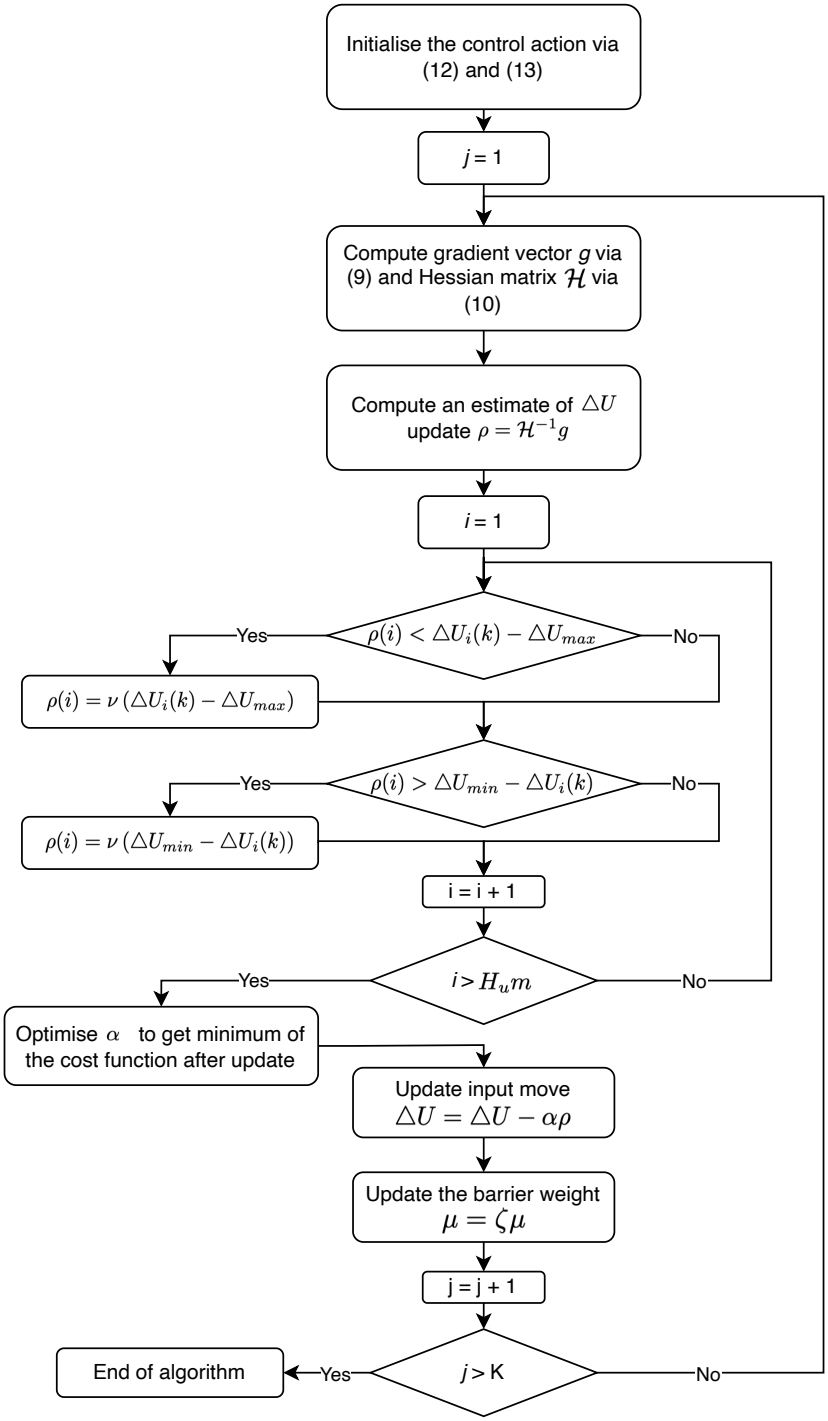


Figure 2. An algorithm for the implementation of constrained model predictive control.

3. Application Development Tools and Structure

In this section, a summary is provided concerning the actual implementation of the above-mentioned MPC algorithm as part of a flexible intelligent control platform, for which the MPC approach is the preferred method, whereas the platform is capable of hosting any conceivable control method as well.

For building the application, Java programming language was selected because of its cross-platform support and wide adoption. Numerous IDEs (Integrated Development Environments) are available at no charge and provide sufficient convenience for the development of complex applications. We considered Eclipse [40] and IntelliJ [41] IDEs, but finally selected IntelliJ due to its greater coding assistance functionality.

A further advantage of using Java is the availability of a large number of freely available libraries that lets one focus on the task at hand without delving deeper into low level operations (such as communication protocols and methods, database access, etc.) that can be implemented from available, oftentimes unit-tested code. Java software dependencies were managed using the project management tool Maven [42]. Maven is another free software package that is maintained by Apache Software Foundation [43] to help manage and arrange the development of software components. The Maven configuration is done in the POM (project object model) XML file, which is easily filled with the proper settings with the help of IntelliJ's project template set and autocomplete features.

The Spring Framework [44] was chosen as the basis for implementing the backend of the application. It provides mechanisms to build complex applications in a relatively simple manner [45]. Inversion of Control, Aspect Oriented Programming, Data Access framework, Context management are some examples of the features of Spring. It is also important that the framework is helpful for building and running modular software since the developed software platform is intended to be flexible and modular. Spring helps to avoid implementing a lot of low level operations, e.g., database access, as these operations are automated and require just a few lines of code to be implemented. Context management allows us to create independent software modules that are automatically combined on the fly into one application.

While Java was considered reasonable for the backend functionality of the application, we decided to use the Angular framework with Typescript for the user interface. The choice of creating a Web UI was dictated by modern day requirements and expectations, but Java frameworks for this type of solution are either premium or do not have adequate functionality. At the same time, Angular provides all of the necessary options to build modern web interfaces.

The main functions that the application has to cover include:

1. Maintain a list of data points—measurements from the actual industrial process—the operator is going to be working with;
2. Configure and run communication via Modbus/TCP protocol with the process control system;
3. Configure and run the control algorithm;
4. Save the configuration to the database whenever it is modified and load it from the database on start-up.

These functions are carried out in the backend of the application, while the frontend works from an internet browser and communicates with the backend to load or update the configuration. The structure of the application is shown in Figure 3. The web application can be considered a software client in relation to the backend. It uses the HTTP protocol to fetch values in JSON format from the backend and displays these in a suitable way in the operator's web browser. All presented data can be edited and saved.

The backend contains an algorithm module where any algorithm can be implemented (in the case of this work, it is MPC). It takes input values from the data points list, makes computations, and saves output values as other data points in the same list. A communication thread runs separately. It reads input values from the process control system and saves them to configured data points, then reads output values from other data points and sends them to the control system.

The following subsections describe the application's main functions in more detail.

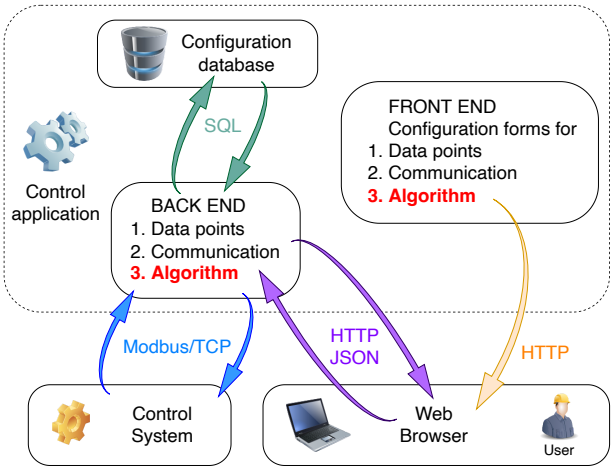


Figure 3. The structure of the developed advanced process control framework. The framework consists of the backend, which is responsible for communicating with a configuration database and the actual industrial control system, and the frontend, which is how users interact with the backend via an intuitive HMI running in the user's browser.

3.1. Data Points

The backend allocates memory for a list of data points, which will be used as inputs and outputs for the algorithm. There are currently three types of data in the list: *double*, *integer*, and *boolean*. The data points list is displayed as a table in the web browser. Each row has Edit/Save and Remove buttons.

3.2. Communication

Communication is implemented as Modbus/TCP master using the JLibModbus Java library [46]. This library is available from Maven repositories; so, no extra import efforts are needed except configuring it in a Maven POM file. The choice of using Modbus stems from its relative popularity in industrial communication.

For Modbus master, we need to configure read and written data points of two types: coils (bit) and registers (words or 16 bit integers). It is possible to use two registers to code a 32 bit single precision real value (IEEE-754). It is sent as two registers and must be decoded on the other communication side as it is not a standard Modbus functionality.

Implementation of the user interface for protocol configuration is more involved. It consists of three levels. On the first level, we have a list of Modbus communication lines as we suppose possibility to use the application for communication with many systems simultaneously. Each line has an Edit button that opens the second level of the protocol line configuration parameters. This level includes also data point reads and writes each with its own Edit button that opens the third level with a list of Modbus addresses of each data item to be sent or received. Here, Modbus addresses are associated with data points list items. Addresses' values read from the Modbus slave are saved to value fields of the configured data point. Values of written addresses are acquired from the value field of the configured data point and sent to the slave. A process control system has to be implemented as a Modbus slave in this configuration.

3.3. Web HMI

Independently of how the high level of automation is achieved in any production process, the final decision of the way to proceed is in the responsibility of a human. Any automation system has operators who follow the process and interact with corrective actions, if something goes wrong. The processes become bigger and more complex with time, so the amount of data to be processed becomes huge; hence, the main trends in building a modern human–machine interface (HMI) move toward efficient data presentation

without overloading the operator with a stream of data that is very difficult to process or to understand [47,48].

In the case considered in this work, a limited amount of data is used in the application. We need to provide visualization of the communication protocol settings, required process parameter values, settings of the algorithm, and the outputs produced by the algorithm that are sent to the process control system. The Web HMI is used in the scope of this work as it has already become a standard HMI solution in many industrial use cases.

As was mentioned earlier, the Angular framework was used to develop the HMI—an open-source framework developed->supported by Google. Angular creates a stand-alone frontend that is compatible with any backend application supporting HTTP communication. It is possible to use any programming language for backend development including Java.

With Angular, one can choose from a variety of UI components that may display any type of information in the browser. For data sharing, Angular employs the widely recognized JSON format. As a result, the framework enabled the creation of a user interface for the MPC application that was both convenient and easily configurable. Parameters of the application are displayed as text. Many of them have matrix form which is easy to present as tables with suitable border formatting. For convenient modifications, matrices are converted into text boxes with values in Matlab format, e.g., [1 2 3; 4 5 6] for

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}. \tag{14}$$

The resulted User Interface example with MPC diagnostic page is presented in Figure 4.

Figure 4. The User Interface of the developed MPC application as used on the factory floor.

4. Practical Aspects of the Implementation and Industrial Integration

Certain development steps must be performed for the MPC implementation to become an efficient practical solution at the final stage when it is deployed to control the industrial process [49,50]. Moreover, since the goal of the present contribution is to also provide technology transfer for the MPC control method, the implementation should be in line with a coherent TT model, see [20].

MPC implementation has five steps that relate to the general TT model as shown in Table 1. Accordingly adopted TT scheme is presented in Figure 5.

The process is thoroughly examined in the first step, and it is determined whether MPC is applicable in the selected situation. Often, it is sufficient to tune existing control loops to achieve satisfactory control performance. If this is not successful, however, then one needs to determine whether the performance bottleneck lies with the existing equipment. Manual process control by an experienced operator could be used in practice. If better performance is obtained with manual control, it is obvious that automatic control has growth potential. Unfortunately, empirically revealing potential opportunities is not always possible, hence a theoretical approach should also be used. In this case, we should keep in mind that MPC is more beneficial for processes with dead time, constraints, and multiple inputs and/or multiple outputs [50]. This step includes formulating the problem statement, discovering the state of the art, and the development of a candidate solution.

The next stage of general TT assumes validation in academia, which means in case of MPC implementation three basic steps: collection of experimental data from the plant under study; data-driven model development based on the collected data; and MPC design using the obtained model. Process data can often be collected relatively easily as many processes are equipped with *Historian databases*. Process inputs should be changed in plant testing to cause process outputs to fluctuate around the main operating point for the goal of collecting dynamic process data [51]. Model identification and MPC design do not present considerable challenges for a seasoned researcher in the academia since in the area of systems and control, there are several tools available for researchers, such as Matlab with relevant extensions including the System Identification toolbox and MPC toolbox. Also free identification solutions exist such as Python-based SIPPY [52]. However, challenges arise with the implementation of the real process since moving the controller from a Matlab simulation to the real process control system is not straightforward [53,54]. There exist Python-based MPC solutions, but these appeared recently and are in the beginning of their career mainly presenting laboratory test results [55,56].

Table 1. The relation of the technology transfer model [20] to the MPC implementation steps as viewed from the perspective of designing and deploying process control.

Steps	General TT Model	MPC Implementation	Steps
1	Problem/issue	Process analysis and	1
2	Study state of the art	preliminary MPC	
3	Candidate solution	design	
4	Validation in academia	Plant testing	2
		Model identification	3
		Controller design	4
5	Static validation	Commissioning	5
6	Dynamic validation	and	
7	Release solution	training	

The final step of the MPC implementation requires the integration of the developed solution into the control system of a real process, whether this means a programmable logic controller (PLC) or a distributed control system (DCS) implementation. Some DCS systems may already include MPC implementations [50], but this is always a commercial solution with a significant associated cost that is often not feasible for smaller process optimization. Migration from step four to step five with academic or open-source software can be difficult because no ready-made appropriate interfaces are available. The application framework

proposed in this manuscript solves the problem of transferring the designed MPC into industrial implementation.

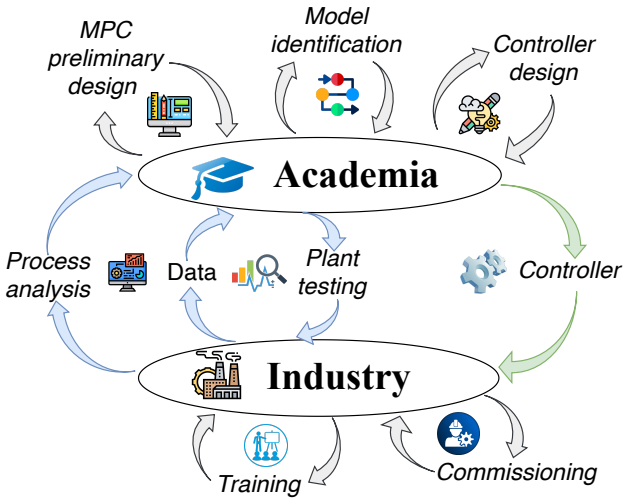


Figure 5. The process diagram of the proposed MPC transfer from academia to industry.

5. Industrial Implementation

5.1. Process

In this paper, we focus on the problem of transferring APC solutions from the research field to the control of an energy generation process in industry. Here, we consider a water boiler that is a part of a bigger combined heat and power (CHP) plant. The primary goal of the CHP plant is to produce heat for nearby cities.

Originally installed in 1978, the boiler produced 100 kcal/h (116.3 MW) of heat power. The specific model of the boiler is KVGM-100 [57]. Several major investments were made to renovate the boiler infrastructure during its lifetime; so, it is now equipped with modern measurement and control devices and is connected to DCS. All the control applications are implemented in the DCS on software level. Applications can be created using predefined or programmable function blocks and downloaded to the DCS without interrupting the process control. Still, control methods presently used in boiler control are quite conservative and are based on the PI control algorithm.

The output water temperature of the boiler is a controlled variable in the boiler's main control loop. The fuel combustion process, the transfer of heat power from furnace to water and hot water flow to the boiler output are relatively slow due to the size of the equipment, so there is a significant delay between the flow of gas into the furnace (manipulated variable), and the measurement of the water temperature at the boiler output (controlled variable). Related process diagram is shown in Figure 6. Since the PI controller uses output errors for control, the delay affects its performance. Therefore, the integration time is set longer than the measurement delay to prevent permanent overshoot of the manipulated variable. Specifically, the integration time was set to 240 s. As the ideal PI control algorithm $c_{out} = K_p(e + \frac{1}{T_i}\int e\,dt)$ is implemented in the related function block, then the real integration time is almost 270 s ($K_p = 0.9$). This causes the controller to react slowly to changes in set points and disturbances. Due to the above considerations, retuning the PID controller is not a feasible option. An opportunity to apply the MPC control algorithm arises naturally according to prior discussion.

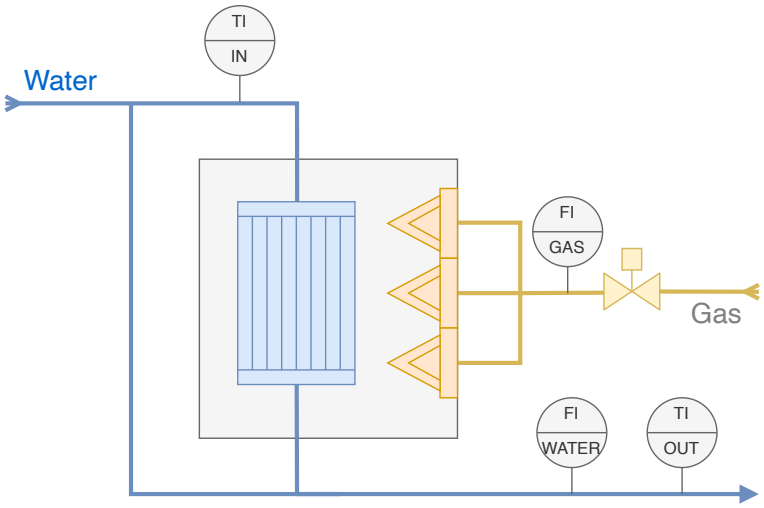


Figure 6. The process diagram of the considered industrial plant—the KVGM-100 boiler in the configuration for district heating whereby the passing water is heated with gas.

5.2. MPC Implementation

According to the implementation procedure described in Section 4, a process investigation and preliminary MPC design was performed. From operators' and automation engineers' interviews, it was learned that the control loop under investigation is frequently set to manual mode and corrective actions by the operator are applied to get better performance. The reason for this is mostly the process dead time. Depending on boiler load, the delay from the system input to output can reach 4 to 6 min. This is a good prerequisite to assume control performance improvement with use of MPC.

The next step is to perform a plant test to collect time series data for model identification. As the process is not controlled in a stable manner, the input/output variations are sufficient for model identification without any special test signals. Historian data are collected to the database all the time. We acquired these data with a sampling time of 10 s. After analyzing the process, it was decided that a 1 min sample time is sufficient for model identification and later process control, as over 5 prediction steps cover the average 5 min process delay and mitigate its effect.

The process variables that mainly affect plant temperature output are: water inlet temperature, total water flow, and gas flow. So, the candidate model structure was selected to have three inputs and one output. Both total water flow and inlet temperature depend on the district heat network load and cannot be manipulated. Thus, two of three inputs are measured disturbances and only one is a manipulated variable—gas flow.

After the data were collected and preprocessed, several state-space models were identified in the Matlab Identification Toolbox using the prediction error minimization (PEM) identification method. These models were used to make initial MPC design in the Matlab MPC Toolbox. An MPC was created with a model of better fit while other models were used to simulate the process in the Matlab/Simulink environment in various scenarios. After achieving satisfactory results in Matlab, more realistic offline tests were performed with the acquired controller.

5.3. Offline Simulation

Offline simulation is required to test MPC functionality without affecting the real process [58]. This ensures that MPC computes adequate manipulated variable values to bring the controlled variables to the desired trajectories.

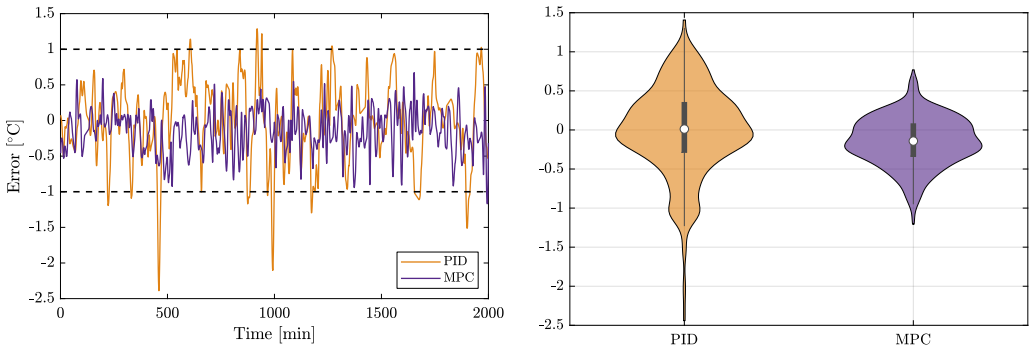
In our case, the first offline test was performed in a computer with a mathematical model of a process different from the model used in MPC. Real DCS software was used for

process simulation. A state-space simulation model was implemented as a Java functional block. Simulation time was increased 15 times compared to real time to make testing faster and more convenient; thus, the simulation sampling time was 4 s instead of 1 min. MPC parameters were transferred from Matlab to our MPC application. Communication between DCS and MPC application was configured using the Modbus/TCP protocol. On this stage, MPC was pre-tuned in the same software that is used for the real process control.

After the MPC functionality was validated, the next offline test was performed on a real process without passing the controlled variables values to the process control. The functionality of the controller was observed during several hours. The controller showed adequate reaction to all process changes, so the process owner accepted the controller for the online implementation.

5.4. Online Implementation

The control performance of the plant has improved significantly following the implementation of the MPC control. Figure 7 shows a comparison of PID and MPC control performance. A temperature deviation of one degree is acceptable in plant output. The PID controller, however, fails to achieve this target, while MPC keeps the temperature close to the set point as required by the specifications. This can be easily seen in the time series shown on the left side of the figure.



(**a**) (**b**)

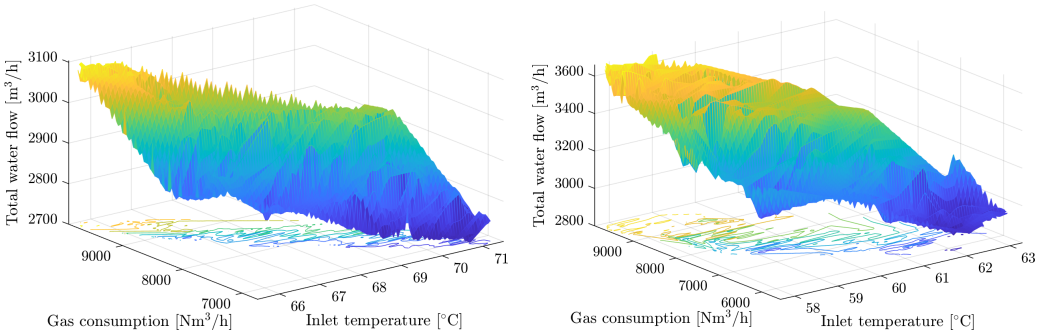
Figure 7. Performance comparison of PID and MPC-based control loops. (**a**) Timeseries showing the dynamics of set-point tracking error; the maximum tolerable deviation from the set point is ±1 °C, the closer the values are to zero, the better. The PID-based control loop frequently escapes the allowed error band. (**b**) Violin plots showing the distributions of deviations from the set point. It can be clearly seen that the MPC-based control loop results in a set point deviation that is more tightly packed around the origin, thus demonstrating the benefit of this control configuration compared to the PID control loop. (**a**) Timeseries showing the deviations from the set point for the compared control algorithms. (**b**) Violin plots for the set point deviations' distributions for the compared control methods.

Furthermore, the error resulting from PID control has a wider distribution than that of the error stemming from the use of MPC. Its variation is far beyond the allowed limits as can be seen from the violin plot on the right of Figure 7. The main parameters for evaluating the controller performance are shown in Table 2. The MPC-driven control loop has nearly three times lower mean square error and sum square error than the PID one. The median value of its error is also lower than that of PID, which is not a global characteristic, but an advantage in this experiment, since it results in lower gas consumption in case of MPC. This will be discussed later in more detail.

Table 2. Quantitative performance comparison of the PID- and MPC-based control loops based on set-point deviation.

	Mean Square	Sum of Squares	Median
PID	0.3173	634.5972	0.0114
MPC	0.1199	239.7205	−0.1421

Furthermore, we have to discuss operating points of the process at which we compare PID and MPC loop performance. The points are not the same as there are certain variations in the set point due to changes in hot water demand and certain ambient conditions, mainly weather. It is not enough to compare performance directly, we need to discuss the comparison environment as well. For that reason, timeseries were selected from source data that demonstrate the performance of the control loop in operating areas that are as close to each other as possible. The operating area is determined by the process inputs and the set-point value selected for the control. The process inputs relationship is depicted in Figure 8, in which we can see that the selected operating areas intersect.



(**a**) (**b**)

Figure 8. Surfaces depicting the operating areas of MPC (left) and PID (right) control loops. Each axis represents an input to the system: the total water flow, gas consumption, and inlet temperature. This visualization aims to demonstrate that the operating areas for the MPC and PID control loops are similar and thus yield a fair comparison of the performance of both control loops. (**a**) A surface depicting the operating area of the MPC control loop. (**b**) A surface depicting the operating area of the PID control loop.

The numerical ranges of operating areas are presented in Table 3. As can be seen, the set point is different for compared cases, but according to the end user, the MPC performance is clearly better at every operating point, since the PID could never reach the same result with low output variation. It was not possible to test the PID controller and MPC under identical settings because the tests were conducted at different times of the year, when the operational environment also differed, but efforts were exhibited in the present work to achieve a fair comparison of the results.

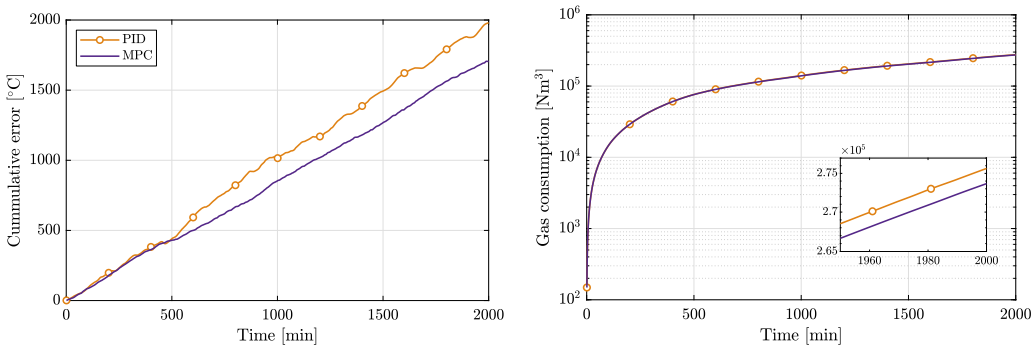
A further improvement in control performance would be achieved by reducing actuator hysteresis, which would result in the actuator moving more frequently, resulting in faster valve wear and would require more frequent maintenance cycles. Thus, the acquired performance is a compromise with the mechanical condition of the control equipment. The implementation of the new control loop caused operators to stop interacting with it manually. Overall, the control performance was adequate to avoid quality penalties in the future. In the end, the industrial stakeholders were satisfied with the obtained results.

Table 3. A summary of the operating areas for the MPC- and PID-based control loops. The operating areas depend on the set point, and three inputs to the system. The control loops are compared according to the operating areas specified in the table.

	Inlet Temp., °C	Water Flow, m^3/h	Gas Flow, Nm^3/h	Output Temp. SP, °C
PID	58–64	3000–3600	6000–10,000	78
MPC	66–72	2800–3200	7000–10,000	90

Finally, it is important to evaluate gas consumption in two scenarios. As indicated earlier, it cannot be compared directly as the PID controller and MPC were tested under slightly different conditions (different set points, different water flows); however, cumulative output error provides a general estimate of gas consumption. Figure 9a illustrates the cumulative error of the plant output in relation to the lower limit (−1) of acceptable error value. The graph demonstrates that the cumulative error of the plant output temperature is higher with PID control. However, from a physics perspective, MPC would provide a reduction in gas consumption since it avoids heating water to unnecessarily high temperatures.

After normalizing the PID gas consumption to the MPC operating area by dividing it with in/out temperature difference and water flow of the PID operating area and multiplying by the same parameters of the MPC operating area, we get the gas consumption comparison in Figure 9b. The consumption in case of the PID controller is 1977 Nm^3 higher after 2000 min of operation (here, Nm^3 means the quantity of Natural Gas which, when absolutely dry, at a temperature of 0 °C and at an absolute pressure of 101,325 bar, occupies the volume of one cubic meter).



(**a**) (**b**)

Figure 9. A comparison of gas consumption versus the cumulative control error for both MPC- and PID-based control loops. The gas consumption is reduced in case of the MPC controller; this becomes evident from the enlarged part of the right-hand plot. (**a**) Comparison of the cumulative control error (lower is better). (**b**) Comparison of gas consumption (lower is better).

6. Conclusions and Discussion

In the present manuscript, the complete process of technology transfer from development to production was presented for an advanced process control method—the model predictive control method. In the proposed application [59], the linear MPC algorithm is implemented, and complemented with a communication and HMI infrastructure, which constitutes a complete framework for various algorithms to be developed and deployed in the industry. The application can be extended in the future to handle nonlinear models and nonlinear control or implement arbitrary new algorithms. The complete planned application structure from [30] is presented in Figure 10 with the current contribution highlighted. Intelligent Control System CORE and Model Predictive Control functionality have been implemented in the scope of the current work, the other algorithms will become available

in the future. The crucial point is that all of those advanced control algorithms will support a smooth technology transfer due to the overall architecture and implementation of the framework presented in this paper.

The technology transfer resulted in a significant improvement of control system performance in case of a combined heat and power plant. Namely, MPC was applied to a gas-fueled boiler. As a result, the fluctuation of the temperature of the outgoing water has been minimized which also led to the reduction of wasted energy and, as a consequence, to minimal CO_2 emissions.

Some items for future research and development are outlined next. First, the communication protocol is Modbus/TCP master-only, which limits communication options. The Modbus/TCP slave functionality will be added, as well as OPC UA, which is becoming increasingly popular in industrial systems. An improvement that would increase application usability is embedded model identification. This would require significant effort for application development, integration of time series database (e.g., InfluxDB [60] or TimescaleDB [61]), and implementation of cutting-edge identification algorithms. This would allow the application to be self-sufficient in the context of the advanced control of real industrial processes. It will also provide the necessary components to implement other control algorithms as mentioned previously.

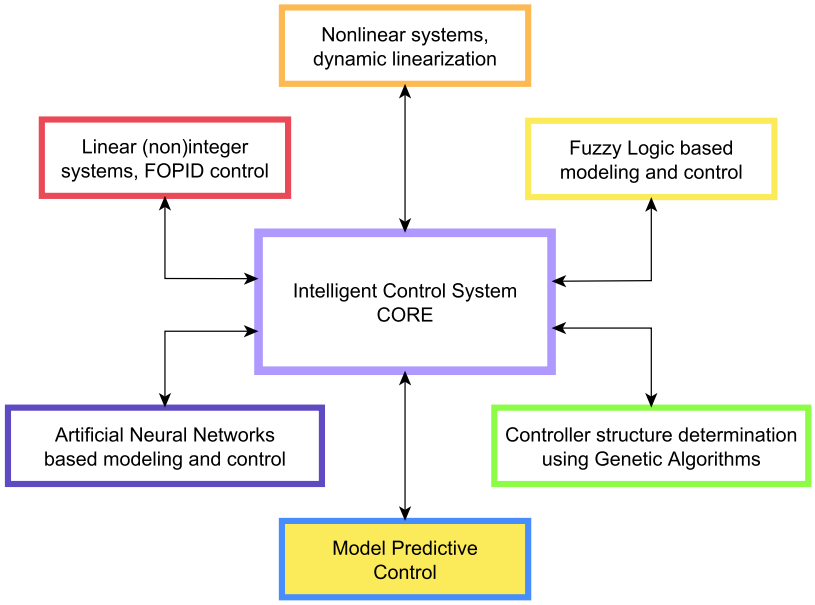


Figure 10. The overall framework developed in the scope of the present effort. The MPC module discussed in this paper is highlighted.

Author Contributions: Conceptualization, V.V., E.P., A.T. and K.V.; methodology, A.T. and V.V.; software, V.V.; validation, V.V.; formal analysis, A.T. and J.B.; investigation, V.V.; resources, K.V., A.T. and E.P.; data curation, V.V.; writing—original draft preparation, V.V.; writing—review and editing, A.T. and J.B.; visualization, A.T. and J.B.; supervision, E.P. and A.T.; project administration, E.P.; funding acquisition, E.P. All authors have read and agreed to the published version of the manuscript.

Funding: The work was partly supported by the Estonian Research Council through the grants PRG658 and PRG1463.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

Abbreviations

The following abbreviations are used in this manuscript:

PID	Proportional integral derivative
APC	Advanced process control
MPC	Model Predictive Control
DCS	Distributed Control System
PLC	Programmable Logic Controller
KTT	Knowledge and Technology Transfer
UITT	University–Industry Technology Transfer
SDG	Sustainable Development Goals
UN	United Nations
POM	Project Object Model
HMI	Human–Machine Interface
JSON	JavaScript Object Notation
TCP	Transmission Control Protocol
HTTP	HyperText Transfer Protocol
CHP	Combined Heat and Power
OPC UA	Open Platform Communications Unified Architecture

References

1. Lamnabhi-Lagarrigue, F.; Annaswamy, A.; Engell, S.; Isaksson, A.; Khargonekar, P.; Murray, R.M.; Nijmeijer, H.; Samad, T.; Tilbury, D.; den Hof, P.V. Systems & Control for the future of humanity, research agenda: Current and future roles, impact and grand challenges. *Annu. Rev. Control* **2017**, *43*, 1–64. [CrossRef]
2. O'Dwyer, A. *Handbook of PI and PID Controller Tuning Rules*; Imperial College Press: London, UK, 2009.
3. Tepljakov, A.; Alagoz, B.B.; Yeroglu, C.; Gonzalez, E.A.; Hosseinnia, S.H.; Petlenkov, E.; Ates, A.; Cech, M. Towards industrialization of FOPID controllers: A survey on milestones of fractional-order control and pathways for future developments. *IEEE Access* **2021**, *9*, 21016–21042. [CrossRef]
4. Rojko, A. Industry 4.0 concept: Background and overview. *Int. J. Interact. Mob. Technol.* **2017**, *11*, 77. [CrossRef]
5. Csalódi, R.; Süle, Z.; Jaskó, S.; Holczinger, T.; Abonyi, J. Industry 4.0-driven development of optimization algorithms: A systematic overview. *Complexity* **2021**, *2021*, 6621235. [CrossRef]
6. Bacci di Capaci, R.; Scali, C. A cloud-based monitoring system for performance assessment of industrial plants. *Ind. Eng. Chem. Res.* **2020**, *59*, 2341–2352. [CrossRef]
7. Cutler, C.; Perry, R. Real time optimization with multivariable control is required to maximize profits. *Comput. Chem. Eng.* **1983**, *7*, 663–667. [CrossRef]
8. Vaccari, M.; di Capaci, R.B.; Brunazzi, E.; Tognotti, L.; Pierno, P.; Vagheggi, R.; Pannocchia, G. Optimally managing chemical plant operations: An example oriented by Industry 4.0 paradigms. *Ind. Eng. Chem. Res.* **2021**, *60*, 7853–7867. [CrossRef]
9. Han, J.; Hu, Y.; Dian, S. The state-of-the-art of model predictive control in recent years. *IOP Conf. Ser. Mater. Sci. Eng.* **2018**, *428*, 012035. [CrossRef]
10. Elmorshedy, M.F.; Xu, W.; El-Sousy, F.F.M.; Islam, M.R.; Ahmed, A.A. Recent achievements in model predictive control techniques for industrial motor: A comprehensive state-of-the-art. *IEEE Access* **2021**, *9*, 58170–58191. [CrossRef]
11. Badii, C.; Bellini, P.; Cenni, D.; Mitolo, N.; Nesi, P.; Pantaleo, G.; Soderi, M. Industry 4.0 synoptics controlled by IoT applications in Node-RED. In Proceedings of the 2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics), Rhodes, Greece, 2–6 November 2020. [CrossRef]
12. Maxim, A.; Copot, D.; Copot, C.; Ionescu, C.M. The 5W's for control as part of Industry 4.0: Why, what, where, who, and when—PID and MPC control perspective. *Inventions* **2019**, *4*, 10. [CrossRef]
13. Qin, S.J.; Badgwell, T.A. A survey of industrial model predictive control technology. *Control Eng. Pract.* **2003**, *11*, 733–764. [CrossRef]
14. UN. THE 17 GOALS, 2015. Available online: https://sdgs.un.org/goals/ (accessed on 10 May 2022).
15. Pérez-Lombard, L.; Ortiz, J.; Pout, C. A review on buildings energy consumption information. *Energy Build.* **2008**, *40*, 394–398. [CrossRef]
16. Gruber, J.K.; Prodanovic, M.; Alonso, R. Estimation and analysis of building energy demand and supply costs. *Energy Procedia* **2015**, *83*, 216–225. [CrossRef]
17. Wang, J.; Wei, S.; Wang, Q.; Sundén, B. Transient numerical modeling and model predictive control of an industrial-scale steam methane reforming reactor. *Int. J. Hydrog. Energy* **2021**, *46*, 15241–15256. [CrossRef]
18. He, Z.; Sahraei, M.H.; Ricardez-Sandoval, L.A. Flexible operation and simultaneous scheduling and control of a CO_2 capture plant using model predictive control. *Int. J. Greenh. Gas Control* **2016**, *48*, 300–311. [CrossRef]

19. Bolzoni, A.; Parisio, A.; Todd, R.; Forsyth, A. Model Predictive Control for optimizing the flexibility of sustainable energy assets: An experimental case study. *Int. J. Electr. Power Energy Syst.* **2021**, *129*, 106822. [CrossRef]
20. Gorschek, T.; Garre, P.; Larsson, S.; Wohlin, C. A model for technology transfer in practice. *IEEE Softw.* **2006**, *23*, 88–95. [CrossRef]
21. Wohlin, C. Empirical software engineering research with industry: Top 10 challenges. In Proceedings of the 2013 1st International Workshop on Conducting Empirical Studies in Industry (CESI), San Francisco, CA, USA, 20–20 May 2013. [CrossRef]
22. Brust, M.F. Technology transfer and the university. *J. Appl. Bus. Res. (JABR)* **1991**, *7*, 1. [CrossRef]
23. Matkin, G. *Technology Transfer and the University*; National University Continuing Education Association American Council on Education: New York, NY, USA, 1990.
24. Wynn, M.G. Technology transfer projects in the UK. *Int. J. Knowl. Manag.* **2018**, *14*, 52–72. [CrossRef]
25. Good, M.; Knockaert, M.; Soppe, B.; Wright, M. The technology transfer ecosystem in academia. An organizational design perspective. *Technovation* **2019**, *82–83*, 35–50. [CrossRef]
26. Daniel, A.D.; Alves, L. University-industry technology transfer: The commercialization of university's patents. *Knowl. Manag. Res. Pract.* **2019**, *18*, 276–296. [CrossRef]
27. Lee, J.; Win, H. Technology transfer between university research centers and industry in Singapore. *Technovation* **2004**, *24*, 433–442. [CrossRef]
28. Larsson, M.; Wall, A.; Norström, C.; Crnkovic, I. Technology transfer. In Proceedings of the 2006 International Workshop on Software Technology Transfer in Software Engineering, Shanghai, China, 22 May 2006; ACM Press: New York, NY, USA, 2006. [CrossRef]
29. Duarte, C.H.C.; Gorschek, T. Technology transfer—Requirements Engineering research to industrial practice an open (ended) debate. In Proceedings of the 2015 IEEE 23rd International Requirements Engineering Conference (RE), Ottawa, ON, Canada, 24–28 August 2015. [CrossRef]
30. Vansovits, V.; Tepljakov, A.; Vassiljeva, K.; Petlenkov, E. Towards an intelligent control system for district heating plants: Design and implementation of a fuzzy logic based control loop. In Proceedings of the 2016 IEEE 14th International Conference on Industrial Informatics (INDIN), Poitiers, France, 19–21 July 2016; pp. 405–410. [CrossRef]
31. Maciejowski, J. *Predictive Control: With Constraints*; Prentice Hall: London, UK, 2002.
32. Nesterov, Y.; Nemirovskii, A. *Interior-Point Polynomial Algorithms in Convex Programming*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 1994. [CrossRef]
33. Wills, A.; Mills, A.; Ninness, B. FPGA implementation of an interior-point solution for linear model predictive control *. *IFAC Proc. Vol.* **2011**, *44*, 14527–14532. [CrossRef]
34. Scales, L.E. *Introduction to Non-Linear Optimization*; Macmillan Education: London, UK, 1985. [CrossRef]
35. Wright, S.J. *Primal-Dual Interior-Point Methods*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 1997. [CrossRef]
36. Roos, C.; Terlaky, T.; Vial, J.P. *Theory and Algorithms for Linear Optimization*; Wiley: Hoboken, NJ, USA, 1997.
37. Hendrix, E.M.; G.-Tóth, B. *Introduction to Nonlinear and Global Optimization*; Springer: New York, NY, USA, 2010. [CrossRef]
38. Wächter, A.; Biegler, L.T. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.* **2005**, *106*, 25–57. [CrossRef]
39. Golub, G.H.; Van Loan, C.F. *Matrix Computations*; JHU Press: Baltimore, MD, USA, 2012; Volume 3.
40. Eclipse Foundation. Eclipse IDE. Available online: https://www.eclipse.org/eclipseide/ (accessed on 10 May 2022).
41. JetBrains. IntelliJ IDEA. Available online: https://www.jetbrains.com/idea/ (accessed on 10 May 2022).
42. Varanasi, B. *Introducing Maven*; Apress: Berkeley, CA, USA, 2019. [CrossRef]
43. Apache Software Foundation. Apache Maven Project. Available online: https://maven.apache.org/ (accessed on 10 May 2022).
44. VMware. Spring Framework. Available online: https://spring.io/ (accessed on 1 May 2022).
45. Johnson, R.; Hoeller, J.; Donald, K.; Sampaleanu, C.; Harrop, R.; Risberg, T.; Arendsen, A.; Davison, D.; Kopylenko, D.; Pollack, M.; et al. The spring framework–reference documentation. *Interface* **2004**, *21*, 27.
46. Kochedykov, V.Y. JLibModbus. Available online: https://github.com/kochedykov/jlibmodbus (accessed on 10 May 2022).
47. Villani, V.; Sabattini, L.; Czerniaki, J.N.; Mertens, A.; Vogel-Heuser, B.; Fantuzzi, C. Towards modern inclusive factories: A methodology for the development of smart adaptive human-machine interfaces. In Proceedings of the 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Limassol, Cyprus, 12–15 September 2017. [CrossRef]
48. Nachreiner, F.; Nickel, P.; Meyer, I. Human factors in process control systems: The design of human–machine interfaces. *Saf. Sci.* **2006**, *44*, 5–26. [CrossRef]
49. Darby, M.L.; Nikolaou, M. MPC: Current practice and challenges. *Control Eng. Pract.* **2012**, *20*, 328–342. [CrossRef]
50. Liptak, B.G. *Instrument Engineers' Handbook, Volume 2: Process Control and Optimization*, 4th ed.; CRC Press: Boca Raton, FL, USA, 2006.
51. Ljung, L. *System Identification: Theory for the User*, 2nd ed.; Prentice-Hall: Upper Saddle River, NJ, USA, 1999.
52. Armenise, G.; Vaccari, M.; Capaci, R.B.D.; Pannocchia, G. An open-source system identification package for multivariable processes. In Proceedings of the 2018 UKACC 12th International Conference on Control (CONTROL), Sheffield, UK, 5–7 September 2018. [CrossRef]

53. Patne, V.; Ingole, D.; Sonawane, D. FPGA implementation framework for explicit hybrid model predictive control. *IFAC-PapersOnLine* **2020**, *53*, 362–367. [CrossRef]
54. Pattel, N.K.B. *Practical Design and Application of Model Predictive Control*; Elsevier: Amsterdam, The Netherlands, 2018. [CrossRef]
55. Sousa, Á.C.E.; Leite, V.J.S.; Scola, I.R. Affordable control platform with MPC application. *Stud. Inform. Control* **2018**, *27*, 265–274. [CrossRef]
56. Tatulea-Codrean, A.; Lindscheid, C.; Farrera-Saldana, R.; Engell, S. Extension of the do-mpc development framework to real-time simulation studies. *IFAC-PapersOnLine* **2019**, *52*, 388–393. [CrossRef]
57. Production Association "SAEM". Boiler KVGM-100 Data Sheet. Available online: https://saem.su/kotel-kv-gm-100-150-kv-gm-116-3-150 (accessed on 10 May 2022).
58. Lahiri, S. *Multivariable Predictive Control: Applications in Industry*; John Wiley & Sons Ltd.: Hoboken, NJ, USA, 2017.
59. Vansovits, V. APCsimple. Available online: https://apc-simple.net/ (accessed on 10 May 2022).
60. InfluxDB. Available online: https://www.influxdata.com/ (accessed on 10 May 2022).
61. TimescaleDB. Available online: https://www.timescale.com/(accessed on 10 May 2022).