*Journal of* ELECTRICAL ENGINEERING, VOL 72(2021), NO2, 78–88

sciendo

PAPERS

# Automatic generation of a PLC controller based on a control system-identified model

**Tomáš Sýkora, Michal Husák, Ondřej Baštán, Tomáš Benešl**[1]

This paper discusses the automatic generation of controller codes through a model created in the MATLAB environment the aim of the research is to simplify the implementation and tuning of the closedloop circuit in industrial systems in this context, the concepts and steps outlined herein could produce a more intuitive and powerful alternative to the pre-defined software components of control systems that enable closed-loop control. We propose a methodology for identifying the system through an industrial control setup, creating a control loop model, and generating a code implementable in the control setup. Importantly, we also present and compare the results obtained with various variants of the automatically generated control circuit containing a pre-defined controller of the industrial control system. The outcomes of the research allow us to conclude that the implementations generated by the model perform better that the option using a built-in controller.

K e y w o r d s: control system, system identification, PLC code generation, simulink PLC coder, industrial automation

## 1 Introduction

Feedback operation or control are traditional concepts within the automation of industrial systems. A vast majority of automated systems include a feedback loop that facilitates automatic control. Designing the algorithms for such loops, however, usually requires skilled operators and relatively good knowledge of the controlled system. These factors and preconditions, together with a suitable industrial control system, then allow us perform an automatic task with a closed loop.

Most classic industrial control systems, such as programmable automata, enable the use of builtin software or hardware blocks serving as some of the basic controllers, including PIDs or PSDs (whose structures are described in paper [1]). As a rule, these blocks are user-friendly and allow easy and quick implementation of a robust controller. The basic disadvantage of most control systems equipped with these blocks is nontransparent implementation. In such standard "predefined" controllers, the individual parameters can be monitored and edited, but it is often not entirely clear how the parameters are subsequently brought into the regulatory process and how a specific implementation is carried out. A regulatory scheme (from the manufacturers' traceable documentation, *eg* , [2]) corresponding to the specific implementation is either not mentioned in the block-related documentation or is very concise.

Predefined controllers are designed to be usable for the widest possible range of closed-loop SISO systems, and in such a manner that their setting is performed more or less automatically, without the need of advanced knowledge of control theory. Although such an approach seems useful, there are also situations where it is inappropriate and its use can cause several problems when putting controllers into practice.

Typical examples of such scenarios consist in the application of these blocks in MIMO or MISO systems or where we do not know the controlled system sufficiently, making automatic identification necessary.

As a rule, the algorithms to execute automatic controller setting are limited to several basic variants [3]. These then include, for example, setting the parameters depending on the system's response to an impulse or calculating the parameters of the controller by adapting the Ziegler-Nichols method, which sets the controller by exploiting the parameters of the marginal cycle, for which the feedback system needs to be set to the limit of stability, *ie* to a mode where the system oscillates with steady oscillations. Setting the controller via Ziegler-Nichols to regulate DC motors is outlined in, for example, paper [4].

However, these methods are completely unsatisfactory in many diverse closed-loop systems. Impulse-based identification can often be insufficient, and bringing the system to the stability limit is even characterizable as dangerous for a wide range of controlled systems. In such cases, the system integrator has to rely on his or her own experience only [5]. Conversely, self-adjustment algorithms and pre-defined controllers limit an experienced integrator by not allowing them an insight into the implementation, thus eliminating the actual possibility of debugging or modifying the module for more sophisticated or special solutions.

We discuss the design and testing of a solution alternative to the pre-set controllers described above, a concept intended to facilitate transparent and verifiable

[1] Department of Control and Instrumentation, Faculty of Electrical Engineering and Communication, Brno University of Technology, Technická 3058/10, 616 00 Brno, Czech Republic, xsykor23@vutbr.cz

https://doi.org/10.2478/jee-2021-0011, Print (till 2015) ISSN 1335-3632, On-line ISSN 1339-309X

automatic implementation of a controller with extended automatic identification options, diversified adaptability to long-term changes, and tuning modes. The resulting product should be suitable for the practical use of the methodology in a real environment.

The aim of the research is to employ MATLAB to generate an implementation code, utilizing a code created in Simulink. The described solution has a potential to ensure a high level of reliability, transparency, and versatility while maintaining simplicity and fostering an overall increase in the resiliency of the control circuit. Another value added may rest in the possibility of using formal code verification methods.

We briefly characterize the state of the art, then outline the conceptual properties of the solution, and present the individual possibilities of identifying open-loop systems usable for incorporation in both common basic and less common complex industrial systems. These identification methods [6, 7] will then be employed in an autonomous system identification algorithm. The elementary types of controllers that are convenient for general control and can be implemented by an automatically generated code through Matlab. To compare the generated implementations and built-in regulator blocks, we have to select the optimum criterion; however, as the optimum solution differs between diverse types of systems. After describing the physical model of the controlled system used to verify the implementations we list the measured data and the calculated criteria, and compare the achieved results.

## 2 Related research

The entire field of industrial control systems as related to feedback systems and Matlab can be primarily divided into two categories, namely, model-based design (MBD) and code verification. In this connection, we should also point to hardware-in-the-loop (HIL/HWIL) simulation, a technique discussed in a significant number of papers. To ensure verifiability and portability, the code has to be generated in such a manner that the resulting format complies with the IEC 61131-3 standard, used by most industrial control systems. The research published to date focuses especially on the verification of generated implementations or overall code generation based on the modelled functionality.

For example, Bayrak *et al* [8] compare the outputs of a system model implemented in Matlab Simulink with the outputs of an industrial control system whose code was generated on the basis of a relevant model (as presented above) and whose implementation was carried out through different development environments into control systems of different manufacturers. The authors then investigated if the output data of the real system with the generated implementation corresponded to those of the model. After comparing the results of systems implemented by different manufacturers via criteria aimed at tracking the shape and deviations, the researchers concluded that the best shape match between the model output data and the real system data with the generated code implementation is achieved in Beckhoff systems, while the deviation criterion best suits the implementation in Siemens control systems. Furthermore, Bayrak *et al* established that if the Continuous flow chart (CFC) syntax is used, all the tested control systems exhibit a difference of not more than 1.5% between the output data of the real implementations and the simulation output data.

Within our research, a real system with real elements is employed to verify the functionality of the proposed controller. For this reason, due to the sensor noise, accuracy of the A/D converters, and real interference, we expect greater deviations. The comparison is centred primarily on a pre-defined controller implemented in a specific control system and a controller generated by Matlab Coder [9] depending on the control circuit model. Article [10] describes the design and generation of software for a B&R platform control system through the MBD method. A block diagram characterizing the authors approach is shown in Fig. 1; the block scheme indicates that the procedure consists in an automatic generation of the control system code from a mathematical model of the production process created in Matlab Simulink. The authors of [10] do not use the resources defined by IEC 61131-3 and generate the resulting code in ANSI C, which is supported by B&R platform control systems.

The researchers describe, among other problems, the detailed formation of a system model in Matlab, and they also characterize the related code generation via the Automation studio environment. A similar approach is

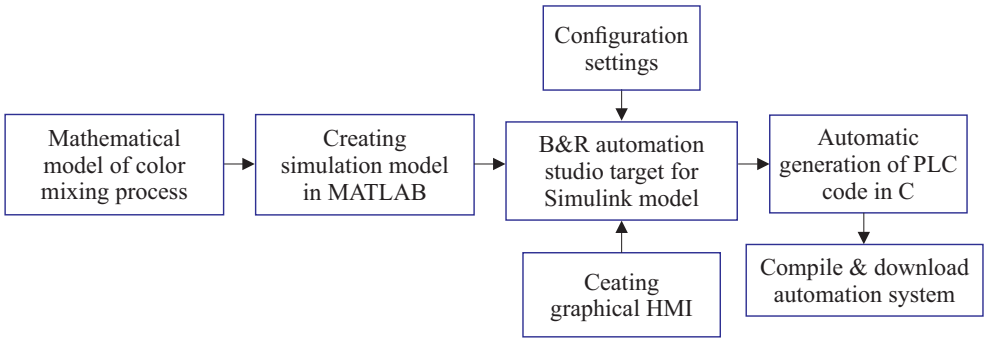


**Fig. 1.** MBD block diagram for a BR control system, [10]

applied in our study. The difference rests primarily in that we use a different modelled system and standard resources defined by IEC 61131-3.

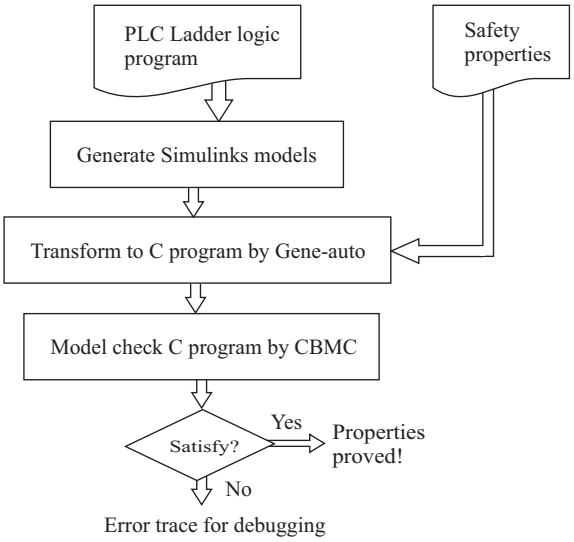


**Fig. 2.** Basic procedure of model based verification, [11]

As mentioned above, a frequently described category in current research, relating to the use of generated code, is the verifiability. A code generated automatically is suitable for implementing safety functions or safety systems. Because such a code has a model characterizing its functionality, it is easier to be verified and, thus, certified as safe.

This issue is addressed in, for example, He, Oke and Allens paper on PLC code verificiation by using Matlab Simulink [11]. One of the methods proposed by the researchers is the MBD technique (Fig. 2), which can be adapted to our model and employed to verify our implementation. The authors of paper [12] focus on a similar issue within the wider framework of generating and verifying codes for safety applications.

Hul, Wagner and Allen describe in their article [13] the generation of a regulator code by utilizing the Simulink Coder tool. The authors exploit a mathematical description of the and generation of an MPC controller (predictive control model) for an industrial PLC control system. The eventual behaviour of the generated code was verified in laboratory conditions. By extension, the problem of code generation is further outlined in paper [14].

One of the target fields for generated codes is virtual commissioning. To obtain a pattern for a generator, we can create a model that describes the operation of a separate industrial system. The control system containing the implementation based on the generated code can also serve as a simulator of a relevant industrial system. In terms of practice, this solution may find use in, for example, verifying the functionality of a program; training operators; and education. The problem is further addressed in in paper [15]. Within our research group, alternatives to this solution were previously examined during simulations of a batch production control system designed to teach industrial control [16].

## 3 Solution concept

Considering the current state of the research and the above requirements, we defined four basic approaches using an automatically generated code to implement solutions to the task of industrial closed-loop system control. The individual approaches are described in the following subchapters.

### *3.1 Mathematical model*

This controller designing technique assumes that the mathematical model of the feedback system is known. The target application lies where it is practically too complicated, costly, or impossible to implement system identification via one of the well-proven methods, such as the technique based on evaluation of the response to a step. This approach also facilitates designing a controller for the system before the real system is formed.

Figure 3 shows a typical solution procedure. Based on a mathematical description, a model of the entire control loop, including models of the system and the controller, is created by using the Simulink tool. The controller model is then processed via Simulink Coder, which ; thus,

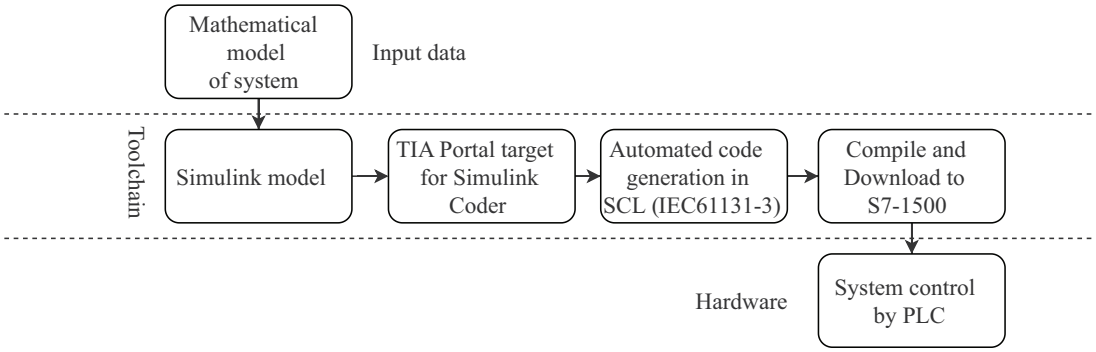


**Fig. 3.** Block diagram of the controller generated from a mathematical description of the system

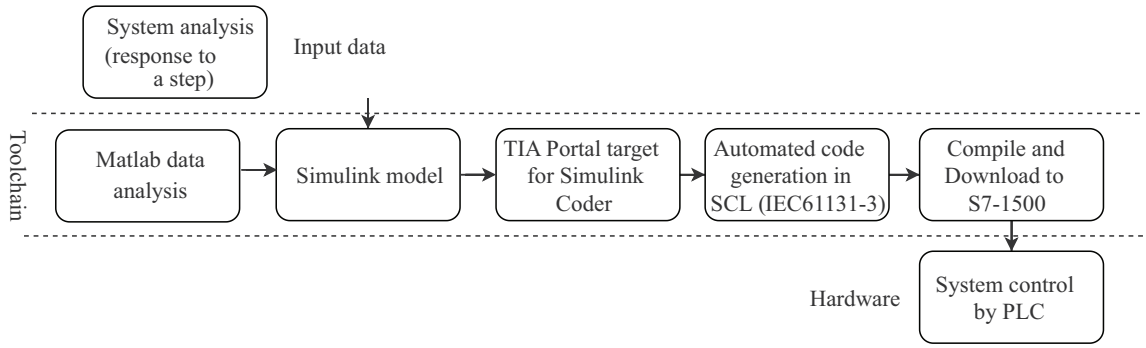


**Fig. 4.** Block diagram of the controller generated from identification according to the step response data

we obtain a code representing the controller in a format that is directly usable in programmable automata.

### *3.2 Manual identification*

The second approach rests in designing the controller through parameters obtained via manual system identification. The individual steps are shown below.

A step signal is brought to the input of the feedback system by utilizing an industrial control system, and the related response is recorded. We then insert the measured data into an analytical module in the Matlab environment to adjust the relevant parameters of the controller in the control loop model inside the Simulink tool. The subsequent procedure is the same as that in the previous variant.

### *3.3 Automatic in-field identification*

One of the more sophisticated techniques for implementing the controller in control systems allows automatic system identification and automatic controller adjustment. In practice, this procedure is materialized by, for example, supplementing the system model created in Simulink with a self-executable function that automatically performs the identification (for instance, via step response), analyses the measured data, and adjusts the specific parameters inside the controller in relation to the selected criterion. This concept is shown in Fig. 5.

The "autotune" functionality described above may be suitable for mass-produced control systems, which are subsequently deployed in various feedback systems.

In certain industrial applications, however, this concept is inappropriate, due to the fact that the identification algorithm can independently bring the feedback system into an unstable or even dangerous state. In such systems, the controller settings should always be performed by an experienced integrator.

### *3.4 Adaptive control*

This approach finds application in systems whose parameters change in the long-term perspective, depending on quantities such as ambient temperature and humidity,

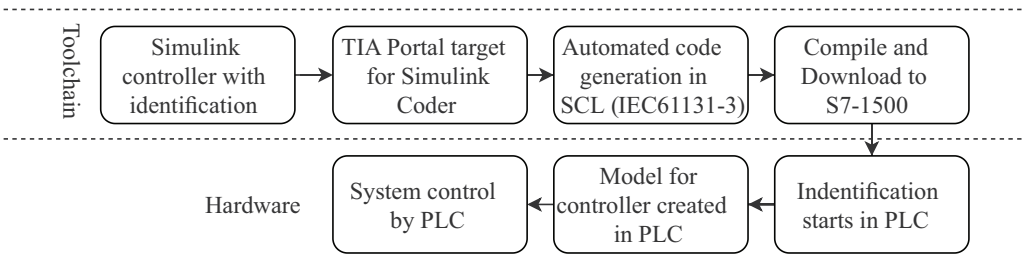


**Fig. 5.** Block diagram of a controller with an identification module

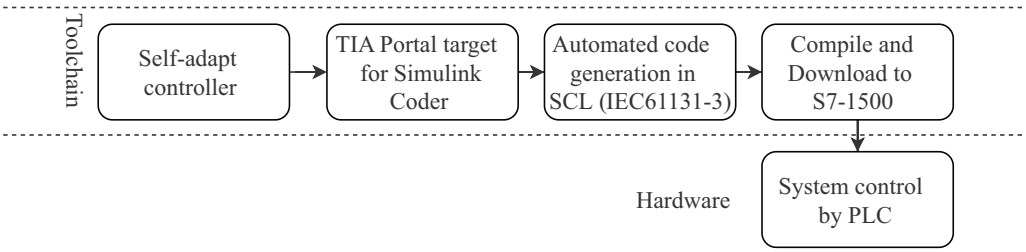


**Fig. 6.** Clock diagram characterizing adaptive controller generation

especially at a turn of the seasons, or the performance of the associated technology.

In this type of feedback systems, it is usually very difficult to perform single-step parameter setting such that the resulting control manages all long-term changes at an approximately identical quality. A possible solution to the problem may consist in utilizing gain-scheduling for the adaptation. This approach, however, requires knowledge of not only the variable that influences the regulatory process but also the ability to measure it; thus, an adaptive controller appears to be a suitable choice (Fig. 6). The advantage of such controllers is that they actively adjust their parameters according to the current course and conditions of control. Conversely, a major drawback may rest in the poor behaviour of the controlled variable at the start-up or during a step change in the parameters of the controlled system. These disadvantages are reducible by convenient initial setting of the control loop before the system is started.

## 4 Methods

This chapter provides a description of the methods used in the identification, modelling, and assessment of our solution.

### *4.1 Identification methods*

The techniques presented in Chapter 3 assume the use of several identification methods to yield a dynamic model of the regulated system. This model will then facilitate implementing the model.

The identification methods can be classified into two categories parametric, which compile the system model analytically, and non-parametric, which determine the parameters of the system experimentally. In the former case, we assume that the structure of the system follows the laws of nature and can be described via internal state variables. By contrast, experimental methods interpret the system primarily as a black-box and analyse its behaviour through its responses to external influences.

Considering the fact that the proposed solution should be applicable to unknown systems with an unknown state description, it is appropriate to use experimental methods to create an initial system model. The most common instruments of nonparametric identification are techniques based on impulse or transient characteristic analysis correlation methods, and frequency and spectral analyses An important requirement lies in the industrial applicability of the resulting control circuit design methodology, and therefore the applied method should be practicable directly in field conditions. In our solution, the system is identified based on its response to the input signal [6].

An important aspect of the actual identification is the type of the input signal used to excite the system. The quality and character of the input signal have a major impact on the accuracy of the identification. The most widely used types of input signals are as follows:

- Step response
- Pseudo Random Binary Sequence (PRBS)
- Sum of harmonic waveforms. The step response class allows easy identification of the static gain, rise time, and overshoot.

Pseudorandom binary sequence is a type of signal that takes on only two values (binary) and exhibits white noise-like behaviour; in PRBSs, however, we are invariably capable of determining the following element (pseudorandom). The signal is generated by using a shift register and modulo 2 adders [7]. The identification can take place in an open or a closed loop. The simplest and most popular option is open loop identification, where the actual identification proceeds near one of the operating points and is based on a steady state.

The closed-loop identification variant is markedly more complicated. The procedure must be used on systems that do not allow the creation of an open loop, as they would become unstable with the change; alternatively, feedback is required for security reasons. In closed-loop identification, we employ either a direct approach, corresponding to open-loop identification, or an indirect approach, in which closed-loop transmission is identified and the system transmission is calculated based on the knowledge of the controller's transfer function.

Another significant parameter in the identification process is the sampling period. This is generally set to 10% of the rise time. A sampling period too short causes practical problems, due to the retrieved poles being close to "1" and also the greater sensitivity in rounding. Conversely, an excessively long sampling period is inferior in terms of the quality of the control procedure.

For the first iteration of our solution, we will employ the principles of open-loop identification.

### *4.2 Controller possibilities*

Without knowing the actual value of the output, we are unable to compensate for the impact of the faults, and it is not possible to ensure reliable monitoring of the setpoint in the event of changed system parameters. In cases of accurate knowledge of the system the action can be calculated in advance, such that the resulting control is optimal. The optimality of the actual control is determined by the selected criteria, including the requirement to minimize the energy consumption or to optimize the speed. To provide a relevant example, we can refer to energy production planning according to the time of the day and season.

The system is controlled by a controller that uses other components to achieve the required goals. The controller delivers a minimum control deviation by changing the action intervention affecting the system. To ensure proper functioning, the controller consists of multiple parts. In practice, three-component controllers are frequently used; these contain a proportional, an integrative, and a derivative component. By combining these components, we can achieve the required regulation properties.

Historically, analogue PID controllers were first used. Due to the fact that control systems are usually discrete, the controllers implemented by them are also discrete. Thus, we are talking about a discrete variant of the PID controller, PSD controller. Predefined controllers in industrial systems could be referred to as pseudo PIDs. The constants of these controllers are designed similarly to those for continuous controllers. The control system then transforms the proposed solution into a discrete form during the compilation.

Another option is to employ a state controller. The essence of state regulation lies in introducing the feedback from the individual states of the regulated system to its input. The obvious disadvantage is the need to measure, or at least to reconstruct, the states of the regulated system. The reward for this effort is the ability to force almost any dynamics on the control loop. Compared to a PID controller, we can achieve a faster transient with a lower frequency. The main limitation of this approach stems from the fact that when an extremely fast response is being designed, state feedback generates action interventions of such amplitudes that are not feasible by a real action member. In addition, designing too much gain in the state links may cause excessive sensor noise amplification. Therefore, when designing the feedback, we should only require such a speed of regulatory processes that the above-mentioned problems do not occur.

### *4.3 Regulation criteria (analysing the dynamic properties of the control circuits)*

To compare and evaluate the results achieved, it will be necessary to use evaluation criteria, whose purpose will be to determine the behaviour of the systems in the transient process. The parameters of interest are the transient stabilization rate, maximum overshoot, and oscillation. These are then the dynamic properties of a system consisting of a regulated controller setup and feedback. The aim is to achieve the most optimal dynamic properties possible, via suitable setting of the controller parameters. The optimality itself must be defined in advance by using a criterion function. An improper choice of the criterion will result in suboptimal control loops.

As we intend to assess the dynamic properties of a transient effect, a criterion that considers time has to be employed. Integral criteria satisfy this requirement; they assess the quality exploiting the course of the control deviation, which is obtained from the response of the control circuit to a step change of the setpoint.

Linear integral criterion

This instrument calculates the area between the course of the control deviation e(t) and the steady state deviation. The area is termed the linear control area. The criterion is

$$J_L = \int_0^\infty [e(t) - e(\infty)]\mathrm{d}t\,. \tag{1}$$

Quadratic criterion

This instrument expresses the quadratic control area. Due to the square multiplication, negative deviations can be included smoothly. The negativity is cancelled because of the multiplication. At the same time, however, this property is a disadvantage: It attaches more weight to larger deviations, which then leads to large overshoots and deviation frequencies. The criterion is

$$J_K = \int_0^\infty [e(t) - e(\infty)]^2\,\mathrm{d}t\,. \tag{2}$$

ITAE criterion

The disadvantage of the quadratic criterion is improved by the ITAE criterion (Integral of Time Multiplied by Absolute value of Error), given in equation (3). As is obvious from the formula, the weight of the deviation increases linearly with time. The problem of negative deviations is solved by utilizing an absolute value; however, due to the non-linear function of the value, it is not possible to calculate the controller design analytically, and thus repeated simulations are employed for the calculation

$$J_{\mathrm{ITAE}} = \int_0^\infty |e(t) - e(\infty)|t\,\mathrm{d}t\,. \tag{3}$$

The proposed solutions are compared with all of the above criteria.

### *4.4 Code generation system models*

The Matlab Simulink tool enabled us to create the system model. Two variants of the model were implemented in this tool: One via the transfer function and the other by using the state description of the system. Subsequently, we selected suitable regulators. The selection criterion consisted in the ability to generate the PLC implementation via the Simulink PLC Coder tool. This requirement is met by the PID controller implemented by exploiting the Simulink-block PID Controller, which allows the block to be switched to the discrete mode, *ie* . the PDS controller mode. The required parameters of the controller were then found in the block by using the "tune" function.

The other controller was designed with the Siso tool, where a PI controller was designed based on the time signals and responses of the controlled system. In performing the task, it was necessary to ensure a sufficient supply of stability (GM (gain margin) and PM (phase margin)). However, the proposed controller it has a continuous form, and, to generate the code, it is necessary to convert its model to a discrete form with a fixed sampling period. The discretized form of the controller designed in the Sisotool tool is visualized in Fig. 7. For this model, it

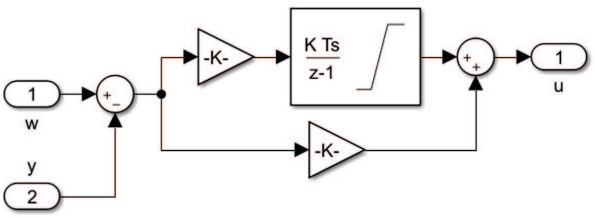


**Fig. 7.** Simulink discrete model4 of the PIPS controller

is now possible to automatically generate a code implementable in a PLC.

The last tested variant of the design was a procedure consisting in the identification and description of the system by using a state model, *ie* ., matrices A, B, C, and D. Subsequently, this description was employed to design a state controller, meaning matrices K and L. The discretization of this controller was solved automatically via a special block.

## 5 Real test system

A real system connected to a real control system was utilized to verify the functionality of the results.

### *5.1 Controlled system*

The regulated test system consisted of a tube having 80 mm in diameter and exhibiting a length of 500 mm. A voltage-controlled fan was placed at the inlet of this tube. The air flow through the tube was measured at its outlet, utilizing a rotary encoder with a vane flow meter. The controlled variable is the air flow (output encoder speed). The system is visualized in Fig. 10. Although the system is a higher order one, it can be simplified, with respect to the time constants, to a system that has three states and a traffic delay. These states are the inlet fan speed, air flow through the tube, and outlet encoder speed. We subsequently approximated the system to a second-order one with delay for testing purposes.

The system itself features a number of nonlinearities that occur in both the inlet fan and the airflow mea-

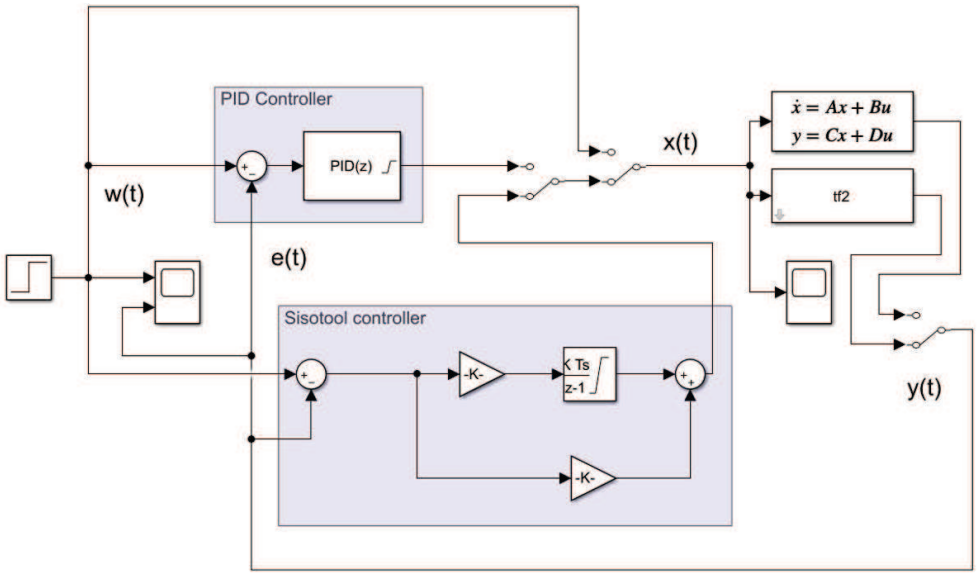


**Fig. 8.** Simulink model with the state and LTI models and two different controllers implemented

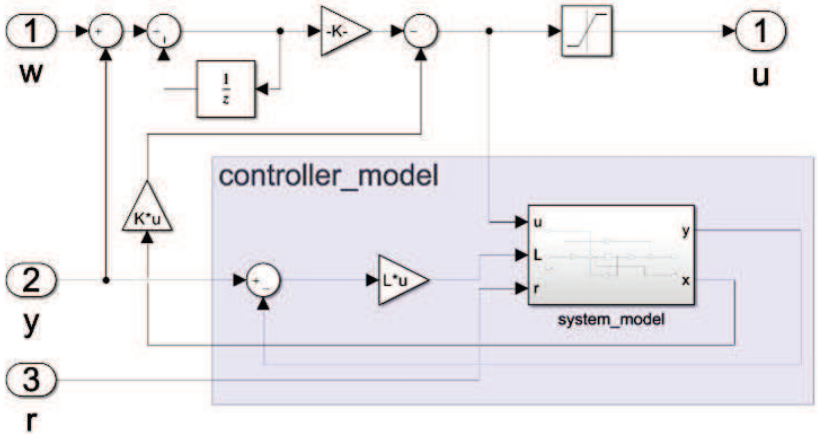


**Fig. 9.** State controller for PLC code generation

surement. The fan starts rotating at an effective action voltage of 12 V and stops when the voltage drops below 9 V. The output flow meter is unable to detect an air flow lower than that generated when the inlet fan is energized with 10 V. For this reason, the system was used in range of 12-24 Vrms value of the output voltage.

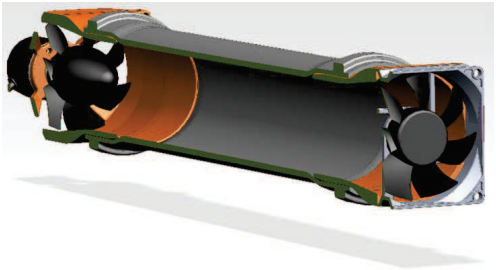

**Fig. 10.** Regulated test system

As part of the verification, our goal is to regulate the air flow through the tube. This regulation should be gentle on the actuator, fast, and, if possible, without an overshoot.

### *5.2 Hardware and software*

The following hardware and software allowed us to control the test system and its subsequent tuning.

The applied control system was a compact PLC manufactured by Siemens, type S7-1214 DC/DC/DC equipped with firmware version 4.4. The engineering framework TIA Portal V16 was employed as the development environment for the HW. Matlab r2020a facilitated the identification, modelling, and code generation, with the following tools: PLC Coder v.3.2; System identification toolbox v.9.12; Simulink tool v.10.1; and control system toolbox v.10.8. The Siemens NX r1914 tool with the Nastran simulation solver was applied to design and simulate the hardware of the testing system.

## 6 Results

This chapter characterizes the results achieved within the individual areas of interest set out in the paper.

The system was identified by using three input-output measurement methods, namely, application to a step response, a linear increase setpoint, and PRBS. A system consisting of a regulated setup, a D/A converter, and an A/D converter of the control system was identified. The sampling period in the control system was set to 10 ms. The D/A converter was implemented via a PWM output, whose period was chosen to be between 10.25 and 50 ms; this allowed us to determine the impact of the period on the result of the system identification. A period of 25 ms was chosen experimentally, and showed the best dynamic parameters.

All of the measured data were assembled, entered into the System identification tool in Matlab, and analysed in such a manner that the response of the identified system corresponded as accurately as possible to the original response.

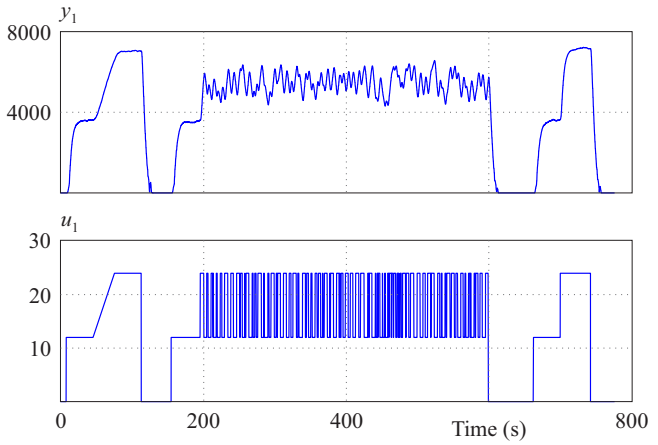


**Fig. 11.** System identification for the ms PWM

For the 25 ms PWM, the model was identified as

$$F_{25\text{msPWM}}(s) = e^{-0.2s}\frac{357.3}{s^2 + 2.523s + 1.186}. \qquad (4)$$

The results of the identification indicate an oscillating system of the second order. We successfully identified the system and then materialized transmission with an accuracy of 93%. The identification was more accurate (by units of percent) with the third order; however, for testing purposes and simpler controller design, the second order of the system was chosen.

The theoretical response of a control loop containing the above system, whose controller was designed by using the SISO tool, is displayed in Fig. 12.

An implementation via a PID controller block was also used for the identified system; this option facilitated quick setting of the controller parameters, depending on the step response of the closed loop with the possibility of defining the aggressiveness, or speed and robustness, as regards overshooting the controlled system.

Another implemented variant then consisted in a controller whose model implemented state control. All of the variants were subsequently generated via the Simulink PLC Coder tool and deployed in a Siemens PLC. In addition to the three above-mentioned automatically generated implementations, we applied in the PLC an option with a predefined controller, tuned by using the autotune and finetune functions. The results of measuring the control loop responses in a real system are presented in Figs. 12, 13, and 14.

Considering the waveforms in Fig. 15 and Fig. 16, it is not easy to decide whether a particular implementation is significantly better than the others. Thus, the criteria were calculated for all of the variants, whose results are shown in Tab. 1.

The action sequences in Fig. 16 indicate that setting the controller via the TIA Portal is not very suitable for applications using actuators that are prone to abrupt

**Table 1.** The controller results according to the pre-defined criteria

| | Linear ($\times 10^5$) | Quadratic ($\times 10^9$) | ITAE ($\times 10^8$) |
|---|---|---|---|
| State controller | 9.47 | 3.568 | 6.27 |
| Matlab PID | 9.48 | 3.459 | 6.41 |
| SISO tool PID | 5.39 | 3.456 | 7.63 |
| TIA Portal PID | 9.97 | 3.604 | 9.10 |

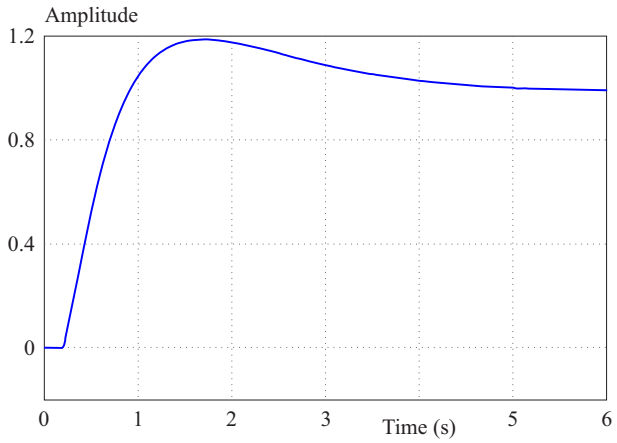


**Fig. 12.** Sisotool controller simulated step response

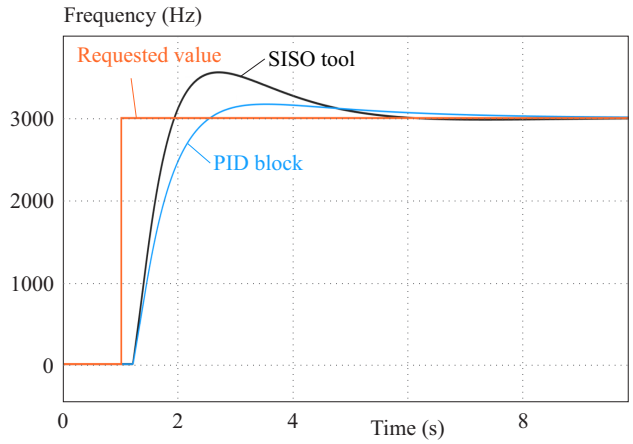


**Fig. 13.** Simulink model step response with the sisotool discretized controller and PID block controller

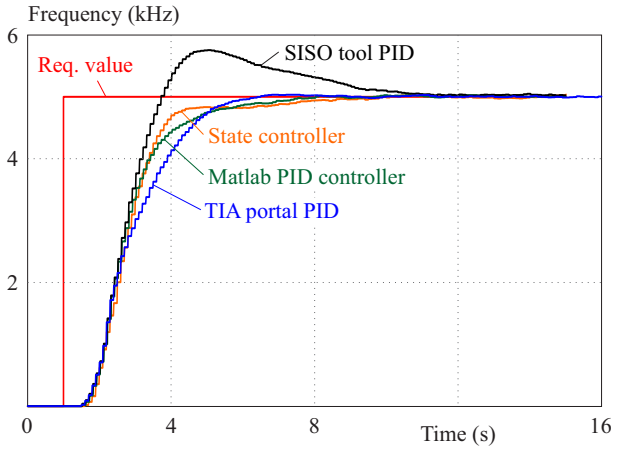


**Fig. 14.** Controlled variable response

changes (typically relays). In the system applied within this paper, the action is filtered by an actuator, *ie* ., a DC motor. In some applications, though, such an action behaviour is undesirable.

### *6.1 Real code generation problems*

When creating a subsystem for code generation, we need to pay due attention to the feasibility in the target control system. From the perspective of practical implementation, it is also necessary to assume, among other aspects, that the backward Euler's methods - not the forward ones - have to be used. Moreover, we have to perform discretization in all time-dependent processes. In the control system, it is then necessary to run the generated code with the same sampling period as that set in the model. Even if these rules are observed, however, the control subsystem may not be generable, due to the presence of algebraic loops, and will need to be resolved manually.

## 7 Discussion and conclusion

We presented in detail the concepts, tasks, and setup options that facilitate the testing of automatic code generation for PLCs; the paths towards the desired goal were explored via a model using Matlab Simulink PLC Coder. Procedurally, we generated various types of controllers, exploiting identification performed by the control system and methods described above. The resulting control circuits were designed on the basis of the identified system model, whose response matched that of the real system at an accuracy of 93%. As a result, there was a minimal difference (in the order of units of percent) between the theoretical response of the controllers with the identified system and the response of the real control loop.

### *7.1 Comparison with the controller natively implemented in the PLC*

The resulting responses of the individual control loops are displayed in the graphs in Figs. 12-14. Compared to the original assumption, the predefined PID controller set

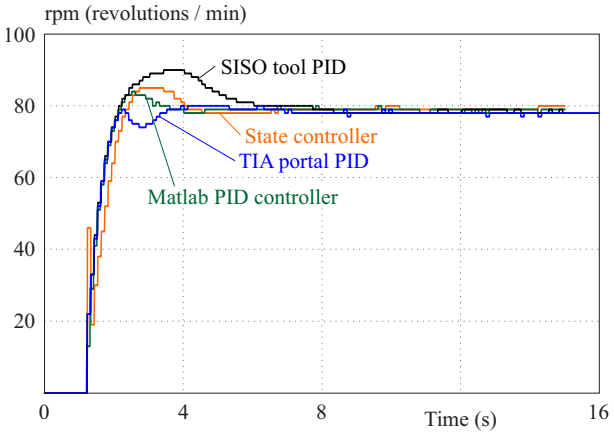


**Fig. 15.** Inlet fan speed (action)

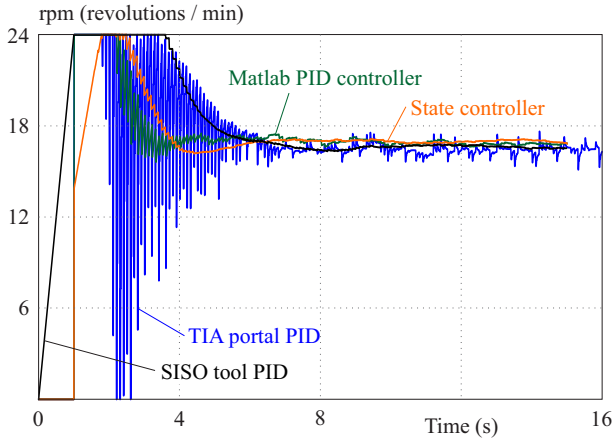


**Fig. 16.** Controller output in the form of a voltage/PWM, written to the PLC output

in the TIA Portal yielded very good results, and its responses to the setpoint were comparable to the outcomes of the other tested implementations.

The controlled system exhibited minimal traffic delay and was relatively fast (see the time constants system equation 4); thus, the design of the controller utilizing automatic tuning in the TIA Portal reached the same level as the designed controllers relying on Matlab. However, in a system having a longer traffic delay or time constants in the order of tens of seconds or more, the result could be markedly worse.

We believe that the good performance of the controller from the TIA Portal followed from the relatively large derivative component but was achieved only at the expense of an oscillating output of the controller, as illustrated in Fig. 16. We can nevertheless point out that the results of the controller designed by using the PID block in MATLAB, and also those of the state controller, are similarly satisfactory but without the oscillation of the output; such an oscillation can adversely affect the life of the actuator and is therefore generally undesirable.

### *7.2 Discussion of the results and applicability*

Within the research characterized in this article, we established that the possibilities of code generation for PLCs, in terms of not only generating control elements but also, for example, offering diverse code verification options and other related aspects, comprise major potential. Using the methodology described herein to control fast systems having good feedback is less beneficial than initially expected; however, system identification and the related design of a controller (or filters) via the tools and methods presented in the paper could prove advantageous in the construction of more complex feedback circuits, filters for noisy feedback signals, and control of MIMO, MISO, and SIMO systems. Such an approach embodies the almost exclusive way to functionally implement a control circuit, except for SISOs.

### *7.3 Future work*

At the follow-up research stages, we intend to test the functionality of the generated controllers on different systems, those requiring adaptive control in particular. The fields and subdomains to be explored in this context include also formal code verification and fail-safe code generation for safety PLCs, as combining adaptive control and formal verification will increase the resilience of control circuits.

The primary aim, however, consists in designing adaptive regulators, which have not found wide use thus far due to their demanding implementation; one of the most prominent related tasks is then to create a universal methodology for a simple automatic design of adaptive control usable in a wide range of industry standard feedback systems.

*Acknowledgements*

The completion of this paper was made possible by the grant No. FEKT-S-20-6205 - Research in Automation, Cybernetics and Artificial Intelligence within Industry 4.0 financially supported by the Internal science fund of Brno University of Technology.

## References

[1] P. Gawthrop, "Self-tuning PID control structures", *IEE Colloquium on Getting the Best Out of PID in Machine Control*, 1996, 10.1049/ic:19961463.

[2] "TF4100 TC3 Controller Toolbox: FB_CTRL_PID", 2020.

[3] Y. Li, C. Tang, and K. Liu, "PID parameter self-setting method base on S7-1200 PLC", *2011 International Conference on Electrical and Control Engineering*, 2011, 10.1109/iceceng.2011.6057410.

[4] P. Meshram and R. Kanojiya, "Tuning of PID controller using Ziegler-Nichols method for speed control of DC motor", *International Conference On Advances In Engineering, Science And Management (ICAESM -2012)*, IEEE- Nagapattinam, India, 2012.

[5] P. Noskievič, *Modelovaní a identifikace systému*, Ostrava, Montanex, 1999.

[6] T. Soederstroem and P. Stoica, *System identification*, New York, Prentice Hall, 1989.

[7] L. Ljung, *System identification - theory for the user*, (second edition), Englewood Cliffs, NJ, Prentice Hall PTR, 1999.

[8] G. Bayrak, P. Murr, S. Ulewicz, and B. Vogel-Heuser, "Comparison of a transformed Matlab/Simulink model into the programming language CFC on different IEC 61131-3 PLC environments", *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies Factory Automation (ETFA 2012)*, 2012, 10.1109/etfa.2012.6489667.

[9] Simulink Coder Reference, The MathWorks, Inc., 2020.

[10] R. Salunke, P. Vikhe, and T. Sarode, "Implementation of automatic PLC code from MATLAB simulation model using BR automation target for Simulink", *Int. Conf. on Control, Communication and Power Engineering*, 2013.

[11] N. He, V. Oke, and G. Allen, "Model-based verification of PLC programs using Simulink design", *2016 IEEE International Conference on Electro Information Technology (EIT)*, 2016, 10.1109/eit.2016.7535242.

[12] M. Schwarz, H. Sheng, A. Sheleh, and J. Boercsoek, "Matlab / Simulink generated source code for safety related systems", *2008 IEEE/ACS International Conference on Computer Systems and Applications*, 2008, 10.1109/aiccsa.2008.4493678.

[13] R. Hyl and R. Wagnerova, "Fast development of controllers with Simulink Coder", *2017 18th International Carpathian Control Conference (ICCC)*, 2017, 10.1109/carpathiancc.2017.7970434.

[14] S. Ozana and T. Docekal, "The concept of virtual laboratory and PIL modeling with REX control system", *2017 21st International Conference on Process Control (PC)*, 2017, 10.1109/pc.2017.7976196.

[15] A. Pereira, C. Lima, and J. Martins, "The use of IEC 61131-3 to enhance PLC control and MMatlab/Simulink process simulations", *2011 IEEE International Symposium on Industrial Electronics*, 2011, 10.1109/isie.2011.5984336.

[16] V. Kaczmarczyk, T. Benešl, Z. Bradáč, P. Fiedler, and Z. Kaczmarczykov, "SkuBATCH - System for control of technological processes", *IFAC-PapersOnLine*, vol. 52, no. 27, pp. 477-483, 2019, 10.1016/j.ifacol.2019.12.709.

Received 29 March 2021

**Tomáš Sýkora** was born in 1994. He received an MSc in electrical engineering from Brno University of Technology in 2019. Currently, he is a PhD student at the Department of Control and Instrumentation, Faculty of Electrical Engineering and Communication, Brno University of Technology. His research interests include industrial applications, Industry 4.0, and industrial system optimization.

**Michal Husák** was born in 1996. He received an MSc in electrical engineering from Brno University of Technology in 2020. Currently, he is a PhD student at the Department of Control and Instrumentation, Faculty of Electrical Engineering and Communication, Brno University of Technology. His research interests are machine learning in predictive maintenance and industry automation.

**Ondřej Baštán** was born in 1993. He received an MSc in Cybernetics, Control and Measurement from Brno University of Technology, Brno, the Czech Republic, in 2017. At present, he is a PhD student at the Department of Control and Instrumentation, Faculty of Electrical Engineering and Communication, Brno University of Technology. His research interests include industrial control, Industry 4.0, embedded systems, and resilient systems.

**Tomáš Benešl** was born in 1993. He received an MSc in Cybernetics, Control and Measurement from Brno University of Technology, Brno, the Czech Republic, in 2017. His research interests are within industrial applications, Industry 4.0, IoT, data mining, and industrial system optimization.