International Journal of Control, Automation, and Systems 22(6) (2024) 1846-1855
http://dx.doi.org/10.1007/s12555-023-0869-6
ISSN:1598-6446 eISSN:2005-4092
http://www.springer.com/12555

PLC-based Implementation of Stochastic Optimization Method in the Form of Evolutionary Strategies for PID, LQR, and MPC Control

Kajetan Zielonacki and Jarosław Tarnawski*

Abstract: Programmable logic controllers (PLCs) are usually equipped with only basic direct control algorithms like proportional-integral-derivative (PID). Modules included in engineering software running on a personal computer (PC) are usually used to tune controllers. In this article, an alternative approach is considered, i.e. the development of a stochastic optimizer based on the (μ,λ) evolution strategy (ES) in a PLC. For this purpose, a pseudo-random number generator (pRNG) was implemented, which is not normally available in most PLCs. The properties of popular random number generation methods were analyzed in terms of distribution uniformity and possibility of implementation in a PLC. The Wichmann-Hill (WH) algorithm was chosen for implementation. The developed generator with a uniform distribution was the basis for the implementation of a generator with a normal distribution. Both generators are the engines of the stochastic optimization algorithm in the form of the (μ, λ) strategy. For verification purposes, a modular servomechanism laboratory set was used as a test object for PID and linear-quadratic regulator (LQR) control. Moreover, the possibility of using the developed optimizer was shown in an application of model predictive control (MPC). Comprehensive tests confirmed the correctness of the implementation and high functionality of the developed software. Calculation time issues are also investigated.

Keywords: Evolution strategies, global optimization, hardware in the loop, model predictive control, PLC, pRNG.

1. INTRODUCTION

The problems of implementing advanced control algorithms using programmable logic controllers (PLCs) have been reported in the literature for a long time. Especially the model predictive control (MPC) technique has been implemented many times in different versions [1-4]. In many items in the literature one can find hybrid, layered approaches based on a combination of a personal computer (PC) and a PLC, where demanding calculations are performed in the PC, and the results of these calculations are directed to be realized by the PLC in the form of simple control systems, e.g., proportional-integral-derivative (PID) [5,6]. Interesting papers on implementation of optimizers in PLC may be classified depending on different types of optimization: general purposes optimizers [7], quadratic programming (QP) [8,9] neural network [10], particle swarm optimization [11] and ant colony [12]. Verification of programs written for the PLC can be carried out directly with the actual control object, but it is most often carried out beforehand in a software-in-the-loop structure, where a model of the object is inside the PLC program and allows the first tests of the implemented control algorithm entirely in the PLC. Another more advanced form of verification is hardware in the loop (HIL), where the control program is contained in the PLC, and the object model is contained in a runtime environment.

Evolution strategies (ES) are meta-heuristic biology-inspired algorithms. The implementation of the algorithm is based on a stochastic mechanism for modifying an existing individual based on a normal distribution to create a new, potentially better one. Continuously modifying individuals through recombination and mutation, and selecting the top performers within the population, enables the identification of the optimal individual based on a specified criterion. This characteristic enables ES to function as a black-box global optimization algorithm. Classic literature on evolutionary strategies are [13-15]. A more contemporary, comprehensive recommended read is [16].

Major contributions of the presented paper:

1) implementation and verification of uniform and normal pRNGs in a PLC,
2) development and verification of a stochastic optimizer on PLC platform and its application to PID, LQR and MPC control problems,
3) comparison of PLC and PC performance in a complex

Manuscript received December 18, 2023; revised March 22, 2024; accepted April 8, 2024. Recommended by Associate Editor Niket Kaisare under the direction of Senior Editor Sangmoon Lee. Financial support of these studies from Gdańsk University of Technology by the 12/1/2023/IDUB/III.1a/Ra grant under the Radium - 'Excellence Initiative - Research University' program is gratefully acknowledged.

Kajetan Zielonacki and Jarosław Tarnawski are with the Faculty of Electrical and Control Engineering, Gdańsk University of Technology, Narutowicza 11/12, Gdańsk, 80-233, Poland (e-mails: s181428@student.pg.edu.pl, jaroslaw.tarnawski@pg.edu.pl).
* Corresponding author.

computational task to determine the applicability of the developed optimizer.

2. PSEUDO-RANDOM NUMBER GENERATOR IN PLC

This section presents the process of implementing a pseudo-random number generator (pRNG) in the PLC, using the structured text (ST) programming language. First, an overview and comparison of different algorithms was made. Then, the chosen generator was tested based on its ability to generate uniformly distributed numbers. Lastly, the method of generating normally distributed numbers was described and verified.

2.1. Overview and comparison of pRNGs

A comparison between twelve pRNGs, of which nine are available in MATLAB software [17] and three unavailable [18-20] were realized programmatically. These algorithms were compared in terms of their period and uniformity of distribution, as shown in Table 1.

Because of its high complexity and resource occupancy, instead of using the Mersenne Twister (MT) generator in PLC, which proved to be the best amongst tested generators, it was decided to use a much simpler to implement WH generator, which does not deviate significantly with its results. The current nanosecond was used as the seed. This algorithm is based on using three seeds in range 1-30000

$$s_1 = (171 \cdot s_1) \mod 30269, \tag{1}$$

$$s_2 = (172 \cdot s_2) \mod 30307, \tag{2}$$

$$s_3 = (170 \cdot s_3) \mod 30323, \tag{3}$$

$$r = \left(\frac{s_1}{30269} + \frac{s_2}{30307} + \frac{s_3}{30323}\right) \mod 1, \tag{4}$$

where s_1, s_2, s_3, are the seeds, and r is the output. A test was carried out, comparing PLC-implemented pRNG with

Table 1. Comparison of pRNGs.

Generator	Period	Average value
MT	$2^{19937}-1$	0.50004
SIMD-MT	$2^{19937}-1$	0.50001
MCG	$2^{31}-2$	0.5004
MLFG	2^{124}	0.5001
CMRG	2^{191}	0.4998
Philox 4x32	2^{193}	0.4994
Threefry 4x64	2^{514}	0.5005
SHR3CONG	2^{64}	0.4996
SWB	2^{1492}	0.5004
Middle-square	2^{6561}	0.5043
Wichmann-Hill	2^{50}	0.5004
BBS	2^{41}	0.4996

Table 2. Comparison of WH and MT.

Average value	0.500006	0.499929
Average value error	5.99e-05	7.06e-05
Standard deviation	0.288812	0.288768
Standard deviation error	1.37e-04	9.35e-05
Maximum value	0.99999989	0.99999991
Minimum value	3.42e-7	5.03e-7

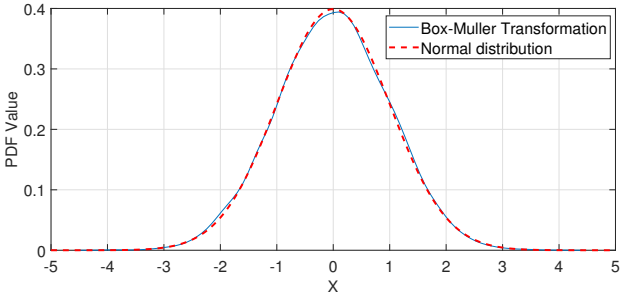


Fig. 1. PDF of normal distribution and obtained values.

MATLAB-based MT for 1000000 generated values. The results in Table 2 show that WH generates numbers very close to the desired mean of 0.5, and does it only slightly worse than MT. It even produced numbers with standard deviation closer to the desired value, which for uniform values in the range 0-1, is equal to $\frac{(1-0)}{\sqrt{12}} \approx 0.288675$.

2.2. Generating normally distributed values

In order to implement an evolutionary optimization algorithm, besides uniformly distributed values, ones with normal deviation are also necessary. For this purpose, the Box-Muller (BM) transformation [21] was used. This operation converts a pair of independent, uniformly distributed numbers U_1 and U_2 to a pair of independent, normally distributed numbers X_1 and X_2, with mean value of 0 and standard deviation of 1. The transformation is given by the formula

$$\begin{cases} X_1 = \sqrt{-2lnU_1}\cos(2\pi U_2), \\ X_2 = \sqrt{-2lnU_1}\sin(2\pi U_2). \end{cases} \tag{5}$$

Fig. 1 depicts the probability density function (PDF) of 10000 values obtained using this method, with a superimposed normal distribution.

3. EVOLUTION STRATEGIES IN PLC

This section contains the description and implementation of the evolutionary algorithm in a PLC, which was later tested with example test functions to search for both minima and maxima, along with execution time comparison in both PLC and PC.

3.1. Program structure

In PLC, a program realizing the (μ,λ) strategy, as described in [22], was implemented in the LAD (ladder logic) programming language. It consists of eight subprograms written in ST, as shown in Fig. 2.

`"Generator_UD"` generates a table of pseudo-random numbers with an uniform distribution. `"Generator_ND"` generates a table of pseudo-random numbers with a normal distribution, using the BM transformation. `"Initialization"` defines all necessary variables and creates an initial population P, consisting of μ individuals. `"Selection"` draws individuals with replacement from the initial population, and puts them in λ-sized temporary population T. `"Recombination"` crosses individuals from population T and puts them in λ-sized descendant population. `"Mutation"` realizes a correction of standard deviation values and modifies values of independent variables. `"Sort"` assigns values of adaptation function to each of individuals and sorts them accordingly, using the bubble-sort algorithm.

After executing all subprograms, the process is repeated up to the stop condition. This structure ensures that each stage of optimization is performed in separate cycles in order to minimize the risk of watchdog timeout.

3.2. Initial tests

The optimizer was tested for its ability to find the global maximum of a two-dimensional function, given by the formula

$$f(\mathbf{x}) = 21.5 + x_1 sin(4+\pi\cdot x_1) + x_2 sin(20\pi\cdot x_2), \quad (6)$$

and limited within the scope of considerations: $x_1 \in [-3, 12.1]$, $x_2 \in [4.1, 5.8]$. The function above consists of many local maxima, and its global maximum is in the point $(11.24, 5.73)$ with value of approximately 38.45. Another test function was a 2-dimensional Rosenbrock function, given by a formula

$$f(\mathbf{x}) = a\cdot(x_2 - x_1^2)^2 + (x_1 - b)^2, \quad (7)$$

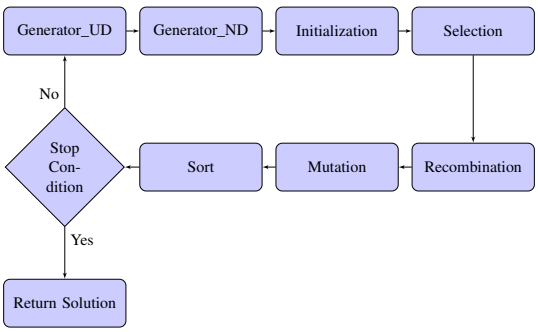
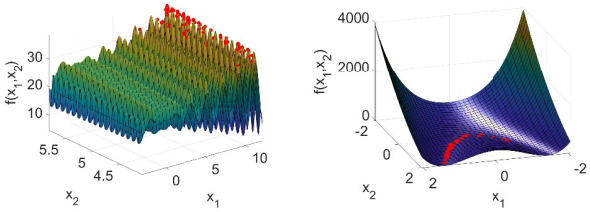


Fig. 2. (μ,λ) strategy block diagram.



(a) (b)

Fig. 3. Optimized functions with overlaid extremes found each iteration.

Table 3. Results of test functions optimization.

	Maximization of (9)	Minimization of (10)
x_1	11.16	0.986
x_2	5.25	0.980
$f(x_1,x_2)$	37.99	0.005
Error	0.46	0.005

with $a = 100$ and $b = 1$, limited within the scope of considerations: $x_1 \in [-2.048, 2.048]$, $x_2 \in [-2.048, 2.048]$. It also consists of many local minima, and its global minimum is in the point $(1, 1)$, with value of 0. The optimizer's population sizes were chosen as: $\mu = 50$ and $\lambda = 100$. After generating first population for the first function (8), the best solution found was in point $(9.23, 5.02)$ with value of 35.72. When repeated 50 times, the best individual from the offspring population was located at the point $(11.22166, 4.40671)$ with value of 34.53 (worse than first generation). This is a result of the stochastic nature of the algorithm and overwriting individuals every iteration. This issue was addressed by assigning the coordinates of the best individual from the offspring population, along with its fitness value and the iteration number in which it was found, to array "Best" every iteration. After a hundred cycles, this array was sorted in ascending (or descending) order based on fitness values.

Fig. 3 shows the plot of the optimized functions presented with overlaid extremes found in each iteration. As evident from the graph, the stochastic nature of the algorithm is highly noticeable. The best maximum is not found in every iteration, but by repeating this process a sufficient number of times, it enables a satisfactory approach towards the global extremum of the objective function. Table 3 shows results for optimizing both test functions.

3.3. Execution time comparison

Tests described above were carried out both in PC and Siemens S7-1200 PLC in order to compare the program processing time. For parameters given above ($\mu = 50$, $\lambda = 100$ and 50 generations), on average it took 20 ms to process the whole program tuning the PID on PC, whereas on PLC, the average execution time was 1 m 44 s.

4. PID TUNING WITH A SIMULATED OBJECT

In this section, the process of finding the optimal PID algorithm is described and applied to a object simulated in the PLC, providing initial test results for practical verification of applied optimizer.

4.1. Object simulated in a PLC

The first test of the optimizer regarding its ability to tune a PID, was carried out entirely in a PLC. A simple object of first-order inertia with delay was being simulated, given by the transfer function

$$G(s) = \frac{1}{s+1} e^{-0.1s}, \tag{8}$$

where s is a complex variable in Laplace transformation. The object has been discretized using a zero-order hold method with a sampling time of 0.01 s, resulting in the discrete transfer function

$$G(z) = \frac{0.00995 z^{-10}}{z - 0.99}, \tag{9}$$

where z is a complex variable in Laurent transformation. From the above transfer function, the differential equation of the object can be determined using the substitution $G(z) = \frac{Y(z)}{U(z)}$, where $Y(z)$ and $U(z)$ are Laurent transformations of input and output signals, respectively

$$y(k+1) = 0.99 y(k) + 0.00995 u(k-10). \tag{10}$$

A diagram of the implemented control system is shown in Fig. 4.

The optimization procedure is performed off-line. The decision variables are the PID settings, and the objective function is the integral of the absolute error (IAE) obtained for each set of tuning parameters. Since the implementation of the algorithm takes place in a PLC, which executes the program in defined cycles, it is necessary to discretize the algorithm

$$u(k) = K_p e(k) + K_i \sum_{0}^{n} (e(k) \cdot T_s) + K_d \frac{e(k) - e(k-1)}{T_s}, \tag{11}$$

where k is current sample, n is total number of samples and T_s is the sampling time. The setpoint is a step reference signal $r(t)$ with a value of 1. The controller is initialized with settings from the first position in the offspring population table O. Then, the control signal, computed by the controller, is applied to the system. After approximately 15 seconds, the performance index value is calculated. The obtained value is assigned to a specific set of settings, and this process is repeated until the entire settings population is tested. Finally, these settings are sorted in ascending order based on their assigned performance index value. The best gains found were as follows: $K_p = 0.9982$, $K_i = 0.9987$, and $K_d = 0.0121$ with J equal to 1.0064.

The program described above yields satisfactory results, allowing it to find PID controller settings that expedite the approach to zero control error while reducing rise and settling times.

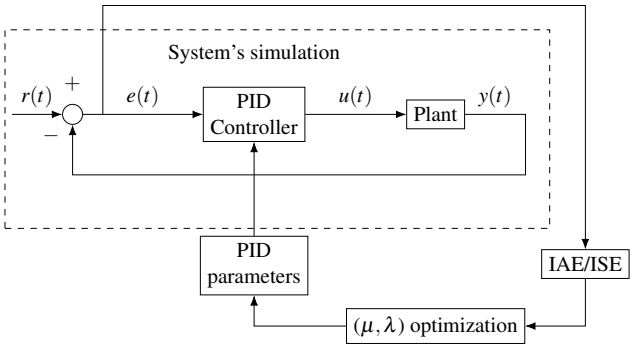


Fig. 4. Scheme of a program for finding optimal PID settings.

4.2. Execution time comparison

Again, the PID-tuning program processing time was compared between PC and Siemens S7-1200 PLC for the same parameters on each device. Since the whole response of a system for each set of PID parameters must be calculated, it took noticeably longer to execute in comparison with evaluation of a function. On average, for a PC it took approximately 230 ms, while on PLC, the average execution time was 19 m 18 s.

5. PID CONTROL WITH A REAL OBJECT

After succesfully applying the (μ, λ) strategy to optimize the parameters of the controller, an attempt was made to use it to control a real object, a modular servo mechanism. This section contains the description of the object, its model derivation and controller tuning and verification.

5.1. Test object

To verify the operation of the optimizing algorithm for the selection of PID controller settings, a servo mechanism provided by INTECO shown in Fig. 5 was utilized. After connecting it to the PLC, this module can transmit control signals to the motor and receive data from the encoder. The control signal is a pulse width modulation (PWM) waveform, with a range of 0-100%.

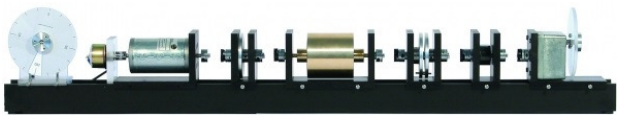


Fig. 5. INTECO modular servo.

5.2. Servo model

The controlled variables in the servo mechanism are velocity and angle. It was decided to control the velocity. The tested object can be represented as a first-order inertial system, given by a transfer function

$$G(s) = \frac{K}{T_b s + 1} e^{-T_d s}, \tag{12}$$

where K is the static gain, T_b is the inertia time constant, ant T_d is the delay time constant. To identify the object, a step response test was carried out, based on the knowledge of step responses of first-order inertial systems [23]

$$K = \frac{\Delta y}{\Delta u}, \quad \frac{y(t) - y_{ss1}}{y_{ss2} - y_{ss1}} = \begin{cases} 0.632 \text{ for } t = T_b, \\ 0.865 \text{ for } t = 2T_b, \\ 0.950 \text{ for } t = 3T_b, \end{cases} \tag{13}$$

where y is the object's response, u is the control signal and y_{ss1}, y_{ss2} are the steady state output values. A PWM step signal of 100% was applied to the servo, resulting in a measurement of a steady-state rotational speed of 162 radians per second. Later, however, it was discovered that above the input signal level of 88% PWM, the steady-state response does not change, hence, the static gain was determined to be $K = \frac{162}{88} = 1.84$. By examining the step response shown in Fig. 6 (note that the signal was applied one second after the measurement started), both time constants were determined to be equal to 1.03 and 0.05, for T_b and T_d accordingly, resulting in a transfer function, which later was discretized with a sample time of 10 ms, which is recommended by the manufacturer, resulting in a differential equation

$$y(k+1) = 0.9903y(k) + 0.01778u(k-5). \tag{14}$$

5.3. PI tuning for the servo

After initial tests, it was decided not to use the differentiating component for regulation, because each time it deteriorated the quality indices compared to PI results. Also, because of the control signal being constrained within the range of 0-88%, an anti-windup filter was applied. The tuning process is similar to the one described in Subsection 4.1, the only difference being the reference trajectory, incorporating both increases and decreases in the setpoint value. In addition to tuning based on the integral of absolute error (IAE) criterion, the integral of squared error (ISE) criterion was also used, resulting in two sets of parameters for further comparisons.

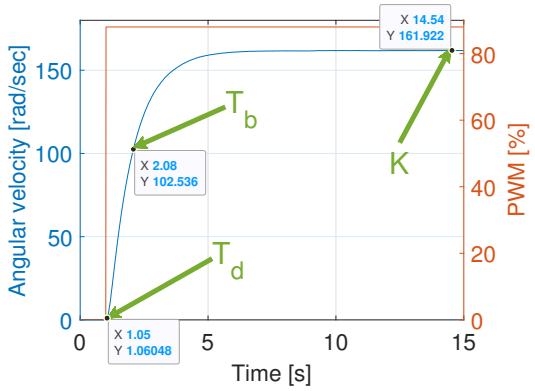


Fig. 6. Plant step response with marked points for its parameter estimation.

5.4. Verification

In order to verify the performance of the developed algorithm, it was decided to compare the obtained quality indices with a PI controller tuned using evolutionary strategies, with a PI controller tuned in Siemens S7-1200 PLC using the built-in tool. The results for tuning the PI controller based on the ISE criterion are shown in Fig. 7, for tuning based on the IAE criterion in Fig. 8, and the results for the controller tuned in Siemens are presented in Fig. 9. These plots represent the reference signal r in $\frac{rad}{sec}$, the output signal y in $\frac{rad}{sec}$ and the control signal u in %. Then, Table 4 represents the comparison of IAE and ISE criteria for both ES-tuned and Siemens-tuned PI controller.

The results of the study indicate that the PI con-

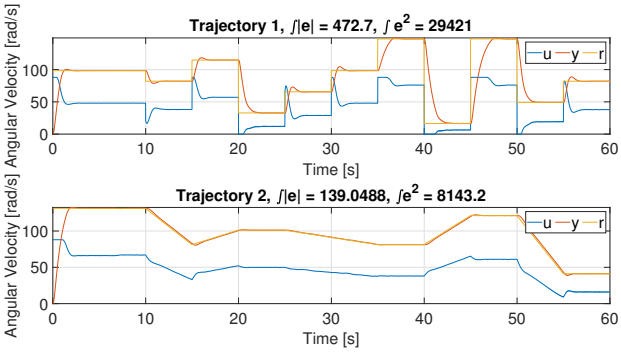


Fig. 7. The results for the controller tuned based on the ISE criterion.

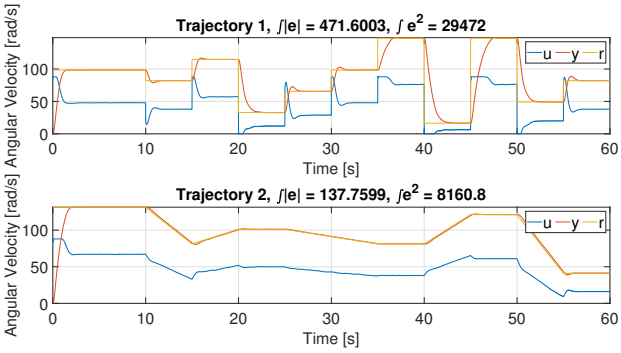


Fig. 8. The results for the controller tuned based on the IAE criterion.

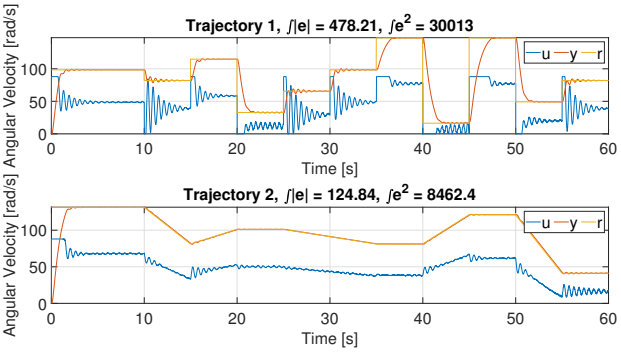


Fig. 9. The results for the controller tuned by Siemens built-in tool.

Table 4. PID verification results.

			1st trajectory		2nd trajectory	
Tuning type	K_p	K_i	IAE	ISE	IAE	ISE
IAE	1.789	3.544	471.6	29472	137.76	8160.8
ISE	1.636	3.457	472.7	29421	139.05	8143.2
Siemens	7.949	3.089	478.2	30013	124.84	8462.4

troller tuned using evolutionary strategies outperformed the Siemens-tuned PI controller in three out of four quality indicators evaluated. This demonstrates the effectiveness of the evolutionary strategy approach in achieving better control performance for the considered system.

These findings highlight the advantages of utilizing evolutionary strategies for PI controller tuning. The results emphasize the potential of evolutionary methods in optimizing control parameters and achieving superior control performance compared to traditional tuning methods.

6. LQR CONTROL WITH A REAL OBJECT

Another attempt on verification of developed optimizer was to utilize the linear-quadratic regulator for control of the position of the servo. As described in Subsection 6.2, the controlled variables in the servo mechanism are its velocity and angle. In order to derive the model for the angle, the state-space representation was used, which allowed for obtaining optimal state feedback gains.

6.1. Discrete LQR algorithm

The full dynamical model of the DC-motor including angle and velocity can be described as a transfer function

$$G(s) = \frac{K}{s(sT_d + 1)}, \tag{15}$$

where K is the static gain, and T_b is the inertia time constant. For simplicity, the delay is omitted in this case. This allows for a discrete state-space representation of the model

$$\begin{cases} x[k+1] = A_D x[k] + B_D u[k], \\ y[k] = C_D x[k], \end{cases} \tag{16}$$

where

$$A_D = e^{AT_s} = \begin{bmatrix} 1 & T_b(1 - e^{-\frac{T_s}{T_b}}) \\ 0 & e^{-\frac{T_s}{T_b}} \end{bmatrix},$$

$$B_D = \begin{bmatrix} K(T_s - T_b(1 - e^{-\frac{T_s}{T_b}})) \\ K(1 - e^{-\frac{T_s}{T_b}}) \end{bmatrix},$$

$$C_D = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

T_s is a sampling time of 10 ms. The task at hand is to find the optimal feedback law

$$u[k] = -Fe[k],\ e[k] = y_r - y[k], \tag{17}$$

where $F = \begin{bmatrix} F_1 & F_2 \end{bmatrix}$ is the vector of feedback gains and y_r is the reference trajectory vector, s.t. it minimizes the cost function

$$J = \sum_{k=0}^{N} (e^T[k] Q e[k] + u^T[k] R u[k]), \tag{18}$$

subject to the state equation (16), where Q and R are positive definitive weight matrices, determining the cost of error and input, respectively.

6.2. Finding the optimal gains

To find the optimal F feedback gain vector, a similar approach was taken as before. For each set of gains being the decision variables, a simulation of system response was generated, and the cost function corresponding to these gains was calculated. The set of considerations for this case was $[0, 1]$ for both gains and matrices Q and R were chosen arbitrarily as $Q = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}$, $R = 50$. The best gains found were as follows: $F_1 = 0.9974$, $F_2 = 0.2702$, with J equal to 214450.

6.3. Verification

After developing a technique to obtain optimal feedback gains for the LQR, it was applied to control the position of the servo mechanism described earlier. Fig. 10 shows the results of the process, where the first plot contains the output angle α of the servo with overlaid reference trajectory α_r, and the second plot depicts the control signal. The control quality is satisfactory, however, it was not possible to eliminate the steady-state error due to the fact that the smallest PWM value from which the motor will respond is 4%. Despite that fact, the steady-state error was not bigger than 1%.

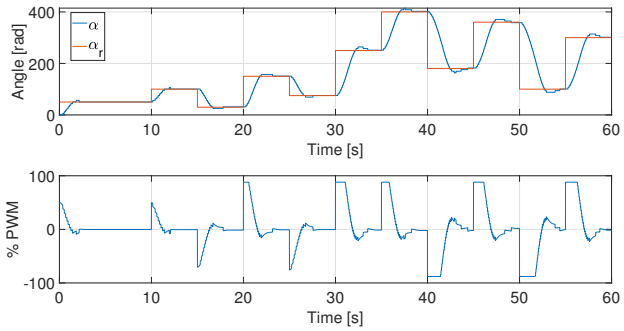


Fig. 10. Results of LQR control.

7. MPC CONTROL FOR DRINKING WATER DISTRIBUTION SYSTEM

Another example of the application of a prepared optimizer based on evolutionary strategies can be the predictive control of balance distribution or supply systems based on forecasted demand and limited generation or storage capacity. Examples of such systems include drinking water distribution systems [24] or control in microgrids [25]. Let's use the example described in [24].

7.1. Control problem description

The control problem addressed there concerns a system for supplying drinking water to a section of a city (simplified for didactic purposes). It is assumed that from the external module of the system a forecast of water demand is provided on a horizon of 24 hours with hourly steps. The system consists of following technological equipment: water intake, pump pumping water to the network, water tank, treatment station and a valve directing water from the pump to the network or to the tank (Fig. 11). It is also assumed that the capacity of the source and the treatment station is greater than the pump. During peak water demand hours, it significantly exceeds the capacity of the pump. It is then necessary to use previously stored water in the reservoir. With the known tariff for electricity, limitations in the form of pump capacity and the dimensions of the tank, optimizer is applied to calculate the control signals of the pump and splitter so that the amount of water directed to the network realizes the externally determined demand at the lowest possible cost.

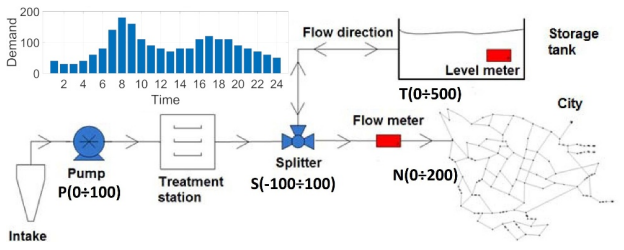


Fig. 11. Case study with part of drinking water distribution systems.

7.2. Optimization problem formulation

Defining the optimization problem can be presented as follows:

$$\min_u f(x) = w_c \sum_{i=1}^{H} (c(i)P(i))^2 + w_e \sum_{i=1}^{H} (D(i) - N(i))^2, \tag{19}$$

such that

$$N(i) = P(i) - S(i), \tag{20}$$
$$T(i+1) = T(i) + S(i), \tag{21}$$
$$T(0) = 200, \tag{22}$$
$$0 \le P(i) \le 100, \tag{23}$$
$$-100 \le S(i) \le 100, \tag{24}$$
$$0 \le T(i) \le 500, \tag{25}$$
$$0 \le N(i) \le 200, \tag{26}$$
$$i \in 1...H, \tag{27}$$

where D is the demand in $[\frac{m^3}{h}]$, N is the flow to network in $[\frac{m^3}{h}]$, P is the pump flow in $[\frac{m^3}{h}]$, S is the splitter flow (positive to tank, negative from tank) in $[\frac{m^3}{h}]$, T is the volume of water in the tank in $[m^3]$, c is the electricity tariff in PLN, H is the control/prediction horizon in [h] and w_c, w_e are cost and error weight factors, respectively.

Previously, in [24], this task was solved in a hierarchical control system, in which the layer determining the optimal trajectories of pump and splitter was realized by a PC with QP optimization software, while the lower layer in the form of PID control realizing the setpoints determined by the trajectory optimizer was built in a PLC. The application of the ES optimizer in the PLC considered in this article makes it possible to achieve the goals of predictive control only with the PLC without any other computational machines.

In each iteration of the MPC controller's operation to obtain a control signal, a nonlinear optimization task with constraints is solved using the developed stochastic optimizer developed. As a result of solving the optimization task, two control variables are obtained simultaneously. Hence, the entire controller is nonlinear and multivariable.

7.3. Adopting ES optimizer for DWDS problem

The main limitation of being able to transfer directly from QP to ES the formulation of the optimization problem is the difficulty of incorporating equality constraints into evolutionary strategies. A literature review was conducted for the inclusion of balance equations in optimization calculations in evolutionary strategies. Review article [26] was particularly useful. An approach that is variously called in different sources was attempted. In [14] we can find "repair algorithms" or "assurance of feasibility". Each of these techniques has been used to some extent in this article.

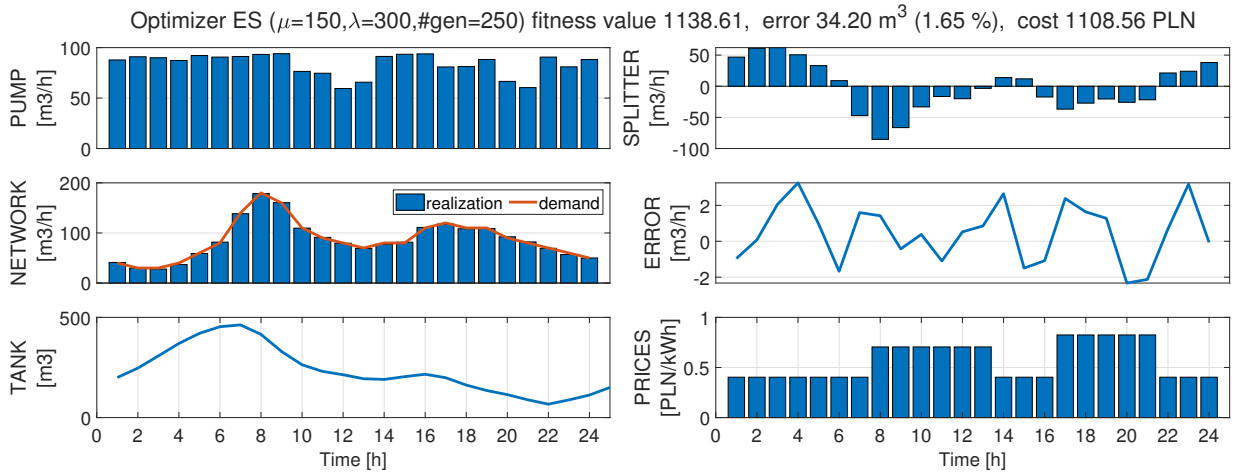


Fig. 12. ES optimizer results.

Table 5. Chromosome of an individual for ES optimizer.

P_1	P_2	...	P_H	S_1	S_2	...	S_H	σP_1	σP_2	...	σP_H	σS_1	σS_2	...	σS_H

Due to the specificity of the task, the following method of coding the chromosome was adopted. When we assume that H is the prediction horizon, the chromosome will take the dimension $4 \cdot H$ as in Table 5.

7.4. ES optimization results and MPC scheme

Example results of solving the water supply task using an evolutionary optimizer are shown in Fig. 12. Observing the results, we see a small and acceptable supply error, maintaining the process sizes of the pump, splitter, reservoir condition within the a priori specified ranges, cost-effective management of pump operation.

7.5. Time of calculation issue

An important aspect of an optimizer that affects its usefulness is its computation time. It is well know that evolutionary strategies have significantly lower speed of convergence than gradient optimizers. It is also known that the PLC has much less computing power than the PC.

In view of the assumption that $\lambda = 2\mu$ and a simulation step of 1 h, in fact there are three variables that affect the computation time and the value of the objective function. These are the population size - μ, the number of generations of the algorithm - NumGen and the control horizon - H. Extensive search calculations were performed on the PC to estimate the value of the objective function and the time required for calculations for different values of μ and NumGen. The results for a control horizon of 24 h are shown in Fig. 13. Since the optimizer operates in a stochastic manner, each scenario was counted 10 times and averaged values are shown in the figures. Then, selecting the results of the objective function below the conventionally determined level of 1200, it was checked which combination of parameters offers such a result in the shortest time. For comparison, several combinations with different values of parameters are shown in Table 6.

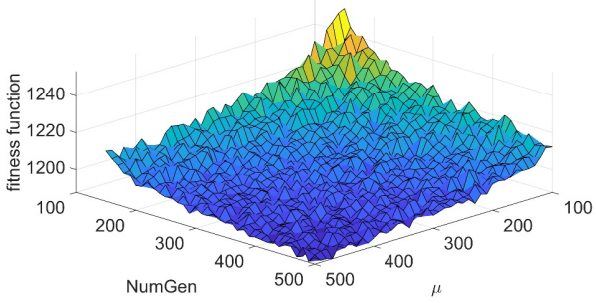


Fig. 13. Fitness function values depending on number of individuals and iterations of the algorithm.

Table 6. Time needed for ES optimizer calculation for different μ, H and NumGen.

μ	NumGen	H	Fitness*	S7-1500	PC i9
200	300	24	1156.1	58 min 49 s	3.7 s
100	250	24	1199.3	23 min 48 s	1.6 s
100	250	12	727.1	12 min 7 s	1.28 s
100	250	6	388.9	6 min 28 s	0.8 s

* comparable only for the same control horizons H

7.6. MPC RH verification

It should be noted that the horizon of 24 steps is quite long and shorter ones, e.g., 12 or 6 hours, are usually sufficient. The results of predictive control based on a one-time solution per day (H24, QP) and scenarios with the receding horizon (RH) strategy for different prediction horizons (24 h, 12 h, 6 h, and 3 h) are shown in Fig. 14 and Table 7. The used weights factors were $w_e = 1000$, $w_c = 1$. The

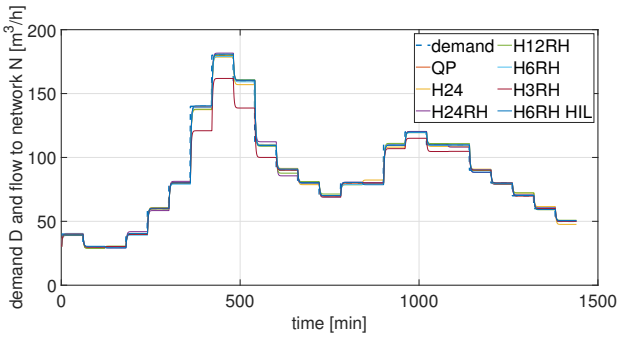


Fig. 14. MPC control strategies based on ES optimizer with different horizons and with/without receding horizon mechanism.

Table 7. Results of MPC control.

Method	Error [m^3]	Error [%]	Cost [PLN]
QP	17.5	0.85	1081.1
H24	51.4	2.48	1105.5
H24RH	51.5	2.49	1149.1
H12RH	33.8	1.64	1156.2
H6RH	23.8	1.15	1138.0
H3RH	71.0	3.43	1124.9
H6RH HIL	27.0	1.31	1126.6

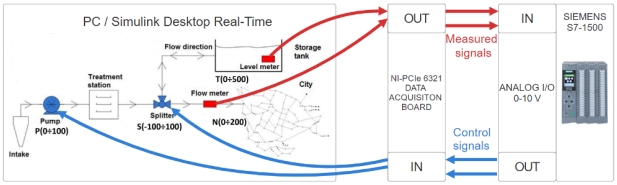


Fig. 15. HIL verification structure.

other shown analysis results were obtained using the ES optimizer.

The results of the QP optimizer are undoubtedly the best. However, the results obtained with the ES optimizer are not much worse and certainly useful for the described task. Actually, the only unacceptable scenario turned out to be the one with a horizon length of 3 h. The scenario with a horizon of 6 h, for which the optimization calculations take less than 7 minutes in the PLC, fulfils its purpose. Verification for this scenario was also done in the hardware loop structure as in Fig. 15. The model was simulated in MATLAB/Simulink, and signals exchanged through the acquisition card.

8. CONCLUSIONS

The great advantages of ES are the lack of need to know the gradient function and the excellent scalability to multiple dimensions. The consequence of such ES is computational redundancy. This paper presents the process of building an optimizer for the PLC platform based on evolutionary strategies.

Taking into account the trade-off between performance quality and ease of implementation, the WH method for obtaining homogeneous distributions and the BM transformation method for obtaining normal distributions based on the homogeneous ones were selected, implemented and verified. Implementation of the optimization algorithms under PLC conditions had to take into account the limited computing power and memory compared to a PC.

The correctness of the implementation was verified on several standard multidimensional test functions. mThe developed estimator was tested with a real device, which was a DC motor. PID control was used for speed control. The results of the confrontation with the tuning mechanism built into the Siemens PLC engineering software show virtually consistent results in terms of control quality. Control of the motor shaft angle was implemented using LQR control. The optimizer was used to determine the optimal state feedback gain matrix (taking into account Q and R matrices) of this controller. Verification indicated the correctness of this solution.

The most challenging example of the use of the developed ES optimizer was applied to the MPC control for a simulated section of the water supply system. A repetition strategy was used, i.e., the optimization task was solved every hour and only the control set for the first hour of the entire control horizon was applied to the object. The results obtained are comparable to calculations with QP optimization, confirming the application potential of such an optimizer.

The functionality of the optimizer developed on the PLC platform has been comprehensively verified and validated. The key issue is the calculation time, which is significantly longer than for PC and gradient optimizers.

CONFLICT OF INTEREST AND DATA AVAILABILITY STATEMENT

All authors have no conflicts of interest. The program code for tuning PID and LQR controllers and for the MPC controller in the form of a Siemens TIA Portal V18 PLC project has been attached to the content of the article as supplementary material. For the convenience of readers, text versions of individual procedures and a description of the program structure have also been prepared.

REFERENCES

[1] M. He and C. Chen, "An effective online computation scheme for constrained model predictive control in embedded systems," *Proc. of 6th World Congress on Intelligent Control and Automation*, vol. 2, pp. 6699-6703, 2006.

[2] B. Käpernick, "PLC implementation of a nonlinear model predictive controller," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 1892-1897, 2014.

[3] P. Krupa, D. Limon, and T. Alamo, "Implementation of model predictive control in programmable logic controllers," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 1117-1130, 2021.

[4] G. Valencia-Palomo and J. Rossiter, "Efficient suboptimal parametric solutions to predictive control for PLC applications," *Control Engineering Practice*, vol. 19, no. 7, pp. 732-743, 2011.

[5] F. Delfino, M. Rossi, R. Minciardi, and M. Robba, "An optimization based architecture for local systems managed by PLC devices," *Proc. of IEEE International Symposium on Systems Engineering (ISSE)*, pp. 17-22, 2015.

[6] J. Tarnawski, P. Kudełka, and M. Korzeniowski, "Advanced control with PLC-code generator for aMPC controller implementation and cooperation with external computational server for dealing with multidimensionality, constraints and LMI based robustness," *IEEE Access*, vol. 10, pp. 10597-10617, 2022.

[7] A. Purohit and J. Buch, "Evaluation of optimization solvers on programmable logic controller," *Proc. of IEEE Conference on Control Applications (CCA)*, pp. 533-538, 2015.

[8] D. Kouzoupis, A. Zanelli, H. Peyrl, and H. Ferreau, "Towards proper assessment of QP algorithms for embedded model predictive control," *Proc. of European Control Conference (ECC)*, pp. 2609-2616, 2015.

[9] C. Schreppel and J. Brembeck, "A QP solver implementation for embedded systems applied to control allocation," *Computation*, vol. 8, no. 4, p. 88, 2020.

[10] M. Mohamed, M. Ghazali, S. Idrus, and N. Wahab, "Optimization through artificial neural network on a programmable logic controller for a sludge drying plant," *Proc. of IEEE 9th International Colloquium on Signal Processing and Its Applications*, pp. 103-105, 2013.

[11] S. Howimanporn, S. Chookaew, and W. Sootkaneung, "Implementation of PSO based gain-scheduled PID and LQR for DC motor control using PLC and SCADA," *Proc. of International Conference on Control and Robots (ICCR)*, pp. 52-56, 2018.

[12] J. Lu, "Design of PLC intelligent control system based on improved genetic ant colony algorithm," *Proc. of IEEE 2nd International Conference on Electronic Technology, Communication and Information (ICETCI)*, pp. 1185-1188, 2022.

[13] H. Schwefel, *Evolution and Optimum Seeking: The Sixth Generation*, John Wiley & Sons, Inc., 1993.

[14] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin, Heidelberg, pp. 372-373, 1996.

[15] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addition-Westly, Reading MA, 1989.

[16] T. Bäck, C. Foussette, and P. Krause, *Contemporary Evolution Strategies*, Springer, Berlin, vol. 86, 2013.

[17] "Creating and Controlling a Random Number Stream,", *Mathworks*, https://www.mathworks.com/help/matlab/math/creating-and-controlling-a-random-number-stream.html, Accessed on 2023-11-20.

[18] V. Neumann, "Various techniques used in connection with random digits," *Notes By GE Forsythe*, pp. 36-38, 1951.

[19] B. Wichmann and I. Hill, "Algorithm AS 183: An efficient and portable pseudo-random number generator," *Journal of The Royal Statistical Society. Series C (Applied Statistics)*, vol. 31, pp. 188-190, 1982.

[20] L. Blum, M. Blum, and M. Shub, "A simple unpredictable pseudo-random number generator," *SIAM Journal on Computing*, vol. 15, pp. 364-383, 1986.

[21] G. Box and M. Muller, "A note on the generation of random normal deviates," *The Annals of Mathematical Statistics*, vol. 29, pp. 610-611, 1958.

[22] J. Arabas, *Wykłady Z Algorytmów Ewolucyjnych*, Wydawnictwa Naukowo-Techniczne, 2004.

[23] K. Åström and T. Hägglund, *Advanced PID Control*, ISA-The Instrumentation, Systems, and Automation Society, 2006.

[24] J. Tarnawski, T. Rutkowski, and A. Cimiński, "Implementation of hierarchical control of drinking water supply system: Didactic project - Computer controlled system," *Proc. of 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*, pp. 831-836, 2017.

[25] A. Kowalczyk, A. Włodarczyk, and J. Tarnawski, "Microgrid energy management system," *Proc. of 21st International Conference on Methods and Models in Automation and Robotics (MMAR)*, pp. 157-162, 2016.

[26] O. Kramer, "A review of constraint-handling techniques for evolution strategies," *Applied Computational Intelligence and Soft Computing*, vol. 2010, pp. 1-19, 2010.

Kajetan Zielonacki was born in Gdynia, Poland, in 2000. He received his B.Eng. degree in automatics and robotics from the Gdańsk University of Technology in 2023. His research interests include programmable logic controllers, optimal control, and evolutionary algorithms.

Jarosław Tarnawski was born in Gdańsk, Poland, in 1974. He received his M.Sc. and Ph.D. degrees from the Gdansk University of Technology. He is currently an Assistant Professor with the Department of Electrical Engineering, Control Systems and Computer Science, Gdańsk University of Technology. His research interests include mathematical modeling, identification, optimization, and hierarchical control systems.