ISSN 1846-6168 (Print), ISSN 1848-5588 (Online)
https://doi.org/10.31803//tg-20201119185015

Original scientific paper
Received: 2020-11-19, Accepted: 2021-07-27

Genetic Algorithm-Based Parametrization of a PI Controller for DC Motor Control

Marko Mavrinac, Ivan Lorencin*, Zlatan Car, Mario Šercer

Abstract: This paper analyses the application of a genetic algorithm (GA) for the purpose of designing the control system with separately excited DC motor controlled according to the rotor angle. The presented research is based on the utilization of a mathematical model designed with separate electrical and mechanical sub-systems. Such an approach allows fine-tuning of PID controllers by using an evolutionary procedure, mainly GA. For purpose of PID tuning, the new fitness function which combines several step response parameters with the aim of forming a unique surface which is then minimized with a genetic algorithm. From the results, it can be seen that the elitism-based algorithm achieved better results compared to the eligibility-based selection. Such an algorithm achieved a fitness value of 0.999982 resulting in a steady-state error of 0.000584 rad. The obtained results indicate the possibility of applying a GA in the parameterization of the PID controller for DC motor control.

Keywords: Control system design; Fitness score; Genetic algorithm; PID control; separately excited DC motor

1 INTRODUCTION

Automatic control represents one of the key elements of modern society. Feedback regulation represents a standard procedure in automatic control [1, 2]. Such a configuration is characterized by controller utilization [3]. In most practical applications, a simple PID controller represents an optimal solution [4]. PID controller parametrization, alongside mathematical modelling, represents one of the key challenges in the design of an automatic control system [5]. The classical approach used for PID parametrization and tuning is based on experimental design and mathematical modelling [6]. Such an approach requires expertise in the field of automatic control and electric drives [7]. In addition, such an approach can be time-consuming and error-prone [8].

Another approach to PID parametrization is the utilization of evolutionary algorithms, mainly GA. Evolutionary computation, alongside artificial intelligence (AI) and machine learning (ML) represents one of the most propulsive fields of science and technology with application in various fields ranging from propulsion systems [9, 10], through medicine [11-13] to satellite surveillance [14].

GA represents a meta-heuristic algorithm that is based on Darwin evolution theory [15]. This algorithm has a long history of utilization in various fields of science and engineering, ranging from robotics [16], through modelling of energy systems [17] to maritime affairs [18]. Such an approach can offer a simpler and more effective method for PID parametrization.

Following the presented facts, the aim of this paper is to present a GA-based method for PID controller parametrization that will be used for parametrization of 3 PI controllers used in the control of separately excited DC motor according to the rotor angle. Controllers will be parametrized by using different variations of GA. Results achieved with each of these GA methodologies will be used in the evaluation of, not only GA but also the PID regulation itself.

2 DESCRIPTION OF THE CONTROL SYSTEM

The mechanical sub-system defines the rotor angle in dependency of motor and load torques. The rotor angle can be defined by the angular speed of the rotor as its integral. By using presented formalism, the mechanical relation can be defined as [19]:

$$\frac{\mathrm{d}\omega}{\mathrm{d}t} = \frac{1}{J_\mathrm{u}} \cdot (M_\mathrm{m} - M_\mathrm{t}), \tag{1}$$

where J_u represents the total moment of inertia defined as a sum of motor and load inertia, M_t represents a load torque including friction, and M_m represents the shaft torque. Shaft torque can be defined as [20]:

$$M_\mathrm{m} = k_\mathrm{m} \psi_\mathrm{ex} i_\mathrm{a}, \tag{2}$$

where k_m represents the motor constant, ψ_ex represents the flux linkage and i_a represents the armature current. For motor operation at rated values, flux linkage can be defined as a constant [21]. For this reason, Eq. (2) can be rewritten as a linear equation, yielding [22]:

$$M_\mathrm{m} = K i_\mathrm{a}. \tag{3}$$

Such a mathematical model can be represented with its block scheme used for simulation in Simulink, presented in Fig. 1.

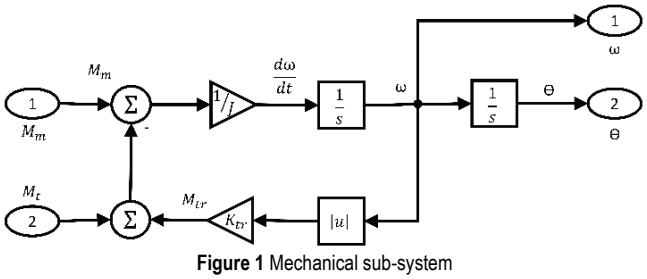

Figure 1 Mechanical sub-system

It must be noticed that such a model is constructed by using the absolute value of rotor angular speed. This property is introduced in order to annul the influence of rotation

direction (clockwise or counterclockwise) on the control system and its mathematical behavior.

The differential equation that describes the behavior of the electrical subsystem of a DC motor can be defined as:

$$\frac{\mathrm{d}i_\mathrm{a}}{\mathrm{d}t} = \frac{1}{L_\mathrm{a}} \cdot (U_\mathrm{a} - i_\mathrm{a} R_\mathrm{a} - e), \tag{4}$$

where U_a represents armature voltage, L_a represents armature inductance, R_a represents armature resistance and e represents an induced voltage. Such a voltage can be defined with [21]:

$$e = k_\mathrm{e} \psi_\mathrm{uz} \omega, \tag{5}$$

where ψ_uz represents the excitation flux linkage. It should be noted that, as it is in the case of armature flux linkage, in this case, voltage equation can be derived to linear, yielding:

$$e = K\omega. \tag{6}$$

It should be noticed that the constant K is the same constant as it is in the case of Eq. (3).

As it is in the case of the mechanical sub-system, the electrical sub-system is also designed and simulated in Simulink. A block-diagram used for the simulation of the mechanical sub-system is presented in Fig. 2.

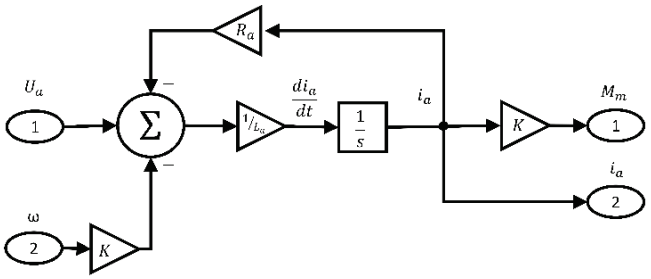

Figure 2 Electrical sub-system

The control system used in the control of DC motor rotor angle is based on three control loops:

- loop for armature current control,
- loop for rotational speed control and
- loop for rotor angle control.

Each of the aforementioned control loops must be designed by using a separate controller. In the case of motor control, due to the derivative nature of its behavior, it is necessary to include only proportional and integrational components of the PID controller (PI controller). The output value of such PI controller can be defined as:

$$y(t) = K_\mathrm{p} e(t) + K_\mathrm{i} \int_0^t e(\tau)\mathrm{d}\tau, \tag{7}$$

where K_p represents the proportional coefficient and K_i represents the integrational coefficient of a PI controller. The whole system for DC motor control composed of mechanical sub-system, electrical sub-system and PI controllers is presented in Fig. 3. This block diagram is used for simulation of the control system in Simulink.

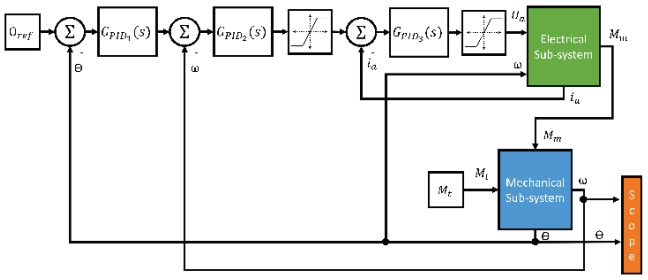

Figure 3 Block-diagram of whole regulation system

For purposes of this research, the control system was simulated and parametrized in Simulink and by using custom Python program. In both cases, DC motor model is designed by following the above presented mathematical formalism and by using parameters presented in Tab. 1.

Table 1 DC motor parameters used for the design of the control system

Rated armature voltage (V)	420
Rated power (W)	31500
Rated armature current (A)	90
Armature resistance (Ω)	0.65
Armature inductance (H)	0.0066
Rated speed (min^{-1})	995
Rated torque (Nm)	302
Rated induced voltage (V)	361.5
Rotor inertia (kgm^2)	3.5
Friction coefficient (Nms/rad)	0.05

3 DESCRIPTION OF GA USED FOR PID PARAMETRIZATION

In this chapter of the article, a brief description of used GA will be presented. The description will include a description of population initialization, variation procedures, and fitness determination. All three parts will be enriched with block diagrams.

The goal of each GA is to generate generations of quality individuals from a set of randomly generated solutions [23]. New generations are generated from selected individuals on which variation procedures are performed. Solution quality is determined by using the fitness function [24].

3.1 Population Initialization

The initial phase of the genetic algorithm is population initialization. Depending on the default size population, the genetic algorithm generates a series of solutions with random parameters. At this stage, it is possible to use heuristic methods of creating solutions to obtain a better initial population. The flowchart of the population initialization procedure is presented in Fig. 4. Population created by the heuristic method will have a higher average of the suitability ratings of the individuals than the one created randomly, but there is a risk of premature convergence of the algorithm, less diversity of results due to favoring certain individuals from the beginning of the algorithm execution.

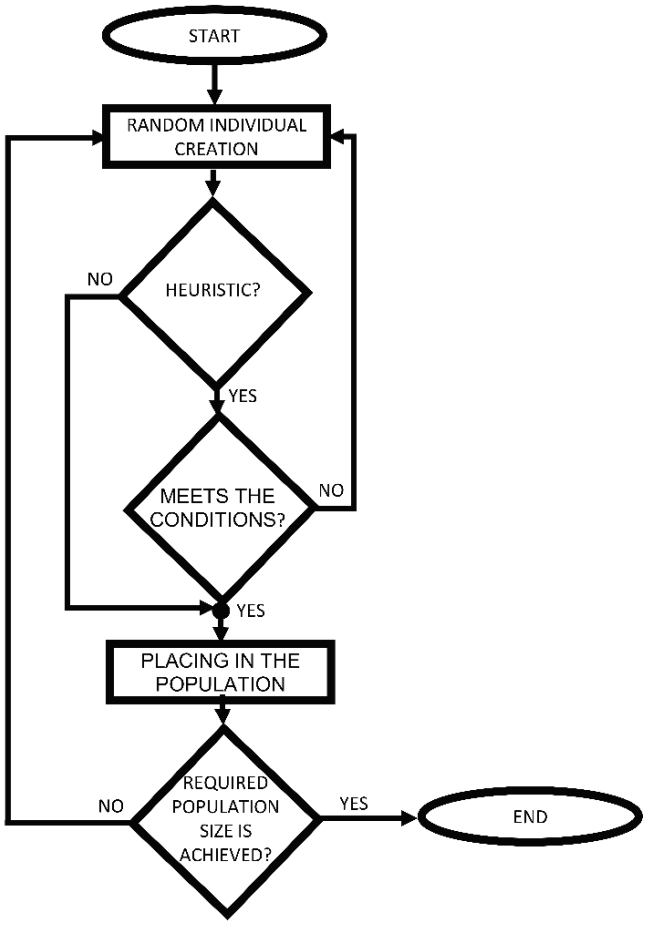

Figure 4 The flowchart of the population initialization procedure

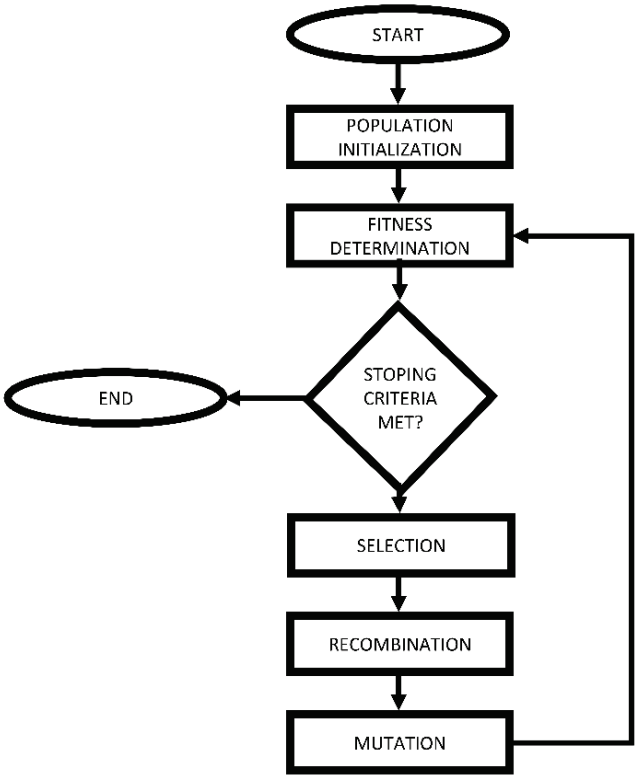

Figure 5 The flowchart of GA procedure

3.2 Variation Procedures

Population initialization is followed by the application of selection, recombination, and mutation to create new generations of the initial population. Each new individual is assigned an appropriate fitness score, and depending on the stopping conditions, the mean fitness score of the population is calculated. If the new generation meets the stop conditions, the algorithm lists it as the optimal solution and stops it. Additional checks on solutions can determine whether the set of algorithm solutions is of appropriate quality. The low quality can be ascribed to premature convergence of the algorithm or a poorly defined fitness function. The flowchart of the genetic algorithm is shown in Fig. 5.

3.3 Fitness Function Definition

In this paper, as a quality measure of the DC motor control system, step response quality indicators such as:

- Rise time,
- Peak time,
- Settling time,
- Overshoot and
- Steady-state error

will be used during the definition of the fitness function. The goal of the GA is to minimize all above-presented indicators. Such an approach will result in a step response that closely follows the reference signal. The fitness function used in this research is designed to reduce several goals to only one goal. This approach is achieved by defining the area between the reference value and step response signal. In this way, a minimization of above-presented quality indicators is achieved. An example of such an area is presented in Fig. 6.

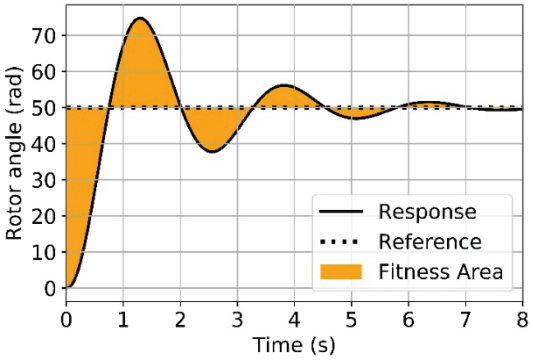

Figure 6 Example of a step response with defined fitness area

Depending on the fitness function and the system model used for the determination fitness of the individual, the algorithm needs to be optimized to minimize run time without affecting its functionality. The chromosome save function returns the parameters of an individual to a string record, and the fitness function uses this record to compare the genetic code of an individual with an already calculated fitness. The calculated fitness values are stored in a variable.

In this case, the keys are variable correspond to the genetic code of individuals, and their values correspond to

calculated fitness values. The optimized fitness determination process is shown in Fig. 7.

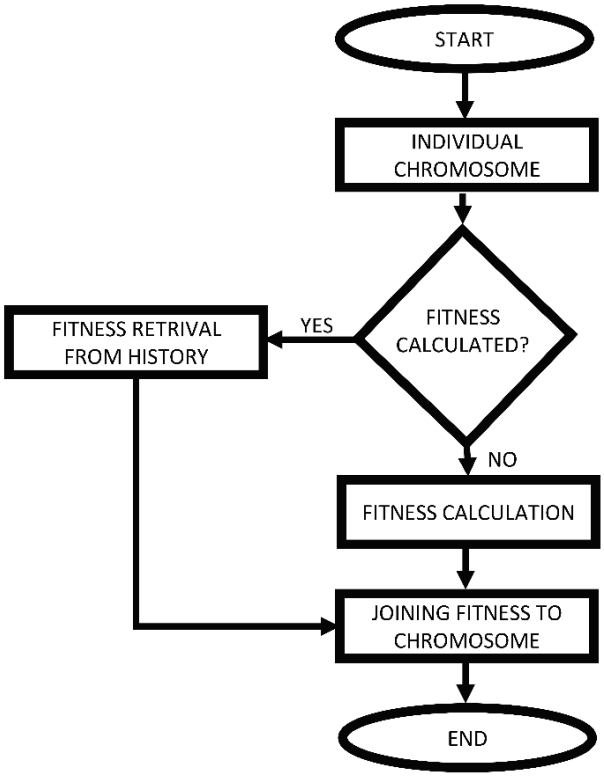

Figure 7 Flowchart of the optimized fitness function

4 RESEARCH METHODOLOGY

In order to define GA parameters that will produce the best performances, from both GA and control system standpoints, multiple GAs will be executed and compared. For all GAs, fitness values (individual and average) will be presented alongside control system quality indicators. The aforementioned measures will be used for the evaluation of different GA methodologies.

A comparison of the two different GAs will be based on:

- population size,
- mutation probability,
- selection method, and
- relative tolerance of the fitness function.

For each comparison, statistics related to response quality indicators, algorithm execution time, and success rate will be presented. The algorithm is considered successful if the rotor angle in the steady-state is the inside limit of 2% of the rotor angle reference value. From this rule, a success rate will be calculated for each variation of GA.

5 RESULTS AND DISCUSSION

Changing population size directly affects execution time, convergence rate, and in many cases also the quality of the final solutions, PI controllers. The impact on the quality of the ultimate solutions is the largest in GAs with selection by rank. The root of this property lays in the fact that small populations often die out before convergence due to the relatively high probability of not selecting population members for recombination.

For elitism, with an algorithm with a larger population, the settling time is somewhat shorter, and the overshoot is noticeably lower. This is also reflected in the fitness of the whole generation. It can be observed that the execution time is three times longer for the population with a double number of members. Similar results are achieved if the rank selection is utilized, as shown in Tab. 2.

Table 2 Comparison of results achieved with GA with elitism for different population sizes

	GA configuration			
	P1	P2	P3	P4
Population size	10	20	20	30
Selection	Elitism	Elitism	Rank	Rank
Duplicates allowed	No	No	No	No
Mutation Probability	0.1	0.1	0.2	0.2
Rise time (s)	1.3776	1.3414	1.3171	1.3334
Peak time (s)	2.9854	3.1172	3.0395	3.1731
Settling time (s)	2.7765	2.2468	2.8522	2.2951
Overshoot (%)	2.0343	0.3816	2.0939	0.4944
Steady-state error (rad)	0.1219	0.0691	0.1189	0.0712
Best individual fitness	0.97078	0.99982	0.95238	0.99976
Best population fitness	0.96954	0.99982	0.94272	0.99921
Execution time (s)	330.85	1047.39	739.55	1185.58
Success rate	98	100	82	98

Increasing the probability of mutation indirectly increases the diversity of the population within the algorithm. Genetic algorithms whose population size is less than ten individuals often will not converge close to the optimal solution if the probability of mutation is not high enough. For these reasons, for GAs with rank selection and elitism for different values of mutation probabilities are 0.1 and 0.2. From Tab. 3 it can be seen that there are no significant differences between GAs. A difference can only be noticed for the case of overshoot, which is significantly lower when elitism is used. On the other hand, there is no significant difference in other relevant quality measures. The execution time of the algorithm is shorter for a higher mutation probability. Furthermore, it can be noticed that the algorithm with a higher probability of mutation converges much faster.

Table 3 Comparison of results achieved with GA for different mutation probabilities

	GA configuration			
	M1	M2	M3	M4
Population size	15	15	25	25
Selection	Elitism	Elitism	Rank	Rank
Duplicates allowed	No	No	No	No
Mutation Probability	0.1	0.2	0.1	0.3
Rise time (s)	1.3688	1.3482	1.2461	1.3819
Peak time (s)	2.9686	2.7948	2.9534	2.9725
Settling time (s)	2.2492	2.2414	2.8668	2.7393
Overshoot (%)	0.4902	0.3874	3.2741	1.1643
Steady-state error (rad)	0.0645	0.0639	0.1216	0.0951
Best individual fitness	0.99981	0.99986	0.98450	0.98185
Best population fitness	0.99966	0.99974	0.98046	0.980865
Execution time (s)	648.82	572.08	956.57	870.61
Success rate	100	100	94	96

Tab. 4 shows that GA without copies generally has better results. Furthermore, it is necessary to consider the fact that the diversity of the population, in this case is significantly lower, therefore the average fitness also depends on the size of the population. It can be seen that in the case of rank selection, the influence of duplicates is significantly higher. This property is a consequence of the premature convergence of GA.

Table 4 Comparison of results achieved with GA with and without duplicate elimination

	GA configuration			
	D1	D2	D3	D4
Population size	15	15	20	20
Selection	Elitism	Elitism	Rank	Rank
Duplicates allowed	Yes	No	Yes	No
Mutation Probability	0.1	0.1	0.1	0.2
Rise time (s)	1.3394	1.334	1.1993	1.3131
Peak time (s)	3.1256	2.8815	3.0029	2.9394
Settling time (s)	2.4374	2.3544	3.4517	2.8198
Overshoot (%)	1.0892	0.4524	6.3599	1.4884
Steady-state error (rad)	0.1152	0.0739	0.2046	0.22
Best individual fitness	0.998615	0.999511	0.914317	0.9913
Best population fitness	0.99856	0.99937	0.90143	0.99021
Execution time (s)	385.64	624.39	418.19	738.49
Success rate	100	100	84	96

In the later stages of GA where the solutions approach locally optimal points, there is less and less room for improvement or the difference in average fitness between two generations is getting smaller. When the difference in fitness becomes smaller than defined relative tolerance, the stop condition is satisfied, and the genetic algorithm ends. With the smaller tolerance parameter, the genetic algorithm will have multiple generations or longer performance, but also more accurate results. Relative fitness tolerance has an impact solely on later stages of the genetic algorithm. Tab. 5 shows the differences between the solution quality of the algorithms of different relative tolerances. GA with less relative tolerance has a longer execution time, but also better results.

Table 5 Comparison of results achieved with GA for different fitness tolerance

	GA configuration	
	T1	T2
Population size	15	15
Tolerance	0.01	0.000001
Selection	Elitism	Elitism
Duplicates allowed	No	No
Mutation Probability	0.1	0.1
Rise time (s)	1.3699	1.3704
Peak time (s)	3.0255	3.0256
Settling time (s)	2.4227	2.2318
Overshoot (%)	1.0435	1.0436
Steady-state error (rad)	0.1179	0.0414
Best individual fitness	0.99188	0.99994
Best population fitness	0.99288	0.999919
Execution time (s)	505.70	1086.36
Success rate	98	100

Selection proportional to fitness, in this case, cannot be used in the initial steps of the algorithm since each individual would have an extremely low probability of recombination. Therefore, the parameter for proportional selections was introduced. A parameter of 0.5 was chosen since then the smallest number of individuals will have secured or disabled recombination. Since this type of selection favors better solutions, it often reduces the diversity population, and thus the quality of the end solutions.

Tab. 6 shows the difference between fitness and the end of algorithm execution. A small saving in performance time, caused by a selection that is proportional to fitness, can also be noticed. As in the case of selection by rank, at small population sizes, selection proportional to fitness did not bring great advantages.

Table 6 Comparison of results achieved with GA for different selection methods

	GA configuration		
	S1	S2	S3
Population size	15	15	15
Selection	Elitism	Rank	Proportional to fitness (0.5)
Duplicates allowed	No	No	No
Mutation Probability	0.1	0.1	0.1
Rise time (s)	1.4024	1.2036	1.4008
Peak time (s)	2.8408	2.8488	2.9829
Settling time (s)	2.4048	3.5436	2.2652
Overshoot (%)	0.6001	5.9579	0.2493
Steady-state error (rad)	0.0731	0.1781	0.0571
Best individual fitness	0.99446	0.7367	0.98783
Best population fitness	0.99439	0.703505	0.98759
Execution time (s)	679.31	591.94	585.46
Success rate	100	50	96

When all achieved results are compared, it can be noticed that the results higher of 0.95 are achieved in the majority of cases. It is interesting to notice that the majority of GA configurations that have achieved high fitness values are designed by using elitism. Furthermore, it can be noticed that no significant changes in fitness values when larger populations are used. This property can particularly be noticed in cases of configurations marked with *M*. All described facts are presented in Fig. 8.

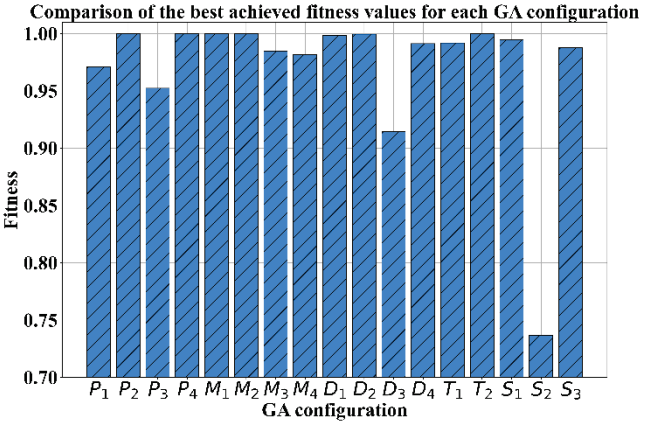

Figure 8 Comparison of best individual fitness values achieved with presented GA configurations

Similar results are achieved when population averages are compared. In this case, it can be noticed that the majority of the highest results are achieved with GA configurations designed with selection based on elitism. Furthermore, it can be noticed that there is no significant difference between best

fitness and population averages. From this fact, it can be seen that by using such GA configurations, multiple individuals with high-quality performances are achieved. All above-mentioned facts are presented in Fig. 9.

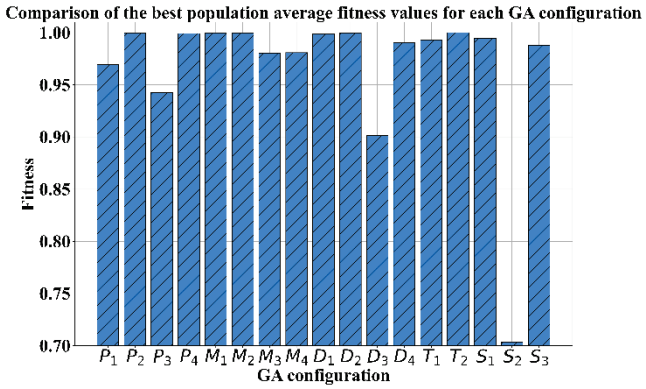

Figure 9 Comparison of best population average fitness values achieved with presented GA configurations

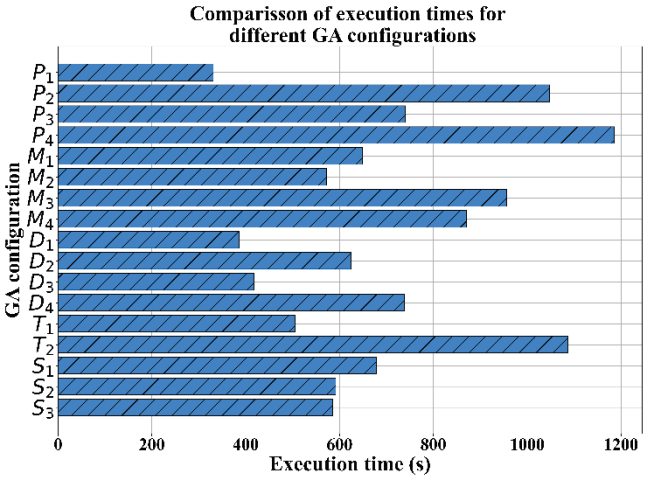

Figure 10 Comparison of execution times for all presented GA configurations

Table 7 GA parameters with which the best result was achieved

Selection	Elitism
Mutation probability	0.2
Tolerance	0.000002
Population size	26
K_{p1}	1.05
K_{i1}	0
K_{p2}	8.65
K_{i2}	9.94
K_{p3}	0.42
K_{i3}	12.81
Fitness score	0.999982
Steady-state error (rad)	0.000584

When execution times are compared, it can be noticed that GA configurations designed with larger populations are demanding a longer execution time. It can also be noticed that GA with lower tolerance is also demanding a longer time for execution. This property has its roots in the fact that configurations with lower tolerance will take more iterations to converge. When observing the influence of the selection method, it is interesting to notice that elitism-based GA will take a longer time to execute. All aforementioned facts are presented in Fig. 10.

According to the presented methodology, the optimal configuration is chosen. This configuration is chosen to balance the best fitness, average population fitness, and execution time. This configuration is presented in Tab. 7.

6 CONCLUSION

From the presented results it can be concluded that there is a possibility for utilization of GA for parametrization of the control system based on DC motor. It can be noticed that the highest performances were achieved if the selection procedure based on elitism is used. Furthermore, the performances with a fitness score up to 0.999982 were achieved if the GA with mutation probability of 0.2 and tolerance of 0.000002. Presented GA was executed by using a population of 26 individuals. GA configuration described above has resulted in a steady-state error of 0.000584 rad. It can be noticed that such a configuration did achieve high accuracy, and at the same time, lower oscillations of the step response.

A main advantage that such an approach offers is a possibility for utilization of such algorithm in design of control system. The GA-based approach can be used for designing a control system without knowledge of modeling and automatic control. Furthermore, such an approach can be used for automatic and autonomous design of control systems that can offer robust performance regardless of parameter change. Future work will be focused on the evolutionary approach for fine-tuning the controller parameters that will be used in order to annul the influence of system decay.

Aknowledgment

This research has been (partly) supported by the CEEPUS network CIII-HR-0108, European Regional Development Fund under the grant KK.01.1.1.01.0009 (DATACROSS), project CEKOM under the grant KK.01.2.2.03.0004, CEI project "COVIDAi" (305.6019-20), project Metalska jezgra Čakovec (KK.01.1.1.02.0023) and University of Rijeka scientific grant uniri-tehnic-18-275-1447

7 REFERENCES

[1] Ardhenta, L. & Subroto, R. K. (2020). Feedback Control for Buck Converter-DC Motor Using Observer. *The 12th IEEE International Conference on Electrical Engineering (ICEENG)*, 30-33. https://doi.org/10.1109/ICEENG45378.2020.9171693

[2] Nouman, K., Asim, Z., & Qasim, K. (2018). Comprehensive Study on Performance of PID Controller and its Applications. *The 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, 1574-1579. https://doi.org/10.1109/IMCEC.2018.8469267

[3] Sun, X., Hu, C., Lei, G., Guo, Y., & Zhu, J. (2019). State feedback control for a PM hub motor based on gray wolf

optimization algorithm. *IEEE Transactions on Power Electronics, 35*(1), 1136-1146. https://doi.org/10.1109/TPEL.2019.2923726

[4] Abdulameer, A., Sulaiman, M., Aras, M. S. M., & Saleem, D. (2016). Tuning methods of PID controller for DC motor speed control. *Indonesian Journal of Electrical Engineering and Computer Science, 3*(2), 343-349. https://doi.org/10.11591/ijeecs.v3.i2.pp343-349

[5] Hernández-Alvarado, R., García-Valdovinos, L. G., Salgado-Jiménez, T., Gómez-Espinosa, A., & Fonseca-Navarro, F. (2016). Neural network-based self-tuning PID control for underwater vehicles. *Sensors, 16*(9), 1429. https://doi.org/10.3390/s16091429

[6] Paiva, E., Soto, J., Salinas, J., & Ipanaqué, W. (2016, October). Modeling, simulation and implementation of a modified PID controller for stabilizing a quadcopter. The IEEE International Conference on Automatica (ICA-ACCA), 1-6. https://doi.org/10.1109/ICA-ACCA.2016.7778507

[7] Leonhard, W. (2001). Control of electrical drives. Springer Science & Business Media. https://doi.org/10.1007/978-3-642-56649-3

[8] Qiao, D., Mu, N., Liao, X., Le, J., & Yang, F. (2020). Improved evolutionary algorithm and its application in PID controller optimization. *Information Sciences, 63*(199205), 1-199205. https://doi.org/10.1007/s11432-019-9924-7

[9] Baressi Šegota, S., Lorencin, I., Anđelić, N., Mrzljak, V., & Car, Z. (2020). Improvement of Marine Steam Turbine Conventional Exergy Analysis by Neural Network Application. *Journal of Marine Science and Engineering, 8*(11), 884. https://doi.org/10.3390/jmse8110884

[10] Baressi Šegota, S., Lorencin, I., Musulin, J., Štifanić, D., & Car, Z. (2020). Frigate Speed Estimation Using CODLAG Propulsion System Parameters and Multilayer Perceptron. *Naše more, 67*(2), 117-125. (in Croatian) https://doi.org/10.17818/NM/2020/2.4

[11] Lorencin, I., Anđelić, N., Španjol, J., & Car, Z. (2020). Using multi-layer perceptron with Laplacian edge detector for bladder cancer diagnosis. *Artificial Intelligence in Medicine*, 102, 101746. https://doi.org/10.1016/j.artmed.2019.101746

[12] Car, Z., Baressi Šegota, S., Anđelić, N., Lorencin, I., & Mrzljak, V. (2020). Modeling the Spread of COVID-19 Infection Using a Multilayer Perceptron. Computational and Mathematical Methods in Medicine, 2020. https://doi.org/10.1155/2020/5714714

[13] Lorencin, I. et al. (2021). Edge Detector-Based Hybrid Artificial Neural Network Models for Urinary Bladder Cancer Diagnosis. In: Hassanien, A. E., Taha, M. H. N., Khalifa, N. E. M. (eds) *Enabling AI Applications in Data Science. Studies in Computational Intelligence, vol. 911*. Springer, Cham. https://doi.org/10.1007/978-3-030-52067-0_10

[14] Lorencin, I., Anđelić, N., Mrzljak, V., & Car, Z. (2019). Marine objects recognition using convolutional neural networks. *Naše more, 66*(3), 112-119. (in Croatian) https://doi.org/10.17818/NM/2019/3.3

[15] Mirjalili S. (2019) Genetic Algorithm. In: *Evolutionary Algorithms and Neural Networks. Studies in Computational Intelligence, vol. 780*. Springer, Cham. https://doi.org/10.1007/978-3-319-93025-1_4

[16] Baressi Šegota, S., Anđelić, N., Lorencin, I., Saga, M., & Car, Z. (2020). Path planning optimization of six-degree-of-freedom robotic manipulators using evolutionary algorithms. *International Journal of Advanced Robotic Systems, 17*(2), 1729881420908076. https://doi.org/10.1177/1729881420908076

[17] Lorencin, I., Anđelić, N., Mrzljak, V., & Car, Z. (2019). Genetic algorithm approach to design of multi-layer perceptron for combined cycle power plant electrical power output estimation. *Energies, 12*(22), 4352. https://doi.org/10.3390/en12224352

[18] Baressi Šegota, S., Lorencin, I., Ohkura, K., & Car, Z. (2019). On the Traveling Salesman Problem in Nautical Environments: an Evolutionary Computing Approach to Optimization of Tourist Route Paths in Medulin, Croatia. *Pomorski zbornik, 57*(1), 71-87. https://doi.org/10.18048/2019.57.05

[19] Krishnan, R. (2017). *Permanent magnet synchronous and brushless DC motor drives*. CRC press. https://doi.org/10.1201/9781420014235

[20] Dubey, G. K. (2001). *Fundamentals of electrical drives*. Alpha Science Int'l Ltd.

[21] Akbar, M. A., Naniwa, T., & Taniai, Y. (2016, August). Model reference adaptive control for DC motor based on Simulink. *The 6th IEEE International Annual Engineering Seminar (InAES)*, 101-106.

[22] Síki, G. Á., Sarvajcz, K., Szántó, A., & Mankovits, T. (2020). Series Wound DC Motor Simulation Applying MATLAB SIMULINK and LabVIEW Control Design and Simulation Module. *Periodica Polytechnica Transportation Engineering, 48*(1), 65-69. https://doi.org/10.3311/PPtr.12908

[23] Mirjalili, S. (2019). Genetic algorithm. In *Evolutionary algorithms and neural networks*, Springer, Cham, 43-55. https://doi.org/10.1007/978-3-319-93025-1_4

[24] Kramer, O. (2017). *Genetic algorithm essentials (Vol. 679)*. Springer. https://doi.org/10.1007/978-3-319-52156-5

Authors' contacts:

Marko Mavrinac, mag. ing. el.
University of Rijeka, Faculty of Engineering
Vukovarska 58, 51000 Rijeka, Croatia
mavrinac07@gmail.com

Ivan Lorencin, mag. ing. el.
University of Rijeka, Faculty of Engineering,
Vukovarska 58, 51000 Rijeka, Croatia
ilorencin@riteh.hr

Zlatan Car, prof. dr. sc.
University of Rijeka, Faculty of Engineering,
Vukovarska 58, 51000 Rijeka, Croatia
car@riteh.hr

dr. sc. Mario Šercer, dipl. Ing.
Razvojno-edukacijski centar za metalsku industriju Metalska jezgra Čakovec,
Bana Josipa Jelačića 22 D,
40 000 Čakovec, Croatia
mario.sercer@mev.hr