STU
SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA

Journal of Electrical Engineering, Vol. 77, No. 4, 2026, pp. 366-375
https://reference-global.com/journal/JEE

# Control of nonlinear continuous-time multi-input and multi-output dynamic systems using neuroevolution

**Ivan Kenický[1], Ivan Sekaj[1], Filip Zúbek[1]**

We propose a neural network-based controller of nonlinear multi-input and multi-output (MIMO) continuous-time dynamic systems. The controller learns using a genetic algorithm (GA) to minimize the chosen criterion function. A new neural network (NN) architecture is proposed that uses difference- and summation-based neurons in the hidden layers. Two control approaches are compared: a decentralized approach, which assumes that individual subsystems are isolated feedback loops, and a centralized one, which assumes that the entire system is controlled as a single system including all internal interactions between subsystems. Both approaches are compared experimentally with a conventional decentralized PID controller approach. We will show that both NN approaches can learn the plant non-linearity, however, the centralized control approach can much better eliminate the mutual internal interactions among the partial subsystems of the MIMO system.

Keywords: neuroevolution, artificial neural network, genetic algorithm, nonlinear dynamic system, MIMO system, new controller architecture, centralized approach, decentralized approach

## 1 Introduction

Artificial neural networks (ANNs) are currently an important approach to problem-solving, including in process control. They perform various functions, ranging from modelling nonlinear dependencies or dynamic systems, through predicting dynamic systems and events, to directly controlling systems. This work focuses on the control (regulation) of nonlinear dynamic systems, in which the neural network serves as a direct controller.

In general, control systems typically have information about the controlled object they are supposed to control. The control objective is usually formulated using a user-selected criterion that evaluates the degree of success of the objective based on the stability and quality of the obtained solution. However, we usually lack information on how to achieve the optimal control objective. In other words, we do not have the complete input-output data required for supervised learning with neural networks, which is the most widespread approach in machine learning (error backpropagation).

Today, the machine learning approach known as reinforcement learning (RL) [1, 2] is most used for decision-making and control without prior knowledge of the output data. An alternative to neural network learning is neuroevolution (NE). Evolutionary algorithms (EA) are used as learning algorithms in this context [3-5]. CMA-ES is currently a very effective approach [6]. But also the genetic algorithm is frequently used in this regard [7,8]. Many authors have addressed neuroevolution over the past three decades, including [9-11].

The fundamental difference between RL and NE is that RL adjusts a single network weights based on gradients and immediate rewards. In contrast, EA operates on an entire population of neural networks and selects the most successful ones. An RL agent typically receives a reward after each action. An evolutionary algorithm evaluates only the overall fitness of the result at the end of the simulation or calculation - that is, after all actions have been performed. If a network performs well, it survives and passes on its genes.

RL is based on gradient calculations. If the objective function is not smooth, is discrete, or contains too much noise, gradients may fail. In contrast, EAs do not require gradients; they use mutation and crossover. As a result, they can optimize even networks with discontinuous activation functions or in noisy environments where gradient methods would get stuck in a local optimum. If an RL agent finds a strategy that works even a little, it begins to favor it at the expense of searching for something fundamentally new. Since EAs maintain a diversified population of candidates and use random mutations, they can explore the vast state space more effectively. However, maintaining a population is computationally expensive because we must repeatedly compute the fitness of the entire population.

[1]Institute of Robotics and Cybernetics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Ilkovicova 3, 841 02 Bratislava, Slovakia
E-mail: ivan.kenicky@stuba.sk, ivan.sekaj@stuba.sk, filip.zubek@stuba.sk

https://doi.org/10.2478/jee-2026-0035, Print (till 2015) ISSN 1335-3632, On-line ISSN 1339-309X

RL has been successfully applied many times to solve process control problems [12-15]. Neural networks trained using RL or NE can also be used in process control in an indirect role, where they perform auxiliary functions such as approximating nonlinear functions, modelling processes, predicting dynamics, and the like. Various architectures with recurrent or LSTM networks are used for this purpose [16-20, 27].

As we have already mentioned, one option is to use a neural network as a control element to generate the control variable directly. In our article, we focus specifically on this second option using NE. It should be noted that NE can design both the architecture and the parameters of a neural network. We will focus solely on finding the network parameters, assuming that the architecture is fixed and all adjacent layers of neurons are fully connected. Under these conditions, ANN computations are performed simply by matrix multiplication.

Many authors have proposed controlling dynamic systems using neural networks. This involves the control of nonlinear, unstable, and complex systems for which analytically based control algorithms often cannot be satisfactorily applied. Some of these works are cited in [21-26].

In our article, we propose a new neural network architecture for controlling nonlinear dynamic systems. It is based on a feedforward MLP-type artificial neural network with nested layers containing "dynamic neurons" (summation and differentiation). These can influence the dynamic properties of the neural network without requiring preprocessing of input signals (e.g., calculating derivatives and integrals) or additional recursions, all using a minimal set of input signals.

The second part of the article describes the architecture of the proposed neural network. The third part explains the neural network training algorithm using a parallel genetic algorithm. The fourth section demonstrates the main objective of the article, which is the design and comparison of two approaches to MIMO system control, decentralized and centralized, and compares the control results achieved with both approaches with conventional decentralized PID controller.

## 2 Neuro-controller architecture

Let us consider a simple closed-loop feedback system (Fig. 1). The controller is based on an ANN (neuro-controller, NC). The goal is to propose ANN parameters that minimize the selected control performance index. In our case, an integral-type performance index is considered: the integral of the absolute control error (IAE), given by (1), where T is the time interval of the control process under consideration [28, 29].

$$IAE = \int_0^T |e(t)|dt \tag{1}$$

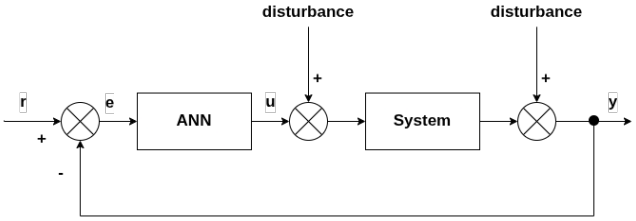


**Fig. 1.** Simple feedback control closed loop

Considering an MLP-type (multilayer perceptron) feedforward neural network (FF) as in Fig. 2, it requires a vector of preprocessed input variables X, to achieve the control goal. Let us consider the following vector of potential input variables

$$X = [e,\ \sum e,\ \delta e,\ \delta^2 e,\ \delta^3 e,\ y,\ \delta y,\ \delta^2 y,\ \delta^3 y,\ \delta u,\ \delta^2 u,\ \ldots] \tag{2}$$

where $\delta x(k) = x(k) - x(k-1)$ and $\delta^n x(k) = \delta^{n-1} x(k) - \delta^{n-1} x(k-1)$. The next step in pre-processing input signals is to normalize vector X to the considered internal working space of the ANN parameters, $S \in \langle -1; 1 \rangle$. Let us use the vector $N = [N_1, N_2, \ldots, N_n]$, where $N_i$ is the normalization of the $i$-th input variable (see Fig. 2) according to the expression

$$x_i' = N_i x_i; \quad N_i = \frac{1}{\max(|x_i|)} \tag{3}$$

Such normalization is only important when the input signal's operating range is smaller than 1. In the opposite case, normalization can be included in the proper setting (optimization) of the weights $w_{1,i,j}$, which are part of the neural network's learning process. Similarly, at the output of the neural network, denormalization (Fig. 2) is applied according to

$$u = U_{max}.u'; U_{max} = \max(|u|) \tag{4}$$

where $U_{max}$ is the maximal required value of the controller's output and $u'$ is the output of the last ANN layer.

Based on the FF network, let us propose a new neural network architecture as in Fig. 3. It contains additional inserted summation-difference (SD) layers of neurons.

Each SD layer is placed after each fully-connected layer. Such SD layers consist of three types of neurons: a) simple shift neuron – only shifts the information without changing it, b) summation neuron – sums the input signal, c) difference neuron – calculates the difference of the input signal. The scheme of S and D neurons is shown in Fig. 3 below. The used ratio of neuron numbers in the SD layer is $a$:$b$:$c$ = 2:1:1. The input vector $X$ of the NC contains here only two variables $X = [y, r]$ in the simplest case (for a SISO system). But other arbitrary variables (information) can be considered as input variables of NC as well. NC with S and D neurons can generate summation and higher-order differences of the network's internal signals, thereby driving the NC's dynamic behavior. The preprocessing step for the input vector variables (2) can be omitted here.

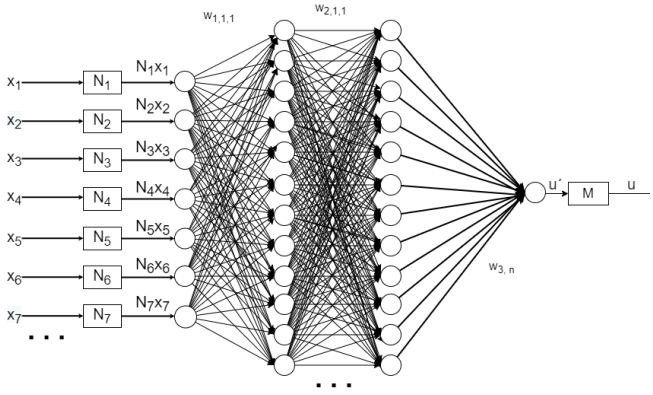


**Fig. 2.** Feedforward (FF) neurocontroller of the type MLP (Multilayer Perceptron Network)

Next, consider two types of neuron activation functions. The first type is in the form

$$f(x) = \tanh(3x) \tag{5}$$

In the basic case, we consider that each neuron's inputs and outputs are operating in the normalized interval $S \in \langle -1; 1 \rangle$. Alternatively, let us consider the linear activation function in the form

$$f(x) = x \tag{6}$$

the use of which will allow us to extend the operating space of each neuron to $S \in \langle -U_{max}; U_{max} \rangle$. This will also eliminate the need for normalization (N) and denormalization (M) (Fig. 3).

When controlling systems with multiple inputs and outputs (MIMO systems), it is necessary to adapt the neurocontroller's input and outputs. This can be achieved in several ways. In a decentralized architecture, we aim to divide a complex system into isolated subsystems (Fig. 4, $S_1$-$S_3$) and control them using decentralized controllers ($NC_1$-$NC_3$). We can treat the existing interactions between subsystems as external disturbances. Each isolated controller can have the architecture shown in Fig. 3. In contrast, in a centralized approach, we consider a single controller whose inputs and outputs meet all the requirements of the MIMO system (Fig. 5).

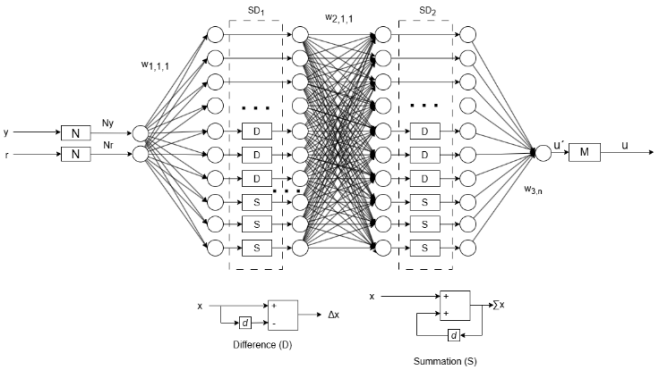


**Fig. 3.** Feedforward (FF) with inserted SD layers, d-single step time-delay

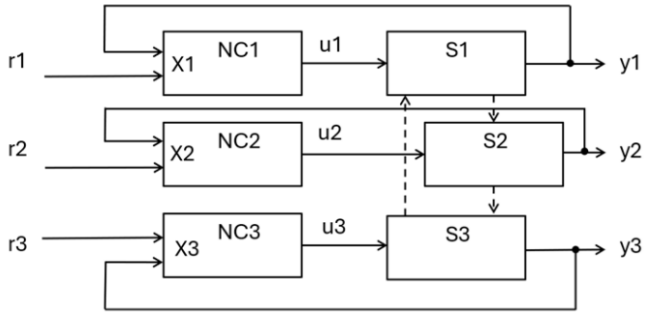


**Fig. 4.** Decentralized control structure of MIMO system

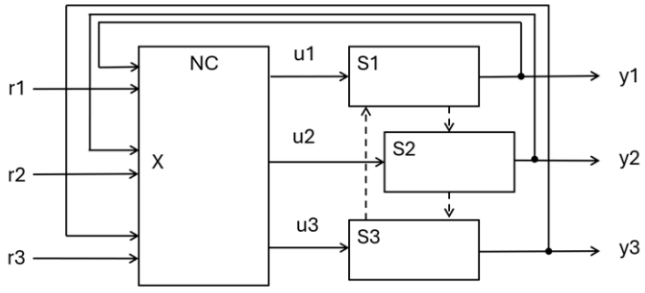


**Fig. 5.** Centralized control structure of MIMO system

## 3 Neuro-evolution of the controller

As mentioned, the goal of neuroevolution is to find parameters for a neurocontroller that minimize the criterion function (fitness function), for example, as in

(1). In our case, a genetic algorithm (GA) was used for this optimization task. The GA works with a population of possible solutions - chromosomes. Let us consider direct encoding of ANN parameters into a chromosome, where each chromosome represents a vector of all ANN parameters – real numbers, i.e., a vector containing all ANN weight elements $w_{i,j,k}$ and biases $b_{i,k}$. Now, let us assume that the neural network's structure is fixed, fully connected, and not subject to optimization. For a network with two hidden layers, as shown in Fig. 2 or Fig. 3, we can consider the format of a single chromosome representing the population as

$$chromosome = [W_1, W_2, W_3, B_1, B_2] = \\ = [w_{1,1,1}, w_{1,1,2}, \dots, w_{3,n}, b_{1,1}, b_{1,2}, \dots, b_{2,n}] \quad (7)$$

where $W_i$ are weight matrices, $B_i$ are the bias vectors, which are transformed into a one-dimensional vector of real numbers from the space $S$, and $n$ is the number of neurons in the hidden or output layer. Let us now assume all biases are zero. We have experimentally verified that this step does not reduce either the convergence rate or the control quality. The chromosome is then simplified to the form

$$chromosome = \\ = [W_1, W_2, W_3] = [w_{1,1,1}, w_{1,1,2}, \dots, w_{3,n}] \quad (8)$$

In the case of three decentralized controllers (Fig. 4), the chromosome will be shaped like

$$chromosome = \begin{bmatrix} W_{11}, W_{12}, W_{13}, \\ W_{21}, W_{22}, W_{23}, W_{31}, W_{32}, W_{33} \end{bmatrix} \quad (9)$$

The goal of NE is to find the values of the chromosome elements that minimize (1). For these purposes, a parallel genetic algorithm (PGA) is proposed, as shown in Fig. 6. Block A represents 20 islands (isolated GA populations) that exchange clones of their best individuals with one another. Every 3 generations of the PGA calculation, one individual migrates between randomly selected islands. The clone of the best individual from the source island replaces a randomly selected individual on the goal island, except for the best individual. Every 20 generations, the best individual migrates from island A to island B and from B to C. B and C are single-population islands. After the specified number of generations has been completed, the solution to the problem is the best individual from island C. The same genetic algorithm is used on each of the islands A, B, and C (according to Algorithm 1). Note that any other PGA/GA or even another EA type can be used for chromosome optimization.

***Algorithm 1***

1. Set the initial population P of $n$ individuals to zero.
2. Evaluate the fitness of each chromosome.
3. Select from P a group of the most successful (elitist) chromosomes E (the size of E is maximum $\frac{1}{4}$P). Select 2 equal groups of parents, $P_1$, $P_2$, using tournament selection.
4. Modify the group $P_1$ using local mutation to $P_1'$. Modify the group $P_2$ using global mutation to $P_2'$.
5. Unite all groups into a new population of size n individuals: $P = E \cup P_1' \cup P_2'$.
6. Continue in step 2 until the termination conditions are met.

A local mutation changes the gene's original value by up to ±10% of the entire search space $S$. A global mutation changes the gene's original value to a random value from the entire space $S$.

*Remark 1*: Note that global/local mutations are related to the terms exploration/exploitation commonly used in literature.

*Remark 2:* We do not use the crossover operation in GA.

The evaluation of the fitness function of each chromosome in step 2 of the GA algorithm consists of three sub-steps:

2.1. Genotype to phenotype transformation, i.e., transformation of chromosome elements from format (8) into weight matrices $W_1$, $W_2$, $W_3$

2.2. Simulation of the control process, where matrix multiplications are used for the calculation of the ANN output:

a) for activation function (5): $a_1=W_1X$; $y_1=\tanh(3a_1)$; $a_2=W_2y_1$; $y_2=\tanh(3a_2)$; $u=MW_3y_2$; $w_{i,j,k} \in \langle -S, S \rangle$; $S=1$, where $w_{i,j,k}$ are the elements of the weight matrices $W_1, W_2, W_3, M = U_{max}$.

b) for activation function (6): $y_1 = W_1X$; $y_2 = W_2y_1$; $u = W_3y_2$; $w_{i,j,k} \in \langle -S, S \rangle$; $S = U_{\max}$, where $w_{i,j,k}$ are the elements of the weight matrices $W_1, W_2, W_3$.

2.3. Evaluation of the performance index (1) or another.

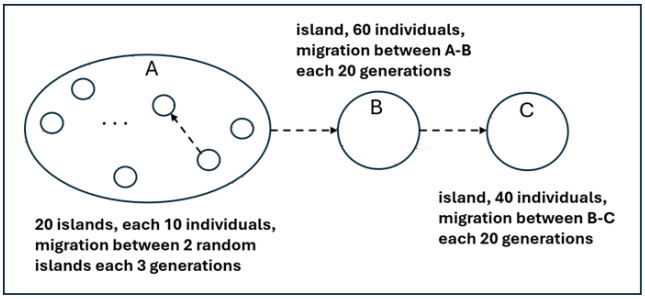


**Fig. 6.** Used parallel genetic algorithm

## 4 Experimental verification and comparison of neuro-controllers

For practical demonstration, let us consider the nonlinear MIMO dynamic system, which is considered in the form of discrete-time difference equations

$$a_{14}\Delta^4 y_1 + a_{13}\left(1+\frac{{y_1}^2}{3}\right)\Delta^3 y_1 + \\ +a_{12}\Delta^2 y_1 + a_{11}\Delta y_1 + a_{10} y_1 \\ -b_{11}\Delta u_1 - b_{10}u_1 + \rho + q_{31} = 0 \qquad (10)$$

$$a_{24}\Delta^4 y_2 + a_{23}\left(1+\frac{{y_2}^2}{3}\right)\Delta^3 y_2 \\ +a_{22}\Delta^2 y_2 + a_{21}\Delta y_2 + a_{20} y_2 \\ -b_{21}\Delta u_2 - b_{20}u_2 + \rho - q_{12} = 0 \qquad (11)$$

$$a_{34}\Delta^4 y_3 + a_{33}\left(1+\frac{{y_3}^2}{3}\right)\Delta^3 y_3 \\ +a_{32}\Delta^2 y_3 + a_{31}\Delta y_3 + a_{30} y_3 \\ -b_{31}\Delta u_3 - b_{30}u_3 + \rho + q_{23} = 0 \qquad (12)$$

where: $a_{10}=1$, $a_{11}=1.3$, $a_{12}=8$, $a_{13}=1$, $a_{14}=1$, $b_{11}=0.1$, $b_{10}=1$, $a_{20}=1$, $a_{21}=1.3$, $a_{22}=8$, $a_{23}=1$, $a_{24}=1$, $b_{21}=0.1$, $b_{20}=0.7$, $a_{30}=1$, $a_{31}=5$, $a_{32}=8$, $a_{33}=1$, $a_{34}=1$, $b_{31}=0.1$, $b_{30}=0.5$,

and $\rho$ is a uniformly distributed pseudorandom variable with amplitude 0.001 and zero mean value (noise). The interactions between the three subsystems are $q_{12} = y_1 K_q T_s$, $q_{23} = y_2 K_q T_s$, and $q_{31} = y_3 K_q T_s$, with $K_q = 0$ in the absence of interactions between subsystems, or $K_q = 0.4$ with interactions. $T_s = 0.05$s is the controller sampling period. The difference operator $\Delta$ is defined in the usual way, $\Delta x(k) = x(k+1) - x(k)$ and $\Delta^n x(k) = \Delta^{n-1} x(k+1) - \Delta^{n-1} x(k)$. The system is characterized by nonlinear dynamics. With an increasing value of the input signal $u$ (from zero to $u_{max}$), the time response of the system output $y$ changes from an aperiodic to an oscillatory, and then to an unstable one. Let the maximum value of the control variable be limited to $u_{max} = 30$.

In PGA (Fig. 6), the total number of individuals in the PGA is 300. In structure A, there are 20 islands, each with 10 individuals, B is an island with 60 individuals and C is an island with 40 individuals. The mutation rate (both local and global) was $m_r$=0.01. The neural network (Fig. 3) consisted of three fully connected hidden layers and three embedded SD layers, each with 36 neurons and a linear activation function (6). The simulation experiment lasted 220 seconds for both training and testing. The number of generations used to learn the neurocontrollers has been set to 2000. All experiments have been performed in the Matlab environment [30].

In Fig. 7 the time responses of the outputs of all three subsystems $y_1$, $y_2$, $y_3$, as the control loop's response to step changes in the reference variables $r_1$, $r_2$, $r_3$ during test scenario are shown. Here the MIMO system without internal interactions is considered ($K_q$=0, $K_q$ is the gain of internal interactions) and the decentralized control approach with PID controllers in each of the three loops is used. Note that the behavior of both neurocontrollers (decentralized and centralized) is similar.

Figures 8-11 show the time responses of the control of the MIMO system with interactions ($K_q$=0.4). Again, the outputs of all three subsystems $y_1$, $y_2$, $y_3$, as the control loop's response to step changes in the reference variables $r_1$, $r_2$, $r_3$ during training and test scenarios, are presented. It is evident that centralized control (C) is better able to follow the reference signal and suppress deviations in the controlled variables during subsystem interactions. The deviations of the controlled variables as response to step changes in the reference variables of neighboring subsystems are indicated by arrows. These deviations are significantly better damped under centralized control (C) than under decentralized control (D). Centralized NC is better able to learn and adapt to the nonlinear dynamics of the controlled subsystems and respond to disturbances than decentralized NC, because it has complete information about all subsystems and their interactions. In Fig. 12, the evolution of the fitness function is shown for the decentralized (blue line), centralized (red line) and PID (green line) cases for systems without interactions ($K_q$=0, dashed line) and with strong interactions ($K_q$=0.4 in Fig. 8, solid line). In Fig. 13, the time responses of the three decentralized closed loops under PID controllers are shown for $K_q$ = 0.3. The control performance is much worse than that of the NC. For $K_q$=0.4, the system becomes unstable. The statistical evaluation of the best 15 of 30 runs for all controller designs is in Tab. 1 and Tab. 2. In Figs. 14 and 15, the box plots of the same statistics are presented. The "best 15 of 30 runs" selection was introduced because several learning runs got stuck during training in a weak local extreme that did not meet the required performance metrics. Such outliers then distort the results. In practice,

a situation where NC training gets stuck in a weak local extreme is not a problem. The training is repeated, and a suitable new solution replaces the weak solution.

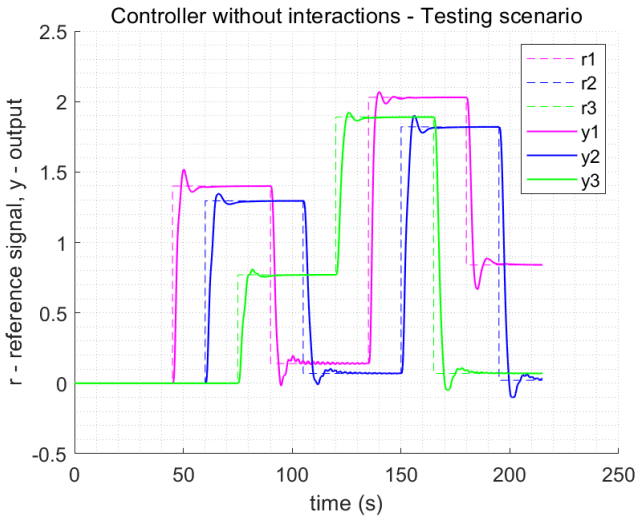


**Fig. 7.** Time-responses of system outputs of the 3×3 MIMO system without internal interactions between subsystems, decentralized control approach, test scenario, PID controllers

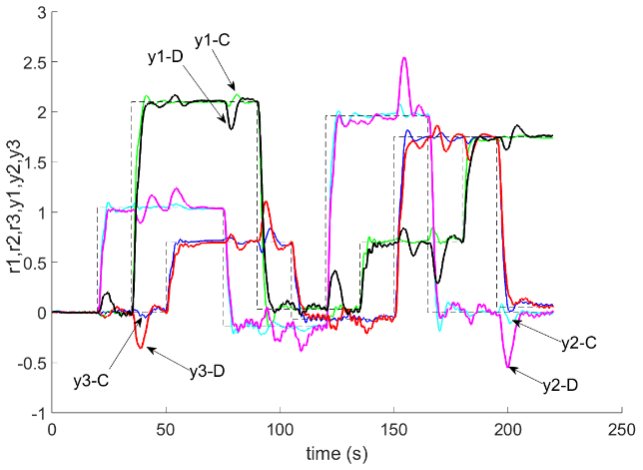


**Fig. 8.** Time-responses of system outputs of the 3×3 MIMO system, comparison of the centralized control approach (C, green, cyan, blue) and decentralized (D, black, magenta, red) approach, training scenario

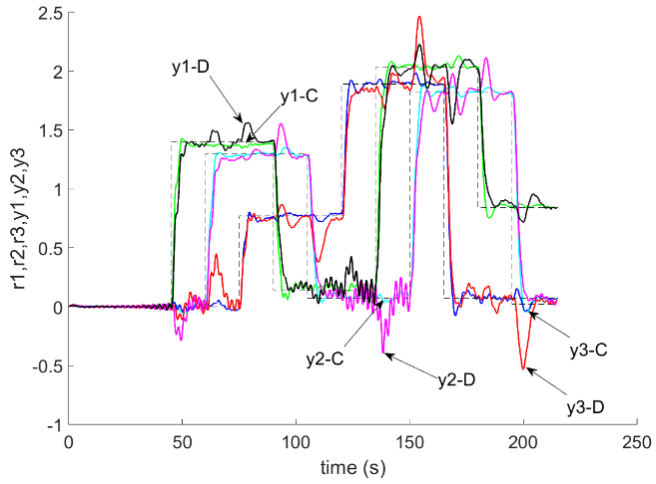


**Fig. 9.** Time-responses of system outputs of the 3×3 MIMO system, comparison of the centralized control approach (C, green, cyan, blue) and decentralized (D, black, magenta, red) approach, test scenario

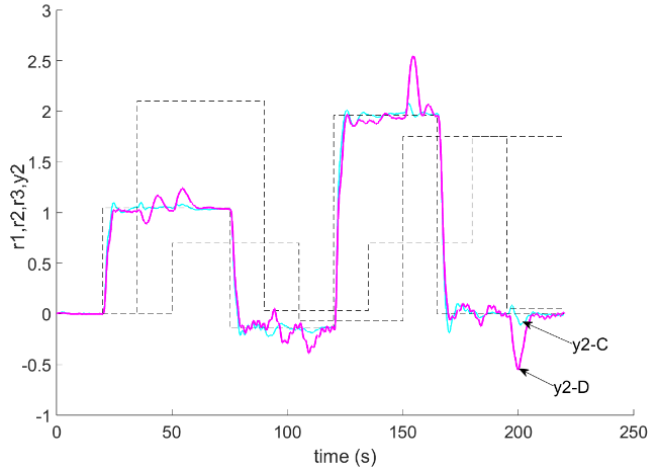


**Fig. 10.** Time-responses of system output 2 of the 3×3 MIMO system, comparison of the centralized control approach (C, cyan) and decentralized (D, magenta) approach, training scenario

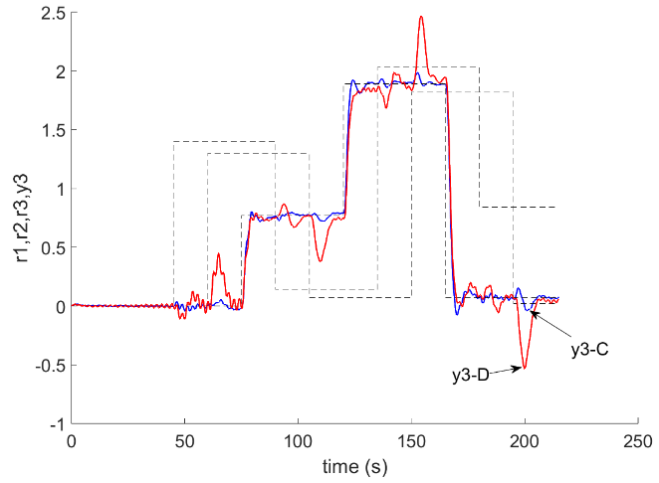


**Fig. 11.** Time-responses of system output 3 of the 3×3 MIMO system, comparison of the centralized control approach (C, blue) and decentralized (D, red) approach, test scenario

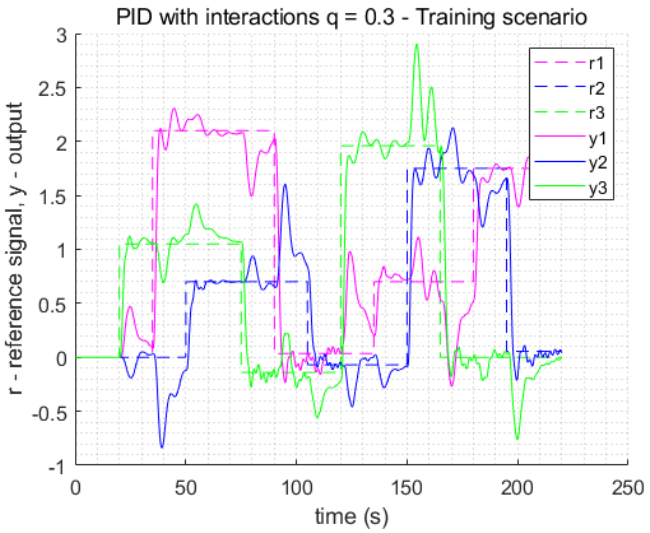


**Fig. 13.** Time-responses of system outputs of the 3×3 MIMO system, 3 decentralized PID controllers, training scenario

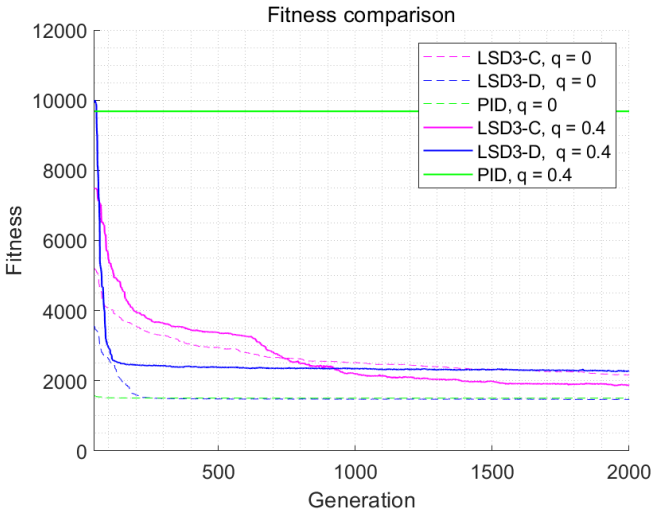


**Fig. 12.** Evolution of the fitness functions during 3×3 MIMO system training of the centralized (blue)/ decentralized (red) neurocontrollers, subsystems without interactions (dashed line) and with interactions (solid line)

**Remark about closed-loop stability:** The closed-loop stability is not explicitly tested during the design process. However, the integral control performance indices are included in the computation of the fitness function, which constantly pushes the system towards stable solutions, the stability issue is implicitly addressed in the proposed solution.

**Table 1.** Statistical comparison of neuro-controllers, best 15 of 30 runs of each controller, without interactions of subsystems

| Con-troller type | IAE train | STD | IAE test | STD | Best IAE test | Unstable | Control loop evals | Comp. time | Inputs/ output used |
|---|---|---|---|---|---|---|---|---|---|
| LSD3D | 6.6978e+03 | 5.7727 | 7.3121e+03 | 5.2021 | 7.0618e+03 | 0 | 222000 | 1797 s | 3×2 |
| LSD3C | **5.1326e+03** | 4.0021 | **6.1624e+03** | 3.6258 | 5.8573e+03 | 0 | 222000 | 1687 s | 6 |
| PID | 1.6553e+04 | 12.8956 | 1.7957e+04 | 11.7543 | **1.6923e+03** | 0 | **90300** | **474 s** | 3×1 |

**Table 2.** Statistical comparison of neuro-controllers, best 15 of 30 runs of each controller, with interactions of subsystems

| Con-troller type | IAE train | STD | IAE test | STD | Best IAE test | Unstab-le | Control loop evals | Comp. time | Inputs/ output used |
|---|---|---|---|---|---|---|---|---|---|
| LSD3D | 1.1800e+04 | 6.7286 | 1.2144e+04 | 6.2648 | 1.2014e+04 | 0 | 222000 | 1797 s | 3×2 |
| LSD3C | **9.0867e+03** | 6.1625 | **9.7075e+03** | 5.5659 | **9.5437e+03** | 0 | 222000 | 1687s | 6 |
| PID | 9.6829e+04 | 0 | 9.6829e+04 | 0 | 9.6829e+04 | 30 | 90300 | 474 s | 3×1 |

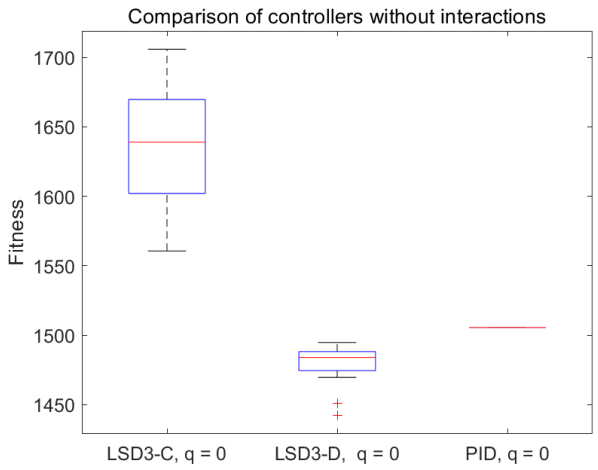


**Fig. 14.** Box plot of best 15 of 30 runs, test scenario, without interactions (better fitness is the smaller)

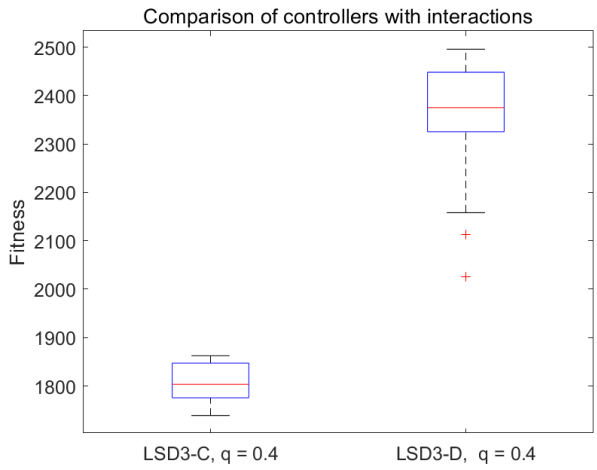


**Fig. 15.** Box plot of best 15 of 30 runs, test scenario, with interactions $q$=0.4 (better fitness is the smaller)

## 5 Conclusion

In our work, we have proposed a new neural network architecture suitable for controlling nonlinear dynamic systems with a nontrivial structure, such as MIMO systems with interactions between individual subsystems. The basis is an MLP neural network with additional layers of neurons that contain specific recursions that perform signal summation and differentiation. Control using the proposed neural networks demonstrated their ability to adapt to the nonlinear dynamics of the controlled object. We then demonstrated and compared two different approaches to applying the proposed architecture. The first is decentralized, where each subsystem is treated as isolated and controlled by its own neurocontroller. The second approach is centralized, involving a single common neurocontroller with inputs and outputs connected to all subsystems of the controlled object. This second approach proved to be more effective. It achieved better tracking of reference signals and better suppression of internal interactions between particular subsystems of the MIMO system. The proposed approach provides reliable results for controlling complex nonlinear dynamic systems. The only condition is the existence of an appropriate model of the controlled system.

## Acknowledgement

This publication was created thanks to the support of the Slovak Research and Development Agency (APVV) within the grant No. APVV-22-0169.

## References

[1] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA: MIT Press, 2018.

[2] V. Mnih et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, pp. 529–533, 2015.

[3] A. E. Eiben and J. E. Smith, Introduction to Evolutionary Computing, 1st ed. Berlin, Germany: Springer-Verlag, 2003.

[4] T. Bäck, Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, 1st ed. New York, NY, USA: Oxford University Press, 1996.

[5] T. Bäck, D. B. Fogel and Z. Michalewicz, eds., Handbook of Evolutionary Computation, 1st ed. Bristol, UK: Institute of Physics Publishing; New York, NY, USA: Oxford University Press, 1997.

[6] N. Hansen, S. D. Müller and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," Evolutionary Computation, vol. 11, no. 1, pp. 1–18, 2003.

[7] J. H. Holland, Adaptation in Natural and Artificial Systems, 2nd ed. Cambridge, MA: MIT Press, 1992.

[8] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, 3rd ed. Berlin, Germany: Springer-Verlag, 1996.

[9] D. Floreano, P. Dürr and C. Mattiussi, "Neuroevolution: From architectures to learning," Evolutionary Intelligence, vol. 1, no. 1, pp. 47–62, 2008.

[10] R. M. Khan, "A Review Study on Neuro Evolution," International Journal of Scientific Research and Engineering Trends, vol. 7, pp. 133–137, 2021.

[11] R. Miikkulainen, "Neuroevolution," in Encyclopedia of Machine Learning, C. Sammut and G. I. Webb, Eds. Boston, MA, USA: Springer, 2010.

[12] S. Bandyopadhyay and S. Bhasin, "Safe reinforcement learning control for continuous-time nonlinear systems without a backup controller," arXiv preprint arXiv:2209.08922, 2022.

[13] Y. Huang, Y. Jia and X. Y. Zhou, "Sublinear regret for an actor-critic algorithm in continuous-time linear-quadratic reinforcement learning," arXiv preprint arXiv:2407.17226, 2024.

[14] H. Wu, J. Zheng, Y.-H. Liu and D. Li, "Output-feedback control of linear continuous-time systems using discounted inverse reinforcement learning," IEEE Transactions on Cybernetics, early access, 2026.

[15] Q. Meng and Y. Peng, "Output-feedback quadratic tracking control of continuous-time systems by using off-policy reinforcement learning with neural networks observer," in Proc. IEEE Chinese Control and Decision Conf. (CCDC), Hefei, China, 2020, pp. 1–6.

[16] R. G. Hart, E. Griffis, O. S. Patil and W. E. Dixon, "Lyapunov-based physics-informed LSTM neural network-based adaptive control," IEEE Control Systems Letters, vol. 8, pp. 13–18, 2024.

[17] "Lyapunov-based long short-term memory (Lb-LSTM) neural network-based control," IEEE Control Systems Letters, 2023, Art. no. 3291328.

[18] "LSTM-MPC: A deep learning based predictive control method for multimode process control," IEEE Transactions on Industrial Electronics, early access, 2022.

[19] "Fast nonlinear model predictive control using LSTM networks: A model linearisation approach," in Proc. European Control Conference (ECC), 2022.

[20] "Robust offset-free constrained model predictive control with long short-term memory networks—Extended version," arXiv preprint arXiv:2303.17304, 2023.

[21] F. J. Gomez and R. Miikkulainen, "Solving non-Markovian control tasks with neuroevolution," in Proc. Int. Joint Conf. Artif. Intell. (IJCAI), San Francisco, CA, USA, 1999, pp. 1356–1361.

[22] P. Pagliuca, N. Milano and S. Nolfi, "Efficacy of modern neuro-evolutionary strategies for continuous control optimization," arXiv preprint arXiv:1912.05239, 2019.

[23] F. J. Gomez, J. Schmidhuber and R. Miikkulainen, "Efficient nonlinear control through neuroevolution," in Machine Learning: ECML 2006, Lecture Notes in Computer Science, vol. 4212. Berlin, Germany: Springer, 2006, pp. 64–74.

[24] S. C. Duong, E. Uezato, H. Kinjo and T. Yamamoto, "Intelligent control strategies for the Acrobot using neurocontroller optimized by genetic algorithm," SICE Journal of Control, Measurement, and System Integration, vol. 2, no. 5, pp. 317–324, 2009.

[25] Y. Park, M. Choi and K. Y. Lee, "An optimal tracking neuro-controller for nonlinear dynamic systems," IEEE Transactions on Neural Networks, vol. 7, no. 5, pp. 1099–1110, 1996.

[26] I. Sekaj, I. Kenický, F. Zúbek and J. Skirkanič, "Design of neuro-controllers for nonlinear continuous-time systems using evolutionary algorithm," in Proc. 10th Int. Conf. Control, Decision and Information Technologies (CoDIT), Valletta, Malta, 2024, pp. 1165–1170.

[27] M. Le Clec'h and P. Bellec, "Neuroevolution of recurrent architectures on control tasks," arXiv preprint arXiv:2304.12431, 2023.

[28] R. C. Dorf, Modern Control Systems. Reading, MA, USA: Addison-Wesley, 1990.

[29] B. C. Kuo, Automatic Control Systems, 6th ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 1991.

[30] The MathWorks, Inc., MATLAB, version R2022a. Natick, MA, USA: The MathWorks, 2022.

Received 10 June 2026