ISSN: 1803-3814 (Printed), 2571-3701 (Online)
https://doi.org/10.13164/mendel.2021.2.007



# Neuro-Evolution of Continuous-Time Dynamic Process Controllers

**Ivan Sekaj✉, Ivan Kenický, Filip Zúbek**

Institute of Robotics and Cybernetics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Bratislava, Slovak Republic

ivan.sekaj@stuba.sk✉, ivan.kenicky@stuba.sk, filip.zubek@stuba.sk

**Abstract**

*Artificial neural networks are means which are, among several other approaches, effectively usable for modelling and control of non-linear dynamic systems. In case of modelling systems input and output signals are a-priori known, supervised learning methods can be used. But in case of controller design of dynamic systems the required (optimal) controller output is a-priori unknown, supervised learning cannot be used. In such case we only can define some criterion function, which represents the required control performance of the closed-loop system. We present a neuro-evolution design for control of a continuous-time controller of non-linear dynamic systems. The controller is represented by an MLP-type artificial neural network. The learning algorithm of the neural network is based on an evolutionary approach with genetic algorithm. An integral-type performance index representing control quality, which is based on closed-loop simulation, is minimised. The results are demonstrated on selected experiments with controller reference value changes as well as with noisy system outputs.*

**Keywords:** *Continuous-Time Controller, Non-linear Dynamic System, Artificial Neural Network, Genetic Algorithm-Based Learning, Control Performance.*

**Received:** 10 December 2021
**Accepted:** 20 December 2021
**Published:** 21 December 2021

## 1 Introduction

Artificial neural networks (ANN) are used in many practical application domains as means for classification, approximation and modelling. Beside these traditional implementations, they can be used also as controllers in the control engineering domain. Under controllers we understand generators of signals for controlling continuous-time dynamic systems, dynamic event systems, logical systems, etc., which are based on feedback information, with the aim to reach desired behaviour of the considered system. The advantage of using ANN in comparison to several other controller design approaches is the ability to deal with non-linear systems, systems with complex internal structure and complex behaviour. The obvious way in utilising ANN in most of the applications is the use of known input/output data which were obtained during observation of the system behaviour during its normal operation in history (supervised learning, reconstruction of the reality, imitation of other existing control systems). But obtaining input/output data of a future control process is usually not possible. The (sub)optimal behaviour of the designed controller is a-priori unknown. Known is only the desired closed-loop behaviour, or the desired behaviour of the controlled object respectively, which can be described by some cost function or the desired trajectory.

The aim of this project is to design an ANN-based controller of continuous-time dynamic processes. There exists a wide spectrum of approaches for designing continuous-time controllers and most of them do not use ANNs. Our goal is to introduce an evolutionary-based unsupervised way of designing a direct ANN controller using neuro-evolution. The controller is represented by an MLP-type ANN and the learning approach uses a Genetic Algorithm-based optimisation of the desired closed-loop performance index.

There are several authors who deal with controlling systems using ANNs. Articles [1, 6, 11] mention among several others the use of ANN for replacement of the very popular PID controllers. The neuro-evolution learning approach in general was reported by many authors for optimizing of the ANN architecture and/or the synaptic weights [12, 4, 3, 10, 16, 17]. Neuro-evolution as a learning approach has been used for solving various practical control implementations in mobile robotics and autonomous driving [5, 15].

In Section 2 of this article, we explain the architecture of the proposed ANN controller and its inputs and outputs. In Section 3 the learning approach based on genetic algorithm is described. Section 4 shows experimental results of the evolution-based design in closed-loops with controller reference value changes as well as with noisy system outputs.

## 2 The ANN-Based Controller Architecture

The most common type of controller used in practice to control dynamic systems is the PID controller. Its

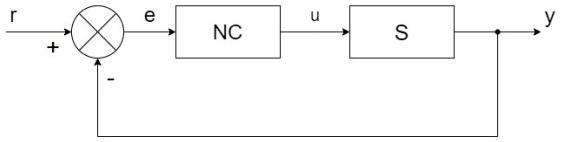


Figure 1: Block scheme of a simple feed-back control closed-loop with neuro-controller (NC), the controlled system (S) where $r$ is the reference value, $e$ is control error, $u$ is the controller output and $y$ is the controlled value.

output (control variable) is calculated according to the equation

$$u(t) = Pe(t) + I\int e(t)dt + D\frac{de(t)}{dt} \tag{1}$$

where $u$ is the control value, $e$ is the control error, $P$ is the proportional gain, $I$ is integral gain and $D$ is derivative gain. The equation represented in the discrete-time domain is

$$\begin{aligned} u(k) &= Pe(k) + I\sum e(k) + D\Delta e(k) \\ \sum e(k) &= \sum e(k-1) + e(k) \\ \Delta e(k) &= e(k) - e(k-1) \end{aligned} \tag{2}$$

where $k$ is the discrete control step.

Our aim is to replace the PID controller or any other type of controller by a neural network controller. Without loss of generality, let us consider a simple feed-back control loop with a controlled system S and an ANN-based NC (neuro-controller) (Fig. 1). Let the neuro-controller be an MLP-type neural network with two hidden layers (Fig. 2). As inputs into the neuro-controller we proposed 7 input signals which are the controlled system output $y$, its first difference $\Delta y$, second difference $\Delta^2 y$ and third difference $\Delta^3 y$, control error $e$, sum of control error $se$ and first difference of control value $\Delta u$. All inputs are normalized to the input range of neurons using multiplication constants $N_i$.

$$N_i = \frac{3}{|X_{i,\max}|} \tag{3}$$

where $X_{i,\max}$ is the maximal absolute value of the $i$-th input signal to the network. This is because of the input range of the activation function of all used neurons, which is the hyperbolic tangent function (see Fig. 3 and Fig. 4). Similarly, the output signal from the controller $u'$ has to be de-normalized from the output range of the output layer neuron $\rho_o \in (-1;1)$ to the required controller output $u$

$$u = Mu' \tag{4}$$

using the constant $M$ which corresponds to the absolute maximal value of the controller output $u$.

The control algorithm of the neuro controller contains following two steps in each sampling period: 1. measuring of the controlled output $y(k)$ in step $k$ and

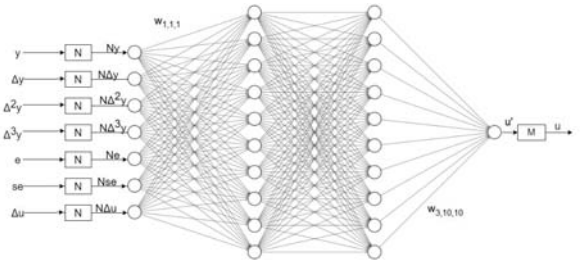


Figure 2: Block scheme of the considered MLP-type neuro-controller.

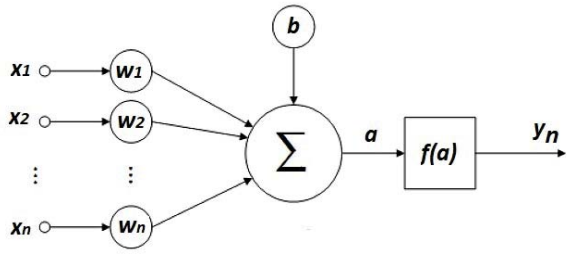


Figure 3: Block scheme of a single neuron of the MLP network, $f(a) = \tanh(a)$, $x_i$ are input signals into the neuron, $w_i$ are weight constants, $a$ is the internal activity of neuron and $y_n$ is the output of neuron.

preprocessing of the input signal vector $X$ consisting of the 7 input signals

$$X = \{y, \Delta y, \Delta^2 y, \Delta^3 y, e, se, \Delta u\} \tag{5}$$

2. signal propagation over the network and calculation of the new controller output $u(k)$.

## 3 The Neuro-Evolution-Based Controller Design

In order the network to be able to calculate the required output and to obtain the required closed-loop dynamic behaviour it must be correctly parametrized. The goal of the learning process is to find such parameters of the ANN, which minimise the control performance index of the closed-loop. The first considered performance index is a simple integral form criterion IAE – integral of the absolute control error

$$J_{IAE} = \int_{T_1}^{T_2} |e(t)|dt \tag{6}$$

where $T_1$ is the start time end $T_2$ is the stop time of closed-loop simulation. To dump oscillations of the system output an extended performance index can be considered in form

$$J = \int (\alpha|e| + \beta|e'|)dt \tag{7}$$

where $e'$ is the first derivative (or difference) of the control error. Note, that several other types of performance indices can be used for reaching various goals

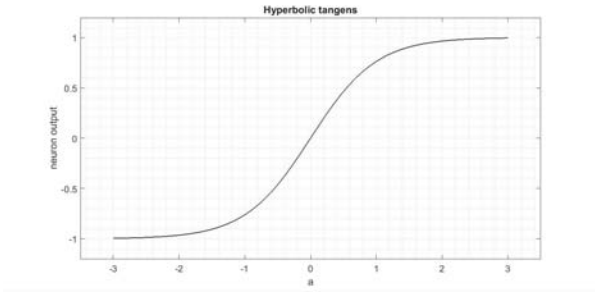


Figure 4: Graph of the neuron activation function $f(a) = \tanh(a)$.

as explained in [13, 14]. The ANN parameters are the items of the: 1. weight matrix $W_1$ (weights of the fully connected connection between the input vector $X$ and first hidden layer of neurons), 2. vector of biases of the first hidden layer $B_1$, 3. weight matrix $W_2$ (weights of the fully connected connection between the first and second hidden layer of neurons), 4. vector of biases of the second hidden layer B2, 5. weight matrix $W_3$ (weights of the fully connected connection between the second hidden layer of neurons and the single output neuron). All these parameters are parts of each chromosome (individual) of the population of the genetic algorithm which is in form

$$\begin{aligned} ch &= \{W_1, B_1, W_2, B_2, W_3\} \\ &= \{w_{1,1,1}, \ldots, w_{1,M_1,N_1}, b_{1,1}, \ldots, b_{1,N_1}, w_{2,1,1}, \\ &\quad \ldots, w_{N_1,N_2}, b_{2,1}, \ldots, b_{2,N_2}, w_{3,1}, \ldots, w_{3,N_2}\} \end{aligned} \tag{8}$$

where $M_1$ is the number of inputs (length of the input vector $X$), $N_1$ is number of neurons in the first hidden layer, $N_2$ is number of neurons in the second hidden layer.

As already mentioned, while searching for the optimal solution the genetic algorithm (GA) is minimising the cost function $fitness(ch)$

$$ch_{opt} = \arg\min fitness(ch) \tag{9}$$

The used genetic algorithm consists of following steps:

1. Random initialization of population of $N_pop$
2. Fitness function evaluation of each chromosome of the population - simulation of the closed-loop process and calculation of the fitness function.
3. If the predefined number of generations is reached, or the required terminating conditions are reached then finish, otherwise continue to step 4.
4. Selection of 40% parents$_1$ for crossover, selection of 40% parents$_2$ for mutation. Stochastic Universal Sampling Selection is used in both cases [2, 8, 9]. Selection of 2% of best chromosomes and selection of 18% random chromosomes which all will survive without modification.

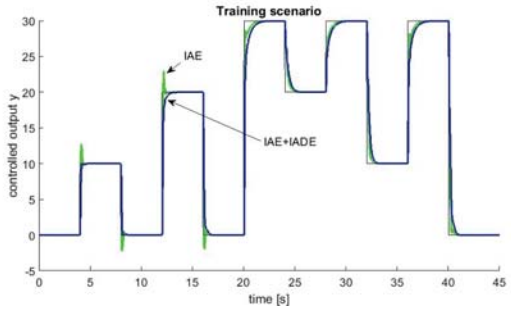


Figure 5: Time responses of the controlled value $y$, comparison of two performance indices IAE (green) and IAE+IADE (blue).

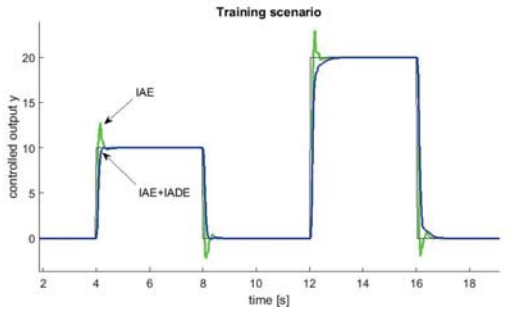


Figure 6: Time responses of the controlled value $y$ - detail, comparison of two performance indices IAE (green) and IAE+IADE (blue).

5. Modification of parents$_1$ by 1-point crossover = children$_1$. Modification of parents$_2$ by the mutation = children$_2$. Mutation rate is 0.1 (10% of genes in population are mutated).
6. Completion of the new population: children + best + unchanged chromosomes.
7. Back to step 2.

## 4 Experimental Results

The controlled dynamic system $S$ is described by the differential equation

$$\begin{aligned} &a_2\Delta^2 y + a_1\Delta y + a_0 y + a_3 y^3 - b_1\Delta u - b_0 u = 0; \\ &a_0 = 1;\ a_1 = 4;\ a_2 = 1;\ a_3 = 0.25;\ b_0 = 1;\ b_1 = 0.1 \end{aligned} \tag{10}$$

In the first experiment (Fig. 5) we compared two performance indices. The first is the simple IAE criterion according to equation (6) and the second is a combined criterion in form (7) with $\alpha = 1, \beta = 1$. The work was programmed in MATLAB [7]. In the training scenario several steps of reference signal were performed (black line), the neuro-controller closed-loop output with IAE criterion (6) is fast, but with oscillations and overshoot (green) and the aperiodic response, which is slower and damped was obtained using minimization of the second combined criterion (7) (blue). The detail of this experiment is in Fig. 6.

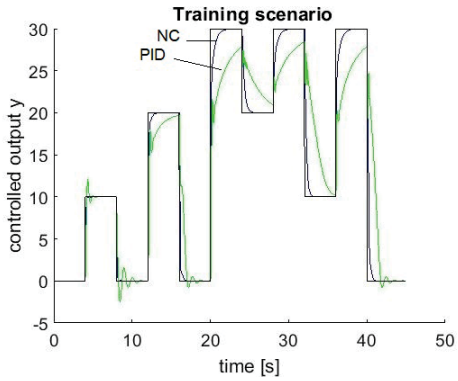


Figure 7: Time responses of the controlled value $y$ during training scenario using PID controller (green) and the neuro-controller (NC, blue).

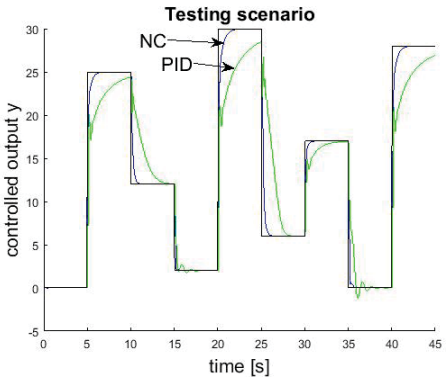


Figure 9: Time responses of the controlled value $y$ during testing scenario using PID controller (green) and the neuro-controller (blue).

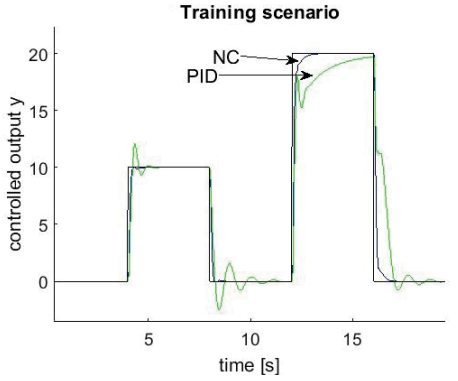


Figure 8: Time responses of the controlled value $y$ during training scenario using PID controller (green) and the neuro-controller (blue) - detail.

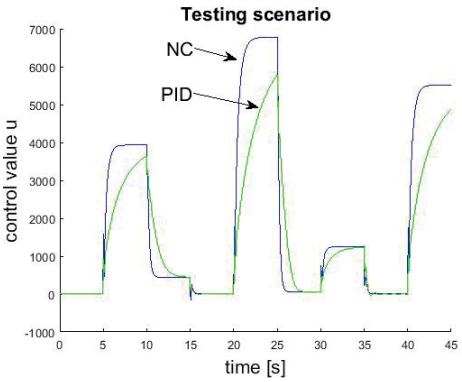


Figure 10: Time responses of the control value $u$ during testing scenario using PID controller (green) and the neuro-controller (blue).

The next experiment compares the PID controller with the neuro-controller. The PID was designed using genetic algorithm with the same criterion (7) $\alpha = 1, \beta = 1$. PID controller does not achieve good performance on controlling the non-linear system. In Fig. 7 and 8 the training scenario is depicted, which was used during the learning phase of the controller design. In Fig. 9 the results of the testing scenario are depicted. The testing scenario was not known during training process. In Fig. 10 the control value of this experiment is shown. In the last experiment time-responses of the controlled system with additional noise on its output $y$ with amplitude 1% of the controlled range during test scenario using PID controller and the neuro-controller are compared (Fig. 11).

## 5 Conclusion

An unsupervised-type learning procedure of ANN based on genetic algorithm was used for design of a neuro-controller of a non-linear dynamic system. This approach can achieve good results which highly outperform the results of a linear PID controller. In this project the genetic algorithm was used for the design of parameters of an a-priori defined architecture of the neural network controller. Feed-forward ANNs have been considered here. According to our experience the neuro-evolution of controllers is an efficient approach for solving a wide spectrum of controller design problems in continuous-time system control for non-linear dynamic systems, systems with complex behaviour as well as systems with multiple inputs and outputs.

## References

[1] Aamir, M. On replacing pid controller with ann controller for dc motor position control. *arXiv preprint arXiv:1312.0148* (2013).

[2] Eiben, A. E., Smith, J. E., et al. *Introduction to evolutionary computing*, vol. 53. Springer, 2003.

[3] Harp, S. A., Samad, T., and Guha, A. Designing application-specific neural networks using the genetic algorithm. In *NIPS* (1989), vol. 2, Citeseer, pp. 447–454.

[4] Harp, S. A., Samad, T., and Guha, A. Towards the genetic synthesis of neural network. In

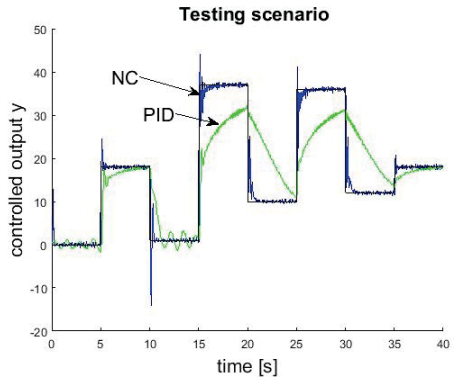


Figure 11: Time responses of the controlled system with additional noise on its output $y$ (with amplitude 1% of the controlled range) during test scenario using PID controller (green) and the neuro-controller (blue).

*Proceedings of the third international conference on Genetic algorithms* (1989), pp. 360–369.

[5] Jalali, S. M. J., Kebria, P. M., Khosravi, A., Saleh, K., Nahavandi, D., and Nahavandi, S. Optimal autonomous driving through deep imitation learning and neuroevolution. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)* (2019), IEEE, pp. 1215–1220.

[6] Kumar, R., Srivastava, S., and Gupta, J. Artificial neural network based pid controller for online control of dynamical systems. In *2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)* (2016), IEEE, pp. 1–6.

[7] MATLAB. *version R2017a*. The MathWorks Inc., Natick, Massachusetts, 2017.

[8] Matousek, R., Dobrovsky, L., and Kudela, J. How to start a heuristic? utilizing lower bounds for solving the quadratic assignment problem. *International Journal of Industrial Engineering Computations 13*, 2 (2022), 151–164.

[9] Michalewicz, Z., and Michalewicz, Z. *Genetic algorithms+ data structures= evolution programs*. Springer Science & Business Media, 1996.

[10] Montana, D. J., Davis, L., et al. Training feedforward neural networks using genetic algorithms. In *IJCAI* (1989), vol. 89, pp. 762–767.

[11] Panbude, A., and Sharma, M. Implementation of neural network for pid controller. *International Journal of Computer Applications 975* (2015), 8887.

[12] Prados, D. L. New learning algorithm for training multilayered neural networks that uses genetic-algorithm techniques. *Electronics Letters 28*, 16 (1992), 1560–1561.

[13] Sekaj, I. Evolutionary based controller design. In *Evolutionary Computation (SMC) (Editor: Wellington Pinheiro dos Santos)* (2009), In-Tech.

[14] Sekaj, I. Control algorithm design based on evolutionary algorithms. In *Introduction to Modern Robotics. iC. Press, Hong Kong* (2011).

[15] Sekaj, I., Cíferský, L., and Hvozdík, M. Neuro-evolution of mobile robot controller. *Mendel Journal 25*, 1 (2019), 39–42.

[16] Stanley, K. O., Clune, J., Lehman, J., and Miikkulainen, R. Designing neural networks through neuroevolution. *Nature Machine Intelligence 1*, 1 (2019), 24–35.

[17] Stanley, K. O., and Miikkulainen, R. Evolving neural networks through augmenting topologies. *Evolutionary computation 10*, 2 (2002), 99–127.