27/28th International Conference
ENGINEERING MECHANICS 2022
Milovy, Czech Republic, May 9 –12, 2022
Paper **#61,** pp. 325–328, doi: 10.21495/51-2-325

COMPARISON OF APPROACHES TO AIR SPRING HEIGHT CONTROL

Rágulík J.*, Sivčák M.*

Abstract: *The paper deals with the comparison of control approaches to the height of the bellows air spring. These approaches are PI control, a deep reinforcement learning algorithm, and a combination of both control techniques, the neuro-PI controller. In the case of a neuro-PI controller, it is essentially control by a reinforcement learning algorithm, but its control neural network contains only two weights that are gained in the learning process. The advantage of the neuro-PI controller created in this way is that its functionality can be formally verified in the same way as a classic PI controller. However, such a procedure is only possible when controlling relatively simple systems, such as the described application for controlling an air spring, which is a dynamic system with one degree of freedom. All the above approaches are applied to a mathematical model of an air spring created in the Simulink environment; a deep reinforcement learning algorithm is created in MATLAB.*

Keywords: Deep reinforcement learning, PI control, Formal verification, Deep neural networks, Air spring.

1. Introduction

Deep reinforcement learning (DRL) is a field of artificial intelligence. It was created by combining the fields of deep learning and reinforcement learning. Deep learning deals with deep neural networks, which are artificial neural networks that have several hidden layers. Reinforcement learning (Fig. 1) is an area of machine learning that looks at how smart agents should act in an environment to maximize the expected cumulative reward, which is a qualitative description of the actions taken. An agent is usually a control algorithm and an environment represents a controlled system.

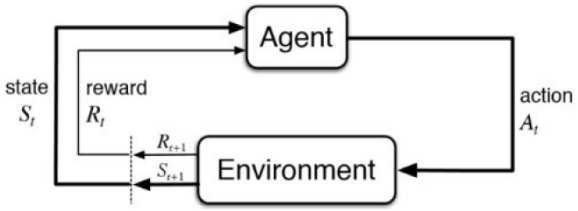

Fig. 1: Reinforcement learning scheme (Sutton and Barto, 1998)

The best-known applications of DRL algorithms include image classification, speech recognition, computer and classical gaming (Silver et al., 2016), robotics and market prediction applications. In some cases, for example when playing complex computer games, it is not possible to create a game boot (using classical programming methods) that would at least approach the level achieved using artificial intelligence algorithms. The problem arises when applying these algorithms to safety-critical systems. Such a system is, for example, autonomous vehicles.

The problem occurs with large deep neural networks. Such networks have several layers. Usually an input layer, several hidden layers and an output layer, where each of these layers can contain hundreds of neurons. Assuming the use of a fully-connected network architecture, each neuron is connected to all neurons in all adjacent layers. Networks are even more complex when using image classification. For example, when

* Ing. Jiří Rágulík.: Department of Applied Mechanics, Technical University of Liberec, Studentská 2, 461 17 Liberec. Czechia, jiri.ragulik@tul.cz

* Ing. Michal Sivčák, PhD.: Department of Applied Mechanics, Technical University of Liberec, Studentská 2, 461 17 Liberec. Czechia, michal.sivcak@tul.cz

navigating a robot through an obstacle environment, it is necessary to determine the appropriate trajectory of the robot's movement based on the classification of data from the cameras and to act on the motors accordingly. Image processing usually uses convolutional neural networks, which essentially serve as video input filters. Several layers of convolutional networks are usually followed by several fully-connected layers with a high number of neurons in the hidden layers of the network. Such complex networks function as so-called black-boxes (Ismailov, 2014).

Deep neural networks are trained on a finite set of data and are thus capable of some degree of generalization. In practice, however, with increasing output quality, there is usually a decrease in the ability to generalize (Cobbe et al., 2018). An example is the game of chess. Today's algorithms are able to beat human professional players quite reliably. In the event of a minor change in the rules, such as removing one of the towers from both players or changing the size of the playing field, the algorithms begin to fail, while the human player's ability to improvise and make meaningful moves based on game knowledge and experience can be assumed. Due to the limited generalization capability, the network may respond poorly to inputs it has never seen before or be too sensitive to input disturbances (Bastani et al., 2016). Thus, networks can behave unpredictably and give poor results, which is why it is problematic to apply them to safety-critical systems (Huang et al., 2017).

Formally verifying means mathematically proving the functionality of the program. The program is practically unverifiable if it is taught on the fly (Katz et al., 2017). However, if a relatively small neural network is trained first and then it is applied without change, convergence in machine learning algorithms can be ensured by standard verification procedures (Hull et al., 2002). It is usually necessary to limit the program at runtime, i.e., to continuously monitor the operation of the program and to check compliance with the set limits.

2. PI controller tuning and training of neural networks

The mathematical model created in Simulink is in the form of a following differential equation, the coefficients of which were obtained experimentally on a real spring, where x represents the height of the spring, F_0 the force to reach the free length of the spring and F the excitation force:

$$40 \cdot \ddot{x}(t) + 600 \cdot \dot{x}(t) + 3000 \cdot x(t) + F_0 = F(t). \tag{1}$$

Specifically, it is the coefficient of stiffness and damping of the entire pneumatic circuit, consisting of a spring, pressure regulator and pneumatic hoses. The pressure regulator used has a built-in control circuit of unknown characteristics and its action in the circuit significantly affects the stiffness and damping of the system. The state of the system is the height of the spring, the action is a change in pressure inside the bellows of the spring.

The PID controller was set by the Ziegler-Nicols heuristic method (Ziegler and Nichols, 1942; Haugen et al., 2013) and applied in the Simulink environment to a mathematical model of a bellows air spring.

The Twin-Delayed DDPG (TD3) algorithm (Fujimoto et al., 2018), a successor to the frequently used Deep Deterministic Policy Gradient (DDPG) algorithm, was selected from a large family of DRL algorithms (Lillicrap et al., 2015). Both algorithms fall into the category of actor-critic algorithms, where the agent consists of two deep neural networks, an actor which, based on the state of the environment, performs actions on the environment and a critic who adjusts the weights of the actor network based on the suitability of actions. Unlike DDPG, TD3 uses two critic networks, i.e., two Q-functions, and uses the output of a Q-function with a lower Q-value to form the targets in the Bellman error loss functions. Another advantage is that the algorithm updates the actor less often than its Q-functions, represented by critic networks. TD3 is significantly more stable in the learning process than DDPG and other older algorithms (Fujimoto et al., 2018). Frequently used hyper-parameter values, learning rate 0.001, discount factor 0.97 and mini-batch size 512 were used in the learning algorithm process. All neural networks have two hidden layers and the size of each of these layers is 200 neurons.

After setting up the PID controller by the Ziegler-Nicols method and learning the TD3 algorithm, a neuro-PI controller was built. It was created using the same already programmed algorithm, but the size of the actor network was significantly reduced (Fig. 2). The input layer still had two neurons and the output layer one neuron. No hidden layers were used. Thus, there will be only two weights in the whole actor network, both input neurons are connected to the output neuron. There are no activating functions in such a reduced actor, and after learning it is therefore easier to formally verify, because activation functions are one of the

main problems in verification, due to its nonlinearity (Katz et al., 2017). The size of the critic networks, the reward function and all other hyper-parameters were kept the same as in the previous TD3 training. The inputs to the agent were chosen in the same way as for training with the TD3 deviation algorithm, i.e., the difference between the required and actual spring height *e* and the integral of this deviation. The weights in the neural network, w_1 and w_2, can thus be considered as the gains of the individual components of the PI controller. Even in this procedure, the use of the derivative component of the regulator proved to be unsuitable.

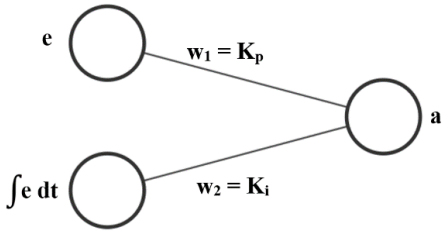

Fig. 2: Neuro-PI regulator scheme

3. PI controller tuning and training of neural networks

The system was excited by the force of the step, sine and saw curves and was additionally loaded with weights of known weight. The following graph (Fig. 3) shows the achieved control quality. The mathematical model of the pneumatic spring was excited at time 3 s after stabilization at the required height by a step change of the loading force from 0 to 400 N. The required height was 115 mm and the free length of the spring was 112 mm. The working range of the spring used is from 92 mm to 132 mm. The system was statically loaded with a weight of 40 kg.

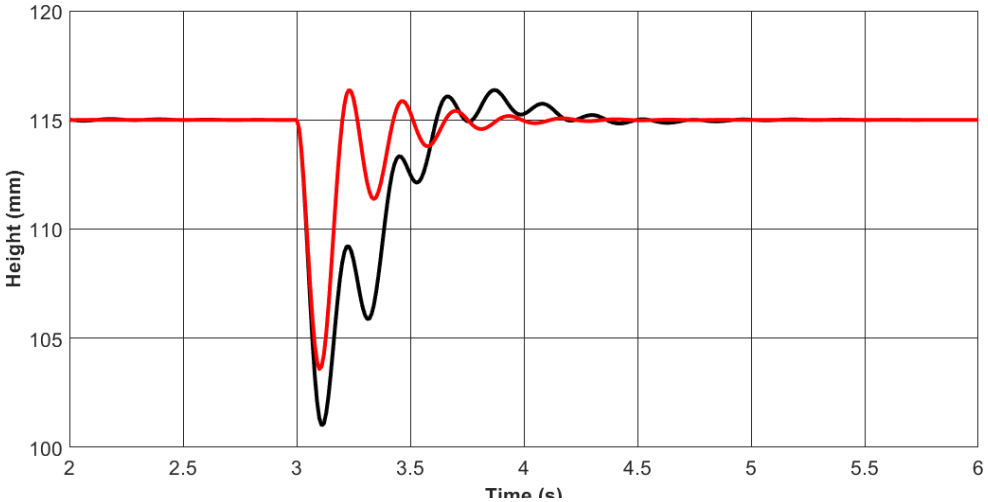

Fig. 3: Air spring height change under load by step change of force during control by both PI controllers (red) and TD3 algorithm (black)

The best results were obtained with a neuro-PI controller and a practically identical result was obtained with a PI controller set by the Ziegler-Nicols method. Only the result of the neuro-PI regulator is shown in the graph, because it would practically merge with the latter method. For a given controlled system, it was possible to change the integration gain by up to 10 % without a visible effect on the quality of regulation. The proportional gains are almost the same and even attempts at small manual tuning have not improved. It can therefore be said that this is a practically ideal setting of the controller. The amplifications obtained by both methods are shown in the following table (Tab. 1).

Tab. 1: Gains of PI controllers

Method	Proportional gain K_p	Integral gain K_i
Ziegler-Nicols	196.7	4.85
Neuro-PI	200.0	4.82

Slightly lower control quality was achieved by the TD3 algorithm. A higher overshoot is noticeable, but the settling time is similar. The algorithm was trained by force excitation with a time-varying course, specifically with sine, jump and sawtooth waveforms. The worst results are achieved with the step excitation waveform, but for the sine and saw waveforms, almost the same control quality is achieved as with the PI controller. Better results could be achieved by optimizing the hyper-parameters of the algorithm. However, this is a sufficient quality of regulation so that it is possible to apply the algorithm thus learned to a real spring and then perform learning on it. Learning on a real spring should bring a significant improvement over the application of an agent trained only on a mathematical model, because the mathematical model is not able to accurately describe the behavior of the spring.

4. Conclusions

The paper presents a description and comparison of the results obtained when controlling an air spring with a PI controller, TD3 algorithm and neuro-PI controller, whose main advantage over the TD3 algorithm is formal verifiability. The results obtained by PI and neuro-PI regulation were practically identical. The quality of control achieved using the DRL algorithm is not as high as in previous approaches, but it could certainly be increased by optimizing the hyper-parameters of the algorithm. However, since the next research procedure focuses on the application of the algorithm to the real spring and the thus trained agent will be used at the beginning of the subsequent learning on the real spring, the quality of the regulation is sufficient. Such pre-learning will significantly reduce the time required for learning.

Acknowledgement

This publication was written at the Technical University of Liberec as part of the project "Research of advanced materials, and application of machine learning in the area of control and modeling of mechanical systems" nr. SGS-2022-5072 with the support of the Specific University Research Grant, as provided by the Ministry of Education, Youth and Sports of the Czech Republic in the year 2022.

References

Bastani, O., Ioannou, Y., Lampropoulos, L., Vytiniotis, D., Nori, A. and Criminisi, A. (2016). Measuring neural net robustness with constraints, In *NIPS'16: Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 2621–2629

Cobbe, K., Klimov, O., Hesse, C., Kim, T. and Schulman, J. (2019) Quantifying Generalization in Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning*, PMLR 97, pp. 1282-1289.

Fujimoto, S., Hoof, H. and Meger, D. (2018) Addressing Function Approximation Error in Actor-Critic Methods. In *35th International Conference on Machine Learning*, pp. 1582-1591

Haugen, F.A., Lie, B., Haugen, F., Bakke, R. and Lie, B. (2013) Relaxed ZieglerNichols Closed Loop Tuning of PI Controllers. Modeling, *Identification and Control: A Norwegian Research Bulletin*. 34. 83-97.

Huang, X., Kwiatkowska, M., Wang, S. and Wu, M. (2017). Safety Verification of Deep Neural Networks. In: Majumdar, R. and Kunčak, V. (eds) *Computer Aided Verification. CAV 2017*. Lecture Notes in Computer Science, vol 10426. Springer, Cham, pp. 3–29. 10.1007/978-3-319-63387-9_1.

Hull, J., Ward, D. and Zakrzewski, R. R. (2002). Verification and validation of neural networks for safety-critical applications. *Proc. of the 2002 American Control Conference,* Vol 6. 4789-4794, 10.1109/ACC.2002.1025416.

Ismailov, V. (2014). On the approximation by neural networks with bounded number of neurons in hidden layers. *Journal of Mathematical Analysis and Applications*, 417, pp. 963–969. 10.1016/j.jmaa.2014.03.092.

Katz, G., Barrett, C., Dill, D.L., Julian, K. and Kochenderfer, M.J. (2017) Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In: Majumdar, R. and Kunčak, V. (eds) *Computer Aided Verification. CAV 2017*. Lect. Notes in Computer Science, vol 10426. Springer, Cham, pp. 97–117. 10.1007/978-3-319-63387-9_5.

Lillicrap, T., Hunt, J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *4th International Conference on Learning Representations*, ICLR 2016.

Silver, D. et al. (2016) Mastering the game of Go with deep neural networks and tree search, *Nature*, 529, pp. 484–489. 10.1038/nature16961.

Sutton, R.S. and Barto, A.G. (1998). Reinforcement Learning: An Introduction. *IEEE Transactions on Neural Networks,* 9(5), 1054–1054. https://doi.org/10.1109/tnn.1998.712192

Ziegler, J.B. and Nichols, N.B. (1942) Optimum settings for automatic controllers, *Transactions of the ASME*, 64, pp. 759–768.