Policy gradient based Reinforcement learning control design of an electro-pneumatic gearbox actuator

Tamás Bécsi ∗ Szilárd Aradi ∗ Ádám Szabó ∗ Péter Gáspár ∗

Department of Control for Transportation and Vehicle Systems, Budapest University of Technology and Economics (e-mail: {becsi.tamas, aradi.szilard, szabo.adam, gaspar.peter}@mail.bme.hu)

Abstract: The paper presents a reinforcement learning based solution for the control design problem of a gearbox actuator. The system is operated by an electro-pneumatic, three-state, floating piston cylinder. Besides the primary goals of positioning the piston, the nonlinear system’s quality objectives are to minimize switching time and overshoot. The control strategy based on the measurable parameters of the system is realized by a dense feedforward neural network. With the utilization of the policy based reinforcement learning architecture, the learning agent develops the optimal strategy for fast and smooth switching, under different and changing conditions.

Keywords: Reinforcement Learning Control, Mechatronic systems, Automotive sensors and actuators, Non-Linear Control Systems

1. INTRODUCTION

Pneumatic actuators achieve force transmission and produce reciprocating linear motion by using the energy of the compressed gas. Pneumatically driven actuators are widely used in industrial applications, mainly in the field of automation, manufacturing and robotics. They are applicable on a wider temperature domain than the hydraulic actuators, and they have higher power density and lower specific weight than an equivalent electro-mechanic actuator. There is an unlimited supply of air to be compressed, which can be easily transported through tubes, and can be stored, while the exhausted air do not need to be collected, which means fluid return lines are unnecessary. These type of actuators are easily maintainable, they have high operational safety and long lifetime. A review of the industrial application of pneumatic systems can be found in Saravanakumar et al. (2017).

Researches connected to the vehicle industry focus on the single- and double acting pneumatic cylinders. These are commonly used in systems, like exhaust gas recirculation, air brake systems Karthikeyan et al. (2011), turbocharger applications Mehmood et al. (2011), electro-pneumatic clutch Szimandl and Németh (2013a) and gearbox actuator systems Szimandl and Németh (2009). While many of the recent studies on the field of robotics focuses on the modeling and control of pneumatic muscle actuators (PMA), such as Hošovský et al. (2016), Oliver-Salazar et al. (2017) and Doumit and Pardoel (2017), since they are one of the most promising actuators for applications that require greater proximity between the humans and the robots.

Besides the advantages of the electro-pneumatic systems, their nonlinear behavior makes predicting and controlling them difficult. Palomares et al. (2017) In literature one of the proposed methods is PID control, which should be used in cascaded control Saleem et al. (2015), or should be enhanced through gain scheduling. Other suggested control methods are the linear quadratic (LQ) control Szimandl and Németh (2009), which provides optimal control solution for a given cost function, the sliding mode control Szimandl and Németh (2013b) and H-infinity method Szimandl and Németh (2014). Further applicable controllers are the fuzzy-based controllers Nuchkrua and Leephakpreeda (2013), the adaptive controllers Liu et al. (2013) and the Neural Network based controllers, such as Son et al. (2017), Chiang and Chen (2017) and XiaoJun et al. (2006).

The paper presents the development of a control system based on feedforward neural networks, through self-learning.

The paper is organized as follows: Section 2 describes the modeled actuator and Section 3 presents the developed non-linear model. Section 4 provides methodological background on the used algorithm, Section 5 describes the training environment, simulation model and rewarding system, and finally Section 6 provides results.
2. SYSTEM DESCRIPTION

The modeled actuator is used for the automated manual transmission of a heavy duty vehicles, where it shifts the proper gear inside a previously selected lane or sets the gearbox to neutral.

![Simplified layout of the modeled gearbox actuator](image)

Fig. 1. Simplified layout of the modeled gearbox actuator

There are two working chambers, a control chamber, and two pistons distinguished inside the cylinder. Both Chamber 1 and Chamber 2 are connected to a 3-way 2-position solenoid valve which serves both input and output purposes. In released state the solenoid valves connect the chambers to the ambient pressure, while in energized state they connect them to the supply pressure. The layout of the modeled system can be seen in Figure 1.

The actuator has three dedicated positions: two gears (High and Low end positions of the piston) and Neutral position. To shift between different gears the main piston actuates the shift finger and through it the gear shift linkage and the synchronizer sleeve. Its movement is generated by the opposing pressure forces inside the chambers, while the control chamber serves as an air spring with no transmission of a heavy duty vehicles, where it shifts the proper gear inside a previously selected lane or sets the gearbox to neutral.

3. SIMPLIFIED, NONLINEAR MODEL OF AN ELECTRO-PNEUMATIC ACTUATOR

The presented model is derived from the detailed nonlinear model of an electro-pneumatic actuator described in Ádám Szabó et al. (2018). The applied simplifications are the following:

- Solenoid valve models are substituted with their mass flow rates
- Detent mechanism is disregarded
- Contact forces are disregarded
- Control chamber pressure is assumed to be equal to the ambient pressure
- Based on the main piston position, the floating piston is either at neutral position, or it is assumed to move together with the main piston

The system contains more than one element that shows hybrid behavior, therefore the equations describing the system change according to defined conditions. Model parameters can be classified into four groups: environmental and supply parameters, construction parameters, tunable parameters, and state variables. Inputs of the model are solenoid valve commands, the simulation timestep, and the state variables of the system: the pressure and temperature of the working chambers, the main piston position and the main piston velocity. Modeled outputs are the state variables of the system. With the given simplifications the system has two model parts: chamber thermodynamics and chamber mechanical dynamics.

3.1 Chamber thermodynamics

The thermodynamical model is based on the conservation of mass and conservation of energy inside the chamber. In case of lumped parameter system with no generation and consumption terms the conservation of mass can be written as the sum of the input and output mass flow rates of the system. Since both working chambers are loaded and unloaded through a sole 3/2 solenoid valve, the mass changes of the chambers are equal to the mass flow rate of the solenoid valves. In case of the control chamber the mass flow rate between the chamber and its environment is caused by the volume change of the chamber. The mass flow rate for all three chambers can be written as:

\[
\frac{dm_{ch}}{dt} = u_{sv} \cdot A_{fl} \cdot p_{1} \sqrt{\frac{2k}{k-1}} \left( \frac{1}{R_{air} \cdot T_{1}} \right) \left( \frac{\pi}{2} - \frac{\pi}{2} \right) \tag{1}
\]

where \( u_{sv} \) is the solenoid valve command, \( A_{fl} \) is the area at vena contracta, \( k \) is the heat capacity ratio, \( R_{air} \) is the gas constant for air, \( T_{1} \) is the temperature inside the chamber, \( T_{2} \) is the ambient temperature and \( p \) is the pressure ratio, which can be determined as:

\[
\pi = \begin{cases} 
\frac{p_{2}}{p_{1}}, & \text{if } \frac{p_{2}}{p_{1}} \geq \pi_{crit} \\
\pi_{crit}, & \text{if } \frac{p_{2}}{p_{1}} < \pi_{crit}
\end{cases} \tag{2}
\]

where \( p_{1} \) is the source side pressure and \( p_{2} \) is the counter side pressure.

The general form of total energy for a given balance volume with \( p \) input and \( q \) output flows is written as:

\[
\frac{dE}{dt} = \sum_{j=0}^{p} m_{in}^{j} (h + e_{k} + e_{p}) - \sum_{k=0}^{q} m_{out}^{k} (h + e_{k} + e_{p}) + W + Q
\tag{3}
\]

where \( h, e_{k} \) and \( e_{p} \) denotes the mass specific enthalpy, kinetic energy and potential energy terms respectively.
$Q$ is the heat transfer and $W$ is the work term. With respect to the given simplifications, the chamber pressure gradient can be expressed from the conservation of energy as follows:

$$\frac{dp_{ch}}{dt} = \frac{\kappa_{air} R_{air} T_{inw} n_{ch} - k_{ht} A_{ht} (T_{ch} - T_{amb})}{V_{ch}} - \frac{\kappa_{air} p_{ch} \frac{dv_{ch}}{dt}}{V_{ch}}$$  \hspace{1cm} (4)

where $T_{inw}$ is the temperature of the flowing air, $V_{ch}$, $p_{ch}$ and $T_{ch}$ are the volume, pressure and temperature of the chamber, $k_{ht}$ and $A_{ht}$ are the heat transfer coefficient and the heat transfer area and $T_{amb}$ is the ambient temperature.

The volume of the chambers is given as the summary of cylinders and rings with height taken from the axial displacement of the concerned elements, while the temperature of the chamber can be calculated according to the ideal gas law.

### 3.2 Chamber mechanical dynamics

The mechanical model of the actuator is based on the conservation equation for momentum. The applied momentum balance of the pistons can be written as:

$$\frac{dv_{fp}}{dt} = \sum F_{p} - F_{fp} + F_{fc}$$  \hspace{1cm} (5)

where $F_{p}$ is the pressure force, which can be calculated from the pressures applied to the pistons, $(F_{fp})$ and $F_{fc}$ are the viscous friction and the Coulomb friction. The piston area in Chamber 1 is a position dependent hybrid parameter, which can be expressed as follows:

$$A_{p2} = \begin{cases} A_{mp}, & \text{if } x_{mp} < 0 \\ A_{mp} + A_{fp}, & \text{if } x_{mp} \geq 0 \end{cases}$$  \hspace{1cm} (6)

where $mp$ refers to the main piston, and $fp$ refers to the floating piston.

There are no contact forces calculated in the model, therefore the piston positions are saturated at the end positions of the cylinder. Since the floating piston is either at neutral position, or it is assumed to move together with the main piston, the piston mass can be written as:

$$m_{p} = \begin{cases} m_{mp}, & \text{if } x_{mp} < 0 \\ m_{mp} + m_{fp}, & \text{if } x_{mp} \geq 0 \end{cases}$$  \hspace{1cm} (7)

4. MACHINE LEARNING-BASED CONTROLLER DESIGN

In the recent years the machine learning is spreading rapidly in the field of control design. The researchers primarily focus on very complex environments which can be sensed and controlled only by humans. Typical examples are the video and table games and self-driving cars. However these algorithms can be used efficiently in mechatronics systems where the environment is simpler, but the behavior is non-linear, thus the control design is more difficult.

Nowadays reinforcement learning is one of the fastest developing machine learning areas. Our gearbox actuator fits well into this paradigm, because its operation is episodic, the control signals are discrete and the control quality parameters can be easily quantified.

### 4.1 Reinforcement Learning

In reinforcement learning (as in other areas of artificial intelligence) the learner and decision maker is called the agent. The thing it interacts with, including everything outside the agent, is called the environment. These interact continually, the agent is selecting actions and the environment is responding to these actions and presenting new situations to the agent (see Fig. 2). The terms agent, environment, and action meet the control engineers’ terms controller, plant and control signals. In reinforce learning the fully-observable environment can be typically formulated as finite-state Markov Decision Process (MDP). It is described with a tuple $(s_{t}, a_{t}, s_{t+1})$ where $s_{t} \in S$ represents the system’s state space, action $a_{t}$ changes the environment state from $s_{t}$ to $s_{t+1}$ with transition $P(s_{t}, a_{t}, s_{t+1})$. The agent and environment interact at each of a sequence of discrete time steps, $t = 0, 1, 2, 3, ...$

Finally the most interesting part of reinforcement learning is the numerical reward $r_{t+1} \in R$ that maximizes the cumulative future reward $R = r_{0} + r_{1} + ... + r_{n}$, exactly the discounted future reward $R = \sum \gamma^{t} r_{t}$. The discount factor $0 \leq \gamma \leq 1$ represents the uncertainty of the future, i.e. how much the future rewards depend on the actions in the past. A prediction of cumulative future reward $V_{\pi}(s) = E_{\pi}[R|S_{0} = s]$ is defined, the “value” of the state which can be used evaluates the badness or goodness of the state and therefore to select between actions. There exists an optimal policy $\pi^{*}$ for which $V^{*}$ is optimal for every state. In a finite MDP, the sets of states, actions, and rewards $(S,A,R)$ all have a finite number of elements. The Markov property of MDP requires that the probability distribution of random variables (transitions and rewards) depend the current state and action only and do not depend on past actions and states.

![Fig. 2. Agent-environment interaction in reinforcement learning](image-url)

Depending on the selected method the RL agent may include policy function, value function and model. The latter is the agent’s representation of the environment, according to its presence in the solution we can categorize RL methods as model-free or model-based solutions.

### 4.2 Value-based Methods

The value function-based reinforcement learning is a heavily researched area thanks to Deepmind’s improvements in deep Q learning. The main idea is to learn a Q-function
instead of the value function. Q-function is a predictor function which outputs a Q-value for each action in a given state. The prediction can be updated according to the Bellmann equation:

\[ Q(s_t, a_t) = r_t + \gamma \max_a Q(s_{t+1}, a) \]  

(8)

The predictor function is typically a deep neural network where the training goal is the more accurate estimate of the Q-function according to the Bellmann equation. It should be noted that the usage of function approximation in RL algorithms do not guarantee the convergence, but these methods usually show good results in practice, in particular with regard to the improved versions, see e.g. van Hasselt et al. (2015) and Wang et al. (2015).

4.3 Policy-based Methods

The other approach is based on the approximation of the policy directly. From this group of RL solutions the policy-gradient methods have gained interest in recent years to solve control problems. These algorithms modifies the parameters of the function approximator (neural network) in the direction that maximizes the expected reward. Its advantage over the value-based methods is the guaranteed convergence, but typically to a local rather than global optimum.

Policy gradient assumes that the actions are given by a stochastic policy \( \pi_\theta(a_t|s_t) \) with parameters \( \theta \). Policy-base RL is an optimization problem, where the goal is to find \( \theta \) that maximizes \( J(\theta) = J(\pi_\theta) \) where

\[ J(\pi_\theta) = \mathbb{E} \left[ \sum_{t=0}^{T} r_t \right] \]  

(9)

in episodic environments. There are several optimization techniques available, some of them do not use gradient, but usually greater efficiency can be achieved using gradient. The policy gradient algorithms search for a local maximum in \( J(\theta) \) by continuously refining a given parameterization vector. It follows the steepest ascent of the expected reward, which can be formulated by the gradient update rule:

\[ \Delta \theta = \alpha \nabla_\theta J(\theta) \]  

(10)

where \( \nabla_\theta J(\theta) \) is the policy gradient vector and \( \alpha \) the learning rate.

Next step is computing the gradients, which can be realized in several different ways. The most straightforward method, which can be easily implemented is computing gradients by finite differences. It is sensitive to how the policy is parameterized, noisy and inefficient, but it works even if the policy is not differentiable.

4.4 Policy Gradient Method

We have applied a policy gradient method based on the Williams’s theory (see Williams (1992) and policy gradient theorem of Sutton et. al. (see Sutton et al. (2000)). This theorem generalizes the likelihood approach to multi-step MDPs. Accordingly the policy gradients can be calculated and the parameters can be updated, thus the core algorithm works as follows

(1) Initialize \( \theta \) and the starting state \( s_1 \)
(2) Run an episode until terminated and store history
(3) Discount rewards \( r_1, r_2, \ldots, r_T \)
(4) For each episode \( s_1, a_1, r_2, \ldots, s_T-1, r_T \) update the gradients:

\[ \theta \leftarrow \theta + \alpha \nabla \log \pi_\theta(s_t, a_t)v_t \]

In our implementation the policy is approximated with an artificial neural network with fully connected hidden layers using ReLu activations. The input layer receives the state observation and the outputs are the probabilities of the discrete actions. During the training phase the actions is selected with a random choice using the probabilities, which acts as an exploration process. Vector \( v_t \) contains the discounted reward of the episode and the gradients are estimated as in Peters and Schaal (2006), thus the loss function for the neural network optimization can be formalized as follows

\[ \text{Loss} = -1/T \sum_{t=0}^{T} \theta \log \pi_\theta(a_t|s_t)v_t \]  

(11)

Since the noisy gradient updates can cause unstable convergence, the parameters \( \theta \) are not updated after each episodes, but accumulated over a number of episodes and then applied to the network. It can reduce the gradient variance which might stabilize the learning. It can be further improved with rescaling the rewards. A normalization procedure runs during the training phase using the running means and standard deviations calculated for each steps, across different episodes.

After the training phase the controller should be tested what is called evaluation process. The controller is the policy approximator ANN, which receives the state vector and the output is the action probability vector \( A_t \) and the selected action is \( a_t = \arg \max_i (A_t[i]) \). While the training process needs high computing power, the the controller ANN can be easily implemented even into an embedded system with limited resources.

5. TRAINING ENVIRONMENT

Reaching the three dedicated states of the actuator seems trivial, since the two end-positions can be reached by actuating the opposite valve, and the neutral position can be reached by actuating both valves, though the accuracy of the position can depend on the design parameters of the actuator. On the other hand When using as an automatic gearbox actuator, there are also quality requirements against the control, which serve passenger comfort and increases expected lifetime. In summary the requirements are the following:

- Tracking of a reference position signal
- Maximum 80ms shift time
- Maximum 0.2m/s impact velocity at reaching the end positions of the cylinder
- Maximum 1mm overshoot when reaching neutral state
- Maximum 6 solenoid valve actuation

The sampling time of the parameters of the model in the actual application is 3ms which is also the update frequency of the valve controls. The environment model uses Euler solver and 0.03ms simulation step for numerical
stability purposes. To ensure diverse environment for the agent’s learning process, some main parameters are altered for each training episode, such as the environmental and supply pressures and temperatures, and naturally low level noise is also added to the forces acting against piston movement.

The agent’s actions are based on the observation state vector that consist of the measured temperature and pressure values of the to chambers and also the position and velocity of the main piston.

The provided example trains the system to switch from low end-position to neutral. In each control step, the agent gets the state vector as input and determines the control commands for the two solenoid valves. One episode of training lasts as long as one of the termination causes occur:

- Time limit $T_L = 80 ms$ reached;
- The piston overshoot is above 1mm;
- The piston stops in time at the desired position.

Naturally the first two cases mean that the episode was not successful since the primary goals are not fulfilled. This also means that zero reward is given at the end. Case three is defined, that the piston distance from the neutral position is below 0.1mm and absolute piston speed is also below 0.004m/s. In this case, the the primary control goals are reached and the reward is based on two subreward, time based and overshoot based rewards. The time based reward encourages the agent to reach the control goal as fast as possible, it’s formulation is the following:

$$R_T = 1 - \frac{t_s}{T_L}, \quad (12)$$

where $R_T$ is the time based subreward, $t_s$ is the control time and $T_L$ is the time limit. The role of overshooting is much more interesting. At first thought overshoot subreward may have the same linear $[0,1]$ value between maximal and zero overshoot. Though this reward results in that the agent totally eliminates overshooting but the control time grows. So the reward for this property is 1 till the half of the limit is reached and only decreases afterwards:

$$R_O = \begin{cases} 
1, 
\frac{x_{mp}^{\text{max}}}{x_{lim}}, & \text{if } x_{mp}^{\text{max}} < 0.5x_{lim} \\
2 \left( 1 - \frac{x_{mp}^{\text{max}}}{x_{lim}} \right), & \text{otherwise} 
\end{cases} \quad (13)$$

The final reward is a combination of a constant and the two subrewards:

$$R = \alpha_1 + \alpha_2 R_T + \alpha_3 R_O, \quad \text{having} \quad \alpha_1 + \alpha_2 + \alpha_3 = 1 \quad (14)$$

6. RESULTS

The performance of the reinforcement learning’s trial-and-error algorithm, is highly affected by its hyperparameters. After several iterations the chosen values for these parameters, the network structure and also the reward function parameters are summarized in Table 1.

In the present case the most significant hyperparameters are the learning rate ($\alpha$), the discount factor ($\gamma$) and the number of episodes ($\xi$) after the parameters are updated. The relatively high learning rate and low $\xi$ results in a reasonably fast learning. Since the rewards are given at the end of the episode, though every action in the process highly affects the output, a high discount factor ($\gamma$) was chosen. Further parameters are the subreward weights ($\alpha_1, \alpha_2, \alpha_3$). The constant success reward ($\alpha_1$) was chosen to be high, to strongly distinguish successful and failed attempts, the two other subreward parameters ($\alpha_2, \alpha_3$) were chosen as equal since their role is not in conflict. These hyperparameters were not changed during the training process.

Fig. 3 shows the progress of the training reward through episodes. Even with the changing conditions described in the training environment, the agent managed to learn to control the piston to the desired position in time with acceptable overshoot in approximately 7000 training episodes, and managed to optimize its behavior after 50000 cycles. Although the mean performance of the controller develops fast, this only means that the agent can generalize the majority of the cases and can provide optimal control for them. The control design must focus on all possible cases, and as the performance of the worst 10% of the cases shows that the process needs much more time to develop optimal behavior under all circumstances.

The evolution of the control strategy of the agent is interesting and gives an insight on the learning process. First, the control goal was reached with 3 to 5 valve
actuations and naturally in longer time (see Fig. 4 as example). At this phase the agent usually tried to reach the target without overshooting and with minor control pulses. While at the end the agent finds the exact actuation point, where only one valve actuation is eligible with maximum one additional correction and in remarkably lower time (See Fig. 5).

First the agent developed the safe switching, which came together with the phenomenon, that the overshoot behavior was optimized before the time constraints. This is mostly because the maximal switching time of 80 ms is high enough, and gives more freedom for the agent than the much more rigorous overshoot limit.

Fig. 6 presents the summarization of the training. An evaluation phase consisting of 100 episodes were executed after every 1000 training episodes, to get information about agent performance. The evaluation shows that the average control time dropped from 45 ms to around 25 ms, while the average overshoot value dropped from 0.15 mm to around 0.05 mm.

7. CONCLUSIONS

The paper proposed a neural network based control for a electropneumatic gearbox actuator, where the control strategy and behavior is developed by a reinforcement learning algorithm. The results showed, that the proposed solution finds optimal strategies for the quality requirements of the problem.
The input observation state holds every measurable parameters (pressure and temperature) for all chambers, and also the position and velocity values. Though in real application, the measurement of the parameters raises system costs, therefore the possibilities to eliminate some of these measurements need to be elaborated.

REFERENCES

Chiang, C.J. and Chen, Y.C. (2017). Neural network fuzzy sliding mode control of pneumatic muscle actuators. Engineering Applications of Artificial Intelligence, 65(Supplement C), 68 – 86. doi: https://doi.org/10.1016/j.engappai.2017.06.021.

Doumit, M.D. and Pardoel, S. (2017). Dynamic contraction behaviour of pneumatic artificial muscle. Mechanical Systems and Signal Processing, 91(Supplement C), 93 – 110. doi: https://doi.org/10.1016/j.ymssp.2017.01.001.

Hoşovský, A., Pitel, J., Žídek, K., Tóthová, M., Sárosi, J., and Cveticanin, L. (2016). Dynamic characterization and simulation of two-link soft robot arm with pneumatic muscles. Mechanism and Machine Theory, 103(Supplement C), 98 – 116. doi: https://doi.org/10.1016/j.mechmachtheory.2016.04.013.

Karthikeyan, P., Chaitanya, C.S., Raju, N.J., and Subramanian, S.C. (2011). Modelling an electropneumatic brake system for commercial vehicles. IET Electrical Systems in Transportation, 1(1), 41–48. doi:10.1049/iet-est.2010.0022.

Liu, Y.T., Kung, T.T., Chang, K.M., and Chen, S.Y. (2013). Observer-based adaptive sliding mode control for pneumatic servo system. Precision Engineering, 37(3), 522 – 530. doi: https://doi.org/10.1016/j.precisioneng.2012.12.003.

Mehmood, A., Laghrhouche, S., and Bagdouri, M.E. (2011). Modeling identification and simulation of pneumatic actuator for vgt system. Sensors and Actuators A: Physical, 165(2), 367 – 378. doi: https://doi.org/10.1016/j.sna.2010.11.006.

Nuchkrua, T. and Leephakpreeda, T. (2013). Fuzzy self-tuning pid control of hydrogen-driven pneumatic artificial muscle actuator. Journal of Bionic Engineering, 10(3), 329 – 340. doi:https://doi.org/10.1016/S1672-6529(13)60228-0.

Oliver-Salazar, M., Szewodowicz-Wasik, D., Blanco-Ortega, A., Aguilar-Acevedo, F., and Ruiz-González, R. (2017). Characterization of pneumatic muscles and their use for the position control of a mechatronic finger. Mechatronics, 42(Supplement C), 25 – 40. doi: https://doi.org/10.1016/j.mechatronics.2016.12.006.

Palomares, E., Nieto, A., Morales, A., Chicharro, J., and Pintado, P. (2017). Dynamic behaviour of pneumatic linear actuators. Mechatronics, 45(Supplement C), 37 – 48. doi: https://doi.org/10.1016/j.mechatronics.2017.05.007.

Peters, J. and Schaal, S. (2006). Policy gradient methods for robotics. In 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2219–2225. doi: 10.1109/IROS.2006.282564.

Saleem, A., Taha, B., Tutunji, T., and Al-Qaisia, A. (2015). Identification and cascade control of servo-pneumatic system using particle swarm optimization. Simulation Modelling Practice and Theory, 52(Supplement C), 164 – 179. doi: https://doi.org/10.1016/j.simpat.2015.01.007.

Saravanakumar, D., Mohan, B., and Muthuramanalingam, T. (2017). A review on recent research trends in servo pneumatic positioning systems. Precision Engineering, 49(Supplement C), 481 – 492. doi: https://doi.org/10.1016/j.precisioneng.2017.01.014.

Son, N.N., Kien, C.V., and Anh, H.P.H. (2017). A novel adaptive feed-forward-pid controller of a scara parallel robot using pneumatic artificial muscle actuator based on neural network and modified differential evolution algorithm. Robotics and Autonomous Systems, 96(Supplement C), 65 – 80. doi: https://doi.org/10.1016/j.robot.2017.06.012.

Sutton, R.S., McAllester, D., Singh, S., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. Advances in Neural Information Processing Systems, 12, 1057–1063.

Szimandi, B. and Németh, H. (2009). Closed loop control of electro-pneumatic gearbox actuator. In 2009 European Control Conference (ECC), 2554–2559.

Szimandi, B. and Németh, H. (2014). Robust servo control design for an electro-pneumatic clutch system using the H∞ method. In 2014 IEEE/ASME 10th International Conference on Mechatronic and Embedded Systems and Applications (MESA), 1–6. doi:10.1109/MESA.2014.6935526.

Szimandi, B. and Németh, H. (2013a). Dynamic hybrid model of an electro-pneumatic clutch system. Mechatronics, 23(1), 21 – 36. doi: https://doi.org/10.1016/j.mechatronics.2012.10.006.

Szimandi, B. and Németh, H. (2013b). Sliding mode position control of an electro-pneumatic clutch system. IFAC Proceedings Volumes, 46(2), 707 – 712. doi:https://doi.org/10.3182/20130204-3-FR-2033.00019. 5th IFAC Symposium on System Structure and Control.

van Hasselt, H., Guez, A., and Silver, D. (2015). Deep reinforcement learning with double q-learning. CoRR, abs/1509.06461. URL http://arxiv.org/abs/1509.06461.

Wang, Z., de Freitas, N., and Lancot, M. (2015). Dueling network architectures for deep reinforcement learning. CoRR, abs/1511.06581. URL http://arxiv.org/abs/1511.06581.

Williams, R.J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning, 8, 229–256.

XiaoJun, L., ChengRui, Z., HongBin, L., and XinLiang, W. (2006). Electronic pneumatic clutch control of the heavy truck based on neural network pid. In 2006 IEEE International Conference on Vehicular Electronics and Safety, 232–235. doi:10.1109/ICVES.2006.371589.

Ádám Szaboló, Bécsi, T., Gáspár, P., and Aradi, S. (2018). Control oriented modeling of an electro-pneumatic gearbox actuator. In 17th annual European Control Conference.