Acta Astronautica 219 (2024) 128–137

Contents lists available at ScienceDirect

Acta Astronautica

journal homepage: www.elsevier.com/locate/actaastro

Research paper

Experimental and Simulative Evaluation of a Reinforcement Learning Based Cold Gas Thrust Chamber Pressure Controller

Till Hörger *, Lukas Werling, Kai Dresia, Günther Waxenegger-Wilfing, Stefan Schlechtriem

German Aerospace Center (DLR), Institute of Space Propulsion, Im Langen Grund, Lampoldshausen, 74239, Germany

ARTICLE INFO

Keywords:
Reinforcement learning
Rocket engine control
Intelligent control
Thrust chamber pressure controller
Sim-to-real transfer

ABSTRACT

At DLR neural networks, as potential future controller for rocket engines, are studied. A neural network-based chamber pressure controller for a simplified cold gas thruster is presented and analyzed in simulation and experiment. The goal of the controller is twofold: It can track a trajectory with different changes of setpoints and it allows to set and control a wide variety of steady state chamber pressures. The neural network gets feeding line pressure measurement data as input and calculates valve positions as output values. The training phase of the controller is done with a reinforcement learning algorithm in an EcosimPro/ESPSS simulation, that is validated with data from the corresponding experimental set up. To increase the robustness and to allow a transfer from the simulation directly to the test facility domain randomization is applied. The controller is evaluated in simulations and experiment. It was found that – in the range of physically possible operation points – the controller achieves a constantly high reward which corresponds to a low error and a good control performance. In the simulation the controller was able to adjust all required set points with a steady state error of less than 0.1 bar while retaining a small overshoot and an optimal settling time. It is found that the controller is also able to regulate all desired set points in the real experiment. A reference trajectory with different steps, linear and sinus changes in target pressure is tested in simulation and experiment. The controller was in both cases able to successfully follow the given trajectory.

1. Introduction and motivation

Rocket engine thrust control is essential for many applications that are currently of high interest. Particularly for propulsive landing of launchers as well as planetary or lunar landing systems. In Europe various projects are developing vertical take of and landing technology [1–4]. This applications have high demands on the engine control system. Precise thrust control, restart capabilities and deep throttling are needed to assure soft landing. Moreover, the potential reuse of engines requires optimal transients e.g. with low thermal loads to avoid damage. Adaptive control systems are beneficial to handle system changes due to degradation of components over their live-time.

Literature about control methods for rocket engines is limited, an overview on publicly available reports is given in [5]. The general goal of rocket engine controllers is to regulate engine thrust at an optimal efficiency. Thrust is mainly influenced by the combustion chamber pressure and hence the total injected mass flow. The mixture ratio control influences the specific impulse and therefor the efficiency of thrust generation per propellant mass [5]. In parallel to thrust and mixture ratio control, several temperature and pressure constraints are in place in order to protect the hardware from damage. The controlled-variables are coupled in a nonlinear way resulting in a non linear multiple-input–multiple-output problem. Actuators in a rocket engine are usually adjustable control valves that regulate the flow rate of the propellant. Pressure, temperature, turbopump speed and mass flow sensors are used as feedback.

Closed loop control of transients in rocket engines, like start-up of the engine, is challenging, due to the designated non-linear behavior. Classic control approaches like PI and PID are successfully used for example in the space shuttle main engine for control of steady state operating points or minor set point changes [6]. In [7,8] throttling from 21% to 109% of a rocket engine controlled by a PI controller is reported. The PI parameters were tuned to step and frequency response test results. More recently, model-based methods like linear–quadratic regulators (LQR), model predictive control (MPC), H-infinity control as well as the inclusion of machine learning techniques are successfully investigated for rocket engine closed loop transient control [5,9,10].

* Corresponding author.
E-mail addresses: till.hoerger@dlr.de (T. Hörger), lukas.werling@dlr.de (L. Werling), kai.dresia@dlr.de (K. Dresia), guenther.waxenegger@dlr.de (G. Waxenegger-Wilfing), stefan.schlechtriem@dlr.de (S. Schlechtriem).

https://doi.org/10.1016/j.actaastro.2024.02.039
Received 19 October 2023; Received in revised form 17 January 2024; Accepted 26 February 2024
Available online 2 March 2024

The benefits of an intelligent engine control system used for control reconfiguration and condition monitoring, have already been investigated in the space shuttle era [11,12]. In 2019 German Aerospace Center (DLR) started investigating machine learning methods for supporting the design and operation of liquid rocket engines and thrusters [13,14]. Control systems including machine learning methods promise several advantages for rocket propulsion systems: Side conditions can be included easily. For example the fatigue life of engine components at a given thrust level could be expanded with an appropriate damage model included in the control policy. Wall temperatures at specific combustion pressures can be kept under a certain level. The efficiency at a particular operation point or during throttling of an engine can be increased. The controller can also deal with pre-trained failure models as well as react intelligently to changes in the system due to anomalies and possibly reconfigure the system.

One approach to design closed-loop optimal control laws using machine learning is deep reinforcement learning (DRL) [15]. DRL is applied successfully to complex control tasks like robot control [16,17], drone flight control [18] and autonomous driving [19]. One advantage of DRL is that the controller can be derived directly from nonlinear high sophisticated simulation models, in which gradient information is potentially not accessible to the user [10]. No derivation of state–space models, model order reduction or linearization is needed as it is required for PID, LQR and MPC controllers. Furthermore, once the controller is trained, the response time is very short [20]. The utilization of deep neural networks as controller offers the possibility to design highly non-linear multiple-input multiple-output controllers, that incorporate all kind of side conditions while delivering a optimal control performance. Nevertheless, since the stability of these algorithms is not mathematically proven until now, the stability and robustness of the controller has to be carefully evaluated before use. Further challenges in using reinforcement learning are the high amount of training data and computational power needed to converge, especially for complex tasks [21]. Transferring controllers, trained in simulation only, to the real application can be challenging as the simulation never perfectly fits the real system [22]. Furthermore it is hard to assure safety and robustness of the controller. One possibility to evaluate the performance and stability of the control algorithm is testing in multiple simulations and experiments.

In this work, a simplified cold gas system is utilized to demonstrate the basic functionality of an artificial neural network trained by a reinforcement learning algorithm as rocket engine controller. This set up allows rapid, cheep and safe testing while offering the opportunity to conduct real world experiments. Goal of the work is to show the principal functionality of the presented method in an academic example, rather than outperforming other control methods in this task.

A simulation model of the thruster was set up in EcosimPro/ESPSS and validated with experimental data. Further, a Python interface allows the use of state of the art deep reinforcement learning algorithms for training and at the test facility. Based on the simulation model a thrust chamber pressure controller for the named cold gas system is trained. The goal of the controller is to set different thrust chamber pressures at different feeding line pressures as well as to follow pressure trajectories. The steady state control deviation should be less than 0.1 bar. Further overshot of less than 0.5 bar and a optimal settling time is required. The paper discusses the control performance achieved by the neural network in simulation as well as at the test facility. The second chapter gives background information about the used machine leaning methods. In section three the experimental set up and the implementation of the controller is described. Chapter four discusses the simulative evaluation of the controller performance while in chapter six the experimental results are shown.

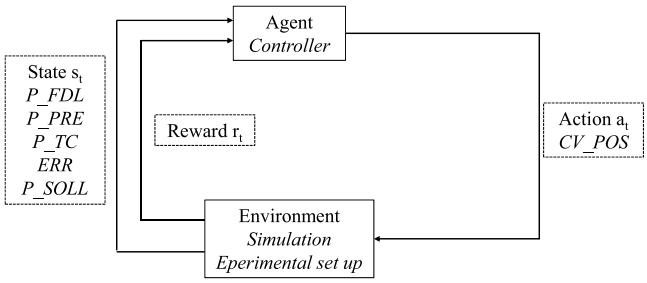


Fig. 1. Reinforcement learning schematic based on [24].

2. Method and theoretical background

2.1. Reinforcement learning

Reinforcement Learning is a sub-field of machine learning (ML). In machine learning a large amount of training data is used to find solutions for complex problems. Three categories (supervised, unsupervised and reinforcement learning) can be distinguished in ML, depending on the quality of information give in the problem [23]. Supervised learning can be applied if the training dataset contains the input and the desired output data. This can be useful to find corresponding mapping rules between input and output, that can be used for example for image classification. Whereas in unsupervised learning, the target output data is unknown. The goal is to discover structures, similarities or hidden patterns in the input. Such algorithms can be used e.g. for cluster analysis or to find similarities in data.

Reinforcement learning (RL) is different from supervised and unsupervised learning because no predefined training datasets is needed. A so-called agent learns self-employed through interaction with a simulation or real-world data. RL operates in discrete time steps. The underlying principle of RL is visualized in Fig. 1: An agent interacts in discrete time steps t with the environment. The environment has defined states s_t. In every time step the agent gets information about the system state from the environment. Environment information known to the agent is called observation space. The state is rated with a user defined scalar value called reward r_t. The goal is to maximize the reward in order to achieve the desired behavior of the controller. Hence experience from interaction with the environment is used to learn an optimal control strategy.

Reinforcement learning methods are bounded to problems that are stated as Markov decision process (MDP) [25]. MDP's are a general framework for decision making processes where the probability to get in the next state s' only depends on s and the current action a and not to steps prior to s (Markov-Property) [24]. The rule, which defines the action is called policy π. A trajectory τ is defined as a sequence of states and actions.

$$\tau = (s_0, a_0, s_1, a_1, \dots, s_n, a_n) \tag{1}$$

The overall goal is to find a decision rule for choosing actions (policy) that maximizes the expected cumulative reward of a trajectory. In deep RL the policy is represented by a artificial neural network that acts as function approximation of the policy. It maps a state of the system to an action a_t. The agent develops a strategy to maximize the expected cumulative reward $G(\tau) = \mathbb{E}\{\Sigma r_t\}$ that it gets. The cumulative reward $G(\tau)$ is the sum of all single reward values in one episode, where the episode is one sequence of simulation following the policy π. Through interaction with the environment the agent gets information, encoded in the reward, which action is best in which situation. To avoid the expected cumulative reward for long episodes growing to infinity, a discount factor $0 \leq \gamma \leq 1$ is introduced (See Eq. (2)).

$$G(\tau) = \mathbb{E}\left\{\sum_{t=0}^{\infty} \gamma^t r(s_t)\right\} \tag{2}$$

Table 1
Key terms in reinforcement learning [15,20].

Expression	Description
Agent	The algorithm or controller. It optimizes the policy.
Environment	System that interacts with the agent. In this case the rocket thruster
State	Physical state of the system
Reward	Scalar value that rates the state of the system. Calculated by a user defined reward function
Action	Control output of the agent. Sent to the environment
Observation space	Variables that serve as input values and system description for the agent
Policy	Decision rule of the agent. A function that maps states to actions. It defines how the agent reacts to different system states. In deep RL the policy is an artificial neural network. During training the policy is optimized through the agent to find actions delivering the maximum reward

The value of γ defines how much future rewards come into account for the current action because γ is raised to the power of t. This process can be seen as a trial and error method with integrated feedback in the form of reward. Table 1 gives an overview on important concepts in RL.

In literature a wide variety of RL-algorithms is described. It is differentiated between model-based and model-free RL. In the following it is focused on model-free methods, as these were used in this work. Different learning paradigms can be distinguished, depending on what function exactly is optimized: In (Deep) Q-learning [26] the so called optimal action-value function is learned. Policy optimization algorithms directly optimize a parameterized policy e.g. by policy gradient methods [27]. Actor critic algorithms try to combine the strengths of Q-learning and policy gradients [28,29]. On-policy algorithms need completely new training samples after each policy update, while off-policy algorithms can learn from past samples using replay buffers. In this study, the off-policy Soft-Actor–Critic (SAC) algorithm [30], implemented in the Ray RLlib framework is used [31]. SAC, released in 2018, can handle continuous state and action spaces. Compared to on-policy algorithms like the widely used PPO [32] or A3C [33], SAC is characterized by a comparatively high sample efficiency which may lead to faster training success. Furthermore, in comparison to other off-policy algorithms like DDPG [29] or TD3 [28], the training process is stable and less hyper parameter tuning is needed. Beside the cumulative reward, the SAC algorithm also maximizes the policy-entropy $\mathcal{H}$, as a measure of how random the agent acts. This can be seen in the following equation for calculating the optimal policy π^* taken from [30]:

$$\pi^* = arg\ \max_{\pi} \sum_{t} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \left[r(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t)) \right] \tag{3}$$

Besides the first term in the expectation representing the reward $r(s_t,a_t)$, a second part $\alpha\mathcal{H}(\pi(\cdot|s_t))$ is added. α is a hyper-parameter controlling the influence of the policy-entropy $\mathcal{H}$ on the overall maximization. In this way the task is completed successfully while acting as randomly as possible [30]. The policy is therefore encouraged to exploration while being robust concerning model and estimation errors [30]. However, several million time steps of training can be required to train a controller with reinforcement learning. Therefore, it is beneficial that SAC can also be trained in a distributed manner to take advantage of multi-CPU machines, further reducing the training time.

2.2. Domain randomization

In the first phase of training, the agent randomly explores the environment. Hence, it will possibly happen, that the agent enters states that are dangerous for the system. Therefore the training has to take place in a simulation environment. One main challenge when using reinforcement learning based controllers is the transfer from the simulation based training environment to the real world application. This is also called Sim-to-Real transfer [34]. Even carefully tuned simulations will always contain small deviations from the real system. The RL algorithm learns to make use of these simulation inaccuracies and the application in the real world may fail due to unforeseen input values. This is problematic for the application of RL based controllers, as they react very sensitively to changes in the simulation environment [35].

Domain randomization can help to overcome the Sim-to-Real gap [35–37]. The idea behind Domain Randomization (DR) is to randomly change crucial variables like friction, delays, temperatures or pressure loss in the simulation environment during the training [38]. In this way policies with a wider area of validity can be generated, as the reality becomes a subset of the randomizes simulation environments [39].

In order to apply the DR method to a specific use case a set of simulation parameters N_{rand} that will be changed randomly during training has to be defined. Each parameter ξ^i is bounded on a randomization interval $[\xi^i_{low}, \xi^i_{high}]$ resulting in a randomization space $\Xi \subset \mathbb{R}^{N_{rand}}$.

The choice of the randomization space Ξ depends on the problem and has a high influence on the training and the resulting policy [39]. Ξ has to be general enough to cover all possible discrepancies between simulation and the real application. Anyway it has to be as small as possible, as a huge randomization space produces more conservative policies which deliver not optimal solutions for the specific environment [40]. The choice of the randomized parameters has to avoid nonphysical solutions like for example negative pressures. The correct assortment and range of the randomized parameters demands high domain knowledge [40]. The training on a randomized environment takes more steps and consequently more computing time, as the variance of the environment is higher [39]. Also the reward achieved on the baseline simulation environment is tending to be inferior to an agent trained solely on the baseline simulation [39].

3. Experimental set-up

A suitable application for a reinforcement learning based controller is for example a 22 N thruster fueled with gaseous nitrous oxide and ethane. Such a system can be constructed either as premixed monopropellant- [41,42] or bipropellant system. For the control task, the bipropellant system offers more degrees of freedom. Because of the high vapor pressure of fuel and oxidizer it is possible to operate the system in self pressurized mode. The propellants are stored as liquefied gases with two phases in separate tanks. In a self-pressurized system the tank pressure decreases over burn-time, as the propellant evaporates, and the enthalpy of vaporization cools the remaining propellant. This is especially the case, when the thruster is fed from the gas phase. Therefore, to achieve a constant thrust level, an active control system is needed. Nitrous oxide/ethane offer a green alternative to the widely used satellite propellant monomethylhydrazine and dinitrogen tetroxide [43]. The latter are attempted to be replace due to their toxic, carcinogenic and environmentally harmful properties. At DLR several encouraging green propellants are under investigation [44–46]. Nitrous oxide/ethane are comparably cheap, widely accessible and offer a high $I_{sp} \approx 300$ s [43].

3.1. Cold gas test set-up

Since the DRL-control method was never before experimentally tested in the field of combustion chamber pressure control, before conducting hot fire tests with nitrous oxide/ethane, a reduced nitrogen cold gas system is used for the first tests. Nitrogen is chosen as simulant for two reasons: Firstly, in comparison to nitrous oxide less safety regulation and concern due to oxidizing atmosphere has to be applied. Secondly, releasing large amounts of un-burned nitrous oxide in the

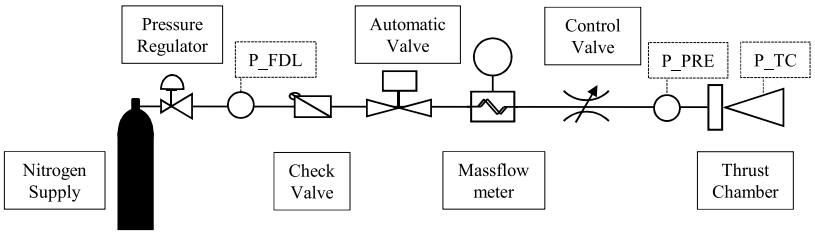


Fig. 2. P&ID of the test set up.

Table 2
Input and output variables of the controller.

Input (Observation space)	Output (Action space)	Controlled variable
P_FDL	dCV_POS/dt	P_TC
P_PRE		
P_TC		
ERR		
P_SOLL		
CV_POS		

atmosphere contributes to climate change [47], while with nitrogen for cold gas experiments less climate impact accounts. The P&ID of the test set up can be seen in Fig. 2. Nitrogen is feed by 200 bar supply pressure. The nitrogen pressure is reduced to a maximum of 100 bar by a PID controlled automatic pressure regulator of type Tescom ER5000. The pressure controller allows to set different feeding line pressures. An automatic main valve is used to start and stop the gas flow. After the main valve a Rheonik coriolis mass flow meter is installed, followed by a proportional control valve of type m-tech MPG 03 PR. The nitrogen is then expanded through a chamber, initially designed for use with nitrous oxide and ethane. The goal of the controller is to set the chamber pressure to a desired value by changing the position of the control valve for varying feeding line pressures. All testing is conducted at DLR Lampoldshausen M11.5 test position [46,48].

Three pressure sensors, P_FDL, P_PRE and P_TC (see Fig. 2), are used as input variables for the controller. Further, the current position of the control valve CV_POS, the desired chamber pressure P_SOLL and the error ERR (difference between the desired pressure and the measured P_TC) are used as input to the neural network. Based on these values the network calculates the action value. The output is defined as the relative change of the position of the control valve dCV_POS/dt. This value is sent to the test facility and changes the position of the control valve. The change in position of the control valve is limited to 5% in each time step, due to low movement speed of the valve. The control frequency is 10 Hz. Since the control valves used for the experiments, have a dead time of about 300 ms and additionally an opening time of about 2 s to fully open, 10 Hz are sufficient for this setup. In a later set up, faster valves will be used. The control frequency is then planned to be higher. For comparison, the engine controller of the space shuttle main engine operated at 50 Hz [6]. Table 2 gives an overview about the input and output parameters of the controller. The test facility is controlled and operated by a python based user interface, that allows the simple inclusion of neural networks in the workflow [20].

3.2. Training of the controller

The controller is essentially an artificial neural network (NN). The NN is used as function approximation to map pressure measurement data to valve positions. To act successfully as controller the internal weights of the NN have to be adapted via training. Essential for good results is the choice of the reward function. This function should return high values, if the controller is close to the desired thrust chamber pressure and low values if it is far away from the set point. Furthermore unnecessary valve movement and oscillations should be punished. The gradient of the reward function with respect to the pressure very close to the set point should be high enough to allow finding the terminal position. Designing a good reward function is a key challenge when applying reinforcement learning to a specific problem. For this application the approach in the following algorithm is chosen:

$r(t) = Rew1 + Rew2 + Rew3$

if $ERR < 2\,\text{bar}$ **then**
  $Rew1 = -\sqrt{ERR}$
else
  $Rew1 = -10$
end if

$Rew2 = -|valvespeed|$

if more than 3 time steps $ERR \leq 0.1\,\text{bar}$ **then**
  $rew3 = 10$
else
  $Rew3 = 0$
end if

The reward function $r(t)$ is subdivided in three parts $Rew1$, $Rew2$, $Rew3$, each of which evaluates a different aspect of the control goal. $Rew1 = -10$ if the deviation from the set point is 2 bar or higher. Apart from that $Rew1$ is equal to minus the square root of the control deviation. $Rew1$ is mainly important at the beginning of the training phase, because large deviations are punished with a large negative value while the square root function offers a steeper gradient close the target pressure than a linear function. $Rew2$ penalizes valve motion and especially fast valve motion. This helps to avoid oscillations around the target pressure. However this may also increase the settling time. Finally, $Rew3$ supports finding and holding the target pressure. $Rew3 = 10$ only if the control deviation was less than 0.1 bar for more than three time steps in succession. The maximum possible reward in one time step is therefor $r(t) = 10$ and the undiscounted maximum cumulative reward of one episode is $G = 10 \cdot (n - 3)$ with n time steps in one episode.

The training of the controller described in this paper is done with the off-policy soft actor critic (SAC) algorithm [30] in the RAY Rllib implementation [31] (see Table 3). 6 CPUs are used for training in parallel. The neural network representing the policy has two hidden layers with 256 neurons each. The Activation function is ReLu. All hyper-parameters of the algorithm are implemented in the same way as in the standard implementation [30,31]

During the training or exploration phase the controller intentionally may enter system states, that are possibly dangerous for the system. This is fined with a low reward and is part of a nominal training process where also low reward areas have to be observed by the algorithm. Therefore, to save cost and to avoid damage of the test facility or test specimen the training process takes place in a simulation environment.

EcosimPro ESPSS [49] is used as simulator. For training of the controller, a simulation model of the cold gas test setup up described in Section 3.1 is created. The simulation model is validated with data

Table 3
Relevant variables for the training of cold gas thrust chamber pressure controller.

Variable	Value or range
P_FDL	$\{P\_FDL \in \mathbb{N} \mid P\_FDL \geq 60 \wedge P\_FDL \leq 90\}$ bar
P_SOLL	$(\{P\_SOLL \in \mathbb{N} \mid P\_SOLL \geq 50 \wedge P\_SOLL \leq P\_FDL\})/10$ bar
T_FDL	$\{T\_FDL \in \mathbb{N} \mid T\_FDL \geq 268 \wedge T\_FDL \leq 298\}$ K
Sensor noise	Gaussian distribution with μ = Sensor value and $\sigma = 1$. The random number is multiplied with 0.5% of the sensor value
Simulated pipe length	Randomized with normal distribution $[0.6, 1.4]$
Valve Speed	Randomized with normal distribution $[0.1, 0.5]$
Algorithm	SAC
Implementation	RAY Rllib 3.0.0.dev
Number CPU	6
Framework	Torch
Neural net	Activation function: ReLu Two hidden layers of size 256

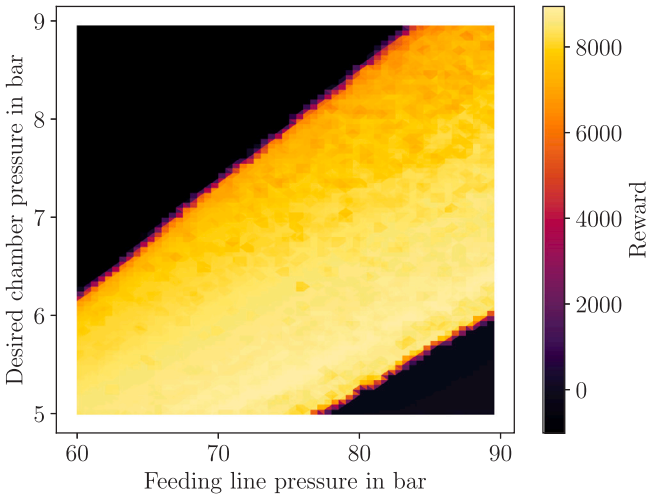


Fig. 3. Reward achieved in simulation for different set points. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

from the experimental setup. The measured thrust chamber pressure at corresponding mass flow rates can be reproduced in simulation with a maximum error of 3% for different feeding line pressures and valve positions.

In order to train a robust control policy that gives useful results also in the experiment, domain randomization as described in Section 2.2 is used. The intervals in which the parameters are randomized have to be chosen wisely. Large randomization intervals massively increases the training effort while narrowly defined randomization intervals may exclude the areas relevant for the real application. For the cold gas set up described in this paper, it was found that the speed of the control valves as well as calculating correct pressure losses in the system are the most challenging parts of the simulation. Hence valve speed and the length of the feeding lines in the simulation was used as randomization space Ξ. The length of each simulated pipe in the feeding line is multiplied by a factor (interval see Table 3) randomly changing after each episode and therefore varying the pressure loss in the feeding lines. The valve speed is varied after each episode in the interval given in Table 3.

During training one episode comprises 40 s simulation time. To further increase the robustness of the controller the following procedures are implemented: During simulation every 10 s the feeding line pressure P_FDL, feeding line temperature T_FDL as well as the desired thrust chamber pressure P_SOLL is changed randomly in the given limits with a equal distribution (compare Table 3). The change in feeding line input temperature simulates conditions in different seasons. Input values are rounded to integers or one decimal respectively. Every 5 episodes the target pressure is following a constantly changing trajectory with different feeding line pressure respectively. This should enable the controller to set different set points as well as to follow a given trajectory. Sensor noise is added to all variables in the observation space with a Gaussian distribution with standard variation of $\sigma = 1$. The noise is multiplied with 0.5% of the sensor value and added to the latter.

4. Simulative analysis

To analyze the control behavior of the neural network after the training is completed, simulations are conducted. The NN has to control the thrust chamber pressure via interaction with the EcosimPro/ESPSS simulation model. 4800 simulations were conducted to systematically analyze the performance in the training area. Feeding line pressure was increased in 0.5 bar steps from 60 bar to 90 bar. Target pressure was simulated in an interval from 5 bar to 9 bar with steps of 0.05 bar each. Thereby the entire operating envelope of the set up is covered with a fine resolution. The set-up is regularly used for hot gas tests. In order to reach comparable chamber pressures with cold gas, high mass flow rates are needed, resulting in high pressure losses. The chosen range of feeding- and chamber pressures is a compromise of slightly to low chamber pressures and relatively high feeding line pressures in comparison to hot gas operation.

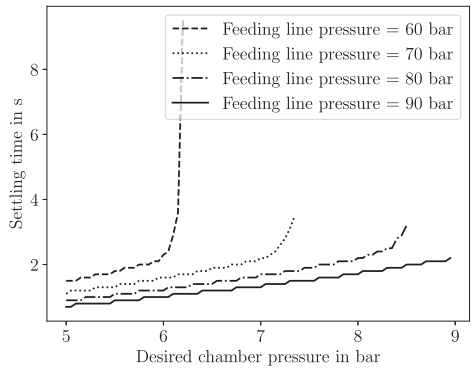
The evaluation simulations were carried out for a run time of 10 s starting always from 10% valve opening. According to the definition of the reward function, the maximum possible reward value for 100 timesteps would be 10 000. As the valve needs up to 2.3 s to open completely, a realistic value for a perfect run is in the order of 8000. The result can be seen in Figs. 3 and 4.

The areas marked with yellow color in Fig. 3 represent set points with a high reward achieved. This means the controller was able to set the chamber pressure to the correct target value. While in the black areas the reward is low and as a consequence the controller failed to set the desired pressure. Both areas on the top left and the lower right where the controller failed are tied to physical limitations of the test set up. For a given feeding line pressure due to pressure losses in the system, only a limited thrust chamber pressure can be achieved at completely opened control valve. This is the case for the dark area in the top left side of Fig. 3. For example at 60 bar feeding line pressure, only 6 bar chamber pressure are possible. Therefor the higher operating points are out of scope. The limit in the lower right in Fig. 3 can be lead back to a limitation of a minimum opening of the control valve to 10%. High feeding line pressures would require control valve positions below 10% to reach low target chamber pressures. The reason for introducing a limit at the lower boundary of the control valve was due to high inaccuracies if the control valve is operated at lower opening levels.

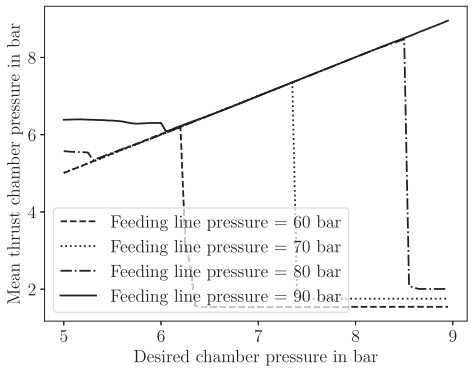
It can be seen that within the physical boundary the controller is able reach a constantly high reward and therefor small deviations at all tested set points in the chosen resolution between 60 bar and 90 bar feeding line pressure and 5 bar and 9 bar chamber pressure are accomplished.

Fig. 4 allows a more detailed analysis of the controller behavior.

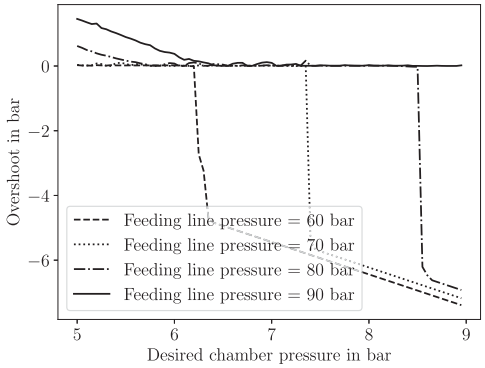
Fig. 4(a) shows the settling time in dependence of the desired chamber pressure at four different feeding line pressures. It can be seen that the settling time rises for higher target pressures, which is a result of the required higher opening of the control valve. Curves representing lower feeding line pressures resulting in higher settling times for the same desired pressure. This is because for the same chamber pressure at a lower feeding line pressure the control valve has to be opened further. At the upper right end of each line, the set point can no longer be



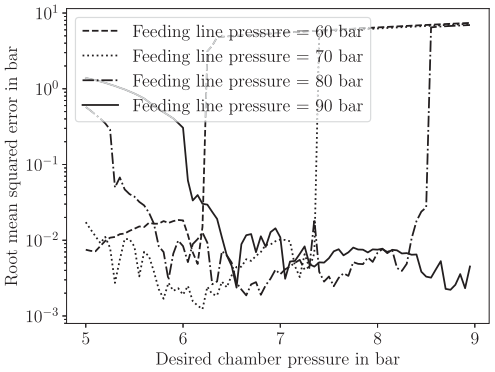
(a) Settling time for different set points



(b) Mean simulated pressure in thrust chamber



(c) Overshoot for different set points



(d) Root mean squared error (RMSE)

Fig. 4. Simulation analysis.

reached du to limited feeding line pressure and the valve being already fully open. Therefore the settling time grows to infinity. The maximum settling time in the controllable area of about 2.4 s corresponds to the opening time of the valve, which is also 2.3 s. So the controller is able to set the target pressure in the minimum possible settling time. The linear progression of the curves in the controllable area is a result of the linear transfer function of the control valve.

Fig. 4(b) represents the mean thrust chamber pressure in dependence of the desire chamber pressure. For the controllable area this is a straight line, which indicates that the target pressure equals the simulated pressure set by the controller. For higher target pressures at a specific point each curve instantly drops to a lower value. The controller closes the control valve to the lower limitation (10%) when it is no longer possible to reach the desired pressure. For higher feeding line pressures, the drop occurs at higher target pressures and results also in a higher remaining pressure in the thrust chamber. This behavior can be problematic. If the desired pressure is too high for the physical limits of the system, the highest possible pressure should be set and the valve should be completely opened. This behavior is probably attributed to the choice of the reward function. Since the achievable pressure is always far away from the desired pressure only negative reward with almost no gradient information is given, independent on the actual pressure. Therefore, for the controller it is beneficial to not move the control valve at all to avoid punishment for valve movement. This should be addressed via changing the reward function in future applications. For higher feeding line pressures (80 and 90 bar), as already described in Fig. 3, the low target pressures cannot be reached, as the minimum opening of 10% results in higher thrust chamber pressures.

Fig. 4(c) represents the overshoot in pressure during the settling time. It can be seen, that for the controllable area nearly no overshoot can be observed. The lines representing 80 bar and 90 bar feeding line pressure have a peak for low desired chamber pressure, which is a result of the inability to reach this pressures given the physical limits of the system.

Fig. 4(d) shows the root mean squared error (RMSE) for the different set points in bar. The RMSE in the controllable area is in the order of 0.01 bar and therefor well within the target accuracy of 0.1 bar defined by the reward function. For areas outside the controllable region the root mean squared error rises.

All simulations shown in Figs. 4 and 3 were run with the same baseline simulation set up. Domain randomization was used in training, but not for evaluation. In order to test the robustness of the controller for changing environmental behavior a Monte Carlo simulation with 5000 sample simulations is implemented. The result is shown in Fig. 5.

The following parameters are changed randomly with equal distribution during the MC simulation in the given ranges in Table 3: Feeding line pipe length, valve speed, feeding line temperature, feeding line pressure P_FDL, desired pressure P_SOLL, sensor noise. This analysis reveals how the controller behaves under changing environment-conditions, e.g. when being transferred to the real test facility. Sensor noise was added between 0,1% and 1% intensity, unlike given in Table 3. For the Mote Carlo simulation only operating points inside the controllable area (where the controller reached a high reward in the baseline simulation, compare Fig. 3) were chosen. The Monte Carlo analysis shows that the vast majority of 4388 simulations lead to a reward of more than 6000. In 1749 of the simulations the reward was higher than 8200. However in some of the simulations the controller failed to achieve the desired chamber pressure, as can be seen by a negative or very low positive reward. In total 331 simulations lead to a negative reward. It was found, that simulations resulting in a negative reward are on the boundary of the controllable area. It is assumed, that with changing for example the pressure loss or temperature a set point which was under baseline conditions inside the controllable area,

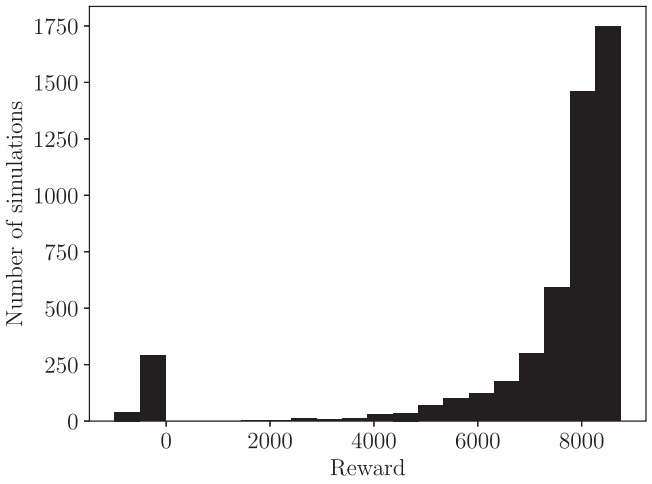


Fig. 5. Result of Monte Carlo simulation.

moved outside and can no longer be controlled by the controller due to physical limits of the system.

As last step, the controller was tested following a predefined trajectory of the chamber pressure at constant feeding line pressure. The trajectory contains different step changes as well as sinus and linear change in the desired chamber pressure. The simulation was conducted with a constant feeding line pressure of 80 bar. The result can be seen in Fig. 6. The target trajectory is marked with dashed line, while the simulated controlled chamber pressure is shown as solid line. The controller was able to follow the trajectory with a root mean squared error of 0.36 bar. The slow settling time for the larger set point changes is a result of the slow valve speed. Nearly no overshoot can be observed.

5. Experimental analysis

In total 142 steady state experiments are used for the experimental investigation of the controller performance. Additionally one test following a trajectory is presented. Experiments are conducted at M11.5 test facility in Lampoldshausen [46,48].

A characteristic pressure plot of one steady state experiment is given in Fig. 7. Data acquisition starts at $t = 0$ with the experiment time t. 4.3 s after begin, the automatic valve is opened, and nitrogen starts flushing the thrust chamber. At the beginning of each experiment the control valve is set to 10% open. After opening of the automatic valve there is a 5 s wait to establish steady flow conditions. In this way equal starting conditions for every experiment are guaranteed. When the flow is established 10 s after the beginning, the neural network takes control and changes the chamber pressure to the desired value, in this case 8 bar. Due to limited valve speed in this experiment it takes 2.2 s to reach the set point. No overshoot can be observed. After 21 s the experiment is over and the control valve is opened completely, to release remaining nitrogen in the feeding lines. This is the reason for the pressure peak at the end of the experiment before the automatic valve is closed after 22 s.

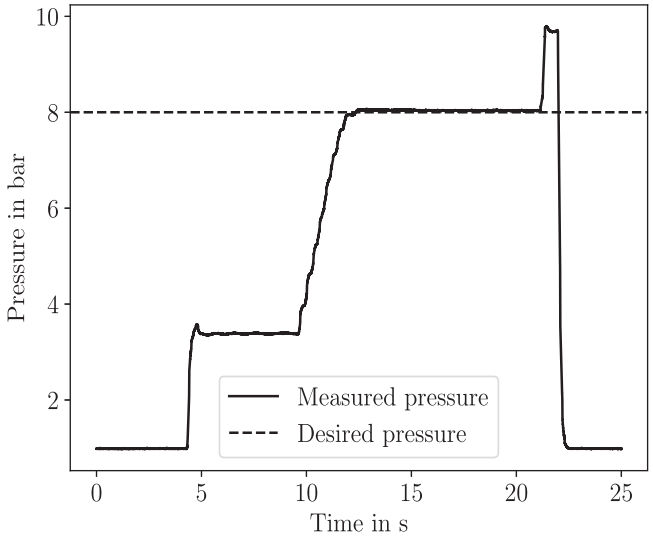


Fig. 7. Thrust chamber pressure plot for a experiment with 80 bar feeding line pressure and a target pressure of 8 bar.

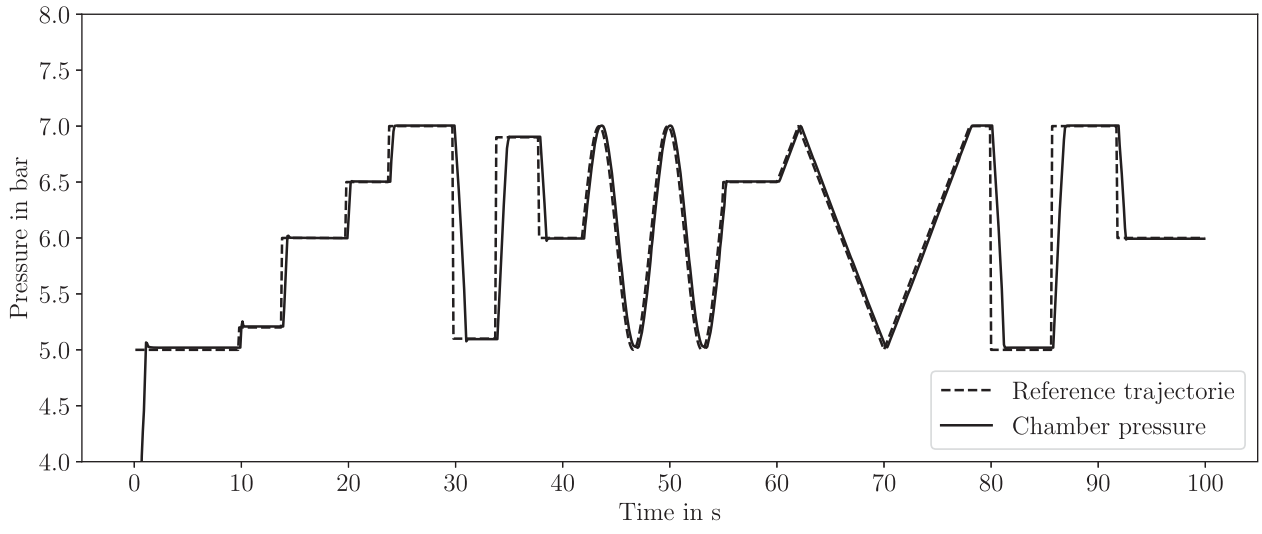
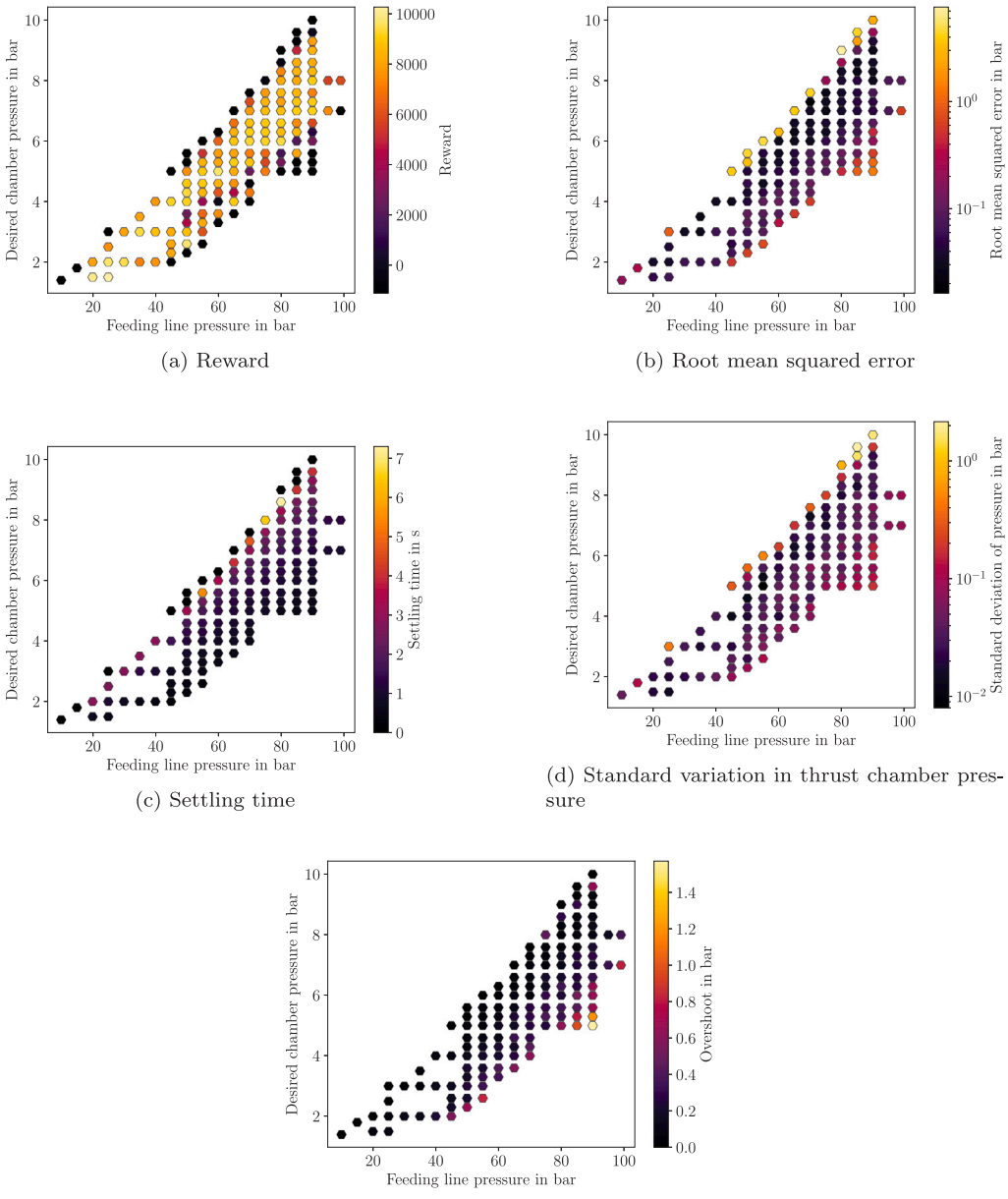


Fig. 6. Simulated chamber pressure following a predefined trajectory.



(a) Reward

(b) Root mean squared error

(c) Settling time

(d) Standard variation in thrust chamber pressure

(e) Overshoot in thrust chamber pressure

Fig. 8. Experimental results.

142 of this test similar to the one described above have been conducted with different feeding line pressures and different target chamber pressures. The results are presented in Fig. 8.

In this plots every hexagon represents one experiment. On the horizontal axis the feeding line pressure and on the vertical axis the desired chamber pressure is shown. Therefore, a hexagon at the same spot always represents the same experiment. The color of the hexagon labels the magnitude of the evaluated variable for the respective experiment. All variables are calculated for the time interval the neural network takes over control ($t = 10$ s) until the end of the experiment ($t = 22$ s). Fig. 8(a) indicates the calculated reward achieved in the experiments. The result is similar to the simulative evaluation (see Fig. 3). Inside the controllable area high reward values are obtained. Control limitation is given for low and high target pressures for minimum and maximum vale opening respectively. The limits can be seen by the black hexagons marking experiments resulting in a low reward. The physical limits of the system in the experiment are the same as in the simulation. In this cases the target pressure could not be set. Once a set point could not be reached the feeding line pressure was changed. As the experimental results were promising, the controller was also tested outside the training area. The controller was trained in simulation for feeding line pressures between 60 bar and 90 bar and target pressures from 5 bar to 9 bar. However also for feeding line pressures as low as 20 bar and up to 100 bar successful control results were observed. For lower feeding line pressures the possible target pressure is also reduced. Obviously, the neural network extrapolated the successful control policy also for other operating envelopes. There is no forecast about the control-quality outside the operating points used in training. Therefor this result has to be evaluated carefully before use in operation.

Fig. 8(b) shows the root mean squared error of the chamber pressure for each experiment in bar. Interestingly a slight decrease of RMSE for higher target pressures can be seen. Which leads to the assumption that the stationary control deviation for higher set points is lower. However,

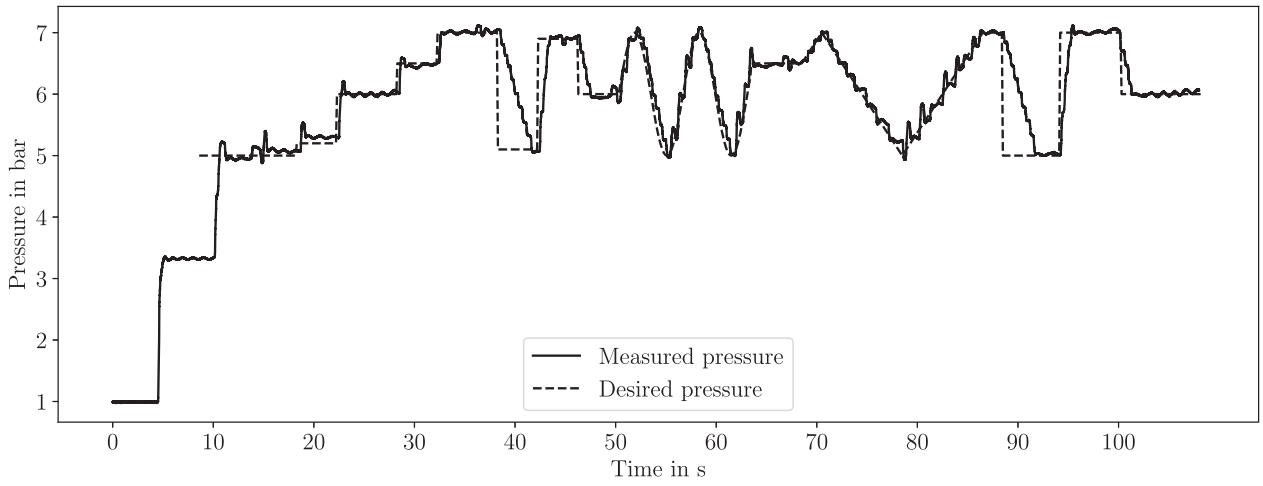


Fig. 9. Chamber pressure following a predefined trajectory.

all experiments within the controllable area resulted in a RMSE not higher than 0.2 bar.

Fig. 8(c) shows the settling time. It is defined as time until only deviations of less than 10% of the desired pressure arise. For higher target pressures the settling time grows, as the valve needs to be opened further. Similar to the simulative results the settling time in the controllable areas is as low as possible with the given speed of the control valves. For set points outside the controllable area no settling time can be calculated. They are marked with settling time zero. Some experiments at the border of the controllable area show larger settling times.

In Fig. 8(d) the standard deviation of the chamber pressure during the stationary time in the experiments is shown. Standard deviation is a measure for deviations and oscillations during the steady state operation. With less than 0.1 bar in the controllable area fluctuations around the target pressure are low.

In Fig. 8(e) the color-map represents the overshoot that was observed during the settling phase of the chamber pressure. The observed overshoot for lower target pressures is higher while for high desired pressures no overshoot can be observed.

Concluding Fig. 8 it can be seen that for the relevant target area between 60 bar and 90 bar the controller is successful in changing and holding the chamber pressure to the required set point. While acting with fast settling times, low overshoot, low fluctuations and low derivations from the set point can be seen.

In order to test not only steady state performance of the neural network based controller, the control performance is tested following the given reference trajectory. Fig. 9 shows the measured thrust chamber pressure, corresponding to the simulation shown in Fig. 6.

It can be seen that the controller was able to follow the trajectory. However, in comparison to the steady state experiments and to the simulative evaluation of the trajectory, more fluctuations and higher overshoot during the change of operating points can be seen. The long settling time for larger set point changes is a result of the slow valves used for the experiments. The root mean squared error is calculated to 1.15 bar. This comparably high value is partly a result of the slow valves resulting in high absolute errors for long time during the long settling time. However the root mean squared error of the full trajectory in the experiment is higher than in the simulation. Observing the step response at 40 s in Fig. 9 it can be seen, that the time needed for pressure reduction is longer than for pressure increase. This effect is not present in the associated simulation in Fig. 6. This indicates rather a problem with closing the valve or an other issue with the experimental set up than with the controller itself, because it is not visible in the respective simulation. However, the exact reason for this behavior is indeterminable. The controller performance in experiment following the trajectory is, despite using domain randomization, worse than in the simulation.

6. Conclusion and outlook

A neural network trained with the reinforcement learning algorithm SAC [30] is used as chamber pressure controller for a nitrogen cold gas thruster. The controller was analyzed in simulation and experiment. Within the physical limits of the system the controller showed satisfying behavior, meeting the requirements of less than 0.1 bar steady state deviation from the set point, nearly no overshoot and a settling time, as fast as the system allows. The controller can be transferred from the simulation to the experiment and also in the real test set up the controller showed good performance. It was also shown that the controller is also able to control the thruster far outside the operating points that were used for training. The controller is able to follow a trajectory that was unknown during training. The trajectory contains different set point changes, linear and sinus pressure changes.

In future this control method will be applied to a nitrous oxide/ethane bipropellant system and hotfire experiments will be conducted. Several challenges come up in comparison to the cold gas experiments described in this paper. The controller has to regulate chamber pressure and mixture ratio independently. The combustion introduces higher roughness, sensor noise and the behavior of the thruster changes depending on the chamber temperature. Faster and more accurate, in house developed electronic control valves will be used in the future experiments to reduce the settling time. In a bipropellant set-up this method has to prove its suitability as rocket engine controller. It will also be possible to test the introduction of different secondary boundaries in the control-law as for example a limit in the wall temperature at a given thrust. Further research is needed to investigate the use of machine learning and neural networks as control method. A direct comparison with other established control mechanisms is desirable to find advantages and disadvantages of the respective method.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] I. Waugh, A. Davies, E. Moore, J. Macfarlane, VTVL technology demonstrator for planetary landers, in: Space Propulsion Conference, 2016.

[2] E. Dumont, S. Ishimoto, P. Tatiossian, J. Klevanski, B. Reimann, T. Ecker, L. Witte, J. Riehmer, M. Sagliano, S. Giagkozoglou, I. Petkov, W. Rotärmel, R. Schwarz, D. Seelbinder, M. Markgraf, J. Sommer, D. Pfau, H. Martens, CALLISTO: a demonstrator for reusable launcher key technologies, in: 32nd ISTS, 2019, URL https://elib.dlr.de/128795/.

[3] J. Vila, J. Hassin, Technology acceleration process for the themis low cost and reusable prototype, in: 8th European Conference for Aeronautics and Space Sciences, 2019, pp. 1–4.

[4] A.P. de Mirand, J.-M. Bahu, O. Gogdet, Ariane next, a vision for the next generation of ariane launchers, Acta Astronaut. 170 (2020) 735–749.

[5] S. Pérez-Roca, J. Marzat, H. Piet-Lahanier, N. Langlois, F. Farago, M. Galeotta, S. Le Gonidec, A survey of automatic control methods for liquid-propellant rocket engines, Prog. Aerosp. Sci. 107 (2019) 63–84.

[6] C.F. Lorenzo, J.L. Musgrave, Overview of rocket engine control, in: AIP Conference Proceedings, Vol. 246, 1992, pp. 446–455.

[7] H. Sunakawa, A. Kurosu, K. Okita, W. Sakai, S. Maeda, A. Ogawara, Automatic thrust and mixture ratio control of le-x, in: 44th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit, 2008, p. 4666.

[8] T. Kai, K. Niu, K. Obase, W. Sakai, Y. Fukuda, T. Hashimoto, M. Sato, S. Takada, T. Kimura, Y. Naruo, et al., Engine control system for the main engine of the reusable sounding rocket, in: Proceedings of the International Astronautical Congress, IAC, Vol. 10, 2015, pp. 7389–7394.

[9] S. Pérez-Roca, J. Marzat, H. Piet-Lahanier, N. Langlois, M. Galeotta, F. Farago, S. Le Gonidec, Model-based robust transient control of reusable liquid-propellant rocket engines, IEEE Trans. Aerosp. Electron. Syst. 57 (1) (2020) 129–144.

[10] G. Waxenegger-Wilfing, U. Sengupta, J. Martin, W. Armbruster, J. Hardi, M. Juniper, M. Oschwald, Early detection of thermoacoustic instabilities in a cryogenic rocket thrust chamber using combustion noise features and machine learning, Chaos 31 (6) (2021) 063128.

[11] C.F. Lorenzo, A. Ray, M.S. Holmes, Nonlinear control of a reusable rocket engine for life extension, J. Propuls. Power 17 (5) (2001) 998–1004.

[12] W. Merrill, C. Lorenzo, A reusable rocket engine intelligent control, in: 24th Joint Propulsion Conference, 1988, p. 3114.

[13] G. Waxenegger-Wilfing, K. Dresia, J. Deeken, M. Oschwald, Machine learning methods for the design and operation of liquid rocket engines–research activities at the DLR institute of space propulsion, 2021, arXiv preprint arXiv:2102.07109.

[14] Günther Waxenegger-Wilfing, Kai Dresia, Jan C. Deeken, Michael Oschwald, A reinforcement learning approach for transient control of liquid rocket engines, IEEE Trans. Aerosp. Electron. Syst. 57 (5) (2021) 2938–2952, URL https://elib.dlr.de/146167/.

[15] R.S. Sutton, A.G. Barto, Reinforcement learning: An introduction, second ed., in: Adaptive Computation and Machine Learning, vol. 2018: 1, The MIT Press, Cambridge, Mass., 2018.

[16] O.M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al., Learning dexterous in-hand manipulation, Int. J. Robot. Res. 39 (1) (2020) 3–20.

[17] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, L. Zhang, Solving Rubik's cube with a robot hand, 2019.

[18] F. Sadeghi, S. Levine, Cad2rl: Real single-image flight without a single real image, 2016, arXiv preprint arXiv:1611.04201.

[19] A. Folkers, Steuerung Eines Autonomen Fahrzeugs Durch Deep Reinforcement Learning, Springer-Verlag GmbH, 2019, URL https://www.ebook.de/de/product/38406162/andreas_folkers_steuerung_eines_autonomen_fahrzeugs_durch_deep_reinforcement_learning.html.

[20] T. Hörger, K. Dresia, G. Waxenegger-Wilfing, L. Werling, S. Schlechtriem, Development of a test infrastructure for a neural network controlled green propellant thruster, in: 8th Space Propulsion Conference, 2022, URL https://elib.dlr.de/186952/.

[21] G. Dulac-Arnold, N. Levine, D.J. Mankowitz, J. Li, C. Paduraru, S. Gowal, T. Hester, An empirical investigation of the challenges of real-world reinforcement learning, 2020, arXiv preprint arXiv:2003.11881.

[22] W. Zhao, J.P. Queralta, T. Westerlund, Sim-to-real transfer in deep reinforcement learning for robotics: a survey, in: 2020 IEEE Symposium Series on Computational Intelligence, SSCI, 2020, pp. 737–744.

[23] G. Rebala, A. Ravi, S. Churiwala, An Introduction to Machine Learning, Springer, 2019.

[24] R.S. Sutton, A.G. Barto, Reinforcement learning: An introduction, second ed., in: Adaptive Computation and Machine Learning, vol. 2018: 1, The MIT Press, Cambridge, Mass., 2018.

[25] M. Sewak, Deep Reinforcement Learning, Springer Verlag, Singapore, 2019, URL https://www.ebook.de/de/product/36524834/mohit_sewak_deep_reinforcement_learning.html.

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, 2013, arXiv preprint arXiv:1312.5602.

[27] R.S. Sutton, D. McAllester, S. Singh, Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, Adv. Neural Inf. Process. Syst. 12 (1999).

[28] S. Fujimoto, H. Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: International Conference on Machine Learning, 2018, pp. 1587–1596.

[29] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, 2015, arXiv preprint arXiv:1509.02971.

[30] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: International Conference on Machine Learning, 2018, pp. 1861–1870.

[31] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M.I. Jordan, et al., Ray: A distributed framework for emerging 5AI6 applications, in: 13th 5USENIX6 Symposium on Operating Systems Design and Implementation, 5OSDI6 18, 2018, pp. 561–577.

[32] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, 2017, http://dx.doi.org/10.48550/ARXIV.1707.06347.

[33] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T.P. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, 2016, http://dx.doi.org/10.48550/ARXIV.1602.01783.

[34] W. Zhao, J.P. Queralta, T. Westerlund, Sim-to-real transfer in deep reinforcement learning for robotics: a survey, in: 2020 IEEE Symposium Series on Computational Intelligence, SSCI, 2020, pp. 737–744.

[35] B. Mehta, M. Diaz, F. Golemo, C.J. Pal, L. Paull, Active domain randomization, in: Conference on Robot Learning, 2020, pp. 1162–1176.

[36] L. Weng, Domain randomization for Sim2Real transfer, 2019, lilianweng.github.io/lil-log. URL http://lilianweng.github.io/lil-log/2019/05/04/domain-randomization.html.

[37] D.-O. Won, K.-R. Müller, S.-W. Lee, An adaptive deep reinforcement learning framework enables curling robots with human-like performance in real-world conditions, Science Robotics 5 (46) (2020) http://dx.doi.org/10.1126/scirobotics.abb9764, URL https://robotics.sciencemag.org/content/5/46/eabb9764.

[38] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, P. Abbeel, Domain randomization for transferring deep neural networks from simulation to the real world, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2017, pp. 23–30, http://dx.doi.org/10.1109/IROS.2017.8202133.

[39] T. Dai, K. Arulkumaran, T. Gerbert, S. Tukra, F. Behbahani, A.A. Bharath, Analysing deep reinforcement learning agents trained with domain randomisation, Neurocomputing 493 (2022) 143–165.

[40] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, D. Fox, Closing the sim-to-real loop: Adapting simulation randomization with real world experience, in: 2019 International Conference on Robotics and Automation, ICRA, 2019, pp. 8973–8979.

[41] L. Werling, Entwicklung und Erprobung von Flammensperren für Einen Vorgemischten, Grünen Raketentreibstoff aus Lachgas und Ethen (Ph.D. thesis), Fakultät für Luft- und Raumfahrttechnik und Geodäsie der Universität Stuttgart.

[42] L.K. Werling, T. Hörger, K. Manassis, D. Grimmeisen, M. Wilhelm, C. Erdmann, H.K. Ciezki, S. Schlechtriem, S. Richter, M. Torsten, et al., Nitrous oxide fuels blends: research on premixed monopropellants at the german aerospace center (DLR) since 2014, in: AIAA Propulsion and Energy 2020, p. 3807.

[43] L. Werling, T. Hörger, Experimental analysis of the heat fluxes during combustion of a N2O/C2H4 premixed green propellant in a research rocket combustor, Acta Astronaut. 189 (2021) 437–451.

[44] M. Kurilov, L. Werling, M. Negri, C. Kirchberger, S. Schlechtriem, Impact sensitiveness of nitromethane-based green-propellant precursor mixtures, Int. J. Energetic Mater. Chem. Propuls. (2022).

[45] F. Lauck, J. Balkenhohl, M. Negri, D. Freudenmann, S. Schlechtriem, Green bipropellant development–a study on the hypergolicity of imidazole thiocyanate ionic liquids with hydrogen peroxide in an automated drop test setup, Combust. Flame 226 (2021) 87–97.

[46] M. Wilhelm, L. Werling, F. Strauss, F. Lauck, C. Kirchberger, H. Ciezki, S. Schlechtriem, Test complex M11: Research on future orbital propulsion systems and scramjet engines, in: International Astronautical Congress, 2019, URL https://elib.dlr.de/133885/.

[47] R.L. Thompson, L. Lassaletta, P.K. Patra, C. Wilson, K.C. Wells, A. Gressent, E.N. Koffi, M.P. Chipperfield, W. Winiwarter, E.A. Davidson, et al., Acceleration of global N2O emissions seen from two decades of atmospheric inversion, Nature Clim. Change 9 (12) (2019) 993–998.

[48] H. Ciezki, L. Werling, M. Negri, F. Strauss, M. Kobald, C. Kirchberger, D. Freudenmann, C. Hendrich, M. Wilhelm, A. Petrarolo, S. Schlechtriem, 50 Years of test complex M11 in lampoldshausen – research on space propulsion systems for tomorrow, in: 7th European Conference for Aeronautics and Space Sciences, EUCASS.

[49] Empresarios Agrupados Internacional, EcosimPro 6.2.0.