A Reinforcement Learning Approach for Transient Control of Liquid Rocket Engines

GÜNTHER WAXENEGGER-WILFING
KAI DRESIA
JAN DEEKEN
MICHAEL OSCHWALD
Institute of Space Propulsion, Hardthausen am Kocher, Germany

Nowadays, liquid rocket engines use closed-loop control at most near-steady operating conditions. The control of the transient phases is traditionally performed in open loop due to highly nonlinear system dynamics. This situation is unsatisfactory, in particular for reusable engines. The open-loop control system cannot provide optimal engine performance due to external disturbances or the degeneration of engine components over time. In this article, we study a deep reinforcement learning approach for optimal control of a generic gas-generator engine's continuous startup phase. It is shown that the learned policy can reach different steady-state operating points and convincingly adapt to changing system parameters. Compared to carefully tuned open-loop sequences and proportional-integral-derivative (PID) controllers, the deep reinforcement learning controller achieves the highest performance. In addition, it requires only minimal computational effort to calculate the control action, which is a big advantage over approaches that require online optimization, such as model predictive control.

Manuscript received September 24, 2020; revised January 11, 2021; released for publication February 25, 2021. Date of publication April 20, 2021; date of current version October 11, 2021.

DOI. No. 10.1109/TAES.2021.3074134

Refereeing of this contribution was handled by G. Chen.

Authors' addresses: The authors are with the Department of Rocket Propulsion, German Aerospace Center (DLR), Institute of Space Propulsion, 74239 Hardthausen am Kocher, Germany, E-mail: (guenther.waxenegger@dlr.de; kai.dresia@dlr.de; jan.deeken@dlr.de; michael.oschwald@dlr.de). *(Günther Waxenegger-Wilfing and Kai Dresia are co-first authors.) (Corresponding author: Günther Waxenegger-Wilfing.)*

I. INTRODUCTION

The demands on the control system of liquid rocket engines have significantly increased in recent years [1], in particular for reusable engines. The aging of reusable engines requires a robust control system as the performance of engine components might degrade over time, e.g., due to soot depositions [2]–[4], increased leakage mass flows caused by seal aging [5], or turbine blade erosions [6]. The cost-efficient operation of a reusable launch vehicle is only possible if the engines possess a long service life without expensive maintenance. Additionally, advanced mission scenarios, e.g., in-orbit maneuvers or propulsive landings, require deep throttling and restart capabilities.

Nowadays, most liquid rocket engines use predefined valve sequences to drive the system from the start signal to a desired steady-state and to shut down the engine safely. These control sequences are usually determined during costly ground tests. Closed-loop control is at most used near steady operating conditions to maintain a desired combustion chamber pressure and mixture ratio [7]. For an extensive overview of the different studied control strategies, which include proportional-integral (PI), PID, linear-quadratic-regulator (LQR), and robust control methods, the reader is referred to the survey by Pérez-Roca *et al.* [8]. The resulting lower deviations of the controlled variables decrease the amount of extra propellant to be carried, which in turn increases the payload capacity of the launch vehicle. Although the importance of closed-loop control has been evident for many years, the majority of rocket engines still employ valves which are operated with pneumatic actuators, too inefficient for a sophisticated closed-loop control system. The development of an all-electric control system started in the late 90s in Europe [9]. The future European Prometheus engine will have such a system [10]. Other countries are also well advanced in the research and development of electrically operated flow control valves [11]. With the electrification of actuators and the grown demands, interest in closed-loop solutions has increased recently and will continue to rise in the future when launch vehicles and the associated rocket engines will be designed with multidisciplinary design optimization tools [12].

Optimal control of the engine operation, including the transient phases, is the only way to realize high-performing systems, which also comply with the aforementioned demands on the control system of future liquid rocket engines [8]. One way to solve optimal control problems is to use reinforcement learning (RL). Although the application of such modern methods of artificial intelligence seems unorthodox in this setting, it offers certain advantages. First, given a suitable simulation environment, RL algorithms can automatically generate optimal transient sequences. Second, the trained RL controller features a minimal computational effort to calculate the control action, so it can easily be used for closed-loop control of the demanding transient phases. Third, RL is perfectly suited for complex control tasks, including multiple objectives and multiple regimes [13]. Optimal control using RL [14] has been

studied in many different areas, from robotics [15], [16] and medical science [17] to flight control [18], [19] and process control [20]. Cheng *et al.* [21] applied RL to real-time control for a fuel-optimal moon landing. A guidance strategy for spacecrafts based on RL has been developed by Hovell and Ulrich [22]. Furthermore, the benefits of an intelligent engine control system, where artificial intelligence techniques are used for control reconfiguration and condition monitoring, have already been investigated in the space shuttle area [23], [24].

The objective of our article is analogous to the investigation of Pérez-Roca *et al.* [25], where a model predictive control (MPC) approach to control the startup transient of a liquid rocket engine was studied. After the derivation of a suitable state-space model [26], a linear MPC controller was synthesized. The controller completes the startup and can track the end-state references with sufficient accuracy. MPC and RL have specific advantages and disadvantages. The work presented here aims to evaluate the capabilities and limitations of RL for liquid rocket engine control.

Our main contributions are the following:

- Formulation of optimal startup control as a RL problem.
- Training and evaluation of the RL controller for multiple operating conditions and degrading turbine efficiencies.
- Quantitative comparison with carefully tuned open-loop sequences and PID controllers.

The remainder of this article is structured as follows. Section II describes the basics of RL and presents pseudocode of the used RL algorithm. The simulation environment and its coupling with the RL algorithms are outlined in Section III. Section IV discusses the test case. Section V reports the results, including the comparison with the performance of PID controllers. Finally, Section VI concludes this article.

II. REINFORCEMENT LEARNING

In this section, we review basic RL concepts [27]. RL algorithms can be used to solve optimal control problems stated as Markov decision processes (MDPs) [28]. MDPs provide a mathematical framework for modeling decision making in situations where the system changes possibly in a stochastic manner. Standard MDPs work in discrete time: at each time step, the controller (usually called the agent in RL) receives information on the state of the system and takes an action in response. The decision rule is called a policy in RL. The action changes the state of the system, and the latest transition is evaluated via a reward function. The optimal control objective is to maximize the (expected) cumulative reward from each initial state. Formally, an MDP consists of the state-space X of the system, the action (input) space U, the transition function (dynamics) f of the system, and the reward function ρ (negative costs). Due to the origins of the field in artificial intelligence, the usual notation would be S for the state-space, A for the action space, P for the dynamics, and R for the reward function. In this article, notation inspired by control theory is used. As a result of the action u_k applied in state x_k at discrete time step k, the state changes to x_{k+1} and a scalar reward $r_{k+1} = \rho(x_k, u_k, x_{k+1})$ is received. The goal is to find a policy π, so that $u_k = \pi(x_k)$, that maximizes the cumulative reward, typically the expected discounted sum over the infinite horizon

$$\mathbb{E}_{x_{k+1}\sim f(x_k,\pi(x_k),\cdot)}\left\{\sum_{k=0}^{\infty}\gamma^k\rho(x_k,\pi(x_k),x_{k+1})\right\} \tag{1}$$

where $\gamma \in (0, 1]$ is the discount factor. The mapping from a state x_0 to the value of the cumulative reward for a policy π is called the (state) value function $V^\pi(x_0)$

$$V^\pi(x_0) = \mathbb{E}_{x_{k+1}\sim f(x_k,\pi(x_k),\cdot)}\left\{\sum_{k=0}^{\infty}\gamma^k\rho(x_k,\pi(x_k),x_{k+1})\right\}. \tag{2}$$

The control objective is to find an optimal policy π^* that leads to the maximal value function, for all x_0

$$V^*(x_0) := \max_\pi V^\pi(x_0), \forall x_0. \tag{3}$$

Although state-values functions suffice to define optimality, it is useful to define action-value functions, called Q-functions. The action-value function gives the expected reward if one starts in state x, takes an arbitrary action u (which may not have come from the policy), and then forever after acts according to policy π

$$Q^\pi(x,u) = \mathbb{E}_{x'\sim f(x,u,\cdot)}\{\rho(x,u,x') + \gamma V^\pi(x')\} \tag{4}$$

where the prime notation indicates quantities at the next discrete time step. The optimal Q-function Q^* is defined using V^*. Once an optimal Q-function Q^* is available, an optimal policy π^* can be computed by

$$\pi^*(x) \in \arg\max_u Q^*(x,u), \tag{5}$$

while the formula to compute π^* from V^* is more complicated. As a consequence of the definitions, the Q-functions Q^π and Q^* fulfill the Bellman equations

$$Q^\pi(x,u) = \mathbb{E}_{x'\sim f(x,u,\cdot)}\{\rho(x,u,x') + \gamma Q^\pi(x,'\pi(x'))\} \tag{6}$$

and

$$Q^*(x,u) = \mathbb{E}_{x'\sim f(x,u,\cdot)}\{\rho(x,u,x') + \gamma \max_{u'} Q^*(x,'u')\} \tag{7}$$

which are of central importance in RL. The crucial advantage of RL algorithms is that they do not require a model of the system dynamics. Instead, an optimal policy can be found by learning from samples of transitions and rewards. The problem formulation with MDPs and the associated solution techniques also handle nonlinear, stochastic dynamics, and nonquadratic reward functions. Perhaps, the most popular RL algorithm is Q-learning. In Q-learning, one starts from an arbitrary initial Q-function Q_0 and updates it using observed state transitions and rewards. The update rule is of the following form:

$$\begin{aligned} Q_{k+1}(x_k,u_k) = {} & Q_k(x_k,u_k) \\ & + \alpha_k[r_{k+1} + \gamma \max_{u'} Q_k(x_{k+1},u') - Q_k(x_k,u_k)] \end{aligned} \tag{8}$$

where $\alpha_k \in (0, 1]$ is the learning rate. The term inside the square bracket is nothing else than the difference between the updated estimate of the optimal Q-value of (x_k, u_k) and the current estimate $Q_k(x_k, u_k)$. Under mild assumptions on the learning rate and that a suitable exploratory policy is used to obtain samples, i.e., data tuples of the form $(x_k, u_k, x_{k+a}, r_{k+1})$, Q-learning asymptotically converges to Q^*, which satisfies the Bellman optimality equation. The reader is referred to [29] for a description of similar RL algorithms. Q-learning and its many variants require that Q-functions and policies are exactly represented, e.g., as a table indexed by the discrete states and actions. Especially for the control of physical systems, the states and actions are continuous; moreover, exact representations are in general impossible. Normal Q-learning does not work in this setting. Fortunately, methods like Q-learning can be combined with function approximation. We denote approximate versions of the Q-function and the policy by $\hat{Q}(x, u; \theta)$ and $\hat{\pi}(x; w)$, where θ and w are the parameters of parametric approximators. There are many different function approximators to choose from.

The combination of RL with deep neural networks (DNNs) as function approximators leads to the field of deep RL. In the last years, deep RL algorithms have achieved impressive results, such as reaching super-human performance in the game of Go. Besides the sensational results in board games or video games, those algorithms are successfully used in areas like robotics. In deep Q-learning, one uses a neural network to approximate the Q-function. Neural networks can represent any smooth function arbitrarily well given enough parameters, and therefore they can learn complex Q-functions. Loss functions and gradient descent optimization are used to fit the parameters of the models. Gradient estimates are usually averaged over individual gradients computed for a batch of experiences.

Nevertheless, the simple training procedure is unstable, because sequential observations are correlated, and techniques like experience replay have to be used. Correlated experiences are saved into a replay buffer. When batches of experiences are needed for training, these batches are generated by sampling from the replay buffer in a randomized order. A further reason for the simple training procedure's instability is that the target values depend on the parameters one wants to optimize. The solution is to use a so-called target network, $\hat{Q}(x, u; \theta^-)$, with target parameters θ^-, which slowly track the online parameters. While deep Q-learning solves problems with continuous state-spaces, it can only handle discrete and low-dimensional action spaces. The reason for that is the following: (deep) Q-learning requires fast maximization of Q-functions over actions. When there are a finite number of discrete actions, this poses no problem. However, when the action space is continuous, this is highly nontrivial (and would be a very computational expensive subroutine).

The deep deterministic policy gradient (DDPG) [30] algorithm is specially adapted for environments with continuous action spaces. It uses neural networks to approximate both the Q-function and a deterministic policy, i.e., the policy network deterministically maps a state to a specific action. For exploration, one adds noise sampled from a stochastic process $\mathcal{N}$ to the actions of the deterministic policy and updates it by a gradient-based learning rule. As in deep Q-learning, the DDPG algorithm uses a replay buffer and target networks to improve stability during neural network training. Further details of the DDPG algorithm and its performance on different simulated physics tasks are given by Lillicrap *et al.* [30].

Algorithm 1: Twin Delayed DDPG (TD3).

Input: initial policy parameters w, Q-function parameters θ_1, θ_2, empty replay buffer $\mathcal{D}$
Set target parameters equal to main parameters $w^- \leftarrow w, \theta_1^- \leftarrow \theta_1, \theta_2^- \leftarrow \theta_2$
repeat
Observe state x and select action $u = \text{clip}(\hat{\pi}(x; w) + \epsilon, x_{\text{Low}}, x_{\text{High}})$, where $\epsilon \sim \mathcal{N}$
Execute u in the environment
Observe next state x', reward r, and done signal d to indicate whether x' is terminal
Store $(x, u, r, x,' d)$ in replay buffer $\mathcal{D}$
If x' is terminal, reset environment state
if it is time to update **then**
  for j in range(however many updates) **do**
    Randomly sample a batch of transitions $B = \{(x, u, r, x,' d)\}$ from $\mathcal{D}$
    Compute target actions

$$u'(x') = \text{clip}(\hat{\pi}(x'; w^-) + \text{clip}(\epsilon, -c, c), x_{\text{Low}}, x_{\text{High}}), \quad \epsilon \sim \mathcal{N}(0, \sigma)$$

    Compute targets

$$q(r, x,' d) = r + \gamma(1 - d) \min_{i=1,2} \hat{Q}(x,' u'(x'); \theta_i^-)$$

    Update Q-functions by one step of gradient descent

$$\nabla_{\theta_i} \frac{1}{|B|} \sum_{(x,u,r,x,'d) \in B} (\hat{Q}(x, u; \theta_i) - q(r, x,' d))^2,$$

  for $i = 1, 2$
    if $j \bmod policydelay = 0$ **then**
      Update policy by one step of gradient ascent

$$\nabla_w \frac{1}{|B|} \sum_{x \in B} \hat{Q}(x, \hat{\pi}(x; w); \theta_1)$$

      Update target networks

$$\theta_i^- \leftarrow (1 - \tau)\theta_i^- + \tau\theta_i, \quad \text{for } i = 1, 2$$
$$w^- \leftarrow (1 - \tau)w^- + \tau w$$

    end if
  end for
end if
until convergence

Although the DDPG algorithm is quite powerful, it has a direct successor, the Twin Delayed DDPG (TD3) [31] algorithm, which further improves the stability by employing three critical tricks. The first trick addresses a particular failure mode of the DDPG algorithm: if the Q-function approximator develops an incorrect sharp peak for some actions, the policy will quickly exploit that peak and then have brittle or incorrect behavior. This failure mode can be averted by smoothing out the Q-function over similar actions. For this, one computes the action that is used to form the Q-learning target in the following way:

$$u'(x') = \text{clip}(\hat{\pi}(x'; w^-) + \text{clip}(\epsilon, -c, c), x_{\text{Low}}, x_{\text{High}}) \quad (9)$$

where $\epsilon \sim \mathcal{N}(0, \sigma)$ is noise sampled from a Gaussian process (target policy noise). The action is based on the target policy, but with clipped noise added (target noise clip c). After adding the noise, the target action is also clipped to lie in the valid action range $(x_{\text{Low}}, x_{\text{High}})$. The second trick is to learn two Q-functions $\hat{Q}(x, u; \theta_i)$, for $i = 1, 2$, instead of one and use the smaller of the two Q-values to form the target. This improvement reduces overestimation in the Q-function.

$$q(r, x,' d) = r + \gamma(1 - d) \min_{i=1,2} \hat{Q}(x,' u'(x'); \theta_i^-). \quad (10)$$

The third trick is to update the policy less frequently than the Q-functions (policy delay) to damp the volatility that arises in the DDPG algorithm. Algorithm 1 shows the full pseudocode of the TD3 algorithm. The done signal d is equal to one when x' is the terminal state and otherwise equal to zero. The done signal guarantees that the agent gets no additional rewards after the current state at the end of an episode.

In addition to enhancements that improve the stability of the training process, research is also carried out to speed up the learning process of RL agents [32]. Besides DDPG, TD3, or SAC [33], which are so-called off-policy algorithms, there are also state-of-the-art on-policy algorithms like TRPO [34] or PPO [35]. Nevertheless, on-policy methods are much more sample inefficient and have longer training time to achieve equivalent performances. From a control perspective, RL converts the system identification problem and the optimal control problem to machine learning problems. Similar to explicit MPC, it also addresses the problem of removing one of the main drawbacks of MPC, namely the need to solve a complex optimization problem online to compute the control action.

The main advantages of RL for control are as follows.

- No derivation of a suitable state-space model, model order reduction, or linearization needed.
- Direct use of a nonlinear simulation model.
- Ideal for highly dynamic situations (no complex online optimization needed).
- Complex reward functions enable complicated goals.

The main disadvantage of RL for control are as follows.

- Stability of the controller is in general not guaranteed.

Concerning the last point (stability), we would like to make a remark. The controller's output can always be tested using the simulation environment, and there has been promising recent work on certifying stability of RL policies [36].

III. SIMULATION ENVIRONMENT AND RL IMPLEMENTATION

A suitable simulation environment for our intended use is given by EcosimPro [37]. EcosimPro is a modeling and simulation tool for 0D or 1D multidisciplinary continuous and discrete systems. The system description is based on differential-algebraic equations and discrete events. Within a graphical user interface, one can combine different components, which are arranged in several libraries. Of particular interest are the European Space Propulsion System Simulation (ESPSS) libraries, which are commissioned by the European Space Agency (ESA). These EcosimPro libraries are suited for the simulation of liquid rocket engines and have continuously been upgraded in recent years. Furthermore, EcosimPro models can be converted to run on typical hardware-in-the-loop simulators that could, for example, be used to demonstrate the reliability of advanced control algorithms deployed on space-graded embedded systems [38].

We use the TD3 implementation of Stable-Baselines [39]. Stable-Baselines is a set of improved implementations of RL algorithms based on OpenAI Baselines. It features a common interface for many modern RL algorithms and additional wrappers for preprocessing, monitoring, and multiprocessing. We encapsulate our simulation environment into a custom OpenAI Gym environment using an interface between EcosimPro and Python. Hence, we can directly use Stable-Baselines for training and testing. A big advantage of the RL approach is that it works regardless of whether one uses a lumped parameter model, continuous state-space models, surrogate models employing artificial neural networks [40], [41], or a combination of the above.

IV. TEST CASE

The engine architecture considered to study the suitability of an RL approach for the control of the transient startup is shown in Fig. 1. It is similar to the architecture of the European Vulcain 1 engine [42], which powered the cryogenic core stage of Ariane 5 launch vehicle before it got replaced by the upgraded Vulcain 2 engine. It is fed with cryogenic liquid oxygen (LOX) at a temperature of 92 K and liquid hydrogen (LH2) at 22 K. The engine generates approximately 1 MN of thrust at a main combustion chamber (CC) pressure of 100 bar and a chamber mixture ratio of 5.6, i.e., the chamber mass flow of the oxidizer divided by the chamber mass flow of the fuel equals 5.6. The engine cycle is an open gas-generator cycle, where a small amount of the propellants is burned in a small combustion chamber, the gas-generator (GG). The gas-generator is operated at a fuel-rich mixture ratio of 0.9.

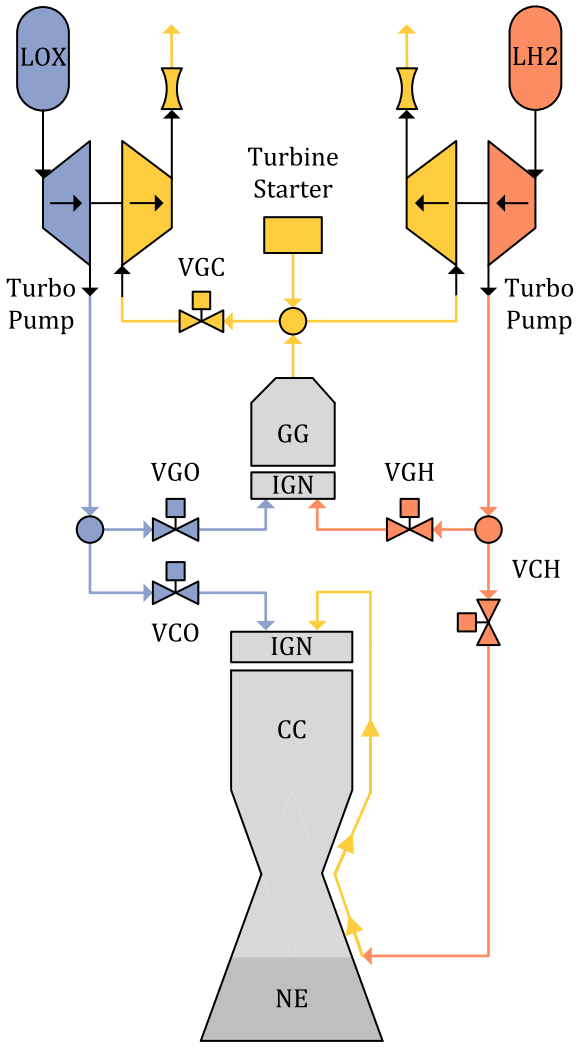

Fig. 1. Flow plan of the considered engine architecture. Some of the propellants are burned in an additional combustion chamber, the gas-generator (GG), and the resulting hot gas is used as the working medium of the turbines which power the engine's pumps. The gas is then exhausted. The engine architecture features five valves, but only three valves (VGH, VGO, VGC) are used for closed-loop control.

The produced hot gas is used to drive the turbines before it is exhausted. The turbines power the pumps which force the propellants into the combustion chambers. LH2 is used to cool the nozzle and main combustion chamber before it gets burned. A convergent-divergent nozzle, which usually includes an uncooled nozzle extension (NE), accelerates the combustion gases to generate thrust.

The actuators are given by five flow control valves (VCO, VCH, VGO, VGH, VGC). VCO and VCH are the main combustion chamber valves that regulate the propellant flow to the combustion chamber. VGO and VGH, the gas-generator valves, are used to control the gas-generator pressure and mixture ratio. The turbine valve, VGC, is located downstream of the gas-generator and is used to determine the hot-gas flow ratio between the LOX and LH2 turbines. Thus, this valve mainly influences the global mixture ratio (PI, pump-inlet). Further actuators are the ignition systems (IGN) for the main combustion chamber and the gas-generator, as well as a turbine starter. The turbine starter produces hot gas for a short period to spin up the turbines during the startup.

To start the engine and reach steady-state conditions, a succession of discrete events, including valve openings and chamber ignitions are necessary. The startup sequence of an engine, i.e., the chronological order of oxidizer and fuel valve openings, as well as the precise ignition timings, determines the engine's thermodynamic conditions and mechanical stresses during start-up. A nonideal startup sequence can damage the engine, e.g., by excessive temperatures. These high temperatures can substantially damage the turbine blades or at least reduce their live expectancy [43]. An optimal startup sequence leads to a smooth ignition of the combustion chamber and gas-generator with low thermal and mechanical stresses. An open-loop startup sequence (OLS) for a steady-state chamber pressure of 100 bar is shown in Fig. 2. The sequence does not correspond exactly to the Vulcain 1 startup sequence, but it is realistic for such an engine cycle. The flow control valves are opened monotonically until the end positions are reached. First, the VCH valve starts to open at $t = 0.1$ s, followed by VCO at $t = 0.6$ s. A fuel-lead transient is usually used for a smooth ignition of the combustion chamber, which takes place at $t = 1.0$ s. At this point, the main combustion chamber is burning at low pressure, only fed by the tank pressurization. At $t = 1.1$ s, the turbine starter activates to spin up the turbopumps, which start to build up the pressure in the main combustion chamber and at the gas-generator valves VGO, VGH. At $t = 1.4$ s and $t = 1.5$ s, the gas-generator valves VGH and VGO open and the gas-generator is ignited. The VGC valve is set to a fixed position during the entire startup sequence. At $t = 2.6$ s, the turbine starter is burned out and the engine reaches steady-state conditions after approximately 4 s. The valve positions in Fig. 2 are tuned to reach a main combustion chamber pressure p_{cc} of 100 bar, a global mixture ratio MR_{PI} of 5.2, and a gas-generator mixture ratio MR_{GG} of 0.9.

Although RL can solve discrete or hybrid control problems, there are controllability and observability issues during the first phase of discrete events due to very low mass flows [24]. Thus, we focus on the fully continuous phase starting at $t = 1.5$ s. The goal of the controller (agent) is to drive the engine as fast as possible toward the desired reference by adjusting the flow control valve positions. In our multi-input–multi-output (MIMO) control tasks, only three flow control valves, VGO, VGH, and VGC, are used for active control of the combustion chamber pressure, and the mixture ratio of the gas-generator as well as the global mixture ratio. The valve actuators are modeled as a first-order transfer function with a time constant of $\tau = 0.05$ s and a linear valve characteristic. The minimum valve position is set to 0.25 for VGH and VGO and 0.20 for VGC, respectively. The maximum valve position is 1.0 for all valves.

We study different reference values for the combustion chamber pressure, namely 80 and 100 bar. The reference mixture ratios remain the same, 5.2 for the global mixture

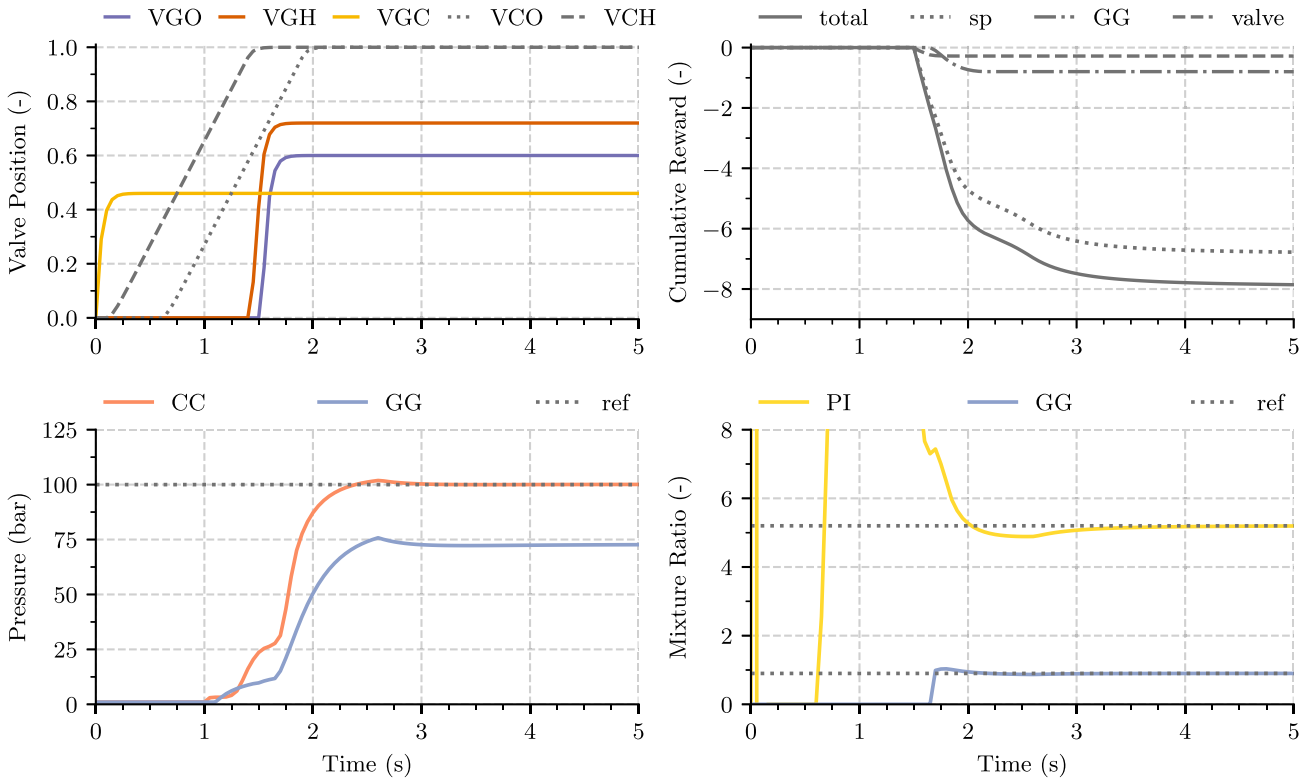

Fig. 2. 100 bar nominal open-loop start-up sequence. The main combustion chamber pressure settles at 100 bar, while the gas-generator pressure reaches 75 bar. The reference mixture ratios are given by 5.2 and 0.9. The engine reaches steady-state conditions after approximately 4 s.

ratio, and 0.9 for the mixture ratio of the gas-generator. For a combustion chamber pressure of 80 bar, the valve timings are the same, but the final valve positions were adjusted accordingly (see Fig. 9). Furthermore, we study the effect of degrading turbine efficiencies on the startup transient. This scenario has practical relevance for future reusable engines. The use of cryogenic propellants leads to significant thermostructural challenges in the operation of turbopumps. Since thermal stresses depend on the temperature gradient, they can cause significant loads on the metal parts that have to react to these stresses. The resulting fatigue deformation [43] affects the performance of the turbines. Furthermore, the aging of seals can cause increased leakage mass flows, which in turn decreases the turbine efficiency [5]. Additional reasons are turbine blade erosions [6] and soot depositions on the turbine nozzles by fuel-rich gases when using hydrocarbons as fuel. These soot depositions can decrease the effective nozzle area up to 20% [2], thus reducing the turbopump performance. Furthermore, soot depositions are a main shortcoming for reusable engines due to the unpredictable impacts for engine restart [44]. To study the effect of degrading turbine efficiencies for our generic test case, we simulate and evaluate the performance of the open-loop startup sequence, a family of PID controllers, and our RL-agent for 16 different combinations of LOX and LH2 turbine efficiencies. For each turbine, four different efficiencies are considered ranging from 100 to 85% of the nominal value.

The reward, which is used to evaluate a startup sequence and to train the RL agent, consists of three different terms

$$r = r_{\text{sp}} + r_{\text{GG}} + r_{\text{valve}}. \tag{11}$$

The first term

$$r_{\text{sp}} = -\sum_{x_{\text{i}}} \text{clip}\left(\left|\frac{x_{\text{i}} - x_{i,ref}}{x_{i,ref}}\right|, 0.2\right) \tag{12}$$

for $x_{\text{i}} \in [p_{\text{CC}}, MR_{\text{GG}}, MR_{\text{PI}}]$ penalizes deviations from the desired set-point for all controlled variables. Each reward component in this term is clipped to a maximum value of 0.2 to improve training and to balance the accumulated reward during startup and steady state. The second term of the reward

$$r_{\text{GG}} = -\begin{cases} \frac{MR_{\text{GG}} - MR_{\text{GG,ref}}}{MR_{\text{GG,ref}}}, & \text{if } \frac{MR_{\text{GG}}}{MR_{\text{GG,ref}}} > 1 \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

additionally penalizes high mixture ratios in the gas-generator. High mixture ratios are dangerous because they result in increased temperatures and thus possible damaging conditions to the turbines. The last reward component

$$r_{\text{valve}} = -\frac{|s_{\text{VGH}}| + |s_{\text{VGO}}| + |s_{\text{VGC}}|}{3} \tag{14}$$

where s is the change in valve position between two time steps, and penalizes excessive valve motion. By adding this component, we encourage the agent to move the valves as little as possible to avoid valve wear, valve oscillations, and

valve jittering. All together, this reward allows the agent to trade off between reaching the desired reference point as fast as possible, avoiding steady-state errors, minimizing overshoots, and reducing valve motion as much as possible. Fig. 2 shows all three components of the cumulative reward for the nominal OLS for 100 bar. Since the valves are only moved once in the OLS, the contribution of r_{valve} to the total reward is low. As the overshoot in the gas-generator mixture ratio is also small (small r_{GG}), the total reward is mainly composed of the set point error r_{sp}.

To train and use an RL agent, one needs to define the observation and action space of the agent. The observation space, i.e., the variables the agent receives from the environment at each time step, should at least contain sufficient information to unambiguously define the state of the system. In our setup, the observation space

$$X = [p_{\text{cc,ref}},\ \epsilon_{\text{cc}},\ \epsilon_{\text{PI}},\ \epsilon_{\text{GG}},\ Pos_{\text{VGO}},\ Pos_{\text{VGH}},\ Pos_{\text{VGC}},\ \omega_{\text{LOX}},\ \omega_{\text{LH2}}] \tag{15}$$

contains nine variables, where $\epsilon_{\text{i}} = x_{\text{i}} - x_{\text{i,ref}}$ is the absolute error for each controlled variable, Pos_{VGO}, Pos_{VGH}, and Pos_{VGC} are the positions of all control valves, and ω_{LOX} and ω_{LH2} are the rotational speeds of the turbopumps. The observation space is normalized with the reference steady-state values. All variables in our observation state are measurable in real engines. Thus, our approach is not limited to simulation environments, where one could possibly use variables that are impossible to measure directly in real engines (e.g., the turbine efficiencies). The agent's action space U consists of all three gas-generator valve positions

$$U = [Pos_{\text{VGO}},\ Pos_{\text{VGH}},\ Pos_{\text{VGC}}]. \tag{16}$$

At each time step, the RL agent receives observations from the environment and sends control signals to the flow control valves of the engine. The frequency of interaction between the controller (RL-agent and PID) and the environment is set to 25 Hz.

The development of an RL-based control system is divided into a training and a evaluation phase. During the training phase, the agent gains experience by interacting with the environment. The agent uses this experience to adapt its policy and thus find an optimal control strategy. During the evaluation phase, the policy is fixed and the agent's performance is evaluated for different conditions.

In our case, during training, we vary the initial state by randomly choosing different turbine efficiencies as described above. Sufficient exploration noise guarantees that the agent learns a suitable control strategy. The set-points are given by a combustion chamber pressure of 80 and 100 bar, respectively. In the evaluation phase, the performance of the policy is calculated for all 16 different combinations of LOX and LH2 turbine efficiencies and both set-points. The state-dependent control action is deterministic and the exploration noise is set to zero. More information on setting up RL agents for continuous control problems is given by Riedmiller [45].

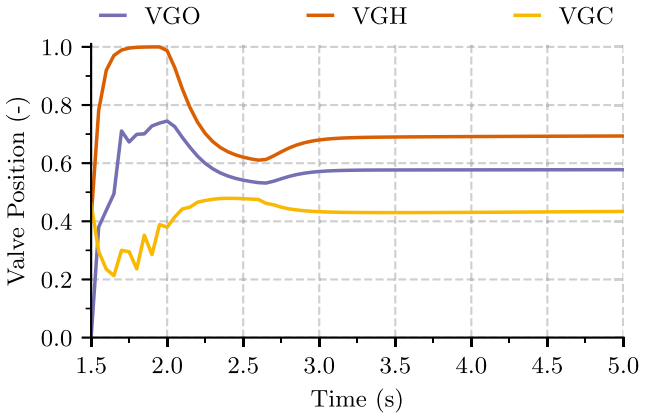

Fig. 3. Manipulated valve positions by the PID controllers for the 100 bar nominal startup. VGO is used to control the mixture ratio of the gas-generator, while VHG and VGC control the pressure of the main combustion chamber and the global mixture ratio, respectively.

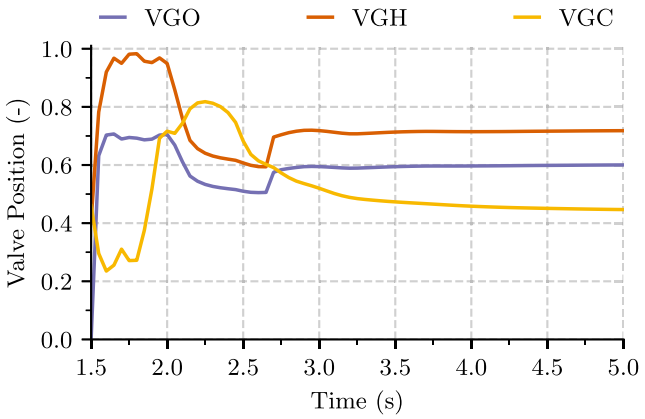

Fig. 4. Manipulated valve positions by the RL agent for the 100 bar nominal startup. The action clearly changes at $t = 2.6$ s, which is the time when the firing of turbine starter stops.

V. RESULTS

In this section, we assess the performance of our RL controller. For this, we use the approximation of the integrated absolute error over one entire episode for each controlled variable

$$(IAE)_i = \int |\epsilon_i|\, dt \approx \sum_{t_j} |\epsilon_i(t_j)| \tag{17}$$

where t_j are the discrete time steps. Furthermore, we evaluate the average steady-state values of the controlled variables from $t = 3.5$ s to $t = 5.0$ s and the value of the cumulative reward.

Before we turn to the performance of closed-loop control, let us record the downsides of OLS. The first column in Fig. 5 shows the resulting engine startup for the nominal OLS and degrading turbine efficiencies. For the latter, the steady-state values deviate strongly from the reference values. The minimum steady-state value of the main combustion chamber pressure is 92 bar. The steady-state of the global mixture ratio varies between 4.9 and 6.0. To

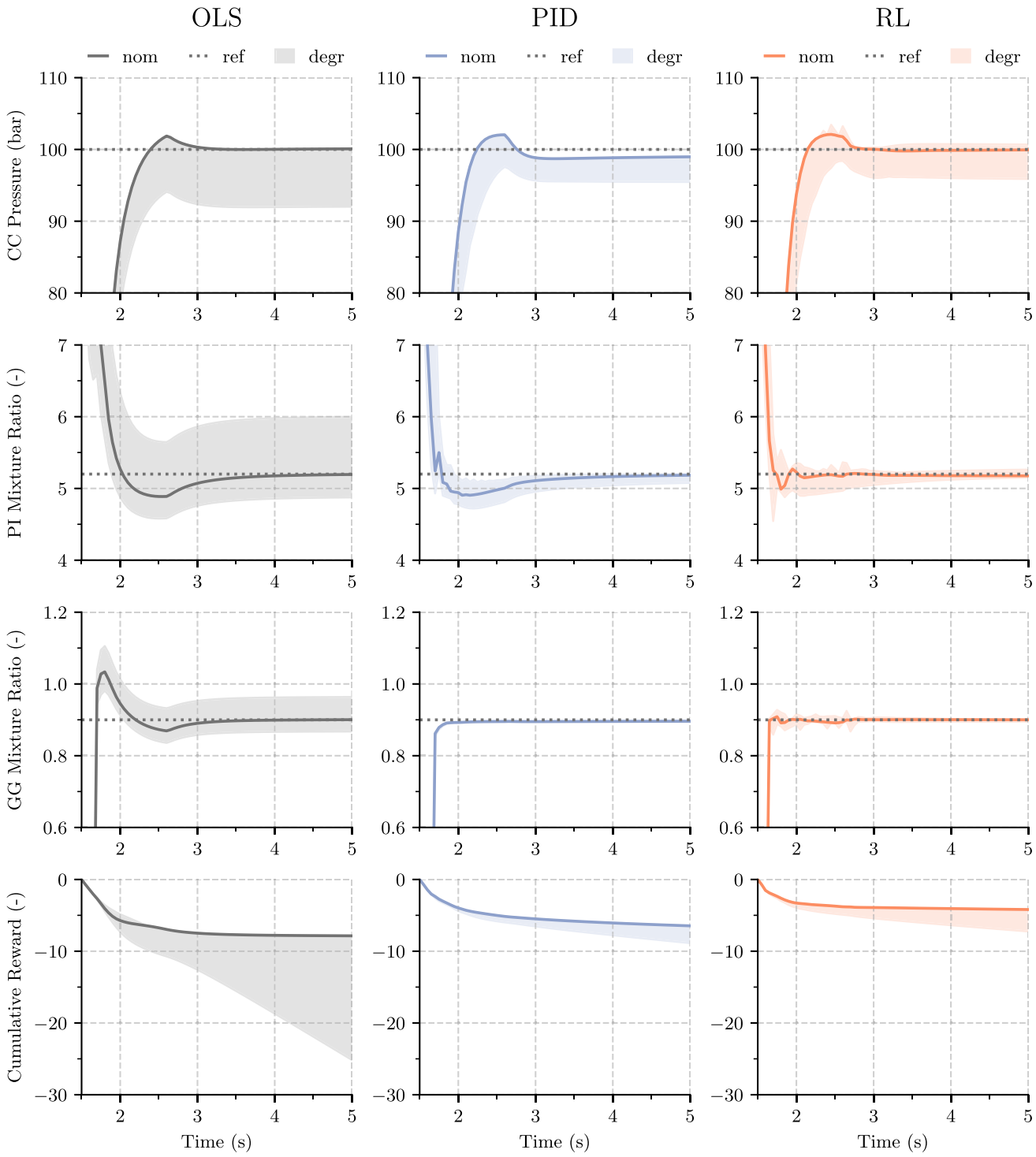

Fig. 5. Comparison of the controlled variables for the 100 bar startup. Shaded area marks the range of the controlled variable for different degraded efficiencies. At different turbine efficiencies, the standard open-loop sequence provides significantly different steady-state values for the chamber pressure and the mixture ratios.

prevent fuel or oxidizer from running out during a mission in the event of a persisting mixing ratio deviation, the loaded propellants must be increased, which reduces the payload capacity of the launch vehicle. A further negative effect is that the temperature in the combustion chamber can rise significantly due to a shift in the mixing ratio, which could reduce the engine's service life. Additionally, the steady-state value of the mixture ratio of the gas-generator changes too. The temperature in the gas-generator is sensitive to the mixture ratio, and an increased temperature can also damage the turbines. These damaging conditions are especially problematic for reusable engines, which must possess a long

TABLE I
Controller Performance for Nominal Turbine Efficiencies

Target	Algo.	Reward	Steady-State Values			*IAE*		
p_{CC} (bar)		cum. (–)	p_{CC} (bar)	MR_{GG} (–)	MR_{PI} (–)	p_{CC} (bar)	MR_{GG} (–)	MR_{PI} (–)
100	OLS	-7.9	**100.0**	**0.90**	**5.18**	591	4.7	27
	PID	-6.5	98.9	**0.90**	5.17	632	4.0	19
	RL	**-4.2**	99.9	**0.90**	**5.18**	**519**	**2.8**	**13**
80	OLS	-7.0	79.7	**0.90**	**5.20**	576	6.0	15
	PID	-5.2	**79.9**	**0.90**	**5.20**	433	3.8	9
	RL	**-4.4**	80.8	**0.90**	5.18	**366**	**2.9**	**8**

service life. The same implications apply to the 80 bar case as shown in Fig. 8.

Those unfavorable effects can be counteracted with a closed-loop control system. First, we tune a family of PID controllers to achieve the startup. The process of controlling the chamber pressure of the main combustion chamber, the mixture ratio of the gas-generator, and the global mixture ratio by manipulating VGO, VGH, and VGC is coupled. For example, changing VGO does affect not only the mixture ratio of the gas-generator but also the other two controlled variables. Nevertheless, for rocket engine control near steady-state conditions, the standard approach is to use separate PID controllers and tune the control loops at different speeds to avoid oscillations [7]. Hence, we also use three separate controllers.

The first controller manipulates VGO to control the mixture ratio of the gas-generator, the second controller manipulates VGH to control the chamber pressure of the main combustion chamber, and the third controller manipulates VGC to control the global mixture ratio. Starting far away from the reference point can be problematic for a simple PID controller because the integrator begins to accumulate a significant error during the rise. Consequently, a large overshoot may occur. Modern PID controllers use different methods to address this problem of integrator-windup. We use a simple feedback loop, where the difference between the actual and the commanded actuator position is fed back to the integrator, to avoid the effects of saturation. If there is no saturation, our anti-windup scheme has no effect. The ratio between the time constant for the anti-windup and the integration time is 0.1 for all PID controllers.

For PID parameter tuning, we directly use the simulation model coupled with a genetic algorithm [46] of the Distributed Evolutionary Algorithms in Python (DEAP) framework [47]. To guarantee a fair comparison, we use the reward function to calculate the fitness value of a certain parameter combination. Table IV presents the optimal PID parameters, which maximize the reward function. The genetic algorithm uses a population of 5000 valid individuals and evolves the population for 20 generations. Fig. 3 shows that the best PID controllers open the valves in a nonmonotonic way, which leads to a faster startup. Furthermore, the PID controllers fulfill their main task: the feedback

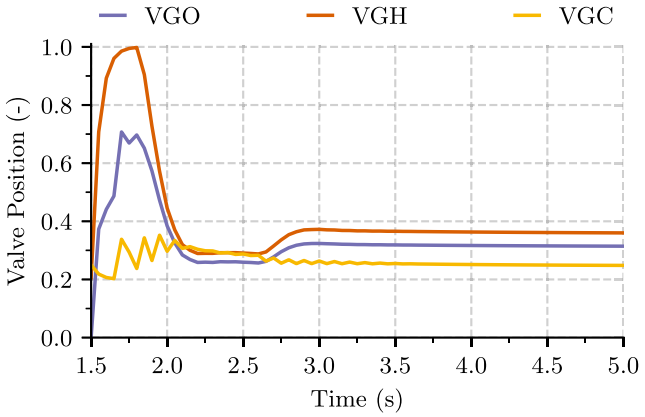

Fig. 6. Manipulated valve positions by the PID controllers for the 80 bar nominal startup. VGO is used to control the mixture ratio of the gas-generator, while VHG and VGC control the pressure of the main combustion chamber and the global mixture ratio, respectively.

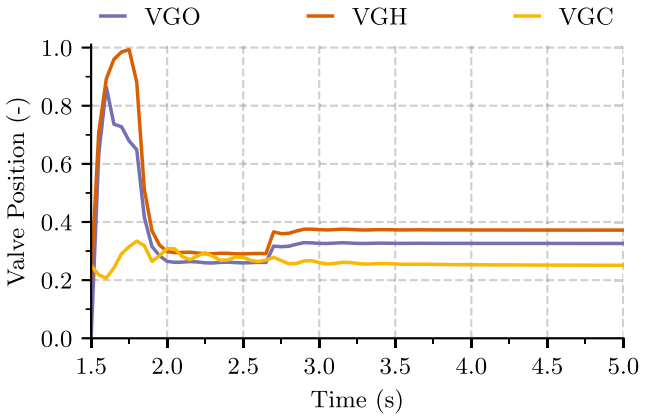

Fig. 7. Manipulated valve positions by the RL agent for the 80 bar nominal startup. The action clearly changes at $t = 2.6$ s, which is the time when the firing of turbine starter stops.

loops lead to an adjustment of the valve positions at lower turbine efficiencies and significantly reduce the deviations from the reference values of the controlled variables. Due to the structure of PID controllers, with their proportional, integral, and derivative terms, the shape of the control input is restricted and does not provide optimal control.

Fig. 5 shows that the optimized PID controllers lead to certain overshoots of the main combustion chamber pressure and the global mixture ratio. It is possible to eliminate the overshoots by changing the PID parameters, but this would significantly increase the settling time. For our parameters, there is still an error in combustion chamber pressure after 4 s even for nominal efficiencies and this steady-state error is only decreasing slowly. In principle, it is possible to reduce the steady-state error of the PID controller faster by increasing the integral component. However, this has a negative effect on the overall dynamic behavior with respect to the reward function used for the comparison. Therefore, the PID controllers shown here provide the optimal result for the reward function, leading to the fairest possible comparison with the other control approaches.

TABLE II
Controller Performance for 16 Different Combinations of Degraded Turbine Efficiencies

Target	Algo.	Reward		Steady-State Values												Integral Absolute Error *IAE*					
p_{CC} (bar)		cumulative (–)		p_{CC} (bar)				MR_{GG} (–)				MR_{PI} (–)				p_{CC} (bar)		MR_{GG} (–)		MR_{PI} (–)	
		mean	sd	min	max	mean	sd	min	max	mean	sd	min	max	mean	sd	mean	std	mean	std	mean	std
100	OLS	-14.9	5.2	92.0	**100.0**	96.1	2.3	0.87	0.96	0.91	0.03	4.9	6.0	5.4	0.3	1841	151	5.8	1.0	45	18
	PID	-7.6	**0.7**	95.5	98.9	97.7	**1.0**	**0.89**	**0.90**	0.89	**0.00**	5.0	**5.2**	**5.2**	0.1	758	98	4.1	**0.1**	**25**	**4**
	RL	**-5.5**	0.9	**96.0**	100.7	**98.8**	1.4	**0.89**	**0.90**	**0.90**	0.02	**5.1**	5.3	**5.2**	**0.0**	**636**	**91**	**2.9**	**0.1**	29	6
80	OLS	-15.4	5.8	71.1	79.7	75.5	2.4	0.86	0.96	0.91	0.03	4.8	6.2	5.5	0.4	815	149	7.0	0.9	36	21
	PID	-5.7	**0.6**	79.5	**79.9**	**79.7**	**0.1**	**0.90**	**0.90**	**0.90**	**0.00**	**5.2**	5.5	**5.2**	0.1	459	**18**	3.9	**0.1**	12	6
	RL	**-4.4**	0.8	**79.6**	82.1	**80.3**	0.6	0.89	**0.90**	**0.90**	**0.00**	**5.2**	**5.4**	**5.2**	**0.0**	**366**	30	**3.3**	**0.1**	**10**	**5**

[0]For each turbine, four different efficiencies are considered ranging from 100% to 85% of the nominal value.

TABLE III
TD3 Hyperparameters

Parameter	Value
number of hidden units per layer	[400, 300]
number of hidden layer	2
activation function	$ReLU$
optimizer	Adam
number of samples per minibatch	256
learning rate	0.001
soft update coefficient (τ)	0.005
train frequency	10
gradient steps	10
discount rate (γ)	0.90
warm-up steps	5000
total training steps	100 000
size of the replay buffer	25 000
target policy noise	0.01
target noise clip	0.02
policy delay	2
action noise type	Ornstein-Uhlenbeck
action noise std (σ)	0.05
rate of mean reversion (θ)	0.25

TABLE IV
PID Parameters

Valve	Controlled Variable	Parameter	Value
VGO	MR_{GG} $(-)$	K_p	98.5
		T_i	36.3
		T_d	3.56×10^{-4}
VGH	p_{cc} (Pa)	K_p	2.59×10^{-7}
		T_i	1.22
		T_d	6.82×10^{-3}
VGC	MR_{PI} $(-)$	K_p	0.786
		T_i	1.06
		T_d	2.12×10^{-2}

The settling time is not the only reason for a large error in the combustion chamber pressure in the case of lower turbine efficiencies. For the lowest turbine efficiencies, a combustion chamber pressure of 100 bar is physically no longer possible while maintaining the other constraints (especially the desired gas-generator mixture ratio). A specific disadvantage of PID controllers is that degenerating efficiencies or other system parameters cannot be considered directly as further input variables. Fig. 6 shows that for the 80 bar startup, VGC oscillates a little. It is challenging to tune a single family of PID controllers for different reference combustion chamber pressures. For even lower combustion chamber pressures (deep throttling), it becomes more and more difficult to achieve a convincing performance for all operating conditions. The prevention of oscillations leads to an increased settling time for all reference values. All in all, the performance of the PID controllers is not perfect but satisfactory for the case of 100 and 80 bar and fixed mixture ratios.

Now we examine the performance of our RL approach. The comparison of Figs. 3 and 4 shows that at first glance, the RL agent's behavior shows strong similarities to the PID controllers. The flow control valves are opened in a nonmonotonic way. Nevertheless, the agent can guarantee an even faster startup, as presented in Fig. 5. The RL controller can better control the combustion chamber pressure and the global mixing ratio. The control of the gas-generator mixture ratio is comparatively good. Furthermore, the RL agent can directly take the firing of the turbine starter into account. The action changes at $t = 2.6$ s, which is the time when the firing of turbine starter stops. Similar to the PID controllers, the RL agent can handle degrading turbine efficiencies to a certain extent. It can detect deviating efficiencies because the relationship between valve positions and controlled variables changes, and adjusts the startup. A prerequisite for this is that the valve positions are also included in the observation space, and that experiences with different efficiencies were generated during the training.

Fig. 5 shows an overshoot in the combustion chamber pressure as well as in the global mixing ratio with both the PID controllers and the RL agent. This characteristic behavior also occurs when an MPC approach is used [25]. Thus, the causes might be system-related.

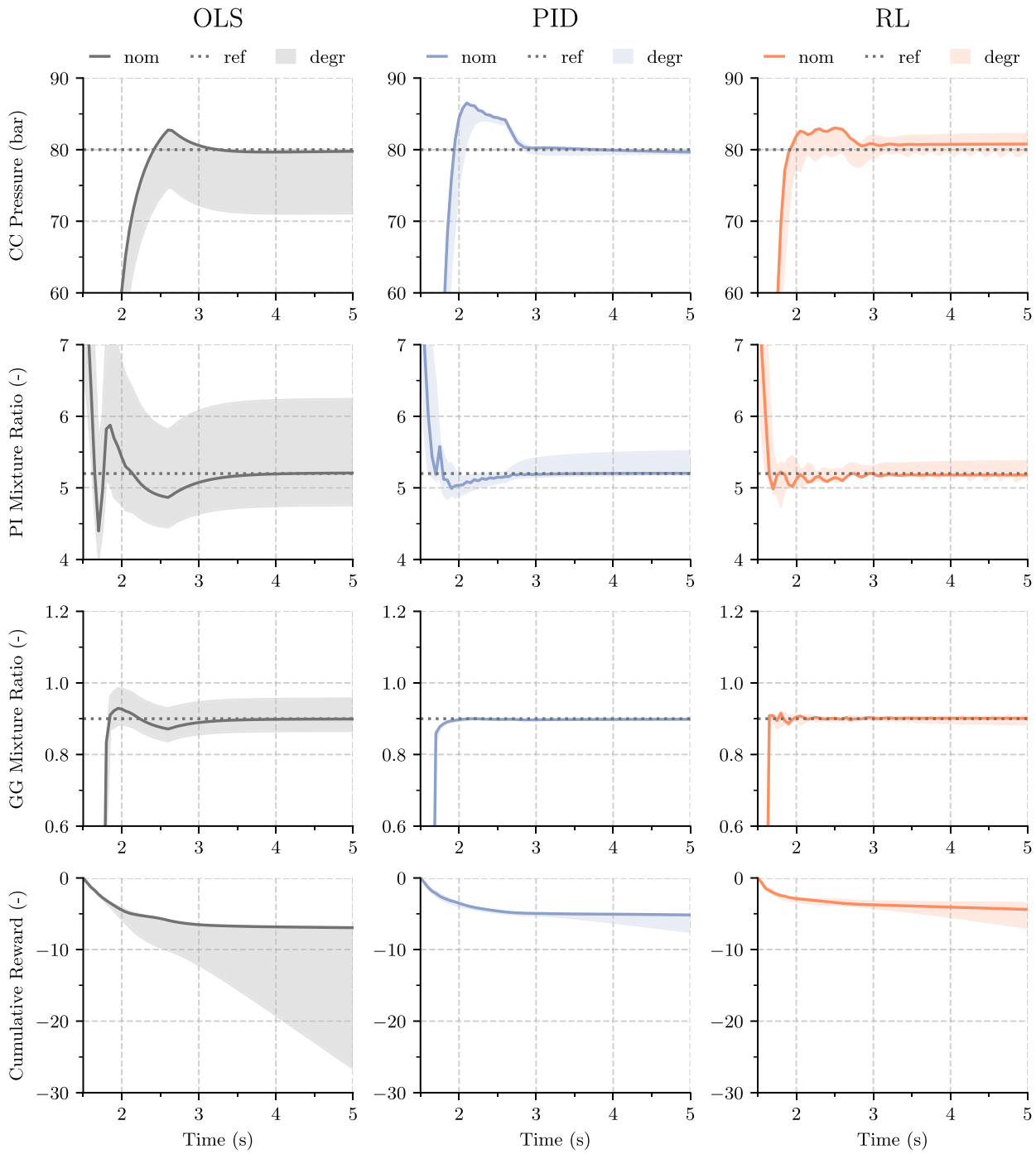

Fig. 8. Comparison of the controlled variables for the 80 bar startup. Shaded area marks the range of the controlled variable for different degraded efficiencies. At different turbine efficiencies, the standard open-loop sequence provides significantly different steady-state values for the chamber pressure and the mixture ratios.

Table I compares the rewards, steady-state values, and IAEs of the studied approaches for nominal turbine efficiencies and both main combustion chamber pressures of 100 and 80 bar. The open-loop sequences are satisfying for the nominal startups. Nevertheless, both *IAE*s and rewards show that improvement is possible. One can start up faster if the valves are opened nonmonotonously. Why is this not done for realistic startup sequences? As already mentioned, it is common practice to determine the control sequences employing tests on test benches, which is expensive and time-consuming. With nonreusable engines, the demands on the control system are not so dramatic, and one can accept good but not optimal sequences as long as a large amount of development costs is saved. Another reason is that, as a rule, disturbances influence the startup anyway and cancel out the advantages of optimized sequences. The advantages can only be realized by closing the control loop. The tuned PID controllers are better than the open-loop

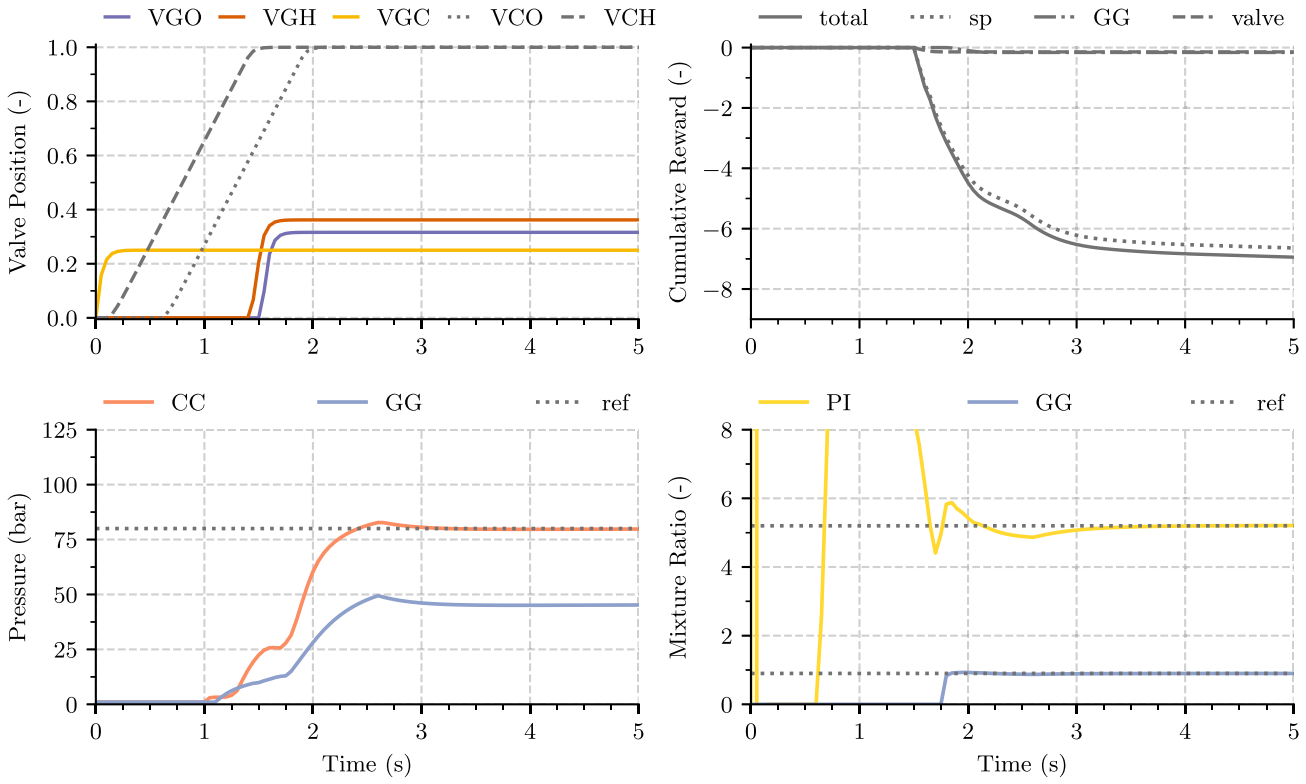

Fig. 9. 80 bar Nominal startup sequence. The main combustion chamber pressure settles at 80 bar, while the gas-generator pressure reaches 45 bar. The reference mixture ratios are given by 5.2 and 0.9. The engine reaches steady-state operating conditions after approximately 4 s. The cumulative reward for the OLS is dominated by the set-point error r_{sp}.

sequences concerning the value of the reward. The RL agent is even better. The RL agent and the PID controllers also achieve decent steady-state values.

Table II compares the rewards, steady-state values, and *IAE*s of the studied approaches for degrading turbine efficiencies. We present the mean and standard deviation of the measures instead of giving all values for the 16 different combinations of turbine efficiencies. For the steady-state values, the minimum and maximum values are also listed in Table II. As already seen in Fig. 5, the OLS results in large deviations for degrading turbine efficiencies. For 100 bar, the steady-state main combustion chamber pressure ranges between 92 and 100 bar. Furthermore, degrading turbine efficiencies strongly influence the overall mixture ratio MR_{PI}. Large deviations in MR_{PI} (here from 4.9 to 6.0) poses two major problems. First, the fuel and oxidizer tank volumes are designed for the nominal mixture ratio. Deviations in MR_{PI} result in a nonideal utilization of the propellants, thus lowering the launcher's performance. Second, the mixture ratio in the main combustion chamber is directly affected by the overall mixture ratio, potentially resulting in more damaging conditions for the main combustion chamber. The cumulative reward for the OLS increases to a mean value of -14.9 with a large standard deviation of 5.2.

The controller performances of both closed-loop controllers highlight the benefits of closed-loop control for degrading turbine efficiencies. The mean and standard deviations of the cumulative rewards are much smaller for the PID controllers and the RL agent. The additional reduction for the RL agent is mainly due to an even faster startup. The mean steady-state value of p_{CC} is given by 96.0 bar for the agent, which is a little bit closer to 100.0 bar than the value of PID controllers and much closer than the value of the open-loop sequence. Furthermore, the maximum deviation is the smallest. The advantages of closed-loop control and especially the RL approach are also reflected in the mixture ratios, which are much closer to their nominal values compared to the OLS. The *IAE*s also show that the RL agent performs better than the PID controllers.

VI. CONCLUSION

In this article, we presented an RL approach for the optimal control of the fully continuous phase of the startup of a gas-generator cycle liquid rocket engine. Using a suitable engine simulator, we employed the TD3 algorithm to learn an optimal policy. The policy achieves the best performance compared with carefully tuned open-loop sequences and PID controllers for different reference states and varying turbine efficiencies. Furthermore, the prediction of the control action takes only 0.7 ms, which allows a high interaction frequency, and, in comparison to MPC, enables the real-time use of RL algorithms for closed-loop control. The modest computational requirements should be met by the current generation of engine control units. A potential drawback of

the RL approach is the lack of stability guarantees. Nevertheless, the control system can be tested using a high-fidelity simulation model, and there is ongoing work on certifying stability of RL policies [36].

The present article can be improved in many directions. It is necessary to carefully examine the performance of the controller when various disturbances occur. Disturbance rejection and state estimation techniques will be the focus of future work. Furthermore, even the most sophisticated models usually have prediction errors due to not included effects or model miss-specifications. Therefore, it is essential to ensure that controllers trained in a simulation environment are robust enough to be used in real applications. There are RL approaches that explicitly consider modeling errors. Domain randomization [48] can produce agents that generalize well to a wide range of environments. Another issue with RL is implementing hard state constraints. Using the example of liquid rocket engine control, one would like to impose hard constraints to limit the maximum rotational speed of the turbopumps and maximum temperatures to prevent damage to the engine. It is possible to approximate hard state constraints by carefully tuning the reward function, e.g., one can give the agent a sizeable negative reward upon constraint violation and possibly terminate the training episode. Besides, there has been recent work on implementing hard constraints in RL using constrained policy optimization [49].

We would like to conclude this publication with an outlook on the potential advantages of this approach for rocket engine control. Controllers trained with RL can depend on many input variables, can be used for very different operating conditions, and can include multiple objectives. The thrust control of rocket engines is crucial for improving the performance of the launch vehicle, but it is particularly critical when using rocket engines for the soft landing of returning rocket stages. Deep throttling domains of an engine, i.e., 25%–100% range of nominal thrust, are not supposed to pose a problem for RL controllers. Regarding multiple objectives, one can modify the reward function to optimize both the system's performance and damage mitigation [50]. The coupling of sophisticated health-monitoring systems, possibly based on machine learning techniques, with suitable policies trained by RL, can increase the reliability of launch systems further. Given a suitable simulation environment, end-to-end RL may even enable the training of integrated flight and engine control systems. Overall, it is hoped that the current work will serve as a basis for future studies regarding the application of RL in the field of rocket engine control.

ACKNOWLEDGMENT

The authors would like to thank W. Kitsche, Robson Dos Santos Hahn, and M. Börner for valuable discussions concerning the startup of a gas-generator cycle liquid rocket engine.

APPENDIX A
IMPLEMENTATION AND TRAINING DETAILS

The agent is trained for 100 000 time steps, which is equal to approximately 1.5 h of simulation time. The agents' hyperparameters are tuned with Optuna [51] and are presented in Table III. For exploration, we use action noise sampled from an Ornstein-Uhlenbeck process [52]. Table IV shows the parameters of all three PID controllers and the corresponding controlled variable and control valve.

APPENDIX B
PLOTS FOR 80 BAR CASE

Fig. 9 shows the nominal OLS for a main combustion chamber pressure of 80 bar. The manipulated valve positions by the PID and RL agent for 80 bar are shown in Figs. 6 and 7. Finally, Fig. 8 compares controller performances for different degraded turbine efficiencies. The figure also reveals that the performance of the RL agent might not be optimal as can be seen by the slightly oscillating PI mixture ratio. These oscillations probably have no physical background, but are probably an artifact of a suboptimal training process. The instability of the training process, its sensitivity to hyperparameters, and general reproducibility issues are still open problems that are heavily discussed in the RL community [53].

REFERENCES

[1] S. Colas, S. L. Gonidec, P. Saunois, M. Ganet, A. Remy, and V. Leboeuf
A point of view about the control of a reusable engine cluster
In *Proc. 8th Eur. Conf. Aeronaut. Space Sci.*, Madrid, Spain, 2019.

[2] L. Meland and F. Thompson
History of the titan liquid rocket engines
In *Proc. 25th Joint Propulsion Conf.*Sacramento, CA, USA: American Institute of Aeronautics and Astronautics, 1989, Art. no. 2389.

[3] M. Lausten, D. Rousar, and S. Buccella
Carbon deposition with LOX/RP-1 propellants
In *Proc. 21st Joint Propulsion Conf.* Monterey, CA, USA: American Institute of Aeronautics and Astronautics, Jul. 1985, Art. no. 1164.

[4] J. A. B. Bossard, W. M. Burkhardt, K. Y. Niiya, and F. Bram
Effect of propellant flowrate and purity on carbon deposition in LO2/methane gas generators
The 1989 JANNAF Propulsion Meeting, vol. 1, 1989.

[5] R. J. Roelke
Miscellaneous losses
Turbine Design Appl., vol. 2, Jan. 1973.

[6] M. Hampson
Reusable rocket engine turbopump condition monitoring
In *Proc. Space Syst. Technol.*, Tech. Rep. 841619, 1984.

[7] C. F. Lorenzo and J. L. Musgrave
Overview of rocket engine control
In *Proc. AIP Conf.*, Albuquerque, NM, USA, 1992, pp. 446–455.

[8] S. Pérez-Roca *et al.*
A survey of automatic control methods for liquid-propellant rocket engines
Prog. Aerosp. Sci., vol. 107, pp. 63–84, May 2019.

[9] J.-N. Chopinet *et al.*
Progress of the development of an all-electric control system of a rocket engine
In *Proc. 48th AIAA/ASME/SAE/ASEE Joint Propulsion Conf. Exhibit*, Atlanta, GA, USA, Jul. 2012, Art. no. 3873.

[10] A. Iannetti
Prometheus, a LOX/LCH4 reusable rocket engine
In *Proc. 7th Eur. Conf. Aeronaut. Space Sci.*, Milano, Italy, Jul. 2017, pp. 3–6.

[11] H. Asakawa, M. Tanaka, T. Takaki, and K. Higashi
Component tests of a LOX/methane full-expander cycle rocket engine: Electrically actuated valve
In *Proc. 8th Eur. Conf. Aeronaut. Space Sci.*, Madrid, Spain, 2019.

[12] K. Dresia *et al.*
Multidisciplinary design optimization of reusable launch vehicles for different propellants and objectives
AIAA J. Spacecraft Rockets, Feb. 2021, p. 13, doi: 10.2514/1.A34944.

[13] V. G. Lopez and F. L. Lewis
Dynamic multiobjective control for continuous-time systems using reinforcement learning
IEEE Trans. Autom. Control, vol. 64, no. 7, pp. 2869–2874, Jul. 2019.

[14] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis
Optimal and autonomous control using reinforcement learning: A survey
IEEE Trans. Neural Netw. Learn. Syst., vol. 29, no. 6, pp. 2042–2062, Jun. 2018.

[15] S. Gu, E. Holly, T. Lillicrap, and S. Levine
Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates
Int. Conf. Robot. Autom. (ICRA), 2017, pp. 3389–3396, doi: 10.1109/ICRA.2017.7989385.

[16] Z. Yang, K. Merrick, L. Jin, and H. A. Abbass
Hierarchical deep reinforcement learning for continuous action control
IEEE Trans. Neural Netw. Learn. Syst., vol. 29, no. 11, pp. 5174–5184, Nov. 2018.

[17] M. Mahmud, M. S. Kaiser, A. Hussain, and S. Vassanelli
Applications of deep learning and reinforcement learning to biological data
IEEE Trans. Neural Netw. Learn. Syst., vol. 29, no. 6, pp. 2063–2079, Jun. 2018.

[18] S. Heyer, D. Kroezen, and E.-J. Van Kampen
Online adaptive incremental reinforcement learning flight control for a CS-25 class aircraft
In *Amer. Inst. Aeronaut. Astronaut. Scitech Forum*, Orlando, FL, USA, Jan. 2020, Art. no. 1844.

[19] B. Gaudet, R. Linares, and R. Furfaro
Deep reinforcement learning for six degree-of-freedom planetary landing
Adv. Space Res., vol. 65, no. 7, pp. 1723–1741, Apr. 2020.

[20] S. Spielberg, R. Gopaluni, and P. Loewen
Deep reinforcement learning approaches for process control
In *Proc. 6th Int. Symp. Adv. Control Ind. Process*. Taipei, Taiwan: IEEE, May 2017, pp. 201–206.

[21] L. Cheng, Z. Wang, and F. Jiang
Real-time control for fuel-optimal moon landing based on an interactive deep reinforcement learning algorithm
Astrodynamics, vol. 3, no. 4, pp. 375–386, Dec. 2019.

[22] K. Hovell and S. Ulrich
On deep reinforcement learning for spacecraft guidance
In *AIAA Scitech Forum*. Orlando, FL, USA: American Institute of Aeronautics and Astronautics, Jan. 2020, Art. no. 1600.

[23] J. L. Musgrave and D. E. Paxson
A demonstration of an intelligent control system for a reusable rocket engine
NASA Tech. Memorandum, Jun. 1992, Art. no. 105794.

[24] E. Nemeth
Reusable rocket engine intelligent control system framework design
phase 2, NASA Contractor Report 187213, Sep. 1991.

[25] S. Pérez-Roca *et al.*
An MPC approach to transient control of liquid-propellant rocket engines
IFAC-PapersOnLine, vol. 52, no. 12, pp. 268–273, 2019.

[26] S. Perez-Roca *et al.*
Derivation and analysis of a state-space model for transient control of liquid-propellant rocket engines
In *Proc. 9th Int. Conf. Mech. Aerosp. Eng.*, Jul. 2018, pp. 58–67.

[27] R. S. Sutton and A. G. Barto
Reinforcement Learning: An Introduction, Ser. Adaptive Computation and Machine Learning Series. Cambridge, MA, USA: The MIT Press, 2018.

[28] D. P. Bertsekas
Reinforcement learning and optimal control. Belmont, MA, USA: Athena Scientific, 2019.

[29] L. Buşoniu, T. de Bruin, D. Tolić, J. Kober, and I. Palunko
Reinforcement learning for control: Performance, stability, and deep approximators
Annu. Rev. Control, vol. 46, pp. 8–28, 2018.

[30] T. P. Lillicrap *et al.*
Continuous control with deep reinforcement learning
Jul. 2019, *arXiv:1509.02971*.

[31] S. Fujimoto, H. van Hoof, and D. Meger
Addressing function approximation error in actor-critic methods
Oct. 2018, *arXiv:1802.09477*.

[32] X. Wang, Y. Gu, Y. Cheng, A. Liu, and C. L. P. Chen
Approximate policy-based accelerated deep reinforcement learning
IEEE Trans. Neural Netw. Learn. Syst., vol. 31, no. 6, pp. 1820–1830, Jun. 2020.

[33] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine
Reinforcement learning with deep energy-based policies
in *Proc. 34th Inter. Conf. Mach. Learn.*, vol. 70, 2017, pp. 1352–1361.

[34] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel
Trust region policy optimization
in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 1352–1361.

[35] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov
Proximal policy optimization algorithms
Aug. 2017, *arXiv:1707.06347*.

[36] M. Jin and J. Lavaei
Stability-certified reinforcement learning: A control-theoretic perspective
in *IEEE Access*, vol. 8, pp. 229086–229100, 2020, doi: 10.1109/ACCESS.2020.3045114.

[37] J. Vilá, J. Moral, V. Fernández-Villacé, and J. Steelant
An overview of the ESPSS libraries: Latest developments and future
In *Proc. Space Propulsion Conf.*, Seville, Spain, 2018.

[38] G. Waxenegger-Wilfing, K. Dresia, M. Oschwald, and K. Schilling
Hardware-in-the-loop tests of complex control software for rocket propulsion systems
In *Proc. Int. Astronautical Congr. IAC, Virtual Event*, 2020.

[39] A. Hill *et al.*
Stable baselines
2018. [Online]. Available: https://github.com/hill-a/stable-baselines

[40] G. Waxenegger-Wilfing, K. Dresia, J. C. Deeken, and M. Oschwald
Heat transfer prediction for methane in regenerative cooling channels with neural networks
J. Thermophysics Heat Transfer, vol. 34, no. 2, pp. 347–357, Apr. 2020.

[41] K. Dresia, G. Waxenegger-Wilfing, J. Riccius, J. Deeken, and M. Oschwald
Numerically efficient fatigue life prediction of rocket combustion chambers using artificial neural networks
In *Proc. 8th Eur. Conf. Aeronaut. Space Sci.*, Madreid, Spain, 2019.

[42] A. Iffly and M. Brixhe
Performance model of the vulcain ariane 5 main engine
In *Proc. 35th Joint Propulsion Conf. Exhibit*. Los Angeles, CA, USA: American Institute of Aeronautics and Astronautics, Jun. 1999.

[43] R. Ryan and L. Gross
Effects of geometry and materials on low cycle fatigue life of turbine blades in LOX/hydrogen rocket engines
In *Proc. 22nd Joint Propulsion Conf.* Salt Lake City, UT, USA: American Institute of Aeronautics and Astronautics, 1986, Art. no. 1443.

[44] P. Pempie, T. Froehlich, and H. Vernin
LOX/methane and LOX/kerosene high thrust engine trade-off
In *Proc. 37th Joint Propulsion Conf. Exhibit. Salt Lake City*, UT, USA: American Institute of Aeronautics and Astronautics, Jul. 2001, Art. no. 3542.

[45] M. Riedmiller
10 Steps and Some Tricks to Set up Neural Reinforcement Controllers
In *Neural Networks: Tricks of the Trade*, G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, vol. 7700, pp. 735–757.

[46] P. Cominos and N. Munro
PID controllers: Recent tuning methods and design to specification
IEE Proc. - Control Theory Appl., vol. 149, no. 1, pp. 46–53, Jan. 2002.

[47] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné
DEAP: Evolutionary algorithms made easy
J. Mach. Learn. Res., vol. 13, pp. 2171–2175, Jul. 2012.

[48] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel
Domain randomization for transferring deep neural networks from simulation to the real world
in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2017, pp. 23–30.

[49] J. Achiam, D. Held, A. Tamar, and P. Abbeel
Constrained policy optimization
in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, pp. 22–31, 2017.

[50] A. Ray, X. Dai, M.-K. Wu, M. Carpino, and C. F. Lorenzo
Damage-mitigating control of a reusable rocket engine
J. Propulsion Power, vol. 10, no. 2, pp. 225–234, Mar. 1994.

[51] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama
Optuna: A next-generation hyperparameter optimization framework
In *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 2623–2631.

[52] G. E. Uhlenbeck and L. S. Ornstein
On the theory of the brownian motion
Phys. Rev., vol. 36, no. 5, pp. 823–841, Sep. 1930.

[53] R. Islam, P. Henderson, M. Gomrokchi, and D. Precup
Reproducibility of benchmarked deep reinforcement learning tasks for continuous control
2017, *arXiv:1707.06347 arXiv:1708.04133*.

Günther Waxenegger-Wilfing received the Ph.D. degree in theoretical physics from the University of Vienna, Vienna, Austria, in 2015.

He is a Senior Research Scientist for intelligent engine control with the German Aerospace Center (DLR) Institute of Space Propulsion, Hardthausen, Germany, and a Lecturer with the University of Würzburg, Würzburg, Germany. As part of the DLR project AMADEUS, he manages the investigation of the use of artificial intelligence in space transportation. He previously was the lead Quantitative Analyst with Nova Portfolio Vermögens Management, Innsbruck, Austria. His research interests include deep learning for control and condition monitoring, optimal control, model-free as well as model-based reinforcement learning, and applications in autonomous launch vehicles and landers.

Kai Dresia received the master's degree in aerospace engineering from RWTH Aachen University, Aachen, Germany. He is currently working toward the Ph.D. degree at the German Aerospace Center (DLR), Institute of Space Propulsion, Hardthausen, Germany, in 2018.

His research interests include combining machine learning with physics-based modeling for rocket engine control and condition monitoring.

Jan Deeken received the Ph.D. degree in aerospace engineering from the University of Stuttgart, Stuttgart, Germany, in 2014.

He is currently Head of the Rocket Engine System Analysis Group, Department of Rocket Propulsion, DLR Institute of Space Propulsion, Lampoldshausen, Germany. He is also managing the DLR LUMEN Project, which aims at developing and operating a LOX/LNG expander-bleed engine testbed in the 25 kN thrust class.

Michael Oschwald received the Ph.D. degree in physics from the University of Freiburg, Freiburg, Germany, in 1987. He is a Professor of Space Propulsion with RWTH Aachen University, Aachen, Germany, and Head of the Department of Rocket Propulsion, Institute of Space Propulsion at the German Aerospace Center (DLR), Lampoldshausen, Germany. His research interests include all aspects of rocket engine design and operation with a specific focus on cryogenic propulsion. He has developed numerical tools to predict the behavior of rocket engines and their components.