DOI: 10.13009/EUCASS2023-682
Aerospace Europe Conference 2023 – 10TH EUCASS – 9TH CEAS

Optimising ConOps of Water Electrolysis Propulsion systems using Deep Q-Learning: A Proof-of-Concept

Sascha Dengler†, Adrian Wagmann Y Otero* and Chiara Manfletti**
** Technical University of Munich*
Chair of Space Propulsion and Mobility
Lise-Meitner-Str. 9, 85521 Ottobrunn, Germany
Sascha.Dengler@tum.de
† Corresponding Author

Abstract
This work explores the use of deep reinforcement learning to derive and optimise the Concept of Operations (ConOps) for Water Electrolysis Propulsion (WEP) systems. WEP systems produce propellants on demand by decomposing liquid water into gaseous hydrogen and oxygen using an electrolyser. The added complexity of the propellant production must be considered for manoeuvre planning as well as for control and operation of the components themselves. An approach to optimising this ConOps using a Deep-Q-Network that controls a simulated spacecraft propelled by a WEP system is derived. The framework developed for this purpose is described and a Proof-of-Concept (PoC) on a reference mission scenario is conducted.

1. Motivation

1.1 Introduction to Water Electrolysis Propulsion

In the pursuit for high-performance green propellants, Water Electrolysis Propulsion is one of the most promising technologies. A spacecraft filled with pure, liquid water guarantees easy handling during ground operations and potentially reduces cost compared to highly-toxic alternatives. Once in orbit, the water is decomposed into GH2 and GO2 by an electrolyser that is powered by solar cells. When being used in a chemical thruster, this propellant combination provides a high theoretical specific Impulse (Isp) of ca. 400 seconds and has the potential to outperform other storable chemical propulsion solutions. For certain mission profiles, it could be shown that WEP can be more mass-efficient than existing chemical propulsion solutions [1]. Water availability on other celestial bodies such as asteroids, the Moon, and planetary bodies, and the possibility of mining this resource and refilling spacecraft in-situ further increases the attractiveness of water-based propulsion solutions.

These advantages however come at the cost of added complexity due to the on-board propellant production and intermediate storage. The time needed to produce a required amount of propellant must be considered and planned for and a suitable supply of electrical power needs to be ensured. This process should make optimal use of the individual component's operational envelopes, as well as their interplay with each other and with other subsystems of the spacecraft, to achieve high systemic efficiency. Additionally, this more complex operation of the propulsion system should not fall back to the operator of the spacecraft. Both aspects could be an inhibitor to a broader use of the technology, despite the performance advantages mentioned.

To tackle these challenges, an automated optimisation process for the Concept of Operation (ConOps) of WEP systems is needed. Such an optimiser should be capable of providing an ideal scheduling of production and manoeuvres as well as an optimal control approach within these processes. Ideally, a level of adaptability of the operations to changing environmental factors and self-diagnosis capabilities are present to enable autonomous functionalities at a later stage, further simplifying operations and improving usability. Reviewing possible solution approaches, reinforcement learning (RL) provides promising advances in optimal control problems in recent years. To pursue a ConOps optimiser based on RL with the aforementioned characteristics, a framework is implemented and described in this work. Making use of this framework, an initial Proof-of-Concept (PoC) on a simple application scenario is achieved and limitations found are discussed. Finally, an outlook on next steps and topics that emerged from this work is given.

DOI: 10.13009/EUCASS2023-682

1.2 WEP components and ConOps

As mentioned before, the WEP technology revolves around bringing water as an inert substance into space and subsequently generating combustible propellants as needed using an electrolyser supplied with electrical power. This categorises the system as an electrochemical propulsion system, possessing characteristics of both purely chemical and electrical propulsion.
A schematic overview of such a system with its components, in this case with a dual-mode thruster setup with hot gas and cold gas capabilities, is shown in figure 1.

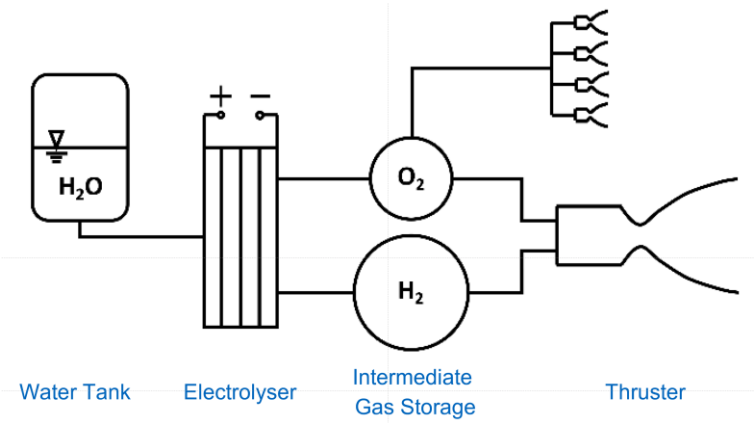

Figure 1: Schematic overview of Water Electrolysis Propulsion.

A water tank stores purified water at low pressure levels and feeds it to an electrolyser. The electrolyser is a central novelty of a WEP system and introduces its own operational envelope. Space-capable electrolyser concepts such as Proton-Exchange-Membrane (PEM) in a Cathode-Vapour-Feed (CVF) variant have different performance characteristics compared to terrestrial electrolysers. An example of such an electrolyser and the respective characteristic envelope (figure 2) is described in [2], while research w.r.t. the transient behaviour is still ongoing in the field.
For the operation of the electrolyser a certain voltage and current is needed. While a sufficient voltage is required to decompose the water molecule, the current is defining the gas production rate. The relation between the two parameters is heavily dependent on the temperatures within the component and consequently driving the efficiency. Furthermore, not only efficiency is affected but also the purity of the gases produced (humidity and gas cross-over). Especially when the spacecraft operation prescribes an intermitting operation pattern, the thermal transient behaviour needs to be considered and optimised for.
Another aspect is the pressure build-up in the gas plenums at the outlet of the electrolyser. A major advantage of the PEM electrolyser technology is that it can pressurise the gases by itself, eliminating the need for additional active components such as pumps. However, with increasing backpressure a decrease in current density and thus gas production rate can be observed [3].
Electrolysers can be operated either voltage or current-controlled. Additionally, they may have the option to operate one or multiple cells independently and on demand, effectively controlling the active area available. These parameters can consequently be controlled by a power-conditioning unit, that needs an operational strategy. The goal of the strategy can be described to produce a sufficient quantity of propellants with a respective quality/purity for a given electrical power availability and time schedule. Furthermore, when flexibility w.r.t. the time and location of manoeuvres is given within the mission profile, this can be optimised with accounting for the electrolyser's envelope.

DOI: 10.13009/EUCASS2023-682

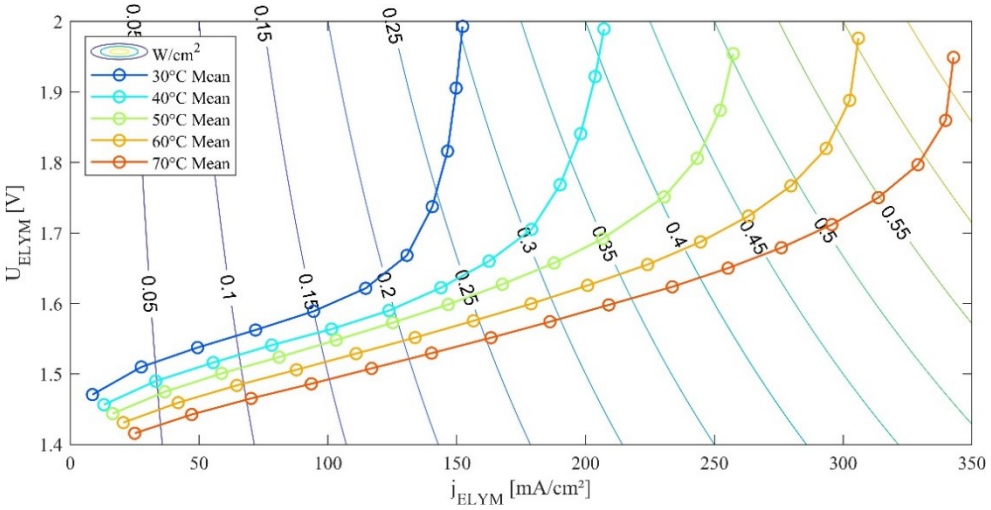

Figure 2: Exemplary characteristic envelope of a CVF Electrolyser [2].

The intermediate gas storage tanks are fundamentally needed to account for the comparatively low mass flow output from the electrolyser, which makes it challenging to downscale a continuously operated hot-gas thruster. The size and storage pressure levels of these tanks need to be determined in a design optimisation which must also account for the operational concept of the system.

When considering a hot gas thruster, one of the central challenges is the thermal management of stoichiometric combustion in the absence of cryogenic cooling agents. Most concepts currently in development combine a film cooling approach with the thermal capacity of the thruster material [4, 5]. Due to the transient heat-up phase of the material, there is an initial phase of reduced efficiency in terms of specific impulse (Isp). Furthermore, depending on the materials used, the maximal burn time can be limited. These factors must be considered for optimal operation and are closely related to the mass and pressure of propellants stored in the intermediate tanks.

For an optimal operation of a WEP system, not only the components and their interaction within the propulsion subsystem needs to be considered, but also it being embedded in a spacecraft. The main elements are thermal interaction in the form of heat conduction, availability of electrical power and imposed scheduling due to e.g. payload operation. Due to the hybrid character of the electrochemical system, the manoeuvre dynamics of the orbital motion are in-between large impulsive manoeuvres of purely chemical propulsion and continuous thrusting/spiralling of electrical systems. Since storage capability of the gaseous propellants comes at the expense of heavy tanks, for most mission scenarios an intermittent operation with regular charging and thrusting cycles is favourable. If flexibility w.r.t. the mission schedule is given, an optimisation trade-off is to be made between manoeuvre scheduling, storage capacity and electrolysis power.

1.3 Reference scenario

To approach a confirmable PoC of an RL optimiser for WEP, an initial application scenario that has a known optimal and interpretable operation strategy is needed as a reference. Such a scenario is described in a previous work [6], in which the feasibility of using a WEP system as a de-orbiting kit for CubeSats is studied. The reference manoeuvre within this study was a decrease of perigee to 200 Km from an initially circular 500 Km orbit of a 3U CubeSat. At perigee altitudes as low as 200 Km, a passive atmospheric re-entry is initiated. The spacecraft parameter and orbital elements of this scenario are displayed in tables 1 and 2.

DOI: 10.13009/EUCASS2023-682

Table 1: Reference mission parameters

Parameter	Initial value	Target value
Perigee altitude [km]	500	200
Semi major axis [km]	500	350
Eccentricity [-]	0.0	0.0223
Argument of periapsis [rad]	0.0	0.0
Inclination [rad]	0.0	0.0
Right ascension of the ascending node [rad]	0.0	0.0

Table 2: Reference spacecraft and propulsion parameters

Parameter	Value
Spacecraft dry mass [kg]	5
Power availability [W]	10
Propellant (water) mass [kg]	0.2
Volume H2 tank [ml]	100
Volume O2 tank [ml]	50
Thruster mass flow [g/s]	0.5

The perigee decrease is best carried out as a series of consecutive thruster burns at every apogee pass, with propellant being produced by electrolysis in between. This ensures that the manoeuvre is completed in the smallest possible timeframe at a minimum required delta-v.
With this relatively simple operational pattern, the performance of an algorithm can later be evaluated and compared. When the respective incentive (to reach the target orbit as fast as possible) is set as an objective through the reward function, a similar or identical operational pattern should be observable when a global optimum is found.

2 Optimal control and reinforcement learning

The optimisation task at hand is an optimal control problem: A dynamic system with a sequential control or decision order that needs to minimise or maximise a to-be-defined objective or reward function. Solution approaches on such a setting can be derived from optimal control theory or reinforcement learning. Neglecting an optimisation, a feasible control approach would also be a rule-based control strategy derived from simple relations, providing a robust setup but not making use of the full potential of the system.
In optimal control, a solution that minimises an objective function is found by mathematically exploiting a model of the dynamic system. Thus, optimal control approaches require knowledge of a model of the dynamic system. The optimised strategy is purely dependent on the model and can only be as good as the approximation of the real system is through the model. However, the model can be adapted or re-evaluated within an ongoing control loop accounting for deviations from the idealised expected behaviour.

DOI: 10.13009/EUCASS2023-682

Reinforcement learning (RL) on the other hand finds optimal actions that maximise a reward function by interacting with an environment and learning from this feedback. A model of the system is not required but inherently learned through the trial-and-error approach. In RL terminology, an Agent is trained to take an Action a in an environment from which it receives information about the current State s of the system and a Reward r, which evaluates the optimality of this current state (see figure 3). Based on the action taken, the environment transitions from the current time t to a successive time t+1. The rule, following which the agent acts is called policy π.

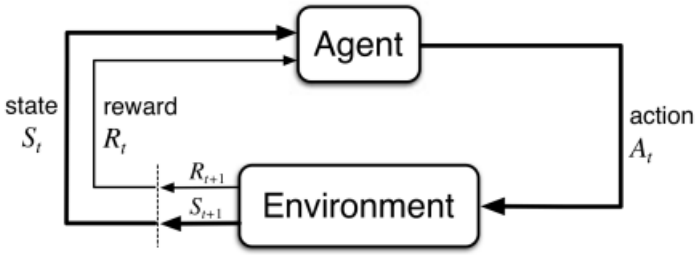

Figure 3: Reinforcement learning control schematic [7]

Choosing one of those two approaches is a non-trivial task, since both fields are subject to ongoing research and there is no clear preference for the given application. A comparative study would be needed to understand better the trade-offs imposed, however implementing and applying algorithms from both domains is time and resource intense. An example of such a comparative study, in this case in the field of industrial energy supply systems, can be found in [8]. Due to the characteristic of RL of learning on self-explored data and not needing to provide prior knowledge, the initial deployment of an RL solution is simpler as compared to optimal control theory. Furthermore, the same characteristic enables good adaptability to changes in system behaviour, which is the case when applying to a real, noisy and individually-variable system.
Thus, the feasibility of RL for the problem at hand should be demonstrated in an initial PoC to identify strengths and weak points in application to the spacecraft propulsion setting.

Within RL, available algorithms can be broadly categorised into Policy Optimisation and Q-Learning. In the first, free parameters of a given policy function are adapted to consequently find the optimal policy. In Q-Learning, the correlation between states, actions and received rewards is learned and exploited. Policy optimisation is better suited for continuous action spaces due to the function-based policy while Q-Learning is preferable for discrete action spaces. For the initial implementation of the spacecraft scenario, the action space can be sufficiently represented using discrete choices (see chapters 3.1 and 3.2). One promising algorithm from the Q-Learning subset is ‘DQN’, which is described in the following.

2.1 Deep-Q-Learning

In Q-Learning, a table or function is learned, that maps the states of an environment with their available actions to the expected cumulative reward of that couple. This is called a ‘Q’ table or function. In high-dimensional and complex environments, due to a high number of potential combinations, a look-up table or simple approximation function would not be sufficient anymore. Here, the attribute of neural networks (NNs) being universal function approximators comes into play. Approximating the Q-table using a NN (a so-called Deep-Q-Network, DQN) enables dealing with more complex environments by generalising features without requiring manual feature engineering. An exemplary Deep-Q-Network is depicted in figure 4.

DOI: 10.13009/EUCASS2023-682

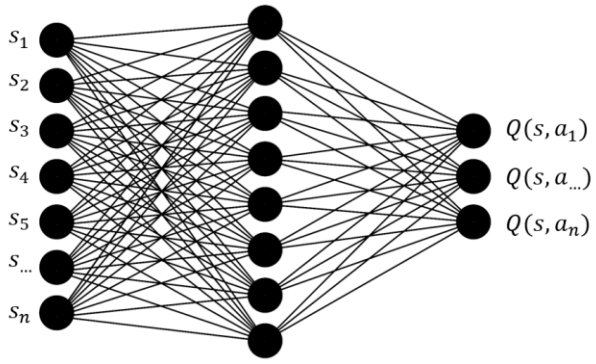

Figure 4: Deep-Q-Network schematic

The DQN needs to be trained on the environment to approximate the 'real' Q-function, which is initially unknown. Through iterative learning, the weights of the NN get updated based on the observed rewards and transitions between states. Once the Q-function is approximated, the optimal policy can be derived by choosing the action with the maximal value (thus the highest expected future reward) for any given state:

$$\pi(s) := arg\ max\ Q(s, a) \quad \textbf{(1)}$$

This algorithm was first proposed in [9] and introduced modifications to standard Q-Learning that improve convergence and stability and thus performance in the learning process:

Double Q-Learning:
In standard Q-learning, the Q-values for each action-state pair are updated based on the maximum Q-value of the following state. However, this can lead to overestimating Q-values, which can negatively impact the learning process. Double-Q learning addresses this issue by decoupling the action selection from the Q-value evaluation.
This is done by handling two sets of Q-networks: the policy network and the target network. During the learning process, the policy network is used to select the action, while the target network is used to evaluate the Q-value of that action. The policy network is continuously updated while leaning. The target network however receives updates at a lower rate by incrementally copying the weights of the policy NN. This rate of updating is determined by factor τ.

Memory replay:
Learning on consecutively experienced data that are temporally correlated can introduce bias and instability to the process. By storing experiences in a memory buffer of a certain capacity and randomly sampling batches from the buffer removes their correlation. The training at every timestep is then conducted on the batch. Using a batch instead of a single data point makes for a reuse of previously made experiences and leverage them to accelerate the overall learning process.

Epsilon-greedy exploration:
Acting according to the currently best policy is called exploitation. Exploitation can lead to a poor representation of the available state space and thus to early convergence in a local optimum. By enforcing exploration in the learning process by randomising the actions taken can mitigate that risk.
Epsilon-greedy exploration is a method used to balance between exploration and exploitation in RL by choosing actions randomly with a probability of epsilon (ε) and selecting the action with the highest Q-value with a probability of 1-ε. Initially, the exploration rate is typically set high (e.g., ε = 1) to encourage the agent to explore the environment and learn about different actions and states. Over time, the exploration rate is gradually decreased to shift the focus towards exploitation and selecting actions based on learned knowledge.

3. Framework

To investigate the suitability and performance of RL algorithms on the technical problem described, a flexible research framework is implemented. The requirements posed to this framework are:

(i) Modularity to allow for iterative adaptations and parallel research work.
(ii) Compliance with community standards to allow for simple verification and benchmarking.
(iii) Accessibility and traceability for continuous usage in research.

DOI: 10.13009/EUCASS2023-682

Following the above-mentioned requirements, the implementation should have a modular separation between the Agent and the Environment. This allows on one hand to interchange and test different Agents on the same Environment and on the other hand to extend one working solution to different Environments.
The Environment will initially be represented by a simulation/digital twin to provide a safe and controlled environment before progressively moving towards deploying to Hardware-in-the-loop (HIL) or actual prototypes.

To comply with this structure and requirements, the implementation uses OpenAI's open-source 'gymnasium' toolkit [10] in Python. This toolkit provides a programming interface between the Agent and Environment, which standardises interfacing between simulation and reinforcement learning. Commonly used in the community, the toolkit enables the comparison of different algorithms on a set of existing environments that are commonly used for benchmarking. Furthermore, new environments can be defined according to the interface. Consequently, a new environment is created specifically for a general representation of a spacecraft consisting of a WEP system and orbiting a given initial orbit.
Due to the wide use in the community, open-source packages are available that provide RL algorithms made for interfacing with the gymnasium environment. An example would be 'Stable-Baselines3' [11], which consists of a set of pre-implemented and verified deep reinforcement learning algorithms.

An overview of the framework architecture including the submodules of the simulation is depicted in figure 5.
In the following, a more detailed description of the building blocks is given.

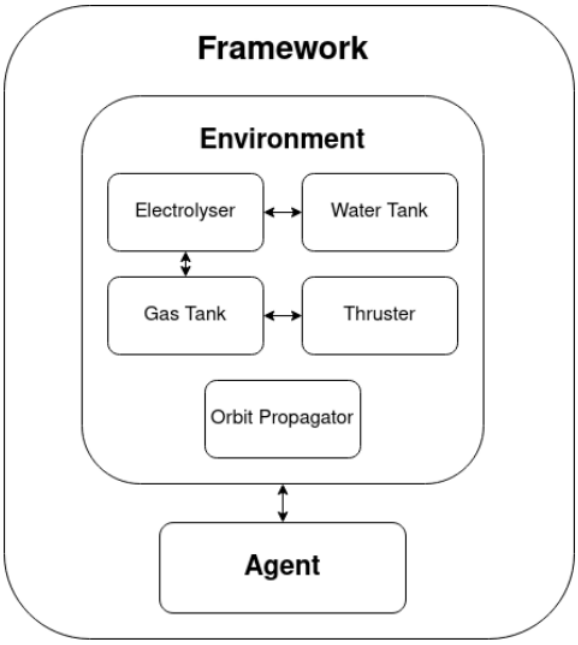

Figure 5: Framework implementation architecture.

3.1 Environment

The API of the gymnasium Environment strongly resembles the general setup in RL as per figure 3. Within a custom-defined environment, a State and Action space as well as a Reward needs to be defined for handover to the Agent. Furthermore, a function that transitions the Environment from one State to the next needs to be defined.

For the spacecraft propulsion scenario, the environment is represented by a simulation of a spacecraft and its orbital motion. The individual components' behaviour is described by means of differential equations. The state transition function corresponds to a stepwise numerical integration of these equations and the propagation of the orbital motion.

DOI: 10.13009/EUCASS2023-682

The implementation of the spacecraft environment follows the same principle of modularity: The spacecraft and its subsystems are hierarchically structured and allow for independent modifications or exchanges. E.g. for a better representation of individual components, external simulation programmes can be interfaced.
For the initial PoC however, the simulation is entirely implemented in Python with self-derived 'low-fidelity' models. While this limits transferability to the real system, it allows for a faster computation and learning process and thus a de-risk of the optimisation approach using DQN.
As shown in figure 5, the Environment is divided into 5 submodules, 4 of which correspond to components of the propulsion system integrated into an overarching spacecraft class and 1 module that propagates the spacecraft in orbit.
In the following, the modeling approach of each of the submodules is described:

Propagator

Following the requirements stated before, it is decided to use the open-source astrodynamics library 'Poliastro' [12] to provide the orbit propagation capabilities needed. Within the framework, the API provided is used to define the initial position and orbital elements of the spacecraft as well as the orbital elements of a target orbit. Furthermore, within the 'step' function of the environment, the position and change of orbit is propagated using Poliastro's Cowell propagation method. For the PoC, perturbations besides accelerations from propulsive manoeuvres are neglected and a simple two-body problem is assumed. The propulsive manoeuvres are carried out exclusively in retrograde direction with the thrust and total impulse being computed by the thruster model.

Spacecraft

Within this parent class, general parameters as well as the subsystems of the spacecraft are implemented and interfaced. For the initial PoC, this is at first limited to the propulsion subsystem. Other subsystems and their interplay, e.g. concerning thermals and power, will be introduced in the future. In this first iteration, the required globally defined parameters that are relevant to fully define the dynamics reduce to the dry mass of the satellite and the availability of electrical power.

WEP model

The main objective in modeling the WEP system for the initial PoC is to represent the essential characteristics of operating. For the given de-orbit scenario, this includes the backpressure effect on the electrolysis and the dependency between thruster burn time and Isp. These effects are modeled by simple means of empirical correlations. Stepwise numerical integration of the physical quantities is carried out following the 1st order forward Euler method.

The electrolyser is represented by an efficiency model that is dependent on the backpressure. While the theoretically maximal propellant mass flow per unit power P supplied can be derived from the standard enthalpy of formation ΔH_R^0, the efficiency η is modelled with a linear dependency on the back-pressure p_{back} with reference taken from [3]:

$$\dot{m}_{propellants} = \frac{P}{\Delta H_R^0}\eta \; ; with \; \eta = \begin{cases} 1 - 0.07 \cdot p_{back} & p_{back} \leq 10 \; bar \\ 0.3 & p_{back} > 10 \; bar \end{cases} \quad \textbf{(2)}$$

Charging of the intermediate gas storage tanks is assumed to be an isothermal process, while emptying over the course of a thrust event is assumed to be isentropic. Furthermore, it is assumed that gas temperatures equilibrate before starting the subsequent charging cycle.

Thruster modelling follows the conclusions drawn from a previously established simulation using EcosimPro [6] and are empirically correlated to the data points found. The Isp dependency on the burn time was slightly adjusted accounting for improvements in cooling and material selection. The burn time t_{burn} needed as input for the correlation is calculated from current pressure levels in the intermediate storage tanks.

$$I_{sp} = 350\left(\frac{1}{1+e^{-2.8 \cdot t_{burn}\left(\frac{350}{110}-1\right)}}\right) \quad \textbf{(3)}$$

DOI: 10.13009/EUCASS2023-682

3.2 Agent

With the Deep-Q-Learning algorithm as described in chapter 2.1, the implemented process results as following:

Algorithm 1 Q-Learning with memory replay, target network and ϵ-greedy policy.

Initialize policy network Q
Initialize target network Q^T with same the parameters as Q
Initialize replay buffer D
for N in range(Max Timesteps) **do**
  Observe state s_t
  Select action a_t
    Generate random number, n, between 0 and 1
    if k $\leq \epsilon$ **then** select random action
    else $a_t = \arg\max Q(s, a)$
  Execute one time step of simulation
  Observe next state s_{t+1} and reward r_{t+1}
  Store (s_t, a_t, r_{t+1}, s_{t+1}) in memory buffer D
  Sample a random batch of memories
  Compute TD Error:

$$\delta = \left(R_{t+1} + \gamma \max_a Q^T(s_{t+1}, a_{t+1}) - Q^{old}(s_t, a_t)\right)$$

  Update policy by minimizing Huber loss:

$$\mathcal{L} = \frac{1}{|B|} \sum \mathcal{L}(\delta)$$

  Update Target Network:

$$Q^T = \tau Q + (1 - \tau) Q^T$$

end for

What remains to be defined is the State space, Action space and Reward function for this particular application scenario.

State space

Choosing what to include in the state space is a non-trivial task. The State vector represents what the Agent can ‘see’ and would correspond to sensorial (measurement data) or observed (calculated from measurement) input in a real system. Providing too little information can lead to major influential factors not being correlated within the neural network and thus not being considered in the optimisation of the policy. E.g., if no information about the current orbit and position is provided, a suitable location and time to perform a manoeuvre cannot be determined. However, providing too much and potentially irrelevant information in the state vector can lead to overfitting and increased computational effort.
Furthermore, since the State vector corresponds to the input layer of the Q-network, the values provided should be within the same order of magnitude. Large variations can cause instability in the process of adjusting the weights of the NN in the learning process.
Thus for the given scenario, the orbital elements of reference (semi-major axis, eccentricity, argument of periapsis and true anomaly) are included in the input layer. The elements are normalised either by trigonometric functions or by dividing with a reference value. Additionally, the core metrics defining the state of the propulsion system (gas tank pressure and remaining water mass) are introduced and normalised.

DOI: 10.13009/EUCASS2023-682

Action space

Due to the discrete output space of the Deep-Q-Network, the respective available choices need to be discrete as well. While it is possible to discretise a continuous action space e.g. when controlling a valve position, a discrete action space is sufficient for the initial PoC. With the established modelling of the propulsion system, the following choices are relevant:

- Electrolyser ON or OFF
- Thruster ON or OFF

By excluding the possibility of both being active in the same timestep and adding the possibility of 'idling', meaning both components inactive, the resulting action space has three options:

- a1: Electrolyser ON, Thruster OFF
- a2: Electrolyser OFF, Thruster ON
- a3: Electrolyser OFF, Thruster OFF

While the third option is not of relevance in the de-orbit scenario, it is anyhow introduced to confirm that the agent is capable of learning to avoid certain actions.

For later iterations, the action space should correspond to the actual control variable. Concerning the electrolyser this would be current or voltage, either in a discretised action space or with a policy optimisation algorithm that can deal with continuous action spaces.

Reward function

The objective(s) of the optimisation problem are being defined by the reward function. It should be noted that the mathematical maximum of that function becomes the optimisation target of the learning process. That being said, defining this function in a way that the mathematical maximum is in line with the expected outcome is not necessarily trivial. Furthermore, to improve efficiency of the learning process in RL, the function should be tailored so that the agent is 'guided' towards the desired behaviour. That could e.g. be realised by a linearly increasing reward towards a goal instead of a step function when having achieved the target.
Transferring these guidelines to the problem at hand leads to the challenge of having to achieve a precise target orbit and defining the respective reward function that does not unevenly weight one orbital element with respect to another. Attempts to define the target orbit as objective by using the sum of the deviations of the individual orbital elements yielded poor results. Consequently, a reward definition based on the distance between the current and target orbital position, synchronised via the true anomaly of the current orbital motion, is used.

$$r := 1 - \frac{distance}{a_{initial} - a_{target}} \tag{4}$$

It should be noted that by evaluating and adding the reward at every state and thus timestep, a clear incentive is being made for the agent to reach the target orbit as fast as possible and not necessarily as precise as possible. This could be mitigated by choosing a non-linear weighing of the distance or only evaluating the reward at the last timestep of the simulation. However, this would come at the cost of less 'guidance' for adjusting the Q-values in training and might lead to divergence instead of a more precisely achieved target.

4. Training

The training process is conducted in an episodic sequence, meaning the Environment gets reset to its initial state after a defined number of timesteps is taken. One such episode simulates 250 hours of operations with a time step of 90 seconds.

4.1 Hyperparameter study on modified environment

As per chapter 2.1, the chosen algorithm and modifications leave a large parameter space to tune the behaviour and performance of the learning process. Depending on the choice of these parameters, the learning process achieves an optimal policy quickly and reliably or only reaches a local optimum or even diverges. Thus, a hyperparameter study was conducted. To find a well-performing parameter set, the learning rate, batch size and memory capacity are iterated on. Further hyperparameters such as τ and ε are not considered due to the high computational burden and limited resource availability. Additionally, the environment was slightly adjusted to speed up simulation times

DOI: 10.13009/EUCASS2023-682

drastically. This includes artificially increasing power availability and specific impulse to conclude the prescribed manoeuvre in a reduced time, reducing the overall simulation steps needed to finish an episode. While this comes at the cost of limited transferability of the resulting hyperparameters to the realistic environment, it does provide a rather quick estimation of the parameter ranges to be used.

Learning rate

The learning rate heavily affects the speed of convergence and achieved performance of the model. If chosen too low for the given problem structure, the learning process will converge towards the reward optimum rather slowly, leading to excessively large computational time. If chosen too large, the process might diverge and thus not achieve to settle in the global optimum. The result is displayed in figure 6, with a favourable range of $1x10^{-5}$ to $1x10^{-4}$.

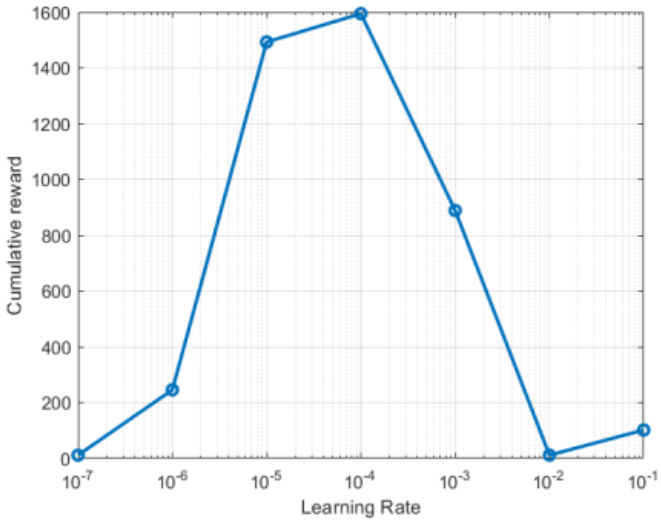

Figure 6: Effect of learning rate variation on cumulative reward.

Memory capacity

A Memory capacity chosen too small can lead to insufficient sampling of the state-action space. Data points would get ‘forgotten’ and not explored again. In the community, this phenomenon is often referred to as ‘catastrophic forgetting’. If chosen too large however can lead to dilution of the data set and data points containing relevant information being dampened. From the hyperparameter study conducted, no clear conclusions regarding the size of the memory could be drawn as no distinctive optimum is present.

Batch size

The batch size determines on how many randomly sampled data points per time step is getting learned on. A large batch leads to statistical regression, which increases learning time. A small batch reduces the amount of data learned on per step and thus leads to an increase in process time as well.
Again, no clear conclusions could be drawn from the study.

The results suggest an overall large level of noise of the learning process with poor reproducibility. Some level of noise however is expected due to the randomising elements of initiation of the Q-network weights and the epsilon-greedy exploration.

5. Results

The results of the hyperparameter study are used as an initial guess for the hyperparameters for the unmodified environment. Those were then varied in a trial-and-error fashion until satisfactory agent behaviour was accomplished at the end of the training process. A typical evolution of the cumulated reward per episode on a training process on 300 consecutive episodes can be seen in figure 7.

DOI: 10.13009/EUCASS2023-682

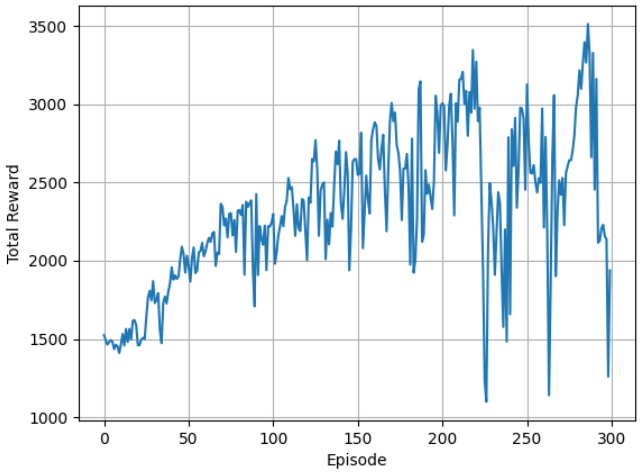

Figure 7: Typical total reward evolution over a learning session of 300 episodes.

Up until episode 200, a steady increase in collected rewards but also in noise level can be observed. This is attributed to the epsilon-greedy enforced exploration, which linearly decreases from an initial 95 % random actions taken to 1% from episode 200 on. Due to shifting towards an exploitation strategy of the policy established from the exploration driven phase, on average an increase in reward is achieved. While this trend continues at first, a sharp decline in performance and increased noise can be observed around episode 220. This points to the known issues of instability and risk of divergence in the process of approximating and exploiting the Q-Network in parallel. In addition, the 'catastrophic forgetting' phenomenon could be at play, where the memory of the exploration phase gets gradually replaced by the data tuples accumulated in the exploitation phase. This can lead to a poor coverage of the state space in the memory buffer and thus to bad Q-Value estimations.
Regardless of these issues, occasionally the highest reward of the session is achieved in this instability region.

The resulting policy of the agent emerging from this training set is mostly in line with the expectations for an optimal operation for the scenario. The behaviour can be obtained by applying the final resulting Q-network to one episode. Looking at the evolution of the pressure in the intermediate gas tanks over an episode, the general approach becomes apparent: As expected, thrust manoeuvres are conducted in the vicinity of the apogee, while the time in between is used for electrolyser operation and thus producing propellants (see figure 8, left). However, certain artifacts were present on subsets of an episode.
One artifact is characterised by not using the entire available time period to charge the gas tanks prior to conducting a manoeuvre (figure 8, middle). This is likely due to incomplete training for this region of the state space or due to the aforementioned 'catastrophic forgetting', where the optimal policy might have been found before but was overwritten by renewing the memory buffer.
The second artifact can be described by the agent choosing to conduct a larger thrust manoeuvre starting from a high pressures level slightly before passing apogee, and then performing multiple smaller manoeuvres directly following (figure 8, right). While the same potential reasons as for the other artifact apply, an additional hypothesis is that the agent learned to exploit how the reward is defined. Charging the tanks for a singular manoeuvre directly at apogee makes the most precise orbital change towards the desired target orbit. This strategy however comes at the disadvantage of decreased efficiently of the electrolysis due to the backpressure as modelled. That means time and power is not ideally used for that time period. Leveraging on the initially higher propellant production rate and sacrificing some accuracy towards the target orbit could lead to higher total rewards. This remains to be proven, however the importance of characterising and accounting for the backpressure effect of the electrolyser is highlighted.

DOI: 10.13009/EUCASS2023-682

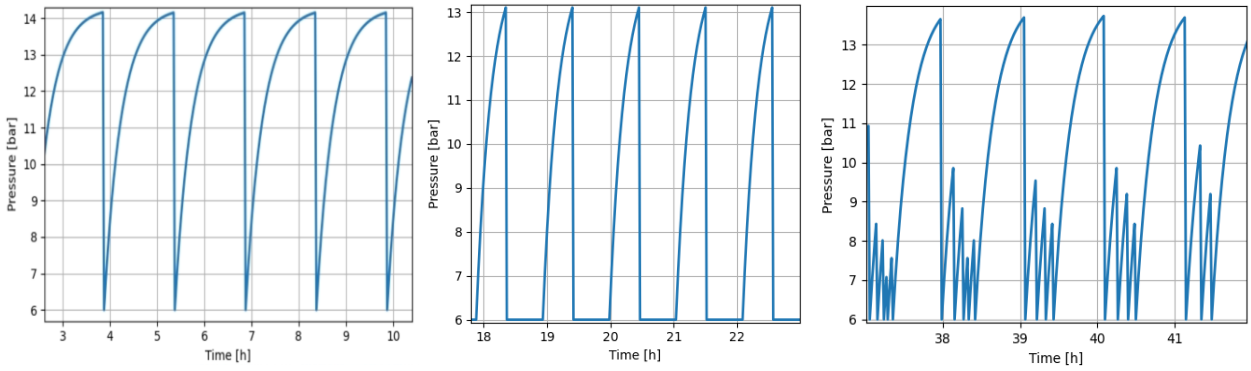

Figure 8: Different ConOps artifacts in charging/discharging the intermediate tanks.

Overall, the agent achieved policies as expected on the given scenario, however artifacts in the operations are occasionally present and need to be further investigated.

7. Summary and outlook

It could be demonstrated that an RL approach using Deep-Q-Learning can be used to train an agent to control a spacecraft and its WEP system that is capable of:

1. Deciding when and where to perform propulsive manoeuvres to achieve a given target orbit
2. Doing so while considering the need to pre-produce propellants
3. Achieving the task as time-efficiently as possible (as defined by the reward function)

and thus serves as an initial PoC of the framework for future improvements in fidelity and performance.

The DQN algorithm is shown to be able to be used to train an agent on the de-orbit scenario presented. However, the training process was prone to instabilities and in some cases divergence. Further work on the basis of DQN should focus on incorporating adjustments to stabilise the process and efficiently and reliably achieve a policy with global optimality.

Policy Optimisation RL algorithms could prove useful with improvements of the modeling fidelity of spacecraft and propulsion subsystem because of the ability to handle continuous action spaces. The output layer of a NN could directly represent the control parameters used on a real system such as current and voltage supplied to the electrolyser.

Furthermore, the flexibility and adaptability of RL approach w.r.t. noise and a change in objective and scenario needs to be assessed. A step-by-step deployment process building on the framework implemented from an improved fidelity digital twin via Hardware-in-the-loop setups on testbenches up to complete prototypes and potentially an In-orbit-Demonstration of autonomous capabilities is foreseen in the future.

DOI: 10.13009/EUCASS2023-682

References

[1] N. Fernandes, A. Herbertz, B. Buergler, U. Gotzig, M. Peukert, M. Lau, J. Farnes and T. Moreira. 2020. Electrolysis based water propulsion for future 1-ton class LEO mission satellites. In: *Space Propulsion Conference 2020*.

[2] S. Heizmann, A. Herbertz, J. Saryczew and C. Manfletti. 2023. Investigation of a Cathode-Vapour-Feed Electrolyser for a Water Electrolysis Propulsion System. In: *Aerospace Europe Conference 2023 - 10th EUCASS - 9th CEAS*

[3] N. Harmansa. 2020. Entwicklung und Charakterisierung eines Satellitenantriebssystems basierend auf Wasserelektrolyse. PhD Thesis. Dr. Hut Verlag, München.

[4] J. Hildebrandt, J. Kaufmann, S. Fasoulas and G. Herdrich. 2022. Development of novel 3D printed ceramic thruster and surface tension tank for water electrolysis propulsion system. In: *Space Propulsion Conference 2022*.

[5] U. Gotzig, M. Wurdak and N. Harmansa. 2023. Development and coupled thruser / electrolyser tests of a water propulsion system. *Acta Astronautica* 202: 751-759

[6] S. Dengler, F. Ebert, C. Manfletti and S. Leichtfuß. 2022. Water Electrolysis Propulsion in CubeSats: A case study. In: *Space Propulsion Conference 2022*.

[7] R. S. Sutton and A. G. Barto. 2018. Reinforcement learning: An introduction. Cambridge, Massachusetts; London, England, The MIT Press.

[8] T. Kohne, H. Ranzau, N. Panten and M. Weigold. 2020. Comparative study of algorithms for optimized control of industrial energy supply systems. *Energy Informatics* 2020 3:12.

[9] DeepMind Technologies. Playing Atari with Deep Reinforcement Learning. Whitepaper.

[10] OpenAI. 2016. OpenAI Gym. Whitepaper.

[11] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus and N. Dormann. 2021. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research* 22:1-8

[12] Juan Luis Cano Rodríguez et al.. (2015). poliastro: poliastro 0.17.0. Zenodo. 10.5281/zenodo.6817189