Deep Q-Learning for Infinite-Horizon Optimal Control Problems Governed by Parabolic PDEs

Master's Thesis

submitted by

Magnus Gohm

at the

Universität Konstanz

Department of Mathematics and Statistics

Superviser and 1st Reviewer: Jun. Prof. Behzad Azmi.
2nd Reviewer: Prof. Dr. Stefan Volkwein.
Konstanz, 2024

Declaration

Hereby I declare that I have written the following thesis, entitled

Deep Q-Learning for Infinite-Horizon Optimal Control Problems Governed by Parabolic PDEs

independently and have only used the sources indicated. All passages in the thesis in which words or ideas are taken from other sources have been marked as borrowings in each individual case by indicating the source and in the case of verbatim quotations by means of quotation marks. Furthermore, I assert that this thesis has not been submitted in fulfillment of course requirements for another course.

Konstanz, _____ _____

Magnus Gohm

Abstract

This thesis considers discounted infinite horizon optimal control problems which are governed by parabolic partial differential equations. A finite element discretization is employed to write these optimal control problems, as linear-quadratic optimal control problems. In order to obtain an optimal control law, we are then entering the field of reinforcement-learning or more specifically Q-Learning, where we want to approximate the optimal Q-function or optimal state-control value function. Using the Hamilton-Jacobi Deep Q-Learning (HJDQN) algorithm, we can then train a neural network which will act as optimal control law. Under the utilization of Lipschitz-continuous control functions, we are then able to reduce the training process to one neural network, instead of two for most actor-critic methods like the deep deterministic policy gradient (DDPG) algorithm. For that are methods involving Deep Q-Learning and Double Q-Learning essential. We then analyse the HJDQN algorithm in numerical simulations, where our examples cover linear-quadratic optimal control problems, with linear and non-linear parabolic partial differential equations.

Zusammenfassung

Diese Arbeit befasst sich mit diskontierten Optimalsteuerungsproblemen mit unendlichem Zeithorizont, die durch parabolische partielle Differentialgleichungen restringiert sind. Um diese Optimalsteuerungsprobleme zu lösen, wird eine finite Elemente Diskretisierung verwendet, um ein linear-quadratisches Optimalsteuerungsproblem zu erhalten. Um eine optimale Steuerung zu bestimmen wird Q-Learning aus dem Bereich des Reinforcement-Learnings verwendet, um die optimale Q-Funktion bzw. die optimale Zustands-Steuerungs Wertefunktion zu approximieren. Dabei wird mithilfe des Hamilton-Jacobi Deep Q-Learning (HJDQN) Algorithmus ein neuronales Netzwerk trainiert, welches als optimale Steuerung verwendet wird. Im Gegensatz zu den meisten Actor-Critic Methoden wie dem Deep Deterministic Policy Gradient (DDPG) Algorithmus welcher zwei neuronale Netzwerke trainiert, wird damit unter Verwendung von Lipschitz-stetigen Steuerungsfunktionen erreicht, den Trainingsprozess auf ein neuronales Netzwerk zu beschränken. Dabei ist die Methode des Deep Q-Learnings (DQN) als auch des Double Q-Learnings essentiell. In numerischen Simulationen wird der HJDQN Algorithmus dann auf das linear-quadratische Optimalsteuerungsproblem angewendet, wobei Beispiele mit linearen und nichtlinearen parabolischen partiellen Differentialgleichungen untersucht werden.

Contents

1 Introduction

The desire for model-free data-driven algorithms to handle problems where the state and control variables evolve in a continuous-time settings is enormous, since many physical control processes arise in continuous-time and not in discrete time. However is the theory which backs up most of the field of reinforcement learning build upon sequential decision making problems. Hence is there a particular interest in ways to adjust these methods to the continuous-time case.

Recent approaches developed by Google Deepmind [20] involves an off-policy model-free actor-critic model which adapts the classical Deep Q-Learning (DQN) method [24] to continuous control spaces using the deterministic policy gradient algorithm [32]. In their work they tried to work around the issue that the DQN algorithm can only work with low-dimensional control spaces, which is due to the fact that the Q-Learning method relies on finding the control that maximizes the Q-function also known as greedy control choice. The problem is that a continuous control space would then also require an iterative optimization method, which needs to be solved in every step. Another approach to tackle problems with a continuous control spaces, is to model the greedy choice for the control using mixed-integer programming (MIP), which is described by the CAQL algorithm [30].

Although the actor-critic methods as well as the CAQL algorithm are used with great success in practice, they don't use any inside knowledge of the underlying dynamical system. Normally is the dynamical programming equation expressed by the Hamilton-Jacobi-Bellman (HJB) equation which is employed to find the optimal state-value function or V-function. This is realized using a variety of methods see for example an approach with neural networks [34] where they use an iterative least squares minimization of the HJB residual with a neural network approximation of the V-function. Or using adaptive dynamical programming methods [40].

This work uses a novel approach which was first introduced in [16] where they derived a version of the HJB-equation which omits the state-control value function or Q-function as a viscosity solution. In addition to that they also introduced a semi-discrete form of this HJB equation which can be used to implement a Q-Learning method. In particular their approach which is called the HJDQN algorithm is an extension of the classical Q-Learning methodology introduced by [39] using Deep Q-Learning [24] and Double Q-Learning ([37], [38]). The main reason on why this HJB equation can be used to find the Q-function is due to the assumption that the control functions are Lipschitz continuous. This assumption is essential since it will enable us to find a choice for the greedy control, using the solution of a simple ODE. Furthermore this implies also, that we can drop the need of an additional actor network, which in turn can save computational cost.

The experiments conducted in [16] also cover some simple examples of linear quadratic regulator problems (LQR), for which the HJDQN algorithm provided a sufficient solution. In this work we want to apply the HJDQN algorithm on an discounted infinite horizon optimal control problem, which is governed by parabolic PDE's. These can be written as a LQR problem using a Galerkin approximation. We will try the algorithm on two linear cases in one and two dimensions, where we used the autonomous case of the optimal control problem given in [4]. In addition to that we also analyse the HJDQN algorithm on a optimal control problem which is constrained by a non-linear PDE, more specifically on a Allen-Cahn equation which is taken out of [5].

1.1 Outline

The sections 2 to 5 should be understand as a deeper introduction into the field of reinforcement learning, where we not only introduce the later applied techniques but go into a bit more detail.

- In section 2 we quickly introduce the basic terms of reinforcement learning, which includes the exploration and exploitation trade-off as a central concept.
- Section 3 then covers the classical discounted Q-Learning methodology in the deterministic case. These concepts are then applied on a general LQR problem. The same is done for the equivalent LQR problem in continuous time, for which we also introduce the HJB equation.
- To calculate the state-value function V or the state-control function Q we will present basic methods like value and policy iteration, as well as temporal difference learning in section 4

- In section 5 we give an introduction in stochastic Q-Learning, which is necessary since the HJDQN algorithm uses the principle of Double Q-Learning. In addition to that we also look at the stochastic gradient descent method and the Adam optimizer which are used in the stochastic Deep Q-Learning algorithm.
- The HJDQN algorithm together with the underlying semi-discrete form of the HJB equation, is established in section 6.
- In section 7 we then apply the HJDQN algorithm on the one and two dimensional example problems. Since the Riccati solution of the LQR problem exists, in case the constraint is a linear system, we can compare it to the solution of the HJDQN model. Additionally we also go into a bit more detail on the model and implementation details. Beyond that we want to apply the HJDQN algorithm on an infinite horizon optimal control problem with a non-linear constraint.
- In section 8 we give a brief outlook on possible improvements which could be used to modify the HJDQN algorithm.

2 Reinforcement Learning

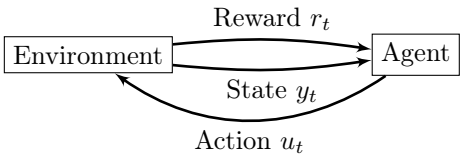
We want to first give a introduction into the basic principles of reinforcement learning, after that we want to present the concept of Q-Learning in the sense of dynamical programming in much more detail.

2.1 Basic Principles of Reinforcement Learning

The introduction to reinforcement learning leans on in ([13], Subsections 1.1 and 1.2).

The concept of reinforcement learning together with supervised and unsupervised learning, are the main topics in machine learning. While the methods in supervised and unsupervised learning learn and find patterns in labelled and unlabelled datasets respectively, the main difference to reinforcement learning is that the methods there simply learn by exploring and exploiting actions in a given environment. In reinforcement learning we are given two components that interact with each other. These are given by the environment and the agent. The main goal of the agent is to learn to take the optimal actions maximizing the incurred reward by interacting with the environment.

- **Environment:** Describes the space where we can execute actions u_t and also observe the current or resulting states y_t or rewards r_t respectively. Additionally an environment can be partially or fully observable. For example the game of chess is fully observable (every position of every piece is known the whole time) and the game of poker is only partially observable (hand card's from opponent's can't be observed).
- **Agent:** It's task is to choose and apply actions u_t on the environment given the observed reward r_t and state y_t.



The success of the methods in supervised and unsupervised learning all depend in some kind on the trade-off between bias and variance. If a model is too simple it usually has high bias and low variance but if the model is too complex than it usually has high variance and low bias. The goal here is to balance the model complexity or to keep the equilibrium between bias and variance. Similarly a fundamental principle of the learning process in reinforcement learning is the trade-off between exploration and exploitation.

- **Exploration:** Exploration describes the ability of the agent to go off the expected path when he needs to find a new action to execute, with this ability the risk is higher to get an action which gives us a lower reward but we can also get the advantage of finding a better action which offers a different and maybe more rewarding path. Therefore does exploration entirely depend on the previously executed action and not on any other existing knowledge. One possible exploration strategy to determine a new action could be to take a random direction from the previously executed one.
- **Exploitation:** The opposite strategy from exploration is exploitation. This means we are given a path and a general direction we can take with our actions and we want to maximize our reward using this path as far as possible. The obvious downside of this strategy is that we can miss potentially better actions which could've lead us to a more rewarding path. Therefore depends the exploitation only on already existing knowledge.

Hence this trade-off needs to be balanced in order to achieve better learning results. As a real life example you could look into clinical trials and compare exploitation: Using a thoroughly known medical therapy vs. exploration: Using a experimental therapy instead.

3 Q-Learning in Deterministic and Continuous Time

3.1 Basic Definitions

We want to introduce Q-Learning using dynamical programming. At first we are going to use deterministic dynamical programming to describe a infinite horizon optimal control problem in a discrete space. This section is orientated on of ([23], Part I).

Definition 3.1. *Let $k \in \mathbb{N}$. We define the discrete-time dynamic system as*

$$\begin{aligned} y_{k+1} &= f(y_k, u_k) \\ y_0 &\in Y \end{aligned} \tag{3.1}$$

where $f : Y \times U \to Y$ is the system function which describes the state change from time k to $k+1$, $y_k \in Y \subseteq \mathbb{R}^n$ is the state at time k, $u_k \in U \subseteq \mathbb{R}^m$ the control at time k and $y_0 \in Y$ the initial state. We further call Y the state space and U the control space.

Given the state equation we further define the cost functional as

Definition 3.2. *Let $y_0 \in Y \subseteq \mathbb{R}^n$ be a given initial state and $u = \{u_k\}_{k\in\mathbb{N}} \subset U \subseteq \mathbb{R}^m$ be a given control sequence, we then define the discounted cost functional as*

$$J(y_0; u) := \sum_{k=0}^{\infty} \gamma^k r(y_k, u_k) \tag{3.2}$$

where $r : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}_+$ is the reward function which describes the incurred reward from taking the control u_k in the state y_k for all $k \in \mathbb{N}$ and $0 \leq \gamma \leq 1$ is the discount factor.

Remark 3.3.

1. The discount factor γ is used to make sure that the chosen controls are neither too short-sighted or too far-sighted, which further helps to prioritize action choices which minimize the total discounted reward and not just the immediate and future rewards.
2. To assure that the discounted cost functional is well defined, we have to assume the following conditions
   a) There is a state $y_e \in Y$ such that there exists a control $u_e \in U$ for which follows that y_e acts as a equilibrium state, which means we have $f(y_e, u_e) = y_e$.
   b) The reward function r is non-negative and it vanishes in this equilibrium state which means $r(y_e, u_e) = 0$.
   c) For every $y_0 \in Y$ there is a control sequence $u^0 \subset U$ and a time $k^0 \in \mathbb{N}$ such that $y_{k^0} = y_e$.
   d) $0 < \gamma < 1$.

To represent the state and control vectors with the incurred reward, we can define the so called trajectory, which describes the path we are going along using our control sequence.

Definition 3.4. *Let $u := \{u_k\}_{k\in\mathbb{N}} \subset U$ be a control sequence and $\{y_k\}_{k\in\mathbb{N}}$ be the associated state sequence defined by* (3.1)*. We then call $\tau := \{\tau_k\}_{k\in\mathbb{N}}$ the control trajectory which is defined by*

$$\tau_k := (y_k, u_k, r_k) \quad \text{for } k \in \mathbb{N}$$

where $r_k := r(y_k, u_k)$ for $k \in \mathbb{N}$.

Using these definitions we define the deterministic infinite horizon optimal control problem as minimizing the discounted cost functional, over all control sequences $u := \{u_k\}_{k\in\mathbb{N}}$ given initial state $y_0 \in Y$. Which means

$$\min_u J(y_0; u). \tag{3.3}$$

If the minimum is attained at $\bar{u} = \{\bar{u}_k\}_{k\in\mathbb{N}}$ we write $J(y_0, \bar{u})$ as the minimum value and say that

Definition 3.5. *The control $\bar{u} = \{\bar{u}_k\}_{k\in\mathbb{N}}$ is optimal for $y_0 \in Y \subseteq \mathbb{R}^n$ if for all sequences $u := \{u_k\}_{k\in\mathbb{N}} \subset U$ follows that*

$$J(y_0; u) \geq J(y_0; \bar{u}) \tag{3.4}$$

From central importance is the Bellman's principle of optimality also called the fundamental theorem of dynamical programming, which serves as characterization of the optimal control sequence $\bar{u} = \{\bar{u}_k\}_{k\in\mathbb{N}}$. The key aspect is to use the so called tail sub-problems which are defined as follows

Definition 3.6. *Let $y_i \in Y \subseteq \mathbb{R}^n$ for $i \in \mathbb{N}$. Then the tail cost functional starting from time i is defined as*

$$J(y_i; u) := \sum_{k=i}^{\infty} \gamma^{k-i} r(y_k, u_k) \tag{3.5}$$

and the tail sub-problem is defined as

$$\min_u J(y_i; u). \tag{3.6}$$

With that we formulate Bellman's optimality principle in the following way

Theorem 3.7. *Let $\bar{u} = \{\bar{u}_k\}_{k\in\mathbb{N}} \subset U$ be the optimal control sequence of* (3.3) *with the associated optimal state sequence $\bar{y} = \{\bar{y}_k\}_{k\in\mathbb{N}} \subset Y$ defined by* (3.1)*. Then is for all $i \in \mathbb{N}$ the control sequence $\bar{u}^i = \{\bar{u}_k\}_{k\geq i} \subset U$ optimal for the tail sub-problem* (3.6)*.*

Proof. The proof follows by contradiction in the following way. Assume that $\bar{u} = \{\bar{u}_k\}_{k\in\mathbb{N}} \subset U$ is a optimal control sequence and that for some $i \in \mathbb{N}$ there is a sequence $v^i = \{v_k\}_{k\geq i} \subset U \setminus \bar{u}$ which is optimal for the associated tail sub-problem. We first show that the sequence $w = \{w_k\}_{k\in\mathbb{N}} \subset U$, where $w_k = \bar{u}_k$ for $0 \leq k < i$ and $w_k = v_k$ for $i \leq k$ is an optimal control sequence for (3.3) with the associated optimal state sequence $x = \{x_k\}_{k\in\mathbb{N}} \subset Y$, where $x_k = \bar{y}_k$ for $0 \leq k < i$ and $x_k = z_k$ for $i \leq k$ defined by (3.1). Now let $u = \{u_k\}_{k\in\mathbb{N}} \subset U$ be arbitrary with associated state sequence $s = \{s_k\}_{k\in\mathbb{N}} \subset Y$ then follows

$$\begin{aligned}
J(\bar{y}_0; u) &= \sum_{k=0}^{\infty} \gamma^k r(s_k, u_k) \\
&= \sum_{k=0}^{i-1} \gamma^k r(s_k, u_k) + \sum_{k=i}^{\infty} \gamma^k r(s_k, u_k) \\
&= \sum_{k=0}^{i-1} \gamma^k r(s_k, u_k) + \gamma^i \left[\sum_{k=i}^{\infty} \gamma^{k-i} r(s_k, u_k) \right] \\
&\geq \sum_{k=0}^{i-1} \gamma^k r(\bar{y}_k, \bar{u}_k) + \gamma^i \left[\sum_{k=i}^{\infty} \gamma^{k-i} r(w_k, v_k) \right] \\
&= \sum_{k=0}^{i-1} \gamma^k r(\bar{y}_k, \bar{u}_k) + \sum_{k=i}^{\infty} \gamma^k r(w_k, v_k) \\
&= J(\bar{y}_0; w).
\end{aligned}$$

Therefore is $w = \{w_k\}_{k\in\mathbb{N}}$ an optimal control sequence which contradicts the assumption that $\bar{u} = \{\bar{u}_k\}_{k\in\mathbb{N}} \subset U$ is optimal.

■

To obtain an optimal control sequence we are going to use the so called value function

Definition 3.8. *Let $y_i \in Y \subseteq \mathbb{R}^n$ for $i \in \mathbb{N}$ and $u = \{u_k\}_{k\in\mathbb{N}} \subset U$. Then is the value function given by*

$$V(y_i, u) := J(y_i; u). \tag{3.7}$$

Using Bellman's optimality principle we can derive a algorithm which solves the optimal control problem by solving it backwards. For that we can define the optimal value function, which is the minimum value of 3.7 over all control sequences, using the tail sub-problem in the following way

$$V^*(y_k) := \min_{u_k\in U} [\, r(y_k, u_k) + \gamma V^*(f(y_k, u_k)) \,] \tag{3.8}$$

We therefore obtained the optimal value function V^* which will be associated with the optimal control sequence. Notice that we have to calculate (3.8) for every $y_k \in Y \subseteq \mathbb{R}^n$ in order to find the optimal control $\bar{u}_k$ which minimizes $V^*(y_k)$.

Having obtained a representation for the optimal value function, we are now able to construct the optimal control sequence $\bar{u} = \{\bar{u}_k\}_{k\in\mathbb{N}} \subset U$ using (3.8) in the following way

$$\bar{u}_k \in \arg\min_{u_k\in U} [r(\bar{y}_k, u_k) + \gamma V^*(\bar{y}_{k+1})] \tag{3.9}$$

for $k \in N$, where $\bar{y}_{k+1} = f(\bar{y}_k, \bar{u}_k)$. The obvious downside of this approach is that in order to determine the optimal control sequence, we first have to obtain the optimal value function, which can be excessively expensive because of (3.8). For that reason one can use approximations of the optimal value function V^* instead. Using the approximations we will naturally obtain a suboptimal control sequence $\tilde{u} = \{\tilde{u}_k\}_{k\in\mathbb{N}} \subset U$ by simply replacing V^* in (3.9) with it's approximation $\tilde{V}$, this means we have

$$\tilde{u}_k \in \arg\min_{u_k\in U} \left[r(\tilde{y}_k, u_k) + \gamma \tilde{V}(\tilde{y}_{k+1}) \right] \tag{3.10}$$

for $k \in \mathbb{N}$, where $\tilde{y}_{k+1} = f(\tilde{y}_k, \tilde{u}_k)$.

The way in which we determined the optimal control sequence as seen in (3.9) is usually expressed as a so called policy, which is defined as follows

Definition 3.9. *A function $\pi : Y \to U$ is called a policy function, which determines the control that should be applied in the current state.*

This methodology on obtaining the optimal control sequence represents one of the main topics in reinforcement learning. A special case of this method is Q-Learning. The Q-function is defined as follows

Definition 3.10. *Let $y_i \in Y \subseteq \mathbb{R}^n$ for $i \in \mathbb{N}$ and $u = \{u_k\}_{k\in\mathbb{N}} \subset U$. Then is the Q-function given by*

$$Q(y_i, u) := J(y_i; u). \tag{3.11}$$

Similar to the definition of the optimal and approximate value function we can define the optimal and approximate Q-function.

Definition 3.11. *Let $y_k \in Y \subseteq \mathbb{R}^n$ for $k \in \mathbb{N}$ and $u = \{u_k\}_{k\in\mathbb{N}} \subset U$. We define for $k \in \mathbb{N}$ the approximate Q-function as*

$$\tilde{Q}(y_k, u_k) := r(y_k, u_k) + \gamma \tilde{V}(f(y_k, u_k)) \tag{3.12}$$

and the optimal Q-function as

$$Q^*(y_k, u_k) := r(y_k, u_k) + \gamma V^*(f(y_k, u_k)) \tag{3.13}$$

This essentially allows us to write the optimal value function as

$$V^*(y_k) = \min_{u_k\in U} Q^*(y_k, u_k). \tag{3.14}$$

The consequence of this is that we can get an equivalent of (3.8), which exclusively uses the optimal Q-function instead of the exact value function. For that we insert (3.14) into (3.13) to get

$$\begin{aligned} Q^*(y_k, u_k) &= r(y_k, u_k) + \gamma V^*(f(y_k, u_k)) \\ &= r(y_k, u_k) + \gamma \min_{u_{k+1} \in U} Q^*(f(y_k, u_k), u_{k+1}). \end{aligned} \tag{3.15}$$

Further using (3.14) we can also rewrite (3.9) as

$$\bar{u}_k \in \arg\min_{u_k \in U} Q^*(\bar{y}_k, u_k). \tag{3.16}$$

3.2 Application: Linear-Quadratic Regulator in Discrete-Time

We now want to use the introduced methodology to solve the linear-quadratic regulator problem in discrete time. For that we have the following dynamical system

$$\begin{aligned} \frac{d}{dt} y &= Ay + Bu \\ y(0) &= y_0 \in \mathbb{R}^n \end{aligned} \tag{3.17}$$

where $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$. The reward function is given as

$$r(y, u) = x^\top Q x + u^\top R u \tag{3.18}$$

where $Q \in \mathrm{Sym}_+(n)$ and $R \in \mathrm{Sym}_{++}(m)$, which means that Q is positive semi-definite and R is positive definite. We assume for now that the value function is quadratic which means that there is a $P \in \mathrm{Sym}_+(n)$ such that $V^*(y) = y^\top P y$. Using this and the definition 3.10 we can write the Q-function as

$$Q^*(y, u) = r(y, u) + \gamma V^*(Ay + Bu) = y^\top Q y + u^\top R u + \gamma (Ay + Bu)^\top P (Ay + Bu) \tag{3.19}$$

Assuming that the Q-function is continuously differentiable, we can use (3.14) and calculate $\nabla_u Q^*(y, u)$ in order to use it as first order optimality condition. We have

$$\nabla_u Q^*(y, u) = 2Ru + 2\gamma B^\top P (Ay + Bu) \tag{3.20}$$

Since $R \in \mathrm{Sym}_{++}(m)$, $P \in \mathrm{Sym}_+(n)$ and $\gamma \in (0, 1)$ it follows that $R + \gamma B^\top P B \in \mathrm{Sym}_{++}(m)$. Hence follows

$$\begin{aligned} 0 = \nabla_u Q^*(y, u) &\Leftrightarrow 0 = 2Ru + 2\gamma B^\top P (Ay + Bu) \\ &\Leftrightarrow 0 = 2Ru + 2\gamma B^\top P A y + 2\gamma B^\top P B u \\ &\Leftrightarrow 0 = 2\gamma B^\top P A y + 2(R + \gamma B^\top P B) u \\ &\Leftrightarrow -\gamma B^\top P A y = (R + \gamma B^\top P B) u \\ &\Leftrightarrow u = -\gamma (R + \gamma B^\top P B)^{-1} B^\top P A y \end{aligned}$$

Using $(y^*, u^*) = \left(y^*, -\gamma (R + \gamma B^\top P B)^{-1} B^\top P A y^*\right)$ to check the second order optimality condition gives us

$$\nabla_u^2 Q^*(y^*, u^*) = 2R + 2\gamma B^\top P B \in \mathrm{Sym}_{++}(m)$$

This means we found $\pi^*(y^*) = u^* = K^* y^*$ where $K^* := -\gamma (R + \gamma B^\top P B)^{-1} B^\top P A$ which satisfies (3.16). Since K^* is dependent on P, we need to find a way to obtain it. We can derive the so called discounted discrete algebraic Riccati equation (DARE), whose solution will coincide with P. For that we are going tu use (3.8) which gives us

$$V^*(y^*) = \min_{u \in U} \left[r(y^*, u) + \gamma V^*(f(y^*, u)) \right] \Leftrightarrow$$

$$\begin{aligned} y^{*\top} P y^* &= \min_{u \in U} \left[y^{*\top} Q y^* + u^\top R u + \gamma \left(Ay^* + Bu\right)^\top P \left(Ay^* + Bu\right) \right] \\ &= \min_{u \in U} \left[\begin{pmatrix} y^* \\ u \end{pmatrix}^\top \begin{pmatrix} Q + \gamma A^\top P A & \gamma A^\top P B \\ \gamma B^\top P A & R + \gamma B^\top P B \end{pmatrix} \begin{pmatrix} y^* \\ u \end{pmatrix} \right] \\ &\overset{u = u^*}{=} y^{*\top} \left(Q + \gamma A^\top P A - \gamma^2 A^\top P B \left(R + \gamma B^\top P B\right)^{-1} B^\top P A \right) y^* \end{aligned}$$

from this we then obtain the DARE in the following form

$$P = Q + \gamma A^\top PA - \gamma^2 A^\top PB \left(R + \gamma B^\top PB\right)^{-1} B^\top PA \tag{3.21}$$

If we set $\tilde{A} := \sqrt{\gamma}$ and $\tilde{R} = \frac{1}{\gamma}R$ we get the standard form of the DARE given by

$$P = Q + \tilde{A}^\top P\tilde{A} - \tilde{A}^\top PB \left(\tilde{R} + B^\top PB\right)^{-1} B^\top P\tilde{A} \tag{3.22}$$

We want restrict the analysis of existence and uniqueness of solutions to the Riccati equation in this work on the continuous-time case. Since we now got the matrix P we can represent the optimal Q-function as well as the optimal value function in the following way

$$V^*(y) = y^\top Py, \quad Q^*(y,u) = \begin{pmatrix} y \\ u \end{pmatrix}^\top \begin{pmatrix} Q + \gamma A^\top PA & \gamma A^\top PB \\ \gamma B^\top PA & R + \gamma B^\top PB \end{pmatrix} \begin{pmatrix} y \\ u \end{pmatrix}$$

3.3 Hamilton-Jacobi-Bellman Equations in Continuous-Time

Since our main problem will be in continuous time, we are going to analogously introduce the value function and Q-function for continuous-time. We furthermore introduce the Hamilton-Jacobi-Bellman equation (HJB) which will serve as optimality condition to determine the optimal control. We are going to start with the un-discounted case

Un-discounted Continuous-time Infinite Horizon Optimal Control

We first have the continuous-time dynamical system defined as

$$\begin{aligned} \frac{d}{dt}y(t) &= f(y(t), u(t)) \quad , t > 0 \\ y(0) &= y_0 \end{aligned} \tag{3.23}$$

where $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, $y_0 \in \mathbb{R}^n$. Further is the cost functional in continuous-time defined as

$$J(y, u) := \int_0^\infty r(y(t), u(t))\, dt \tag{3.24}$$

where $r : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}_+$.

Remark 3.12.

1. To assure that the cost functional in continuous time is well defined, we have to assume the following conditions

   a) There is a state $y_e \in Y$ such that there exists a control $u_e \in U$ for which follows that y_e acts as a equilibrium state where we have $f(y_e, u_e) = 0$.

   b) The reward function r is non-negative and it vanishes in this equilibrium state which means $r(y_e, u_e) = 0$.

With that we can define the optimal value function for $t \in [0, \infty)$ and $y_t \in \mathbb{R}^n$ as

$$V^*(y_t) := \min_{u \in U} \left\{ \int_t^\infty r(y(s), u(s))\, ds \,\middle|\, y(t) = y_t \right\}. \tag{3.25}$$

The next step is to use Bellman's optimality principle to get an analogue statement to (3.8). For that let $h > 0$, we then get

$$\begin{aligned} V^*(y_t) &= \min_{u \in U} \left\{ \int_t^\infty r(y(s), u(s))\, ds \,\middle|\, y(t) = y_t \right\} \\ &= \min_{u \in U} \left\{ \int_t^{t+h} r(y(s), u(s))\, ds + \int_{t+h}^\infty r(y(s), u(s))\, ds \,\middle|\, y(t) = y_t \right\}. \\ &= \min_{u \in U} \left\{ \int_t^{t+h} r(y(s), u(s))\, ds + V^*(y(t+h)) \,\middle|\, y(t) = y_t \right\} \end{aligned} \tag{3.26}$$

Using (3.26) we now are able to determine the HJB equation. We first subtract $V^*(y_t)$ on both sides and divide by $h > 0$.

$$0 = \min_{u \in U} \left\{ \frac{1}{h} \int_t^{t+h} r(y(s), u(s))\, ds + \frac{1}{h} \left(V^*(y(t+h)) - V^*(y_t) \right) \,\middle|\, y(t) = y_t \right\}$$

The next step is to let $h \to 0$. Using the fundamental theorem of differentiation and integration we first get that

$$\lim_{h\to 0} \frac{1}{h} \int_t^{t+h} r(y(s), u(s))\, ds = r(y_t, u_t) \tag{3.27}$$

and assuming that the value function is continuously differentiable we obtain

$$\lim_{h\to 0} \frac{1}{h} \left(V^*(y(t+h)) - V^*(y(t))\right) = \nabla V^*(y(t)) \cdot \frac{d}{dt} y(t) = \nabla V^*(y(t)) \cdot f(y(t), u(t)). \tag{3.28}$$

Using (3.27) and (3.28) we then get the HJB equation

$$0 = \min_{u\in U} \left[\, r(y(t), u(t)) + \nabla V^*(y(t)) \cdot f(y(t), u(t)) \,\right]. \tag{3.29}$$

This implies that we can now use (3.29) to determine the optimal control. If we assume that (3.29) has a unique solution we can get the optimal control with

$$u_t^* = \pi^*(y_t^*) = \operatorname*{argmin}_{u\in U} \left[\, r(y_t^*, u) + \nabla V^*(y_t^*) \cdot f(y_t^*, u) \,\right] \tag{3.30}$$

Equation (3.30) is of a special form. We call

$$H(y, p, u) := r(y, u) + p^\top f(y, u) \tag{3.31}$$

the Hamiltonian. Notice that $Q^*(y, u) = H(y, \nabla V^*(y), u)$.

Discounted Continuous-time Infinite Horizon Optimal Control

We now want to use the discounted cost functional in continuous-time defined as

$$J(y, u) := \int_0^\infty e^{-\gamma t} r(y(t), u(t))\, dt. \tag{3.32}$$

Remark 3.13. To also assure that the discounted cost functional in continuous time is well defined, we have to assume the conditions from Remark 3.12 an additionally that $0 < \gamma < 1$.

With that we can again define the optimal value function for $t \in [0, \infty)$ and $y_t \in \mathbb{R}^n$ as

$$V^*(y_t) := \min_{u\in U} \left\{ \int_t^\infty e^{-\gamma(s-t)} r(y(s), u(s))\, ds \,\middle|\, y(t) = y_t \right\}. \tag{3.33}$$

We also want to use Bellman's optimality principle to get an analogue statement to (3.8). For that let $h > 0$ and define $g(s, t) := e^{-\gamma(s-t)} r(y(s), u(s))$. Since $e^{-\gamma h} g(s, t+h) = g(s, t)$ we get

$$\begin{aligned}
V^*(y_t) &= \min_{u\in U} \left\{ \int_t^\infty g(s,t)\, ds \,\middle|\, y(t) = y_t \right\} = \min_{u\in U} \left\{ \int_t^{t+h} g(s,t)\, ds + \int_{t+h}^\infty g(s,t)\, ds \,\middle|\, y(t) = y_t \right\} \\
&= \min_{u\in U} \left\{ \int_t^{t+h} g(s,t)\, ds + e^{-\gamma h} \int_{t+h}^\infty g(s, t+h)\, ds \,\middle|\, y(t) = y_t \right\} \\
&= \min_{u\in U} \left\{ \int_t^{t+h} g(s,t)\, ds + e^{-\gamma h} V^*(y(t+h)) \,\middle|\, y(t) = y_t \right\}
\end{aligned} \tag{3.34}$$

Using (3.34) we now are able to determine the HJB equation. We first subtract $V^*(y_t)$ on both sides and divide by $h > 0$ to obtain

$$0 = \min_{u \in U} \left\{ \frac{1}{h} \int_t^{t+h} g(s,t)\, ds + \frac{e^{-\gamma h} - 1}{h} V^*(y(t+h)) + \frac{1}{h} \left(V^*(y(t+h)) - V^*(y_t) \right) \middle| y(t) = y_t \right\}.$$

The next step is to let $h \to 0$. Using the fundamental theorem of differentiation and integration we first get that

$$\lim_{h \to 0} \frac{1}{h} \int_t^{t+h} g(s,t)\, ds = g(t,t) = r(y_t, u_t) \tag{3.35}$$

further follows that

$$\lim_{h \to 0} \frac{e^{-\gamma h} - 1}{h} V^*(y(t+h)) = \lim_{h \to 0} \frac{e^{-\gamma(h+0)} - e^{-\gamma 0}}{h} V^*(y(t+h)) = -\gamma V^*(y_t) \tag{3.36}$$

and assuming that the value function is continuously differentiable we obtain

$$\lim_{h \to 0} \frac{1}{h} \left(V^*(y(t+h)) - V^*(y(t)) \right) = \nabla V^*(y(t)) \cdot \frac{d}{dt} y(t) = \nabla V^*(y(t)) \cdot f(y(t), u(t)). \tag{3.37}$$

Using (3.35), (3.36) and (3.37) we then get the HJB equation

$$0 = \min_{u \in U} \left[r(y(t), u(t)) - \gamma V^*(y(t)) + \nabla V^*(y(t)) \cdot f(y(t), u(t)) \right]. \tag{3.38}$$

3.4 Application: Linear-Quadratic Regulator in Continuous-Time

Since some of the main problems of this work will also be in linear-quadratic form we will now also apply the HJB equation for continuous time on the linear-quadratic regulator problem. The only difference to the discrete case is that we now use the cost functional (3.32) with $r(y,u) = y^\top Q y + u^\top R u$ with Q and R defined as before. We assume that the optimal value function can be expressed with a matrix $P \in Sym_+(n)$ such that $V^*(y) = y^\top P y$.

To show that this is a valid assumption we show that the optimal value function V^* is a quadratic form.

Lemma 3.14. *Let $Q \in Sym_+(n)$, $R \in Sym_{++}(m)$ and $0 < \gamma < 1$, then holds for the optimal value function V^* the following conditions*

1. $\forall \lambda \in \mathbb{R}$, $\forall y \in \mathbb{R}^n$: $V^*(\lambda y) = \lambda^2 V^*(y)$
2. $\forall y_1, y_2 \in \mathbb{R}^n$: $V^*(y_1) + V^*(y_2) = \frac{1}{2} \left(V^*(y_1 + y_2) + V^*(y_1 - y_2) \right)$

Hence is V^* a quadratic form and there exists a $P \in Sym_+(n)$ such that $V^*(y) = y^\top P y$.

Proof. For a proof of 1.) and 2.) see ([3], Section 2.3). Assume now that 1.) and 2.) hold. If we take the derivative on both sides of 1.) with respect to y gives us

$$\lambda \nabla V^*(\lambda y) = \lambda^2 \nabla V^*(y) \Leftrightarrow \nabla V^*(\lambda y) = \lambda \nabla V^*(y)$$

and if we take the derivative on both sides of 2.) one time with respect to y_1 and then with y_2 we get

$$\nabla V^*(y_1) = \frac{1}{2} \left(\nabla V^*(y_1 + y_2) + \nabla V^*(y_1 - y_2) \right)$$

$$\nabla V^*(y_2) = \frac{1}{2} \left(\nabla V^*(y_1 + y_2) - \nabla V^*(y_1 - y_2) \right)$$

which we add up to obtain $\nabla V^*(y_1) + \nabla V^*(y_2) = \nabla V^*(y_1 + y_2)$. This shows that $\nabla V^*(y)$ is linear in y. Hence exists $\tilde{P} \in \mathbb{R}^{n\times n}$ such that $\nabla V^*(y) = \tilde{P}y$. Using the fundamental theorem of integration gives us

$$V^*(y) = V^*(0) + \int_0^1 \nabla V^*(\tau y)^\top y \, d\tau = V^*(0) + \int_0^1 \tau y^\top \tilde{P}^\top y \, d\tau = V^*(0) + \frac{y^\top \tilde{P}^\top y}{2}.$$

Since 1.) holds we further have $V^*(0) = V^*(0 \cdot y) = 0^2 V^*(y) = 0$, which concludes $V^*(y) = \frac{y^\top \tilde{P}^\top y}{2}$. Now notice that $y^\top \tilde{P}^\top y = y^\top \tilde{P} y$ which then gives us

$$V^*(y) = \frac{1}{2}\left(\frac{y^\top \tilde{P}^\top y}{2} + \frac{y^\top \tilde{P} y}{2}\right) = \frac{1}{4} y^\top \left(\tilde{P}^\top + \tilde{P}\right) y.$$

With that we define $P := \frac{1}{4}\left(\tilde{P}^\top + \tilde{P}\right)$. P is by definition symmetric and also positive semi-definite since $r(y,u) \geq 0$ for all $y \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$.

■

Using the above lemma we substitute everything into (3.38) and get

$$\begin{aligned} 0 &= \min_{u\in U} \left[y^\top Q y + u^\top R u - \gamma y^\top P y + (2Py)^\top (Ay + Bu)\right] \\ \Leftrightarrow 0 &= \min_{u\in U} \left[u^\top R u + (2Py)^\top B u\right]. \end{aligned}$$

We see that the minimizer of this coincides with the minimizer of the un-discounted case for the linear quadratic regulator. But to find practical solutions for the discounted case we don't use (3.38) but the discrete version of the same equation described by (3.34). By using the Taylor expansion we get

$$\int_t^{t+h} g(s,t)\, ds =: G(h) = G(0) + G'(0)h + \mathcal{O}(h^2) = g(t,t)h + \mathcal{O}(h^2) = r(y(t),u(t))h + \mathcal{O}(h^2) \tag{3.39}$$

and

$$V^*(y(t+h)) = V^*(y(t)) + \nabla V^*(y(t)) \cdot (y(t+h) - y(t)) + \mathcal{O}(h^2). \tag{3.40}$$

Using (3.39) and (3.40) in (3.34) we get that

$$\begin{aligned} V^*(y(t)) &= \min_{u\in U} \left\{ \int_t^{t+h} g(s,t)\, ds + e^{-\gamma h} V^*(y(t+h)) \,\middle|\, y(t) = y_t \right\} \\ &= \min_{u\in U} \left\{ r(y(t),u(t))h + e^{-\gamma h}\left(V^*(y(t)) + \nabla V^*(y(t)) \cdot (y(t+h) - y(t))\right) \,\middle|\, y(t) = y_t \right\} \\ &\quad + \mathcal{O}(h^2). \end{aligned}$$

If we now subtract $V^*(y(t))$ from the left-hand side of the above equation at first and then divide by $h > 0$, we get with $y(t+h) - y(t) = f(y(t),u(t))h$ that

$$0 = \min_{u\in U} \left\{ r(y(t),u(t)) + \frac{e^{-\gamma h} - 1}{h} V^*(y(t)) + e^{-\gamma h} \nabla V^*(y(t)) \cdot f(y(t),u(t)) \,\middle|\, y(t) = y_t \right\} + \mathcal{O}(h). \tag{3.41}$$

We now want to again apply that on the linear quadratic problem. For that we again set $r(y,u) = y^\top Q y + u^\top R u$ and $V^*(y) = y^\top P y$ where Q, R and P are defined as before. Substituting this into (3.41) gives us

$$\begin{aligned} 0 &= \min_{u\in U} \left[y^\top Q y + u^\top R u + \frac{e^{-\gamma h} - 1}{h} y^\top P y + (2e^{-\gamma h} P y)^\top (Ay + Bu)\right] \\ \Leftrightarrow 0 &= \min_{u\in U} \left[u^\top R u + (2e^{-\gamma h} P y)^\top B u\right]. \end{aligned} \tag{3.42}$$

Using the first order optimality condition for (3.42) or $L(u) := u^\top Ru + (2e^{-\gamma h}Py)^\top Bu$ we get

$$0 = \nabla_u L(u) = 2Ru + 2e^{-\gamma h}B^\top Py \Leftrightarrow u^* = -e^{-\gamma h}R^{-1}B^\top Py$$

which is a minimum since $\nabla_u^2 L(u^*) = R > 0$. To obtain the Ricatti equation for P we only have to substitute $u = u^*$ into (3.42) and get

$$\begin{aligned}
0 =& y^\top Qy + (-e^{-\gamma h}R^{-1}B^\top Py)^\top R(-e^{-\gamma h}R^{-1}B^\top Py) + \frac{e^{-\gamma h}-1}{h}y^\top Py \\
&+ (2e^{-\gamma h}Py)^\top (Ay + B(-e^{-\gamma h}R^{-1}B^\top Py)) \\
\Leftrightarrow 0 =& y^\top Qy + e^{-2\gamma h}y^\top PBR^{-T}RR^{-1}B^\top Py - \frac{1-e^{-\gamma h}}{h}y^\top Py \\
&+ 2e^{-\gamma h}y^\top PAy - 2e^{-2\gamma h}y^\top PBR^{-1}B^\top Py \\
\Leftrightarrow 0 =& y^\top Qy + e^{-2\gamma h}y^\top PBR^{-1}B^\top Py - \frac{1-e^{-\gamma h}}{h}y^\top Py \\
&+ e^{-\gamma h}y^\top \left(PA + A^\top P\right) y - 2e^{-2\gamma h}y^\top PBR^{-1}B^\top Py \\
\Leftrightarrow 0 =& y^\top Qy - e^{-2\gamma h}y^\top PBR^{-1}B^\top Py - \frac{1-e^{-\gamma h}}{h}y^\top Py + e^{-\gamma h}y^\top \left(PA + A^\top P\right) y \\
\Leftrightarrow 0 =& y^\top \left[Q - e^{-2\gamma h}PBR^{-1}B^\top P - \frac{1-e^{-\gamma h}}{h}P + e^{-\gamma h}\left(PA + A^\top P\right)\right] y
\end{aligned}$$

If we set $\tilde{A} = e^{-\gamma h}A - \frac{1-e^{-\gamma h}}{2h}I_n$ and $\tilde{B} = e^{-\gamma h}B$ we get

$$0 = y^\top \left[Q + (P\tilde{A} + \tilde{A}^\top P) - P\tilde{B}R^{-1}\tilde{B}^\top P\right] y$$

which gives us the standard form of the continuous algebraic ricatti equation (CARE)

$$0 = Q + (P\tilde{A} + \tilde{A}^\top P) - P\tilde{B}R^{-1}\tilde{B}^\top P \tag{3.43}$$

This will be useful in our practical examples, where we want to solve the discounted continuous-time LQR. We now want to analyse the existence and the uniqueness for solutions P to equation (3.43). The theory for Riccati solutions uses the un-discounted cost

$$J(y,u) = \int_0^\infty r(y(t),u(t))\,dt \tag{3.44}$$

instead of (3.32). Additionally to that we need a few more assumptions on the system

$$\begin{aligned}
y' &= Ay(t) + Bu(t) \\
y(t_0) &= y_0 \in \mathbb{R}^n
\end{aligned} \tag{3.45}$$

which are given by the following.

Definition 3.15. *The system (A,B), where $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ is called controllable if for the controllability matrix*

$$\Gamma(A,B) := \left[B, AB, \ldots, A^{n-1}B\right]$$

holds that rank$(\Gamma) = n$.

Definition 3.16. *The system (A,B), where $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$, is called stabilizable if every uncontrollable state is asymptotically stable.*

in addition to that we also use

Definition 3.17. *The system (A, B), where $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$, is called observable if $(B^\top, A^\top)$ is controllable.*

Definition 3.18. *The system (A, B), where $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$, is called detectable if $\left(B^\top, A^\top\right)$ is stabilizable.*

Furthermore is the associated CARE defined as

$$0 = Q + PA + A^\top P - PBR^{-1}B^\top P. \tag{3.46}$$

The first step is to check if the solutions of the Riccati equation (3.46) also minimizes the undiscounted cost (3.44). For that we have

Theorem 3.19. *If there exists a $P \in Sym_+(n)$ which solves* (3.46)*, then there also exists a unique solution $P^* \in \mathbb{R}^{n\times n}$, such that the optimal control*

$$u^*(t) = -R^{-1}B^\top P^* y(t), \quad t \geq 0$$

minimizes (3.44) *and for the optimal value function holds that $V^*(y(t)) = y(t)^\top P^* y(t)$.*

Proof. For a proof we refer to ([41], Theorem 1.4). ■

Using the stabilizability definition on (3.45) allows us to determine the existence of solutions to (3.46). This is given by

Theorem 3.20. *If the system (A, B) is stabilizable, then has* (3.46) *at least one solution. Moreover if there exists $C \in \mathbb{R}^{n\times n}$ such that $Q = C^\top C$ and the system (A, C) is detectable, then exists at most one solution P of* (3.46) *such that $A - BR^{-1}B^\top P$ is stable.*

Proof. A proof can be found in ([41], Theorem 1.5). ■

For completion we also can get the uniqueness of a solution P if we strengthen the assumptions and use controllability and observability instead of stabilizibility and detectability.

Corrolary 3.21. *If the system (A, B) is controllable and there exists $C \in \mathbb{R}^{n\times n}$ such that $Q = C^\top C$ and the system (A, C) is observable, then exists exactly one unique solution P of* (3.46) *such that $A - BR^{-1}B^\top P$ is stable.*

Proof. A proof can be found in ([41], Theorem 1.6). ■

4 Dynamical Programming

4.1 Value and Policy Iteration in Discrete-Time

We now want to introduce two essential methods, namely value iteration and policy iteration, which will provide a solution that fulfills the optimality principle by Bellman. This section is orientated on ([23], Part I).

The value iteration method approximates the exact value function iteratively using (3.8) in the following way. Let V_0 be an initial approximation for V^* we then define the next iteration as

$$V_{n+1}(y_k) := \min_{u_k \in U} \left[r(y_k, u_k) + \gamma V_n(f(y_k, u_k)) \right] \tag{4.1}$$

for $k \in \mathbb{N}$ where $y_k \in Y$ and $n \geq 0$. If we iteratively insert the recursion formula into itself we get that

$$V_{n+1}(y_0) := \min_{\{u_k\}_{k \leq n}} \left[\sum_{k=0}^{n} \gamma^k r(y_k, u_k) + \gamma^{n+1} V_0(f(y_n, u_n)) \right]. \tag{4.2}$$

We can proof this by induction. For that let $V_0(y_0) \equiv 0$ be the initial approximation for V^*. Now let $n = 0$ then follows

$$V_1(y_0) := \min_{\{u_k\}_{k \leq 0}} \left[\sum_{k=0}^{0} \gamma^k r(y_k, u_k) + \gamma^1 V_0(f(y_0, u_0)) \right] = \min_{u_0} \left[r(y_0, u_0) + \gamma V_0(f(y_0, u_0)) \right].$$

Now assume that (4.2) holds for all $n \in \mathbb{N}$. For the induction step we go from $n \to n+1$. This means using the induction assumption we get

$$\begin{aligned}
V_{n+1}(y_0) &= \min_{u_0} \left[r(y_0, u_0) + \gamma V_n(f(y_0, u_0)) \right] \\
&= \min_{u_0} \left[r(y_0, u_0) + \gamma \min_{\{u_k\}_{1 \leq k \leq n}} \left[\sum_{k=0}^{n-1} \gamma^k r(y_{k+1}, u_{k+1}) + \gamma^n V_0(f(y_n, u_n)) \right] \right] \\
&= \min_{u_0} \left[r(y_0, u_0) + \min_{\{u_k\}_{1 \leq k \leq n}} \left[\sum_{k=0}^{n-1} \gamma^{k+1} r(y_{k+1}, u_{k+1}) + \gamma^{n+1} V_0(f(y_n, u_n)) \right] \right] \\
&= \min_{\{u_k\}_{0 \leq k \leq n}} \left[\sum_{k=0}^{n} \gamma^k r(y_k, u_k) + \gamma^{n+1} V_0(f(y_n, u_n)) \right]
\end{aligned}$$

where the last equation holds since the inner minimum isn't dependent on u_0. Hence follows by the principle of complete induction that (4.2) hold for all $n \in \mathbb{N}$.

We now want to make sure that this iteration converges, which is in general given by the following conditions

Proposition 4.1. *Assume that*

1. *Y and U are finite.*
2. *The reward function r is non-negative and it vanishes only in the equilibrium state which means $r(y_e, u_e) = 0$, furthermore let J be finite valued.*
3. *V_0 is chosen with non-negative entries and aside from that set $V_0(x_e) = 0$.*

Then there is a $n_0 \geq 1$ such that

$$V_n(y_0) = J(y_0, \bar{u})$$

for all $n \geq n_0$, where $\bar{u}$ is the optimal control sequence.

Proof. Let π^* be an optimal policy. It follows from (2) that there is an $n_0 \geq 1$ such that $(\bar{y}_k, \bar{u}_k) = (y_e, u_e)$ for all $k \geq n_0$. Now let $y_0 \in Y$. Using (4.2) we get for $n \in \mathbb{N}$

$$V_n(y_0) = \min_{\{u_k\}_{k\leq n}} \left[\sum_{k=0}^{n-1} \gamma^k r(y_k, u_k) + \gamma^n V_0(f(y_n, u_n)) \right] \leq \sum_{k=0}^{n-1} \gamma^k r(\bar{y}_k, \bar{u}_k) + \gamma^n V_0(f(\bar{y}_n, \bar{u}_n)). \tag{4.3}$$

Furthermore define the finite cost function for $y_0 \in Y$ and a control sequence $\{u_k\}_{k\in\mathbb{N}} \subset U$ as

$$J_n(y_0, u) := \sum_{k=0}^{n} \gamma^k r(y_k, u_k) \tag{4.4}$$

Using (4.3), (4.4) and (3) for $n \geq n_0$ we get that

$$V_n(y_0) \leq \sum_{k=0}^{n-1} \gamma^k r(\bar{y}_k, \bar{u}_k) + \gamma^n V_0(f(\bar{y}_n, \bar{u}_n)) = J_n(y_0, \bar{u}) + V_0(y_e) = J_n(y_0, \bar{u}).$$

Hence follows from the optimality of π^* and (2) that $V_n(y_0) = J(y_0, \bar{u})$.

■

In contrary to the value iteration method will the policy iteration method iteratively determine an optimal policy using (3.9) in the following way:

Let π_0 be an initial policy. To find a new iteration π_{n+1} we need to solve the following minimization problem

$$\pi_{n+1}(y_k) \in \arg\min_{u_k \in U} \left[r(y_k, u_k) + V^{\pi_n}(f(y_k, u_k)) \right] \tag{4.5}$$

where $V^{\pi_n}(y_k) = J(y_k, u)$ for $y_k \in Y$ and $u = \{u_i\}_{i\geq k} \subset U$ with $u_i = \pi_n(y_i)$. Here we have to do essentially two defining calculations the first one is to find V^{π_n}. For that we are going to use Bellman's optimality principle again which means we want to determine V^{π_n} from the following equation

$$V^{\pi_n}(y_k) = r(y_k, \pi_n(y_k)) + \gamma V^{\pi_n}(f(y_k, \pi_n(y_k))) \tag{4.6}$$

The other necessary calculation is the minimization in (4.5). Both of them can be very expensive. Using this procedure we get an improvement of the value function in every iteration, which is shown in the following

Proposition 4.2. *Let π_0 be an initial policy such that V^{π_0} is bounded. Let $y \in Y$ then follows for all $n \in \mathbb{N}_{\geq 0}$ that*

$$r(y, \pi_{n+1}(y)) + V^{\pi_n}(f(y, \pi_{n+1}(y))) \leq V^{\pi_n}(y) \tag{4.7}$$

and furthermore

$$V^{\pi_n}(y) \geq V^{\pi_{n+1}}(y). \tag{4.8}$$

Proof. Let $y_0 \in Y$ and let π_n be a given policy defined by (4.7). This policy then also fulfills Bellman's principle of optimality given by (4.8). Using (4.7) and (4.8) we get that

$$V^{\pi_n}(y_0) = r(y_0, \pi_n(y_0)) + \gamma V^{\pi_n}(f(y_0, \pi_n(y_0))) \geq r(y_0, \pi_{n+1}(y_0)) + \gamma V^{\pi_n}(f(y_0, \pi_{n+1}(y_0))). \tag{4.9}$$

This gives us (4.7). If we apply (4.7) again on (4.9) for $\gamma V^{\pi_n}(f(y_0, \pi_n(y_0)))$ we get that

$$\begin{aligned} V^{\pi_n}(y_0) &\geq r(y_0, \pi_{n+1}(y_0)) + \gamma V^{\pi_n}(f(y_0, \pi_{n+1}(y_0))) \\ &\geq r(y_0, \pi_{n+1}(y_0)) + \gamma \left[r(y_1, \pi_{n+1}(y_1)) + \gamma V^{\pi_n}(f(y_1, \pi_{n+1}(y_1))) \right] \\ &= r(y_0, \pi_{n+1}(y_0)) + \gamma r(y_1, \pi_{n+1}(y_1)) + \gamma^2 V^{\pi_n}(f(y_1, \pi_{n+1}(y_1))) \end{aligned}$$

Since there is a $k_0 \in \mathbb{N}$ such that $r(y_{k_0}, \pi_{n+1}(y_{k_0})) = 0$ for all $k \in \mathbb{N}$ s.t. $k \geq k_0$, we further get that

$$V^{\pi_n}(y_0) \geq \sum_{k=0}^{k_0-1} \gamma^k r(y_k, \pi_{n+1}(y_k)) = \sum_{k=0}^{\infty} \gamma_k r(y_k, \pi_{n+1}(y_k)) = V^{\pi_{n+1}}(y_0)$$

which shows (4.8).

■

4.2 Temporal Difference Learning in Discrete-Time

Now we are going to introduce one of the most used algorithms in Q-Learning which is temporal difference learning or TD-Learning for short. TD-Learning is essentially a special case of the value iteration algorithm. Furthermore we are going to focus on off-policy learning, that means that we are updating the Q-function or value function using the greedy policy which means choosing the optimal controls, such that they reach the minima in (3.15) and (4.1) respectively.

To approximate the value function or Q-function one methodology which is often used in practise, is to obtain a solution to a projected problem. Simplistically we can write (4.1) or (3.15) in a more general fixed point equation form. For that let H be the space of Q-functions, the goal is then to solve the fixed point equation

$$Q^* = T(Q^*). \tag{4.10}$$

Since this can be either too complex or too computationally expensive to solve we resort to a projected problem. For that let $\tilde{H} \subset H$ and define a projection $P_{\tilde{H}} : H \to \tilde{H}$, $Q \mapsto \tilde{Q}$. Then instead of solving (4.10) we can solve

$$\tilde{Q} = \tilde{T}(\tilde{Q}) := P_{\tilde{H}}(T(\tilde{Q})). \tag{4.11}$$

This problem can also turn out to be either too complex or too computationally expensive. An additional generalization step which can be applied to (4.11) is to use another function space G and projection $P_G : \tilde{H} \to G$ to solve the equation

$$0 = \left(\tilde{Q} - P_G(\tilde{T}(\tilde{Q}))\right). \tag{4.12}$$

Remark 4.3. This methodology does not have a supporting theory. But it will arise more or less naturally when we introduce the TD-Learning algorithm. ■

Now let T be a mapping defined by (3.15) in the following way

$$T(Q)(y, u) := r(y, u) + \gamma Q(y', \pi(y)) \tag{4.13}$$

where $Q : Y \times U \to \mathbb{R}$, π is a fixed policy and $y' = f(y, u)$. With that we can write (3.15) as $Q = T(Q)$. We now need a space $\tilde{H}$. For that we are going to use a Galerkin approach and define $\tilde{H} = \operatorname{span}\{\varphi_1, \ldots, \varphi_d\}$ where $\varphi_i : Y \times U \to \mathbb{R}$ are linearly independent. Now let $Q^\theta \in \tilde{H}$ where $\theta \in \mathbb{R}^d$ such that $Q^\theta = \sum_{i=1}^d \theta_i \varphi_i$.

The next usual step in the Galerkin method would be to multiply (4.13) by φ_i and then to integrate. Since we are in the deterministic setting we first need to define how we apply the Galerkin method. To start with that we need a error definition, to decide how good our approximation of the Q-function in a current state is. For that we are using

$$\mathfrak{D}_{k+1}(Q^\theta) := T(Q^\theta)(y_k, u_k) - Q^\theta(y_k, u_k) = \left[r(y_k, u_k) + \gamma Q^\theta(f(y_k, u_k), \pi(y_k))\right] - Q^\theta(y_k, u_k) \tag{4.14}$$

which is called the temporal difference. With that we can define a loss function L which describes the quality of the approximation of the Q-function. For that let $M \in \mathbb{N}_{\geq 1}$ be a given time horizon and let $\{(y_k, u_k) \in Y \times U \,|\, k \in \mathbb{N}_{\leq M}\}$ be the observed samples. We then define the loss function with $N = M + 1$ as

$$L(Q) = \frac{1}{N} \sum_{k=1}^{N} [r(y_{k-1}, u_{k-1}) + \gamma Q(y_k, \pi(y_{k-1}))] - Q(y_{k-1}, u_{k-1}) = \frac{1}{N} \sum_{k=1}^{N} \mathfrak{D}_k(Q). \tag{4.15}$$

Since we want our approximation to fulfill $L(Q) = 0$ or $\mathfrak{D}_k(Q) = 0$ for all $k \in \mathbb{N}_{\leq N}$ as good as possible, we can then apply the Galerkin method if we set $Q = Q^*$ followed by multiplying with φ_i which then results in the following Galerkin orthogonality

$$0 = \frac{1}{N} \sum_{k=1}^{N} \mathfrak{D}_k(Q^\theta) \varphi_i(\pi(y_k)) \quad \text{for } 1 \leq i \leq d \tag{4.16}$$

Since our problem has an infinite horizon we need to make additional assumptions to be able to let $N \to \infty$. For this we need to modify our policy definition in the following way.

Definition 4.4. *Let $(\varepsilon_k)_{k\in\mathbb{N}}$ be a bounded sequence which advances on $\Omega \subseteq \mathbb{R}^p$ for some $p \geq 0$. We then call a function $\hat{\pi} : Y \times \Omega \to U$ perturbed policy. Hence we can obtain a new control on a current state $k \geq 0$ using*

$$u_k = \hat{\pi}(y_k, \varepsilon_k)$$

The policy in this definition which is also called quasi-stochastic policy allows us to apply the concept of exploration in the deterministic case of Q-Learning. One restriction we have on the sequence $(\varepsilon_k)_{k\in\mathbb{N}}$ is that we need to assume the existence of a continuous function $H : \Omega \to \Omega$ for which holds $\varepsilon_{k+1} = H(\varepsilon_k)$ for all $k \geq 0$. In order to reach a limit for $N \to \infty$ in (4.16) we additionally need the following assumptions which are taken from ([23], Assumption (Aξ)).

Assumptions for Quasi-Stochastic Policies

Let $Y \subseteq \mathbb{R}^n$ and $U \subseteq \mathbb{R}^m$ be two closed subsets. Further let the dynamical state function f, the quasi-stochastic policy $\hat{\pi}$ and the function H be continuous. Then is there a extension of the sequence $\Phi(k) := \{(y_k, u_k)\}_{k\in\mathbb{N}}$ described by $\hat{\Phi}(k) := \{(y_k, u_k, \varepsilon_k)\}_{k\in\mathbb{N}}$ which fulfills the following properties:

1. The function $w : Y \times U \to Y \times U \times \Omega$, $w(\Phi(k)) = \hat{\Phi}(k)$ is continuous.
2. There is a probability measure ω such that for any continuous function $g : Y \times U \to \mathbb{R}$ exists

$$\mathbb{E}_\omega[g(\Phi)] := \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} g(\Phi(k)) \tag{4.17}$$

   for each initial condition $\Phi(0)$.

3. The limit in (4.17) is uniformly convergent on the subspace of Lipschitz continuous functions on $Z = Y \times U$ which is defined by

$$C_L^0(Z) := \{g : Z \to \mathbb{R} \,|\, \forall z, z' \in Z : |g(z) - g(z')| \leq L|z - z'|_2\}$$

   which means that we have for each $L < \infty$

$$\lim_{N\to\infty} \sup_{g\in C_L^0(Z)} \left| \frac{1}{N} \sum_{k=1}^{N} g(\Phi(k)) - \mathbb{E}_\omega\left[g(\Phi)\right] \right| = 0 \tag{4.18}$$

For further details we refer to ([23], chapter 5). With these assumptions we can now continue with the Galerkin method by multiplying (4.14) by φ_i and then to integrate which is now done by taking the expected value $\mathbb{E}_\omega$ defined in the above assumption. We then obtain the following d equations

$$\begin{aligned} 0 &= \mathbb{E}_\omega\left[\left(Q^\theta(y_k, u_k) - T(Q^\theta)(y_k, u_k)\right) \varphi_i(\pi(y_k))\right] \\ &= \mathbb{E}_\omega\left[\left(Q^\theta(y_k, u_k) - \left[r(y_k, u_k) + \gamma Q^\theta(f(y_k, u_k), \pi(y_k))\right]\right) \varphi_i(\pi(y_k))\right] \end{aligned}$$

for all $i \in \mathbb{N}_{\leq d}$. Or equivalently

$$0 = \mathbb{E}_\omega\left[\mathfrak{D}_{k+1}(Q^\theta)\varphi(k)\right], \tag{4.19}$$

where $\varphi(k) = (\varphi_1(\pi(y_k)), \dots, \varphi_d(\pi(y_k)))^\top$. Using this now we can define the TD(λ) algorithm in the following way:

Algorithm 1 TD(λ)-Learning

Require: $\lambda \in [0,1]$, $\gamma \in [0,1)$, step sizes $(\alpha_n)_{n\in\mathbb{N}} \subset \mathbb{R}_{>0}$, initial conditions ζ_0, θ_0 and $N \in \mathbb{N}$ observed samples $\{(y_k, u_k) \in Y \times U \mid k \in \mathbb{N}_{\leq N}\}$.

1: $r_n = r(y_n, u_n)$
2: $\theta_{n+1} = \theta_n + \alpha_{n+1}\mathfrak{D}_{n+1}\zeta_n$
3: $\mathfrak{D}_{n+1} = \left[r(y_n, u_n) + \gamma Q^\theta(f(y_n, u_n), \pi(y_n))\right] - Q^\theta(y_n, u_n)$
4: $\zeta_{n+1} = \lambda\gamma\zeta_n + \varphi(n+1)$

The result of the algorithm is then an estimate for the Q-function given by

$$Q^{\theta_N} = \sum_{i=1}^{d} (\theta_N)_i \varphi_i. \tag{4.20}$$

The iteration formula $\theta_{n+1} = \theta_n + \alpha_{n+1}\mathfrak{D}_{n+1}\zeta_n$ is known as the Robbins-Monro algorithm [29]. Given a constant scalar value $c \in \mathbb{R}$ and a function $M : \mathbb{R}^d \to \mathbb{R}$ their method seeks for the solution of the equation

$$c = M(\theta). \tag{4.21}$$

If M isn't directly observable but there is a random variable $X : \mathbb{R}^d \to \mathbb{R}$ for which holds that

$$M(\theta) = \mathbb{E}\left[X(\theta)\right],$$

they then obtain the following iteration formula for θ given by

$$\theta_{n+1} = \theta_n - \alpha_n(X(\theta_n) - c). \tag{4.22}$$

The Robbins-Monro algorithm then converges under the following conditions

Theorem 4.5. *Assume that $\theta^* \in \mathbb{R}^d$ is the unique solution of (4.21). Now let $(\alpha_n)_{n\in\mathbb{N}} \subset \mathbb{R}_{>0}$ be a sequence of step sizes such that*

$$\sum_{n=0}^{\infty} \alpha_n = \infty \text{ and } \sum_{n=0}^{\infty} \alpha_n^2 < \infty.$$

Assume further that N is uniformly bounded, M is nondecreasing and $M'(\theta^*) > 0$. Then follows for the sequence $(\theta_n)_{n\in\mathbb{N}}$ defined by (4.22)

$$\lim_{n\to\infty} \mathbb{E}\left[(\theta_n - \theta^*)^2\right] = 0.$$

Proof. For a proof we refer to ([29], Theorem 2). ■

In the case of the TD(λ) iteration formula $\theta_{n+1} = \theta_n + \alpha_{n+1}\mathfrak{D}_{n+1}\zeta_n$ we then set $c := 0$ and $M(\theta) := \mathbb{E}_\omega\left[X(\theta)\right] := \mathbb{E}_\omega\left[\left(Q^\theta(y_k, u_k) - \left[r(y_k, u_k) + \gamma Q^\theta(f(y_k, u_k), \pi(y_k))\right]\right)\zeta_n\right] = \mathbb{E}_\omega\left[-\mathfrak{D}_{k+1}(Q^\theta)\zeta_n\right]$.

Since we used a linear approximation space for the Galerkin approach, we also want to be able to use non-linear approximation functions like neural networks. For that we want to use the following loss function, also known as mean-squared Bellman error

$$L(Q) = \frac{1}{N}\sum_{k=1}^{N} \mathfrak{D}_k(Q)^2. \tag{4.23}$$

Our goal here is to find a approximation Q^θ, where θ could be the weights of a neural network, such that $L(Q) = 0$ holds. For that we want to use the first order optimality condition $\nabla_\theta L(Q^\theta) = 0$ which gives us

$$\nabla_\theta L(Q^\theta) = \frac{1}{N}\sum_{k=1}^{N} \mathfrak{D}_k(Q^\theta)\nabla_\theta\mathfrak{D}_k(Q^\theta) = 0.$$

This has now a similar structure to the Galerkin orthogonality we defined in (4.16), if we let $\zeta_k^\theta = \nabla_\theta \mathfrak{D}_k(Q^\theta)$. The problem with that is that the Galerkin basis functions usually do not depend on a parameter θ. Although this is the case, the result which were obtained in practise still showed that its a reasonable choice to build the non-linear TD(λ)-Learning algorithm almost analogous to the linear case, which means

Algorithm 2 Non-Linear TD(λ)-Learning

Require: $\lambda \in [0,1]$, $\gamma \in [0,1)$, step sizes $(\alpha_n)_{n\in\mathbb{N}} \subset \mathbb{R}_{>0}$, initial conditions ζ_0, θ_0, $g_1 = \nabla_\theta Q(y_1,u_1)|_{\theta=\theta_1}$ and $N \in \mathbb{N}$ observed samples $\{(y_k,u_k) \in Y \times U \,|\, k \in \mathbb{N}_{\leq N}\}$.

1: $r_n = r(y_n,u_n)$
2: $\theta_{n+1} = \theta_n + \alpha_{n+1}\mathfrak{D}_{n+1}\zeta_n$
3: $\mathfrak{D}_{n+1} = \big[r(y_n,u_n) + \gamma Q^\theta(f(y_n,u_n),\pi(y_n))\big] - Q^\theta(y_n,u_n)$
4: $\zeta_{n+1} = \lambda\gamma\zeta_n + g_{n+1}$
5: $g_n = \nabla_\theta Q(y_n,u_n)|_{\theta=\theta_n}$

4.3 Q(λ)-Learning

A downside is that the convergence of the linear and especially the non-linear TD(λ)-Learning algorithm is not always guaranteed which also applies for the $\lambda = 0$ case. We now did define the TD(λ)-Learning algorithm with a fixed policy π which is also called on-policy TD(λ)-Learning, but what we really want according to (3.15) is the greedy policy. This is then called off-policy TD(λ)-Learning. We hence set for the policy in (4.13)

$$\pi^\theta(y) := \arg\min_{u\in U} Q^\theta(y,u).$$

The algorithm for Q(λ)-Learning is then basically the same as TD(λ)-Learning, which we see here for the sake of completeness.

Algorithm 3 Q(λ)-Learning

Require: $\lambda \in [0,1]$, $\gamma \in [0,1)$, step sizes $(\alpha_n)_{n\in\mathbb{N}} \subset \mathbb{R}_{>0}$, initial conditions ζ_0, θ_0 and $N \in \mathbb{N}$ observed samples $\{(y_k,u_k) \in Y \times U \,|\, k \in \mathbb{N}_{\leq N}\}$.

1: $r_n = r(y_n,u_n)$
2: $\theta_{n+1} = \theta_n + \alpha_{n+1}\gamma\mathfrak{D}_{n+1}\zeta_n$
3: $\mathfrak{D}_{n+1} = \big[r(y_n,u_n) + \gamma Q^\theta(f(y_n,u_n),\pi^\theta(y_n))\big] - Q^\theta(y_n,u_n)$
4: $\zeta_{n+1} = \lambda\gamma\zeta_n + \varphi(n+1)$

4.4 Deep Q-Learning

A huge challenge in practise is that the TD(λ)-Learning algorithm (linear and non-linear) wasn't really applicable for more complex problems e.g. chess or go. It was assumed that the method acted like a special case working well only in the case of the original problem it was applied on. A more recent development was the introduction of Deep Q-Networks by DeepMind [24] where this section is orientated on which marked a huge breakthrough in reinforcement learning.

The Deep Q-Learning method essentially consists of the following two steps.

1. Apply the concept of experience replay, to create a replay set from which we can sample mini-batches.
2. Use the mini-batch from step one to apply a non-linear version of the Q(0)-Learning algorithm.

These two steps combined then give the following algorithm

Algorithm 4 Deep Q-Learning with Experience Replay

Require: Replay set capacity $N \in \mathbb{N}$, Number of episodes $M \in \mathbb{N}$, Terminal time step $T \in \mathbb{N}$, $t = 0$ and initial state $y_t = y_0$.

Initialize the replay set $\mathfrak{R}$ with capacity N.
Initialize the neural network Q^θ with random or zero weights.
for episode $= 1$ to M **do**
  for $t = 0$ to T **do**
    Given a probability ε set $u_t = \begin{cases} \text{Choose a random policy} \\ \arg\min_{u \in U} Q^\theta(y_t, u) \end{cases}$.
    Determine incurred reward $r_t = r(y_t, u_t)$.
    Apply control u_t to obtain $y_{t+1} = f(y_t, u_t)$.
    Save experience sample $(y_t, u_t, r_t, y_{t+1}) \in \mathfrak{R}$.
    Sample randomized mini-batch from replay set $\mathfrak{R}$.
    Apply the non-linear Q(0)-Learning algorithm on the sampled mini-batch.
  end for
end for

Experience Replay

Lets start with the concept of experience replay. This short subsection is orientated on ([21], Section 3.5) where it was first introduced.

Assume that we are in a current state y_t, where we have the control u_t. Beside this information we actually also have the achieved reward r_t and if we apply the control u_t on the state y_t using the system function, we also can observe the next state y_{t+1}. This provides us with a quadruple of a so called experience sample of the form

$$e_t := (y_t, u_t, r_t, y_{t+1}). \tag{4.24}$$

These samples are then collected to form the so called replay set $\mathfrak{R} := \{e_t \,|\, 0 \leq t \leq I_{\max}\}$, where $I_{\max} \in \mathbb{N}$ is the maximum number of iterations. If we take a look at the Q(λ)-Learning algorithm we notice that the parameter θ, is only updated using each state-control pair from the sample $\{(y_t, u_t) < 0 \leq t \leq I_{\max}\}$ once. This is essentially a huge loss of information, meaning for example that the model could have experienced a useful strategy of choosing a reasonable control, which potentially would never be reused.

This is where the replay set comes into play, instead of using a fixed sample of state and controls we re-draw a sample from the replay set. Using this we can potentially gain two advantages, the first one is the speed up of the learning process, since we wouldn't have to determine a huge number of controls to obtain a reasonable total reward. The second one is that the model would be able to kind of refresh the already learned strategies. In order for the experience replay to work, the system function shouldn't be changing too quickly, since this can make the already experienced strategies either useless or it can lead future strategies astray.

A problem which can arise using the experience replay method is the over-replay-problem. This can occur if we only sample replay sequences, which are too recent but discard earlier or initial replay sequences, which usually consist of poor control choices. The reason for that is that the Q(λ)-Learning algorithm or the TD(λ)-Learning algorithm rely on visiting all states equally often. A way to tackle this problem is to use each replay sample approximately the same number of times.

Mini-Batch Non-Linear Q(0)-Learning

Now we want to introduce the second part of the DQN-Learning algorithm with experience replay consisting of a non-linear Q(λ)-Learning algorithm, which is applied on a mini-batch of the replay set.

We first want to divide the time interval $0, \ldots, N$ where $N \in \mathbb{N}$ into so called batches of equal size. For that we define $B \in \mathbb{N}$ batches on the equidistant grid $t_0 = 0 < t_1 < \ldots < t_{B-1} < t_B = N$.

Then is the update formula for the parameters of Q^θ for $n \in \mathbb{N} : 0 \leq n \leq B-1$, $(\alpha_n)_{n\in\mathbb{N}} \subset \mathbb{R}_{>0}$ and given $\theta_0 \in \mathbb{R}^d$ defined as

$$\theta_{n+1} = \arg\min_{\theta\in\mathbb{R}^d} \left(L_n(\theta) + \frac{1}{\alpha_{n+1}} \|\theta - \theta_n\|_2^2 \right), \tag{4.25}$$

where the loss function $L_n(\theta)$ is similar to (4.23). The crucial difference here is that the loss function depends on a single batch

$$L_n(\theta) = \frac{1}{(t_{n+1} - t_n)} \sum_{k=t_n}^{t_{n+1}-1} (r(y_k, u_k) + Q^{\theta_n}(f(y_k, u_k), \pi^{\theta_n}(y_k)) - Q^\theta(y_k, u_k))^2. \tag{4.26}$$

As we can see from (4.25) is that we get the next set of parameters θ_{n+1} if we not only minimize the mean-squared bellman error but also factor in the obtained parameters from the previous batch. Furthermore we get an alternative representation of θ_{n+1} if we assume the following

Lemma 4.6. *Let Q^θ be continuously differentiable in θ for all $\theta \in \mathbb{R}^d$. Then (4.25) can be written equivalently for $0 \leq n \leq B-1$, $(\alpha_n)_{n\in\mathbb{N}} \subset \mathbb{R}_{>0}$ and given $\theta_0 \in \mathbb{R}^d$ as*

$$\theta_{n+1} = \theta_n + \frac{\alpha_{n+1}}{t_{n+1} - t_n} \sum_{k=t_n}^{t_{n+1}-1} D_k(\theta_n, \theta_{n+1}) \nabla_{\theta_{n+1}} Q^{\theta_{n+1}}(y_k, u_k), \tag{4.27}$$

where

$$D_k(\theta_n, \theta_{n+1}) := r(y_k, u_k) + Q^{\theta_n}(f(y_k, u_k), \pi^{\theta_n}(y_k)) - Q^{\theta_{n+1}}(y_k, u_k). \tag{4.28}$$

Proof. To obtain (4.27) we take the derivative of the right side in (4.25) with respect to θ. This gives us

$$\begin{aligned}
0 &= \nabla_\theta \left(L_n(\theta) + \frac{1}{\alpha_{n+1}} \|\theta - \theta_n\|_2^2 \right) \\
\Leftrightarrow 0 &= \nabla_\theta \left(\frac{1}{(t_{n+1} - t_n)} \sum_{k=t_n}^{t_{n+1}-1} D_k(\theta_n, \theta)^2 + \frac{1}{\alpha_{n+1}} \|\theta - \theta_n\|_2^2 \right) \\
\Leftrightarrow 0 &= \frac{1}{2(t_{n+1} - t_n)} \sum_{k=t_n}^{t_{n+1}-1} D_k(\theta_n, \theta) \nabla_\theta Q^\theta(y_k, u_k) + \frac{1}{2\alpha_{n+1}} (\theta - \theta_n) \\
\overset{\theta=\theta_{n+1}}{\Leftrightarrow} 0 &= \frac{1}{2(t_{n+1} - t_n)} \sum_{k=t_n}^{t_{n+1}-1} D_k(\theta_n, \theta_{n+1}) \nabla_\theta Q^{\theta_{n+1}}(y_k, u_k) + \frac{1}{2\alpha_{n+1}} (\theta_{n+1} - \theta_n) \\
\Leftrightarrow \theta_{n+1} &= \theta_n + \frac{\alpha_{n+1}}{t_{n+1} - t_n} \sum_{k=t_n}^{t_{n+1}-1} D_k(\theta_n, \theta_{n+1}) \nabla_{\theta_{n+1}} Q^{\theta_{n+1}}(y_k, u_k).
\end{aligned}$$

■

Using the assumptions from lemma 4.6 we can now define the update formulas for the non-linear Q(0)-Learning which are given by

Definition 4.7. *(Batch Non-Linear Q(0)-Learning). Let $0 \leq n \leq B-1$, $(\alpha_n)_{n\in\mathbb{N}} \subset \mathbb{R}_{>0}$ and $\theta_0 \in \mathbb{R}^d$ be given then the update formula for batch non-linear $Q(0)$-Learning defined by*

$$\theta_{n+1} = \theta_n + \frac{\alpha_{n+1}}{t_{n+1} - t_n} \sum_{k=t_n}^{t_{n+1}-1} \mathfrak{D}_{k+1}(Q^{\theta_n}) \nabla_{\theta_n} Q^{\theta_n}(y_k, u_k) \tag{4.29}$$

where

$$\mathfrak{D}_{k+1}(Q^{\theta_n}) := \left[r(y_k, u_k) + Q^\theta(f(y_k, u_k), \pi^{\theta_n}(y_k)) \right] - Q^{\theta_n}(y_k, u_k)$$

More detail on the above definition can be found in ([23], section 5.4.5). If we now assume that the sequence $(\theta_n)_{n\in N}$ defined by (4.29) is convergent we get the following result

Theorem 4.8. *Let Q^θ be continuously differentiable for all $\theta \in \mathbb{R}^d$. Furthermore let $\nabla Q^\theta(y,u)$ be globally Lipschitz-continuous with a Lipschitz-constant independent of (y,u), $B = \infty$ and $(\alpha_n)_{n\in\mathbb{N}} \subset \mathbb{R}_{>0}$ such that*

$$\sum_{n=0}^{\infty} \alpha_n = \infty \text{ and } \sum_{n=0}^{\infty} \alpha_n^2 < \infty.$$

If we assume that $(\theta_n)_{n\in N}$ defined by (4.29) converges to $\theta_\infty \in \mathbb{R}^d$. Then follows

$$0 = \mathbb{E}_\omega \left[\mathfrak{D}_{k+1}(Q^{\theta_\infty}) \nabla_{\theta_n} Q^{\theta_n}(y_n, u_n) \right].$$

Remark 4.9. This theorem is generally really hard to proof which is dependent on the non-linear approximation function. It gets simpler if we assume a linear approximation space. For more details see ([23], proposition 5.10).

5 Stochastic Q-Learning

Although the main problem of this work is described in terms of deterministic Q-Learning we will use two essential techniques from stochastic Q-Learning to train our model. For that we introduce the analogous stochastic Q-Learning methodology.

We start with our dynamical system. The first step is that we change from control and state sequences to stochastic processes.

Definition 5.1. *Let $\mathsf{Y} \subset \mathbb{R}^n$ and $\mathsf{U} \subset \mathbb{R}^m$ be the either finite or countably infinite state and control space respectively. Let further $(\Omega, \mathscr{F}, P)$ be a probability space and $(\mathsf{Y}, d_{\mathsf{Y}})$ as well as $(\mathsf{U}, d_{\mathsf{U}})$ be measurable spaces. We then call*

$$Y := \{Y_t := Y(t) \,|\, t \geq 0\} \text{ and } U := \{U_t := U(t) \,|\, t \geq 0\}$$

the state and control process, where Y_t and U_t are Y and U-valued random variables respectively.

Just as in the deterministic case, our objective will be to find an optimal policy $\pi^* : \mathsf{Y} \to \mathsf{U}$, $\pi^*(Y(t)) = U(t)$ for $t \geq 0$ which provides us with an optimal control for a given state. With that we can define the so called Markov Decision Process or MDP for short. The following definition and remark is based on ([23], section 6.1 and 7.1).

Definition 5.2. *Let Y and U be two stochastic processes for which holds that*

$$\begin{aligned} &P\left(Y(k+1) = y' \,|\, Y(0), \ldots, Y(k), U(0), \ldots, U(k), Y(k) = y, U(k) = u\right) \\ &= P\left(Y(k+1) = y' \,|\, Y(k) = y, U(k) = u\right) =: P_u(y, y') \end{aligned} \tag{5.1}$$

for $k \geq 0$. Further let $N := \{N(k)\}_{k \geq 0}$ be an i.i.d sequence of $\mathbb{R}^l$-valued random variables called the disturbance and $F : \mathsf{Y} \times \mathsf{U} \times \mathsf{N} \to \mathsf{Y}$ is our system function for which holds

$$\begin{aligned} Y(k+1) &= F(Y(k), U(k), N(k+1)), \quad \text{for } k > 0 \\ Y(0) &\sim Y_0 \end{aligned} \tag{5.2}$$

where Y_0 is the initial distribution which is independent to N. We speak of a Markov decision process if there exists

1. *The transition probability is given by $P_u(y, y')$ for all $y, y' \in \mathsf{Y}$ and $u \in \mathsf{U}$.*
2. *A reward function $r : \mathsf{Y} \times \mathsf{U} \to \mathbb{R}$ and*
3. *An objective function given by the total reward*

$$J(x) := \sum_{k=0}^{\infty} \gamma^k \mathbb{E}\left[r(Y(k), U(k))\right] \tag{5.3}$$

where $0 < \gamma < 1$.

Remark 5.3. Similarly to the definition of a equilibrium state, we also want that our state process is stationary. Which means that there is an equilibrium measure P^e with probability mass function p^e such that the state process satisfies the following ergodic property. For that let $A \subset \mathsf{Y}$, $y \in \mathsf{Y}$ and $u \in \mathsf{U}$. We then have the one step transition probability defined as

$$P_u(y, A) = P\left(Y(k+1) \in A \,|\, Y(k) = y, U(k) = u\right)$$

and the j-step transition probability defined by

$$P_u^j(y, A) = P\left(Y(k+j) \in A \,|\, Y(k) = y, U(k) = u\right).$$

We say that the state process is ergodic if

$$\lim_{j \to \infty} P_u^j(y, A) = P^e(y, A) \tag{5.4}$$

for any $y \in \mathsf{Y}$ and $A \subset \mathsf{Y}$. This implies that the state process will be stationary if $Y(0) \sim P^e$. ■

In many practical applications will a policy function π not only depended on the current state but also on previous states, in addition to that we also want to apply the concept of exploration, which requires the policy to be disturbed by noise. For this we define the admissible inputs ([23], section 7.1.1) in the following way

Definition 5.4. *Let $(\Omega_\varepsilon, \mathscr{F}_\varepsilon, P_\epsilon)$ be a probability space where Ω_ε is a countable set. Further let ε be an i.i.d. sequence of Ω_ε-valued random variables. Further let the sequence ε be exogenous to the control system which means that*

$$\begin{aligned} &P\left(Y(k+1)=y', \varepsilon(k+1)=w' \,|\, Y^\varepsilon(0),\dots,Y^\varepsilon(k), U(0),\dots,U(k), Y(k)=y, U(k)=u\right) \\ &= P_u(y,y')\mu_\varepsilon(w') \end{aligned} \tag{5.5}$$

for all $k \geq 0$, $u \in \mathsf{U}$, $y, y' \in \mathsf{Y}$ and $w' \in \Omega_\varepsilon$. Furthermore is $Y^\varepsilon(k) = (Y(k), \varepsilon(k))$ and μ_ε is the probability mass function of $\varepsilon(1)$, since ε is an i.i.d. sequence. We call the control inputs admissible if they are given by a policy $\pi_k : \mathsf{X}^{k+1} \to \mathsf{U}$ of the form

$$U(k) = \pi_k(Y^\varepsilon(0),\dots,Y^\varepsilon(k)). \tag{5.6}$$

We further define the new transition probability which is dependent on ε as

$$P_u(y,y',w';\varepsilon) := P_u(y,y')\mu_\varepsilon(w'). \tag{5.7}$$

To simplify the notation we don't want to use definition 5.4, since the dependency of the noise and the previous states lies exclusively on the policy. For this we define the randomized policy ([23], section 7.1.1) in the following way

Definition 5.5. *Let ε be given as in definition 5.4. We say that the control input is defined by a randomized stationary policy $\hat{\pi}$ if*

$$\hat{\pi}\left(u \,|\, y\right) = P\left(U(k)=u \,|\, Y^\varepsilon(0),\dots,Y^\varepsilon(k-1), U(0),\dots,U(k-1), Y(k)=y\right). \tag{5.8}$$

The transition probability is then defined for $y, y' \in \mathsf{Y}$ as

$$P_{\hat{\pi}}(y,y') = \sum_u \hat{\pi}\left(u \,|\, y\right) P_u(y,y') \tag{5.9}$$

This allows us now to write the following conditional expectations

$$P_u V(y) := \mathbb{E}\left[\, V(Y(k+1)) \,|\, Y(k)=y, U(k)=u \,\right] = \sum_{y'} \hat{\pi}\left(u \,|\, y\right) P_u(y,y') V(y') \tag{5.10}$$

$$P_{\hat{\pi}} V(y) := \mathbb{E}\left[\, V(Y(k+1)) \,|\, Y(k)=y \,\right] = \sum_{y'} \sum_u \hat{\pi}\left(u \,|\, y\right) P_u(y,y') V(y') \tag{5.11}$$

where (5.10) describes the expected value of the next state in the value function, if the current state and current control are given. Furthermore describes (5.11) the expected value of the next state in the value function, if only the current state is given and the control is arbitrary.

Using definition 5.5 we can now also define a Markov decision process for the combined state and control process which is taken from ([23], chapter 9).

Definition 5.6. *Let $\mathsf{Z} \subset \mathbb{R}^{m+n}$ be either finite or countably infinite and let $(\mathsf{Z}, d_{\mathsf{Z}})$ be a measurable space. For a $\hat{\pi}$ randomized policy, the state process X has transition probability $P_{\hat{\pi}}$ defined by (5.9). The combined state and control process is then defined as*

$$Z = \{Z_t := Z(t) = (Y(t), U(t)) \,|\, t \geq 0\} \tag{5.12}$$

where Z_t is a Z-valued random variable. The transition probability for $z = (y,u), z' = (y',u') \in \mathsf{Z}$ is then defined by

$$T_{\hat{\pi}}(z,z') = P_u(y,y')\hat{\pi}\left(u' \,|\, y'\right). \tag{5.13}$$

The equilibrium measure T^e for the joined state and control process is then defined by

$$T^e(y,u) := P^e(y)\hat{\pi}\left(u \,|\, y\right) \tag{5.14}$$

Using this we can then define the Q-function and value function, using this combined process where the conditional expectation is also used.

Definition 5.7. *Let π be a given policy and $y \in \mathsf{Y}$. Then is the value function which applies π defined by*

$$V^{\pi}(y) = \sum_{k=0}^{\infty} \gamma^k \mathbb{E}\left[\, r(Z_k) \,|\, Y(0) = y \,\right] \quad , U(k) = \pi(Y(k)) \text{ for } k \geq 0. \tag{5.15}$$

Further let $y \in \mathsf{Y}$ and $u \in \mathsf{U}$, then is the Q-function which applies π defined by

$$Q^{\pi}(y,u) = \sum_{k=0}^{\infty} \gamma^k \mathbb{E}\left[\, r(Z_k) \,|\, Y(0) = y, U(0) = u \,\right] \quad , U(k) = \pi(Y(k)) \text{ for } k \geq 1. \tag{5.16}$$

We also want to formulate the stochastic equivalent to (3.8) and (3.9) using the Bellman optimality principle. This means that the optimal value function should satisfy the equation

$$V^*(y) = \min_{u \in \mathsf{U}} \left[r(y,u) + \gamma \sum_{y'} P_u(y,y') V^*(y') \right]. \tag{5.17}$$

This implies then that the optimal policy is given by

$$\pi^*(y) \in \operatorname*{argmin}_{u \in \mathsf{U}} \left[r(y,u) + \gamma \sum_{y'} P_u(y,y') V^*(y') \right]. \tag{5.18}$$

This also gives us the optimal Q-function defined by

$$Q^*(y,u) = r(y,u) + \gamma \sum_{y'} P_u(y,y') V^*(y') \tag{5.19}$$

and further we can also write the optimal value function as

$$V^*(y) = \min_{u \in \mathsf{U}} Q^*(y,u). \tag{5.20}$$

5.1 Stochastic Dynamical Programming

Now that we introduced the stochastic definitions for the policies, value function and Q-function, we are now able to reintroduce the temporal difference learning or TD(λ)-Learning algorithm and also the policy iteration algorithm for stochastic dynamical programming. We first start with the policy iteration algorithm which isn't that much different from the deterministic case. For simplicity we only going to look at the algorithm with a deterministic policy.

Let π_0 be an initial policy. To find a new iteration π_{n+1} we need to solve the following minimization problem

$$\pi_{n+1}(y) = \arg\min_{u \in \mathsf{U}} \left[\, r(y,u) + \gamma P_u V^{\pi_n}(y) \,\right] \quad \text{for } y \in \mathsf{Y} \tag{5.21}$$

where $P_u V^{\pi_n}(y)$ is defined by (5.10) for $y \in \mathsf{Y}$ and $u \in \mathsf{U}$. Like in the deterministic case we have two defining calculations which are necessary to obtain the new policy iteration. The first one is to find V^{π_n}. For that we are going to use again the Bellman's optimality principle, which means determine V^{π_n} from the following equation

$$V^{\pi_n}(y) = r(y, \pi_n(y)) + \gamma P_{\pi_n} V^{\pi_n}(y) \tag{5.22}$$

where P_{π_n} describes the transition probability in which the controls are obtained by the policy π_n. The other necessary calculation is the minimization in (5.22). Both of these can be also very expensive.

Just like in the deterministic case we want to show that the value function is decreasing after every policy iteration. Here we can use the projected Bellman Ansatz described by (4.10) and (4.11).

For that we introduce the concept of the advantage function. The policy iteration step defined by (5.22) can be also expressed in terms of the Q-function similarly to (5.19) which means

$$\pi_{n+1}(y) = \arg\min_{u \in \mathsf{U}} Q^{\pi_n}(y,u) \quad \text{for } y \in \mathsf{Y}.$$

Let $G : \mathsf{Y} \to \mathbb{R}$ be a function which is independent of u. We now want to estimate $Q^{\pi_n} - G$ instead of Q^π. This is allowed by the policy iteration formula since

$$\arg\min_{u \in \mathsf{U}} Q^{\pi_n}(y,u) = \arg\min_{u \in \mathsf{U}} \left(Q^{\pi_n}(y,u) - G(y)\right).$$

A reasonable choice for G would be the value function V^{π_n} since it minimizes the mean-square-error.

Remark 5.8. For $H := \left\{g : \mathsf{Y} \to \mathbb{R} \,\middle|\, \mathbb{E}\left[g(Y)^2\right] < \infty\right\}$ we want to approximate the random variable Z with $\mathbb{E}\left[Z^2\right] < \infty$ on H. The solution is the conditional expectation which is therefore defined by

$$\mathbb{E}\left[Z \,|\, Y\right] := g^* = \arg\min_{g \in H} \mathbb{E}\left[\left(Z - g(Y)\right)^2\right] \tag{5.23}$$

or equivalently formulated with the projection property

$$0 = \mathbb{E}\left[\left(Z - g^*(Y)\right) g(Y)\right] \quad \forall g \in H. \tag{5.24}$$

Notice that (5.24) looks similar to the Galerkin approach we used for the deterministic case. Instead of the Robbins-Monro algorithm which was used to solve equations of the form (5.24) in the deterministic case, we want to use the similar stochastic gradient descent method, which will solve (5.23). ■

Like in remark 5.8 we can then see that V^{π_n} which is defined as a conditional expectation in (5.15) solves the following optimization problem

$$\arg\min_{G \in \mathscr{G}} \mathbb{E}_{p^e}\left[\left(Q^{\pi_n}(Z_t) - G(Y_t)\right)^2\right]$$

where $\mathscr{G} := \left\{G : \mathsf{Y} \to \mathbb{R} \,\middle|\, \mathbb{E}_{p^e}\left[G(Y_t)^2\right] < \infty\right\}$. Using that we define the advantage function in the following way

Definition 5.9. *Let π be a policy $y \in \mathsf{Y}$ and $u \in \mathsf{U}$. We then call*

$$A^\pi(y,u) := Q^\pi(y,u) - V^\pi(y) \tag{5.25}$$

the advantage function.

Using the advantage function we can now represent the difference between two value functions, which are dependent on two different policies. The following proposition and proof is taken out of ([23], proposition 9.2)

Proposition 5.10. *Let π and $\tilde{\pi}$ be two different policies* then holds for $y \in \mathsf{Y}$

$$V^{\tilde{\pi}}(y) = V^\pi(y) + \mathbb{E}_{\tilde{\pi}}\left[\sum_{k=0}^{\infty} \gamma^k A^\pi(Z_k) \,\middle|\, Y(0) = y\right] \tag{5.26}$$

where the underlying policy for Z_k is $\tilde{\pi}$.

Proof. Using definition 5.9 we have for $y \in \mathsf{Y}$ and $k \geq 0$

$$\begin{aligned}\mathbb{E}_{\tilde{\pi}}\left[A^\pi(Z_k) \,|\, Y(0)y\right] &= \mathbb{E}_{\tilde{\pi}}\left[Q^\pi(Z_k) - V^\pi(Y_k) \,|\, Y(0) = y\right] \\ &= \mathbb{E}_{\tilde{\pi}}\left[r(Z_k) + \gamma V^\pi(Y_{k+1}) - V^\pi(Y_k) \,|\, Y(0) = y\right]\end{aligned}$$

With that follows for $y \in \mathsf{Y}$

$$\begin{aligned}
&\mathbb{E}_{\tilde{\pi}}\left[\sum_{k=0}^{\infty}\gamma^k A^{\pi}(Z_k)\,\middle|\, Y(0)=y\right]\\
&=\mathbb{E}_{\tilde{\pi}}\left[\sum_{k=0}^{\infty}\gamma^k\left(r(Z_k)+\gamma V^{\pi}(Y_{k+1})-V^{\pi}(Y_k)\right)\,\middle|\, Y(0)=y\right]\\
&=\mathbb{E}_{\tilde{\pi}}\left[\sum_{k=0}^{\infty}\gamma^k r(Z_k)+\sum_{k=0}^{\infty}\gamma^{k+1}V^{\pi}(Y_{k+1})-\gamma^k V^{\pi}(Y_k)\,\middle|\, Y(0)=y\right]\\
&=\mathbb{E}_{\tilde{\pi}}\left[\sum_{k=0}^{\infty}\gamma^k r(Z_k)+\sum_{k=0}^{\infty}\gamma^{k+1}V^{\pi}(Y_{k+1})-\gamma^k\left(\mathbb{E}_{\pi}\left[r(Z_k)\,|\,Y(k)=Y_k\right]+\gamma V^{\pi}(Y_{k+1})\right)\,\middle|\, Y(0)=y\right]\\
&=\mathbb{E}_{\tilde{\pi}}\left[\sum_{k=0}^{\infty}\gamma^k r(Z_k)-\sum_{k=0}^{\infty}\gamma^k\mathbb{E}_{\pi}\left[r(Z_k)\,|\,Y(k)=Y_k\right]\,\middle|\, Y(0)=y\right]\\
&=\mathbb{E}_{\tilde{\pi}}\left[-V^{\pi}(Y_0)+\sum_{k=0}^{\infty}\gamma^k r(Z_k)\,\middle|\, Y(0)=y\right]=V^{\tilde{\pi}}(y)-V^{\pi}(y)
\end{aligned}$$

■

We can now use 5.10 to show that the policy iteration formula (5.21) is decreasing the value function. For that let $\tilde{\pi}(y) := \pi_{n+1}(y) = \arg\min_{u\in\mathsf{U}} Q^{\pi_n}(y,u) = \arg\min_{u\in\mathsf{U}} A^{\pi_n}(y,u)$ and $\pi(y) := \pi_n(y)$. With that we get

$$\begin{aligned}
A^{\pi}(y,\tilde{\pi}(y)) &= \min_{u\in\mathsf{U}} A^{\pi}(y,u) = \min_{u\in\mathsf{U}}\left(Q^{\pi}(y,u)-V^{\pi}(y)\right) = -V^{\pi}(y)+\min_{u\in\mathsf{U}} Q^{\pi}(y,u)\\
&\leq -V^{\pi}(y)+Q^{\pi}(y,\pi(y)) = -V^{\pi}(y)+V^{\pi}(y)=0\\
&\Leftrightarrow \mathbb{E}_{\tilde{\pi}}\left[\sum_{k=0}^{\infty}\gamma^k A^{\pi}(Z_k)\,\middle|\, Y(0)=y\right]\leq 0
\end{aligned}$$

Hence using proposition 5.10 we showed that for all $y \in \mathsf{Y}$ follows that $V^{\tilde{\pi}}(y) \leq V^{\pi}(y) \Leftrightarrow V^{\pi_{n+1}}(y) \leq V^{\pi_n}(y)$. This concludes the stochastic policy iteration algorithm.

5.2 Stochastic Gradient Descent

Besides the Robbins-Monro algorithm [29] is the Stochastic Gradient Descent (SGD) method one of the most often used methods in practise to solve problems of the form (5.24). The following section leans on [8].

The optimization problem we are looking at, has the form

$$\theta^* = \arg\min_{\theta\in\mathbb{R}^d} F(\theta) \tag{5.27}$$

where $F(\theta) = \mathbb{E}\left[f(\theta,\varepsilon)\right]$, for $f : \mathbb{R}^d \times \Omega \to \mathbb{R}^d$ and Ω-valued random variable ε. Then is the stochastic gradient (SGD) method for a step size sequence $(\alpha_n)_{n\in\mathbb{N}} \subset \mathbb{R}_{>0}$ given by the iteration formula

$$\theta_{n+1} = \theta_n - \alpha_n \nabla_\theta f_{k_n}(\theta_n,\varepsilon) \tag{5.28}$$

where $f_{k_n}(\theta,\varepsilon) := f(\theta,\varepsilon_k)$ for $\varepsilon_k \in \{\varepsilon_i \,|\, i \in \mathbb{N}_{\leq N}\}$, which is a sample from the distribution which defines ε of size $N \in \mathbb{N}$.

Analogously to (5.28) we can also define a batch form of the SGD algorithm. Here we write F in the form

$$F(\theta) = \frac{1}{N}\sum_{k=1}^{N} f_k(\theta, \varepsilon). \tag{5.29}$$

where the iteration formula is then defined as

$$\theta_{n+1} = \theta_n - \frac{\alpha_n}{N}\sum_{k=1}^{N} \nabla_\theta f_k(\theta_n, \varepsilon). \tag{5.30}$$

We notice that (5.29) has exactly the same structure as the mean-squared Bellman error used in (4.23). In this work we want to focus on the batch SGD method for which we can define the following algorithm

Algorithm 5 Mini-Batch SGD

Require: Initial parameter $\theta_0 \in \mathbb{R}^d$, batch size $N \in \mathbb{N}$.

1: **for** n $= 0, 1 \ldots$ **do**
2: Sample a batch ε^n of size N from the distribution of ε.
3: Calculate $\nabla_\theta f_k(\theta_n, \varepsilon^n)$ for all $k \in \mathbb{N}_{\leq N}$.
4: Choose a steps size α_n.
5: Calculate $\theta_{n+1} = \theta_n - \frac{\alpha_n}{N}\sum_{k=1}^{N} \nabla_\theta f_k(\theta_n, \varepsilon^n)$.
6: **end for**

Now we want to see under which conditions F decreases using the mini-batch SGD iteration. The following is taken from ([8], lemma 4.2)

Lemma 5.11. *Let F be continuously differentiable and moreover let $\nabla_\theta F$ be Lipschitz-continuous with Lipschitz constant $L > 0$ i.e.*

$$\forall \theta, \theta' \in \mathbb{R}^d : \|\nabla_\theta F(\theta) - \nabla_\theta F(\theta')\|_2 \leq L\|\theta - \theta'\|_2.$$

Then does the following inequality hold for the mini-batch SGD iterates in (5.30) *and all $n \in \mathbb{N}$*

$$\mathbb{E}_{\varepsilon^n}\left[F(\theta_{n+1})\right] \leq -\alpha_n \nabla_\theta F(\theta_n)^\top \mathbb{E}_{\varepsilon^n}\left[g(\theta_n, \varepsilon^n)\right] + \frac{L\alpha_n^2}{2}\mathbb{E}_{\varepsilon^n}\left[\|g(\theta_n, \varepsilon^n)\|_2^2\right] \tag{5.31}$$

where $g(\theta_n, \varepsilon^n) := \frac{1}{N}\sum_{k=1}^{N} \nabla_\theta f_k(\theta_n, \varepsilon^n)$.

Proof. Let $\theta, \theta' \in \mathbb{R}^d$. Then follows using the fundamental theorem of integration and the assumption that $\nabla_\theta F$ is Lipschitz-continuous that

$$\begin{aligned}
F(\theta) &= F(\theta') + \int_0^1 \nabla_\theta F\left(\theta' + t\left(\theta - \theta'\right)\right)^\top \left(\theta - \theta'\right)\, dt \\
&= F(\theta') + \nabla\theta F(\theta')^\top \left(\theta - \theta'\right) + \int_0^1 \left(\nabla_\theta F\left(\theta' + t\left(\theta - \theta'\right)\right) - \nabla_\theta F(\theta')\right)^\top \left(\theta - \theta'\right)\, dt \\
&\leq F(\theta') + \nabla\theta F(\theta')^\top \left(\theta - \theta'\right) + \int_0^1 Lt\|\theta - \theta'\|_2^2\, dt \\
&= F(\theta') + \nabla\theta F(\theta')^\top \left(\theta - \theta'\right) + \frac{L}{2}\|\theta - \theta'\|_2^2.
\end{aligned}$$

Using this we get for the sequence $(\theta_n)_{n\in\mathbb{N}}$ the inequality

$$\begin{aligned} F(\theta_{n+1}) - F(\theta_n) &\leq \nabla_\theta F(\theta_n)^\top (\theta_{n+1} - \theta_n) + \frac{L}{2}\|\theta_{n+1} - \theta_n\|_2^2 \\ &\leq -\alpha_n \nabla_\theta F(\theta_n)^\top g(\theta_n, \varepsilon^n) + \frac{L\alpha_n^2}{2}\|g(\theta_n, \varepsilon^n)\|_2^2. \end{aligned}$$

If we now apply the expectation on both sites and use that θ_n by definition does not depend on θ^n we get

$$\begin{aligned} \mathbb{E}_{\varepsilon^n}\left[F(\theta_{n+1}) - F(\theta_n)\right] &\leq -\alpha_n \mathbb{E}_{\varepsilon^n}\left[\nabla_\theta F(\theta_n)^\top g(\theta_n, \varepsilon^n)\right] + \frac{L\alpha_n^2}{2}\mathbb{E}_{\varepsilon^n}\left[\|g(\theta_n, \varepsilon^n)\|_2^2\right] \\ &= -\alpha_n \nabla_\theta F(\theta_n)^\top \mathbb{E}_{\varepsilon^n}\left[g(\theta_n, \varepsilon^n)\right] + \frac{L\alpha_n^2}{2}\mathbb{E}_{\varepsilon^n}\left[\|g(\theta_n, \varepsilon^n)\|_2^2\right]. \end{aligned}$$

■

From the inequality (5.31) we notice that we need to assure to limit the second moments of $g(\theta_n, \varepsilon^n)$. Using that we obtain can obtain the following lemma, which was taken from ([8], lemma 4.4)

Lemma 5.12. *Let F be continuously differentiable and moreover let $\nabla_\theta F$ be Lipschitz-continuous with Lipschitz-constant $L > 0$. If in addition to this holds for $(\theta_n)_{n\in\mathbb{N}}$ that*

1. *$\exists V \subset \mathbb{R}^d$: open, such that $(\theta_n)_{n\in\mathbb{N}} \subset V$ and F is bounded from below on V.*
2. *$\exists \mu_g \geq \mu > 0$ such that for all $n \in \mathbb{N}$*

$$\nabla_\theta F(\theta_n)^\top \mathbb{E}_{\varepsilon^n}\left[g(\theta_n, \varepsilon^n)\right] \geq \mu\|\nabla_\theta F(\theta_n)\|_2^2 \quad \text{and} \quad \|\mathbb{E}_{\varepsilon^n}\left[g(\theta_n, \varepsilon^n)\right]\|_2 \leq \mu_g\|\nabla_\theta F(\theta_n)\|_2.$$

3. *$\exists M, M_V \geq 0$ such that for all $n \in \mathbb{N}$*

$$\mathbb{V}ar\left[g(\theta_n, \varepsilon^n)\right] \leq M + M_V\|\nabla_\theta F(\theta_n)\|_2^2.$$

Then follows

$$\mathbb{E}_{\varepsilon^n}\left[F(\theta_{n+1}) - F(\theta_n)\right] \leq -\left(\mu - \frac{\alpha_n L}{2}\left(M_V + \mu_g^2\right)\right)\alpha_n\|\nabla_\theta F(\theta_n)\|_2^2 + \frac{\alpha_n^2 LM}{2}. \tag{5.32}$$

■

This lemma guarantees now that the mini-batch SGD sequence decreases in expectation for appropriately chosen step sizes. The choice of step width depends heavily on the convexity of the objective function. Notice further that the left-hand side of (5.32) only depends on θ_n and not on any previous iterates.

Common practise in deep learning is to use modifications of the classic SGD or mini-batch SGD algorithm. Two popular ones are the SGD algorithm with momentum and the accelerated SGD method as well as the Adam optimizer, which is also based on the SGD method. Since in this work we use the Adam optimizer to minimize the loss function of our neural network, we want to introduce the Adam algorithm which is build upon its predecessors RMSProp and AdaGrad. Since the application of neural networks is a very broad field, its often the case that the algorithms that minimize the associated loss function are more tried and tested in practice and less supported by theory. But we will also introduce some general conditions under which the loss decreases.

5.3 Adam optimizer

This section is orientated on [17] and we start from the analogous situation described by (5.27). The Adam optimizer will then update the iteration variable θ_n by using exponentially moving averages to estimate the first and second moment of the gradients of f. The exponential moving averages are realized in the following way

Definition 5.13. *Let $\kappa \in [0,1)$ be the exponential decay rate. Furthermore let $(a_n)_{n\in\mathbb{N}}$ be a general sequence (scalars, vectors, etc.). Then defines the sequence $(s_n)_{n\in\mathbb{N}}$ for $s_0 := a_0$ and*

$$s_n := (1-\kappa)a_n + \kappa s_{n-1}$$

for $n > 0$ the exponentially moving average.

Its called this way because the earlier we are in the sequence $(a_n)_{n\in\mathbb{N}}$, the lower is the assigned weight. This can be seen if we assume that our sequence $(a_n)_{n\in\mathbb{N}}$ is finite and has $N \in \mathbb{N}$ elements and we solve the given recursion which results in

$$s_N = (1-\kappa)\sum_{i=1}^{N} \kappa^{N-i} a_i + \kappa^N a_0.$$

We then define the Adam optimizer algorithm in the following way

Algorithm 6 Mini-Batch Adam optimizer algorithm

Require: Step size $\alpha \in \mathbb{R}_{>0}$, Exponential decay rates $\kappa_1, \kappa_2 \in [0,1)$, $\delta > 0$, batch size $N \in \mathbb{N}$, $n = 0$ and initial parameters $\theta_0 \in \mathbb{R}^d$, $v_0 = 0 \in \mathbb{R}^d$ and $m_0 = 0 \in \mathbb{R}^d$.

1: **while** θ_n not converged **do**
2: $n = n + 1$.
3: Sample a batch ε^n of size N from the distribution of ε.
4: Calculate $\nabla_\theta f_k(\theta_n, \varepsilon^n)$ for all $k \in \mathbb{N}_{\leq N}$.
5: Calculate gradients $g_n = \dfrac{1}{N}\sum_{k=1}^{N} \nabla_\theta f_k(\theta_n, \varepsilon^n)$.
6: Update biased first moment estimate $m_n = \kappa_1 m_{n-1} + (1-\kappa_2) g_n$.
7: Update biased second moment estimate $v_n = \kappa_1 v_{n-1} + (1-\kappa_2) g_n^{\circ 2}$.
8: Update bias-corrected first moment estimate $\hat{m}_n = \dfrac{m_n}{1-\kappa_1^n}$.
9: Update bias-corrected second moment estimate $\hat{v}_n = \dfrac{v_n}{1-\kappa_2^n}$.
10: Calculate $\theta_n = \theta_{n-1} - \alpha\, \hat{m}_n \oslash \left(\hat{v}_n^{\circ \frac{1}{2}} + \delta \mathbf{1}_d\right)$.
11: **end while**

where $(\cdot)^{\circ 2}$, $(\cdot)^{\circ \frac{1}{2}}$ and $\oslash$ are component-wise square, square-root and division respectively and $\mathbf{1}_d$ is the d-dimensional vector containing only ones.

Since the algorithm initializes the estimates for the first and second moments with zero, they are naturally biased towards zero. This needs to be corrected and can be done by solving the recursion equation, from the moving exponential averaging and calculate the expected value, in order to find the relation between the estimates and the exact moments. From this relation we then derive a scalar by which we can multiply the estimates. For further details on this explanation for the bias correction we refer to ([17], section 3, p. 3). A more intuitive explanation comes from the field of signal processing. We want to demonstrate that for the estimate of the fist moment, the calculation will work analogously for the estimate of the second moment.

Here we assume that the sequence for the gradients g_n can be written as

$$g_n = m_n + \nu_n, \tag{5.33}$$

where m_n is our deterministic estimate for the gradient (also called low frequency trend component in signal processing) and ν_n describes i.i.d. zero mean noise (referred as broadband component in signal processing). If we now take the expected value in both sides of (5.33) we get that $\mathbb{E}[g_n] = m_n$.

Or in other words we want our estimate to minimize the function $J(a) = \mathbb{E}\left[(g_n - a)^2\right]$ which is then solved by

$$\min_a J(a) \Leftrightarrow a^* = \mathbb{E}\left[g_n\right], \tag{5.34}$$

where a^* is the minimizer which was obtained by taking the derivative of J with respect to a. In most cases we are not able to calculate the expected value defining J exactly, which means that we are going to use an approximation for J. This is where the exponential decay rate comes in to play. For that we define the approximation $\hat{J}$ for $\kappa_1 \in (0,1)$ in the following way

$$\hat{J}(a) = \sum_{i=1}^{n} \kappa_1^{n-i} \left(g_i - a\right)^2. \tag{5.35}$$

If we now take the derivative of $\hat{J}$ with respect to a we get

$$\frac{d}{da}\hat{J}(a) = -2\sum_{i=1}^{n} \kappa_1^{n-i}\left(g_i - a\right) = 0 \quad \Leftrightarrow \quad \sum_{i=1}^{n} \kappa_1^{n-i} a = \sum_{i=1}^{n} \kappa_1^{n-i} g_i$$

$$\Leftrightarrow \quad a = \frac{\sum\limits_{i=1}^{n} \kappa_1^{n-i} g_i}{\sum\limits_{i=1}^{n} \kappa_1^{n-i}} \quad \Leftrightarrow \quad a = \frac{1-\kappa_1}{1-\kappa_1^n}\sum_{i=1}^{n} \kappa_1^{n-i} g_i,$$

where we used the formula for the finite geometric series to obtain $\sum_{i=1}^{n} \kappa_1^{n-i} = \frac{1-\kappa_1^n}{1-\kappa_1}$. Since $\hat{J}$ is quadratic and hence convex we get that $a^* := a$ is the minimum of $\hat{J}$. From this we immediately see that

$$\hat{m}_n = a^* = \frac{1-\kappa_1}{1-\kappa_1^n}\sum_{i=1}^{n} \kappa_1^{n-i} g_i = \frac{m_n}{1-\kappa_1^n}.$$

This shows that the update formula for the bias-corrected estimate, of the first moment from the Adam optimizer algorithm, is the best estimate with respect to the mean squared error. The estimator $\hat{m}_n$ is then by design unbiased but unfortunately not asymptotically stable, since the variance of $\hat{m}_n$ is given by

$$\begin{aligned}
\mathbb{V}ar\left[\hat{m}_n\right] = \mathbb{C}ov\left[\hat{m}_n, \hat{m}_n\right] &= \mathbb{C}ov\left[\frac{1-\kappa_1}{1-\kappa_1^n}\sum_{i=1}^{n} \kappa_1^{n-i} g_i, \frac{1-\kappa_1}{1-\kappa_1^n}\sum_{j=1}^{n} \kappa_1^{n-j} g_j\right] \\
&= \left(\frac{1-\kappa_1}{1-\kappa_1^n}\right)^2 \sum_{i,j=1}^{n} \kappa_1^{n-i}\kappa_1^{n-j}\mathbb{C}ov\left[g_i, g_j\right] = \left(\frac{1-\kappa_1}{1-\kappa_1^n}\right)^2 \sum_{i=1}^{n} \kappa_1^{2(n-i)}\mathbb{C}ov\left[g_i, g_i\right] \\
&= \sigma_g^2\left(\frac{1-\kappa_1}{1-\kappa_1^n}\right)^2 \sum_{i=1}^{n} \kappa_1^{2(n-i)} = \sigma_g^2\left(\frac{1-\kappa_1}{1-\kappa_1^n}\right)^2 \frac{1-\kappa_1^{2n}}{1-\kappa_1^2} = \sigma_g^2 \frac{1-\kappa_1}{1+\kappa_1}\frac{1+\kappa_1^n}{1-\kappa_1^n},
\end{aligned}$$

which does not tend to zero for $n \to \infty$. But if we choose κ_1 close to 1, we can keep the variance as close to zero as desired. Further context from the signal processing perspective can be found in ([26], ch. 6)

We now want to analyse the update formula for the parameter θ_n. A practical detail here is that the vector $\delta \mathbf{1}_d$ is used in case some elements of the vector $\hat{v}_k$ are too close to zero. For that reason we are going to ignore that for now. The important aspect of the update formula, is that the step which is taken can be carefully controlled. For that we look at the i-th component from the update rule. We then get

$$\alpha\left|\frac{\hat{m}_n^i}{\sqrt{\hat{v}_n^i}}\right| \leq \begin{cases} \alpha\dfrac{1-\kappa_1}{\sqrt{1-\kappa_2}} & 1-\kappa_1 > \sqrt{1-\kappa_2} \\ \alpha & \text{otherwise} \end{cases}. \tag{5.36}$$

This follows firstly because for the exact first and second moments holds that

$$0 \leq \mathbb{E}\left[\left(g^2 - \mathbb{E}\left[g\right]\right)^2\right] = \mathbb{E}\left[g^2\right] - \mathbb{E}\left[g\right]^2 \Leftrightarrow \mathbb{E}\left[g\right]^2 \leq \mathbb{E}\left[g^2\right]$$

$$\Leftrightarrow \mathbb{E}\left[g\right] \leq \sqrt{\mathbb{E}\left[g^2\right]} \Leftrightarrow \left|\frac{\mathbb{E}\left[g\right]}{\sqrt{\mathbb{E}\left[g^2\right]}}\right| \leq 1.$$

The first case in (5.36) happens only rarely for highly sparse gradients where every previous gradient step was zero except for the current step. We can therefore say that the gradient step is effectively bounded by α. Depending on the information we have for a practical problems like a initial distribution of the parameter θ, we can then choose the right scale for α, especially since the update step is by definition invariant to scaling. Generally speaking we want a small value of (5.36), since then there is a greater uncertainty that the gradient estimate $\hat{m}_n$ is closer to the true gradient direction.

Tried and tested hyper-parameter choices for the Adam optimizer are

$$\alpha = 10^{-3},\ \kappa_1 = 0.9,\ \kappa_2 = 0.999 \text{ and } \varepsilon = 10^{-8} \tag{5.37}$$

We now want to give some convergence results, under which the Adam optimizer converges. For that we need an error measure, which will be the so called average regret defined by

Definition 5.14. *Let the function f from* (5.27) *and* (5.30) *be convex. Then is the average regret for $N \in \mathbb{N}$ steps given by*

$$R(N) = \sum_{k=1}^{N} \left[f_k(\theta^*, \varepsilon) - f_k(\theta_k, \varepsilon)\right],$$

where θ^* is the optimal parameter.

The regret describes basically the cumulative difference between the optimal parameter evaluated on f for different time steps and the Adam parameters also evaluated on f, which the generated sequence misses out on. With that we can now formulate the necessary conditions under which the regret converges to zero

Theorem 5.15. *Let the function f defined in* (5.27) *and* (5.30) *be convex and assume that the gradients for f_k are bounded such that*

$$\exists G, G_\infty > 0,\ \forall k \in \mathbb{N}_{\leq N},\ \forall \theta \in \mathbb{R}^d :\ \left\|f_k(\theta, \varepsilon)\right\|_2 \leq G \text{ and } \left\|f_k(\theta, \varepsilon)\right\|_\infty \leq G_\infty$$

Further assume that the difference between the sequence generated by the Adam algorithm is bounded such that

$$\exists D, D_\infty > 0,\ \forall m, n \in \{0, \ldots, N\} :\ \left\|\theta_n - \theta_m\right\|_2 \leq D \text{ and } \left\|\theta_n - \theta_m\right\|_\infty \leq D_\infty.$$

Choose the exponential decay rates as $\kappa_1, \kappa_2 \in [0, 1)$ such that $\gamma := \frac{\kappa_1^2}{\sqrt{\kappa_2}} < 1$. Furthermore let the step sizes and the exponential decay for the first moment be decaying sequences such that

$$\alpha_n = \alpha/\sqrt{n} \text{ and } \kappa_{1,n} = \kappa_1 \lambda^{n-1}, \lambda \in (0, 1).$$

Then follows for every $N \geq 1$ that $R(N)$ is bounded by

$$\frac{D^2}{2\alpha(1+\kappa_1)} \sum_{i=1}^{d} \sqrt{N \hat{v}_N^i} + \frac{\alpha(1-\kappa_1) G_\infty}{(1-\kappa_1)\sqrt{1-\kappa_2}(1-\gamma)^2} \sum_{i=1}^{d} \left\|g_{1:N,i}\right\|_2 + \sum_{i=1}^{d} \frac{D_\infty^2 G_\infty \sqrt{1-\kappa_2}}{2\alpha(1-\kappa_1)(1-\lambda)^2},$$

where $g_{1:N,i} = \left(g_1^i, \ldots, g_N^i\right)^\top$ describes a length N vector which contains the i-th component of $g_n = \nabla_\theta f_n(\theta_n, \varepsilon)$ for all $n \in \mathbb{N}_{\leq N}$.

Proof. For a proof see ([17], Theorem 4.1). ■

Using this you can show that the average regret converges in the following way

Corrolary 5.16. *Under the assumptions from 5.15 follows for the average regret for every $N \geq 1$ that*

$$\frac{R(N)}{N} = \mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$$

Proof. A proof can be found in ([17], Corollary 4.2). ■

5.4 Deep Q-Learning for Stochastic Approximation

Similar to the deterministic case we also want to use the DQN-algorithm with the stochastic gradient descent or the Adam optimizer. Here we want to use the loss function

$$L(\theta_k) = \mathbb{E}\left[\mathfrak{D}_k(Q^{\theta_k})^2\right] = \mathbb{E}\left[\left(r(y_k,u_k) + \gamma \max_{u\in \mathsf{U}} Q^{\theta_k}(f(y_k,u_k),u) - Q^{\theta_k}(y_k,u_k)\right)^2\right], \tag{5.38}$$

where Q^θ is the approximation of the optimal Q-function Q^* which can be a linear function or in most cases a neural network. By taking the derivative with respect to θ_k we get

$$\nabla_{\theta_k} L(\theta_k) = \mathbb{E}\left[\left(r(y_k,u_k) + \gamma \max_{u\in \mathsf{U}} Q^{\theta_k}(f(y_k,u_k),u) - Q^{\theta_k}(y_k,u_k)\right) \nabla_{\theta_k} Q^{\theta_k}(y_k,u_k)\right]. \tag{5.39}$$

We then seek for θ_k such that $\nabla_{\theta_k} L(\theta_k) = 0$ by using the stochastic gradient descent algorithm or the Adam optimizer. The algorithm is then similar to the one before.

Algorithm 7 Deep Q-Learning with Experience Replay using SGD or Adam

Require: Replay set capacity $N \in \mathbb{N}$, Number of episodes $M \in \mathbb{N}$, Terminal time step $T \in \mathbb{N}$, $t = 0$ and initial state $y_t = y_0$.

1: Initialize the replay set $\mathfrak{R}$ with capacity N.
2: Initialize the neural network Q^θ with random or zero weights.
3: **for** episode $= 1$ to M **do**
4: **for** $t = 0$ to T **do**
5: Given a probability ε set $u_t = \begin{cases} \text{Choose a random policy} \\ \arg\min\limits_{u\in U} Q^\theta(y_t,u) \end{cases}$.
6: Determine incurred reward $r_t = r(y_t,u_t)$.
7: Apply control u_t to obtain $y_{t+1} = f(y_t,u_t)$.
8: Save experience sample $(y_t,u_t,r_t,y_{t+1}) \in \mathfrak{R}$.
9: Sample randomized mini-batch from replay set $\mathfrak{R}$.
10: Apply SGD or Adam on (5.39) using sampled mini-batch to update the parameter θ.
11: **end for**
12: **end for**

5.5 Double Q-Learning

The classical approach in calculating the temporal difference target is given by

$$r(y_k,u_k) + \gamma \max_{u\in \mathsf{U}} Q^{\theta_k}(f(y_k,u_k),u),$$

which uses a single estimator function Q^{θ_k}. The following is orientated on [37], which introduced this methodology first. To show why using a single estimator can lead to overestimations of the chosen control value, we more generally consider $N \in \mathbb{N}$ random $\mathbb{R}$-valued variables $X_1,\dots,X_N$ and are essentially interested in the maximum expected value given by

$$\max_i \mathbb{E}[X_i]. \tag{5.40}$$

To approximate this value we want to use an estimator B_i which estimates the variable X_i for all $i \in \{1,\dots,N\}$, the next step would be to use the a sample mean to estimate the value of $\mathbb{E}[B_i]$. To do that we can take $M_i \in \mathbb{N}$ samples $x_1^i,\dots,x_{M_i}^i \in \mathbb{R}$ from each random variable X_i for all $i \in \{1,\dots,N\}$, where we additionally need to assume that the data points in each sample are identically distributed and independent. If we have that, we can use the sample average

$$\hat{\mu}_i = \frac{1}{M_i}\sum_{j=1}^{M_i} x_j^i$$

as an unbiased estimate of $\mathbb{E}[B_i]$, hence is $\max_i \hat{\mu}_i$ an unbiased estimate of $\mathbb{E}[\max_i B_i]$ but unfortunately not an unbiased estimate for $\max_i \mathbb{E}[X_i]$, which is stated more generally by the following

Lemma 5.17. *Let $B_1, \dots, B_N$ be the estimators of the random $\mathbb{R}$-valued variables $X_1, \dots, X_N$ such that the estimators are conditionally unbiased which means it holds that*

$$\mathbb{E}\left[B_i \,|\, X_1 = x_1, \dots, X_N = x_n\right] = x_i \quad \forall i \in \{1, \dots, N\}$$

and for all realizations $x_1, \dots, x_n \in \mathbb{R}$. Let $i^* \in \{1, \dots, N\}$ be the index for which holds that $B_{i^*} = \max_i B_i$. Then follows that

$$\mathbb{E}\left[X_{i^*} - B_{i^*}\right] \leq 0$$

Proof. A proof can be found in ([33], Proposition 1). ■

The main point of this is that this overestimation can have negative impact on the Q-Learning algorithm. To now mitigate this issue we want to compare this single estimator approach to the Double Q-Learning methodology. For that we define the estimators B_i^1 and B_i^2 for $i \in \{1, \dots, N\}$. We then take analogously to before $M_i \in \mathbb{N}$ samples $x_1^i, \dots, x_{M_i}^i \in \mathbb{R}$ from each random variable X_i for all $i \in \{1, \dots, N\}$, which fulfill the same conditions as before. We then partition each sample disjointly into two sets S_i^1 and S_i^2 of equal size, which then enables us to calculate the sample averages $\hat{\mu}_i^1$ over S_i^1 and $\hat{\mu}_i^2$ over S_i^2 for $i \in \{1, \dots, N\}$. In addition to that let

$$\hat{I}_1 := \left\{ j \in \mathbb{N} \,\middle|\, \hat{\mu}_j^1 = \max_i \hat{\mu}_i^1 \right\}$$

and choose the estimator $B_{i^*}^2$ for $i^* \in \hat{I}_1$. If there is more then one we pick one randomly. Then follows that $\hat{\mu}_{i^*}^2$ is an unbiased estimate of $\mathbb{E}\left[\max_i B_i^2\right]$. At first it seems that we just created the same situation as with the single estimator but the following shows why we now get an underestimation of $\max_i \mathbb{E}\left[X_i\right]$.

Lemma 5.18. *Let $X_1, \dots, X_N$ be random variables and let B_i^1 and B_i^2 be unbiased estimates for $i \in \{1, \dots, N\}$. Let $i^* \in \hat{I}_1$, then follows*

$$\mathbb{E}\left[B_{i^*}^2\right] = \mathbb{E}\left[X_{i^*}\right] \leq \max_i \mathbb{E}\left[X_i\right].$$

Furthermore holds the inequality strictly if and only if $\mathbb{P}\left(i^* \notin \mathcal{M}\right) > 0$, where

$$\mathcal{M} := \left\{ j \in \mathbb{N} \,\middle|\, \mathbb{E}\left[X_j\right] = \max_i \mathbb{E}\left[X_i\right] \right\}.$$

Proof. For a proof see ([37], Lemma 1). ■

This suggests that we should change the way we update the Q-values which we can see in the following algorithm sketch

Algorithm 8 Basic Double Q-Learning

```
Initialize two Q-function approximations Q^1, Q^2 and set y = y_0 as initial state.
while 1 do
    Choose control u with regards to Q^1(y, ·) and Q^2(y, ·).
    Get reward r and next state y'.
    Choose randomly which Q-function should be updated.
    if Q^1 is updated then
        Calculate u* = arg max_u Q^1(y', u).
        Update Q^1(y, u) ← Q^1(y, u) + α (r + γ Q^2(y', u*) − Q^1(y, a)).
    end if
    if Q^2 is updated then
        Calculate u* = arg max_u Q^2(y', u).
        Update Q^2(y, u) ← Q^2(y, u) + α (r + γ Q^1(y', u*) − Q^2(y, a)).
    end if
    y ← y'.
end while
```

This algorithm converges under similar conditions like the regular Q-Learning algorithm, for more details see ([37], Theorem 1).

In contrast to the basic Double Q-Learning algorithm, we have to update two sets of parameters, $\theta^- \in \mathbb{R}^d$ for the target network and $\theta \in \mathbb{R}^d$ which serves as a copy of the target network. The main difference is that we update θ by calculating the temporal difference target on the target network using

$$y_i' = r_i + \gamma Q^{\theta^-}(y_{t+1}, u_i')$$

but use the copy of the target network to make the control selection by calculating

$$u_i' = \arg\max_u Q^{\theta}(y_{i+1}, u),$$

then to determine the update of θ we minimize a quadratic loss function between y_i' and $Q^{\theta}(y_i, u_i)$. Another important point is the way in which the parameters of the target network get updated. We do it by using the so called polyak averaging where the update formula is given by

$$\theta^- = (1-\alpha)\theta^- + \alpha\theta$$

for $\alpha \ll 1$. Here are the target parameters quiet restricted to changing, which proved itself to help with the stability of the learning process as shown in many practical use cases, see e.g. [38]. This is combined in the following algorithm which was first introduced in [38]. Notice that we didn't add any exploration strategy to keep it simple.

Algorithm 9 Deep-Double Q-Learning

Require: Replay set capacity $N \in \mathbb{N}$, Number of episodes $M \in \mathbb{N}$, Terminal time step $T \in \mathbb{N}$, $\alpha \ll 1$ and $t = 0$.

1: Initialize the replay set $\mathfrak{R}$ with capacity N.
2: Initialize the neural network Q^{θ} with zero weights.
3: Initialize the target neural network Q^{θ^-} with $\theta^- = \theta$.
4: **for** episode $= 1$ to M **do**
5: Randomly sample initial state-control pair (y_0, u_0).
6: **for** $t = 0$ to T **do**
7: Determine incurred reward $r_t = r(y_t, u_t)$.
8: Apply control u_t on the state y_t to obtain y_{t+1}.
9: Save experience sample $(y_t, u_t, r_t, y_{t+1}) \in \mathfrak{R}$.
10: Sample randomized mini-batch from replay set $\mathfrak{R}$.
11: Calculate the temporal difference target for the target network for all elements in the mini-batch by

$$y_i' = r_i + \gamma Q^{\theta^-}(y_{t+1}, u_i'),$$

where $u_i' = \arg\max_u Q^{\theta}(y_{i+1}, u)$.

12: Apply Adam on the loss function $\sum_i (y_i' - Q^{\theta}(y_i, u_i))^2$ using the sampled mini-batch to update the parameter of the Q^{θ} network.
13: Update the target network parameters $\theta^- = (1-\alpha)\theta^- + \alpha\theta$.
14: **end for**
15: **end for**

6 Hamilton-Jacobi Deep Q-Learning

The results and statements in this section are based on ([16], Section 2).

Consider the following dynamical system given by the differential equation

$$\begin{aligned} y'(t) &= f(y(t), u(t)) \quad \text{for } t \in \mathbb{R}_{>0}, \\ y(0) &= y_0 \in \mathbb{R}^n \end{aligned} \tag{6.1}$$

where the system function is given by the vector field $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$. Here we have the system state function $y(t) \in \mathbb{R}^n$ and the control function $u(t) \in \mathbb{R}^m$.

We can get the existence of solutions of 6.1 if we for example employ the theorem of Picard & Lindelöf ([11], Theorem 16.1 & 16.4), which gives us the existence of exactly one solution $\bar{y} \in C^1(\mathbb{R}_{\geq 0}, \mathbb{R}^n)$, if f is Lipschitz-continuous. With that we can now define the optimal control problem

$$\begin{aligned} &\sup_{u \in U} J(u) := \int_0^\infty e^{-\gamma t}\, r(y(t), u(t))\, dt \\ &s.t. \quad y(0) = y_0 \text{ and } y'(t) = f(y(t), u(t)) \text{ for } t \in \mathbb{R}_{>0} \end{aligned} \tag{6.2}$$

where U is the control space, $r : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is the reward function and $\gamma > 0$ the discount factor.

6.1 Continuous Hamilton-Jacobi-Bellman-Equation

In order to solve this problem using the Q-Learning approach we need to find the optimal Q-function of (6.2). Which is defined by

$$Q^*(\tilde{y}, \tilde{u}; t) := \sup_{u \in U} \left\{ \int_t^\infty e^{-\gamma(s-t)} r(y(s), u(s))\, ds \,\middle|\, y(t) = \tilde{y}, u(t) = \tilde{u} \right\}. \tag{6.3}$$

The control space would usually be chosen as $U = \{u : \mathbb{R}_{\geq 0} \to \mathbb{R} \,|\, u \text{ measurable}\}$. But the problem is that then the control function does not have any influence on the Q-function.

Remark 6.1. Let $U = \{u : \mathbb{R}_{\geq 0} \to \mathbb{R}^m \,|\, u \text{ measurable }\}$. Then the optimal Q-function Q^* coincides with the optimal value function which means that $\forall \tilde{u} \in \mathbb{R}^m$:

$$Q^*(\tilde{y}, \tilde{u}; t) = \sup_{u \in U} \left\{ \int_t^\infty e^{-\gamma(s-t)} r(y(s), u(s))\, ds \,\middle|\, y(t) = \tilde{y} \right\}$$

for a proof of this statement refer to ([16], Proposition 1).

For that reason we want to restrict the control space to Lipschitz continuous functions, which essentially implies also that the first derivative of a control function is bounded by the Lipschitz constant.

$$U = \{u : \mathbb{R}_{\geq 0} \to \mathbb{R}^m \,|\, u \text{ measurable}, \exists L > 0 : |u'|_2 \leq L\}. \tag{6.4}$$

This assumption will play a huge role, in the way in which we will determine the next control, in the developed Q-Learning algorithm.

In order to determine the optimal Q-Function Q^*, we will use the fundamental principle of dynamical programming also called the Bellman optimality principle. This enables us under some additional assumptions on Q^* to find a partial differential equation similar to the Hamilton-Jacobi-Bellman equation which acts as as optimality condition.

Theorem 6.2. *Let the optimal Q-function be continuously differentiable and assume that f and r are uniformly bounded and Lipschitz continuous. Then holds*

$$\gamma Q^*(\tilde{y}, \tilde{u}; t) - \nabla_{\tilde{y}} Q^*(\tilde{y}, \tilde{u}; t) \cdot f(\tilde{y}, \tilde{u}) - L|\nabla_{\tilde{u}} Q^*(\tilde{y}, \tilde{u}; t)| - r(\tilde{y}, \tilde{u}) = 0 \tag{6.5}$$

for all $t \in \mathbb{R}_{\geq 0}$ and $(\tilde{y}, \tilde{u}) \in \mathbb{R}^n \times \mathbb{R}^m$.

Proof. Let $g(s,t) := e^{-\gamma(s-t)} r(y(s), u(s))$. Since $e^{-\gamma h} g(s, t+h) = g(s,t)$ we can rewrite the Q-function in the following way

$$\begin{aligned} Q^*(\tilde{y}, \tilde{u}; t) &= \sup_{u \in U} \left\{ \int_t^\infty g(s,t)\, ds \,\middle|\, y(t) = \tilde{y}, u(t) = \tilde{u} \right\} \\ &= \sup_{u \in U} \left\{ \int_t^{t+h} g(s,t)\, ds + e^{-\gamma h} \int_{t+h}^\infty g(s, t+h)\, ds \,\middle|\, y(t) = \tilde{y}, u(t) = \tilde{u} \right\}. \end{aligned}$$

Using the dynamic programming principle which states that

$$\sup_{u \in U} \left\{ \int_t^\infty g(s,t)\, ds \,\middle|\, y(t) = \tilde{y}, u(t) = \tilde{u} \right\} = \sup_{u \in U} \left\{ \int_{t+h}^\infty g(s,t)\, ds \,\middle|\, y(t+h) = \tilde{y}, u(t+h) = \tilde{u} \right\}.$$

we can then write

$$Q^*(\tilde{y}, \tilde{u}; t) = \sup_{u \in U} \left\{ \int_t^{t+h} g(s,t)\, ds + e^{-\gamma h} Q^*(y(t+h), u(t+h); t+h) \,\middle|\, y(t) = \tilde{y}, u(t) = \tilde{u} \right\}.$$

If we now subtract $Q^*(\tilde{y}, \tilde{u}; t)$ we get

$$\begin{aligned} 0 = \sup_{u \in U} \Bigg\{ & \int_t^{t+h} g(s,t)\, ds + \left(e^{-\gamma h} - 1\right) Q^*(y(t+h), u(t+h); t+h) \\ & + Q^*(y(t+h), u(t+h); t+h) - Q^*(\tilde{y}, \tilde{u}; t) \;\Bigg|\; \begin{matrix} y(t) = \tilde{y} \\ u(t) = \tilde{u} \end{matrix} \Bigg\}. \end{aligned}$$

After division by $h > 0$ follows

$$\begin{aligned} 0 = \sup_{u \in U} \Bigg\{ & \frac{1}{h} \int_t^{t+h} g(s,t)\, ds + \frac{e^{-\gamma h} - 1}{h} Q^*(y(t+h), u(t+h); t+h) \\ & + \frac{1}{h} \left(Q^*(y(t+h), u(t+h); t+h) - Q^*(\tilde{y}, \tilde{u}; t)\right) \;\Bigg|\; \begin{matrix} y(t) = \tilde{y} \\ u(t) = \tilde{u} \end{matrix} \Bigg\}. \end{aligned} \tag{6.6}$$

We now want to let $h \to 0$. Using the fundamental theorem of differentiation and integration we get

$$\lim_{h \to 0} \frac{1}{h} \int_t^{t+h} g(s,t)\, ds = g(t,t) = r(\tilde{y}, \tilde{u}), \tag{6.7}$$

we further have

$$\lim_{h \to 0} \frac{e^{-\gamma h} - 1}{h} Q^*(y(t+h), u(t+h); t+h) = \frac{d}{dt} e^{-\gamma t} \Big|_{t=0} Q^*(\tilde{y}, \tilde{u}; t) = -\gamma Q^*(\tilde{y}, \tilde{u}; t) \tag{6.8}$$

and using the chain rule we get

$$\begin{aligned} \frac{d}{dt} Q^*(y(t), u(t); t) \Big|_{y(t)=\tilde{y},\ u(t)=\tilde{u}} &= \lim_{h \to 0} \frac{1}{h} \left(Q^*(y(t+h), u(t+h); t+h) - Q(y(t), u(t); t)\right) \\ &= \nabla_{\tilde{y}} Q^*(\tilde{y}, \tilde{u}; t) \cdot f(\tilde{y}, \tilde{u}) + \nabla_{\tilde{u}} Q^*(\tilde{y}, \tilde{u}; t) \cdot \frac{d}{dt} u(t) \Big|_{u(t)=\tilde{u}}. \end{aligned} \tag{6.9}$$

Using (6.7), (6.8), (6.9) and $|u'|_2 \leq L$ in (6.6) gives us combined

$$0 = \gamma Q^*(\tilde{y},\tilde{u};t) - \sup_{b\in\mathbb{R}^m,|b|_2\leq L}\{\nabla_{\tilde{y}}Q^*(\tilde{y},\tilde{u};t)\cdot f(\tilde{y},\tilde{u}) + \nabla_{\tilde{u}}Q^*(\tilde{y},\tilde{u};t)\cdot b + r(\tilde{y},\tilde{u})\}. \tag{6.10}$$

Now we want to calculate the value of $\sup_{b\in\mathbb{R}^m,|b|\leq L} c^\top b$. Using the Cauchy-Schwarz inequality we get

$$\|c^\top b\| \leq |c|_2|b|_2 \leq L|c|_2.$$

But if we let $\hat{b} = L\frac{c}{|c|_2}$ and use the fact that $\hat{b}$ is linear dependent on c we get equalities in the Cauchy-Schwarz inequality which means

$$\|c^\top \hat{b}\| = |c|_2|\hat{b}|_2 = |c|_2\left|L\frac{c}{|c|_2}\right|_2 = L|c|_2.$$

Hence $\sup_{b\in\mathbb{R}^m,|b|_2\leq L} c^\top b = L\frac{c}{|c|_2}$. Using this we can conclude

$$\begin{aligned}
0 &= \gamma Q^*(\tilde{y},\tilde{u};t) - \sup_{b\in\mathbb{R}^m,|b|_2\leq L}\{\nabla_{\tilde{y}}Q^*(\tilde{y},\tilde{u};t)\cdot f(\tilde{y},\tilde{u}) + \nabla_{\tilde{u}}Q^*(\tilde{y},\tilde{u};t)\cdot b + r(\tilde{y},\tilde{u})\} \\
&= \gamma Q^*(\tilde{y},\tilde{u};t) - \nabla_{\tilde{y}}Q^*(\tilde{y},\tilde{u};t)\cdot f(\tilde{y},\tilde{u}) - \sup_{b\in\mathbb{R}^m,|b|_2\leq L}\{\nabla_{\tilde{u}}Q^*(\tilde{y},\tilde{u};t)\cdot b\} - r(\tilde{y},\tilde{u}) \\
&= \gamma Q^*(\tilde{y},\tilde{u};t) - \nabla_{\tilde{y}}Q^*(\tilde{y},\tilde{u};t)\cdot f(\tilde{y},\tilde{u}) - L|\nabla_{\tilde{u}}Q^*(\tilde{y},\tilde{u};t)| - r(\tilde{y},\tilde{u}).
\end{aligned}$$

■

The Q-function is not differentiable in general hence we need to consider weak solutions of the HJB equation. If we consider the unique viscosity solution of the HJB equation we can see that they satisfy the HJB equation and coincide with the optimal Q-function. A proof of this result can be found in ([16], Theorem 7). Since the HJB equation now has a solution we want to show how we can use the fact that the optimal control $\bar{u}\in U$ s.t. $u(t)=\tilde{u}$ satisfies

$$\begin{aligned}
&\nabla_y Q^*(\bar{y}(s),\bar{u}(s);s)\cdot f(\bar{y}(s),\bar{u}(s)) + \nabla_u Q^*(\bar{y}(s),\bar{u}(s);s)\cdot \bar{u}'(s) + r(\bar{y}(s),\bar{u}(s)) \\
=& \sup_{b\in\mathbb{R}^m,|b|_2\leq L}\{\nabla_y Q^*(\bar{y}(s),\bar{u}(s);s)\cdot f(\bar{y}(s),\bar{u}(s)) + \nabla_u Q^*(\bar{y},\bar{u}(s);s)\cdot b + r(\bar{y}(s),\bar{u}(s))\}.
\end{aligned} \tag{6.11}$$

For that we first need to introduce the necessary objects for the generalized derivative of the viscosity solution, the following definitions are taken out of ([9], Section 2) and [10] respectively

Definition 6.3. *Let $g: S\to\mathbb{R}$ where $S\subseteq\mathbb{R}^q$ is open. We define the set of super-differentials of g in a point $x\in S$ as*

$$D^+g(x) := \left\{p\in\mathbb{R}^q \;\middle|\; \limsup_{\hat{x}\to x}\frac{g(\hat{x})-g(x)-p\cdot(\hat{x}-x)}{|\hat{x}-x|_2}\leq 0\right\}.$$

and the set of sub-differentials of g in a point $x\in S$ as

$$D^-g(x) := \left\{p\in\mathbb{R}^q \;\middle|\; \liminf_{\hat{x}\to x}\frac{g(\hat{x})-g(x)-p\cdot(\hat{x}-x)}{|\hat{x}-x|_2}\geq 0\right\}.$$

We further write $D^\pm g(x) = D^+g(x)\cup D^-g(x)$.

Remark 6.4. The elements $p\in D^+$ or D^- of the super- and sup-differential sets can be viewed as hyperplanes of the form

$$\hat{x}\mapsto g(\hat{x}) - g(x) - p\cdot(\hat{x}-x)$$

which lie tangent above or below of $g(x)$ respectively

Definition 6.5. *Let g be a locally Lipschitz continuous function $g : S \subseteq \mathbb{R}^q \to \mathbb{R}$. We define the Clarke's generalized directional derivative of g in $x \in S$ along the direction $\tilde{x} \in \mathbb{R}^q$ as*

$$g^\circ(x, \hat{x}) := \limsup_{\hat{x} \to x, h \downarrow 0} \frac{g(\hat{x} + h\tilde{x}) - g(\hat{x})}{h}$$

With that is Clarke's generalized gradient of g in $x \in S$ defined as

$$\partial^\circ g(x) := \{\eta \in \mathbb{R}^q \mid \forall \tilde{x} \in \mathbb{R}^q : \langle \eta, \tilde{x} \rangle \leq g^\circ(x, \tilde{x})\}$$

Now that we are able to express the generalized derivative of the viscosity solution, in the case where Q^* is not continuously differentiable. We can formulate the equation which provides us with the optimal control.

Theorem 6.6. *Assume that f and r are uniformly bounded and Lipschitz continuous. Define the control trajectory $\bar{u}(s)$ for $s \geq t$ by*

$$\bar{u}'(s) = L\frac{p_2}{|p_2|_2}, \quad \bar{u}(t) = \tilde{u} \tag{6.12}$$

$\forall p = (p_1, p_2) \in D^{\pm} Q^*(\bar{y}(s), \bar{u}(s)) \subseteq \mathbb{R}^{n+m} \setminus \{v \in \mathbb{R}^{n+m} \mid \forall i \in \{n+1, \ldots, m+n\} : v_i = 0\}$ for a.e. $s \geq t$, where $\bar{y}' = f(\bar{y}, \bar{u})$, $\bar{y}(t) = \tilde{y}$ for $s \geq t$.

Assume that Q^* is locally Lipschitz in a neighbourhood of $(\bar{y}(s), \bar{u}(s))$ and that $D^+ Q^*(\bar{y}(s), \bar{u}(s)) = \partial^\circ Q^*(\bar{y}(s), \bar{u}(s))$ for a.e. $s \geq t$. Then follows that

$$\bar{u} \in \underset{u \in U}{arg\ max} \left\{ \int_t^\infty e^{-\gamma(s-t)} r(y(s), u(s))\, ds \;\middle|\; y(t) = \tilde{y}, u(t) = \tilde{u} \right\}.$$

If we further have

$$\tilde{u} \in \underset{\hat{u} \in \mathbb{R}^m}{arg\ max}\ Q^*(\tilde{y}, \hat{u}).$$

Then $\bar{u}$ is an optimal control, which satisfies

$$\bar{u} \in \underset{u \in U}{arg\ max} \left\{ \int_t^\infty e^{-\gamma(s-t)} r(y(s), u(s))\, ds \;\middle|\; y(t) = \tilde{y} \right\}.$$

Proof. For more details see ([16], Theorem 2) ■

Remark 6.7. If for $p = (p_1, p_2) \in D^{\pm} Q^*(\bar{y}(s), \bar{u}(s))$ with $p = 0 \in R^m$, we can choose the right side of (6.12) arbitrary.

Theorem 6.6 provides us with (6.12) that gives us the optimal control function $\bar{u}$. This is an immense advantage over the classical Q-Learning approach, since we do not have to solve a optimization problem every time we want to calculate a new control. Notice further that if Q^* is continuously differentiable in $(\tilde{y}, \tilde{u})$ then (6.12) reduces to

$$\bar{u}'(s) = L\frac{\nabla_u Q^*(\bar{y}(s), \bar{u}(s))}{|\nabla_u Q^*(\bar{y}(s), \bar{u}(s))|}, \quad \bar{u}(t) = \tilde{u} \tag{6.13}$$

for a.e. $s \geq t$. To see this we first need to characterize an element $p \in D^{\pm} g(x)$ for a function $g \in C(S)$, where $S \subseteq \mathbb{R}^q$ open.

Lemma 6.8. *Let $g \in C(S)$, where $S \subseteq \mathbb{R}^q$ is open. Then follows*

(i) $p \in D^+ g(x) \Leftrightarrow \exists h \in C^1(S)$ s.t. $\nabla h(x) = p$ and $g - h$ has a local maximum in x.

(ii) $p \in D^- g(x) \Leftrightarrow \exists h \in C^1(S)$ s.t. $\nabla h(x) = p$ and $g - h$ has a local minimum in x.

Proof. This proof is taken out of ([9], Lemma 2.1). (i): „⇒“ : Let $p \in D^+g(x)$. By definition of the super-differential of $g(x)$ follows that

$$\alpha := \limsup_{\hat{x}\to x} \frac{g(\hat{x}) - g(x) - p \cdot (\hat{x} - x)}{|\hat{x} - x|_2} \leq 0. \tag{6.14}$$

Using the definition of the limes superior there exists a $\varepsilon > 0$ and a $\delta > 0$ s.t. $\forall \hat{x} \in \mathbb{R}^q : |x - \hat{x}|_2 < \delta$ follows that

$$\frac{g(\hat{x}) - g(x) - p \cdot (\hat{x} - x)}{|\hat{x} - x|_2} < \alpha + \varepsilon. \tag{6.15}$$

We can formulate this more general, for that define $\varphi \in C([0, \infty))$ which is monotonically increasing and $\varphi(0) = 0$. We then can write (6.15) as

$$g(\hat{x}) - g(x) - p \cdot (\hat{x} - x) \leq \varphi(|\hat{x} - x|_2)|\hat{x} - x|_2 \tag{6.16}$$

for $|\hat{x} - x|_2 < \delta$. Now define the function $\phi(t) := \int_0^t \varphi(t)\, dt$. We have $\phi(0) = \int_0^0 \varphi(t)\, dt = 0$, $\phi'(0) = \varphi(0) = 0$ by the fundamental theorem of calculus. Using that φ is monotonically increasing we also have for all $s \geq 0$

$$s\varphi(s) \leq \int_s^{2s} \varphi(s)\, ds \leq \int_0^{2s} \varphi(s)\, ds = \phi(2s). \tag{6.17}$$

Define the function $h(y) := g(x) + p \cdot (y - x) + \varphi(2|y - x|_2)$. We see that $h \in C^1(S)$ since $g \in C(S)$ by assumption and $p \in D^+g(x)$. This gives us $h(x) = g(x)$ and $\nabla h(x) = p$. Further follows with (6.16) and (6.17) for all $\hat{x} \in \mathbb{R}^q$ s.t. $|\hat{x} - x|_2 < \delta$

$$g(\hat{x}) - h(\hat{x}) \leq \varphi(|\hat{x} - x|_2)|\hat{x} - \hat{x}| - \phi(2|\hat{x} - x|_2) \leq 0.$$

Hence attains the function $g - h$ a local maximum at x, since $g(x) - h(x) = 0$.
„⇐“ : Now let $h \in C^1(S)$ with $\nabla h(x) = p$. Let $g - h$ attain a local maximum in x or let $\delta > 0$ s.t. $\forall \hat{x} \in \mathbb{R}^q$ with $|\hat{x} - x|_2 < \delta$ holds that $m := (g - h)(x) \geq (g - h)(\hat{x})$. With that follows

$$\begin{aligned}
\limsup_{\hat{x}\to x} \frac{g(\hat{x}) - g(x) - p \cdot (\hat{x} - x)}{|\hat{x} - x|_2} &\leq \limsup_{\hat{x}\to x} \frac{h(\hat{x}) + m - h(x) - m - p \cdot (\hat{x} - x)}{|\hat{x} - x|_2} \\
&= \limsup_{\hat{x}\to x} \frac{h(\hat{x}) - h(x) - p \cdot (\hat{x} - x)}{|\hat{x} - x|_2} \\
&= 0
\end{aligned}$$

which means that $p \in D^+g(x)$.

(ii): The second statement follows similarly to (i).

■

Now we are able to show how the sub- and super-differential sets of a function coincide with the gradient of that function, if the function is differentiable.

Lemma 6.9. *Let $g \in C(S)$, where $S \subseteq \mathbb{R}^q$ is open. If g is differentiable in x we have that $D^+g(x) = D^-g(x) = \{\nabla g(x)\}$.*

Proof. This proof is taken out of ([9], Lemma 2.3 (ii)). Let g be differentiable in x. By the definition of the set $D^+g(x)$ we see immediately that $\nabla g(x) \in D^+g(x)$. Now let $p \in D^+g(x)$. Using the first statement in lemma 6.8 we get the existence of $h \in C^1(S)$ such that $\nabla h(x) = p$ and that $g - h$ obtains a local minimum in x. Using the first order optimality condition we have that

$$\nabla g(x) - \nabla h(x) = 0 \Leftrightarrow \nabla g(x) = \nabla h(x) = p.$$

Hence $D^+g(x) = \{\nabla g(x)\}$. Similarly follows $D^-g(x) = \{\nabla g(x)\}$.

■

Using 6.9 we can see that if Q^* is Lipschitz continuous and continuously differentiable in a neighbourhood of $(\bar{y}, \bar{u})$ then equation (6.12) is equivalent to (6.13).

6.2 Semi-discrete HJB Equation

The authors of the original paper [16] introduced a semi-discrete version of the HJB equation from the previous subsection, without discretizing the dynamical system. The following subsection is based on ([16], Subsection 3.1).

At first we need a semi-discrete representation of the optimal Q-function defined by (6.3). This is done in two steps. The first step is to define a semi-discrete version for a normal Q-function given in the inside of the supremum in (6.3).

Definition 6.10. *Let $h > 0$ and $g \in \mathcal{G} := \{(g_k)_{k\in\mathbb{N}} \,|\, g_k \in \mathbb{R}^m, |g_k|_2 \leq L\}$. Given $(\tilde{y}, \tilde{u}) \in \mathbb{R}^n \times \mathbb{R}^m$ we define*

$$Q^{h,g}(\tilde{y}, \tilde{u}) := h \sum_{k=0}^{\infty} r(y_k, u_k)(1-\gamma h)^k \tag{6.18}$$

where

- *$y_{k+1} = \Psi(y_k, u_k; h)$. The term $\Psi(y_k, u_k; h)$ is the solution of the dynamical system (6.1) at $t = h$ with $y(0) = y_k$ and $u(t) \equiv u_k$ for $t \in [0\ ,\ h)$.*
- *$u_{k+1} = u_k + hg_k$, which represents the Euler approximation for the defined ODE from which we can obtain a optimal control.*
- *The term $(1-\gamma h)$ is the first order Taylor expansion of $e^{-\gamma h}$ around 0 which means*

$$e^{-\gamma h} = e^{-\gamma 0} + \frac{d}{dh} e^{-\gamma h}\Big|_{h=0} h + \mathcal{O}(h^2) = (1-\gamma h) + \mathcal{O}(h^2).$$

- *$(y_0, u_0) = (\tilde{y}, \tilde{u})$.*

The second step is now to obtain a approximation for the optimal Q-function using $Q^{h,g}$. This approximation will then solve the following semi-discrete version of the HJB equation from the previous section.

Proposition 6.11. *Let $0 < \gamma h < 1$ and assume that f and r are uniformly bounded and Lipschitz continuous. Define the optimal semi-discrete Q-function as*

$$Q^{h,*}(\tilde{y}, \tilde{u}) = \sup_{g\in\mathcal{G}} Q^{h,g}(\tilde{y}, \tilde{u}). \tag{6.19}$$

Then $Q^{h,*}$ is the unique solution of the semi-discrete HJB equation

$$Q^{h,*}(\tilde{y}, \tilde{u}) = hr(\tilde{y}, \tilde{u}) + (1-\gamma h) \sup_{\tilde{g}\in\mathbb{R}^m : |\tilde{g}|_2 \leq L} Q^{h,*}(\Psi(\tilde{y}, \tilde{u}; h), \tilde{u} + h\tilde{g}) \tag{6.20}$$

for all $(\tilde{y}, \tilde{u}) \in \mathbb{R}^n \times \mathbb{R}^m$.

Proof. The proof is taken out of ([16], Proposition 3). Let $0 < \gamma h < 1$. Further let f and g be uniformly bounded and Lipschitz continuous. Using this we get that $Q^{h,g}$ is well defined and that the supremum $Q^{h,*}$ exists.

We first show that $Q^{h,*}$ is a solution of the semi-discrete HJB equation.
"$\leq$": Let $g \in \mathcal{G}$ and $(\tilde{y}, \tilde{u}) \in \mathbb{R}^n \times \mathbb{R}^m$, we then get using definition 6.10 that

$$Q^{h,g}(\tilde{y}, \tilde{u}) = h \sum_{k=0}^{\infty} r(y_k, u_k)(1-\gamma h)^k = hr(\tilde{y}, \tilde{u}) + (1-\gamma h) Q^{h,g^1}\left(\Psi(\tilde{y}, \tilde{u}; h), \tilde{u} + hg_0\right),$$

where $g^1 = \{g_1, g_2, \ldots\}$. Using this we get that

$$\begin{aligned} Q^{h,g}(\tilde{y}, \tilde{u}) &\overset{(6.19)}{\leq} hr(\tilde{y}, \tilde{u}) + (1-\gamma h) Q^{h,*}\left(\Psi(\tilde{y}, \tilde{u}; h), \tilde{u} + hg_0\right) \\ &\leq hr(\tilde{y}, \tilde{u}) + (1-\gamma h) \sup_{\tilde{g}\in\mathbb{R}^m : |\tilde{g}|_2 \leq L} Q^{h,*}\left(\Psi(\tilde{y}, \tilde{u}; h), \tilde{u} + h\tilde{g}\right). \end{aligned} \tag{6.21}$$

Taking the supremum on both sides of (6.21) with respect to $g \in \mathcal{G}$, then gives us

$$\sup_{g\in\mathcal{G}} Q^{h,g}(\tilde{y},\tilde{u}) = \sup_{g\in\mathcal{G}} \left\{ hr(\tilde{y},\tilde{u}) + (1-\gamma h) \sup_{\tilde{g}\in\mathbb{R}^m : |\tilde{g}|_2 \le L} Q^{h,*}\left(\Psi(\tilde{y},\tilde{u};h), \tilde{u}+h\tilde{g}\right) \right\}$$

$$\Leftrightarrow \quad Q^{h,*}(\tilde{y},\tilde{u}) = hr(\tilde{y},\tilde{u}) + (1-\gamma h) \sup_{\tilde{g}\in\mathbb{R}^m : |\tilde{g}|_2 \le L} Q^{h,*}\left(\Psi(\tilde{y},\tilde{u};h), \tilde{u}+h\tilde{g}\right)$$

since the left-hand side is independent of $g \in \mathcal{G}$.

"$\ge$": Let $\tilde{g} \in \mathbb{R}^m$, such that $|\tilde{g}|_2 \le L$ and $(\tilde{y},\tilde{u}) \in \mathbb{R}^n \times \mathbb{R}^m$. Further set $y' := \Psi(\tilde{y},\tilde{u};h)$ and $u' := \tilde{u} + h\tilde{g}$. Now choose $\varepsilon > 0$ arbitrary. By the definition of the supremum defining $Q^{h,*}$ we can then choose $e \in \mathcal{G}$ such that

$$Q^{h,*}(y',u') \le Q^{h,e}(y',u') + \varepsilon.$$

Using that and the sequence $\tilde{e} = \{\tilde{g}, e_1, e_2, \ldots\}$ we get

$$\begin{aligned} Q^{h,*}(\tilde{y},\tilde{u}) \ge Q^{h,\tilde{e}}(\tilde{y},\tilde{u}) &= hr(\tilde{y},\tilde{u}) + (1-\gamma h) Q^{h,e}(y',u') \\ &\ge hr(\tilde{y},\tilde{u}) + (1-\gamma h)\left(Q^{h,*}(y',u') - \varepsilon\right). \end{aligned} \tag{6.22}$$

Taking the supremum on both sides of (6.22) with respect to $\tilde{g} \in \mathbb{R}^m$ gives us

$$\sup_{\tilde{g}\in\mathbb{R}^m . |\tilde{g}|_2 \le L} Q^{h,*}(\tilde{y},\tilde{u}) \ge \sup_{\tilde{g}\in\mathbb{R}^m . |\tilde{g}|_2 \le L} \left\{ hr(\tilde{y},\tilde{u}) + (1-\gamma h)\left(Q^{h,*}(y',u') - \varepsilon\right)\right\} \quad \Leftrightarrow$$

$$Q^{h,*}(\tilde{y},\tilde{u}) \ge hr(\tilde{y},\tilde{u}) - (1-\gamma h)\varepsilon + (1-\gamma h) \sup_{\tilde{g}\in\mathbb{R}^m . |\tilde{g}|_2 \le L} Q^{h,*}(y',u').$$

Since ε was chosen arbitrary we get

$$Q^{h,*}(\tilde{y},\tilde{u}) \ge hr(\tilde{y},\tilde{u}) + (1-\gamma h) \sup_{\tilde{g}\in\mathbb{R}^m . |\tilde{g}|_2 \le L} Q^{h,*}(y',u').$$

This shows that $Q^{h,*}$ satisfies the semi-discrete HJB equation.

■

With the solution of the semi-discrete HJB equation we now want to show that it's solution $Q^{h,*}$ converges to the optimal Q-function Q^*, which solves the HJB equation $Q^{h,*}$ introduced in the previous section. Then we can define an iteration formula Q_k^h based on the semi-discrete HJB equation, which approximates $Q^{h,*}$ and lastly show that the iteration formula Q_k^h converges to the optimal Q-function Q^*.

Proposition 6.12. *Let $0 < \gamma h < 1$ and assume that f and r are uniformly bounded and Lipschitz continuous. Then we have*

$$\lim_{h\to 0} \sup_{\substack{(\tilde{y},\tilde{u})\in K: \\ K\Subset\mathbb{R}^n\times\mathbb{R}^m}} |Q^{h,*}(\tilde{y},\tilde{u}) - Q^*(\tilde{y},\tilde{u};0)| = 0,$$

where $K \Subset \mathbb{R}^n \times \mathbb{R}^m$ is a compact subset.

Proof. The proof is based on some characteristics of the viscosity solution Q^* of the HJB equation. For a sketch of the proof and further details we refer to ([16], Proposition 4).

■

6.3 Approximation of the semi-discrete HJB Equation

Now we can use the semi-discrete form of the HJB equation to find an iteration formula Q_k^h, which approximates the semi-discrete solution $Q^{h,*}$ (This subsection is based on ([16], Subsection 3.2)). For that we define

Definition 6.13. *For $0 < \gamma h < 1$, learning rates $(\alpha_k)_{k\in\mathbb{N}} \subset \mathbb{R}_{>0}$ and $(y_0, u_0) := (\tilde{y}, \tilde{u}) \in \mathbb{R}^n \times \mathbb{R}^m$ we define for $k \geq 1$ the iteration formula*

$$Q_{k+1}^h(y_k, u_k) := (1-\alpha_k)Q_k^h(y_k, u_k) + \alpha_k \left[hr_k + (1-\gamma h) \sup_{\tilde{g}\in\mathbb{R}^m : |\tilde{g}|_2 \leq L} Q_k^h(y_{k+1}, u_k + h\tilde{g}) \right] \quad (6.23)$$

where $y_{k+1} := \Psi(y_k, u_k; h)$ as before and $r_k := r(y_k, u_k)$.

Remark 6.14. The iteration formula (6.23) is very similar to the standard Q-Learning update formula since we can rewrite it to

$$Q_{k+1}^h(y_k, u_k) = Q_k^h(y_k, u_k) + \alpha_k \left[hr_k + (1-\gamma h) \sup_{\tilde{g}\in\mathbb{R}^m : |\tilde{g}|_2 \leq L} Q_k^h(y_{k+1}, u_k + h\tilde{g}) - Q_k^h(y_k, u_k) \right]$$

From that we can derive the temporal difference defined as

$$\mathfrak{D}_k^h(Q_k^h) := hr_k + (1-\gamma h) \sup_{\tilde{g}\in\mathbb{R}^m : |\tilde{g}|_2 \leq L} Q_k^h(y_{k+1}, u_k + h\tilde{g}) - Q_k^h(y_k, u_k)$$

Which can be seen as L^∞-mapping, if we have the usual assumptions on f and r. This also makes it naturally possible to be able to apply the Q-Learning techniques, if the convergence to the optimal Q-function Q^* can be shown.

We are now able to show that the sequence defined by Q_k^h will converge against the solution $Q^{h,*}$ of the semi-discrete HJB equation.

Theorem 6.15. *Let $0 < \gamma h < 1$, $0 \leq \alpha_k \leq 1$ and assume that f and r are uniformly bounded and Lipschitz continuous. If the sequence $(\alpha_k)_{k\in\mathbb{N}} \subset [0,1]$ of learning rates satisfies $\sum_{k=0}^{\infty} \alpha_k = \infty$, then*

$$\lim_{k\to\infty} \|Q_k^h - Q^{h,*}\|_{L^\infty} = 0.$$

Proof. The proof is analogous to the one in ([16], Theorem 5) but we want to go through it since it's an essential part for this Q-Learning method. We first want to show that the temporal difference $\mathfrak{D}^h$ is monotone decreasing and it describes a contraction mapping. For that let the assumptions of this theorem be given.

Under the given assumptions we can choose $Q_1, Q_2 \in L^\infty$ as two Q-functions where $Q_1(y,u) \leq Q_2(y,u)$ for all $(y,u) \in \mathbb{R}^n \times \mathbb{R}^m$. With that we obtain

$$\sup_{\tilde{g}\in\mathbb{R}^m : |\tilde{g}|_2 \leq L} Q_1(\Psi(y,u,h), u+\tilde{g}h) \leq \sup_{\tilde{g}\in\mathbb{R}^m : |\tilde{g}|_2 \leq L} Q_2(\Psi(y,u,h), u+\tilde{g}h).$$

Using this we get

$$\begin{aligned}\mathfrak{D}^h(Q_1) &= hr(y,u) + (1-\gamma h) \sup_{\tilde{g}\in\mathbb{R}^m : |\tilde{g}|_2 \leq L} Q_1(\Psi(y,u,h), u+\tilde{g}h) \\ &\leq hr(y,u) + (1-\gamma h) \sup_{\tilde{g}\in\mathbb{R}^m : |\tilde{g}|_2 \leq L} Q_2(\Psi(y,u,h), u+\tilde{g}h) = \mathfrak{D}^h(Q_2),\end{aligned}$$

which shows that $\mathfrak{D}^h$ is monotone decreasing. Now define for $Q \in L^\infty$, $(y,u) \in \mathbb{R}^n \times \mathbb{R}^m$ and $\tilde{g} \in \mathbb{R}^m$: $|\tilde{g}|_2 \leq L$

$$\mathcal{D}^h(Q) := hr(y,u) + (1-\gamma h)\, Q(\Psi(y,u,h), u+\tilde{g}h).$$

Let again be $Q_1, Q_2 \in L^\infty$ then follows with $\mathcal{D}^h$

$$\begin{aligned}\mathcal{D}^h(Q_1) - \mathcal{D}^h(Q_2) &= ((1-\gamma h)Q_1(\Psi(y,u,h), u+\tilde{g}h) - (1-\gamma h)Q_2(\Psi(y,u,h), u+\tilde{g}h)) \\ &= (1-\gamma h)\left(Q_1(\Psi(y,u,h), u+\tilde{g}h) - Q_2(\Psi(y,u,h), u+\tilde{g}h)\right) \\ &\leq (1-\gamma h)\left\|Q_1 - Q_2\right\|_\infty .\end{aligned}$$

Since by definition of $\mathcal{D}^h$ we have $\sup_{\tilde{g}\in\mathbb{R}^m : |\tilde{g}|_2 \leq L} \mathcal{D}^h(Q) = \mathfrak{D}^h(Q)$, it follows that

$$\mathfrak{D}^h(Q_1) - \mathfrak{D}^h(Q_2) \leq (1-\gamma h)\left\|Q_1 - Q_2\right\|_\infty .$$

If we analogously switch the roles of Q_1 and Q_2 we get

$$|\mathfrak{D}^h(Q_1) - \mathfrak{D}^h(Q_2)| \leq (1-\gamma h)\left\|Q_1 - Q_2\right\|_\infty$$

and hence

$$\|\mathfrak{D}^h(Q_1) - \mathfrak{D}^h(Q_2)\|_\infty \leq (1-\gamma h)\left\|Q_1 - Q_2\right\|_\infty ,$$

which shows that $\mathfrak{D}^h$ is a contraction mapping with contraction factor $1-\gamma h < 1$.

We can now use $\mathfrak{D}^h$ to write Q^h_{k+1} as

$$Q^h_{k+1} = (1-\alpha_k)Q^h_k + \alpha_k \mathfrak{D}^h_k(Q^h_k) - Q^{h,*}.$$

Since $Q^{h,*}$ solves the semi-discrete HJB equation (6.20) we have $\mathfrak{D}^h_k(Q^{h,*}) = Q^{h,*}$. This gives us

$$\begin{aligned}Q^h_{k+1} - Q^{h,*} &= (1-\alpha_k)Q^h_k + \alpha_k \mathfrak{D}^h_k(Q^h_k) - Q^{h,*} \\ &= (1-\alpha_k)Q^h_k + \alpha_k \mathfrak{D}^h_k(Q^h_k - Q^{h,*} + Q^{h,*}) - Q^{h,*} \\ &= (1-\alpha_k)Q^h_k - (1-\alpha_k)Q^{h,*} + \alpha_k \left(\mathfrak{D}^h_k(Q^h_k - Q^{h,*} + Q^{h,*}) - Q^{h,*}\right) \\ &= (1-\alpha_k)\left(Q^h_k - Q^{h,*}\right) + \alpha_k \left(\mathfrak{D}^h_k(Q^h_k - Q^{h,*} + Q^{h,*}) - \mathfrak{D}^h_k(Q^{h,*})\right)\end{aligned}$$

Or equivalently if we define the error terms $e^h_k := Q^h_k - Q^*$ and $e^h_{k+1} := Q^h_{k+1} - Q^*$ we have

$$e^h_{k+1} = (1-\alpha_k)e^h_k + \alpha_k \left(\mathfrak{D}^h_k(e^h_k + Q^{h,*}) - \mathfrak{D}^h_k(Q^{h,*})\right). \tag{6.24}$$

Since $\mathfrak{D}^h_k$ is a contraction mapping we can now show that the error e^h_{k+1} decrease in the following way. For all $k \in \mathbb{N}$ holds that

$$\|e^h_k\|_\infty \leq \left(\prod_{i=0}^{k-1}(1-\alpha_i\gamma h)\right)\|e^h_0\|_\infty, \tag{6.25}$$

which we will show by complete induction. For that let $k=1$, which gives us using (6.24)

$$\begin{aligned}\|e^h_1\|_\infty &= \|(1-\alpha_0)e^h_0 + \alpha_0\left(\mathfrak{D}^h_0(e^h_0 + Q^{h,*}) - \mathfrak{D}^h_0(Q^{h,*})\right)\|_\infty \\ &\leq (1-\alpha_0)\|e^h_0\|_\infty + \alpha_0\|\mathfrak{D}^h_0(e^h_0 + Q^{h,*}) - \mathfrak{D}^h_0(Q^{h,*})\|_\infty \\ &\leq (1-\alpha_0)\|e^h_0\|_\infty + \alpha_0(1-\gamma h)\|e^h_0\|_\infty \\ &= (1-\alpha_0\gamma h)\|e^h_0\|_\infty = \left(\prod_{i=0}^{0}(1-\alpha_i\gamma h)\right)\|e^h_0\|_\infty,\end{aligned}$$

where we used that $\mathfrak{D}^h_k$ is a contraction mapping in line three. This concludes the initial case. Now assume for our induction hypotheses **IH** that for all $k \in \mathbb{N}$ holds that

$$\|e^h_k\|_\infty \leq \left(\prod_{i=0}^{k-1}(1-\alpha_i\gamma h)\right)\|e^h_0\|_\infty.$$

Then follows for the induction step $k \to k+1$ that

$$\begin{aligned}
\|e_{k+1}^h\|_\infty &= \|(1-\alpha_k)e_k^h + \alpha_k\left(\mathfrak{D}_k^h(e_k^h + Q^{h,*}) - \mathfrak{D}_k^h(Q^{h,*})\right)\|_\infty \\
&\leq (1-\alpha_k)\|e_k^h\|_\infty + \alpha_k\|\mathfrak{D}_k^h(e_k^h + Q^{h,*}) - \mathfrak{D}_k^h(Q^{h,*})\|_\infty \\
&\leq (1-\alpha_k)\|e_k^h\|_\infty + \alpha_k(1-\gamma h)\|e_k^h\|_\infty \\
&= (1-\alpha_k\gamma h)\|e_0^h\|_\infty \\
&\overset{\text{IH}}{\leq} (1-\alpha_k\gamma h)\left(\prod_{i=0}^{k-1}(1-\alpha_i\gamma h)\right)\|e_0^h\|_\infty \\
&= \left(\prod_{i=0}^{k}(1-\alpha_i\gamma h)\right)\|e_0^h\|_\infty.
\end{aligned}$$

Hence follows (6.25) by the principle of complete induction. Using the shown inequality we are now able to proof the main statement of this theorem. Since we have for all $x \in \mathbb{R}$ that $1-x \leq \exp(-x)$ it follows for the error e_k^h using (6.25) that

$$\|e_k^h\|_\infty \leq \left(\prod_{i=0}^{k-1}(1-\alpha_i\gamma h)\right)\|e_0^h\|_\infty \leq \exp\left(-\gamma h\sum_{i=0}^{k-1}\alpha_i\right)\|e_0^h\|_\infty.$$

Since we assumed that for $(\alpha_k)_{k\in\mathbb{N}} \subset [0,1]$ holds that $\sum_{k=0}^\infty \alpha_k$ diverges to positive infinity we get

$$\lim_{k\to\infty}\|e_k^h\|_\infty = 0,$$

which is equivalent to the main statement of this theorem.

■

The above theorem now enables us to proof the main theorem of the Hamilton-Jacobi DQN method, which is that the iteration formula Q_k^h converges to the optimal Q-function Q^*.

Corrolary 6.16. *Let $0 < \gamma h < 1$ and assume that f and r are uniformly bounded and Lipschitz continuous. If the sequence $(\alpha_k)_{k\in\mathbb{N}} \subset [0,1]$ of learning rates satisfies $\sum_{k=0}^\infty \alpha_k = \infty$. Then holds that for all $h > 0$ such that $0 < \gamma h < 1$ exists a $k_h \in \mathbb{N}$ such that*

$$\lim_{h\to 0}\sup_{k\geq k_h}\sup_{\substack{(\tilde{y},\tilde{u})\in K:\\ K\in\mathbb{R}^n\times\mathbb{R}^m}}|Q_k^h(\tilde{y},\tilde{u}) - Q^*(\tilde{y},\tilde{u};0)| = 0$$

Proof. The proof is also analogous to the one in ([16], Corollary 1) but we also go through this one since it's the main statement of the Hamilton-Jacobi DQN method. Let the assumptions of this theorem be given.

Since $\sum_{k=0}^\infty \alpha_k = \infty$, there exits a $k_h \in \mathbb{N}$ such that

$$\sum_{k=0}^{k_h-1}\alpha_k > \frac{1}{h^2} \Leftrightarrow h\sum_{k=0}^{k_h-1}\alpha_k > \frac{1}{h},$$

which shows that $h\sum_{k=0}^{k_h-1}\alpha_k \to \infty$ if $h \to 0$. Since for $(\tilde{y},\tilde{u}) \in \mathbb{R}^n \times \mathbb{R}^m$ the proof of the last theorem gives us

$$\|e_k^h\|_\infty \leq \exp\left(-\gamma h\sum_{k=0}^{k-1}\alpha_k\right)\|e_0^h\|_\infty.$$

From this we get using our choice of k_h that

$$\sup_{k\geq k_h}\|e_k^h\|_\infty \leq \sup_{k\geq k_h}\left[\exp\left(-\gamma h\sum_{k=0}^{k-1}\alpha_k\right)\|e_0^h\|_\infty\right] \leq \exp\left(-\gamma h\sum_{k=0}^{k_h-1}\alpha_k\right)\|e_0^h\|_\infty \to 0 \qquad (6.26)$$

for $h \to 0$. Using (6.26) and proposition 6.12 then gives us

$$\begin{aligned}
&\sup_{k \geq k_h} \sup_{\substack{(\tilde{y},\tilde{u}) \in K: \\ K \in \mathbb{R}^n \times \mathbb{R}^m}} |Q_k^h(\tilde{y},\tilde{u}) - Q(\tilde{y},\tilde{u};0)| \\
&\leq \sup_{k \geq k_h} \sup_{\substack{(\tilde{y},\tilde{u}) \in K: \\ K \in \mathbb{R}^n \times \mathbb{R}^m}} \big(|Q_k^h(\tilde{y},\tilde{u}) - Q^{h,*}(\tilde{y},\tilde{u})| - |Q^{h,*}(\tilde{y},\tilde{u}) - Q^*(\tilde{y},\tilde{u};0)| \big) \\
&\leq \sup_{k \geq k_h} \sup_{\substack{(\tilde{y},\tilde{u}) \in K: \\ K \in \mathbb{R}^n \times \mathbb{R}^m}} |Q_k^h(\tilde{y},\tilde{u}) - Q^{h,*}(\tilde{y},\tilde{u})| - \sup_{k \geq k_h} \sup_{\substack{(\tilde{y},\tilde{u}) \in K: \\ K \in \mathbb{R}^n \times \mathbb{R}^m}} |Q^{h,*}(\tilde{y},\tilde{u}) - Q^*(\tilde{y},\tilde{u};0)| \\
&\leq \sup_{k \geq k_h} \|e_k^h\|_\infty - \sup_{k \geq k_h} \sup_{\substack{(\tilde{y},\tilde{u}) \in K: \\ K \in \mathbb{R}^n \times \mathbb{R}^m}} |Q^{h,*}(\tilde{y},\tilde{u}) - Q^*(\tilde{y},\tilde{u};0)| \to 0
\end{aligned}$$

for $h \to 0$, which proofs the statement.

■

6.4 Hamilton-Jacobi Deep Q-Learning Algorithm

In the algorithm which applies the iteration formula given in definition 6.13 we are now going to use the introduced concepts of Deep Q-Learning and Double Q-Learning. This means that we want to approximate the optimal Q-function Q^* by using a neural network approximation Q^θ, defined by the parameters $\theta \in \mathbb{R}^d$. Additionally to that we are going to use a separate target network Q^{θ^-}, where the parameters $\theta^- \in \mathbb{R}^d$ are updated according to the Double-Q Learning strategy.

At first we are going to initialize a replay buffer and fill it partially by gathering a fixed amount of control samples. The data is then sampled in a mini-batch fashion, where we take mini-batches from the replay buffer, which was build using the experience replay method. In the next step we are going to apply the main iteration formula, given by definition 6.13 on the target network Q^{θ^-}. This is where we will use the knowledge of theorem 6.6 to get a optimal control, with respect to the network Q^θ, which will simplify the calculation complexity of the update formula as well as apply the principle of Double Q-Learning.

After that we will apply the Adam optimizer to minimize the loss between the update on the target network Q^{θ^-} and the network Q^θ to ultimately update the parameters θ. With the parameters θ we are then going to update the parameters θ^-. The last step is to update the control using again theorem 6.6 and in addition to that also an exploration strategy, where we just add centralized normally distributed noise with fixed variance to the control iteration.

The whole procedure is given in the following algorithm. For further details see ([16], Algorithm 1).

Algorithm 10 Hamilton-Jacobi Deep-Double Q-Learning

Require: Replay set capacity $N \in \mathbb{N}$, Number of episodes $M \in \mathbb{N}$, Terminal time step $T \in \mathbb{N}$, $\alpha \ll 1$, $\sigma > 0$ and $t = 0$.

Initialize the replay set $\mathfrak{R}$ with capacity N
Initialize the neural network Q^θ with random weights.
Initialize the target neural network Q^{θ^-} with $\theta^- = \theta$.
for episode $= 1$ to M **do**
  Randomly sample initial state-control pair (y_0, u_0)
  for $t = 0$ to T **do**
    Determine incurred reward $r_t = r(y_t, u_t)$
    Apply control u_t on the state y_t to obtain y_{t+1}.
    Save experience sample $(y_t, u_t, r_t, y_{t+1}) \in \mathfrak{R}$.
    Sample randomized mini-batch from replay set $\mathfrak{R}$.
    Calculate the temporal difference target for the target network for all elements in the mini-batch by

$$y_i' = hr_i + (1 - \gamma h) Q^{\theta^-}(y_{i+1}, u_i') \tag{6.27}$$

    where $u_i' = u_i + hL \dfrac{\nabla_u Q^\theta(y_i, u_i)}{|\nabla_u Q^\theta(y_i, u_i)|}$.

    Apply Adam on the loss function $\sum_i (y_i' - Q^\theta(y_i, u_i))^2$ using the sampled mini-batch to update the parameter of the Q^θ network.
    Update the target network parameters $\theta^- = (1 - \alpha)\theta^- + \alpha\theta$
    Update control and apply exploration

$$u_{t+1} = u_t + hL \frac{\nabla_u Q^\theta(y_t, u_t)}{|\nabla_u Q^\theta(y_t, u_t)|} + \varepsilon$$

    where $\varepsilon \sim N(0, \sigma^2 I_m)$.
  end for
end for

A crucial step in this algorithm is the calculation of the temporal difference target. Notice that we didn't calculate the supremum as defined in definition 6.13, we instead use the estimation u_i'

in (6.27). The following shows that this is a valid strategy to choose an estimate for the temporal difference target.

Lemma 6.17. *Suppose Q^{θ^-} is twice continuously differentiable with bounded first and second derivative. If $\nabla_u Q^{\theta^-}(y_i, u_i) \neq 0$ on the sample (y_i, u_i, r_i, y_{i+1}), we have*

$$e^{i,h} := \left| \max_{\tilde{g} \in \mathbb{R}^m : |\tilde{g}|_2 \leq L} Q^{\theta^-}(y_{i+1}, u_i + h\tilde{g}) - Q^{\theta^-}(y_{i+1}, u_i + h\tilde{g}_i) \right| = \mathcal{O}(h^2) \tag{6.28}$$

with $\tilde{g}_i = L \dfrac{\nabla_u Q^{\theta^-}(y_i, u_i)}{|\nabla_u Q^{\theta^-}(y_i, u_i)|}$.

Proof. This proof follows the one in ([16], Proposition 6). Using the triangle inequality we can split (6.28) for every i into the following parts

$$\begin{aligned} e^{i,h} \leq & \left| \max_{\tilde{g} \in \mathbb{R}^m : |\tilde{g}|_2 \leq L} Q^{\theta^-}(y_{i+1}, u_i + h\tilde{g}) - Q^{\theta^-}\left(y_{i+1}, u_i + hL \frac{\nabla_u Q^{\theta^-}(y_{i+1}, u_i)}{|\nabla_u Q^{\theta^-}(y_{i+1}, u_i)|_2} \right) \right| \\ & + \left| Q^{\theta^-}\left(y_{i+1}, u_i + hL \frac{\nabla_u Q^{\theta^-}(y_{i+1}, u_i)}{|\nabla_u Q^{\theta^-}(y_{i+1}, u_i)|_2} \right) - Q^{\theta^-}(y_{i+1}, u_i + h\tilde{g}_i) \right| =: |e_1^{i,h}| + |e_2^{i,h}|. \end{aligned}$$

We are first going to look at the term $e_1^{i,h}$. For that let $u^* := u_i + h\tilde{g}^*$, where $\tilde{g}^* \in \mathbb{R}^m$: $|\tilde{g}^*|_2 \leq L$ maximizes $Q^{\theta^-}(y_{i+1}, u_i + h\tilde{g})$. We then can write

$$\max_{\tilde{g} \in \mathbb{R}^m : |\tilde{g}|_2 \leq L} Q^{\theta^-}(y_{i+1}, u_i + h\tilde{g}) = Q^{\theta^-}(y_{i+1}, u^*).$$

Using that we write the first order Taylor expansion of Q^{θ^-} around u_i as

$$Q^{\theta^-}(y_{i+1}, u^*) = Q^{\theta^-}(y_{i+1}, u_i) + \nabla_u Q^{\theta^-}(y_{i+1}, u_i) \cdot (u^* - u_j) + \mathcal{O}(h^2)$$

and in addition to that we also get the first order Taylor expansion around u_i of

$$\begin{aligned} Q^{\theta^-}\left(y_{i+1}, u_i + hL \frac{\nabla_u Q^{\theta^-}(y_{i+1}, u_i)}{|\nabla_u Q^{\theta^-}(y_{i+1}, u_i)|_2} \right) = & Q^{\theta^-}(y_{i+1}, u_i) + \mathcal{O}(h^2) \\ & + \nabla_u Q^{\theta^-}(y_{i+1}, u_i) \cdot \left(hL \frac{\nabla_u Q^{\theta^-}(y_{i+1}, u_i)}{|\nabla_u Q^{\theta^-}(y_{i+1}, u_i)|_2} \right) \\ = & Q^{\theta^-}(y_{i+1}, u_i) + hL |\nabla_u Q^{\theta^-}(y_{i+1}, u_i)|_2 + \mathcal{O}(h^2). \end{aligned}$$

Hence follows

$$\begin{aligned} e_1^{i,h} &= \max_{\tilde{g} \in \mathbb{R}^m : |\tilde{g}|_2 \leq L} Q^{\theta^-}(y_{i+1}, u_i + h\tilde{g}) - Q^{\theta^-}\left(y_{i+1}, u_i + hL \frac{\nabla_u Q^{\theta^-}(y_{i+1}, u_i)}{|\nabla_u Q^{\theta^-}(y_{i+1}, u_i)|_2} \right) \\ &= Q^{\theta^-}(y_{i+1}, u^*) - Q^{\theta^-}\left(y_{i+1}, u_i + hL \frac{\nabla_u Q^{\theta^-}(y_{i+1}, u_i)}{|\nabla_u Q^{\theta^-}(y_{i+1}, u_i)|_2} \right) \\ &= \nabla_u Q^{\theta^-}(y_{i+1}, u_i) \cdot (u^* - u_j) - hL |\nabla_u Q^{\theta^-}(y_{i+1}, u_i)|_2 + \mathcal{O}(h^2). \end{aligned}$$

Using the definition of u^* it follows that

$$|u^* - u_i|_2 = |u_i + h\tilde{g}^* - u_i|_2 = h|\tilde{g}^*|_2 \leq hL,$$

we therefore get

$$\begin{aligned} e_1^{i,h} &= \nabla_u Q^{\theta^-}(y_{i+1}, u_i) \cdot (u^* - u_j) - hL |\nabla_u Q^{\theta^-}(y_{i+1}, u_i)|_2 + \mathcal{O}(h^2) \\ &\leq |\nabla_u Q^{\theta^-}(y_{i+1}, u_i)|_2 \, |u^* - u_j|_2 - hL |\nabla_u Q^{\theta^-}(y_{i+1}, u_i)|_2 + \mathcal{O}(h^2) \\ &\leq hL |\nabla_u Q^{\theta^-}(y_{i+1}, u_i)|_2 - hL |\nabla_u Q^{\theta^-}(y_{i+1}, u_i)|_2 + \mathcal{O}(h^2) = \mathcal{O}(h^2) \end{aligned}$$

which also shows $|e_1^{i,h}| = \mathcal{O}(h^2)$, since the components of u^* are by definition less then the components of u_i.

For the second term $e_2^{i,h}$ follows

$$\begin{aligned}
|e_2^{i,h}| &= \left| Q^{\theta^-}\left(y_{i+1}, u_i + hL\frac{\nabla_u Q^{\theta^-}(y_{i+1},u_i)}{|\nabla_u Q^{\theta^-}(y_{i+1},u_i)|_2}\right) - Q^{\theta^-}(y_{i+1}, u_i + h\tilde{g}_i)\right| \\
&= \left| Q^{\theta^-}\left(y_{i+1}, u_i + hL\frac{\nabla_u Q^{\theta^-}(y_{i+1},u_i)}{|\nabla_u Q^{\theta^-}(y_{i+1},u_i)|_2}\right) - Q^{\theta^-}\left(y_{i+1}, u_i + hL\frac{\nabla_u Q^{\theta^-}(y_i,u_i)}{|\nabla_u Q^{\theta^-}(y_i,u_i)|_2}\right)\right| \\
&\leq hL\|\nabla_u Q^{\theta^-}\|_\infty \left| \frac{\nabla_u Q^{\theta^-}(y_{i+1},u_i)}{|\nabla_u Q^{\theta^-}(y_{i+1},u_i)|_2} - \frac{\nabla_u Q^{\theta^-}(y_i,u_i)}{|\nabla_u Q^{\theta^-}(y_i,u_i)|_2}\right|_2,
\end{aligned}$$

where we used that the existing first derivative of Q^{θ^-} is bounded. Since for any $v, w \in \mathbb{R}^m \setminus \{0\}$ follows that

$$\begin{aligned}
\left|\frac{v}{|v|_2} - \frac{w}{|w|_2}\right|_2 &= \left|\frac{v}{|v|_2} - \frac{w}{|v|_2} + \frac{w}{|v|_2} - \frac{w}{|w|_2}\right|_2 \leq \frac{|v-w|_2}{|v|_2} + |w|_2\left(\frac{1}{|v|_2} - \frac{1}{|w|_2}\right) \\
&= \frac{|v-w|_2}{|v|_2} + \frac{|w|_2 - |v|_2}{|v|_2} \leq \frac{2|v-w|_2}{|v|_2},
\end{aligned}$$

where we used the inverse triangle inequality in the last step. With that and $v = \nabla_u Q^{\theta^-}(y_{i+1}, u_i)$ as well as $w = \nabla_u Q^{\theta^-}(y_i, u_i)$ follows

$$|e_2^{i,h}| \leq hL\|\nabla_u Q^{\theta^-}\|_\infty \frac{2|\nabla_u Q^{\theta^-}(y_{i+1},u_i) - \nabla_u Q^{\theta^-}(y_i,u_i)|_2}{|\nabla_u Q^{\theta^-}(y_{i+1},u_i)|_2}.$$

Since we assumed that Q^{θ^-} is twice continuously differentiable and that also the second derivative is bounded we get

$$|\nabla_u Q^{\theta^-}(y_{i+1},u_i) - \nabla_u Q^{\theta^-}(y_i,u_i)|_2 \leq \|\nabla_{yu} Q^{\theta^-}\|_\infty |y_{i+1} - y_i|_2 = \mathcal{O}(h). \tag{6.29}$$

Using $\nabla_u Q^{\theta^-}(y_i, u_i) \neq 0$ on the sample (y_i, u_i, r_i, y_{i+1}) we get that $|\nabla_u Q^{\theta^-}(y_i,u_i)|_2 > C$ for some $C > 0$. This together with the assumption that $\nabla_u Q^{\theta^-}$ is continuous implies that we can choose $h > 0$ such that

$$|\nabla_u Q^{\theta^-}(y_{i+1},u_i)|_2 > \frac{C}{2} > 0. \tag{6.30}$$

Applying (6.29) and (6.30) on the estimate $|e_2^{i,h}|$ concludes

$$|e_2^{i,h}| \leq hL\|\nabla_u Q^{\theta^-}\|_\infty \frac{2|\nabla_u Q^{\theta^-}(y_{i+1},u_i) - \nabla_u Q^{\theta^-}(y_i,u_i)|_2}{|\nabla_u Q^{\theta^-}(y_{i+1},u_i)|_2} = \mathcal{O}(h^2)$$

which proofs $e^{i,h} = \mathcal{O}(h^2)$.

■

A little issue still remains. Since we want to use Double Q-Learning we have to use the Q^θ network for the control selection. This means we have to use

$$\tilde{g}_i = L\frac{\nabla_u Q^\theta(y_i,u_i)}{|\nabla_u Q^\theta(y_i,u_i)|_2}$$

If we do that we will loose the $\mathcal{O}(h^2)$ convergence order from the above lemma but not the convergence itself.

7 Numerical Results

7.1 Problem Definition for Linear Examples

We want to apply the HJDQN algorithm on infinite horizon optimal control problems, where we use an example of a purely linear and a non-linear parabolic PDE constraint. Other than in most optimal control problems acts the control function in our examples only on a sub domain determined by a set of actuators, given by indicator functions $\mathbb{1}_i = \mathbb{1}_t(x)$, which supports are part of an open subset of our domain. The example for the linear parabolic PDE constraint in this section is orientated on [4].

Our example problem is given by the infinite horizon optimal control problem

$$\begin{aligned} \inf_{u\in U} J(u) &:= \int_0^\infty e^{-\gamma t}\, r(y(t),u(t))\, dt \\ s.t. \quad y(0) &= y_0 \text{ and } y'(t) = f(y(t),u(t)) \text{ for } t \in \mathbb{R}_{>0}, \end{aligned} \tag{7.1}$$

where $0 < \gamma < 1$. The constraint is defined by the parabolic PDE

$$\begin{aligned} \partial_t y(t) - \nu\Delta y(t) + a(t)y(t) + \nabla\cdot(b(t)y(t)) &= \sum_{i=1}^m u_i(t)\mathbb{1}_i \quad \text{in } (0,\infty)\times\Omega \\ y = 0 \quad \text{on } (0,\infty)\times\partial\Omega \quad \text{and} \quad y(0) &= y_0 \quad \text{on } \Omega, \end{aligned} \tag{7.2}$$

where the control vector is given by $u(t) := [u_1(t,\dots,u_m(t))]^\top \in L^2((0,\infty);\mathbb{R}^m)$, $d \in \{1,2\}$, $\Omega \subset \mathbb{R}^d$ is a bounded domain with a smooth boundary $\partial\Omega$ and $\nu > 0$. Furthermore let $a(t) = a(t,x) \in L^\infty((0,\infty);L^r(\Omega,\mathbb{R}))$ with $r \geq d := \dim(\Omega)$ and $b(t) = b(t,x) \in L^\infty((0,\infty)\times\Omega;\mathbb{R}^d)$. Note that we will use only autonomous coefficient functions for the main examples of this work, nevertheless will we show the existence of the state y using non-autonomous coefficient functions. Using that the dynamical state system function is defined by

$$f(y(t),u(t)) := \sum_{i=1}^m u_i(t)\mathbb{1}_i + \nu\Delta y(t) - a(t)y(t) - \nabla\cdot(b(t)y(t)). \tag{7.3}$$

We further define the reward function

$$r(y(t),u(t)) = \frac{\alpha}{2}\|\nabla y(t)\|^2_{L^2(\Omega)} + \frac{\beta}{2}|u(t)|^2_{l^2(\mathbb{R}^m)} \tag{7.4}$$

where $\alpha,\beta \in \mathbb{R}_{>0}$. Note that the model is trained using the negative reward, since the HJDQN problem uses the supremum over the objective function J.

7.2 PDE Constraint

We first want to analyse the PDE constraint given by (7.2) for the existence of a weak state solution y given a control vector. Using that we then can apply a Galerkin method to discretize the associated weak formulation.

7.2.1 Existence of Weak State Solutions

To make sure that (7.2) is well posed let $H := L^2(\Omega,\mathbb{R})$, $V := H^1_0(\Omega,\mathbb{R})$ and $V' := H^{-1}(\Omega,\mathbb{R})$ which defines a Gelfand triple $V \hookrightarrow H \simeq H' \hookrightarrow V'$. The norm on V is given by $\|v\|_V := \sqrt{\langle\nabla v,\nabla v\rangle_{L^2}}$ for all $v \in V$. To show that (7.2) is well posed we want to show that the respective weak variational formulation of (7.2) with a more general right hand side is well posed. This means that instead of (7.2) we are going to look at

$$\begin{aligned} \partial_t y(t) - \nu\Delta y(t) + a(t)y(t) + \nabla\cdot(b(t)y(t)) &= f(t) \quad \text{in } (t_0,t_0+T)\times\Omega \\ y = 0 \quad &\text{on } (t_0,t_0+T)\times\partial\Omega \\ y(0) = y_0 \quad &\text{on } \Omega \end{aligned} \tag{7.5}$$

instead. The weak formulation of this equation is given by

Definition 7.1. *Let $(T, t_0, y_0, f) \in \mathbb{R}_+^2 \times H \times L^2(t_0, t_0+T; V)$. Then is $y \in W(t_0, t_0+T; V, V')$ the weak solution of (7.5) if for almost every $t \in (t_0, t_0+T)$ holds*

1. $\langle \partial_t y(t), \varphi \rangle_{V',V} + \nu \langle \nabla y(t), \nabla \varphi \rangle_H + \langle a(t)y(t), \varphi \rangle_{V',V} - \langle b(t)y(t), \nabla \varphi \rangle_H = \langle f(t), \varphi \rangle_{V',V}$, $\forall \varphi \in V$.
2. $y(t_0) = y_0$ *in* H.

To obtain this formulation let $t \in (t_0, t_0+T)$ and $\varphi \in V$. We then multiply the first equation in (7.5) with φ and integrate over the domain Ω to get

$$\int_\Omega \partial_t y(t)\varphi \, dx - \nu \int_\Omega \Delta y(t)\varphi \, dx + \int_\Omega a(t)y(t)\varphi \, dx + \int_\Omega \nabla \cdot (b(t)y(t))\varphi \, dx = \int_\Omega f(t)\varphi \, dx$$

$$\overset{\text{(I)}}{\Leftrightarrow} \quad \langle \partial_t y(t), \varphi \rangle_{V',V} - \nu \left(\int_{\partial\Omega} (y(t) \cdot \vec{n})\, \varphi \, dS - \int_\Omega \nabla y(t) \nabla \varphi \, dx \right) + \langle a(t)y(t), \varphi \rangle_{V',V}$$

$$+ \left(\int_{\partial\Omega} (b(t)y(t) \cdot \vec{n})\, \varphi \, dS - \int_\Omega b(t)y(t) \cdot \nabla \varphi \, dx \right) = \langle f(t), \varphi \rangle_{V',V}$$

$$\overset{\text{(II)}}{\Leftrightarrow} \quad \langle \partial_t y(t), \varphi \rangle_{V',V} + \nu \langle \nabla y(t), \nabla \varphi \rangle_H + \langle a(t)y(t), \varphi \rangle_{V',V} - \langle b(t)y(t), \nabla \varphi \rangle_H = \langle f(t), \varphi \rangle_{V',V},$$

where in (I) we used Green's identity and the product rule of divergence. In (II) we used the fact that $y = 0$ on $\partial\Omega$ to eliminate the integrals over $\partial\Omega$. Using Gronwall's lemma given by

Lemma 7.2. *Let $I := [t_0, t_1]$ be an interval, $y, \alpha \in C(I, \mathbb{R})$ and $\beta \in C(I, [0, \infty))$. Moreover, it holds that*

$$y(t) \le \alpha(t) + \int_{t_0}^{t} \beta(s)y(s) \, ds$$

for all $t \in I$. Then follows

$$y(t) \le \alpha(t) + \int_{t_0}^{t} \alpha(s)\beta(s) \exp\left(\int_s^t \beta(\tau) \, d\tau \right) ds$$

Proof. A proof can be found in ([11], Theorem 16.6). ■

we can now formulate and proof the well-posedness in the following way

Proposition 7.3. *For every $(T, t_0, y_0, f) \in \mathbb{R}_+^2 \times H \times L^2(t_0, t_0+T; V')$ exists a unique weak solution $y \in W(t_0, t_0+T; V, V')$ of (7.5) satisfying*

$$\|y\|^2_{C(t_0, t_0+T; H)} + \|y\|^2_{W(t_0, t_0+T; V; V')} \le c_1 \left(\|y_0\|_H^2 + \|f\|^2_{L^2(t_0, t_0+T; V')} \right) \tag{7.6}$$

with c_1 depending on (T, ν, a, b, Ω).

Proof. The proof of existence is given in ([35], Chapter 3). For the proof of inequality (7.6) we follow the proof of the analogous statement in ([4], Proposition 3.2). At first we show that for every $a \in L^r(\Omega)$ with $r \ge d$ and $\varphi, \psi \in V$, there exists a $c > 0$ only dependent on Ω such that

$$\langle a\varphi, \varphi \rangle_{V',V} \le c \|a\|_{L^r(\Omega)} \|\varphi\|_H \|\varphi\|_V \quad \text{for } d \ge 1, \tag{7.7}$$

$$\langle a\varphi, \psi \rangle_{V',V} \le c \|a\|_{L^r(\Omega)} \left(\|\varphi\|_H \|\varphi\|_V \|\psi\|_H \|\psi\|_V \right)^{\frac{1}{2}} \quad \text{for } d \in \{1, 2\}, \tag{7.8}$$

$$\langle a\varphi, \psi \rangle_{V',V} \le c \|a\|_{L^r(\Omega)} \|\varphi\|_H \|\psi\|_V \quad \text{for } d \ge 3. \tag{7.9}$$

$\underline{d=1}$: Using Agmon's inequality ([1], Lemma 13.2), we get that

$$\begin{aligned}\langle a\varphi,\psi\rangle_{V',V} &\le \|a\|_{L^r(\Omega)}\|\varphi\|_H\|\psi\|_H \le \gamma_1\|a\|_{L^r(\Omega)}\|\varphi\|_{L^\infty(\Omega)}\|\psi\|_{L^\infty(\Omega)}\\ &\le c\|a\|_{L^r(\Omega)}\left(\|\varphi\|_H\|\varphi\|_V\|\psi\|_H\|\psi\|_V\right)^{\frac{1}{2}},\end{aligned}$$

where we used the compatibility between $\langle\cdot,\cdot\rangle_{V',V}$ and $\langle\cdot,\cdot\rangle_H$ provided by the Gelfand triple in the first inequality, that $\Omega\subset\mathbb{R}^d$ is bounded in the second inequality ($\gamma_1>0$) and Agmons inequality in the last inequality. Hence does $c>0$ only depend on Ω. This gives us for $d=1$ the inequalities (7.7) and (7.8) for $\psi=\varphi$.

$\underline{d=2}$: By using that $V\hookrightarrow L^4(\Omega)$ and an interpolation inequality (from for e.g. [22]) we get that

$$\begin{aligned}\langle a\varphi,\psi\rangle_{V',V} &\le \|a\|_{L^r(\Omega)}\|\varphi\|_H\|\psi\|_H \le c_1\|a\|_{L^r(\Omega)}\|\varphi\|_V\|\psi\|_V\\ &\le c_2\|a\|_{L^r(\Omega)}\|\varphi\|_{L^4(\Omega)}\|\psi\|_{L^4(\Omega)} \le c\|a\|_{L^r(\Omega)}\left(\|\varphi\|_H\|\varphi\|_V\|\psi\|_H\|\psi\|_V\right)^{\frac{1}{2}}\end{aligned}$$

which gives us for $d=2$ the inequalities (7.7) and (7.8) for $\psi=\varphi$.

$\underline{d\ge 3}$: Using that $V\hookrightarrow L^{\frac{2d}{d-2}}(\Omega)$ and $r\ge d$ we get

$$\begin{aligned}\langle a\varphi,\psi\rangle_{V',V} &\le \|a\|_{L^d(\Omega)}\|\varphi\|_H\|\psi\|_H \le \|a\|_{L^d(\Omega)}\|\varphi\|_V\|\psi\|_V\\ &\le \|a\|_{L^d(\Omega)}\|\varphi\|_H\|\psi\|_{L^{\frac{2d}{d-2}}} \le c\|a\|_{L^r(\Omega)}\|\varphi\|_H\|\psi\|_V\end{aligned}$$

which gives us for $d\ge 2$ the inequalities (7.7) and (7.9).

Assuming that the solution y is regular enough, we multiply $y(t)$ with the first equation in (7.5) integrate over Ω and apply (7.7)-(7.9) which gives us for $t\in(t_0,t_0+T)$

$$\langle\partial_t y(t),y(t)\rangle_{V',V}+\nu\langle\nabla y(t),\nabla y(t)\rangle_H=\langle b(t)y(t),\nabla y(t)\rangle_H-\langle a(t)y(t),y(t)\rangle_{V',V}+\langle f(t),y(t)\rangle_{V',V}$$

$$\begin{aligned}\Leftrightarrow\qquad \frac{1}{2}\frac{d}{dt}\|y(t)\|_H^2+\nu\|y(t)\|_V^2 &= \langle b(t)y(t),\nabla y(t)\rangle_H-\langle a(t)y(t),y(t)\rangle_{V',V}+\langle f(t),y(t)\rangle_{V',V}\\ &\le |\langle b(t)y(t),\nabla y(t)\rangle_H-\langle a(t)y(t),y(t)\rangle_{V',V}+\langle f(t),y(t)\rangle_{V',V}|\\ &\le |\langle b(t)y(t),\nabla y(t)\rangle_H|+|\langle a(t)y(t),y(t)\rangle_{V',V}|+|\langle f(t),y(t)\rangle_{V',V}|\\ &\le c\mathcal{N}(a,b)\|y(t)\|_H\|y(t)\|_V\\ &\quad+\|f(t)\|_{V'}\|y(t)\|_V,\end{aligned}\tag{7.10}$$

where $\mathcal{N}(a,b):=\left(\|a\|_{L^\infty((0,\infty);L^r(\Omega))}+\|b\|_{L^\infty((0,\infty)\times\Omega)}\right)$. Integrating 7.10 over (t_0,t_0+T) and using Gronwall's lemma 7.2 and Young's inequality gives us

$$\|y\|^2_{L^\infty((t_0,t_0+T),H)}+\nu\|y\|^2_{L^2((t_0,t_0+T),V)}\le c_1'\left(\|y_0\|_H^2+\|f\|^2_{L^2((t_0,t_0+T),V')}\right),\tag{7.11}$$

where $c_1'>0$ depends on (Ω,ν,a,b,T). Using 7.5 and (7.7)-(7.9) we also get that

$$\|\partial_t y\|_{L^2((t_0,t_0+T),V')}\le c(\nu+\mathcal{N}(a,b))\|y\|_{L^2((t_0,t_0+T),V)}+\|f\|_{L^2((t_0,t_0+T),V')}\tag{7.12}$$

Finally follows 7.6 then from 7.11 and 7.12.

■

7.2.2 Galerkin-Method

In the one and two dimensional linear cases of our constraint we will be able to use the previously introduced riccati equation, to obtain the exact optimal control solution. But the step before that is to use the Galerkin method to discretize the constraint and also the objective function. For the Galerkin method we are first going to choose $n \in \mathbb{N}$ linear independent functions $\varphi_1, \dots, \varphi_n \in V$ which form a basis of the space V_h defined by

$$V_h := \operatorname{span}\{\varphi_1, \dots, \varphi_n\} \subset V$$

which uses the same topology as in V. We now want to find for the given right-hand side described by $\sum_{i=1}^m u_i(t)\mathbb{1}_i \in V'$ in (7.2) an approximation $y_h \in V_h$, which fulfills the weak form given by definition 7.1 in the following way

$$\langle \partial_t y_h(t), \varphi_h\rangle_{V',V} + \nu\langle \nabla y_h(t), \nabla\varphi_h\rangle_H + \langle a y_h(t), \varphi_h\rangle_{V',V} - \langle b y_h(t), \nabla\varphi_h\rangle_H = \langle u(t)\cdot\mathbb{1}, \varphi_h\rangle_{V',V} \tag{7.13}$$

for all $\varphi_h \in V_h$, where $\mathbb{1} = [\mathbb{1}_1, \dots, \mathbb{1}_m]^\top$. Here we write $y_h \in V_h$ as

$$y_h = \sum_{i=1}^n y_i\varphi_i$$

with coefficients $y_i \in \mathbb{R}$ for all $i \in \{1, \dots, n\}$. We now divide (7.13) into three different parts. For the first part which contains $\partial_t y(t)$ we get for the coefficients $y_i' \in \mathbb{R}$ the equations

$$\langle \partial_t y_h(t), \varphi_j\rangle_{V',V} = \sum_{i=1}^n \langle y_i'\varphi_i, \varphi_j\rangle_H = \sum_{i=1}^n \langle \varphi_i, \varphi_j\rangle_H y_i' = (My')_{\cdot,j} \tag{7.14}$$

for $j \in \{1, \dots, n\}$, where we defined the so called mass matrix $M := (m_{ij})_{1\le i,j\le n} \in \mathbb{R}^{n\times n}$ with $m_{ij} = \langle \varphi_i, \varphi_j\rangle_V$. The second part contains the remaining parts of the left-hand side of (7.13). For that we get

$$\nu\langle \nabla y_h(t), \nabla\varphi_j\rangle_H = \sum_{i=1}^n \langle y_i\nabla\varphi_i, \nabla\varphi_j\rangle_H = \sum_{i=1}^n \langle \nabla\varphi_i, \nabla\varphi_j\rangle_H y_i = (A_1 y)_{\cdot,j}, \tag{7.15}$$

$$\langle a y_h(t), \varphi_j\rangle_H = \sum_{i=1}^n \langle a y_i\varphi_i, \varphi_j\rangle_H = \sum_{i=1}^n \langle a\varphi_i, \varphi_j\rangle_H y_i = (A_2 y)_{\cdot,j} \text{ and} \tag{7.16}$$

$$\langle b y_h(t), \nabla\varphi_j\rangle_H = \sum_{i=1}^p \langle b y_i\varphi_i, \nabla\varphi_j\rangle_H = \sum_{i=1}^p \langle b\varphi_i, \nabla\varphi_j\rangle_H y_i = (A_3 y)_{\cdot,j} \tag{7.17}$$

for $j \in \{1, \dots, n\}$. Where we defined the matrices

$$A_1 := (a_{1ij})_{1\le i,j\le n} \in \mathbb{R}^{n\times n} : a_{1ij} = \langle \nabla\varphi_i, \nabla\varphi_j\rangle_H,$$

$$A_2 := (a_{2ij})_{1\le i,j\le n} \in \mathbb{R}^{n\times n} : a_{2ij} = \langle a\varphi_i, \varphi_j\rangle_H \text{ and}$$

$$A_3 := (a_{3ij})_{1\le i,j\le n} \in \mathbb{R}^{n\times n} : a_{3ij} = \langle b\varphi_i, \nabla\varphi_j\rangle_H.$$

We will combine the matrices A_1, A_2, A_3 into a single one $A := A_3 - A_2 - A_1$. What's remaining now is the right-hand side of (7.13). For this we want to approximate or represent the indicator functions in V_h which means we write

$$\mathbb{1}_i = \sum_{k=1}^n \mathbb{1}_i^k\varphi_k,$$

where the components $\mathbb{1}_i^k \in \mathbb{R}$ are known for all $k \in \{1, \dots, n\}$ and all $i \in \{1, \dots, m\}$.

With that then follows

$$\begin{aligned}\langle u(t)\cdot \mathbb{1}, \varphi_j\rangle_H &= \left\langle \sum_{i=1}^{m} u_i(t)\mathbb{1}_i, \varphi_j \right\rangle_H = \left\langle \sum_{i=1}^{m} u_i(t)\left(\sum_{k=1}^{n}\mathbb{1}_i^k\varphi_k\right), \varphi_j\right\rangle_H \\ &= \sum_{i=1}^{m}\left\langle u_i(t)\left(\sum_{k=1}^{n}\mathbb{1}_i^k\varphi_k\right), \varphi_j\right\rangle_H = \sum_{i=1}^{m}\sum_{k=1}^{n}\left\langle u_i(t)\mathbb{1}_i^k\varphi_k, \varphi_j\right\rangle_H \qquad (7.18)\\ &= \sum_{i=1}^{m}\left(\sum_{k=1}^{n}\langle\varphi_k,\varphi_j\rangle_H\,\mathbb{1}_i^k\right)u_i(t) =: \sum_{i=1}^{m}(MA_I)_{ij}u_i(t) =: (Bu(t))_{\cdot,j}\end{aligned}$$

for $j \in \{1,\dots,n\}$, where $A_I := (a_{Iij})_{1\le i,j\le n} \in \mathbb{R}^{n\times m} : a_{Iij} = \mathbb{1}_j^i$ and $B := MA_I$. This now enables us to write (7.13) using (7.14), (7.15), (7.16), (7.17) and (7.18) as

$$My' = Ay + Bu \qquad (7.19)$$

Example 7.4. Let $\Omega = [0,1]$ and choose a equidistant grid of points $\mathcal{T} := \{x_i := ih \,|\, i \in \{0,\dots,n\}\}$, where $h = \frac{1}{n}$. On that we define the linear finite element space P_1 given by the basis functions φ_i for the grid point x_i as

$$\varphi_i(x) := \begin{cases} \dfrac{x-x_{i-1}}{h} & ,x\in[x_{i-1},x_i] \\ \dfrac{x_{i+1}-x}{h} & ,x\in[x_i,x_{i+1}] \\ 0 & ,\text{otherwise}\end{cases}, \quad \nabla\varphi_i(x) := \begin{cases} \dfrac{1}{h} & ,x\in[x_{i-1},x_i] \\ -\dfrac{1}{h} & ,x\in[x_i,x_{i+1}] \\ 0 & ,\text{otherwise}\end{cases}.$$

Using that we can calculate the entries of the matrices M, A and B. For the mass matrix M we only have to calculate m_{ii} for $i \in \{1,\dots,n\}$

$$\begin{aligned} m_{ii} &= \int_0^1 \varphi_i(x)^2\,dx = \int_{x_{i-1}}^{x_i}\left(\frac{x-x_{i-1}}{h}\right)^2 dx + \int_{x_i}^{x_{i+1}}\left(\frac{x_{i+1}-x}{h}\right)^2 dx \\ &= \frac{h}{3}\left(\frac{x-x_{i-1}}{h}\right)^3 + C\Big|_{x_{i-1}}^{x_i} + \left[-\frac{h}{3}\left(\frac{x_{i+1}-x}{h}\right)^3 + C\Big|_{x_i}^{x_{i+1}}\right] \\ &= \frac{h}{3}\left(\frac{x_i-x_{i-1}}{h}\right)^3 + \frac{h}{3}\left(\frac{x_{i+1}-x_i}{h}\right)^3 = \frac{2h}{3}\end{aligned}$$

and $m_{i,i-1} = m_{i-1,i}$ for $i \in \{2,\dots,n\}$, where

$$\begin{aligned} m_{i,i-1} &= \int_0^1 \varphi_i(x)\varphi_{i-1}(x)\,dx = \int_{x_{i-1}}^{x_i}\left(\frac{x-x_{i-1}}{h}\right)\left(\frac{x_i-x}{h}\right)dx \\ &= \frac{1}{h^2}\int_{x_{i-1}}^{x_i} x_i x - x^2 - x_{i-1}x_i + x_{i-1}x\,dx = \frac{1}{h^2}\left[\frac{1}{2}x_i x^2 - \frac{1}{3}x^3 - x_{i-1}x_i x + \frac{1}{2}x_{i-1}x^2\right]\Big|_{x_{i-1}}^{x_i} \\ &= \frac{h}{6}.\end{aligned}$$

The remaining entries are zero since $supp(\varphi) = [x_{i-1}, x_{i+1}]$. Hence follows for $M \in \mathbb{R}^{n\times n}$

$$M = \begin{pmatrix} \frac{2h}{3} & \frac{h}{6} & 0 & \cdots & 0 \\ \frac{h}{6} & \frac{2h}{3} & \frac{h}{6} & 0 & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & 0 & \frac{h}{6} & \frac{2h}{3} & \frac{h}{6} \\ 0 & \cdots & 0 & \frac{h}{6} & \frac{2h}{3} \end{pmatrix}$$

For simplicity we are going to set the coefficient to be scalar values $a, b \in \mathbb{R}$. Hence is the matrix A_2 very similar to M. For A_1 follows for a_{1ii} for $i \in \{1, \dots, n\}$

$$a_{1ii} = \int_0^1 \nabla\varphi_i^2(x)\,dx = \int_{x_{i-1}}^{x_i} \left(\frac{1}{h}\right)^2 dx + \int_{x_i}^{x_{i+1}} \left(-\frac{1}{h}\right)^2 dx = \frac{1}{h^2}x + C\Big|_{x_{i-1}}^{x_i} + \frac{1}{h^2}x + C\Big|_{x_i}^{x_{i+1}} = \frac{2}{h}$$

and further for $a_{1i,i-1} = a_{1i-1,i}$ for $i \in \{2, \dots, n\}$ follows

$$a_{1i,i-1} = \int_0^1 \varphi_i(x)\varphi_{i-1}(x)\,dx = \int_{x_{i-1}}^{x_i} \frac{1}{h}\left(-\frac{1}{h}\right) dx = -\frac{1}{h^2}x + C\Big|_{x_{i-1}}^{x_i} = -\frac{1}{h}.$$

Hence we get the matrices $A_1 \in \mathbb{R}^{n\times n}$ and $A_2 \in \mathbb{R}^{n\times n}$

$$A_1 = \begin{pmatrix} \frac{2}{h} & -\frac{1}{h} & 0 & \cdots & 0 \\ -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} & 0 & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & 0 & -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} \\ 0 & \cdots & 0 & -\frac{1}{h} & \frac{2}{h} \end{pmatrix}, \quad A_2 = \begin{pmatrix} \frac{2h}{3}a & \frac{h}{6}a & 0 & \cdots & 0 \\ \frac{h}{6}a & \frac{2h}{3}a & \frac{h}{6}a & 0 & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & 0 & \frac{h}{6}a & \frac{2h}{3}a & \frac{h}{6}a \\ 0 & \cdots & 0 & \frac{h}{6}a & \frac{2h}{3}a \end{pmatrix}.$$

For the matrix A_3 follows for a_{3ii} for $i \in \{1, \dots, n\}$

$$a_{3ii} = \int_0^1 b\varphi_i(x)\nabla\varphi_i(x)\,dx = \int_{x_{i-1}}^{x_i} b\,\frac{x - x_{i-1}}{h}\,\frac{1}{h}\,dx + \int_{x_i}^{x_{i+1}} b\,\frac{x_{i+1} - x}{h}\,\left(-\frac{1}{h}\right) dx = 0$$

and further for $a_{3i,i-1}$ and $a_{3i-1,i}$ for $i \in \{2, \dots, n\}$ follows

$$a_{3i,i-1} = \int_0^1 b\varphi_i(x)\nabla\varphi_i(x)\,dx = \int_{x_{i-1}}^{x_i} b\,\frac{x - x_{i-1}}{h}\left(-\frac{1}{h}\right) dx = -\frac{b}{h^2}\left(\frac{1}{2}x^2 - x_{i-1}x + C\Big|_{x_{i-1}}^{x_i}\right) = -\frac{b}{2}$$

and similarly $a_{3i-1,i} = \dfrac{b}{2}$. We hence get the matrices $A_3 \in \mathbb{R}^{n\times n}$ and $A \in \mathbb{R}^{n\times n}$ defined by

$$A_3 = \begin{pmatrix} 0 & -\frac{b}{2} & 0 & \cdots & 0 \\ \frac{b}{2} & 0 & -\frac{b}{2} & 0 & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & 0 & \frac{b}{2} & 0 & -\frac{b}{2} \\ 0 & \cdots & 0 & \frac{b}{2} & 0 \end{pmatrix} \text{ and}$$

$$A = \begin{pmatrix} -\left(\frac{2h}{3}a + \frac{2}{h}\right) & -\left(\frac{b}{2} + \frac{h}{6}a - \frac{1}{h}\right) & 0 & \cdots & 0 \\ \frac{b}{2} - \frac{h}{6}a + \frac{1}{h} & -\left(\frac{2h}{3}a + \frac{2}{h}\right) & -\left(\frac{b}{2} + \frac{h}{6}a - \frac{1}{h}\right) & 0 & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & 0 & \frac{b}{2} - \frac{h}{6}a + \frac{1}{h} & -\left(\frac{2h}{3}a + \frac{2}{h}\right) & -\left(\frac{b}{2} + \frac{h}{6}a - \frac{1}{h}\right) \\ 0 & \cdots & 0 & -\left(\frac{2h}{3}a + \frac{2}{h}\right) & -\left(\frac{b}{2} + \frac{h}{6}a - \frac{1}{h}\right) \end{pmatrix}.$$

For the last matrix B we only have to compute the matrix A_I, which represents the indicator functions $\mathbb{1}_i$ for $i \in \{1, \dots, m\}$. We want to represent m indicator functions given by

$$\mathbb{1}_k = \mathbb{1}_{[c_k, d_k]}, \quad \forall k \in \{1, \dots, m\},$$

where $c_k, d_k \in [0,1]$ and $c_k < d_k$ for all $k \in \{1, \dots, m\}$ on the space P_1. This is done by interpolation using the basis functions φ_i. Here we are using a simple interpolation given by

$$\mathbb{1}_k = \sum_{i=1}^{n} \mathbb{1}_k(x_i)\varphi_i, \quad \forall k \in \{1, \dots, m\}.$$

This implies that any entry in the matrix A_I, is either 0 or 1 depending on which basis function is used to interpolate the given indicator function, at a grid point x_i. We can therefore write the matrix $A_I \in \mathbb{R}^{n \times m}$ as

$$A_I = \begin{pmatrix} \mathbb{1}_1(x_1) & \mathbb{1}_1(x_2) & \cdots & \mathbb{1}_1(x_{n-1}) & \mathbb{1}_1(x_n) \\ \mathbb{1}_2(x_1) & \mathbb{1}_2(x_2) & \cdots & \mathbb{1}_2(x_{n-1}) & \mathbb{1}_2(x_n) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbb{1}_m(x_1) & \mathbb{1}_m(x_2) & \cdots & \mathbb{1}_m(x_{n-1}) & \mathbb{1}_m(x_n) \end{pmatrix}^\top.$$

7.3 Riccati Solution of the Linear-Quadratic Regulator Problem

Since we are now given the existence of a state solution and a discretization of the PDE-constraint, we are now able to determine the solution of the following infinite horizon optimal control problem

$$\begin{aligned} &\inf_{u \in U} J(u) := \int_0^\infty e^{-\gamma t}\, r(y(t), u(t))\, dt \\ &s.t. \quad y(0) = y_0 \text{ and } y'(t) = f(y(t), u(t)) \text{ for } t \in \mathbb{R}_{>0}, \end{aligned} \tag{7.20}$$

where for the above defined matrices $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$ and $M \in \mathbb{R}^{n \times n}$ we have

$$\begin{aligned} My'(t) = f(y(t), u(t)) := Ay(t) + Bu(t) \quad &\text{for } t \in \mathbb{R}_{>0}, \\ y(0) = y_0 \in \mathbb{R}^n & \end{aligned} \tag{7.21}$$

and for $\alpha, \beta > 0$, $\gamma \in (0,1)$, $A_1 \in \mathbb{R}^{n \times n}$ and $I_m \in \mathbb{R}^{m \times m}$ we have the reward given by

$$r(y(t), u(t)) := \frac{\alpha}{2} y(t)^\top A_1 y(t) + \frac{\beta}{2} u(t)^\top I_m u(t). \tag{7.22}$$

Now we want to determine the associated Riccati equation similar as in section 3.3 and check under which condition there exists a optimal control solution. To get to the continuous algebraic Riccati equation we first make a substitution given by $\hat{y} = My$. In addition to that we need that the matrix M is invertible. To show that let $v \in \mathbb{R}^n \setminus \{0\}$, then follows

$$v^\top M v = \sum_{i=1}^{n} v_i \left(\sum_{j=1}^{n} \langle \varphi_j, \varphi_i \rangle v_j \right) = \sum_{i=1}^{n} \sum_{j=1}^{n} \langle \varphi_j v_j, \varphi_i v_i \rangle = \left\langle \sum_{j=1}^{n} \varphi_j v_j, \sum_{i=1}^{n} \varphi_i v_i \right\rangle = \left\| \sum_{j=1}^{n} \varphi_j v_j \right\|^2 > 0,$$

where the last inequality follows since the functions φ_i are linear independent for all $i \in \{1, \dots, n\}$. Hence is the matrix M positive definite which implies also regularity. Therefore we can also write $y = M^{-1}\hat{y}$. Using that we get for the constraint in (7.21) and the reward that

$$\begin{aligned} \hat{y}'(t) &= AM^{-1}\hat{y}(t) + Bu \text{ and} \\ r(\hat{y}(t), u(t)) &= \frac{\alpha}{2} \hat{y}(t)^\top M^{-\top} A_1 M^{-1} \hat{y}(t) + \frac{\beta}{2} u(t)^\top I_m u(t). \end{aligned}$$

If we now define $Q := \frac{\alpha}{2} M^{-\top} A_1 M^{-1}$, $R := \frac{\beta}{2} I_m$ and $\hat{A} = AM^{-1}$ as well as

$$\tilde{A} = e^{-\gamma h}\hat{A} - \frac{1 - e^{-\gamma h}}{2h} I_n \text{ and } \tilde{B} = e^{-\gamma h} B$$

we can substitute that into (3.43) which then gives us

$$\begin{aligned}
&\Leftrightarrow 0 = Q + (P\tilde{A} + \tilde{A}^\top P) - P\tilde{B}R^{-1}\tilde{B}^\top P \\
&\Leftrightarrow 0 = \frac{\alpha}{2} M^{-\top} A_1 M^{-1} - e^{-2\gamma h} PBR^{-1}B^\top P - \frac{1 - e^{-\gamma h}}{h} P + e^{-\gamma h}\left(PAM^{-1} + \left(AM^{-1}\right)^\top P\right).
\end{aligned}$$

If we furthermore multiplying by M from both sides, since M is by definition symmetric, we then get

$$\begin{aligned}
0 &= \frac{\alpha}{2} A_1 - e^{-2\gamma h} MPBR^{-1}B^\top PM - \frac{1 - e^{-\gamma h}}{h} MPM + e^{-\gamma h}\left(MPA + A^\top PM\right) \\
\Leftrightarrow 0 &= \frac{\alpha}{2} A_1 - e^{-2\gamma h} MPBR^{-1}B^\top PM \\
&\quad - \frac{1 - e^{-\gamma h}}{2h} MPM - \frac{1 - e^{-\gamma h}}{2h} MPM + e^{-\gamma h}\left(MPA + A^\top PM\right) \\
\Leftrightarrow 0 &= \frac{\alpha}{2} A_1 - e^{-2\gamma h} MPBR^{-1}B^\top PM \\
&\quad + MP\left(e^{-\gamma h} A - \frac{1 - e^{-\gamma h}}{2h} M\right) + \left(e^{-\gamma h} A - \frac{1 - e^{-\gamma h}}{2h} M\right)^\top PM.
\end{aligned}$$

If we now for simplicity reuse the variable names for $\tilde{A}$ and Q we get

$$Q := \frac{\alpha}{2} A_1 \text{ and } \tilde{A} := e^{-\gamma h} A - \frac{1 - e^{-\gamma h}}{2h} M$$

which implies the CARE in a more generalized form

$$0 = Q + (MP\tilde{A} + \tilde{A}^\top PM) - MP\tilde{B}R^{-1}\tilde{B}^\top PM. \tag{7.23}$$

To see under which conditions there are solutions to the generalized CARE we are using the following results which are taken from [19]. As before is the un-discounted cost

$$J(y, u) = \int_0^\infty r(y(t), u(t))\, dt, \tag{7.24}$$

with $r(y(t), u(t)) = \frac{1}{2} y(t)^\top Q y(t) + \frac{1}{2} u(t)^\top R u(t)$ used together with the dynamical system

$$\begin{aligned}
My' &= Ay(t) + Bu(t) \\
y(t_0) &= y_0 \in \mathbb{R}^n
\end{aligned} \tag{7.25}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $R \in \mathrm{Sym}_{++}(m)$, $Q \in \mathrm{Sym}_{+}(n)$ and $M \in \mathbb{R}^{n\times n}$ which in theory could be also singular but for our problem we need it to be regular. Using the following additional definitions we can formulate two result on the existence of solutions which use stronger and weaker assumptions on the dynamical system (7.25).

Definition 7.5. *The triple (M, A, B), where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ and $M \in \mathbb{R}^{n\times n}$ is called stabilizable if*

1. *$\forall s \in \mathbb{C}$: $\mathrm{rank}\,[\,sM - A \quad B] = n$ follows that $Re(s) \geq 0$ and*
2. *$\mathrm{rank}\,[\,M \quad B] = n$.*

Definition 7.6. *The pair (M, A), where $A \in \mathbb{R}^{n\times n}$ and $M \in \mathbb{R}^{n\times n}$ is called regular and impulse-free if*

1. *$det(Ms - A) \neq 0$ for all $s \in \mathbb{C}$ and*
2. *$rank\, M = n$.*

Furthermore is the associated generalized continuous algebraic Riccati equation defined as

$$0 = Q + MPA + A^\top PM - MPBR^{-1}B^\top PM. \tag{7.26}$$

With that we get the following result

Theorem 7.7. *Assume that the pair (M, A) is regular and impulse-free and let (E, A, B) be stabilizable. If there exists $C \in \mathbb{R}^{n\times n}$ such that $Q = C^\top C$ then exists a solution $P \in Sym_+(n)$ of (7.26).*

Proof. A proof can be found in ([19], Theorem 3.2). ■

We can get the uniqueness of a solution P if we strengthen the assumptions and use observability of the triple (M, A, C) where $C \in \mathbb{R}^{n\times n}$ such that $Q = C^\top C$.

Definition 7.8. *The triple (M, A, C) where $M \in \mathbb{R}^{n\times n}$, $A \in \mathbb{R}^{n\times n}$ and $C \in \mathbb{R}^{n\times n}$ $(Q = C^\top C)$ is observable if and only if*

1. *$rank \begin{pmatrix} sM - A \\ C \end{pmatrix} = n \quad \forall s \in \mathbb{C}$ and*
2. *$rank \begin{pmatrix} M \\ C \end{pmatrix} = n$.*

Theorem 7.9. *Assume that the pair (M, A) is regular and impulse-free. If there exists $C \in \mathbb{R}^{n\times n}$ such that $Q = C^\top C$ and the triple (M, A, B) is observable, then exists a unique solution $P \in Sym_+(n)$ of (7.26).*

Proof. A proof can be found in ([19], Theorem 3.3).

■

7.4 Implementation Details

7.4.1 Software Details

To implement the HJDQN algorithm we used the following software

- PyTorch 2.0 [27]: Used to implement the Deep-Double Q-Learning algorithm with the Adam Optimizer.
- Replay Buffer: Implementation is analogous to the baseline implementation provided by OpenAI. For more detail see [12].
- FEniCSx 0.7 [6]: Used to implement PDE solvers which are employed in model training and evaluation.
- Gymansium 0.29.0 [36]: Used to build the single agent reinforcement learning environment which wraps the PDE solver.
- MATLAB 2023a [15]: The *icare* solver is used to calculate the Riccati solution of our Linear-Quadratic Regulator problems.

For model training we used the computing resources provided by Google Colab, along with the KNIME Google Connectors extension (KNIME 5.2.0, [7]) to handle the files between local and cloud storage. The computing resources have the following specifications

- CPU: Used for one dimensional and non-linear example. Single-Core Hyper Threaded Intel Xeon CPU with 2.20 GHz and 12 GB of RAM.
- GPU: Used to train two dimensional example. Depending on the availability of the resource we used:
  1. NVIDIA A100 Tensor Core GPU with 40 GB RAM.
  2. NVIDIA Tesla V100 GPU with 16 GB RAM.

The complete code can be found in GitHub [14].

7.5 PDE Solver

To solve the PDE constraint for the state y given a control vector $[u_1, \dots, u_m]^\top \in \mathbb{R}^m$, we apply the Rothe method where we first carry out a semi discretization in time and then use our weak formulation to do the space discretization. For that we have the equidistant grid on $[0, T]$ defined by

$$0 = t_0 < t_1 < \cdots < t_p = T,$$

where $p \in \mathbb{N}$ is the number of grid points. We now want to represent the state y as time discrete function $y_k := (y_k^q)_{q=0}^p$, where $y_k^q \in V$ for all $q \in \{0, \dots, p\}$. With that we can obtain the following iteration formula which uses the implicit Euler method. The initial state is given by $y_k^0 = y_0$ and the iterates using (7.2) by

$$y_k^q - h\nu\Delta y_k^q + hay_k^q + h\nabla \cdot (by_k^q) = y_k^{q-1} + hf^q,$$

where $f^q := \sum_{i=1}^m u_i(t_q)\mathbb{1}_i$. The next step is then to use the finite space $V_h \subset V$ for the discretization in space. This means we get

$$\langle y_{kh}^q, \varphi_h\rangle_H + h\nu\langle\nabla y_{kh}^q, \nabla\varphi_h\rangle_H + h\langle a y_{kh}^q, \varphi_h\rangle_H - h\langle b\, y_{kh}^q, \nabla\varphi_h\rangle_H = \langle y_{kh}^{q-1} + hf^q, \varphi_h\rangle_H \tag{7.27}$$

for all $\varphi_h \in V_h$, where $y_{kh}^q \in V_h$ for all $q \in \{0, \dots, p\}$. Moreover for the initial state we get

$$\langle y_{kh}^0, \varphi_h\rangle_H = \langle y_0, \varphi_h\rangle_H \quad \forall\varphi_h \in V_h. \tag{7.28}$$

The dsicretization given by (7.27) and (7.28) can now be used to implement the PDE constraint in FEniCSx, where the emerging linear system then gets solved using a linear solver provided by FEniCSx.

7.6 Model Details

For the linear one dimensional example we have the following parameters

Domain Parameters	
Spatial Domain Ω	$[-1, 1]$
Terminal Time T	2
Time increment h	0.01
Grid size Δx	0.01

Table 1: Domain Parameters Linear One Dimensional Example

PDE Parameters	
Proportionality constant ν	0.01
Coefficient function $a(x)$	$\sin(x)$
Coefficient function $b(x)$	$-x$
Initial Condition y_0	$(x-1)(x+1)+5$
Acctuator domains	$\text{supp}\,(\mathbb{1}_1) = [-0.7, 0.4]$ $\text{supp}\,(\mathbb{1}_2) = [-0.2, 0.2]$ $\text{supp}\,(\mathbb{1}_3) = [0.4, 0.7]$

Table 2: PDE Parameters Linear One Dimensional Example

LQR Parameters	
α	1
β	1
Discount factor γ	0.99999

Table 3: LQR Parameters Linear One Dimensional Example

Hyper Parameters	
Lipschitz constant L	$L \in \{5, 10\}$
Learning rate η	$\eta \in \{10^{-2}, 10^{-4}\}$
Double Q-Learning constant τ	$\tau \in \{10^{-3}, 10^{-6}\}$
Noise variance σ	$\sigma \in \{0.1, 1\}$
Batch size	512
Replay Buffer size	10^6

Table 4: Hyper Parameters Linear One Dimensional Example

In [16], where the HJDQN-Algorithm was introduced, they used a very simple network structure which was given by the following configuration

Layer	Input Shape	Output Shape	Number of Parameters
Original Model	$([512, 201], [512, 3])$	$[512, 1]$	-
⊢ Linear Layer 1	$[512, 204]$	$[512, 256]$	52480
⊢ ReLU Layer 1	$[512, 256]$	$[512, 256]$	-
⊢ Linear Layer 2	$[512, 256]$	$[512, 256]$	65792
⊢ ReLU Layer 2	$[512, 256]$	$[512, 256]$	-
⊢ Linear Layer 3	$[512, 256]$	$[512, 1]$	257
Total Number of Parameters: 118,529			
Trainable Parameters: 118,529			

Table 5: Original Model Structure Of HJDQN-Algorithm

This network here fits the input structure of our LQR problem where we have 201 parameters for the state and 3 parameters for the control since we have three acctuators. The obvious problem here, which becomes even more apparent in the two dimensional example, is that the control parameters are highly under represented. The easiest solution is to just expand the control inputs into a vector which has the same input size as the state. This control parameter vector gets created by using the matrix A_I. Here we multiply each column with the corresponding control parameter and then sum up all columns into a single vector, which gives us the acctuator vector for the one dimensional case as

$$a_I = \begin{pmatrix} \sum_{i=1}^{m} u_i \mathbb{1}_i(x_0) \\ \sum_{i=1}^{m} u_i \mathbb{1}_i(x_1) \\ \vdots \\ \sum_{i=1}^{m} u_i \mathbb{1}_i(x_n) \end{pmatrix},$$

since the actuator domains are disjoint. The next adjustment is that instead of concatenating the state and control vectors into one input parameter vector, we use convolution to merge them together into one input parameter vector, which then has the size equal to the number of state parameters. We therefore get the following network architecture

Layer	Input Shape	Output Shape	Number of Parameters
One Dimensional Model	$([512, 201], [512, 201])$	$[512, 1]$	-
⊢ Convolution Sequence	$[512, 2, 201]$	$[512, 4, 50]$	-
⊢ 1D Convolutional Layer	$[512, 2, 201]$	$[512, 4, 100]$	28
⊢ 1D Batch Normalization Layer	$[512, 4, 100]$	$[512, 4, 100]$	8
⊢ ReLU Layer	$[512, 4, 100]$	$[512, 4, 100]$	-
⊢ 1D Maximum Pooling Layer	$[512, 4, 100]$	$[512, 4, 50]$	-
⊢ Linear Layer 1	$[512, 200]$	$[512, 128]$	25728
⊢ ReLU Layer 1	$[512, 128]$	$[512, 128]$	-
⊢ Linear Layer 2	$[512, 128]$	$[512, 64]$	8256
⊢ ReLU Layer 2	$[512, 64]$	$[512, 64]$	-
⊢ Linear Layer 3	$[512, 64]$	$[512, 1]$	65
Total Number of Parameters: 34,085			
Trainable Parameters: 34,085			

Table 6: Model Structure For Linear One Dimensional Example

The linear two dimensional example has the following parameters

Domain Parameters	
Spatial Domain Ω	$[0, 1]^2$
Terminal Time T	5
Time increment h	0.01
Grid size Δx	0.03125

Table 7: Domain Parameters Linear Two Dimensional Example

PDE Parameters	
Proportionality constant ν	0.01
Coefficient function $a(x)$	$\sin(x_1)$
Coefficient vector $b(x)$	$(-x_1, -x_2)^\top$
Initial Condition y_0	$3\sin(\pi x_1)\sin(\pi x_2)$
Acctuator domains	$\text{supp}(\mathbb{1}_1) = [0.1, 0.3] \times [0.1, 0.3]$, $\text{supp}(\mathbb{1}_2) = [0.4, 0.6] \times [0.1, 0.3]$ $\text{supp}(\mathbb{1}_3) = [0.7, 0.9] \times [0.1, 0.3]$, $\text{supp}(\mathbb{1}_4) = [0.1, 0.3] \times [0.4, 0.6]$ $\text{supp}(\mathbb{1}_5) = [0.4, 0.6] \times [0.4, 0.6]$, $\text{supp}(\mathbb{1}_6) = [0.7, 0.9] \times [0.4, 0.6]$ $\text{supp}(\mathbb{1}_7) = [0.1, 0.3] \times [0.7, 0.9]$, $\text{supp}(\mathbb{1}_8) = [0.4, 0.6] \times [0.7, 0.9]$ $\text{supp}(\mathbb{1}_9) = [0.7, 0.9] \times [0.7, 0.9]$

Table 8: PDE Parameters Linear Two Dimensional Example

The Hyperparameter selection as well as the LQR parameters are the same as in the linear one dimensional case. The network architecture is also similar to the linear one dimensional case, except for an additional convolutional sequence. The following network architecture was employed

Layer	Input Shape	Output Shape	Number of Parameters
Two Dimensional Model	$([512, 1089], [512, 1089])$	$[512, 1]$	-
⊢ Convolution Sequence 1	$[512, 2, 33, 33]$	$[512, 6, 14, 14]$	-
⊢ 2D Convolutional Layer	$[512, 2, 33, 33]$	$[512, 6, 28, 28]$	438
⊢ 2D Batch Normalization Layer	$[512, 6, 28, 28]$	$[512, 6, 28, 28]$	12
⊢ ReLU Layer	$[512, 6, 28, 28]$	$[512, 6, 28, 28]$	-
⊢ 2D Maximum Pooling Layer	$[512, 6, 28, 28]$	$[512, 6, 14, 14]$	-
⊢ Convolution Sequence 2	$[512, 6, 14, 14]$	$[512, 16, 5, 5]$	-
⊢ 2D Convolutional Layer	$[512, 6, 14, 14]$	$[512, 16, 10, 10]$	2416
⊢ 2D Batch Normalization Layer	$[512, 16, 10, 10]$	$[512, 16, 10, 10]$	32
⊢ ReLU Layer	$[512, 16, 10, 10]$	$[512, 16, 10, 10]$	-
⊢ 2D Maximum Pooling Layer	$[512, 16, 10, 10]$	$[512, 16, 5, 5]$	-
⊢ Linear Layer 1	$[512, 400]$	$[512, 120]$	48120
⊢ ReLU Layer 1	$[512, 120]$	$[512, 120]$	-
⊢ Linear Layer 2	$[512, 120]$	$[512, 84]$	10164
⊢ ReLU Layer 2	$[512, 84]$	$[512, 84]$	-
⊢ Linear Layer 3	$[512, 84]$	$[512, 1]$	85
Total Number of Parameters: 61,267 Trainable Parameters: 61,267			

Table 9: Model Structure For Linear Two Dimensional Example

7.7 Numerical Results Of Linear One Dimensional Example

Given the parameters from table 1, table 2 and table 3 we get the uncontrolled state solution y_h^u. Furthermore if we use the MATLAB *icare*-solver to solve the continuous-time infinite horizon LQR problem with the LQR parameters given in table 3, we also get the controlled state solution y_h^c.

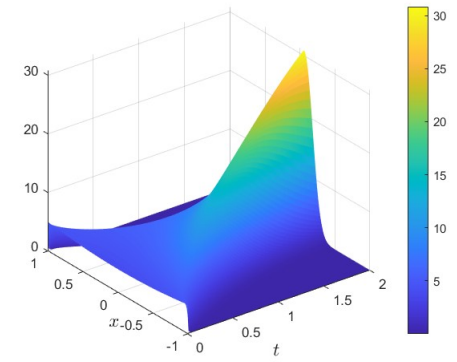


Figure 1: Uncontrolled State Solution y_h^u

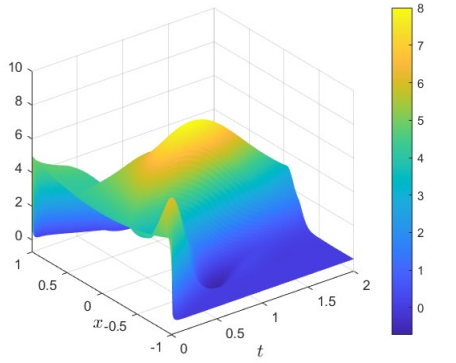


Figure 2: Controlled State Solution y_h^c

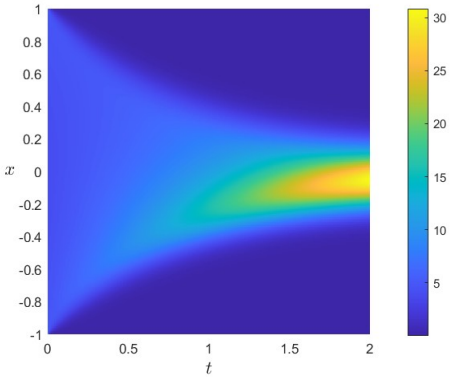


Figure 3: Over time evolution of y_h^u

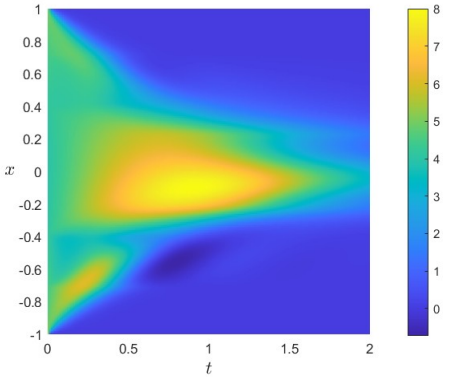


Figure 4: Over time evolution of y_h^c

To analyse the error between the controlled state solution y_h^c and the model solution y_M^c we use the relative L^2-error and the L^2-error between the controlled state solution and the model solution defined by

$$\varepsilon_{hM}^{rel} := \frac{\|y_h^c - y_M^c\|_{L^2((0,T),\mathbb{R}^n)}}{\|y_h^c\|_{L^2((0,T),\mathbb{R}^n)}} \quad \text{and} \quad \varepsilon_{hM} := \|y_h^c - y_M^c\|_{L^2((0,T),\mathbb{R}^n)}.$$

Furthermore define for fixed $t \in [0,T]$ the error terms

$$\varepsilon_{hM}^{rel}(t) := \frac{\|y_h^c(t,\cdot) - y_M^c(t,\cdot)\|_{L^2((\mathbb{R}^n)}}{\|y_h^c(t,\cdot)\|_{L^2(\mathbb{R}^n)}} \quad \text{and} \quad \varepsilon_{hM}(t) := \|y_h^c(t,\cdot) - y_M^c(t,\cdot)\|_{L^2(\mathbb{R}^n)}.$$

In addition to that we also analyse the evolution of the absolute value as well as the L^2-norm, for the control and state during the learning process. For that we use

$$\eta_y^M(t) := \|y_M^c(t,\cdot)\|_{L^2(\mathbb{R}^n)}, \quad \eta_y^h(t) := \|y_h^c(t,\cdot)\|_{L^2(\mathbb{R}^n)}, \quad \left(\eta_u^M(t)\right)_i := |u_i^M(t)|, \quad \left(\eta_u^h(t)\right)_i := |u_i^h(t)|,$$

$$\left(\eta_u^{hM,rel}(t)\right)_i := \ln\left(\frac{|u_i^h(t) - u_i^M(t)|}{|u_i^h(t)|}\right) \quad \text{and} \quad \left(\eta_u^{hM}(t)\right)_i := |u_i^h(t) - u_i^M(t)|$$

for $i \in \{1,\dots,m\}$ and fixed $t \in [0,T]$, where u_i^M and u_i^h are the control components obtained by the model and the Riccati solution respectively.

Furthermore gets the model evaluated on the incurred reward over the interval $t \in [0, T]$ which we will denote by $\mathcal{R}_M$ every 10 iterations. The same is done using the pre-calculated Riccati solution, where we analogously write $\mathcal{R}_h$. To compare both rewards we then use $\log \frac{\mathcal{R}_M}{\mathcal{R}_h}$. In addition to that we also analyse the collected rewards during the model training and denote that by $\mathcal{R}_T$.

The best solution in the given hyper parameter space was obtained with $L = 10$, $\tau = 10^{-6}$, $\eta = 10^{-2}$ and $\sigma = 1$. During the model training process the errors involving the state y_h^c and y_M^c evolve in the following way

$\|y_h^c\|_{L^2((0,2),\mathbb{R}^n)}$	$\|y_M^c\|_{L^2((0,2),\mathbb{R}^n)}$	ε_{hM}	ε_{hM}^{rel}	Iterations
82.90854	161.92710	86.59674	1.04449	0
82.90854	146.35724	66.44411	0.80141	2222
82.90854	143.70125	64.23419	0.77476	4444
82.90854	143.29805	63.32652	0.76381	6666
82.90854	152.90550	71.51150	0.86253	8888
82.90854	144.61106	64.22419	0.77464	11110
82.90854	137.57340	57.67118	0.69560	13332
82.90854	133.48537	54.04860	0.65191	15554
82.90854	134.03790	54.93583	0.66261	17776
82.90854	141.13848	61.25058	0.73877	19998

Table 10: Development of State Errors for the Linear One Dimensional Example

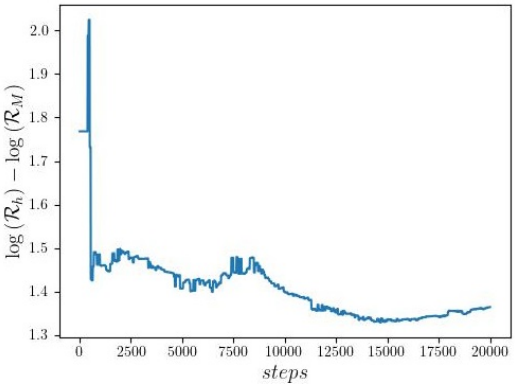


Figure 5: Evolution of $\log \frac{\mathcal{R}_M}{\mathcal{R}_h}$, 1d

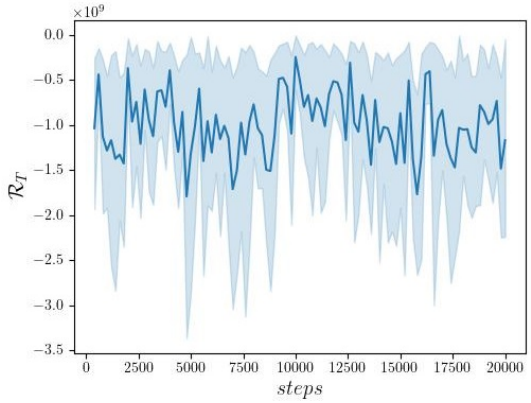


Figure 6: Episodic evolution of $\mathcal{R}_T$, 1d

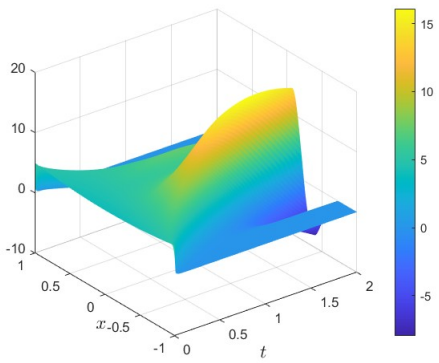


Figure 7: Controlled State Solution y_M^c

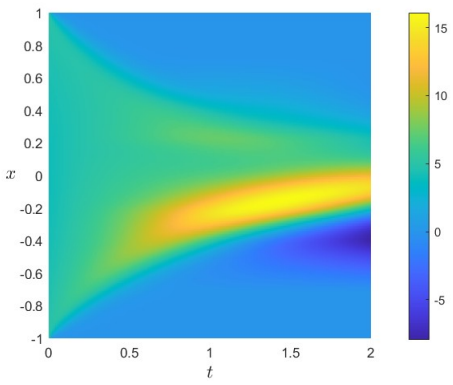


Figure 8: Over time evolution of y_M^c

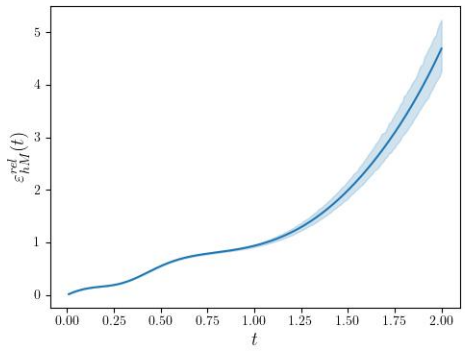


Figure 9: $\eta_{hM}^{rel}(t)$ during Training, 1d

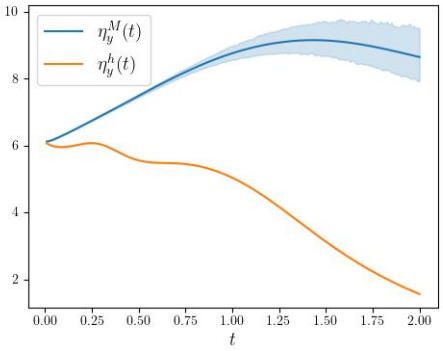


Figure 10: $\eta_y^M(t)$ and $\eta_y^h(t)$ during Training, 1d

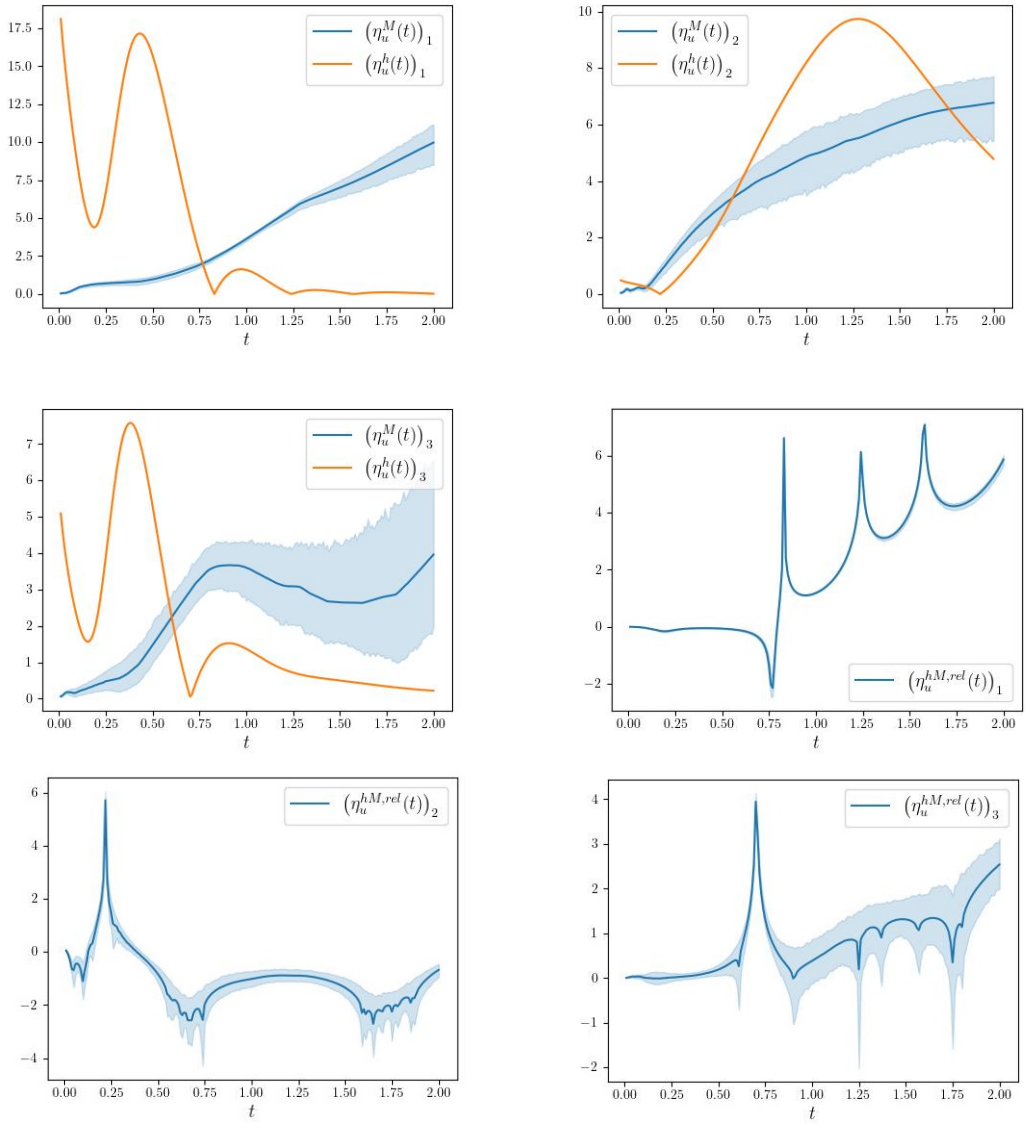


Figure 11: $\left(\eta_u^M(t)\right)_i$, $\left(\eta_u^h(t)\right)_i$ and $\left(\eta_u^{hm,rel}(t)\right)_i$ during Training for $i \in \{1,2,3\}$, 1d

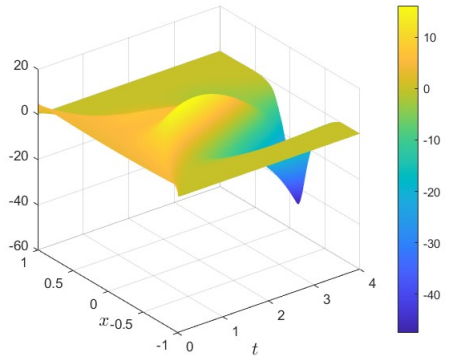


Figure 12: y_M^c, for $T = 4$

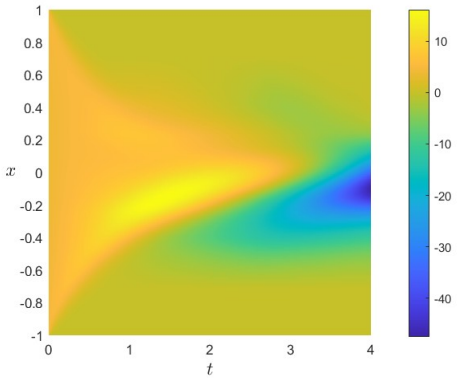


Figure 13: Over time evolution of y_M^c for $T = 4$

7.7.1 Conclusion

Although the results in table 10 speak for a decreasing relative error ε_{hm}^{rel} and the difference between $\mathcal{R}_M$ and $\mathcal{R}_h$ decreases according to figure 5, is the evolution of the control components $\left(\eta_u^M(t)\right)_i$ against $\left(\eta_u^h(t)\right)_i$ for $i \in \{1, 3\}$ not quite fitting. Despite the fact that the relative errors $\eta_u^{hM,rel}$ are quite low (except for the spikes which are caused when $|u_i^h(t)|$ is close to zero). One of the reasons for that is the obvious difference of the underlying problems for the HJDQN algorithm and the Riccati solution, since the HJDQN algorithm is restricted on choosing Lipschitz-continuous controls. This difference effectively implies the sub-optimality of the obtained model solution. Though we can see from figures 7 and 8 that the model can effectively act against the uncontrolled components of y_h^u. Nevertheless will it take a larger time interval for η_y^M to get to zero compared to η_y^h as we can see in figure 10. Unfortunately does this imply that we would need to train the model on a bigger time interval, which will consequently take more training time. This conclusion is also supported if we analyse how the model generalizes beyond the trained interval of $[0, 2]$. Here we can see from figure 12 and 13 that η_y^M diverges from 0 for $T = 4$.

We can also see from figure 5 that our choice of neural network architecture is adequate, because the range of $\log \mathcal{R}_h - \log \mathcal{R}_M$ is already relatively close to zero (meaning that the experiments conducted in [16] had similar performance). This can also suggest that in order to achieve better performance, we are possibly in need of a specific initial neural network configuration, to reduce the amount of exploration needed. This is also supported by figure 6 where the reward $\mathcal{R}_T$ does not really exceed a certain range. Instead of a initial guess for the neural network configuration, we could also try to achieve better results using much longer training times as suggested by the basic principles of Q-Learning, which say that convergence is achieved if all state and control value pairs are visited an infinite amount of times. This has the major issue that the used training time wouldn't be worth the effort for the given LQR problem. To counter an increase in training time, we could also choose a larger time increment, this would on the one hand give us more trainable episodes but on the other hand take away valuable resolution of the state solution on the time axis.

7.8 Numerical Results Of Linear Two Dimensional Example

Given the parameters from table 7 , table 8 and table 3 we get the controlled state y_h^c using the Riccati solution from the MATLAB *icare*-solver.

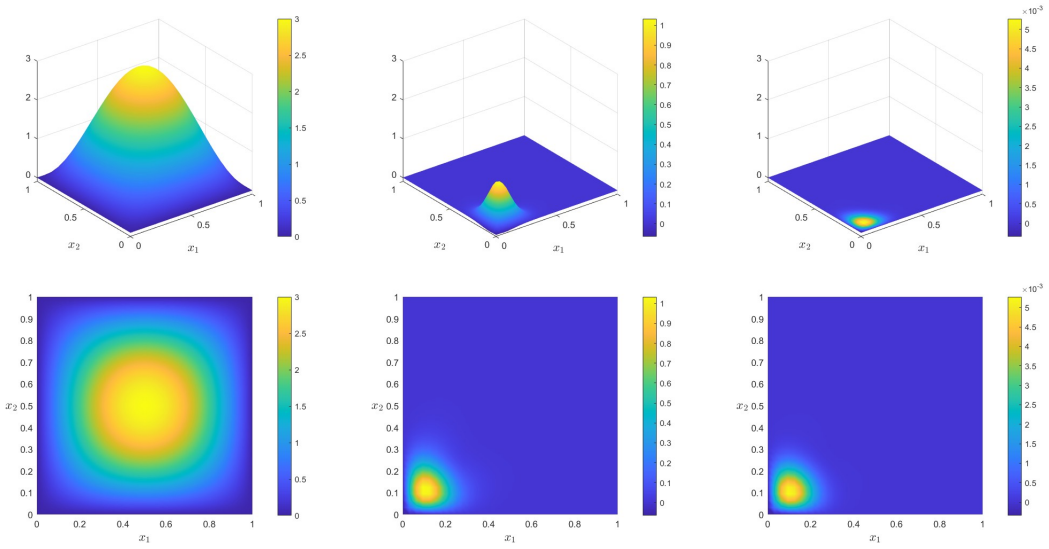


Figure 14: Controlled State Solution $y_h^c(t,\cdot)$ for $t \in \{0, 2.5, 5\}$ from left to right

The best solution in the given hyper parameter space was obtained with $L = 5$, $\tau = 10^{-3}$, $\eta = 10^{-2}$ and $\sigma = 0.1$. The model for the two dimensional example was evaluated every 50 iterations. During the model training process the errors involving the state y_h^c and y_M^c evolve in the following way

$\|y_h^c\|_{L^2((0,2),\mathbb{R}^n)}$	$\|y_M^c\|_{L^2((0,2),\mathbb{R}^n)}$	ε_{hM}	ε_{hM}^{rel}	Iterations
19.18655	24.60355	7.76647	0.40479	0
19.18655	33.16998	18.05524	0.94104	2222
19.18655	21.25686	9.34209	0.48691	4444
19.18655	20.64667	11.33457	0.59076	6666
19.18655	18.22709	4.55981	0.23766	8888
19.18655	16.41511	6.33835	0.33035	11110
19.18655	25.65483	7.82237	0.40770	13332
19.18655	15.77268	6.46531	0.33697	15554
19.18655	15.01458	6.68357	0.34835	17776
19.18655	18.48643	3.03485	0.15818	19998

Table 11: Development of State Errors for the Linear Two Dimensional Example

7.8.1 Conclusion

The two dimensional example was a better fit for the HJDQN algorithm, which is supported by figure 15, although the behaviour of $\log \mathcal{R}_M - \log \mathcal{R}_h$ diverged for a shorter period of the training process. This is a consequence of the small range of values in which the error $\log \mathcal{R}_M - \log \mathcal{R}_h$ lies, in combination with the relatively high variance of $\sigma = 0.1$ used for the exploration strategy. Similar to the one dimensional example, we can see from the incurred reward $\mathcal{R}_T$ in figure 16, that the neural network architecture we chose is quite fitting but we weren't able to improve the model more significantly. This tells us again that the training process could benefit from a better initial neural network configuration. Nonetheless is the relative error ε_{hM}^{rel} smaller than in the one dimensional example, which is backed up by the results in table 11. Noticeable is also that in comparison to the one dimensional example, we could obtain a better result using a smaller Lipschitz constant. The reason on why a smaller Lipschitz-constant works better in this case is

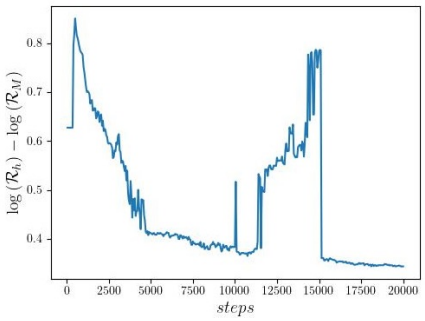


Figure 15: Evolution of $\log \frac{\mathcal{R}_M}{\mathcal{R}_h}$, 2d

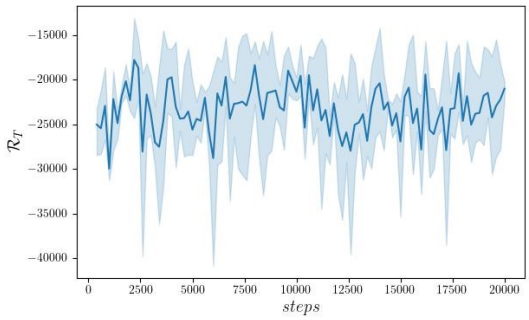


Figure 16: Episodic evolution of $\mathcal{R}_T$, 2d

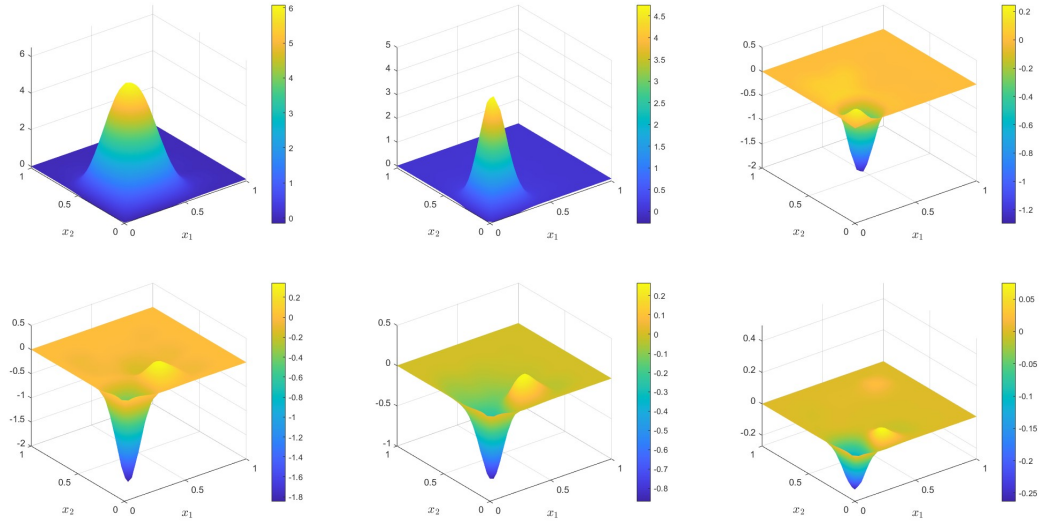


Figure 17: Controlled State Solution $y_M^c(t,\cdot)$ for $t \in \{0.6, 1.6, 2.4, 3.2, 4, 5\}$ from left to right

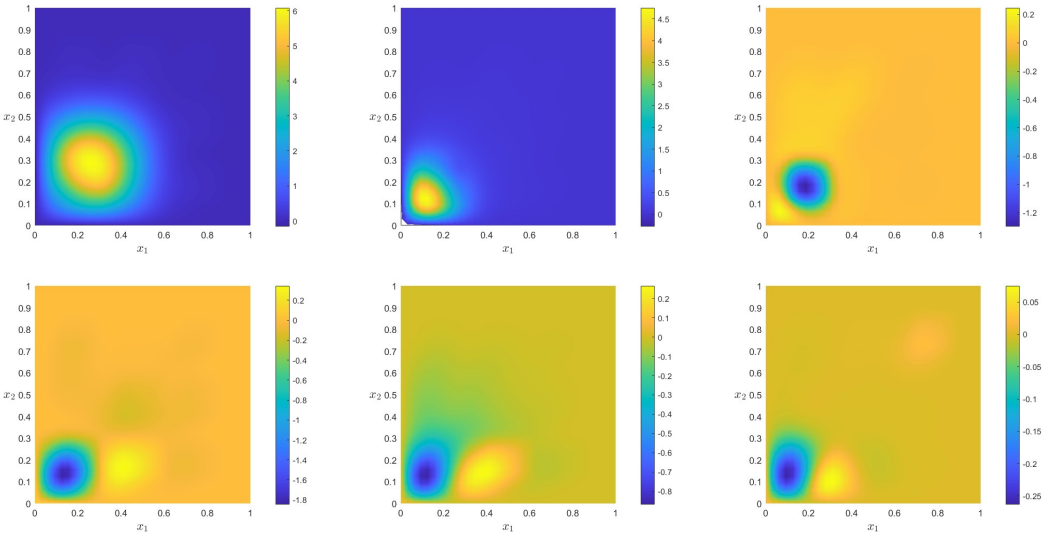


Figure 18: Contour plot of Controlled State Solution $y_M^c(t,\cdot)$ for $t \in \{0.6, 1.6, 2.4, 3.2, 4, 5\}$ from left to right

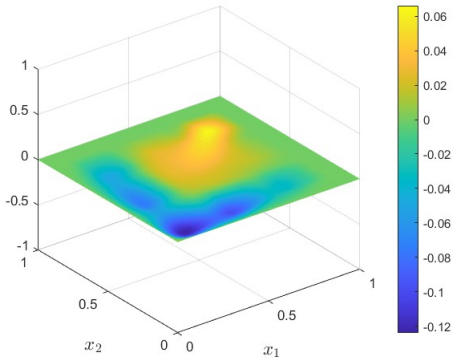


Figure 19: y_M^c, for $T = 6$

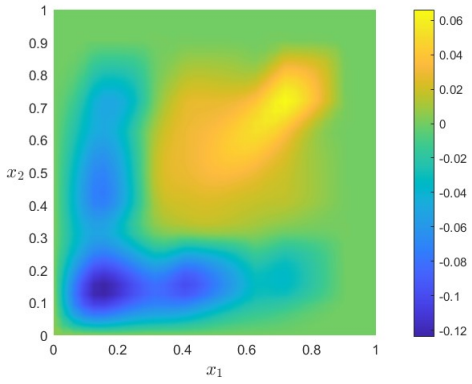


Figure 20: Over time evolution of y_M^c for $T = 6$

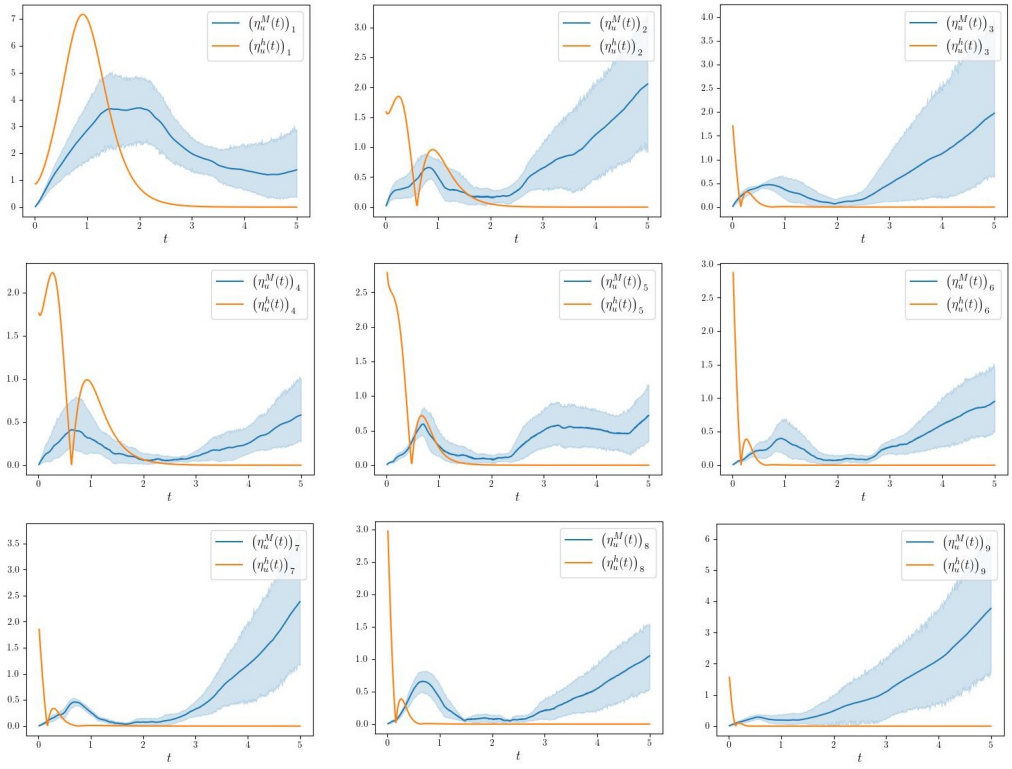


Figure 21: $\left(\eta_u^M(t)\right)_i$ and $\left(\eta_u^h(t)\right)_i$ during Training for $i \in \{1, \ldots, 9\}$, 2d

that we have less leeway to improve the relative error ε_{hM}^{rel}. One might think that this implies that we could achieve even better results, for the two dimensional example, with a Lipschitz constant smaller than $L = 5$. But this would reduce the influence of the second term on the right-hand side of

$$u_{t+1} = u_t + hL \frac{\nabla_u Q^\theta(y_t, u_t)}{|\nabla_u Q^\theta(y_t, u_t)|} + \varepsilon,$$

which can't be balanced out by a higher or lower variance for the exploration strategy. If we directly compare the evolution of $\left(\eta_u^M(t)\right)_i$ against $\left(\eta_u^h(t)\right)_i$ for $i \in \{1, \ldots, 9\}$, we can see that in the beginning of the time interval we have a better fit to the Riccati solution but it seem that this get way worse for later times. This again is due to the fact that the HJDQN algorithm solves a slightly different underlying problem, which is not an enormous problem since the norm $\eta_y^M(t)$ still decreases as we can see in figure 23. We furthermore observe in figures 17 and 18, that the model reduces the norm $\eta_y^M(t)$ similarly efficient as η_y^h (figure 14) for $0 < t < 2.5$. In spite of that does

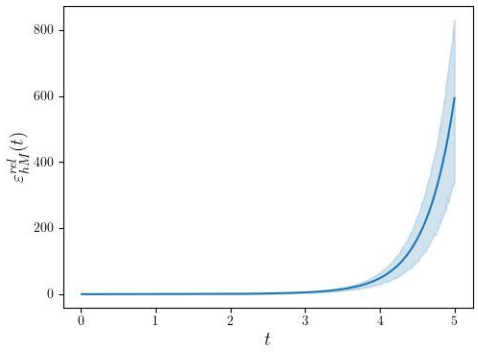


Figure 22: $\eta_{hM}^{rel}(t)$ during Training, 2d

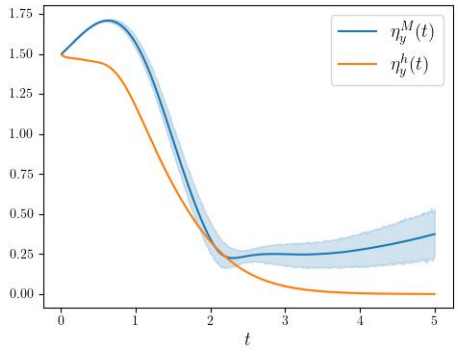


Figure 23: $\eta_y^M(t)$ and $\eta_y^h(t)$ during Training, 2d

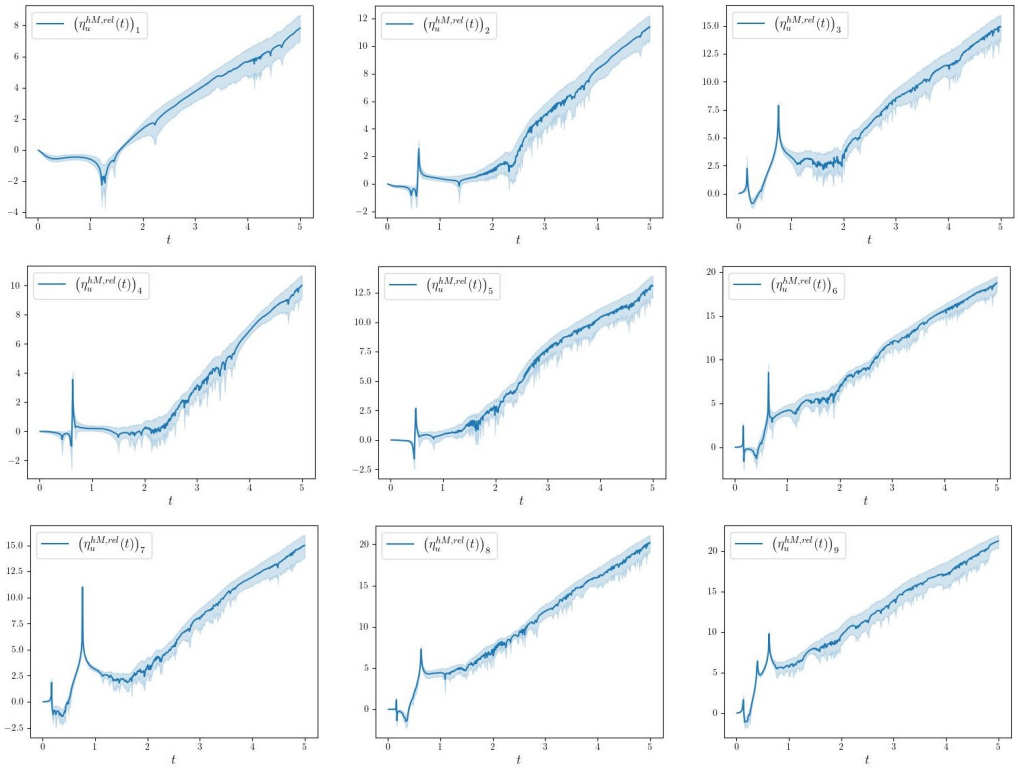


Figure 24: $\left(\eta_u^{hm,rel}(t)\right)_i$ during Training for $i \in \{1, \ldots, 9\}$, 2d

the model overshoot into the negative z-direction similar to the one dimensional case. However changes this behaviour for $t > 3.2$, where the model solution starts to tend towards zero, but it gets not quite there till the end of the training interval at $t = 5$.

To check if η_y^M goes further towards zero, we analyse how well the model generalizes for times outside of the training interval. We can see in figures 19 and 20 that the model can achieve a further improvement for $T = 6$ by reducing η_y^M even more.

7.9 Numerical Results Of Non-Linear Example

7.9.1 Problem Definition

We want to apply the HJDQN algorithm also on a optimal control problem, which has a non-linear parabolic PDE constraint. For that we use the following example taken from [5].

$$\min_{u\in L^2((0,T);\mathbb{R}^2)} \int_0^T \|y(t)\|^2_{L^2(0,1)} + \beta\|u(t)\|_2^2 \, dt \tag{7.29}$$

such that

$$\begin{aligned} \partial_t y - \nu\partial_x^2 y - y(1-y^2) &= \sum_{i=1}^{3} u_i(t)\mathbb{1}_i \quad \text{in } (0,T)\times\Omega \\ \partial_x y(t,-1) = \partial_x y(t,1) &= 0 \quad \text{in } (0,T) \\ y(0,x) &= y_0 \quad \text{in } \Omega \end{aligned} \tag{7.30}$$

where $T = 4$, $\Omega = (-1,1)$, $\nu = 0.1$ and $\beta = 0.01$. Further is $u(t) = [u_1(t), u_2(t), u_3(t)]^\top \in L^2((0,4),\mathbb{R}^3)$, $\text{supp}\,(\mathbb{1}_1) := (-0.7,-0.4)$, $\text{supp}\,(\mathbb{1}_2) := (-0.2,0.2)$, $\text{supp}\,(\mathbb{1}_3) := (0.4,0.7)$ and $y_0(x) = (x+1)(x-1)+5$. The constraint is solved for a given $u \in L^2((0,4),\mathbb{R}^3)$ using the *NonLinearProblem* function provided by FEniCSx. This function represents problems of the form

$$F(y,\varphi) = 0 \quad \forall\varphi \in V_h \tag{7.31}$$

for a finite dimensional space V_h. To get this form we can again use the Galerkin approach with a finite space $V_h \subset V := H^1(\Omega)$, $H = L^2(\Omega)$ and Galerkin triple $V \hookrightarrow H \simeq H' \hookrightarrow V'$ to write the weak formulation of (7.30) as

$$F(y,\varphi) := \langle \partial_t y, \varphi\rangle_H + \nu\langle\partial_x y, \partial_x\varphi\rangle_H - \langle y(1-y^2),\varphi\rangle_H - \left\langle \sum_{i=1}^{3} u_i(t)\mathbb{1}_i, \varphi \right\rangle_H . \tag{7.32}$$

Then is the Jacobian $\frac{dF}{du}$ calculated using the automatic differentiation algorithm using the Unified Form Language package which is part of FEniCSx. Now can the newton method be applied using (7.31) and (7.32) to solve for y. Further is the reward function for $V_h = \{\varphi_1,\dots,\varphi_n\}$, $y \in \mathbb{R}^n$ and $u \in \mathbb{R}^3$ defined by

$$r(y,u) = y^\top M y + \beta u^\top I_3 u \tag{7.33}$$

where $M := (m_{ij})_{1\le i,j\le n} \in \mathbb{R}^{n\times n} : m_{ij} = \langle\varphi_i,\varphi_j\rangle_H$. We again use (7.33) in the negative form to be able to apply the HJDQN algorithm. In the non-linear case we do not have a Riccati solution which means we can only analyse the time evolution of our model solution. The uncontrolled state y is given in figure 25 and 26.

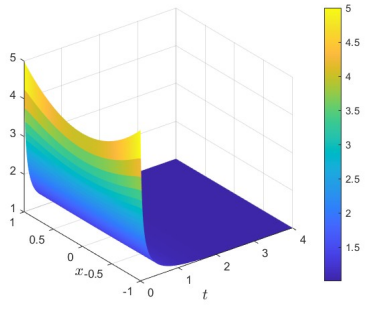


Figure 25: Uncontrolled State Solution y^u

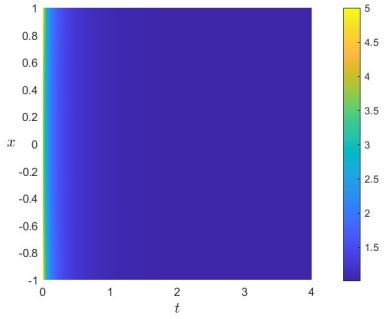


Figure 26: Over time evolution of y^u

The best solution in the given hyper parameter space was obtained with $L = 5$, $\tau = 10^{-3}$, $\eta = 10^{-2}$ and $\sigma = 0.1$. The model for the non-linear example was evaluated every 10 iterations. During the model training process the norm of the state evolved in the following way

$\|y_M^c\|_{L^2((0,4),\mathbb{R}^n)}$	Iterations
74.27539	0
64.02730	2222
63.88505	4444
63.58433	6666
63.37221	8888
18.95246	11110
22.27077	13332
63.49288	15554
21.27801	17776
62.96183	19998

Table 12: Development of State Errors for the Non-Linear Example

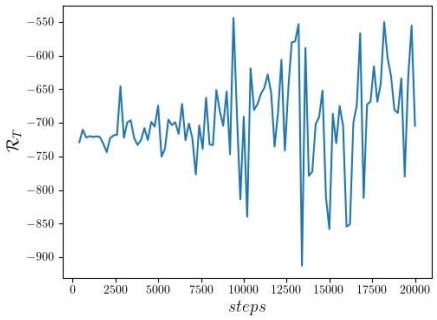


Figure 27: Evolution of $\mathcal{R}_T$ for best y_M^c.

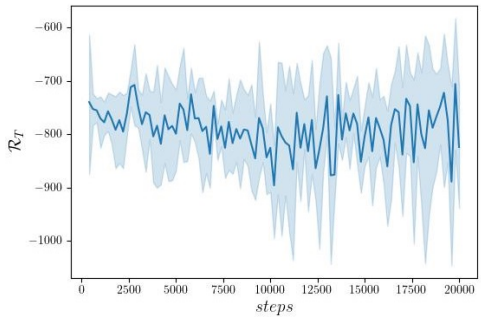


Figure 28: Episodic evolution of $\mathcal{R}_T$ for best y_M^c.

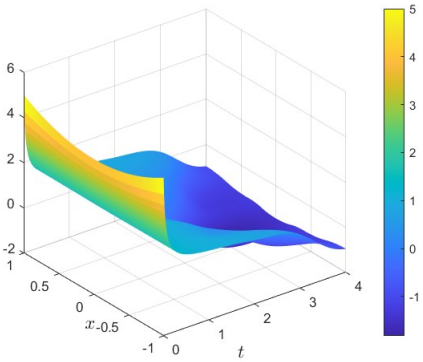


Figure 29: Controlled State Solution of best y_M^c

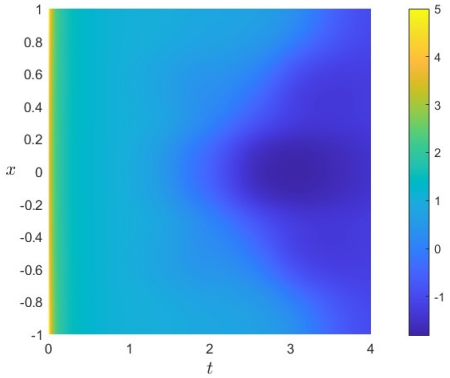


Figure 30: Over time evolution of best y_M^c

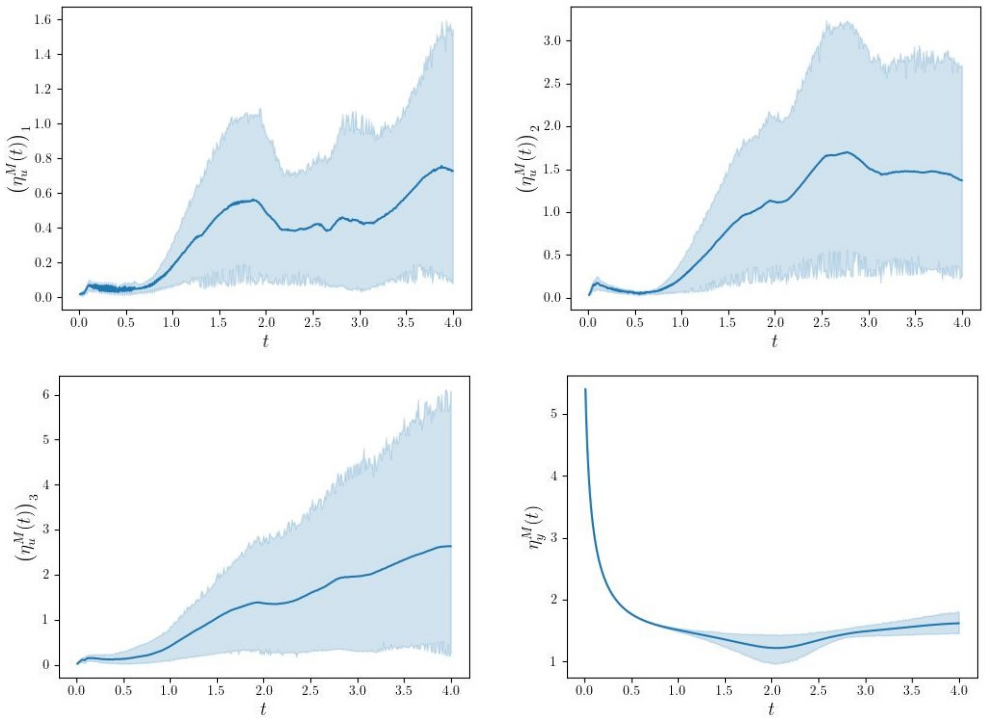


Figure 31: $\left(\eta_u^M(t)\right)_i$ and $\eta_y^M(t)$ during Training for $i \in \{1, 2, 3\}$

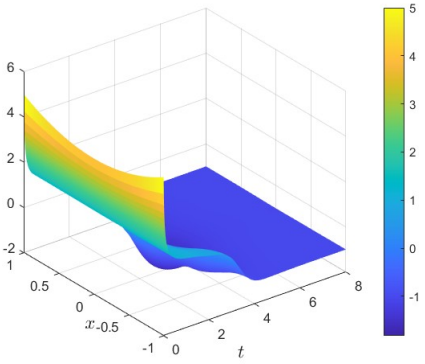


Figure 32: y_M^c for $T = 8$.

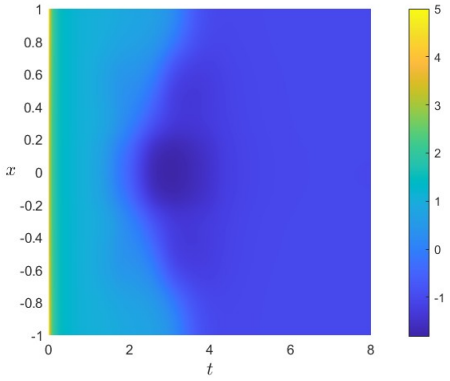


Figure 33: Over time evolution of y_M^c for $T = 8$.

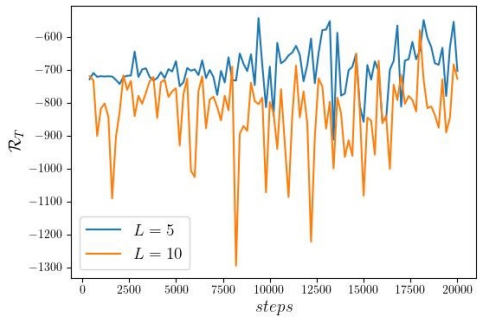


Figure 34: Best solution y_M^c vs. $L = 10$

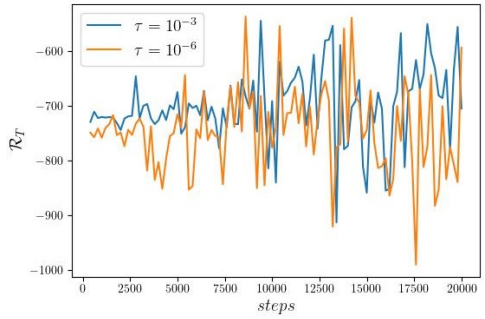


Figure 35: Best solution y_M^c vs. $\tau = 10^{-6}$

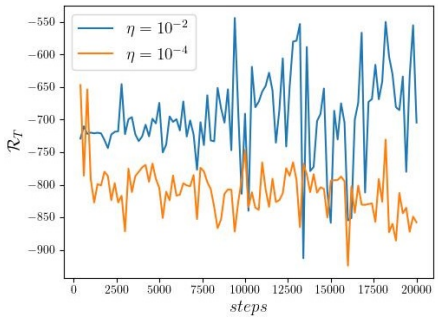


Figure 36: Best solution y_M^c vs. $\eta = 10^{-4}$

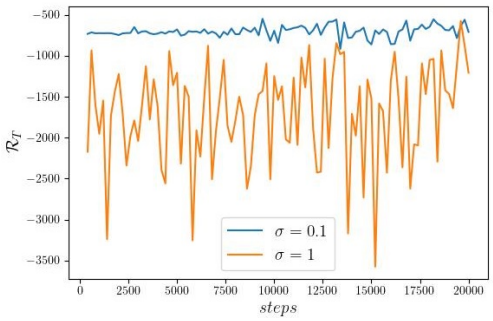


Figure 37: Best solution y_M^c vs. $\sigma = 1$

7.9.2 Conclusion

We analysed the previous examples to get a feeling if it was even possible to get solutions for optimal control problems, which are governed by linear parabolic PDEs. The question is now how the HJDQN method handles a non-linear problem. The best model we could obtain only achieved a very unstable solution regarding the development of $\|y_M^c\|_{L^2((0,4),\mathbb{R}^n)}$, which we can see in table 12. This is also reflected in the large variance for the control components $\left(\eta_u^M(t)\right)_i$ for $i \in \{1,2,3\}$ during the training process as we can see in figure 31.

According to figures 27 and 28, was the model capable to improve the reward in the beginning of the training process, but it then suffered a lot of fluctuation in the later stages and was thus only able to improve the reward slightly. If we compare the reward development of the best obtained solution to other models, where we only changed one hyper-parameter at a time, we see in figure 34 that a higher Lipschitz constant resulted expectedly in even more variance but not in an improvement of the reward. Similarly was the effect when choosing a better value for the Double Q-Learning parameter τ with 10^{-6}. Unfortunately was more variance for the exploration strategy also not successful in increasing the reward as we can see in figure 37.

Although the reward didn't increase much over the course of the training process, is this still not a definite sign that the algorithm failed completely. On the trained interval $[0,4]$, we can see in figures 29 and 30, that the model is obviously able to decrease η_y^M which can also be seen in 31. Nevertheless does it again overshoot zero and drives in the negative z-direction. We did see before in the linear two dimensional example, that this behaviour could be corrected by the algorithm. But if we check how the model generalizes beyond the training interval, we can see that the state converges towards $y = -1$ for $T = 8$ (figure 32 and 33). This is not surprising since (7.30) has in $y = -1$, in addition to $y = 0$ and $y = 1$, an equilibrium.

Unfortunately was this behaviour essentially observed for every trained model. A last strategy to improve the results could be again a concrete initial neural network configuration, which isn't trivial at all.

8 Outlook

All the conclusions had one similarity. Each models could potentiality benefit from an initial neural network configuration. There are a lot of different strategies involving different approaches. We want to give a quick overview for a few of them, for a greater overview we refer to [25].

We can generally divide the approaches into initialization with and without pre-training. We want to restrict ourselves to the initialization without pre-training. One of the most used approaches is to use random initialization for bias and weights. The random initialization can be divided into several different approaches, for example the

Interval based initialization: These methods try to select the weights and biases from an optimal interval or within a maximum bound. In [28] they introduced for example the so called Mutual Information Weight Initialization (MIWI) method, for feedforward neural networks, which use sigmoidal activation. The MIWI algorithm calculates mutual information (MI) between input and output variables, by assigning higher importance to variables with greater MI. The positive correlation between the sigmoid function and initial weights connecting the neurons, ensures alignment with the information distribution in the input variables, which contributes to stable performance. Furthermore establishes the algorithm bounds for initial weights, ensuring activation of all hidden neurons and also controlling convergence rates. This enhances the adaptability and optimization during the training process.

Next to random initialization exists also the data-driven initialization approach, where the network weights and biases are directly derived from the training data. An approach using k-means and PCA is given by

Data-Driven initialization approach: In [18] is a data-dependent initialization approach introduced, which ensures that all units in the network train at a uniform rate, mitigating problems associated with gradient behaviour. They analyse two data-dependent techniques for weight initialization, namely PCA-based and k-means based. Their strategy is to normalize the weights, such that they could achieve equally distributed activations and maintaining a consistent gradient ratio across all layers.

There are also methods which combine the randomized and data-driven approaches

Hybrid initialization approach: A hybrid approach is proposed in [2], where they initialize the weights of layers using orthonormal matrices and force the output of a layer to have a predetermined standard deviation using an initialization set. The novel idea is the usage of an additional hyperparameter, which is called the active fraction, this parameter allows to specify how likely it should be for a unit in a ReLU layer to produce non-zero activation.

These initialization approaches are in general either very specific to a special practical problem or to a specific neural network layer. Furthermore is the literature on initialization for Deep Q-Networks very limited and therefore far out of scope for this work. Apart from the neural network initialization we can also improve on the experience replay used in the DQN algorithm.

Prioritized Experience Replay: The DQN algorithm uses experience replay, to store past experiences and randomly sample them during training, to break temporal correlations. But not all experiences are equally informative. The prioritized experience replay introduced in [31], tries to improve this by assigning different priorities to experiences, based on their temporal-difference error. This prioritization improves the learning efficiency of the neural network, as experiences with higher importance are sampled more frequently. This method provided an performance and learning speed improvement for the DQN algorithm compared to traditional experience replay methods.

References

[1] Shmuel Agmon. *Lectures on elliptic boundary value problems*, volume 369. AMS Chelsea Publishing, 2010. doi: https://doi.org/10.1090/chel/369.

[2] Diego Aguirre and Olac Fuentes. Improving weight initialization of relu and output layers. In Igor V. Tetko, Vera Kurková, Pavel Karpov, and Fabian J. Theis, editors, *Artificial Neural Networks and Machine Learning - ICANN 2019: Deep Learning - 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17-19, 2019, Proceedings, Part II*, volume 11728 of *Lecture Notes in Computer Science*, pages 170–184. Springer, 2019. doi: 10.1007/978-3-030-30484-3_15. URL https://doi.org/10.1007/978-3-030-30484-3_15.

[3] Brian D. O. Anderson and John B. Moore. *Optimal Control: Linear Quadratic Methods*. Prentice-Hall, Inc., USA, 1990. ISBN 0136385605.

[4] Behzad Azmi and Karl Kunisch. A hybrid finite-dimensional RHC for stabilization of time-varying parabolic equations. *SIAM J. Control. Optim.*, 57(5):3496–3526, 2019. doi: 10.1137/19M1239787. URL https://doi.org/10.1137/19M1239787.

[5] Behzad Azmi, Dante Kalise, and Karl Kunisch. Optimal feedback law recovery by gradient-augmented sparse polynomial regression. *Journal of Machine Learning Research*, 22(48):1–32, 2021. URL http://jmlr.org/papers/v22/20-755.html.

[6] Igor A. Baratta, Joseph P. Dean, Jørgen S. Dokken, Michal Habera, Jack S. Hale, Chris N. Richardson, Marie E. Rognes, Matthew W. Scroggs, Nathan Sime, and Garth N. Wells. DOLFINx: the next generation FEniCS problem solving environment. preprint, 2023.

[7] Michael R. Berthold, Nicolas Cebron, Fabian Dill, Thomas R. Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Christoph Sieb, Kilian Thiel, and Bernd Wiswedel. KNIME: The Konstanz Information Miner. In *Studies in Classification, Data Analysis, and Knowledge Organization (GfKL 2007)*. Springer, 2007. ISBN 978-3-540-78239-1.

[8] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Rev.*, 60(2):223–311, 2018. doi: 10.1137/16M1080173. URL https://doi.org/10.1137/16M1080173.

[9] Alberto Bressan. Lecture notes: Viscosity solutions of hamilton-jacobi equations and optimal control problems, 2011. URL https://bpb-us-e1.wpmucdn.com/sites.psu.edu/dist/c/168431/files/2023/07/HJlnotes22.pdf.

[10] Frank H. Clarke. Generalized gradients and applications. *Transactions of the American Mathematical Society*, 205:247–262, 1975.

[11] Robert Denk and Reinhard Racke. *Kompendium der ANALYSIS*. 01 2011. ISBN 978-3-8348-1565-1. doi: 10.1007/978-3-8348-8184-7.

[12] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. https://github.com/openai/baselines, 2017.

[13] Hao Dong, Zihan Ding, and Shanghang Zhang. *Deep Reinforcement Learning Fundamentals, Research and Applications: Fundamentals, Research and Applications*. Springer Singapore, 2020. ISBN 978-981-15-4094-3. doi: 10.1007/978-981-15-4095-0.

[14] Magnus Gohm. Hjdqn. https://github.com/gohmmagn/HJDQN, 2024.

[15] The MathWorks Inc. Matlab version: R2023a, 2023. URL https://www.mathworks.com.

[16] Jeongho Kim, Jaeuk Shin, and Insoon Yang. Hamilton-jacobi deep q-learning for deterministic continuous-time systems with lipschitz continuous controls. *J. Mach. Learn. Res.*, 22:206:1–206:34, 2021. URL http://jmlr.org/papers/v22/20-1235.html.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

[18] Philipp Krähenbühl, Carl Doersch, Jeff Donahue, and Trevor Darrell. Data-dependent initializations of convolutional neural networks. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.06856.

[19] F. L. Lewis. Optimal control for singular systems. In *1985 24th IEEE Conference on Decision and Control*, pages 266–272, 1985. doi: 10.1109/CDC.1985.268843.

[20] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[21] Long-Ji Lin. *Reinforcement Learning for Robots Using Neural Networks*. PhD thesis, USA, 1992. UMI Order No. GAX93-22750.

[22] Jacques Louis Lions and Enrico Magenes. *Non-Homogeneous Boundary Value Problems and Applications - Vol. 1*. Springer Science and Business Media, Berlin Heidelberg, 2012. ISBN 978-3-642-65161-8.

[23] Sean Meyn. *Control Systems and Reinforcement Learning*. Cambridge University Press, 2022. doi: 10.1017/9781009051873.

[24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL http://arxiv.org/abs/1312.5602.

[25] Meenal V Narkhede, Prashant P Bartakke, and Mukul S Sutaone. A review on weight initialization strategies for neural networks. *Artificial intelligence review*, 55(1):291–322, 2022.

[26] Sophocles J. Orfanidis. Applied optimum signal processing, 2018. URL https://eceweb1.rutgers.edu/~orfanidi/aosp/. Updated Online Version of ISBN 0-07-047794-9.

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[28] Junfei Qiao, Sanyi Li, and Wenjing Li. Mutual information based weight initialization method for sigmoidal feedforward neural networks. *Neurocomputing*, 207:676–683, 2016. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2016.05.054.

[29] Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407, 1951. doi: 10.1214/aoms/1177729586. URL https://doi.org/10.1214/aoms/1177729586.

[30] Moonkyung Ryu, Yinlam Chow, Ross Anderson, Christian Tjandraatmadja, and Craig Boutilier. CAQL: continuous action q-learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=BkxXe0Etwr.

[31] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.05952.

[32] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin A. Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 387–395. JMLR.org, 2014. URL http://proceedings.mlr.press/v32/silver14.html.

[33] James E. Smith and Robert L. Winkler. The optimizer's curse: Skepticism and postdecision surprise in decision analysis. *Manag. Sci.*, 52(3):311–322, 2006. doi: 10.1287/MNSC.1050.0451. URL https://doi.org/10.1287/mnsc.1050.0451.

[34] Yuval Tassa and Tom Erez. Least squares solutions of the hjb equation with neural network value-function approximators. *IEEE Transactions on Neural Networks*, 18(4):1031–1041, 2007. doi: 10.1109/TNN.2007.899249.

[35] R. Temam. *Navier-Stokes Equations: Theory and Numerical Analysis*. Studies in mathematics and its applications. North-Holland, 1984. ISBN 9780444853080.

[36] Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierre, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023. URL https://zenodo.org/record/8127025.

[37] Hado van Hasselt. Double q-learning. In John D. Lafferty, Christopher K. I. Williams, John Shawe-Taylor, Richard S. Zemel, and Aron Culotta, editors, *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*, pages 2613–2621. Curran Associates, Inc., 2010. URL https://proceedings.neurips.cc/paper/2010/hash/091d584fced301b442654dd8c23b3fc9-Abstract.html.

[38] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2094–2100. AAAI Press, 2016. doi: 10.1609/AAAI.V30I1.10295. URL https://doi.org/10.1609/aaai.v30i1.10295.

[39] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992. ISSN 1573-0565. doi: 10.1007/BF00992698. URL https://doi.org/10.1007/BF00992698.

[40] Yongliang Yang, Donald Wunsch, and Yixin Yin. Hamiltonian-driven adaptive dynamic programming for continuous nonlinear dynamical systems. *IEEE Transactions on Neural Networks and Learning Systems*, 28:1929 – 1940, 08 2017. doi: 10.1109/TNNLS.2017.2654324.

[41] Jerzy Zabczyk. *Mathematical Control Theory - An Introduction*. Springer Science and Business Media, Berlin Heidelberg, 2009. ISBN 978-0-817-64733-9.