Journal of Mathematics in Industry
a SpringerOpen Journal

RESEARCH Open Access

A projected primal-dual gradient optimal control method for deep reinforcement learning

Simon Gottschalk[1*], Michael Burger[1] and Matthias Gerdts[2]

*Correspondence:
simon.gottschalk@itwm.fraunhofer.de
[1]Fraunhofer ITWM, Kaiserslautern, Germany
Full list of author information is available at the end of the article

Abstract

In this contribution, we start with a policy-based Reinforcement Learning ansatz using neural networks. The underlying Markov Decision Process consists of a transition probability representing the dynamical system and a policy realized by a neural network mapping the current state to parameters of a distribution. Therefrom, the next control can be sampled. In this setting, the neural network is replaced by an ODE, which is based on a recently discussed interpretation of neural networks. The resulting infinite optimization problem is transformed into an optimization problem similar to the well-known optimal control problems. Afterwards, the necessary optimality conditions are established and from this a new numerical algorithm is derived. The operating principle is shown with two examples. It is applied to a simple example, where a moving point is steered through an obstacle course to a desired end position in a 2D plane. The second example shows the applicability to more complex problems. There, the aim is to control the finger tip of a human arm model with five degrees of freedom and 29 Hill's muscle models to a desired end position.

MSC: 49K15; 90C40; 93E35

Keywords: Reinforcement learning; Optimal control; Necessary optimality conditions

1 Introduction

Techniques in order to solve optimal control problems are interesting for many operators in different research areas because of the prevalence of these problems. For instance, such problems naturally arise if one aims to steer a car to a desired end position or one searches the optimal activations of Hill's muscles of a digital human model. This explains the long standing history of contributions from mathematicians to this area. In particular, we emphasize on the development of the Pontryagin's maximum principle (cf. [1]) in 1956. For a detailed overview of the history of optimal control, we refer the reader to [2].

While in the last decade the importance of deep learning and its great potential became apparent, new approaches based on deep learning and tackling optimal control type problems came up as well. The interest in Reinforcement Learning (see e.g. [3]) did rapidly grow in the last years, because of its great potential to control many dynamical systems without knowing the exact equations of motion. This class of techniques, which focuses on closed loop control problems, is based on Markov Decision Processes and is able to handle systems implemented on a computer as well as real life system which can only be observed. Despite the slightly different starting point in well-known classical optimal control theory and Reinforcement Learning, strong connections between these topics are well-known (see e.g. [4]). We are convinced that the discussion in this contribution extends the number of connections.

Reinforcement Learning approaches, which are based on neural networks forming a policy, also called Deep Reinforcement Learning, are very common choices, because of their flexibility and their famous learning approaches. Successfully implemented examples are, for instance, shown in [5] and [6]. On the other hand, learning algorithms of neural networks are often criticized for being black box methods. Although neural networks are motivated by the neurons of the human brain, often, they do not seem to have an interpretable importance in the actual application. Thus, the interest in finding connections to well-established areas and new interpretations grows. For instance, the authors of [7] use an ODE-interpretation of neural networks in order to discuss the notion of stability for neural networks. In [8], a classification problem is interpreted as an optimal control problem by using the same ODE-interpretation.

The possibility to benefit from well-known research areas motivates us to combine the interpretation of the neural network as ODE and Reinforcement Learning. Then, in the Reinforcement Learning setting we derive an optimal control problem similar to the classical well-known one. Beside the ODE-interpretation, we focus on the transfer of necessary optimality conditions for optimal control problems into the Reinforcement Learning setting. As this is one of the main results in the area of optimal control, it might well open powerful new possibilities in the area of Reinforcement Learning.

2 Reinforcement learning

2.1 Markov decision process

We start our discussion focusing on the underlying structure of Reinforcement Learning (RL) approaches [3]. The whole theory is based on the assumption of having a Markov Decision Process (MDP). Such a MDP consists of a *state space* S and a *control space* A. Elements of the former one describe the state of an underlying system such as the gesture of a human arm or a position and velocity of a car. The control space contains admissible controls, which can influence these states like muscle actuations, acceleration of a car or a steering angle. Furthermore, we assume to have an *environment* and a *policy*. The environment takes the role of the system, which we aim to control. It gets the current control and generates a next state. This received state is plugged in into the policy, which provides the next control. In other words, one can think of a controller in a closed control loop. In the Reinforcement Learning community this controlling part is known as *agent*. However, the MDP differs in the realization of the environment and the policy from a usual closed control loop. The environment is assumed to be given by a transition distribution. Furthermore, the idea of the policy is, that it samples the next control from a distribution, which depends on the current state. A discussion about defining characteristics of robust and stochastic optimization can be found in [9]. Since we aim to find an optimal policy, we need to specify what optimal means. For this, every state-control pair of a trajectory is rated by a *reward function* and the expected sum over the rewards of one trajectory takes the role of an objective function. Having introduced the basic ideas, in a next step, we give a mathematical introduction, which is essentially based on the *Handbook of Markov Decision Processes* published by E.A. Feinberg and A. Shwartz [10]. We allow that the state and control spaces are continuous. This property is necessary since, most of the time, our applications are based on continuous spaces like a human arm model actuated by muscles, which get activations between zero and one. Overall, we assume that the state space S and control space A are measurable spaces endowed with corresponding σ-algebras $\mathcal{S}$ and $\mathcal{A}$. The reward function $r : S \times A \to \mathbb{R}$ is assumed to be a measurable function on $(S \times A, \mathcal{S} \times \mathcal{A})$. In the following, we assume that the absolute value of the reward is bounded by a constant C_{reward}. From our point of view, this is not a very restrictive assumption for most practical application cases, since very undesirable (respectively desirable) state-control pairs can still be rated with a huge negative (positive) reward. Solely unbounded rewards, which, most of the time, only appear for state-control pairs non-representative and far from optimal state-control pairs, are not possible anymore. Furthermore, we require a transition probability from $(S \times A, \mathcal{S} \times \mathcal{A})$ to $(S, \mathcal{S})$. This transition probability is represented by $K(B, x, u)$ for $(x, u) \in S \times A$ and $B \in \mathcal{S}$ and is assumed to fulfill the following two properties. Firstly, $K(\cdot, x, u)$ is a probability measure on $(S, \mathcal{S})$ for any $(x, u) \in S \times A$. Secondly, $K(B, \cdot, \cdot)$ is measurable on $(S \times A, \mathcal{S} \times \mathcal{A})$ for any $B \in \mathcal{S}$. At this point, we stress out that, despite the fact that we introduced the MDP for general measurable spaces, we restrict ourselves to the state space $\mathbb{R}^{n_s}$ and control space $\mathbb{R}^{n_a}$ or at least subsets of these in this contribution, which are suitable for the most application cases. For these spaces, we can use the Borel σ-algebra. Additionally, we assume for further discussions, that the transition probability is given by a transition density $p : S \times A \times S \to \mathbb{R}_+$ fulfilling the equation $K(B, x, u) = \int_B p(x'|x, u)\, dx'$. Having this at hand, we can define histories, respectively trajectories, $\tau_n = x_0 u_0 x_1 u_1 x_2 u_2 \ldots x_n, n \in \mathbb{N}$. The set $H_n = S \times (A \times S)^n$ contains all possible trajectories of length n. $\mathcal{H}_n = \mathcal{S} \times (\mathcal{A} \times \mathcal{S})^n$ denotes its σ-algebra. For the sake of completeness, we introduce $H_\infty = (S \times A)^\infty$ as the set of all trajectories of infinite length $\tau = x_0 u_0 x_1 u_1 x_2 u_2 \ldots$. Then, the policy π is introduced as a transition density. From this density, the next control can be sampled. The policy is called Markov if this transition density does not depend on the whole history, but only on the current state. This means, that π is a transition density from $(S, \mathcal{S})$ to $(A, \mathcal{A})$. In the end, the Ionescu Tulcea theorem [11] ensures the existence of a unique strategic measure $P^{\pi}_{p_0}$, which is a transition probability from $(S, \mathcal{S})$ to $(H_\infty, \mathcal{H}_\infty)$ (respectively $P^{\pi,n}_{p_0}$ from $(S, \mathcal{S})$ to $(H_n, \mathcal{H}_n)$) for a given initial density p_0 regarding the first state and any given policy π. This means that, for a measurable rectangle $F^S_0 \times \prod_{i=0}^{n-1}(F^A_i \times F^S_{i+1})$, we have that:

$$\begin{aligned} P^{\pi,n}_{p_0}\left[F^S_0 \times \prod_{i=0}^{n-1} F^A_i \times F^S_{i+1}\right] \\ = \int_{x_0 \in F^S_0} \int_{u_0 \in F^A_0} \cdots \int_{x_n \in F^S_n} p_0(x_0) \prod_{k=0}^{n-1} p(x_{k+1}|x_k, u_k)\pi(u_k|x_k)\, dx_n \ldots u_0\, dx_0. \end{aligned} \tag{1}$$

It follows that an expected value of a measurable function $R : H_n \to \mathbb{R}$ has the following form:

$$\begin{aligned} \mathbb{E}^{\pi,n}_{p_0}[R] = \int_{x_0 \in S} \int_{u_0 \in A} \cdots \int_{x_n \in S} p_0(x_0) \prod_{k=0}^{n-1} p(x_{k+1}|x_k, u_k)\pi(u_k|x_k) \ldots \\ \times R(x_0, u_0, x_1, u_1, \ldots, x_n)\, dx_n \ldots du_0\, dx_0. \end{aligned} \tag{2}$$

Since this expression is not very handy and bulky, we introduce a new notation representing parts of this expression. Instead of having a lot of integral symbols, we shorten this by writing only one integral symbol with H_n in the index representing the integral over the state and control spaces. Furthermore, we replace $dx_n \dots dx_2\, du_1\, dx_1\, du_0\, dx_0$ by $d\tau_n$:

$$\mathbb{E}_{p_0}^{\pi,n}[R] = \int_{H_n} p_0(x_0) \prod_{k=0}^{n-1} p(x_{k+1}|x_k, u_k)\pi(u_k|x_k)R(\tau_n)\, d\tau_n. \tag{3}$$

2.2 Goal of reinforcement learning

Having introduced the MDP in Sect. 2.1, we have everything at hand in order to formulate the optimization problem forming the starting point for the Reinforcement Learning [3]. As we have mentioned, the reward function is defined in order to rate the current situation. For example, if we are interested in controlling a car to a desired end-position as fast as possible, the reward function might be minus the distance between the current position of the car and the end-position. In the end, we aim to maximize the summed up reward of one trajectory. This, which is also called *total reward*, is computed as $R(\tau_n) := \sum_{k=0}^{n} r(x_k, u_k)$, where $\tau_n = x_0u_0x_1u_1x_2u_2 \dots x_n$ is a trajectory. The optimization problem then reads as:

$$\max_{\pi} \mathbb{E}_{p_0}^{\pi,n}\big[R(\tau_n)\big]. \tag{4}$$

We formulated the optimization problem for trajectories with finite length $n \in \mathbb{N}$ and focus on application cases, which have a certain finishing time. This is a necessary assumption for further considerations, where we need to define a function mapping from the space of trajectories.

Up to now, it is not clear how the optimal policy looks like and this makes the optimization problem difficult. Thus, we focus on parameterized policies in this contribution. To be more precise, we assume that the current state, which is plugged in into the policy, is inputted into a neural network. The outputs of the neural network are parameters of a Gaussian distribution, from which the next control can be sampled. This ansatz has already been considered and discussed, inter alia, in [3, 5]. This way, we do not search directly for the optimal policy anymore. Instead, we optimize the weights and biases (parameters θ) of the neural network in order to get an optimal policy. The policy defined by the parameters θ is denoted as π_θ and the optimization problem can be written as:

$$\arg\max_\theta \mathbb{E}_{p_0}^{\pi_\theta,n}\big[R(\tau_n)\big]. \tag{5}$$

Techniques in order to solve such an optimization problem are essentially based on the idea of initializing a policy, using this policy in the system simulation in order to generate trajectories, and updating the policy based on this data afterwards. Here, one distinguishes between model-based and model-free techniques. The model-free methods make use of observations for improving the policy directly. In the model-based case, the observations of simulations are used for system identification, while the identified system generates the trajectories for the policy update. These techniques are said to be more sample efficient than model-free techniques (see e.g. [12]). Furthermore, one can classify Reinforcement Learning techniques as policy-based or value-function based techniques. The latter one makes use of a value-function, which is successively improved and defines the policy indirectly. This can be observed in the famous Q-learning technique [13]. In contrast to value-function based techniques, policy-based methods directly update the (parameterized) policy. The very famous REINFORCE [14], based on a simple gradient ascent idea, and the approach, which will be introduced in this contribution, belong to this class of techniques.

2.3 Parameterized policy

Here, we have a closer look at the parameterized policy π_θ, which we mentioned in the previous subsection. It is very common to build a policy out of a neural network and a distribution, which gets the parameters from the neural network. For further discussions, we define the mapping $\mathrm{NN}_\theta : S \to A$ representing the neural network for fixed weights and biases θ. It gets the current state x_k as input. Then, the activation functions, which are in our case hyperbolic tangents, and the affine linear functions are applied successively as it will be further explained in Sect. 2.4. The outputs $\mu_k^\theta := \mathrm{NN}_\theta(x_k)$ are used as parameters of the distribution $\rho(u_k|\mu_k^\theta)$ being the second component of the policy. Therefrom, the next control u_k can be sampled. For a later discussion, we introduce the function $\mathcal{F}_\theta : H_n^S \to H_n^A$ mapping the whole state-trajectory, where H_n^S (respectively H_n^A) is the set of state-trajectories $\tau_n^x = [x_0^T, x_1^T, \dots x_n^T]^T \in \mathbb{R}^{n_s\cdot(n+1)}$ (respectively control-trajectories $\tau_n^u = [u_0^T, u_1^T, \dots u_n^T]^T \in \mathbb{R}^{n_a\cdot(n+1)}$):

$$\mu_{\tau_n}^{\theta,n} := \mathcal{F}_\theta\bigl(\tau_n^x\bigr) = \begin{bmatrix} \mathrm{NN}_\theta(x_0) \\ \mathrm{NN}_\theta(x_1) \\ \vdots \\ \mathrm{NN}_\theta(x_n) \end{bmatrix}. \tag{6}$$

We visualize the new introduced notation in Fig. 1. We see the structure of the policy and the corresponding mathematical function in this picture.

Before we go on and use these notations in order to rewrite the RL optimization problem into a new form, we introduce one additional definition:

$$P\bigl(\tau_n|\mu_{\tau_n}^{\theta,n}\bigr) := p_0(x_0)\prod_{k=0}^{n-1} p(x_{k+1}|x_k,u_k)\rho_\theta\bigl(u_k|\mu_k^\theta\bigr). \tag{7}$$

Obviously, only the policy depends on $\mu^{\theta,n}$. This means that if the density $\rho_\theta(u_k|\mu_k^\theta)$ is differentiable with respect to its parameters μ_k^θ, we know that the derivative of $P(\tau_n|\mu_{\tau_n}^{\theta,n})$ with respect to $\mu_{\tau_n}^{\theta,n}$ exists. Typically, $\rho_\theta(\cdot|\mu_k^\theta)$ is a density function of a Gaussian distribution with mean value μ_k^θ and a suitable fixed variance σ_k. In this case, the differentiability

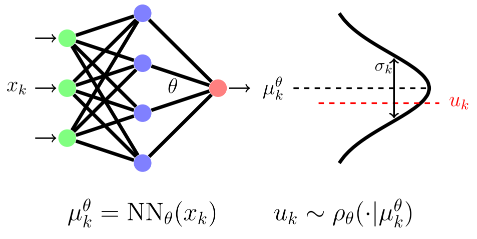
x_k
θ
μ_k^θ
σ_k
u_k
$\mu_k^\theta = \mathrm{NN}_\theta(x_k)$
$u_k \sim \rho_\theta(\cdot|\mu_k^\theta)$

Figure 1 Schematic representation of the policy

is ensured. Thus, from now on we assume to have this differentiability, which we will need in future discussions.

2.4 Neural network as an ODE

In this part, we have a closer look at continuous interpretations of neural networks, which we are going to use in order to motivate a new policy ansatz consisting of an ODE instead of a neural network. A classical neural network is defined by weight matrices W^i, biases b^i, where i describes the layer, and an activation function σ. From a mathematical point of view, a feedforward network works by the following computation rule:

$$z^{i+1} = \sigma\big(W^i z^i + b^i\big). \tag{8}$$

Another type of network is a residual neural network. The difference here is that the activation function together with the affine linear function $\sigma(W^i z^i + b^i)$ does not describe z^{i+1}, but the residuum between z^i and z^{i+1}. This means, that in this case the computation rule is as follows:

$$z^{i+1} = z^i + \sigma\big(W^i z^i + b^i\big). \tag{9}$$

The authors in [7] make use of the similarities between this computation rule and an explicit Euler sheme with step size one for the following ODE:

$$\frac{dz(s)}{ds} = \sigma\big(W(s)z(s) + b(s)\big), \tag{10}$$

$$z(s_0) = z_0. \tag{11}$$

Based on this connection between residual neural networks and ODEs, they discuss the notion of stability, which is defined for ODEs, in the neural network context. This motivates us to replace the neural network in the policy by this ODE. We will see in further sections, that this leads us to a continuous optimal control problem. Keep in mind, that the above mentioned connection can only be deduced, if the state and control dimension are the same. But we do not see this as a strong restriction, since we can use, for instance, an en- and decoder with neural networks [15] to circumvent this assumption. For a better readability, we derive the results in this contribution with residual neural networks. Note however that using the connection between ODEs and feedforward neural networks, which can be seen by replacing $\sigma(W(s)z(s) + b(s))$ by $\sigma(W(s)z(s) + b(s)) - z(s)$ in (10), or other types of neural networks leaves the following discussion unaffected.

The replacement of the neural network by the corresponding ODE leads to a deviation in the generated actions. The size of this deviation depends on the solution strategy of the ODE. If the ODE is solved numerically by an explicit Euler with step size one, the ODE-policy and the policy based on the neural network may not be distinguished from each other. If the ODE is solved analytically, classical error estimation results for the explicit Euler hold.

In order to be able to derive the continuous optimal control problem and to discuss the benefits of this connection in the Reinforcement Learning case, we introduce a solution operator. This solution operator $\mathcal{F} : H_n^S \times L^\infty \to W^{1,\infty}$ shall take the same role as $\mathcal{F}_\theta$ in the neural network case. This means, that the state-trajectory τ_n^x, whose components are the inputs for the neural network (see (6)), is now the initial value of the ODE we aim to solve with the solution operator. For the right-hand side we define:

$$f\big(z(s),\theta(s)\big) := \begin{bmatrix} \sigma\left(W(s)z_0(s)+b(s)\right) \\ \sigma\left(W(s)z_1(s)+b(s)\right) \\ \vdots \\ \sigma\left(W(s)z_n(s)+b(s)\right) \end{bmatrix}, \tag{12}$$

where $\theta(s)$ collects $W(s)$ and $b(s)$ and z_k are the entries of z that correspond to x_k of τ_n^x. Overall, the solution operator with $z = \mathcal{F}(z_0,\theta)$ is defined such that the following ODE-initial-value problem is fulfilled:

$$\frac{dz(s)}{ds} = f\big(z(s),\theta(s)\big) \quad \text{a.e. on } [s_0,s_f], \tag{13}$$

$$z(s_0) = z_0. \tag{14}$$

The first argument of the solution operator is the initial value. As we already mentioned, the initial value is the state-trajectory τ_n^x and we need this solution operator for many different trajectories. The second argument is the control $\theta \in L^\infty$. The operator $\mathcal{F}$ is well-defined for a bounded θ since for a fixed initial value the theorem of Picard Lindelöf (see e.g. [16]) guarantees the existence and uniqueness of a solution of the initial value problem. The necessary assumption of this theorem that the function f satisfies the Lipschitz condition is in general fulfilled since the activation function we consider is the hyperbolic tangent. Even other usual activation functions like the sigmoid or the ReLu function would lead to a fulfilled Lipschitz condition.

2.5 Reinforcement learning and optimal control

In the previous subsections, we discussed the underlying optimization problem of RL, possible policies and solution strategies. At this point, we aim to establish a novel connection to optimal control problems with differential equations as constraints. We start by reformulating the optimization problem (5) based on the firstly introduced neural network policy. Afterwards, we replace the parts where the neural network appears in the policy and receive an infinite optimization problem. We expect that this connection enables great possibilities to learn from an old and well established field. We start with the optimization problem in (5) and plug in our definition of the expected value. We obtain:

$$\arg\max_\theta \int_{H_n} p_0(x_0)\prod_{k=0}^{n-1} p(x_{k+1}|x_k,u_k)\pi_\theta(u_k|x_k)R(\tau_n)\,d\tau_n. \tag{15}$$

First, we can replace the maximization problem by a minimization problem. In order to do this, we spend a minus in the objective function. Now, we split the policy as discussed in Sect. 2.3. Thus, the objective function becomes a function depending on $\mu^{\theta,n}$, for which additional constraints need to be added:

$$\min_{\theta} -\int_{H_n} P\big(\tau_n|\mu_{\tau_n}^{\theta,n}\big)R^n(\tau_n)\,d\tau_n \tag{16}$$

$$\text{s.t. } \mu_{\tau_n}^{\theta,n} = \mathcal{F}_\theta\big(\tau_n^x\big), \quad \forall \tau_n^x \in H_n^S. \tag{17}$$

Now, we use that the neural network can be interpreted as an ODE (see Sect. 2.4). This means that for each state-trajectory τ_n^x an ODE is solved in order to get the corresponding $\mu_{\tau_n}^{\theta,n}$. The right-hand side is given as in (12) and we get:

$$\frac{dz(s)}{ds} = f\big(z(s),\theta(s)\big) \quad \text{a.e. on } [s_0,s_f], \tag{18}$$

$$z(s_0) = \tau_n^x. \tag{19}$$

We make use of the solution operator of this differential equation, which we introduced in Sect. 2.4. In our optimization problem, $\mathcal{F}$ will replace $\mathcal{F}_\theta$. This induces the need for the pseudo time s. While we have finite parameters in the optimization problem (16)–(17), now, $\theta \in L^\infty$ depends on s. We do not have a finite optimization problem anymore. Overall, we get:

$$\text{Find } \theta \in L^\infty \text{ that minimizes} \tag{20}$$

$$J(\theta) := -\int_{H_n} P\big(\tau_n|z_{\tau_n}^{\theta,n}(s_f)\big)R(\tau_n)\,d\tau_n, \tag{21}$$

$$\text{s.t. } z_{\tau_n}^{\theta,n}(s) = \mathcal{F}\big(\tau_n^x,\theta\big)(s), \quad \forall \tau_n^x \in H_n^S. \tag{22}$$

$$\theta(s) \in U \quad \text{a.e. on } [s_0,s_f]. \tag{23}$$

The constraint (23) represents our assumption that θ is bounded, which we need for the existence of the solution operator. Typically, this will be a box constraint. We conclude this section by summarizing the assumptions we needed so far and we assume to hold in the remaining of this paper:

1. The state and action space are (subsets of) Euclidean spaces,
2. the considered time horizon is finite,
3. the reward is bounded by a constant C_{reward},
4. the control is bounded $\theta(s) \in U$ a.e. on $[s_0,s_f]$,
5. the activation functions are sufficiently smooth (e.g. tanh,sigmoid), and
6. the distribution embedded into the policy is differentiable w.r.t. its parameters (e.g. Gaussian distribution is differentiable w.r.t. its mean value).

3 Optimal control problems

At this point, we will discuss the connection to well-established optimal control theory. In order to do this, we make a short excursus to the optimal control formulation and the form of the optimization problem, which is typically considered in this case.

The information about the goal is mainly encoded in the objective function $J : \mathbb{R}^{n_z} \to \mathbb{R}$. It is defined by a mapping $\Phi : \mathbb{R}^{n_z} \to \mathbb{R}$ describing the costs in the end position and a mapping $\phi : \mathbb{R} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_\theta} \to \mathbb{R}$ representing the costs in each time step. The natural numbers n_z and n_θ describe the dimension of the state space and the control space. The constraint $\Psi : \mathbb{R}^{n_z} \to \mathbb{R}^{n_\Psi}$ for some $n_\Psi \in \mathbb{N}$ can be used to define a desired end state of the underlying system. Overall, we end up with an optimization of the following form:

$$\min_{\theta \in L^\infty} J(\theta) := \Phi\big(z(s_f)\big) + \int_{s_0}^{s_f} \phi\big(s, z(s), \theta(s)\big)\, ds \tag{24}$$

$$\text{s.t. } z(s) = \mathcal{F}^*(\theta)(s) \tag{25}$$

$$\Psi\big(z(s_f)\big) = 0, \tag{26}$$

$$\theta(s) \in U \subseteq \mathbb{R}^{n_\theta} \quad \text{a.e. on } [s_0, s_f]. \tag{27}$$

Here, $\mathcal{F}^* : L^\infty \to W^{1,\infty}$ denotes the solution operator of an initial value problem $\dot{z} = f^*(z,\theta), z(s_0) = z_0$, where $W^{1,\infty}$ represents the set of absolutely continuously functions $z : [s_0, s_f] \to \mathbb{R}^{n_z}$:

$$\begin{aligned} W^{1,\infty} &= W^{1,\infty}\big([s_0, s_f], \mathbb{R}^{n_z}\big) \\ &:= \big\{ z : [s_0, s_f] \to \mathbb{R}^{n_z} | z \text{ absolutely continuously with } \|z\|_{1,\infty} < \infty \big\}. \end{aligned} \tag{28}$$

The problem statement of this infinite optimization problem and discretization methods are discussed in [17]. Furthermore, one can find a derivation and discussion of a local minimum principle, which we state here, since the conditions resulting from this minimum principle can be used in order to get possible candidates for the optimal solution. Then, the quality of those candidates can be verified using the original optimization problem. Using the Hamiltonian function $\mathcal{H} : \mathbb{R} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_\theta} \times \mathbb{R}^{n_x} \times \mathbb{R} \to \mathbb{R}$ with $\mathcal{H}(s, z, \theta, \lambda, l_0) := l_0 \phi(s, z, \theta) + \lambda^T f(z, \theta)$ these conditions take the following form.

Theorem 1 (Necessary Optimality Conditions [17]) *Let $U \subseteq \mathbb{R}^{n_\theta}$ be closed and convex and let $\hat{\theta} \in L^\infty$ be a local minimum of the problem (24)–(27). Then, there exist multipliers $l_0 \in \mathbb{R}, \sigma \in \mathbb{R}^m, \lambda \in W^{1,\infty}$ such that:*

$$l_0 \geq 0, \quad (l_0, \sigma, \lambda) \neq (0,0,0), \tag{29}$$

$$\dot{\lambda}(s) = -\mathcal{H}_z\big(s, \mathcal{F}^*(\hat{\theta})(s), \hat{\theta}(s), \lambda(s), l_0\big)^T, \tag{30}$$

$$\lambda(s_f)^T = l_0 \Phi_z\big(\mathcal{F}^*(\hat{\theta})(s_f)\big) - \sigma^T \Psi_z\big(\mathcal{F}^*(\hat{\theta})(s_f)\big), \tag{31}$$

$$\mathcal{H}_\theta\big(s, \mathcal{F}^*(\hat{\theta})(s), \hat{\theta}(s), \lambda(s), l_0\big)\big(\theta - \hat{\theta}(s)\big) \geq 0, \quad \forall \theta \in U \text{ a.e. on } [s_0, s_f]. \tag{32}$$

The two mentioned optimization problems ((20)–(23) and (24)–(27)) share many common properties. First, we remark that the objective function of the former optimization problem is a special case of the classical objective function. In order to see this, one can define:

$$\Phi\big(z(s_f)\big) := -\int_{H_n} P\big(\tau_n | z_{\tau_n}^{\theta,n}(s_f)\big) R(\tau_n)\, d\tau_n \quad \text{and} \tag{33}$$

$$\phi\big(s, z(s), \theta(s)\big) := 0, \quad \forall s \in [s_0, s_f]. \tag{34}$$

Furthermore, we remark, that the constraint in (26) does not appear in the Reinforcement Learning optimization problem. The biggest difference and the reason, why we cannot apply the results of the local minimum principle to this optimization problem directly, lies in the differential equations. In the classical case, we only have one differential equation with one fixed initial value. In the RL case, we have infinitely many differential equations, since one for each trajectory appears. Note that this depends on the assumption that the state and control space are continuous sets. If this was not the case and the state and control space were finite sets, the integrals would vanish and they would be replaced by sums. In the constraints, we would have one ODE for each trajectory. But, the number of trajectories would be finite and the ODEs could be collected in one ODE with a huge dimensionality. In this case, the classical necessary optimality conditions could be applied, since the RL optimization problem would be a special case of the classical one. Nevertheless, in our case, we need to spend effort to receive similar results.

4 Derivation of the necessary optimality conditions

This section is devoted to the derivation of a type of necessary optimality conditions for the RL optimization problem (20)–(23).

4.1 Basic results from functional analysis

The proof of the necessary optimality conditions from the previous section is essentially based on the Fritz–John conditions for optimization problems in Banach spaces. Since we are inspired by this proof for the necessary optimality conditions of the RL problem, we recall the meaning of the Fritz–John conditions here. The starting point is the following form of an optimization problem on Banach spaces:

$$\min_{\hat{x}\in X} J(\hat{x}) \tag{35}$$

$$\text{s.t. } \hat{x} \in S, \tag{36}$$

$$G(\hat{x}) \in K, \tag{37}$$

$$H(\hat{x}) = 0_Z. \tag{38}$$

Here, X, Y and Z are Banach spaces and $J : X \to \mathbb{R}, G : X \to Y, H : X \to Z$. Furthermore, $S \subset X$ is a closed and convex subset, $K \subset Y$ a closed and convex cone with vertex 0_y. Both, S and K, are not allowed to have empty interiors. While the theorem of Weierstraß ([17], p. 78) guarantees that a lower semi-continuous J achieves its minimum on a compact set $\{x \in X | x \in S, G(\hat{x}) \in K, H(\hat{x}) = 0_Z\}$, the Fritz–John theorem provides optimality conditions (cf. [18, 19]).

Theorem 2 (Fritz–John Conditions) *Let J and G Fréchet-differentiable and H be continuously Fréchet-differentiable. Let $\hat{x}$ be a local minimum of problem* (35)–(38). *Assume that* $\operatorname{Im}(H'(\hat{x})) \subset Z$ *is not a proper dense subset. Then, there exist non-trivial multipliers* $(0,0,0) \neq (l_0, \lambda^*, \mu^*) \subset \mathbb{R} \times Y^* \times Z^*$ *with* $l \geq 0, \lambda^*(G(\hat{x})) = 0, \lambda^* \in K^+$ *and*:

$$l_0 J'(\hat{x})(d) - \lambda^*\bigl[G'\bigl(\hat{x}(d)\bigr)\bigr] - \mu^*\bigl[H'(\hat{x})(d)\bigr] \geq 0, \quad \forall d \in S - \{\hat{x}\}. \tag{39}$$

If a Banach space is equipped with a star, the dual space is meant. The dual cone $K^+ := \{x^* \in X^* | x^*(x) \geq 0, \forall x \in X\}$ represents the set of functionals of the dual space, which map all elements of the cone K into the positive real numbers or zero. This result will be the starting point for deriving the necessary optimality conditions.

4.2 Necessary optimality conditions

Finally, in this subsection, we derive the necessary optimality conditions. The most challenging part is to show, that the assumptions of the Fritz–John conditions are fulfilled. Especially, the Fréchet-differentiability of our objective function needs to be shown. Because of the chain rule, we also need to know the Fréchet-derivative of the newly introduced solution operator $\mathcal{F}$ with respect to the optimization variables θ. This will be the next step. Fortunately, we have a special case of the solution operator discussed in [20]. There, the author shows the Fréchet-differentiability of a solution operator for an initial value problem and determines the derivative. Despite the fact that our solution operator additionally depends on the initial value, the proof can be applied here since we can consider the initial value as varying but fix.

Lemma 1 ([20], pp. 56–59) *For a fixed $z_0 \in H_n^S$ and a sufficiently smooth right-hand-side f the Fréchet-derivative of $\mathcal{F}_{z_0} : L^\infty \to W^{1,\infty}$, $\mathcal{F}_{z_0}(\theta) := \mathcal{F}(z_0, \theta)$ w.r.t. θ is given as follows*:

$$\mathcal{F}'_{z_0}(\hat{\theta})(\delta\theta) = \delta z_{z_0},$$

where δz_{z_0} is given by the following ODE:

$$\delta\dot{z}_{z_0}(s) = f_z\big(\hat{z}(s), \hat{\theta}(s)\big)\delta z_{z_0}(s) + f_\theta\big(\hat{z}(s), \hat{\theta}(s)\big)\delta\theta(s) \quad \text{a.e. on } [s_0, s_f],$$
$$\delta z_{z_0}(s_0) = 0.$$

The smoothness assumption in Lemma 1 allows, for instance, the activation functions hyperbolic tangent and sigmoid but prohibit using the ReLU function. Nevertheless, in case of a ReLU function, the following discussion can be made for a smooth approximation of the ReLU function.

We have seen in Theorem 1 that necessary optimality conditions are described with the help of the adjoint variable λ, which itself needs to fulfill an ODE. Also in our case, we make use of an auxiliary definition in order to express the necessary optimality conditions. But, like in the case of $\mathcal{F}$, we define a solution operator for an ODE and need to allow varying initial values. We introduce the operator $\mathcal{G} : H^S \times W^{1,\infty} \times L^\infty \to W^{1,\infty}$ such that $\lambda = \mathcal{G}(\lambda_{s_f}, z, \theta)$ solves the ODE-end-value problem:

$$\frac{d\lambda(s)}{ds} = -\big(\lambda^T(s) f_z\big(z(s), \theta(s)\big)\big)^T, \tag{40}$$
$$\lambda(s_f) = \lambda_{s_f}. \tag{41}$$

Again, the existence and uniqueness are guaranteed for bounded θ by the theorem of Picard Lindelöf (see e.g. [16]). To see this, one has to notice that $-\lambda^T f_z(z, \theta)$ satisfies the Lipschitz-property w.r.t. λ, since it is linear in λ and f_z is bounded.

At this point, we have everything at hand to derive the necessary optimality conditions. We stress out that parts and ideas of the proof of the classical necessary optimality conditions in Theorem 1 inspire the following proof. For example, in our proof, we will use the Fritz–John conditions in Theorem 2 as starting point of our proof and continue by rewriting the inequality condition afterwards. Parts of these steps are motivated by [20] and [17]. Nevertheless, crucial parts of the following proof are new. The most important innovation is the derivation of the Fréchet-derivative of the objective function, which is needed in order to satisfy the assumptions of the Fritz–John conditions.

Theorem 3 *Let $U \subseteq \mathbb{R}^{n_\theta}$ for $n_\theta \in \mathbb{N}$ be closed and convex and let $\hat{\theta} \in L^\infty$ be a local minimum of the optimal control problem*:

$$\begin{aligned}
&\textit{Find } \theta \in L^\infty \textit{ that minimizes} \\
&\quad J(\theta) := -\int_{H_n} P\bigl(\tau_n | \mathcal{F}\bigl(\tau_n^x, \theta\bigr)(s_f)\bigr) R(\tau_n)\, d\tau_n \\
&\quad \theta(s) \in U \quad \textit{a.e. on } [s_0, s_f].
\end{aligned}$$

Furthermore, we assume that $\sup_\theta |\nabla_z P(\tau_n|z)|_{z=F(\tau_n^x,\theta)(s_f)}(\delta z_{\tau_n^x}(s_f))|$ *is Lebesgue integrable. Here, $\delta z_{\tau_n^x}$ denotes the Fréchet derivative of the solution operator from Lemma 1 with $z_0 = \tau_n^x$. Then, the following local minimum principle holds*:

$$\begin{aligned}
&\mathbb{E}_{p_0}^{\pi_\theta,n}\bigl[R(\tau_n)\lambda(\tau_n,\hat{\theta})^T(s) f_\theta\bigl(z(\tau_n,\hat{\theta})(s),\hat{\theta}(s)\bigr)\bigr]\bigl(\theta - \hat{\theta}(s)\bigr) \geq 0, \\
&\quad \forall \theta \in U \textit{ a.e. on } [s_0, s_f],
\end{aligned} \tag{42}$$

with

$$\begin{aligned}
&z(\tau_n,\hat{\theta}) = \mathcal{F}\bigl(\tau_n^x,\hat{\theta}\bigr), \\
&\lambda(\tau_n,\hat{\theta}) = \mathcal{G}\bigl(-\nabla_z \log\bigl(P(\tau_n|z)\bigr)^T|_{z=\mathcal{F}(\tau_n^x,\hat{\theta})(s_f)}, \mathcal{F}\bigl(\tau_n^x,\hat{\theta}\bigr),\hat{\theta}\bigr).
\end{aligned}$$

Proof First, we have a closer look at the problem statement (35)–(38). We recognize that we have a special case of such a problem. We also have an objection function $J : L^\infty \to \mathbb{R}$, which we want to minimize. The function G and H in (37)–(38) do not appear in our optimization problem. Since we do not have the function in the constraints, we do not care about a convex cone K. Overall, we have a special case of this problem statement and thus, we can check the assumptions of Theorem 2. We notice, that if we can ensure that J is Fréchet-differentiable, the assumptions are satisfied. We claim that $J'(\theta)(\delta\theta) = -\int_{H_n} \nabla_z P(\tau_n|z)|_{z=\mathcal{F}(\tau_n^x,\theta)(s_f)}(\delta z_{\tau_n^x}(s_f)) R(\tau_n)\, d\tau_n$ is the Fréchet-derivative of the objective function. We remark, that it is linear and continuous with respect to $\delta\theta$ since only $\delta z_{\tau_n^x}(s_f)$ depends on $\delta\theta$ and this is the Fréchet derivative from Lemma 1. Thus, it is linear and continuous with respect to $\delta\theta$. It remains to show that the defining equation of a Fréchet derivative is fulfilled:

$$J(\theta + \delta\theta) - J(\theta) = J'(\theta)(\delta\theta) + o\bigl(\|\delta\theta\|\bigr).$$

In order to do this, we start by considering the following difference:

$$\begin{aligned}
&J(\theta+\delta\theta)-J(\theta)\\
&\quad=-\int_{H_n} P\bigl(\tau_n|\mathcal{F}\bigl(\tau_n^x,\theta+\delta\theta\bigr)(s_f)\bigr)R(\tau_n)\,d\tau_n\\
&\qquad+\int_{H_n} P\bigl(\tau_n|\mathcal{F}\bigl(\tau_n^x,\theta\bigr)(s_f)\bigr)R(\tau_n)\,d\tau_n\\
&\quad=-\int_{H_n}\bigl(P\bigl(\tau_n|\mathcal{F}\bigl(\tau_n^x,\theta+\delta\theta\bigr)(s_f)\bigr)-P\bigl(\tau_n|\mathcal{F}\bigl(\tau_n^x,\theta\bigr)(s_f)\bigr)\bigr)R(\tau_n)\,d\tau_n\\
&\quad=-\int_{H_n}\nabla_z P(\tau_n|z)|_{z=\mathcal{F}(\tau_n^x,\theta)(s_f)}\bigl(\mathcal{F}\bigl(\tau_n^x,\theta+\delta\theta\bigr)(s_f)-\mathcal{F}\bigl(\tau_n^x,\theta\bigr)(s_f)\bigr)R(\tau_n)\\
&\qquad+\underbrace{T_{\text{rest}}}_{\in o(\|\mathcal{F}(\tau_n^x,\theta+\delta\theta)(s_f)-\mathcal{F}(\tau_n^x,\theta)(s_f)\|)}R(\tau_n)\,d\tau_n,
\end{aligned}$$

where T_{rest} represents the rest term. The last equation holds since P is Fréchet-differentiable with respect to its second argument. Furthermore, if we substract the expression, which we aim to identify as derivative, we get:

$$\begin{aligned}
&J(\theta+\delta\theta)-J(\theta)-\left(-\int_{H_n}\nabla_z P(\tau_n|z)|_{z=\mathcal{F}(\tau_n^x,\theta)(s_f)}\bigl(\delta z_{\tau_n^x}(s_f)\bigr)R(\tau_n)\,d\tau_n\right)\\
&\quad=-\int_{H_n}\nabla_z P(\tau_n|z)|_{z=\mathcal{F}(\tau_n^x,\theta)(s_f)}\\
&\qquad\times\underbrace{\bigl(\mathcal{F}\bigl(\tau_n^x,\theta+\delta\theta\bigr)(s_f)-\mathcal{F}\bigl(\tau_n^x,\theta\bigr)(s_f)-\delta z_{\tau_n^x}(s_f)\bigr)}_{T:=}R(\tau_n)\\
&\qquad+T_{\text{rest}}R(\tau_n)\,d\tau_n.
\end{aligned}\tag{43}$$

From the Fréchet-differentiability in Lemma 1 we know that T grows slower than $\delta\theta$, i.e. $T\in o(\|\delta\theta\|)$. Since $\delta z_{\tau_n^x}$ is linear with respect to $\delta\theta$, we have the following property:

$$\mathcal{F}\bigl(\tau_n^x,\theta+\delta\theta\bigr)-\mathcal{F}\bigl(\tau_n^x,\theta\bigr)=\underbrace{\delta z_{\tau_n^x}+\underbrace{T_{\mathcal{F}}}_{\in o(\|\delta\theta\|)}}_{\in\mathcal{O}(\|\delta\theta\|)}.\tag{44}$$

Again, $T_{\mathcal{F}}$ denotes the rest term. Summarizing our observations, we end up with:

$$T_{\text{rest}}\in o\bigl(\bigl\|\mathcal{F}\bigl(\tau_n^x,\theta+\delta\theta\bigr)(s_f)-\mathcal{F}\bigl(\tau_n^x,\theta\bigr)(s_f)\bigr\|\bigr)\quad\text{and}\tag{45}$$

$$\mathcal{F}\bigl(\tau_n^x,\theta+\delta\theta\bigr)(s_f)-\mathcal{F}\bigl(\tau_n^x,\theta\bigr)(s_f)\in\mathcal{O}\bigl(\|\delta\theta\|\bigr)\tag{46}$$

$$\Rightarrow T_{\text{rest}}\in o\bigl(\|\delta\theta\|\bigr).\tag{47}$$

In the end, we receive for equation (43) an integral over H_n applied to an expression, which is an element of $o(\|\delta\theta\|)$. Because of the Lebesgue integrability assumption and the fact that $\int_{H_n} |P(\tau_n|\mathcal{F}(\tau_n^x,\theta)(s_f))R(\tau_n)|\, d\tau_n \le (n+1)C_{\text{reward}}$, we can make use of the dominated convergence theorem. This provides that the whole expression (43) is an element of $o(\|\delta\theta\|)$. Overall, we showed the Fréchet-differentiability and determine the derivative of the objective function by showing:

$$\begin{aligned} &J(\theta+\delta\theta) - J(\theta) \\ &\quad = \biggl(-\int_{H_n} \nabla_z P(\tau_n|z)|_{z=\mathcal{F}(\tau_n^x,\theta)(s_f)} \bigl(\delta z_{\tau_n^x}(s_f)\bigr) R(\tau_n)\, d\tau_n\biggr) + o\bigl(\|\delta\theta\|\bigr). \end{aligned}$$

Now, all assumptions of the Fritz–John-Necessary-Optimality-Conditions are fulfilled and we can apply it to our optimization problem. From the left-hand-side of the inequality (39) only the objective function remains. In the end, we have, that the Fréchet-derivative at $\hat{\theta}$ applied to a small change $\delta\theta := \theta - \hat{\theta}, \theta \in L^\infty$ needs to be greater or equal to zero:

$$0 \le J'(\hat{\theta})(\delta\theta) = -\biggl(\int_{H_n} P\bigl(\tau_n|\mathcal{F}\bigl(\tau_n^x,\cdot\bigr)\bigr)R(\tau_n)\, d\tau_n\biggr)'(\hat{\theta})(\delta\theta) \tag{48}$$

$$= -\int_{H_n} \nabla_z P(\tau_n|z)|_{z=\mathcal{F}(\tau_n^x,\hat{\theta})(s_f)} R(\tau_n)\bigl(\delta z_{\tau_n^x}(s_f)\bigr)\, d\tau_n \tag{49}$$

$$= -\int_{H_n} P\bigl(\tau_n|\mathcal{F}\bigl(\tau_n^x,\hat{\theta}\bigr)(s_f)\bigr) \nabla_z \log\bigl(P(\tau_n|z)\bigr)|_{z=\mathcal{F}(\tau_n^x,\hat{\theta})(s_f)} R(\tau_n)\bigl(\delta z_{\tau_n^x}(s_f)\bigr)\, d\tau_n \tag{50}$$

$$= \int_{H_n} P\bigl(\tau_n|\mathcal{F}\bigl(\tau_n^x,\hat{\theta}\bigr)(s_f)\bigr)R(\tau_n)\lambda^T(\tau_n,\hat{\theta})(s_f)\bigl(\delta z_{\tau_n^x}(s_f)\bigr)\, d\tau_n. \tag{51}$$

We plugged in the derivative, which we determined in the beginning of this proof. Afterwards, we made use of the fact, that the derivative of the logarithm w.r.t. the argument is one over the argument (see (49)–(50)) and finally the definition of the end value of $\lambda(\tau_n,\hat{\theta})$ led us to equation (51). Before we continue, at this point, we make two remarks. First, we have a closer look at the partial integration:

$$\begin{aligned} &\int_{s_0}^{s_f} \dot{\lambda}^T(\tau_n,\hat{\theta})(s)\delta z_{\tau_n^x}(s)\, ds \\ &\quad = \lambda^T(\tau_n,\hat{\theta})(s_f)\delta z_{\tau_n^x}(s_f) - \int_{s_0}^{s_f} \lambda^T(\tau_n,\hat{\theta})(s)\delta \dot{z}_{\tau_n^x}(s)\, ds. \end{aligned} \tag{52}$$

As a second remark, we recap that we know from Lemma 1:

$$\begin{aligned} &\delta\dot{z}_{\tau_n^x}(s) \\ &\quad = f_z\bigl(z(\tau_n,\hat{\theta})(s),\hat{\theta}(s)\bigr)\delta z_{\tau_n^x}(s) + f_\theta\bigl(z(\tau_n,\hat{\theta})(s),\hat{\theta}(s)\bigr)\delta\theta(s). \end{aligned} \tag{53}$$

Using integration by parts and the property in (53) in order to rewrite (51) is motivated by the proof of the necessary conditions for an optimal control problem with a DDAE initial value problem in ([20], pp. 88–94). Thus, we plug in (52) and (53) into (51). This, together with the ODE (40) defining λ, explains the following transformations:

$$
\begin{aligned}
0 &\le J'(\hat{\theta})(\delta\theta) && (54)\\
&= \int_{H_n} P\big(\tau_n | \mathcal{F}\big(\tau_n^x, \hat{\theta}\big)(s_f)\big) R(\tau_n) \int_{s_0}^{s_f} \dot{\lambda}^T(\tau_n, \hat{\theta})(s)\delta z_{\tau_n^x}(s) + \lambda^T(\tau_n, \hat{\theta})(s)\delta \dot{z}_{\tau_n^x}(s)\, ds\, d\tau_n && (55)\\
&= \int_{H_n} P\big(\tau_n | \mathcal{F}\big(\tau_n^x, \hat{\theta}\big)(s_f)\big) R(\tau_n) \int_{s_0}^{s_f} -\lambda^T(\tau_n, \hat{\theta})(s) f_z\big(z(\tau_n, \hat{\theta})(s), \hat{\theta}(s)\big)\delta z_{\tau_n^x}(s)\\
&\quad + \lambda^T(\tau_n, \hat{\theta})(s)\big(f_z\big(z(\tau_n, \hat{\theta})(s), \hat{\theta}(s)\big)\delta z_{\tau_n^x}(s) + f_\theta\big(z(\tau_n, \hat{\theta})(s), \hat{\theta}(s)\big)\delta\theta(s)\big)\, ds\, d\tau_n && (56)\\
&= \int_{H_n} P\big(\tau_n | \mathcal{F}\big(\tau_n^x, \hat{\theta}\big)(s_f)\big) R(\tau_n) \int_{s_0}^{s_f} \lambda^T(\tau_n, \hat{\theta})(s) f_\theta\big(z(\tau_n, \hat{\theta})(s), \hat{\theta}(s)\big)\delta\theta(s)\, ds\, d\tau_n && (57)\\
&= \mathbb{E}_{\tau_n}\bigg[R(\tau_n) \int_{s_0}^{s_f} \lambda^T(\tau_n, \hat{\theta})(s) f_\theta\big(z(\tau_n, \hat{\theta})(s), \hat{\theta}(s)\big)\delta\theta(s)\, ds\bigg]. && (58)
\end{aligned}
$$

Once more, we received an expected value in the end. At this point, we are almost done. It remains to show, that the claim of this theorem follows from the proved above. In order to show this, inspired by ([17], p. 117), we introduce a control $\theta_\epsilon^\theta \in L^\infty$ depending on $\epsilon \ge 0$, which obviously satisfies the inequality discussed so far in this proof. This control θ_ϵ^θ is defined for a fixed $s \in [s_0, s_f]$ and fixed, but arbitrary $\theta \in U$:

$$
\theta_\epsilon^\theta(\tilde{s}) := \begin{cases} \theta, & \tilde{s} \in [s, s+\epsilon],\\ \hat{\theta}(\tilde{s}), & \text{else.} \end{cases}
$$

We will show that from plugging θ_ϵ^θ into the inequality from above, we can deduce that the inequality of the claim of this theorem is satisfied as well. This means that we will show:

$$
\begin{aligned}
&0 \le \mathbb{E}\bigg[R(\tau_n) \int_{s_0}^{s_f} \lambda_{\tau_n}^T(s) f_\theta\big(z_{\tau_n}(s), \hat{\theta}(s)\big)\big(\theta_\epsilon^\theta(s) - \hat{\theta}(s)\big)\, ds\bigg],\\
&\qquad \forall \theta_\epsilon^\theta \text{ with } \epsilon \ge 0, \forall \theta \in U\\
&\Rightarrow 0 \le \mathbb{E}\big[R(\tau_n)\lambda_{\tau_n}^T(s) f_\theta\big(z_{\tau_n}(s), \hat{\theta}(s)\big)\big]\big(\theta - \hat{\theta}(s)\big), \quad \forall \theta \in U \text{ a.e. on } [s_0, s_f]. \qquad (59)
\end{aligned}
$$

Before we can show this, we need to define two auxiliary functions:

$$
\begin{aligned}
h_1(s) &:= \int_{s_0}^{s} \lambda_{\tau_n}^T(\tilde{s}) f_\theta\big(z_{\tau_n}(\tilde{s}), \theta(\tilde{s})\big)\, d\tilde{s},\\
h_2(s) &:= \int_{s_0}^{s} \lambda_{\tau_n}^T(\tilde{s}) f_\theta\big(z_{\tau_n}(\tilde{s}), \theta(\tilde{s})\big)\hat{\theta}(\tilde{s})\, d\tilde{s}.
\end{aligned}
$$

Note, that the function h_1 as well as h_2 are differentiable with respect to the pseudo time s and it holds that $h_1'(s) = \lambda_{\tau_n}^T(s) f_\theta(z_{\tau_n}(s), \theta(s))$, respectively $h_2'(s) = \lambda_{\tau_n}^T(s) f_\theta(z_{\tau_n}(s), \theta(s))\hat{\theta}(s)$. Having all the preparations at hand, we can start with the inequality (58). Afterwards, the definition of θ_ϵ^θ can simplify this expression:

$$\begin{aligned}
0 &\le \mathbb{E}\left[R(\tau_n)\frac{1}{\epsilon}\int_{s_0}^{s_f} \lambda_{\tau_n}^T(\tilde{s})f_\theta\bigl(z_{\tau_n}(\tilde{s}),\hat{\theta}(\tilde{s})\bigr)\bigl(\theta_\epsilon^\theta(\tilde{s})-\hat{\theta}(\tilde{s})\bigr)\,d\tilde{s}\right] \\
&= \mathbb{E}\left[R(\tau_n)\frac{1}{\epsilon}\int_{s}^{s+\epsilon} \lambda_{\tau_n}^T(\tilde{s})f_\theta\bigl(z_{\tau_n}(\tilde{s}),\hat{\theta}(\tilde{s})\bigr)\bigl(\theta-\hat{\theta}(\tilde{s})\bigr)\,d\tilde{s}\right] \\
&= \mathbb{E}\Biggl[R(\tau_n)\underbrace{\frac{1}{\epsilon}\int_{s}^{s+\epsilon} \lambda_{\tau_n}^T(\tilde{s})f_\theta\bigl(z_{\tau_n}(\tilde{s}),\hat{\theta}(\tilde{s})\bigr)\,d\tilde{s}\,\theta}_{=\frac{h_1(s+\epsilon)-h_1(s)}{\epsilon}} \\
&\qquad - R(\tau_n)\underbrace{\frac{1}{\epsilon}\int_{s}^{s+\epsilon} \lambda_{\tau_n}^T(\tilde{s})f_\theta\bigl(z_{\tau_n}(\tilde{s}),\hat{\theta}(\tilde{s})\bigr)\hat{\theta}(\tilde{s})\,d\tilde{s}}_{=\frac{h_2(s+\epsilon)-h_2(s)}{\epsilon}}\Biggr] \\
&\xrightarrow{\epsilon\to 0} \mathbb{E}\bigl[R(\tau_n)\lambda_{\tau_n}^T(s)f_\theta\bigl(z_{\tau_n}(s),\hat{\theta}(s)\bigr)\bigl(\theta-\hat{\theta}(s)\bigr)\bigr] \\
&= \mathbb{E}\bigl[R(\tau_n)\lambda_{\tau_n}^T(s)f_\theta\bigl(z_{\tau_n}(s),\hat{\theta}(s)\bigr)\bigr]\bigl(\theta-\hat{\theta}(s)\bigr).
\end{aligned}$$

Simple rearrangements leave us with an expression, where we can find the difference quotients of the auxiliary functions. Since these are differentiable, the limits for ϵ going to zero exist. At this point, we stress that the limit for ϵ going to zero can be brought into the expected value. The argument for this is again provided by the dominated convergence theorem and the assumption of the Lebesgue integrability expression. In the last step, we extract $(\theta-\hat{\theta}(s))$ from the expected value. This is possible, since θ and $\hat{\theta}(s)$ are independent of the trajectories. □

In the case that $\hat{\theta}$ lies in the interior of $U \subset \mathbb{R}^{n_\theta}$, the inequality in (42) would become an equality. In the next step, we will use this new achieved result in order to derive a new algorithm.

4.3 Algorithm based on necessary optimality conditions and its challenges

The necessary optimality conditions can be used to solve the optimization problem. One can solve the optimality conditions and get potential candidates for an optimal solution. The conditions that need to be solved consist of three main parts. An initial value problem for an ordinary differential equation represented by the solution operator $\mathcal{F}$, its adjoint end value problem represented by $\mathcal{G}$, and the minimum principle. In order to solve such conditions, it turns out to be worth it, at least in the classical theory, to initialize all searched quantities and iterate trough these conditions. This means by starting with random initial parameters θ_0 we can solve the initial value problem first and the adjoint afterwards. In the end, these trajectories are used to determine the left-hand-side of the minimum principle, which turns out to be a good search direction, in which we update the parameters.

But we need to keep in mind that the expected value in our minimum principle cannot be computed exactly because it is an expected value over infinitely many trajectories, which is not practicable. This is because we would need to solve the initial value problem and end-value problem for infinitely many trajectories. Instead, we make use of the procedure many model-free Reinforcement Learning approaches are based on: Initialize the policy and continue generating finitely many trajectories with this policy. Afterwards, this data is used to estimate the update direction. In our case, this means that we initialize θ and generate trajectories by producing actions for observations we make and input these actions into the environment. Here, it is enough to be able to observe the environment and do not know the dynamical system at all. In this way, we solve an ODE (e.g. numerically) for each time we produce the action for a corresponding observation. Afterwards, we need to find the adjoint λ for each point in time by solving the end-value problem (adjoint ODE). Now, we have $z(s)$ and $\lambda(s)$ for each point in time and for finitely many trajectories, which are used to estimate the left-hand-side of the minimum principle for a good update direction. Afterwards, we start generating trajectories and start the whole procedure again. In this way, we can improve θ iteratively. The structure of this algorithm can be seen in the pseudo code in Algorithm 1. The capital $M \in \mathbb{N}$ is the batch size and indicates the number of trajectories used in one iteration in order to estimate the update direction. The above algorithm is for the sake of overview written down for a fixed step size α but can be equipped with suitable step size adaptations. The projection into U is denoted by $\mathcal{P}_U$. In the case of box constraints $U = \{\theta \in \mathbb{R}^{n_\theta} | \underline{\theta}_i \leq \theta_i \leq \bar{\theta}_i\}$, the projection takes the form $(\mathcal{P}_U(\theta))_i(s) = \max\{\min\{\underline{\theta}_i, \theta_i(s)\}, \bar{\theta}_i\}$ for bounds $\underline{\theta}_i, \bar{\theta}_i$ given component-wise.

```
The Algorithm:
• random initialized parameters $\theta^0, l = 0$;
while stopping criterion is not fulfilled do
    for $i = 1, \dots, M$ do
        • initialize $x_0^i$;
        for $k = 0, \dots, n$ do
            • solve the ODE $\frac{\partial z_k^i(s)}{\partial s} = f(z_k^i(s), \theta^l(s)), z_k^i(0) = x_k^i$;
            • simulate the model in order to get $x_{k+1}^i$;
        end
        for $k = 0, \dots, n$ do
            • solve the ODE $\frac{\partial \lambda_k^i(s)}{\partial s} = -(\lambda_k^i(s)^T f_z(z_k^i(s), \theta^l(s)))^T$, with
            $\lambda_k^i(s_f) = -\nabla_z(\log P(\tau_n | z_k^i(s_f)))^T$
        end
        ;
    end
    update parameters in the gradient direction with learning
    rate $\alpha$ and use a projection into $U$:
    • $\bar{\theta}^{l+1} = \theta_l - \alpha \sum_{k,i} R(\tau_n) \lambda_k^i(s)^T f_\theta(z_k^i(s), \theta^l(s))$;
    • $\theta^{l+1} = \mathcal{P}_U(\bar{\theta}^{l+1})$;
    • $l = l + 1$;
end
```

Algorithm 1: Algorithm based on NOC

5 Application

We will show the applicability of the newly developed algorithm in two test cases. The first one is a simple model of controlling to move a point in the two dimensional plane with obstacles. This example helps to recall all important parts of the introduced approach and how it is actually applied. As a second test case, we consider a human arm model with 29 muscle models and five degrees of freedom. This shows us the applicability to complex models with many control inputs compared to the degrees of freedom.

5.1 Moving point in the 2D plane

We start with the simple dynamical model. The goal is to move a point to a desired end position. The position of the point is described by a two dimensional vector x_k describing the position in two directions at time step k. The two dimensional vector u_k represents the velocity and the direction of the velocity. This quantity forms our control such that we aim to find a controller, which gives us the velocity for the current position in order to achieve our goal. Overall, we have:

$$\begin{aligned} &x_0 = [0.5, 1.5]^T, \\ &x_{k+1} = x_k + 0.05 \cdot u_k, \quad \text{for } k = 0, \dots, n. \end{aligned}$$

Since Reinforcement Learning approaches do not need the explicit equations of motion, we only need these equations in order to simulate forward in time to collect trajectories. We decided to use time steps of 0.05 min and define the final time to be 2.5 min (i.e. $n = 50$). The reward function gives back the negative distance between the current position and the target position $[4.5, -2.5]^T$. Additionally, the final position of one trajectory is rewarded by minus the square of ten times the distance and achieving the desired position leads to a positive reward of 10. Furthermore, we define two forbidden areas visualized by red rectangles in Fig. 2. Entering these areas will be penalized by a negative reward of 50. In Fig. 2 and 3, the starting position is marked with a yellow triangle and the desired end position is located at the blue rectangle.

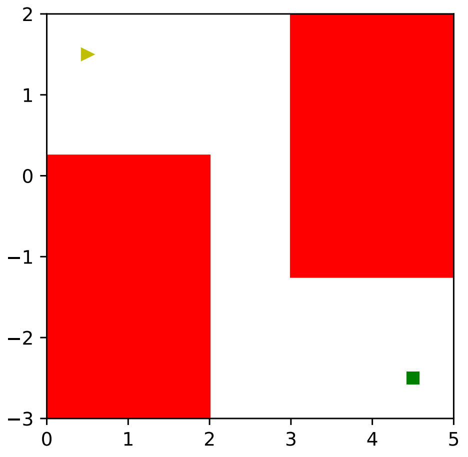

Figure 2 Task of the moving point problem

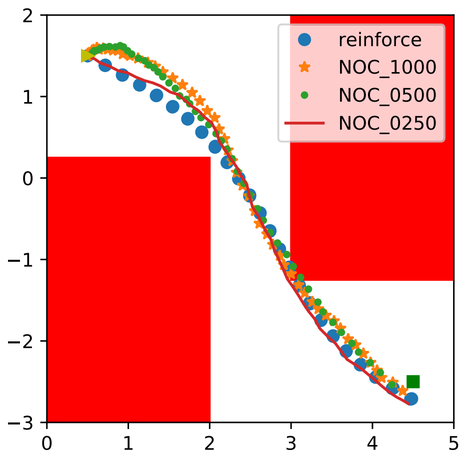

Figure 3 Trajectories in the obstacle course

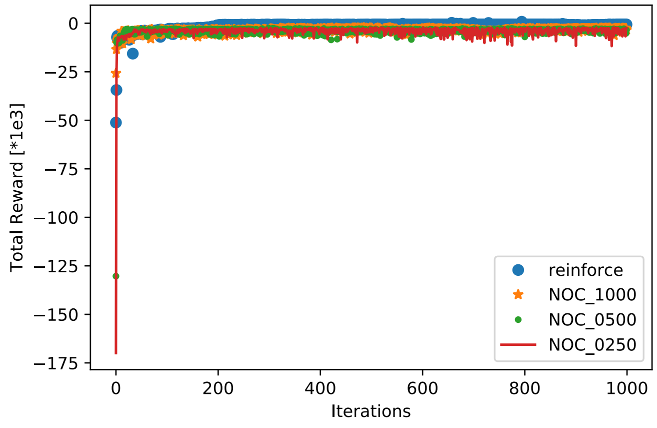

Figure 4 The summed up reward during the training

At this point, the environment and the reward function are fully introduced. The policy consists of an ODE replacing the neural network in Fig. 1 and a Gaussian distribution with a mean value determined by the final state of the ODE. We consider the ODE on the interval $[0, 10]$. Overall, we initialize θ randomly and simulate trajectories with the introduced discrete motion equations and the policy based on the ODE depending on θ. We generate ten trajectories in each iteration. Afterwards for each forward ODE, we solve the corresponding adjoint ODE. In the end, we use the collected data in order to compute the update direction. Then, we update the parameters in this direction with step size 10^{-2}. After doing this training, we have a controller, which is able to robustly control the point trough the obstacle course. We applied the algorithm with different methods to solve the ODE in the policy numerically. We decide to use an explicit Euler method with step size 1 (NOC_1000), 0.5 (NOC_0500) and 0.25 (NOC_0250). Trajectories, produced with the trained controller, can be observed in Fig. 3. Additionally, we run the same example with REINFORCE [14] for comparison. Most of the time, the trajectories lie on top of each other or at least close to each other. At this point, we stress that along these trajectories, the controls are randomly disturbed. This emphasizes the robustness of the controller. The reward during the training can be seen in Fig. 4.

5.2 Human arm model

At this point, we focus on a more complex model. The human arm model, which the authors of [21] have introduced and which is discussed in [22], consists of 29 Hill's muscle models and five degrees of freedom. Three of five degrees of freedom are located in the shoulder, where a spherical joint is modeled. The other two are in the elbow defined by two revolute joint. A visualization of the arm can be seen in Fig. 5(a). The gray cylinders represent the upper, forearm and the hand. The red, violet and blue lines portray the muscles. Here, the color stands for the activation of the muscle. If a muscle is not activated (activation = 0), the muscle is blue. If it is red, it is fully activated (activation = 1). On the finger tip, a small ball is shown. This is the marker, which is used to determine the distance from the finger tip to its desired end position. Keep in mind that the dimension of the control and state space are not the same anymore. This is solved by using an encoder implemented with neural networks. In [23], the authors discussed RL techniques as possibility to control this biomechanical model. In the present contribution, we apply the newly introduced technique instead of a trust region policy optimization (see [5]) as used in [23]. The goal of our optimization is to reach a desired final position with the finger tip. In Figs. 5(b)–5(e), one can see snapshots of the movement of the arm. Obviously, the finger tip touches the final end position in the end. This shows us that the task of reaching a certain point is fulfilled. Furthermore, the movement looks human-like and is plausible from this point of view. As in the previous example, we have a look at the summed up reward during the training in Fig. 6. Now, we run the training several times (black lines). In the end, we calculate the mean and plot it with a red line. We can see in each run, that the value of the objective function, which we maximize, increases iteratively up to a saturation limit.

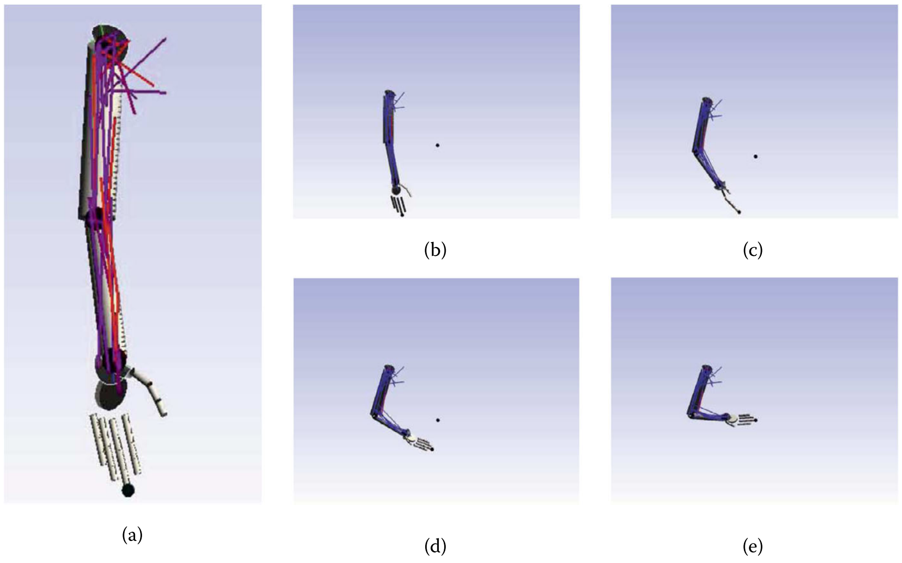

Figure 5 (**a**) Human arm model. (**b**) Trajectory after 1 time steps. (**c**) Trajectory after 5 time steps. (**d**) Trajectory after 8 time steps. (**e**) Trajectory after 15 time steps

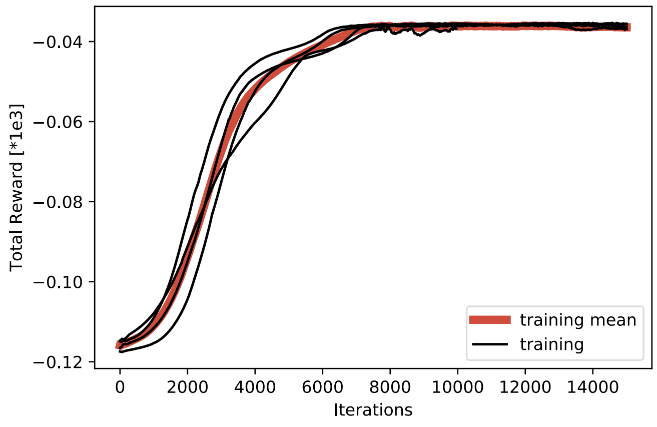

Figure 6 The summed up reward during the training

6 Conclusion

After introducing the basic setting of a policy-based Reinforcement Learning algorithm based on neural networks and recapitulating the connection between neural networks and ODEs, we successfully derived an infinite optimization problem consisting of an objective function and constraints given by ordinary differential equations. We pointed out that the main difference to the well-known optimal control problems is the endless number of ODE constraints. Nevertheless, we were able to prove the necessary optimality conditions of this optimization problem. These conditions were then used to define a new algorithm, which leads to a candidate for the optimal solution. In a simple example, we made clear how to apply the algorithm and we showed that it provides very well results. Afterwards, the algorithm was successfully applied to a more complex system. The human arm model is an overactuated system since it has more controls than degrees of freedom, and thus, is hard to control. Even though the algorithm leads to a policy which is able to control the finger tip to a desired end position.

In future works, we aim at benchmarking the quality of our algorithm against well-established approaches of Reinforcement Learning and against approaches for Optimal Control problems. Furthermore, as an example, the necessary depth of a neural network in a Reinforcement Learning approach might be derived. Defining the depth of a neural network a priori is often only based on the experience of the user. Now, our approach offers the possibility to use adaptive step size techniques for the corresponding ODE and uses the above introduced method directly or one defines the depth of the neural network based on that information. Overall, we are convinced that the presented results from above contribute to a more profound knowledge about Deep Reinforcement Learning approaches.

Funding
The authors are grateful for the funding by the Federal Ministry of Education and Research of Germany (BMBF), project number 05M16UKD.

Abbreviations
DDAE, Delay Differential-Algebraic Equations; MDP, Markov Decision Process; NOC, Necessary Optimality Conditions; ODE, Ordinary Differential Equations; RL, Reinforcement Learning.

Availability of data and materials
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
All authors contributed to all parts of the manuscript. All authors read and approved the final manuscript.

Author details
[1]Fraunhofer ITWM, Kaiserslautern, Germany. [2]Universität der Bundeswehr, München, Germany.

Publisher's Note

Received: 3 November 2019 Accepted: 24 March 2020 Published online: 07 April 2020

References
1. Pontryagin LS, Boltyanskii VG, Gamkrelidze RV, Mishchenko EF. The mathematical theory of optimal processes. New York: Wiley; 1962.
2. Sussmann HJ, Willems JC. 300 years of optimal control: from the brachystochrone to the maximum principle. IEEE Control Syst. Mag.. 1997;17:32–44.
3. Sutton RS, Barto AG. Reinforcement learning: an introduction. 2nd ed. Cambridge: MIT Press; 2018.
4. Bertsekas D. Reinforcement learning and optimal control. Athena Scientific; 2019.
5. Schulman J, Levine S, Abbeel P, Jordan MI, Moritz P. Trust region policy optimization. In: International conference on machine learning, JMLR. vol. 37. 2015. p. 1889–97.
6. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. CoRR 2017. arXiv:1707.06347.
7. Haber E, Ruthotto L. Stable architectures for deep neural networks. In: Inverse problems. vol. 34. Bristol: IOP Publishing; 2017.
8. Benning M, Celledoni E, Ehrhardt MJ, Owren B, Schönlieb C. Deep learning as optimal control problems: models and numerical methods. CoRR 2019. arXiv:1904.05657.
9. Ben-Tal A, El Ghaoui L, Nemirovski AS. Robust optimization. Princeton series in applied mathematics. Princeton: Princeton University Press; 2009.
10. Feinberg EA, Shwartz A. Handbook of Markov decision processes: methods and applications. Berlin: Springer; 2002.
11. Neveu J. Mathematical foundations of the calculus of probability. Oakland: Holden-Day; 1965.
12. Yang Y, Caluwaerts K, Iscen A, Zhang T, Tan J, Sindhwani V. Data efficient reinforcement learning for legged robots. CoRR 2019. arXiv:1907.03613.
13. Watkins CJCH, Dayan P. Q-learning. Mach. Learn. 1992;8:279–92.
14. Williams RJ. Simple statistical gradient-following algorithms for connectionist reinforcement. Mach. Learn.. 1992;8:229–56.
15. Buyya R, Calheiros RN, Dastjerdi AV. Big data: principles and paradigms. Amsterdam: Elsevier; 2016.
16. Grüne L, Junge O. Gewöhnliche Differentialgleichungen. 2nd ed. Wiesbaden: Springer; 2016.
17. Gerdts M. Optimal control of ODEs and DAEs. Berlin: De Gruyter; 2012.
18. Gerdts M. Optimal control of ordinary differential equations and differential algebraic equation. Habilitation, University of Bayreuth, Dr. Hut Verlag; 2007.
19. Machielsen KCP. Numerical solution of optimal control problems with state constraints by sequential quadratic programming in function space. Centrum voor Wiskunde en Informatica. 1988.
20. Burger M. Optimal control of dynamical systems: calculating input data for multibody system simulation. Dissertation, Technical. University of Kaiserslautern, Dr. Hut Verlag; 2011.
21. Roller M, Björkenstam S, Linn J, Leyendecker S. Optimal control of a biomechanical multibody model for the dynamic simulation of working tasks. In: Proceedings of the 8th ECCOMAS thematic conference on multibody dynamics. Prague. 2017. p. 817–26.
22. Obentheuer M, Roller M, Björkenstam S, Berns K, Linn J. Comparison of different actuation modes of a biomechanical human arm model in an optimal control framework. In: Proceedings of the 5th joint international conference on multibody system dynamics. Lisbon. 2018.
23. Burger M, Gottschalk S, Roller M. Reinforcement learning applied to a human arm model. In: Proceedings of the 9th ECCOMAS thematic conference on multibody dynamics. Duisburg. 2019. p. 68–75.