Deep Q-Learning for Deterministic Control Problems

Mamadou Waly Dia MANGA (mamadou.w.d.manga@aims-senegal.org)
African Institute for Mathematical Sciences (AIMS), Senegal

Supervised by: Moustapha Pemy
Towson University, USA

June 27, 2022

Submitted in partial fulfilment of the requirements for the award of Master of Science in Mathematical Sciences at AIMS Senegal

DECLARATION

This work was carried out at AIMS Senegal in partial fulfilment of the requirements for a Master of Science Degree.

I hereby declare that except where due acknowledgement is made, this work has never been presented wholly or in part for the award of a degree at AIMS Senegal or any other University.

Student: Mamadou Waly Dia MANGA ....

Supervisor: Moustapha Pemy

Tutor: Mouhamed Sylla

ACKNOWLEDGEMENTS

Firstly, I give glory to God the Almighty, the contribution of my primary supervisor, Prof. Moustapha Pemy, my tutor, Mouhamed Sylla and my referees towards the success of this research essay are well appreciated.

DEDICATION

I dedicate this to my family and friends.

Abstract

In this work we develop a Deep Q network to learn and solve Hamilton-Jacobi-Bellman equations associated with or continuous-time deterministic optimal control problems with Lipschitz continuous controls. Applying the dynamic programming principle to continuous-time Q-functions results in a new class of Hamilton-Jacobi-Bellman (HJB) problems. Our approach employs data gathered in discrete time without discretizing or estimating the system dynamics, and is based on a novel semi-discrete version of the HJB equation. We define the circumstances in which the Q-function calculated by this approach converges to the ideal Q-function. Numerical examples were presented to illustrate these results through benchmark tasks and high-dimensional linear-quadratic problems, we empirically show how well our method performs.

Contents

List of Figures

1. Introduction

1.1 Background

The HJB approach is a strategy for studying optimal control issues that is based on a functional equation called the Dynamic Programming Principle, which was introduced in the 1950s by Richard Bellman. This functional equation holds under relatively weak hypotheses, and it is essentially an optimality condition, implying that some quantity remains constant along the optimal paths of the dynamic optimization issue at hand. Adaptive strategies for tackling optimal control problems are Reinforcement Learning (RL) techniques Munos (2000). The difficulty of learning to select behaviors in unknown, dynamic situations is addressed using reinforcement learning techniques. It is commonly accepted that reinforcement learning approaches must be paired with generalizing function approximation methods such as artificial neural networks in order to be useful in complicated domains. Thrun and Schwartz (1993) presented a theoretical description of the phenomena using Watkins' Q-Learning as an example, identifying conditions under which it might cause learning to fail. Also gave experimental data that support the theoretical findings using some of the most prominent function approximators . Kim and Yang (2020) presented Hamilton–Jacobi–Bellman (HJB) equations for Q-functions in Lipschitz continuous controls in continuous-time optimum control problems. The HJB equation's unique viscosity solution is proven to be the conventional Q-function used in reinforcement learning. The viscosity solution framework provides a necessary and sufficient criterion for optimality. They created a Q-learning method for continuous-time dynamical systems utilizing the HJB equation. For high-dimensional state and control spaces, a DQN-like method is also presented. Q-learning techniques for continuous-time deterministic optimum control problems with Lipschitz continuous controls have been suggested once more by Kim et al. (2021). Applying the dynamic programming principle to continuous-time Q-functions yields a new class of Hamilton–Jacobi–Bellman (HJB) equations. Our solution is based on a novel semi-discrete version of the HJB equation that is presented to develop a Q-learning algorithm that employs discrete temporal data without discretizing or estimating the system dynamics. We determine the conditions under which our algorithm's estimated Q-function converges to the optimal Q-function.

1.2 Statement of the Problem

Q-learning is a model-free reinforcement learning algorithm for learning the value of an action in a certain state, and it is one of the most common Reinforcement Learning (RL) methods. The main idea behind Q-learning is to use trajectory samples to combine dynamic programming with stochastic approximation to estimate the optimal state-action value function, often known as the Q-function. Inspired by robotic and autonomous systems, there has recently been a growing interest in and demand for adapting prior ideas to challenging physical control tasks. Many physical processes, on the other hand, evolve in real time, necessitating RL approaches that can handle continuous-time dynamical systems in a systematic manner. Ordinary differential equations are frequently used to explain these systems (ODEs). In discrete time, the Bellman equation for Q-functions can be easily defined by applying dynamic programming. The dynamic programming problem is represented as a Hamilton–Jacobi–Bellman (HJB) equation in continuous-time instances, which provides a sound theoretical basis. Furthermore, to our knowledge, HJB equations have not been addressed in earlier ways to admit Q-functions (or state-action value functions) as solutions, despite a few attempts to create Q-function variations for continuous-time dynamical systems. Researchers developed a new class of HJB equations by applying the dynamic programming principle to the continuous-time Q-function. It is demonstrated that the HJB equation admits a single viscosity solution that corresponds to the ideal Q-function. The aim of this work is to develop a Deep Q network to learn and solve Hamilton-Jacobi-Bellman equations.

1.3 Definitions

Definition 1.3.1. Let $A \subset \mathbf{R}^n$ be a nonempty open set.

• Let a function $f : A \to \mathbf{R}$ have at each point of the set A all partial derivatives continuous (i.e., function $\boldsymbol{x} \mapsto \frac{\partial f}{\partial x_j}(\boldsymbol{x})$ are continuous on A for each $j \in \{1, \ldots, n\}$). Then we say that f is of the class $\mathcal{C}^1$ on A. The set of all these functions is denoted by $\mathcal{C}^1(A)$.

• Let a function $f : A \to \mathbf{R}$ have at each point of the set A all second partial derivatives continuous (i.e., function $\boldsymbol{x} \mapsto \frac{\partial^2 f}{\partial x_j^2}(\boldsymbol{x})$ are continuous on A for each $j \in \{1, \ldots, n\}$). Then we say that f is of the class $\mathcal{C}^2$ on A. The set of all these functions is denoted by $\mathcal{C}^2(A)$.

Definition 1.3.2. • **Local Maximum**: a maximum within a restricted domain, especially a point on a function whose value is greater than the values of all other points near it.

• **Local Minimum**: a point on a graph (or its associated function) such that the points each side have a greater value even though another point exists with a smaller value.

Definition 1.3.3. Lipschitz functions are the smooth functions of metric spaces. A real-valued function f on a metric space X is said to be L-Lipschitz if there is a constant $L \geq 1$ such that

$$|f(x) - f(y)| \leqslant L|x - y|$$

for all x and y in X.

Definition 1.3.4. Let $A \subset \mathbb{R}^n$. Suppose $(f_i)_{i=0}^{\infty}$ is a sequence of functions $f_i : A \to \mathbb{R}^m$. We say that $f_i \to f$ uniformly if the following holds: given $\epsilon > 0$, there exists $N \in \mathbb{N}$ such that for all $i \geq N$ we have:

$$\|f_i(x) - f(x)\| \leq \epsilon, \quad \text{for all } x \in A.$$

Definition 1.3.5. L^p convergence Let $p \in [1, \infty)$. For $f : \Omega \to \overline{\mathbb{R}}$, we define the $L^p(\Omega, \mu)$ norm by

$$\|f\|_{L^p(\Omega,\mu)} = \left(\int_\Omega |f|^p d\mu \right)^{\frac{1}{p}} \leq \infty$$

For $p = \infty$, we define the $L^\infty(\Omega, \mu)$ norm by

$$\|f\|_{L^\infty(\Omega,\mu)} := \mu\text{-ess} \sup_{x \in \Omega} |f(x)| = \inf\{C : |f| \leq C\mu\text{-a.e. }\}.$$

For $1 \leq p \leq \infty$, we say that a sequence of μ-measurable functions $(f_n)_{n \in \mathbb{N}}$ converges in $L^p(\Omega, \mu)$ to a measurable function f if

$$\lim_{n \to \infty} \|f_n - f\|_{L^p(\Omega,\mu)} = 0.$$

2. Preliminaries on Control Systems and Optimization problems

2.1 Deterministic control problems

We begin by considering a parametrized dynamical system dwelling on $\mathbb{R}^n$:

$$\begin{cases} \dot{x}(s) = f(s, x(s), u(s)), \quad \text{for a.e. } s \in (t, T) \\ u(s) \in U, \quad \text{for all } s \in (t, T) \\ x(t) = \boldsymbol{x}. \end{cases} \tag{2.1.1}$$

The elements that define a control system are the following: $T \in \mathbb{R} \cup \{+\infty\}$ is the final horizon, $t \in (-\infty, T)$ is the initial time, $\boldsymbol{x} \in \mathbb{R}^n$ is the initial position, $u : \mathbb{R} \to U$ is the control function with values in the control space U and $f : \mathbb{R} \times \mathbb{R}^n \times U \to \mathbb{R}^n$ is the dynamics mapping. In this chapter we assume that

$$U \subseteq \mathbb{R}^m \text{ is compact and nonempty.} \tag{2.1.2}$$

For sake of simplicity, we introduce the set

$$\mathscr{U}(a, b) = \{u : (a, b) \to U \text{ measurable }\}.$$

Unless otherwise stated, all along this chapter we are assuming that

$$\begin{cases} (i) & f : \mathbb{R} \times \mathbb{R}^n \times U \to \mathbb{R}^n \text{ is continuous.} \\ (ii) & \exists L_f > 0, \forall x, y \in \mathbb{R}^n, \forall t, s \in \mathbb{R}, \forall u \in U \\ & |f(t, x, u) - f(s, y, u)| \leq L_f(|t - s| + |x - y|). \end{cases} \tag{2.1.3}$$

Under mild hypotheses 2.1.3, given a measurable control function $u \in \mathscr{U}(t, +\infty)$, the control system 2.1.1 admits a unique solution which is an absolutely continuous arc defined on $(t, +\infty)$. To emphasize the dependence upon control and initial data, we reserve the notation $x^u_{t,\boldsymbol{x}}(\cdot)$ for such trajectory, which is called the state of the control system. In the case of autonomous systems, we assume $t = 0$ and denote the trajectory by $x^u_{\boldsymbol{x}}(\cdot)$.

An optimal control problem with fixed final horizon is an optimization problem that aims at minimizing the following functional

$$\mathscr{U}(t, T) \ni u \mapsto \int_t^T e^{-\lambda s} \ell\left(s, x^u_{t,\boldsymbol{x}}(s), u(s)\right) ds + e^{-\lambda T} \psi\left(x^u_{t,\boldsymbol{x}}(T)\right), \tag{2.1.4}$$

where $\lambda \geq 0$ is the discount factor, $\ell(\cdot)$ is the running cost and $\psi(\cdot)$ is the final cost. In this chapter the running cost is supposed to satisfy:

$$\begin{cases} (i) & \ell : \mathbb{R} \times \mathbb{R}^n \times U \to \mathbb{R} \text{ is continuous.} \\ (\text{ ii }) & \forall R > 0, \exists L^R_f > 0, \forall x, y \in \mathbb{B}_R, \forall t, s \in (-R, R), \forall u \in U \\ & |\ell(t, x, u) - \ell(s, y, u)| \leq L^R_\ell(|t - s| + |x - y|) \\ (iii) & \exists c_\ell > 0, \lambda_\ell \geq 1, \forall (x, u) \in \mathbb{R}^n \times U : \\ & 0 \leq \ell(x, u) \leq c_\ell \left(1 + |x|^{\lambda_\ell}\right) \end{cases} \tag{2.1.5}$$

The problem takes different name according to the data. In particular, we are interested in one particular problem: the Infinite Horizon problem $(T = +\infty$ and $\psi \equiv 0)$.

The value function is the mapping that associates any initial time t and initial position $\boldsymbol{x}$ with the optimal value of the problem 2.1.4. In the case of the infinite horizon we are only considering the autonomous case. Hence the value function

$$v(x) = \inf_{u\in\mathscr{U}(0,+\infty)} \left\{ \int_0^{\infty} e^{-\lambda s} \ell\left(x^u_{t,\boldsymbol{x}}(s), u(s)\right) ds \right\}. \tag{2.1.6}$$

On the other hand, for the Bolza problem the value function is

$$v(t,x) = \inf_{u\in\mathscr{U}(t,T)} \left\{ \int_t^{T} \ell\left(s, x^u_{t,\boldsymbol{x}}(s), u(s)\right) ds + \psi\left(x^u_{t,\boldsymbol{x}}(T)\right) \right\} \tag{2.1.7}$$

and in addition it satisfies the final condition:

$$v(T,\boldsymbol{x}) = \psi(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \mathbb{R}^n \tag{2.1.8}$$

Furthermore, in the formulation of 2.1.4 we may also consider that the final horizon is not fixed, which leads to a more general class of optimal control processes. Among these, the most relevant for the exposition is the so-called Minimum time problem to reach a given target $\Theta \subseteq \mathbb{R}^n$. In this case we write the value function as $T^{\Theta}(\cdot)$ and name it the minimum time function, which is given by

$$T^{\Theta}(\boldsymbol{x}) = \inf_{u\in\mathscr{U}(0,+\infty)} \left\{ T \geq 0 \mid x^u_{\boldsymbol{x}}(T) \in \Theta \right\}. \tag{2.1.9}$$

This function satisfies by definition the condition $T^{\Theta}(\boldsymbol{x}) = 0$ at any $\boldsymbol{x} \in \Theta$. We remark that this case can be viewed as an Infinite Horizon problem with $\lambda = 0$ and $\ell(y,u) \equiv 1$.

2.2 Dynamic Programming Principle and HJB equations

The HJB approach for optimal control problems is based in a functional equation known as the Dynamic Programming Principle. This equation has different forms based on the issue at hand:

•**Infinite Horizon problem**: for any $\boldsymbol{x} \in \mathbb{R}^n$ and $\tau \in (0,+\infty)$

$$v(\boldsymbol{x}) = \inf_{u\in\mathscr{U}(0,\tau)} \left\{ \int_0^{\tau} e^{-\lambda s} \ell\left(x^u_{\boldsymbol{x}}(s), u(s)\right) ds + e^{-\lambda\tau} v\left(x^u_{\boldsymbol{x}}(\tau)\right) \right\}. \tag{2.2.1}$$

• **Bolza problem**: for any $x \in \mathbb{R}^n$ and $\tau \in (t,T)$

$$v(t,\boldsymbol{x}) = \inf_{u\in\mathscr{U}(t,\tau)} \left\{ \int_t^{\tau} \ell\left(s, x^u_{t,\boldsymbol{x}}(s), u(s)\right) ds + v\left(\tau, x^u_{t,\boldsymbol{x}}(\tau)\right) \right\} \tag{2.2.2}$$

• **Minimum time problem**: for any $\boldsymbol{x} \in \mathbb{R}^n$ and $\tau \in \left(0, T^{\Theta}(\boldsymbol{x})\right)$

$$T^{\Theta}(\boldsymbol{x}) = \inf_{u\in\mathscr{U}(0,\tau)} \left\{ \tau + T^{\Theta}\left(x^u_{\boldsymbol{x}}(\tau)\right) \right\} \tag{2.2.3}$$

Let us assume that the value functions are continuously differentiable functions. Then, some standard calculations yield to the following HJB equation:

- **Infinite Horizon problem**:

$$\lambda v(\boldsymbol{x}) + H(\boldsymbol{x}, \nabla v(\boldsymbol{x})) = 0, \quad \boldsymbol{x} \in \mathbb{R}^n, \tag{2.2.4}$$

with $H(\boldsymbol{x}, p) := \sup\{-\langle f(\boldsymbol{x}, u), p\rangle - \ell(\boldsymbol{x}, u) \mid u \in U\}$.

- **Bolza problem**:

$$-\partial_t v(t, \boldsymbol{x}) + H\left(t, \boldsymbol{x}, \nabla_{\boldsymbol{x}} v(t, \boldsymbol{x})\right) = 0, \quad (t, \boldsymbol{x}) \in (-\infty, T) \times \mathbb{R}^n$$

with $H(t, \boldsymbol{x}, p) := \sup\{-\langle f(t, \boldsymbol{x}, u), p\rangle - \ell(t, \boldsymbol{x}, u) \mid u \in U\}$.

- **Minimum time problem**:

$$-1 + H\left(\boldsymbol{x}, \nabla T^{\Theta}(\boldsymbol{x})\right) = 0, \quad \boldsymbol{x} \in \operatorname{int}\left(\operatorname{dom} T^{\Theta}\right),$$

with $H(\boldsymbol{x}, p) := \sup\{-\langle f(\boldsymbol{x}, u), p\rangle \mid u \in U\}$.

However, because the value function is rarely differentiable, the Hamilton-Jacobi-Bellman equations must be understood in a hazy way. The Viscosity Solutions Theory, developed by Crandall and Lions in their landmark publication in 1983 Crandall and Lions (1983), is the most appropriate framework for dealing with these equations.

Let us now introduce the notion of viscosity solution of the HJ equation

$$F(\boldsymbol{x}, v(\boldsymbol{x}), \nabla v(\boldsymbol{x})) = 0, \quad \boldsymbol{x} \in \Omega \tag{2.2.5}$$

where Ω is an open domain of $\mathbb{R}^n$ and the Hamiltonian $F = F(\boldsymbol{x}, r, p)$ is a continuous, real valued function on $\Omega \times \mathbb{R} \times \mathbb{R}^n$.

2.3 Viscosity solutions

Crandall and Lions (1983) established the viscosity solution notion in mathematics in the early 1980s as a generalization of the classical concept of what is meant by a'solution' to a partial differential equation (PDE). The viscosity solution has been found to be the natural solution concept to use in many applications of PDEs, such as first-order equations arising in dynamic programming (the Hamilton–Jacobi–Bellman equation), differential games (the Hamilton–Jacobi–Isaacs equation), or front evolution problems, as well as second-order equations arising in stochastic optimal control or stochastic differential games.

Definition 2.3.1. A continuous function $v : \Omega \to \mathbb{R}$ is a viscosity solution of the equation 2.2.5 if the following conditions are satisfied:

- for any test function $\phi \in C^1(\Omega)$, if $x_0 \in \Omega$ is a local maximum point for $v - \phi$, then

$$F\left(x_0, v\left(x_0\right), \nabla \phi\left(x_0\right)\right) \leq 0 \quad \text{(viscosity subsolution)}$$

- for any test function $\phi \in C^1(\Omega)$, if $x_0 \in \Omega$ is a local minimum point for $v - \phi$, then

$$F\left(x_0, v\left(x_0\right), \nabla \phi\left(x_0\right)\right) \geq 0 \quad \text{(viscosity supersolution)}$$

The motivation for the terminology viscosity solutions is that this kind of solution can be recovered as the limit function $v = \lim_{\varepsilon \to 0^+} v^\varepsilon$ where $v^\varepsilon \in C^2(\Omega)$ is the classical solution of the parabolic problem

$$-\varepsilon \Delta v^\varepsilon + F\left(x, v^\varepsilon, \nabla v^\varepsilon\right) = 0, \quad x \in \Omega \tag{2.3.1}$$

Under some technical assumptions v^ε exists and converges locally uniformly to the viscosity solution v. This method is named vanishing viscosity, and it is the original idea behind this notion of solution proposed by Crandall and Lions (1983).

We present some comparison results between viscosity sub- and supersolutions. As simple corollary, each comparison result produces a uniqueness theorem for the associated Dirichlet problem. In the following of the section we assume F of the form $F(x, r, q) = ar + H(x, q)$ where the positive constant a (possibly zero) will be specified in each case.

Theorem 2.3.1. *Let Ω be a bounded open subset of $\mathbb{R}^n$. Assume that $v_1, v_2 \in C(\bar{\Omega})$ are, respectively, viscosity sub- and supersolution of*

$$v(x) + H(x, \nabla v(x)) = 0, \quad x \in \Omega \tag{2.3.2}$$

and

$$v_1 \leq v_2 \quad \text{on } \partial\Omega \tag{2.3.3}$$

Assume also that H satisfies

$$|H(x,p) - H(y,p)| \leq \omega_1(|x-y|(1+|p|)), \tag{2.3.4}$$

for $x, y \in \Omega, p \in \mathbb{R}^n$, where ω_1 is a modulus, that is $\omega_1 : [0, +\infty) \to [0, +\infty)$ is continuous non decreasing with $\omega_1(0) = 0$. Then $v_1 \leq v_2$ in $\bar{\Omega}$.

The proof of the previous Theorem 2.3.1 can be found in Bardi et al. (1997) Chapter II, Theorem 3.1.

Theorem 2.3.2. *Assume that $v_1, v_2 \in C\left(\mathbb{R}^n\right) \cap L^\infty\left(\mathbb{R}^n\right)$ are, respectively, viscosity suband supersolution of*

$$v(x) + H(x, \nabla v(x)) = 0, \quad x \in \mathbb{R}^n \tag{2.3.5}$$

Assume also that H satisfies (178) and

$$|H(x,p) - H(x,q)| \leq \omega_2(|p-q|), \quad \text{for all } x, p, q \in \mathbb{R}^n. \tag{2.3.6}$$

where ω_2 is a modulus. Then $v_1 \leq v_2$ in $\mathbb{R}^n$.

Remark. Theorem 2.3.2 (for the proof we refer to Bardi et al. (1997) Chapter II, Theorem 3.5.) can be generalized to cover the case of a general unbounded open set $\Omega \subset \mathbb{R}^n$.

Moreover, the assumptions $v_1, v_2 \in C\left(\mathbb{R}^n\right) \cap L^\infty\left(\mathbb{R}^n\right)$ can be replaced by $v_1, v_2 \in UC\left(\mathbb{R}^n\right)$ (uniformly continuous). A comparison result for the more general case

$$H(x, \nabla v(x)) = 0, \quad x \in \Omega \tag{2.3.7}$$

can be stated if we assume the convexity of H with respect to the p variable. This assumption plays a key role in many theoretical results.

Theorem 2.3.3. *Let Ω be a bounded open subset of $\mathbb{R}^n$. Assume that $v_1, v_2 \in C(\bar{\Omega})$ are, respectively, viscosity sub- and supersolution of 2.3.7 with $v_1 \leq v_2$ on $\partial\Omega$. Assume also that H satisfies 2.3.4 and the two following conditions*

$\star$ *$p \to H(x,p)$ is convex on $\mathbb{R}^n$ for each $x \in \Omega$,*

$\star$ *there exists $\phi \in C(\bar{\Omega}) \cap C^1(\Omega)$ such that $\phi \leq v_1$ in $\bar{\Omega}$ and $\sup_{x\in B} H(x, D\phi(x)) < 0$ for all $B \subset \Omega$. Then $v_1 \leq v_2$ in Ω.*

The proof of this result can be found in Bardi et al. (1997) Chapter II, Theorem 3.9.

The crucial point for viscosity solution is to prove uniqueness. This is done via a comparison principle (also called maximum principle). The comparison principle is enough to get uniqueness.

Remark. If $H(,,)$ is uniformly continuous and H satisfies the equation 2.3.4, then the comparison principle holds for (HJ), i.e. the viscosity solution is unique.

In fact, The goal is to prove that $M = \max_{x\in\bar{\Omega}}(u - v)$ is negative. We introduce a test function depending on two variables

$$\psi_\varepsilon(x,y) \equiv u(x) - v(y) - \frac{|x-y|^2}{\varepsilon^2}.$$

Due to the penalization term, we can expect that the maximum points $(x_\varepsilon, y_\varepsilon)$ for ψ_ε should have x_ε and y_ε close enough for ε small. Moreover, for $\varepsilon \to 0^+$we have:

$$M_\varepsilon \to M, \quad \frac{|x_\varepsilon - y_\varepsilon|^2}{\varepsilon^2} \to 0 \text{ and } u\left(x_\varepsilon\right) - v\left(y_\varepsilon\right) \to M$$

Those properties allow to pass to the limit and get the comparison result.

3. Hamilton–Jacobi–Bellman Equations for Q-Functions

Consider a continuous-time dynamical system of the form

$$\begin{cases} \dot{x}(t) = f(x(t), a(t)), & \text{for } t > 0 \\ a(s) \in U, & \text{for all } t > 0 \\ x(0) = \boldsymbol{x}. \end{cases} \tag{3.0.1}$$

Where $x(t) \in \mathbb{R}^n$ and $a(t) : \mathbb{R} \to U$ are the system state and the control action, respectively. $U \subseteq \mathbb{R}^m$ is compact and nonempty. . Here, the vector field $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is an unknown function.

The standard infinite-horizon discounted optimal control problem that aims at minimizing the following functional

$$J(\boldsymbol{x}, a) = \int_0^\infty e^{-\gamma t} r(x(t), a(t)) \mathrm{d}t \tag{3.0.2}$$

In other words, we have

$$\sup_{a \in \mathcal{A}} J(\boldsymbol{x}, a) = J(\boldsymbol{x}, \bar{a}) \tag{3.0.3}$$

with $x(0) = \boldsymbol{x}$, where $r : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is an unknown reward function of interest and $\gamma > 0$ is a discount factor.

The (continuous-time) Q-function of (3.0.3 is defined as

$$Q(\boldsymbol{x}, \boldsymbol{a}) := \sup_{a \in \mathcal{A}} \left\{ \int_0^\infty e^{-\gamma t} r(x(t), a(t)) \mathrm{d}t \mid x(0) = \boldsymbol{x}, a(0) = \boldsymbol{a} \right\}, \tag{3.0.4}$$

which represents the maximal reward incurred from time 0 when starting from $x(0) = \boldsymbol{x}$ with $a(0) = \boldsymbol{a}$.

Proposition 1 Suppose that $\mathcal{A} := \{a : \mathbb{R}_{\geq 0} \to \mathbb{R}^m \mid \text{a is measurable } \}$. Then, the optimal Q-function 3.0.4 corresponds to the optimal value function v for each $\boldsymbol{a} \in \mathbb{R}^m$, that is, $Q(\boldsymbol{x}, \boldsymbol{a}) = v(\boldsymbol{x})$ for all $(\boldsymbol{x}, \boldsymbol{a}) \in \mathbb{R}^n \times \mathbb{R}^m$.

Where, the corresponding standard optimal value function is express as:

$$v(\boldsymbol{x}) := \sup_{a \in \mathcal{A}} \left\{ \int_0^\infty e^{-\gamma t} r(x(t), a(t)) \mathrm{d}t \mid x(0) = \boldsymbol{x} \right\} \text{for all } \boldsymbol{a} \in \mathbb{R}^m \tag{3.0.5}$$

Proof: Fix $(\boldsymbol{x}, \boldsymbol{a}) \in \mathbb{R}^n \times \mathbb{R}^m$.

Let ε be an arbitrary positive constant.

Then, there exists $a \in \mathcal{A}$ such that $\int_t^\infty e^{-\gamma(s-t)} r(x(s), a(s)) \mathrm{d}s < v(\boldsymbol{x}) + \varepsilon$, where $x(s)$ satisfies 3.0.1 with $x(t) = \boldsymbol{x}$ in the Carathéodory sense: $x(s) = \boldsymbol{x} + \int_t^s f(x(\tau), a(\tau)) \mathrm{d}\tau$.

We now construct a new control $\tilde{a} \in \mathcal{A}$ as $\tilde{a}(s) := \boldsymbol{a}$ if $s = t; \tilde{a}(s) := a(s)$ if $s > t$.

Such a modification of controls at a single point does not affect the trajectory or the total reward. Therefore, we have

$$v(\boldsymbol{x}) \leq Q(\boldsymbol{x}, \boldsymbol{a}) \leq \int_t^\infty e^{-\gamma(s-t)} r(x(s), \tilde{a}(s)) \mathrm{d}s < v(\boldsymbol{x}) + \varepsilon$$

Since ε was arbitrarily chosen, we conclude that

$v(\boldsymbol{x}) = Q(\boldsymbol{x}, \boldsymbol{a})$ for any $(\boldsymbol{x}, \boldsymbol{a}) \in \mathbb{R}^n \times \mathbb{R}^m$. ■

Now using the fact that any Lipschitz continuous function is differentiable almost everywhere, we choose the set of admissible controls as:

$$\mathcal{A} := \{ a : \mathbb{R}_{\geq 0} \to \mathbb{R}^m \mid a \text{ is measurable, } |\dot{a}(t)| \leq L \text{ a.e. } \}, \tag{3.0.6}$$

where $|\cdot|$ denotes the standard Euclidean norm, and L is a fixed constant. From now on, we will focus on the optimal control problem 3.0.3 with Lipschitz continuous controls, and the corresponding Q-function 3.0.4.

3.1 Dynamic Programming and HJB Equations

Dynamic programming is a method of solving the credit-assignment problem in multi-stage decision processes. The scope of dynamic programming is often misrepresented in the computer science literature-the true variety of its applications is perhaps best explained in Dreyfus and Bellman (1962) or Dreyfus and Averill (1977). The basic principle of dynamic programming is to solve the credit assignment problem by constructing an evaluation function, also known as a value function or a return function, on the state-space. Dynamic programming provides an alternative approach to designing optimal controls, assuming we can solve a nonlinear partial differential equation, called the Hamilton-Jacobi-Bellman equation.

By the dynamic programming principle, we have

$$\begin{aligned} Q(\boldsymbol{x}, \boldsymbol{a}) = \sup_{a \in \mathcal{A}} \Bigg\{ & \int_t^{t+h} e^{-\gamma(s-t)} r(x(s), a(s)) \mathrm{d}s \\ & + e^{-\gamma h} \int_{t+h}^\infty e^{-\gamma(s-(t+h))} r(x(s), a(s)) \mathrm{d}s \mid x(t) = \boldsymbol{x}, a(t) = \boldsymbol{a} \Bigg\} \\ = \sup_{a \in \mathcal{A}} \Bigg\{ & \int_t^{t+h} e^{-\gamma(s-t)} r(x(s), a(s)) \mathrm{d}s + e^{-\gamma h} Q(x(t+h), a(t+h)) \mid x(t) = \boldsymbol{x}, a(t) = \boldsymbol{a} \Bigg\} \end{aligned}$$

for any $h > 0$. Rearranging this equality, adding and removing the term $\frac{1}{h} Q(x(t+h), a(t+h))$, we obtain

$$\sup_{a\in\mathcal{A}}\left\{\frac{1}{h}\int_t^{t+h} e^{-\gamma(s-t)}r(x(s),a(s))\mathrm{d}s+\frac{1}{h}[Q(x(t+h),a(t+h))-Q(\boldsymbol{x},\boldsymbol{a})]\right.$$
$$\left.+\frac{e^{-\gamma h}-1}{h}Q(x(t+h),a(t+h)) \mid x(t)=\boldsymbol{x},a(t)=\boldsymbol{a}\right\}=0.$$

Letting h tend to zero and assuming for a moment that the Q-function is continuously differentiable, its Taylor expansion yields

$$\gamma Q(\boldsymbol{x},\boldsymbol{a})-\sup_{\boldsymbol{b}\in\mathbb{R}^m,|\boldsymbol{b}|\leq L}\{\nabla_{\boldsymbol{x}}Q\cdot f(\boldsymbol{x},\boldsymbol{a})+\nabla_{\boldsymbol{a}}Q\cdot\boldsymbol{b}+r(\boldsymbol{x},\boldsymbol{a})\}=0 \tag{3.1.1}$$

where the optimization variable $\boldsymbol{b}$ represents $\dot{a}(t)$, and the constraint $|\boldsymbol{b}|\leq L$ is due to the Lipschitz constraint on control trajectories, $|\dot{a}(t)|\leq L$ a.e.

The supremum in 3.1.1 is attained at $\boldsymbol{b}^\star=L\frac{\nabla_a Q}{|\nabla_a Q|}$, Since the terms $\nabla_{\boldsymbol{x}}Q\cdot f(\boldsymbol{x},\boldsymbol{a})$ and $r(\boldsymbol{x},\boldsymbol{a})$ are independent of $\boldsymbol{b}$.

We have, by replacing $\boldsymbol{b}^\star$ in equation 3.1.1

$$\gamma Q(\boldsymbol{x},\boldsymbol{a})-\nabla_{\boldsymbol{x}}Q\cdot f(\boldsymbol{x},\boldsymbol{a})-L\left|\frac{(\nabla_a Q)^2}{\nabla_a Q}\right|-r(\boldsymbol{x},\boldsymbol{a})=0,$$

Hence, we have the HJB equation for the Q-function,

$$\gamma Q(\boldsymbol{x},\boldsymbol{a})-\nabla_{\boldsymbol{x}}Q\cdot f(\boldsymbol{x},\boldsymbol{a})-L\,|\nabla_{\boldsymbol{a}}Q|-r(\boldsymbol{x},\boldsymbol{a})=0, \tag{3.1.2}$$

3.2 Viscosity Solution: Existence and Uniqueness

In general, the Q-function is not a C^1 function. As a weak solution of the HJB equation, we use the framework of Viscosity Solution. Considering the definition 2.3.1 with $\Omega=\mathbb{R}^k$. Then the function $u\in C\left(\mathbb{R}^k\right)$ is a viscosity solution of 2.2.5 if It satisfies the two conditions of the definition 2.3.1.

Note that the viscosity solution does not need to be differentiable. In our case, the HJB equation 3.1.2

$$\gamma Q(\boldsymbol{x},\boldsymbol{a})-\nabla_{\boldsymbol{x}}Q(\boldsymbol{x},\boldsymbol{a})\cdot f(\boldsymbol{x},\boldsymbol{a})-L\,|\nabla_{\boldsymbol{a}}Q(\boldsymbol{x},\boldsymbol{a})|-r(\boldsymbol{x},\boldsymbol{a})=0$$

can be expressed as 2.2.5 with

$$F(\boldsymbol{z},q,\boldsymbol{p})=\gamma q-\boldsymbol{p}_1\cdot f(\boldsymbol{z})-L\,|\boldsymbol{p}_2|-r(\boldsymbol{z})$$

where $\boldsymbol{z}=(\boldsymbol{x},\boldsymbol{a})\in\mathbb{R}^n\times\mathbb{R}^m$ and $\boldsymbol{p}=(\boldsymbol{p}_1,\boldsymbol{p}_2)\in\mathbb{R}^n\times\mathbb{R}^m$.

Assumption 3.2.1. From now on, we assume the following regularity conditions on f and r:

- $(A1)$ The functions f and r are bounded: $\|f\|_{L^\infty}+\|r\|_{L^\infty}<C$.

- $(A2)$ The functions f and r are Lipschitz continuous: $\|f\|_{\text{Lip}}+\|r\|_{\text{Lip}}<C$,

where $\|\cdot\|_{\text{Lip}}$ is a Lipschitz constant of argument.

Then, the HJB equation 3.1.2 has a unique viscosity solution, which corresponds to the Q-function.

Theorem 3.2.1. *Suppose that the Assumption 3.2.1 hold. Then, the optimal continuous-time Q-function is the unique viscosity solution to the HJB equation 3.1.2.*

Proof First, recall that our control trajectory satisfies the constraint $|\dot{a}| \leq L$.

Therefore, our dynamical system can be written in the following extended form:

$$\dot{x}(t) = f(x(t), a(t)), \quad \dot{a}(t) = b(t), \quad t > 0, \quad |b(t)| \leq L,$$

by viewing $x(t)$ and $a(t)$ as state variables. More precisely, the dynamics of the extended state variable $z(t) = (x(t), a(t))$ can be written as

$$\dot{z}(t) = G(z(t), b(t)), \quad t > 0, \quad |b(t)| \leq L, \tag{3.2.1}$$

where $G(\boldsymbol{z}, \boldsymbol{b}) = (f(\boldsymbol{z}), \boldsymbol{b})$. Applying the dynamic programming principle to the Q-function, we have

$$Q(\boldsymbol{z}) = \sup_{|b(s)| \leq L} \left\{ \int_t^{t+h} e^{-\gamma(s-t)} r(z(s)) \mathrm{d}s + e^{-\gamma h} Q(z(t+h)) \mid z(t) = \boldsymbol{z} \right\}$$

The remaining proof is almost the same as the proof of Proposition 2.8, Chapter 3 in Bardi et al. (1997). However, for the self-completeness of the thesis, we provide a detailed proof. In the following, we show that the Q-function satisfies the two conditions in Definition 2.3.1

First, let $\phi \in C^1(\mathbb{R}^{n+m})$ such that $Q - \phi$ attains a local maximum at $\boldsymbol{z}$.

Then, there exists $\delta > 0$ such that $Q(\boldsymbol{z}) - Q(\boldsymbol{z}') \geq \phi(\boldsymbol{z}) - \phi(\boldsymbol{z}')$ for $|\boldsymbol{z}' - \boldsymbol{z}| < \delta$.

Since f and r are bounded Lipschitz continuous, there exists $h_0 > 0$, which is independent of $b(s)$, such that $|z(s) - \boldsymbol{z}| \leq \delta, |r(z(s)) - r(\boldsymbol{z})| \leq C(s-t)$ and $|f(z(s)) - f(\boldsymbol{z})| \leq C(s-t)$ for $t \leq s \leq t + h_0$, where $z(s)$ is a solution to 3.2.1 for $s \geq t$ with $z(t) = \boldsymbol{z}$.

Now, the dynamic programming principle for the Q-function implies that, for any $0 < h < h_0$ and $\varepsilon > 0$, there exists $b(s)$ with $|b(s)| \leq L$ such that

$$Q(\boldsymbol{z}) \leq \int_t^{t+h} e^{-\gamma(s-t)} r(z(s)) \mathrm{d}s + e^{-\gamma h} Q(z(t+h)) + h\varepsilon$$

where $z(s)$ is now a solution to 3.2.1 with $z(t) = z$ under the particular choice of b.

On the other hand, it follows from our choice of h that

$$\int_t^{t+h} e^{-\gamma(s-t)} r(z(s)) \mathrm{d}s = \int_t^{t+h} e^{-\gamma(s-t)} r(\boldsymbol{z}) \mathrm{d}s + o(h)$$

which implies that

$$Q(\boldsymbol{z}) \leq \int_t^{t+h} e^{-\gamma(s-t)} r(\boldsymbol{z}) \mathrm{d}s + e^{-\gamma h} Q(z(t+h)) + h\varepsilon + o(h)$$

Therefore, we have

$$\begin{aligned}\phi(\boldsymbol{z})-\phi(z(t+h)) &\leq Q(\boldsymbol{z})-Q(z(t+h))\\ &\leq \int_t^{t+h} e^{-\gamma(s-t)} r(\boldsymbol{z}) \mathrm{d}s+\left(e^{-\gamma h}-1\right) Q(z(t+h))+h\varepsilon+o(h)\end{aligned}$$

Since the left-hand side of the inequality above is equal to

$$-\int_t^{t+h} \frac{\mathrm{d}}{\mathrm{d}s} \phi(z(s)) \mathrm{d}s=-\int_t^{t+h} \nabla_z \phi(z(s)). G(z(s), b(s)) \mathrm{d}s$$

,

where, $\nabla_z \phi(z(s))=\left(\nabla_{\boldsymbol{x}} \phi(z(s)), \nabla_{\boldsymbol{a}} \phi(z(s))\right)$, and we obtain that

$$\begin{aligned}0 \leq & \int_t^{t+h} \nabla_{\boldsymbol{z}} \phi(z(s)) \cdot G(z(s), b(s)) \mathrm{d}s\\ & +\int_t^{t+h} e^{-\gamma(s-t)} r(\boldsymbol{z}) \mathrm{d}s+\left(e^{-\gamma h}-1\right) Q(z(t+h))+h\varepsilon+o(h)\\ \leq & \int_t^{t+h}\left(\nabla_{\boldsymbol{x}} \phi(z(s)) \cdot f(\boldsymbol{z})+L\left|\nabla_{\boldsymbol{a}} \phi(z(s))\right|\right) \mathrm{d}s\\ & +\int_t^{t+h} e^{-\gamma(s-t)} r(\boldsymbol{z}) \mathrm{d}s+\left(e^{-\gamma h}-1\right) Q(z(t+h))+h\varepsilon+o(h)\end{aligned}$$

By dividing both sides by h and letting $h \rightarrow 0$, we conclude that

$$\nabla_{\boldsymbol{x}} \phi(\boldsymbol{z}) \cdot f(\boldsymbol{z})+L\left|\nabla_{\boldsymbol{a}} \phi(\boldsymbol{z})\right|+r(\boldsymbol{z})-\gamma Q(\boldsymbol{z})+\varepsilon \geq 0$$

Since ε was arbitrarily chosen, we confirm that the Q-function satisfies the first condition in Definition 2.3.1, that is,

$$\gamma Q(\boldsymbol{z})-\nabla_{\boldsymbol{x}} \phi(\boldsymbol{z}) \cdot f(\boldsymbol{z})-L\left|\nabla_{\boldsymbol{a}} \phi(\boldsymbol{z})\right|-r(\boldsymbol{z}) \leq 0$$

We now consider the second condition.

Let $\phi \in C^1\left(\mathbb{R}^{n+m}\right)$ such that $Q-\phi$ attains a local minimum at $\boldsymbol{z}$, that is, there exists δ such that $Q(\boldsymbol{z})-Q\left(\boldsymbol{z}^{\prime}\right) \leq \phi(\boldsymbol{z})-\phi\left(\boldsymbol{z}^{\prime}\right)$ for $\left|\boldsymbol{z}^{\prime}-\boldsymbol{z}\right|<\delta$.

Fix an arbitrary $\boldsymbol{b} \in \mathbb{R}^m$ such that $|\boldsymbol{b}| \leq L$ and let $b(s) \equiv \boldsymbol{b}$ be a constant function.

Let $z(s)$ be a solution to 3.2.1 for $s \geq t$ with $z(t)=\boldsymbol{z}$ under the particular choice of $b(s) \equiv \boldsymbol{b}$.

Then, for sufficiently small $h, |z(t+h)-\boldsymbol{z}| \leq \delta$, and therefore we have

$$\begin{aligned}Q(\boldsymbol{z})-Q(z(t+h)) \leq \phi(\boldsymbol{z})-\phi(z(t+h)) &=-\int_t^{t+h} \frac{\mathrm{d}}{\mathrm{d}s} \phi(z(s)) \mathrm{d}s\\ &=-\int_t^{t+h} \nabla_{\boldsymbol{z}} \phi(z(s)) \cdot G(z(s), \boldsymbol{b}) \mathrm{d}s\end{aligned} \tag{3.2.2}$$

On the other hand, the dynamic programming principle yields

$$Q(\boldsymbol{z}) - Q(z(t+h)) \geq \int_t^{t+h} e^{-\gamma(s-t)} r(z(s))\mathrm{d}s + \left(e^{-\gamma h} - 1\right) Q(z(t+h)) \tag{3.2.3}$$

By 3.2.2 and 3.2.3, we have

$$\left(e^{-\gamma h} - 1\right) Q(z(t+h)) + \int_t^{t+h} e^{-\gamma(s-t)} r(z(s))\mathrm{d}s \leq -\int_t^{t+h} \nabla_{\boldsymbol{z}}\phi(z(s)) \cdot G(z(s), \boldsymbol{b})\mathrm{d}s$$

Dividing both sides by h and letting $h \to 0$, we obtain that

$$-\gamma Q(\boldsymbol{z}) + r(\boldsymbol{z}) \leq -\nabla_{\boldsymbol{z}}\phi(\boldsymbol{z}) \cdot (f(\boldsymbol{z}), \boldsymbol{b})$$

or equivalently

$$\gamma Q(\boldsymbol{z}) - \nabla_{\boldsymbol{x}}\phi(\boldsymbol{z}) \cdot f(\boldsymbol{z}) - \nabla_{\boldsymbol{a}}\phi(\boldsymbol{z}) \cdot \boldsymbol{b} - r(\boldsymbol{z}) \geq 0$$

Since $\boldsymbol{b}$ was arbitrarily chosen from $\{\boldsymbol{b} \in \mathbb{R}^m : |\boldsymbol{b}| \leq L\}$, we have

$$\gamma Q(\boldsymbol{z}) - \nabla_{\boldsymbol{x}}\phi(\boldsymbol{z}) \cdot f(\boldsymbol{z}) - L\left|\nabla_{\boldsymbol{a}}\phi(\boldsymbol{z})\right| - r(\boldsymbol{z}) \geq 0$$

which confirms that the Q-function satisfies the second condition in Definition 2.3.1. Therefore, we conclude that the Q-function is a viscosity solution of the HJB equation 3.1.2.

Lastly, the uniqueness of the viscosity solution can be proved by using Theorem 2.12, Chapter 3 in Bardi et al. (1997).

4. Hamilton–Jacobi DQN and Reinforcement Learning

In recent years, there has been a surge in the use of machine learning, and many companies are now exploring how it can be used to utilize their data in new ways. This popularity can partly be attributed to the increase in computational power, but is also likely an effect of the availability of large amounts of data.

Reinforcement learning is a machine learning training method based on rewarding desired behaviors and/or punishing undesired ones. In general, a reinforcement learning agent is able to perceive and interpret its environment, take actions and learn through trial and error. Reinforcement Learning is the science of decision making. It is about learning the optimal behavior in an environment to obtain maximum reward.

Q-learning is an off policy reinforcement learning algorithm that seeks to find the best action to take given the current state. It's considered off-policy because the q-learning function learns from actions that are outside the current policy, like taking random actions, and therefore a policy isn't needed. In other hand, Q-learning is one of the most well known reinforcement learning methods that seek efficient control policies without the knowledge of an explicit system model. The critical though of Q-learning is to combine dynamic programming and stochastic approximation in a way to estimate the optimal state-action value function, also called the Q-function, by using trajectory samples Kim and Yang (2020).

4.1 Some Terminology

The terminology used in this thesis is standard in Dynamic Programming and optimal control, and in an effort to forestall confusion of readers that are accustomed to either the artificial intelligence or the optimal control terminology, we provide a list of terms commonly used in RL, and their optimal control counterparts.

(a) **Environment** = System.

(b) **Agent** = Decision maker or controller.

(c) **Action** = Decision or control.

(d) **Reward of a stage** = (Opposite of) Cost of a stage.

(e) **State value** = (Opposite of) Cost starting from a state.

(f) **Value (or reward) function** = (Opposite of) Cost function.

(g) **Maximizing the value function** = Minimizing the cost function.

(h) **Action (or state-action) value** = Q-factor (or Q-value) of a statecontrol pair. (Q-value is also used often in RL.)

(i) **Planning** = Solving a DP problem with a known mathematical model.

(j) **Learning** = Solving a DP problem without using an explicit mathematical model. (This is the principal meaning of the term "learni ng" in RL. Other meanings are also common.)

(k) **Self-learning (or self-play in the context of games)** = Solving a DP problem using some form of policy iteration.

(l) **Deep reinforcement learning** = Approximate DP using value and/or policy approximation with deep neural networks.

(m) **Prediction** = Policy evaluation.

(n) **Generalized policy iteration** = Optimistic policy iteration.

(o) **State abstraction** = State aggregation.

(p) **Temporal abstraction** = Time aggregation.

(q) **Learning a model** = System identification.

(r) **Episodic task or episode** = Finite-step system trajectory.

(s) **Continuing task** = Infinite-step system trajectory.

(t) **Experience replay** = Reuse of samples in a simulation process.

(u) **Bellman operator** = DP mapping or operator.

(v) **Backup** = Applying the DP operator at some state.

(w) **Sweep** = Applying the DP operator at all states.

(x) **Greedy policy with respect to a cost function** J = Minimizing policy in the DP expression defined by J.

(y) **Afterstate** = Post-decision state.

(z) **Ground truth** = Empirical evidence or information provided by direct observation.

4.2 Hamilton–Jacobi Q-Learning and DQN

4.2.1 Hamilton–Jacobi Q-Learning

Q-learning is an off policy reinforcement learning algorithm that seeks to find the best action to take given the current state. It's considered off-policy because the Q-learning function learns from actions that are outside the current policy, like taking random actions, and therefore a policy isn't needed.

To design a concrete algorithm for learning the Q-function using such discrete-time data in practical problems, we propose a novel semi-discrete version of the HJB equation 3.1.2 without discretizing or approximating the continuous-time system.

Let $h > 0$ be a fixed sampling interval, and let $\mathcal{B} := \{b := \{b_k\}_{k=0}^{\infty} | b_k \in \mathbb{R}^m, | b_k | \leq L\}$, where b_k is analogous to $\dot{a}(t)$ in the continuous-time case. Given $(\boldsymbol{x}, \boldsymbol{a}) \in \mathbb{R}^n \times \mathbb{R}^m$ and a sequence $b \in \mathcal{B}$, we let

$$Q^{h,b}(\boldsymbol{x}, \boldsymbol{a}) := \sum_{k=0}^{\infty} r\left(x_k, a_k\right)(1-\gamma h)^k$$

where $\{(x_k, a_k)\}_{k=0}^{\infty}$ is defined by $x_{k+1} = \xi\left(x_k, a_k; h\right)$ and $a_{k+1} = a_k + hb_k$ with $(x_0, a_0) = (\boldsymbol{x}, \boldsymbol{a})$. Here, $\xi\left(x_k, a_k; h\right)$ denotes the state of 3.0.1 at time $t = h$ with initial state $x(0) = x_k$ and constant action $a(t) \equiv a_k, t \in [0, h)$.

It is worth emphasizing that our semi-discrete approximation does not approximate the system dynamics and thus is more accurate than the standard semi-discrete method (Section VI, (Bardi et al. (1997))).

The optimal semi-discrete Q-function $Q^{h,\star} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is then defined as

$$Q^{h,\star}(\boldsymbol{x}, \boldsymbol{a}) := \sup_{b \in \mathcal{B}} Q^{h,b}(\boldsymbol{x}, \boldsymbol{a}) \tag{4.2.1}$$

Then, $Q^{h,\star}$ satisfies a semi-discrete version of the HJB equation 3.1.2.

Proposition 4.2.1. Suppose that $0 < h < \frac{1}{\gamma}$. Then, the function $Q^{h,\star}$ is a solution to the following semi-discrete HJB equation:

$$Q^{h,\star}(\boldsymbol{x}, \boldsymbol{a}) = hr(\boldsymbol{x}, \boldsymbol{a}) + (1-\gamma h) \sup_{|\boldsymbol{b}| \leq L} Q^{h,\star}(\xi(\boldsymbol{x}, \boldsymbol{a}; h), \boldsymbol{a} + h\boldsymbol{b}) \tag{4.2.2}$$

Moreover, the optimal semi-discrete Q-function converges uniformly to its original counterpart in every compact subset of $\mathbb{R}^n \times \mathbb{R}^m$.

Proposition 4.2.2. Suppose that $0 < h < \frac{1}{\gamma}$ and 3.2.1 holds. Then, the function $Q^{h,\star}$ is the unique solution to the semi-discrete HJB equation 4.2.2. Furthermore, we have

$$\lim_{h \to 0} \sup_{(\boldsymbol{x}, \boldsymbol{a}) \in K, K \text{ compact}} \left| Q^{h,\star}(\boldsymbol{x}, \boldsymbol{a}) - Q(\boldsymbol{x}, \boldsymbol{a}) \right| = 0.$$

This proposition justifies the use of the semi-discrete HJB equation for small h. We aim to estimate the optimal Q-function using sample data collected in discrete time, enjoying the benefits of both the semi-discrete HJB equation 4.2.2 and the original HJB equation 3.1.2.

Convergence Properties :

Let's consider the following model-free update of Q-functions using the semi-discrete HJB equation : In the k^{th} iteration, for each $(\boldsymbol{x}, \boldsymbol{a})$ we collect data $(x_k := \boldsymbol{x}, a_k := \boldsymbol{a}, r_k, x_{k+1})$ and update the Q-function, with learning rate α_k, by

$$Q_{k+1}^h(\boldsymbol{x}, \boldsymbol{a}) := (1-\alpha_k) Q_k^h(\boldsymbol{x}, \boldsymbol{a}) + \alpha_k \left[hr_k + (1-\gamma h) \sup_{|\boldsymbol{b}| \leq L} Q_k^h\left(x_{k+1}, \boldsymbol{a} + h\boldsymbol{b}\right) \right], \tag{4.2.3}$$

where x_{k+1} is obtained by running (or simulating) the continuous-time system from x_k with action a_k fixed for h period without any approximation, that is, $x_{k+1} = \xi\left(x_k, a_k; h\right)$, and $r_k = r\left(x_k, a_k\right)$. We refer to this synchronous Q-learning as Hamilton-Jacobi Q-learning (HJ Q-learning).

Note that this method is not practically useful because the update must be performed for all state-action pairs in the continuous space. In the following section, we propose a DQN-like algorithm to approximately perform HJ Q-learning by employing deep neural networks as a function approximator.

Before doing so, we identify the conditions under which the Q-function updated by 4.2.3 converges to the optimal semi-discrete Q-function 4.2.1 in L^∞.

Theorem 4.2.1. *Suppose that* $0 < h < \frac{1}{\gamma}, 0 \leq \alpha_k \leq 1$ *and that Assumption 3.2.1 holds. If the sequence* $\{\alpha_k\}_{k=0}^{\infty}$ *of learning rates satisfies* $\sum_{k=0}^{\infty} \alpha_k = \infty$*, then*

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

Finally, by Propositions 4.2.3 and Theorem 4.2.1, we establish the following convergence result associating HJ Q-learning 4.2.3 with the optimal Q-function in the original continuous-time setting.

4.2.2 Hamilton–Jacobi DQN

In this section we are going to explore the Hamilton–Jacobi DQN by using some results in section 3 Kim et al. (2021) which talk about convergence and showing that the optimal Q-function can be estimated in a model-free manner via the semi-discrete HJB equation.

Deep Q-Learning means a Neural Network maps input states to (action, Q-value) pairs which is different from Q-Learning that means a table maps each state-action pair to its corresponding Q-value.

In a general setting, the DQN Algorithm can be expressed into three parts which are:

1. Initialize your Main and Target neural networks,
2. Choose an action using the Epsilon-Greedy Exploration Strategy,
3. Update your network weights using the Bellman Equation.

Furthermore, It has been said that, It is intractable to directly implement HJ Q-learning 4.2.3 over a continuous state-action space, then here we employ the deep neural networks.

The Hamilton-Jacobi DQN algorithm is then proposed, which approximates the update 4.2.3 without discretizing or approximating the continuous-time system.

We only consider a parameterized Q-function $Q_\theta(\boldsymbol{x}, \boldsymbol{a})$, where θ is the network's parameter vector, because our algorithm includes no actor.

We utilise a second target function Q_{θ^-}, similar to DQN, in which the network parameter vector θ^- is updated more slowly than θ. We also use experience replay by storing transition data (x_k, a_k, r_k, x_{k+1}) in a buffer with fixed capacity and by randomly sampling a mini-batch of transition data $\{(x_j, a_j, r_j, x_{j+1})\}$ to update the target value.

The target Q-function must be maximised across all admissible actions when establishing the target value in DQN, that is, $y_j^- := hr_j + \gamma' \max_a Q_{\theta^-}(x_{j+1}, \boldsymbol{a})$, where $\gamma' := 1 - \gamma h$ is the corresponding semi-discrete discount factor.

However, in the case of continuous action spaces, maximising the target Q-function with respect to the action variable is computationally difficult. To solve this problem, we return to the original HJB equation and apply Theorem 2 (of Kim et al. (2021)) related optimal action. We look at the action dynamics (equation 6 of Theorem 2) in Kim et al. (2021) in particular $b_j := L\frac{\nabla_a Q_{\theta^-}(x_j,a_j)}{|\nabla_a Q_{\theta^-}(x_j,a_j)|}$ fixed over sampling interval h to obtain

$$a_{j+1} = a_j + hb_j := a_j + hL\frac{\nabla_a Q_{\theta^-}(x_j, a_j)}{|\nabla_a Q_{\theta^-}(x_j, a_j)|}.$$

Using this optimal control action, we can approximate the maximal target Q-function value as

$$\max_{|\boldsymbol{a}-a_j|\leq hL} Q_{\theta^-}(x_{j+1}, \boldsymbol{a}) \approx Q_{\theta^-}(x_{j+1}, a_j + hb_j).$$

As h drops, this approximation gets more accurate. In particular, the approximation error has an $O(h^2)$ bound, as shown in the following proposition:

Proposition 4.2.3. Suppose that $Q_{\theta^-}$is twice continuously differentiable with bounded first and second derivatives. If $\nabla_a Q_{\theta^-}(x_j, a_j) \neq 0$, we have

$$\lim_{h\to 0}\left|\max_{|\boldsymbol{a}-a_j|\leq hL} Q_{\theta^-}(x_{j+1}, \boldsymbol{a}) - Q_{\theta^-}(x_{j+1}, a_j + hb_j)\right| = 0.$$

Moreover, the difference above is $O(h^2)$ as $h \to 0$.

The main benefit of employing the optimal action found in the continuous-time case is that it avoids having to solve the nonlinear optimization problem directly $\max_{|\boldsymbol{a}-a_j|\leq hL} Q_{\theta^-}(x_{j+1}, \boldsymbol{a})$, which is a computationally intensive task. With this choice of target Q-function value and the semi-discrete HJB equation , we set the target value as $y_j^- := hr_j + (1-\gamma h)Q_{\theta^-}(x_{j+1}, a_j + hb_j\ )$. We can use double Q-learning (Van Hasselt et al. (2016)) to reduce overestimation of Q-functions by simply changing the parameters b_j as $b_j := L\frac{\nabla_a Q_\theta(x_j,a_j)}{|\nabla_a Q_\theta(x_j,a_j)|}$ to use a greedy action with respect to Q_θ instead of Q_{θ^-}. In this double Q-learning version, Proposition 4.2.3 remains valid except for the $O(h^2)$ convergence rate.

The network parameter θ can then be trained to minimize the loss function $\sum_j \left(y_j^- - Q_\theta(x_j, a_j)\right)^2$.

For exploration, we add the additional Gaussian noise $\varepsilon \sim N(0, \sigma^2 I_m)$ to generate the next action as $a_{k+1} := a_k + hL\frac{\nabla_a Q_\theta(x_k,a_k)}{|\nabla_a Q_\theta(x_k,a_k)|} + \varepsilon$.

Other exploration strategies, such as the solution of a stochastic differential equation, can be used instead (Tallec et al. (2019)). The overall algorithm is presented below in Algorithm 1

As in DQN, we used a replay buffer to address these issues. The replay buffer is a finite sized. Transitions were sampled from the environment according to the exploration policy and the tuple $\{(x_k, a_k, r_k, x_{k+1})\}$ was stored in the replay buffer. When the replay buffer was full the oldest samples were discarded. At each timestep the actor and critic are updated by sampling a minibatch uniformly from the buffer. Because we are using an off-policy algorithm, the replay buffer can be large, allowing the algorithm to benefit from learning across a set of uncorrelated transitions.

Algorithm 1 : Hamilton-Jacobi DQN

```
Initialize $Q-$function $Q_\theta$ with random weights $\theta$, and target $Q-$function $Q_{\theta^-}$ with
weights $\theta^- = \theta$;
Initialize replay buffer with fixed capacity;
for episode $= 0$ to $M$ do
    Randomly sample initial state-action pair $(x_0, a_0)$;
    for $k = 0$ to $K$ do
        Execute action $a_k$ and observe reward $r_k$ and the next state $x_{k+1}$;
        Store $(x_k, a_k, r_k, x_{k+1})$ in buffer;
        Sample the random mini-batch $\{(x_j, a_j, r_j, x_{j+1})\}$ from buffer;
        Set $y_{j^-} := h_{r_j} + (1 - \gamma h) Q_{\theta^-}(x_{j+1}, a'_j) \forall j$ where $a'_j := a_j + hL \frac{\nabla_a Q_\theta(x_j, a_j)}{|\nabla_a Q_\theta(x_j, a_j)|}$;
        Update $\theta$ by minimizing $\sum_j (y_{j^-} - Q_\theta(x_j, a_j))^2$;
        Update $\theta^- \leftarrow (1 - \alpha)\theta^- + \alpha\theta$ for $\alpha << 1$;
        Set the next action as $a_{k+1} := a_k + hL \frac{\nabla_a Q_\theta(x_k, a_k)}{|\nabla_a Q_\theta(x_k, a_k)|}$
    end for
end for
```

4.3 Experiments

4.3.1 Actor Networks vs. Optimal Control ODE

Deep Reinforcement Learning has recently gained a lot of traction in the machine learning community as a result of significant progress made in the last few years. Traditionally, reinforcement learning algorithms were restricted to tiny, discretized grid worlds, severely impeding their credibility as viable machine learning tools. Now with DQN, people realized that deep learning methods could be used to solve high-dimensional problems. Many researchers as Van Hasselt et al. (2016) have been working on building algorithms for tackling the continuous action space problem using a Model Free called Deep Deterministic Policy Gradients (DDPG) that is off-policy.

Accordingly to the literature Van Hasselt et al. (2016), while we are dealing with a continuous control problem, we chose deep deterministic policy gradient (DDPG) as a baseline for comparison. DDPG is an actor-critic technique that use separate actor networks, whereas our method is valued-based and does not employ a parameterized policy. However, we focus on comparison of HJ DQN and DDPG to examine whether the role of actor networks can be replaced by the optimal control characterized through our HJB equation. Fujimoto et al. (2018) with TD3 which is a concurrent work that proposes a deterministic algorithm that substantially improves on DDPG and Haarnoja et al. (2018) with SAC which avoids the complexity and potential instability associated with approximate inference in prior off-policy maximum entropy algorithms based on soft Q-learning. Those two methods are state-of-the-art methods built upon DDPG.

We used the same The hyperparameters used in the experiments of Kim et al. (2021). We look at continuous control benchmark tasks in the OpenAI gym Brockman et al. (2016), which are emulated by the MuJoCo engine Todorov et al. (2012). Considering the four Environments HalfCheater-v2, Hopper-v2, Swimmer-v2 and Walker2d-v2, Figure 4.2 demonstrates the learning curves for the two techniques, each of which was evaluated for 1 million steps using five distinct random seeds. When the default sample interval is employed, as seen in Figure 4.2, our technique performs similarly to DDPG. On Walker2d-v2, our approach performs better than DDPG, however HalfCheetah-v2 yields the opposite result. In addition to the default sampling interval, we additionally determine an ideal h for each task because the sampling interval h is a hyper-parameter of Algorithm 1. We adjust the learning rate α as described in when testing various sample intervals Tallec et al. (2019) (Specifically, the learning rate is multiplied by the same constant when the sample period is multiplied.). With the exception of HalfCheetah-v2, this additional tweaking procedure enhances final performances and learning speed. Overall, the findings suggest that the ODE characterization (see Theorem 2 Kim et al. (2021)) of optimal control produced using our HJB framework may be used in place of actor networks. In terms of hyper-parameter tweaking and computational load, our technique clearly outperforms DDPG without the use of actor networks.

Our results are similar to the results obtained in Kim et al. (2021) wherein it has shown the the behavior of the actions trajectories of both methods using only HalfCheater-v2 environment. When compared to DDPG, the action trajectories derived by HJ DQN oscillate less. This demonstrates that oscillations in action occur frequently in optimal control (see Figure 4.3). This

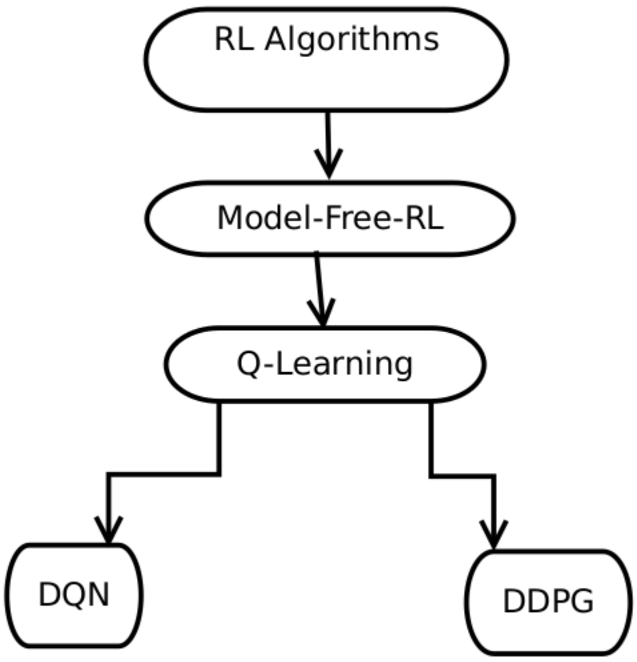


Figure 4.1: Reinforcement Learning Model-free

good performance of HJ DQN over DDPG in term of the actions, in the HalfCheater-v2 environment can be explained by the intervention of the Lipschitz constraint in HJ DQN, which acts as regularizer, preventing radical changes in motion.

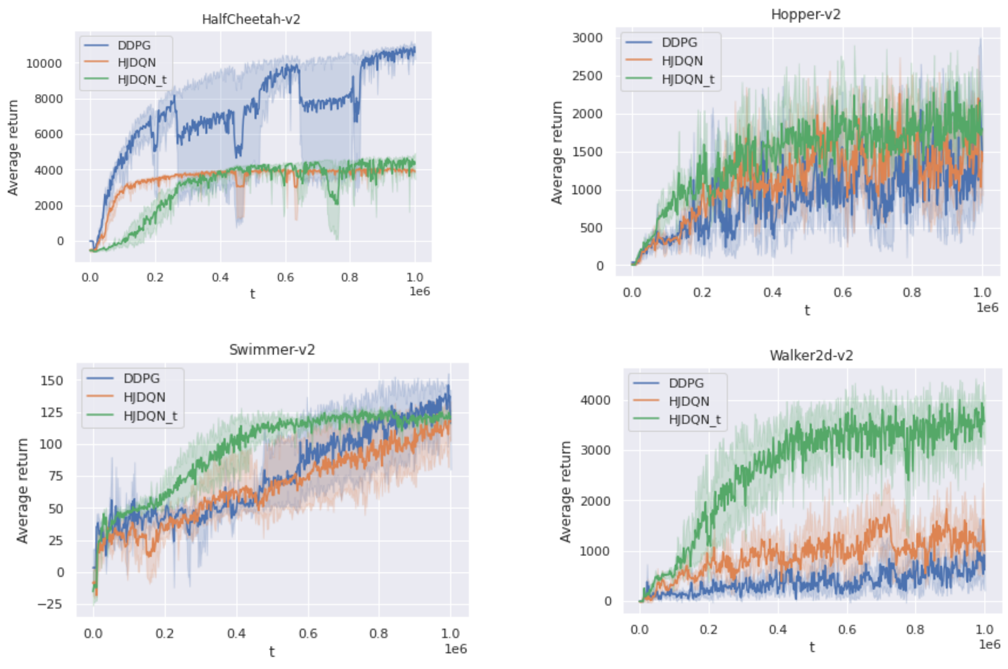
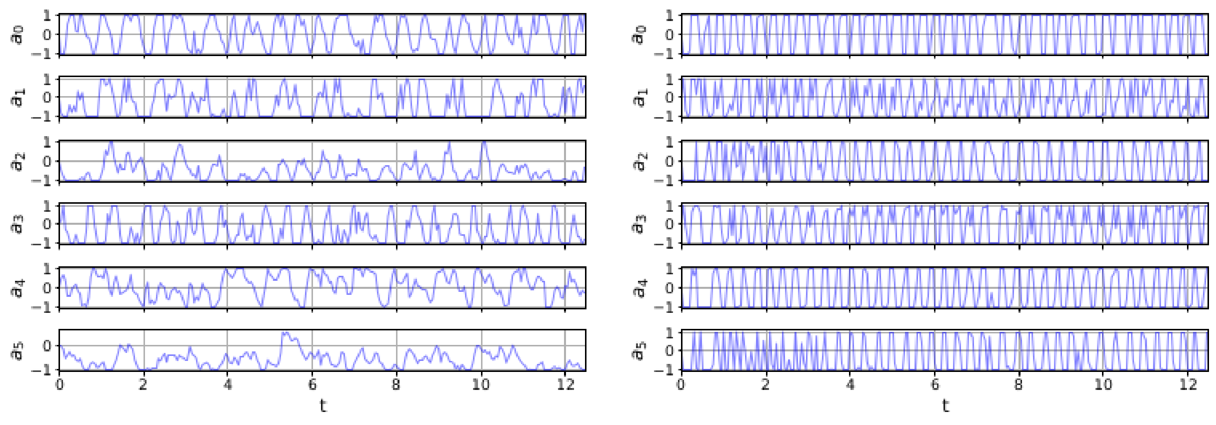


Figure 4.2: Learning curves for OpenAI gym continuous control tasks.



(a) HJ DQN (b) DDPG

Figure 4.3: Action trajectories of HalfCheetah-v2, obtained by HJ DQN and DDPG.

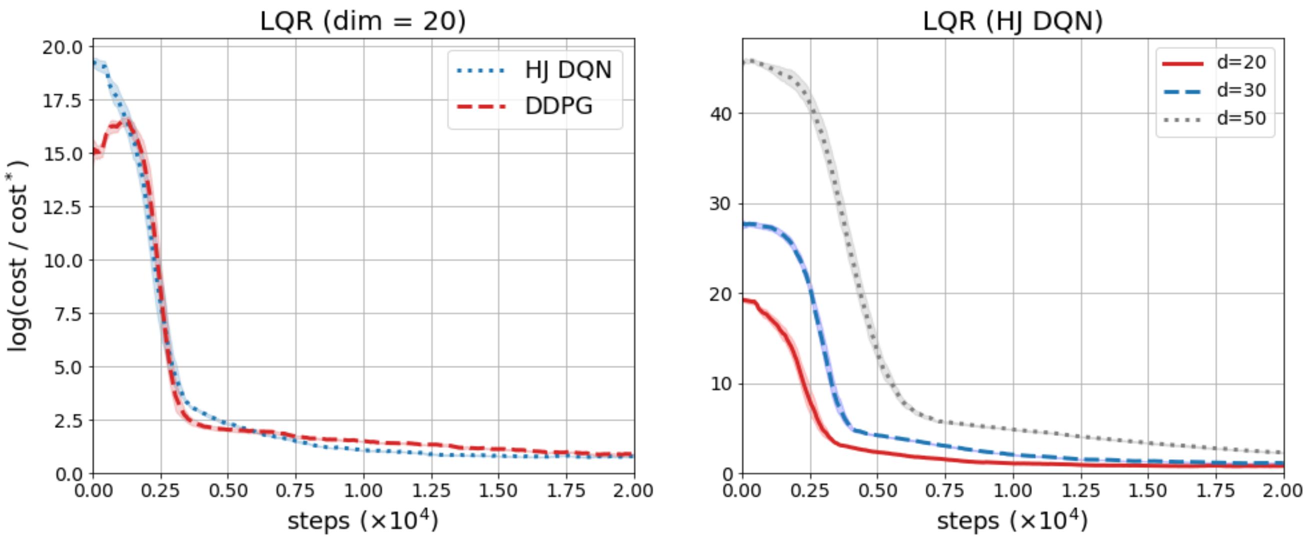


Figure 4.4: Learning curves for the LQ problem: (a) comparing HJ DQN and DDPG; (b) the effect of problem sizes.

4.3.2 Linear-Quadratic Problems

In this part we are going to investigate a special case of optimal control problems where the state equations are linear in both the state and control with nonhomogeneous terms, and the cost functionals are quadratic. Such a control problem is called a linear quadratic (LQ) optimal control problem (LQ problem, for short). The LQ problems constitute an extremely important class of optimal control problems, since they can model many problems in applications, and more importantly, many nonlinear control problems can be reasonably approximated by the LQ problems. On the other hand, solutions of LQ problems exhibit elegant properties due to their simple and nice structures.

Let's consider an Linear-Quadratic (LQ) problem with system dynamics

$$\dot{x}(t) = Ax(t) + Ba(t), t > 0, x(t), a(t) \in \mathbb{R}^d$$

and reward function (negative cost)

$$r(\boldsymbol{x}, \boldsymbol{a}) = -\left(\boldsymbol{x}^T Q \boldsymbol{x} + \boldsymbol{a}^T R \boldsymbol{a}\right)$$

where $Q = Q^T \geq 0$ and $R = R^T > 0$ (see, for example, (Anderson and Moore (2007)) for details about the theory of the classical LQ control). Due to the Lipchitz constraint on controls, it should be noted that our solution addresses a problem distinct from the traditional LQ problem. As a result, our method's control must be less than ideal.

Considering the same matrices, and setting the discount factor λ and Lipschitz constant L as Kim et al. (2021), we have the following results.

Figure 4.4 (a) shows the outcomes of our initial comparison of HJ DQN and DDPG performance for the situation of d = 20. Similar to Figure 4.2, the learning curves are plotted. Each graph's y-axis shows the log of the difference between the actual and ideal costs. As a result, the curve moves closer to the x-axis as performance increases. According to these findings, HJ DQN is able to successfully learn a useful (though suboptimal) strategy, whereas DDPG is completely unable to minimise the cost. The outcome suggests that, without the assistance of a separate actor network, the HJ DQN successfully learns a useful (suboptimal) policy that is comparable to the DDPG policy.

The learning curves for HJ DQN with various system sizes up to 50 dimensions are shown in Figure 4.4 (b). Although the scale of the challenge has an impact on learning rate, HJ DQN uses high-dimensional systems to tackle the LQ problem. Additionally, after around 10^4 steps, it is seen that the learning curves have essentially little fluctuation over trials and the standard deviations across trials are rather small. The stability of our technique is demonstrated by this outcome.

5. Conclusion

With the Lipschitz restriction on controls, we have introduced a fresh theoretical and computational framework that extends DQN to continuous-time deterministic optimum control over continuous action space. A Q-learning method for continuous-time control has been developed using a novel class of HJB equations for Q-functions. The theoretical convergence characteristics of this approach have been demonstrated. We have merged the HJB-based approach with DQN for practical implementation, resulting in a straightforward solution that resolves continuous-time control issues devoid of an actor network. This model-free off-policy approach uses our theoretical study of the HJB equations to identify greedy actions without the need for any numerical optimization. According to the findings of our studies, our optimal control, which can be readily described using an ODE, can replace actor networks in DDPG while requiring less computational work. Our HJB framework may provide an exciting avenue for future research in continuous-time RL in terms of optimal control problem under Markov switching model for solving optimal stock liquidation problems under regime switching.

References

Brian DO Anderson and John B Moore. *Optimal control: linear quadratic methods*. Courier Corporation, 2007.

Martino Bardi, Italo Capuzzo Dolcetta, et al. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*, volume 12. Springer, 1997.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Michael G Crandall and Pierre-Louis Lions. Viscosity solutions of hamilton-jacobi equations. *Transactions of the American mathematical society*, 277(1):1–42, 1983.

Stuart E Dreyfus and M Averill. Law. the art and theory of dynamic programming. *Mathematics in Science and Engineering*, 130, 1977.

Stuart E Dreyfus and Richard Bellman. *Applied dynamic programming*. Princeton University Press, 1962.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

Jeongho Kim and Insoon Yang. Hamilton-jacobi-bellman equations for q-learning in continuous time. In *Learning for Dynamics and Control*, pages 739–748. PMLR, 2020.

Jeongho Kim, Jaeuk Shin, and Insoon Yang. Hamilton-jacobi deep q-learning for deterministic continuous-time systems with lipschitz continuous controls. *Journal of Machine Learning Research*, 22(206):1–34, 2021.

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. *arXiv preprint arXiv:1509.02971*, 2015.

Rémi Munos. A study of reinforcement learning in the continuous case by the means of viscosity solutions. *Machine Learning*, 40(3):265–299, 2000.

Corentin Tallec, Léonard Blier, and Yann Ollivier. Making deep q-learning methods robust to time discretization. In *International Conference on Machine Learning*, pages 6096–6104. PMLR, 2019.

Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, volume 6, 1993.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.