TECHNISCHE UNIVERSITÄT WIEN

DIPLOMARBEIT

Deep Learning Techniques in Portfolio Optimization under Constraints

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Masterstudiums

Finanz- und Versicherungsmathematik

ausgeführt am

Institut für
Stochastik und Wirtschaftsmathematik
der Technischen Universität Wien

unter der Betreuung von

Prof. Dr. Josef Teichmann

durch

Kristof Wiedermann, BSc
Matrikelnummer: 01635829

Wien, 12. August 2021

Kristof Wiedermann

Josef Teichmann

Abstract

We consider the constrained utility maximization problem and the corresponding dual problem with regard to theoretical results which allow the formulation of algorithmic solvers which make use of deep learning techniques. We place great emphasis on detailed proofs of the underlying theoretical results. At first, the deep controlled 2BSDE algorithm from [5] is derived in a Markovian setting. It combines the dynamic programming approach with the adjoint equation from the stochastic maximum principle (SMP). In the case of random coefficients, we prove stochastic maximum principles for the primal and the dual problem, respectively. Furthermore, we show that the strong duality property holds under additional assumptions. This leads to the formulation of the deep SMP algorithm as in [5]. Moreover, we use the aforementioned result for the primal problem for defining a new algorithm, which we call deep primal SMP algorithm. Numerical examples illustrate the effectiveness of the studied algorithms – in particular for higher-dimensional problems and problems with random coefficients, which are either path dependent or satisfy their own SDEs. Moreover, our numerical experiments for constrained problems show that the novel deep primal SMP algorithm overcomes the deep SMP algorithm's weakness of erroneously producing the value of the corresponding unconstrained problem. Furthermore, in contrast to the deep controlled 2BSDE algorithm, this algorithm is also applicable to problems with path dependent coefficients. As the deep primal SMP algorithm even yields the most accurate results in many of our studied problems, we can highly recommend its usage.

Keywords: Portfolio optimization under constraints, utility maximization problem, dual problem, deep learning, machine learning, stochastic maximum principle, dynamic programming approach.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Wien, am 12. August 2021

Kristof Wiedermann

Contents

1 Introduction

In mathematical finance, as well as in practical applications, it is of greatest interest to utilize the given freedom in a model, e.g. the (potentially constrained) choice of an investment strategy, in order to optimize the expected result. In the present thesis we consider the utility maximization problem, where the quality of the terminal result, i.e. the portfolio value at terminal time, is measured by means of a strictly concave utility function U. Clearly, solving this infinite-dimensional problem is extremely difficult as one intends to maximize a functional over a space of progressively measurable processes. This is aggravated by the fact that, in practice, one usually has certain limitations in choosing the strategy. For example, selling stocks short is either limited or prohibited. We take this into consideration by requiring that the strategies have to take values in a closed, convex set K. However, the convexity of K and the strict concavity of U allow us to formulate a closely related problem by means of convex duality methods, the so-called dual problem (cf. [15, 16]). This problem is potentially easier to solve. For example, the dual control is forced to be the zero process, if the primal problem is unconstrained.

There are two popular theoretical concepts which can be used for solving the aforementioned control problems: The dynamic programming approach (DP) in a Markovian setting and the stochastic maximum principle (SMP), which is also applicable to problems with random coefficients. The first approach is centered around a nonlinear PDE, the so-called Hamilton-Jacobi-Bellman (HJB) equation, which is derived by varying the initial condition of the control problem. In some cases, the value function can then be obtained by solving the HJB equation. The second concept is based on a necessary condition for an optimal control. It states that if a control is optimal, then it necessarily has to maximize a certain function almost everywhere, where the other arguments are given by the solution to an FBSDE system.

We will see that also the first approach can be used for formulating an FBSDE system. The main idea, which then leads to the algorithmic solvers, is to model all processes which only appear in the integrands of the FBSDE systems by means of neural networks for each point in a time discretization. This concept was first studied in [1] and [6], where the authors aim at solving FBSDE systems via deep learning. In [5], this method was extended to controlled FBSDE systems which appear in the study of stochastic control problems. Hence, also the control process has to be modeled by means of neural networks. Since a neural network can be described by means of a finite-dimensional parameter vector, the algorithm, therefore, is required to solve a problem which is only finite-dimensional. The choice of neural networks is insofar appealing, as they have desirable approximation properties. Moreover, this puts the problem into the context of deep learning.

The rest of this thesis is structured as follows. In Chapter 2, we introduce the constrained utility maximization problem and derive its dual problem. In Chapter 3, we restrict ourselves to a Markovian setting. At first, we consider the dynamic programming approach and discuss theoretical conditions which ensure that the value function indeed solves the HJB equation (almost everywhere). Then the adjoint equation is introduced, which also appears in the general SMP for Markovian problems. These two ingredients are later used for proving a mathematical result which justifies the formulation of the deep controlled 2BSDE algorithm (cf. [5]). In Chapter 4, we prove SMPs for the primal problem and its dual problem, respectively. These results are then used for defining the deep SMP algorithm (cf. [5]) and the novel deep primal SMP algorithm. Moreover, we provide numerical examples which display the performance of the studied algorithms for concrete utility maximization problems (cf. Chapter 5). This includes high-dimensional problems and problems with random coefficients which are either path dependent or satisfy their own SDEs. Moreover, we apply the studied algorithms to Heston's stochastic volatility model, where the parameters are calibrated to market data. Finally, in Chapter 6, we review our results and give a preview of future work.

2 The Utility Maximization Problem and Its Dual Problem

The aim of the first section of this chapter is to present the abstract market model we are going to use in the following. In contrast to Chapter 3 of [19], we choose a rather general formulation here in order to allow the price processes to be non-Markovian. Moreover, in Section 2.2, we present the utility maximization problem and discuss useful additional assumptions on utility functions. Finally, we introduce the Legendre-Fenchel transform of a concave function in Section 2.3. We also derive some important properties of the aforementioned transform, which will be needed later. The chapter concludes with a derivation of the dual problem.

2.1 The Underlying Market Model

Let us fix a finite time horizon $T \in \mathbb{R}^+$. We consider a standard m-dimensional, $m \in \mathbb{N}^+$, Brownian motion B on a filtered probability space $(\Omega, \mathcal{F}, \mathbb{F} = (\mathcal{F}_t)_{t\in[0,T]}, \mathbb{P})$, where we choose the filtration $\mathbb{F}$ as the natural filtration generated by B, completed with all subsets of null sets of $(\Omega, \mathcal{F}, \mathbb{P})$. By construction, we obtain a complete filtration. Hence, as Brownian motion has independent future increments and continuous paths, we conclude the right-continuity of our filtration from a version of Blumenthal's zero-one law, which states that under these assumptions suitable sets from $\mathcal{F}_t$ and $\mathcal{F}_t^+$ only differ by a set of measure zero. Therefore, the filtration fulfills the so-called usual conditions. Since we have a Brownian filtration, we are allowed to apply the martingale representation theorem to local $\mathbb{F}$-martingales, which is of greatest importance for the considerations in the following chapters, in particular for Chapter 4.

Analogously to [5] and [16], we consider a market consisting of m stocks and one risk-free bond. Let $r : \Omega \times [0,T] \to \mathbb{R}$, $\mu : \Omega \times [0,T] \to \mathbb{R}^m$ and $\sigma : \Omega \times [0,T] \to \mathbb{R}^{m\times m}$ be $\mathbb{F}$-progressively measurable processes. For notational convenience, we are going to omit the argument ω in the following. Furthermore, we assume that those three processes are uniformly bounded and σ satisfies the strong non-degeneracy condition:

$$\exists k \in \mathbb{R}^+, \forall (y,t) \in \mathbb{R}^m \times [0,T] : \quad y^\intercal \sigma(t)\, \sigma^\intercal(t)\, y \geq k\, |y|^2,$$

where we denote the Euclidean norm by $|\cdot|$. According to Section 5.8 of [10], this ensures that $\sigma(t)$ and its transpose are invertible for all $t \in [0,T]$ with the inverse matrices also being uniformly bounded.

We can now define the dynamics of the risk-free bond S_0, i.e. the bank account, and the stocks S_i, $i \in \{1, \dots, m\}$, via

$$\begin{aligned} dS_0(t) &= S_0(t) r(t)\, dt, \quad t \in [0,T], \quad S_0(0) = 1, \quad \text{and} \\ dS_i(t) &= S_i(t)\big(\mu_i(t)\, dt + \sigma_{i\cdot}(t)\, dB(t)\big), \quad t \in [0,T], \quad S_i(0) > 0. \end{aligned} \tag{2.1}$$

Due to our assumptions on r, μ and σ, essentially the uniform boundedness and the progressive measurability, the SDEs in (2.1), which remind us of the dynamics of stochastic exponentials, admit unique strong solutions.

2.2 The Utility Maximization Problem in Portfolio Optimization

Suppose that a market model as discussed in Section 2.1 is given. At first, we aim at constructing a portfolio for a small investor with initial capital $x_0 \in \mathbb{R}^+$. The notion of a "small" investor is insofar important as we can assume as a consequence that trades which are executed by our investor do not affect the stock prices.

Consider a progressively measurable process $\pi : \Omega \times [0,T] \to \mathbb{R}^m$. For notational convenience, we call the set of these processes $\mathcal{A}_{prog}$. We denote by $\pi_i(t)$, $i \in \{1, \dots, m\}$, the portion of wealth invested into the ith stock at time t. Since we focus on the utility maximization problem under constraints, we restrict ourselves to processes whose images lie in a closed, convex subset $K \subseteq \mathbb{R}^m$. The set K is a priori given and we suppose that it contains the zero vector, which is equivalent to only holding the risk-free bond. As we shall see later, when checking the solvability of the SDE (2.3), it is sufficient to require that π is square-integrable with respect to the product measure $\mathbb{P} \otimes \lambda|_{[0,T]}$. Therefore, we choose to define the set of all admissible portfolio strategies like in [16] as

$$\mathcal{A} := \left\{ \pi \in \mathcal{A}_{prog} \;\middle|\; \pi(t) \in K \text{ a.s. for a.e. } t \in [0,T], \; \mathbb{E}\left[\int_0^T |\pi(t)|^2\, dt\right] < \infty \right\}. \tag{2.2}$$

Note that this implies in particular that the $H^2(0,T;\mathbb{R}^m)$-norm (cf. (3.21) below) of π is finite and the stochastic integral process $\pi \bullet B$ is a martingale for all $\pi \in \mathcal{A}$. By requiring the portfolio to be self-financing, we can now define the dynamics of the associated wealth process X^π for any given portfolio process $\pi \in \mathcal{A}$ as

$$dX^\pi(t) = \sum_{i=1}^m \frac{X^\pi(t)\pi_i(t)}{S_i(t)}\, dS_i(t) + \frac{X^\pi(t)\big(1 - \sum_{j=1}^m \pi_j(t)\big)}{S_0(t)}\, dS_0(t), \quad t \in [0,T].$$

By plugging in according to (2.1) and simplifying we obtain for every $t \in [0,T]$:

$$dX^\pi(t) = X^\pi(t)\big[\big(r(t) + \pi^\intercal(t)\sigma(t)\theta(t)\big)\, dt + \pi^\intercal(t)\sigma(t)\, dB(t)\big], \tag{2.3}$$

with initial condition $X^\pi(0) = x_0$. Here, we defined $\theta(t)$ as the market price of risk

$$\theta(t) := \sigma^{-1}(t)\big(\mu(t) - (r(t), \dots, r(t))^\intercal\big).$$

Of course, this process is also progressively measurable and uniformly bounded thanks to the strong non-degeneracy condition. Due to $\pi \in \mathcal{A}$, the uniform boundedness of σ, θ and

r and the inclusion $L^2 \subseteq L^1$ for finite measure spaces, we can conclude that the expression within the square brackets in (2.3) is the differential of a continuous semimartingale. Therefore, we obtain that there exists a unique strong solution X^π to this SDE, namely the corresponding stochastic exponential. Note that the structure of (2.3) implies that the wealth process remains positive after starting in $x_0 \in \mathbb{R}^+$. Hence, negative wealth levels, i.e. the investor's ruin, are a priori excluded.

In the following, we give a definition of a utility function which requires stronger differentiability assumptions than the classical definition. We shall see in Chapter 4 below that this proves to be helpful.

Definition 2.1. Let $U \in C^2(\mathbb{R}^+)$ be a real-valued function which is strictly increasing and strictly concave. We then call U a utility function. Furthermore, if

$$\lim_{x \searrow 0} U'(x) = \infty \quad \text{and} \quad \lim_{x \nearrow \infty} U'(x) = 0$$

hold, we say that U satisfies the so-called Inada conditions.

Due to the strict concavity of U according to the definition above, we obtain from the inverse function theorem that $I = (U')^{-1}$ is continuously differentiable, as well. This is an important observation with regard to Section 2.3. Furthermore, the Inada conditions guarantee that the domain of I is $\mathbb{R}^+$. We recall that requiring a utility function to be increasing and concave is motivated by the following practical interpretation: More wealth is more favorable to an investor and an increase of wealth, e.g. by one monetary unit, affects an average investor more than a billionaire.

For our considerations in Chapter 4, we are going to need the following additional assumption (see [11]):

Assumption 2.2. Let U be a utility function according to Definition 2.1 which satisfies the Inada conditions. We require that there exist constants $\beta \in (0,1)$ and $\gamma \in (1,\infty)$ such that

$$\forall x \in \mathbb{R}^+ : \quad \beta U'(x) \geq U'(\gamma x), \tag{2.4}$$

and $\mathrm{id}_{\mathbb{R}^+} \cdot U'$ is nondecreasing, where we denote the identity function of $\mathbb{R}^+$ by $\mathrm{id}_{\mathbb{R}^+}$.

An application of the product rule shows under the premise of the last statement that the Arrow-Pratt index of relative risk aversion of U is bounded above by 1.

Example 2.3. Consider $U_1 := \log$ and $U_{2,p} := p^{-1} \cdot (\mathrm{id}_{\mathbb{R}^+})^p$ as two prominent examples of utility functions, namely the log utility function and the power utility function with parameter $p \in (0,1)$. Clearly, these functions satisfy Definition 2.1. The derivatives are given by

$$U_1'(x) = \frac{1}{x} \quad \text{and} \quad U_{2,p}'(x) = x^{p-1}, \quad x \in (0,\infty).$$

From $(p-1) \in (-1,0)$ we can, therefore, conclude that in both cases the Inada conditions are satisfied. Furthermore, we record the fact that $\mathbb{1}_{\mathbb{R}^+}$ and $(\mathrm{id}_{\mathbb{R}^+})^p$, $p \in (0,1)$, are nondecreasing, where $\mathbb{1}_{\mathbb{R}^+}$ denotes the indicator function which maps every $x \in \mathbb{R}^+$ to 1.

Due to the simple structure of U_1', we can even choose for any $\beta \in (0,1)$ a constant γ with the desired properties, namely β^{-1}. In the case of power utility we obtain by β^q, where $q := (p-1)^{-1} \in (-\infty, -1)$, likewise a constant γ for any $\beta \in (0,1)$, i.e. an even stronger result than required via (2.4). Hence, U_1 and $U_{2,p}$, $p \in (0,1)$, satisfy Assumption 2.2.

We are now in position to formulate the constrained utility maximization problem.

Definition 2.4. Let U be a utility function according to Definition 2.1 and X^π the solution to the SDE (2.3) for $\pi \in \mathcal{A}$, where $\mathcal{A}$ is defined in (2.2). We define the gain function, which maps every $\pi \in \mathcal{A}$ to the expected utility of the portfolio at time T, by

$$J(\pi) := \mathbb{E}[U(X^\pi(T))].$$

Maximizing the expected utility corresponds to finding

$$V := \sup_{\pi \in \mathcal{A}} J(\pi). \tag{2.5}$$

A control $\pi^* \in \mathcal{A}$ is called optimal, if it attains the supremum in (2.5), i.e. $V = J(\pi^*)$.

Quite naturally, we are interested in the case where V is a real number. Initially, one has to verify that the expectation in the definition of $J(\pi)$ is well-defined. In [19] it is suggested to additionally require boundedness from below or a quadratic growth condition from U. Indeed, in the first case this is achieved since the integral of the negative part of $U(X^\pi(T))$ is finite. Clearly, this is satisfied, if $\lim_{x \searrow 0} U(x)$ is greater than $-\infty$, which holds for the power utility functions from Example 2.3 but not for the log utility function. If we assume that the closed, convex set K is bounded, then the same holds true for all processes $\pi \in \mathcal{A}$. Hence, if U satisfies a quadratic growth condition, the expectations are also well-defined in this case because of the square-integrability of $X^\pi(T)$ according to Theorem 1.3.15 of [19]. In the following, the additional assumptions for ensuring that $J(\pi)$ is well-defined might vary from one section to another. For example, in our considerations of the general non-Markovian setup in Chapter 4 we are going to require that even $U(X^\pi(T)) \in L^1$ holds for all strategies (cf. Assumption 4.10 below). We close this section with a remark regarding initial wealth and time.

Remark 2.5. As we considered a very general, not necessarily Markovian setup in this section, we studied the constrained utility maximization problem with a fixed pair of initial time and wealth $(0, x_0)$. As we shall see in Chapter 3, where all processes X^π are controlled Markov processes, we are allowed to apply the dynamic programming approach in this case. This means that we consider a family of control problems with different initial times and wealth, i.e. pairs (t,x). Hence, we obtain the so-called value function by mapping every pair (t,x) to the corresponding value given by (2.5), where the initial condition in (2.3) has to be adapted accordingly.

2.3 The Dual Problem

The aim of this section is introducing the Legendre-Fenchel transform of a concave function and proving some useful properties, especially in connection with Definition 2.1 and Assumption 2.2. Furthermore, we derive and formulate the dual problem accompanying the constrained utility maximization problem from Definition 2.4.

2.3.1 The Legendre-Fenchel Transform

The following considerations are similar to [9] and [11], since we are interested in properties of the Legendre-Fenchel transform under the special circumstances of Definition 2.1. We refer to [20] for a general theory under milder assumptions. At first, we define the Legendre-Fenchel transform of a utility function.

Definition 2.6. Let U be a utility function as defined in Definition 2.1. Then the Legendre-Fenchel transform, $\widetilde{U} : \mathbb{R}^+ \to \mathbb{R}$, is defined by

$$\widetilde{U}(y) := \sup_{x\in\mathbb{R}^+} \big\{U(x) - xy\big\}, \quad y \in \mathbb{R}^+. \tag{2.6}$$

Hence, $\widetilde{U}$ maps every $y \in \mathbb{R}^+$ to the maximum signed distance between U and a linear function starting in zero whose first derivative is equal to y. In the literature, the Legendre-Fenchel transform is usually defined for convex functions f first via $\widetilde{f}(y) := \sup_{x\in D}\{xy - f(x)\}$, where D denotes the domain of f. However, applying this definition to the strictly convex, strictly increasing function $-U(-x)$, $x \in \mathbb{R}^-$, leads precisely to (2.6).
The next lemma shows some general properties of $\widetilde{U}$.

Lemma 2.7. *Let U be a utility function satisfying the Inada conditions according to Definition 2.1 and $\widetilde{U}$ the corresponding Legendre-Fenchel transform. Then the following properties hold:*

(i) $\widetilde{U}(y) = U(I(y)) - yI(y), \quad y \in \mathbb{R}^+,$

(ii) $\widetilde{U} \in C^2(\mathbb{R}^+)$, *strictly decreasing, strictly convex,*

(iii) $U(x) = \widetilde{U}(U'(x)) + xU'(x) = \inf_{y\in\mathbb{R}^+} \big\{\widetilde{U}(y) + xy\big\}, \quad x \in \mathbb{R}^+,$

(iv) $\widetilde{U}'(y) = -I(y), \quad y \in \mathbb{R}^+,$

(v) $\widetilde{U}(0) := \lim_{y\searrow 0} \widetilde{U}(y) = \lim_{x\nearrow\infty} U(x) =: U(\infty)$ *and*

(vi) $\widetilde{U}(\infty) := \lim_{y\nearrow\infty} \widetilde{U}(y) = \lim_{x\searrow 0} U(x) =: U(0).$

Proof. (i): Let $y \in \mathbb{R}^+$ be fixed. We can infer from the strict concavity of U that the same holds true for the mapping $u_y : x \mapsto \big(U(x) - xy\big)$, $x \in \mathbb{R}^+$. Since U' is strictly decreasing and continuous and the Inada conditions hold by assumption, we are guaranteed to find a unique solution to $U'(x) = y$, namely $I(y)$, where $I := (U')^{-1}$. Hence, we can conclude from the strict concavity and $u_y \in C^2(\mathbb{R}^+)$ that is attains the global maximum at $I(y)$. This shows the desired formula.
(ii): As mentioned below Definition 2.1, I is continuously differentiable. Therefore, *(i)* and an application of chain and product rule yield

$$\begin{aligned}\widetilde{U}'(y) &= U'(I(y))I'(y) - yI'(y) - I(y) = -I(y) \quad \text{and} \\ \widetilde{U}''(y) &= -\,I'(y), \quad y \in \mathbb{R}^+.\end{aligned} \tag{2.7}$$

Hence, $\widetilde{U} \in C^2(\mathbb{R}^+)$ holds as I and I' are continuous functions. (2.7) shows as well that $\widetilde{U}$ is strictly decreasing and strictly convex, because I maps to $\mathbb{R}^+$ and $I'(y) < 0$ holds true

since the function I is strictly decreasing.
(iii): With the same argument as in the proof of *(i)*, we get that U' maps $\mathbb{R}^+$ bijectively onto itself. Therefore, a change of variables via $x := I(y)$ leads to the first equality. Due to $\widetilde{U}$ being strictly convex according to *(ii)*, the same holds true for $\widetilde{u}_x : y \mapsto \big(\widetilde{U}(y) + xy\big)$, where $x \in \mathbb{R}^+$ is fixed. Moreover, *(ii)* ensures that $\widetilde{u}_x \in C^2(\mathbb{R}^+)$ holds. Following the argument in (i), the global minimum is attained at the unique solution of $-\widetilde{U}'(y) = x$, which, combined with (2.7), shows the second equality.
(iv): This is an immediate consequence of the first equation of (2.7).
(v): At first, it has to be pointed out that, as a result of the monotonicity of $\widetilde{U}$ and U, $\widetilde{U}(0)$ and $U(\infty)$ are well-defined. We show the equality by proving both inequalities. Since $xU'(x)$ is always greater than 0, we obtain from *(iii)* and applying the limit

$$U(\infty) \geq \lim_{x \nearrow \infty} \widetilde{U}(U'(x)) = \widetilde{U}(0),$$

where we used the monotonicity of $\widetilde{U}$ (see *(ii)*) and the suitable Inada condition for the last equality.
For the proof of the reverse inequality, we take arbitrary, fixed numbers $\varepsilon, y \in \mathbb{R}^+$. Hence, also εy^{-1} lies in $\mathbb{R}^+$. Therefore, we get

$$U(y) \leq \widetilde{U}(\varepsilon y^{-1}) + \varepsilon, \quad y \in \mathbb{R}^+.$$

Since this holds for any $y \in \mathbb{R}^+$, we can send y towards infinity. Due to the fact that $\widetilde{U}$ is strictly monotone, it holds in particular that $\widetilde{U}(0)$ does not depend on the choice of the sequence converging to 0, as the limit is well-defined. Since ε was arbitrary, we are done.
(vi): The proof of this statement is very similar to *(v)*. Since $yI(y) > 0$ holds for any $y \in \mathbb{R}^+$, we can conclude from *(i)* that $\widetilde{U}(y) \leq U(I(y))$ is valid. Applying $\lim_{y \nearrow \infty}$ to this inequality, using the monotonicity of U and $\lim_{y \nearrow \infty} I(y)$=0, which follows from the suitable Inada condition, results in the first inequality.
We take again arbitrary numbers $\varepsilon, z \in \mathbb{R}^+$. Analogously to above, it follows from the definition of $\widetilde{U}$ that

$$\widetilde{U}(z) \geq U(\varepsilon z^{-1}) - \varepsilon$$

holds. Sending z towards ∞ results in $\widetilde{U}(\infty) \geq U(0) - \varepsilon$, since U is strictly monotone. Because ε was again chosen arbitrarily, the proof is completed. $\square$

In the following lemma, we summarize the implications of Assumption 2.2 for $\widetilde{U}$ and $I = -\widetilde{U}'$, which is of greatest importance for our considerations in Chapter 4.

Lemma 2.8. *In addition to the assumptions in Lemma 2.7, we require that U satisfies Assumption 2.2. Then we additionally obtain*

(i) $x \mapsto xI(x)$ *is nonincreasing on* $\mathbb{R}^+$,

(ii) $x \mapsto \widetilde{U}(\exp(x))$ *is convex on* $\mathbb{R}$ *and*

(iii) $\exists \beta \in (0,1), \gamma \in (1,\infty) : \quad \big(\forall x \in \mathbb{R}^+ : \quad I(\beta x) \leq \gamma I(x)\big).$

Proof. *(i)*: The Inada conditions and U' being continuous and strictly decreasing guarantee that U' maps $\mathbb{R}^+$ bijectively onto itself. Hence, we are allowed to change variables via $z := I(x)$, where $I := (U')^{-1}$ as usual. This leads to the mapping $z \mapsto zU'(z)$, $z \in \mathbb{R}^+$, which is nondecreasing by assumption. But since I corresponds to a strictly decreasing transformation, it follows from the chain rule, that the original function is nonincreasing.
(ii): By means of the chain rule we obtain

$$\frac{d}{dx}\widetilde{U}(\exp(x)) = \widetilde{U}'(\exp(x))\exp(x) = -I(\exp(x))\exp(x), \tag{2.8}$$

where we used Lemma 2.7 *(iv)* for the last equality. Due to exp being strictly increasing we can conclude from *(i)* that this expression is nondecreasing in x. Hence, we obtain the desired convexity, because $\widetilde{U}$ is sufficiently smooth according to Lemma 2.7 *(ii)*.
(iii): At first, we apply the strictly decreasing function I to (2.4), which results in

$$\forall x \in \mathbb{R}^+ : \quad I\big(\beta U'(x)\big) \leq \gamma x.$$

By using the same argument as in the proof of *(i)*, we can express every x uniquely via $I(y)$ with $y \in \mathbb{R}^+$. Applying this change of variables to the above inequality leads to the desired result. □

By multiplying in *(i)* and *(iii)* with -1 and Lemma 2.7 *(iv)*, we obtain the reverse results for $\widetilde{U}'$. Finally, we briefly consider Example 2.3 in the light of Lemma 2.8.

Example 2.9. Let U_1 and $U_{2,p}$, $p \in (0,1)$, be given as in Example 2.3. One can easily see that

$$I_1(x) = \frac{1}{x} \quad \text{and} \quad I_{2,p}(x) = x^{\frac{1}{p-1}}, \quad x \in \mathbb{R}^+,$$

holds. Hence, by Lemma 2.7 *(i)* we can write $\widetilde{U}_1$ and $\widetilde{U}_{2,p}$ explicitly as

$$\widetilde{U}_1(y) = -\log(y) - 1 \quad \text{and} \quad \widetilde{U}_{2,p}(y) = \frac{1-p}{p} y^{\frac{p}{p-1}}, \quad y \in \mathbb{R}^+.$$

We know from Example 2.3 that these utility functions satisfy Assumption 2.2. Therefore, Lemma 2.8 is applicable. Explicit calculation shows that the same constants β, γ as in Example 2.3 work for the statement of Lemma 2.8 *(iii)*, as indicated by its proof. Like in Example 2.3, we obtain a stronger result here, because we find for every $\beta \in (0,1)$ a constant $\gamma \in (1,\infty)$ which does the job.

2.3.2 Derivation of the Dual Problem

Solving the constrained utility maximization problem (cf. Definition 2.4) explicitly can become very difficult in many instances. Therefore, formulating an accompanying problem, that can potentially be solved with less computational effort, would be highly favorable, especially with regard to our numerical implementations using deep learning. We aim at obtaining an upper bound for the value of the classical problem while using the Legendre-Fenchel transform $\widetilde{U}$ in the formulation of this new problem. Similarly to [12], we are interested in nonnegative semimartingales Y such that $X^\pi Y$ is a supermartingale for every

$\pi \in \mathcal{A}$. By the definition of $\widetilde{U}(y)$, it is for every $x \in \mathbb{R}^+$ an upper bound for $U(x) - xy$. This observation will serve as motivation. In the following, we are going to elaborate on the details while following the approach presented in [16].

In accordance with the above mentioned idea, we choose the following ansatz to describe the dynamics of the desired one-dimensional process Y:

$$dY(t) = Y(t)\big[a(t)\,dt + b^\intercal(t)\,dB(t)\big], \quad t \in [0,T],$$

with initial condition $Y(0) = y$, where $y \in \mathbb{R}^+$. Here, the progressively measurable processes $a : \Omega \times [0,T] \to \mathbb{R}$ and $b : \Omega \times [0,T] \to \mathbb{R}^m$ have to be chosen such that Y is uniquely given by the corresponding stochastic exponential. Clearly, requiring $a \in L^1$ and $b \in L^2$ with respect to the product measure $\mathbb{P} \otimes \lambda|_{[0,T]}$ is sufficient for this purpose. Hence, we can apply for every $\pi \in \mathcal{A}$ the integration by parts formula for continuous semimartingales to $X^\pi Y$ and we obtain

$$d(X^\pi Y)(t) = (X^\pi Y)(t)\big[\big(a(t){+}r(t){+}\pi^\intercal(t)\sigma(t)\theta(t){+}\pi^\intercal(t)\sigma(t)b(t)\big)\,dt{+}\big(\pi^\intercal(t)\sigma(t){+}b^\intercal(t)\big)\,dB(t)\big],$$

for $t \in [0,T]$. Since x_0 and y are positive, the same is true for the process $X^\pi Y$ as it is a well-defined stochastic exponential thanks to our integrability assumptions. We can conclude from its explicit representation that the local martingale part is nonnegative due to the corresponding property of the exponential function. Hence, an application of Fatou's lemma for conditional expectations shows that this part is even a supermartingale. Now, if we require that the remaining part is decreasing, we are done. This is clearly satisfied if and only if we have for every $\pi \in K$:

$$a(t) + r(t) + \pi^\intercal\big(\sigma(t)\theta(t) + \sigma(t)b(t)\big) \le 0, \quad \text{a.s. for a.e. } t \in [0,T].$$

At first, we bring the expression which contains $\pi^\intercal$ to the right-hand side. Since this holds for any $\pi \in K$, the inequality is also true, if we apply the infimum with respect to this set. Converting this infimum into a supremum leads to

$$a(t) + r(t) \le -\sup_{\pi \in K}\big\{\pi^\intercal\big(\sigma(t)\theta(t) + \sigma(t)b(t)\big)\big\}, \quad \text{a.s. for a.e. } t \in [0,T].$$

Remark 2.10. We remember that the support function of $-K$, where K is a closed, convex set, is defined via

$$\delta_K(z) = \sup_{\pi \in K}\{-\pi^\intercal z\}, \quad z \in \mathbb{R}^m. \tag{2.9}$$

We observe that δ_K is positive homogeneous and subadditive since the corresponding properties also hold for the supremum. Therefore, it is a convex function, which we shall use in Chapter 4. Furthermore, δ_K is nonnegative because we assumed $0 \in K$.

By means of δ_K, we can now write the above inequality as

$$a(t) + r(t) + \delta_K\big(-\sigma(t)(\theta(t) + b(t))\big) \le 0, \quad \text{a.s. for a.e. } t \in [0,T]. \tag{2.10}$$

For notational convenience, we denote the input of δ_K in (2.10) by $v(t)$, $t \in [0,T]$. Hence, we obtain the following properties for the processes a and b from the ansatz:

$$\begin{aligned} b(t) &= -\left(\theta(t) + \sigma^{-1}(t)v(t)\right), \quad \text{and} \\ a(t) &\le -r(t) - \delta_K(v(t)), \quad \text{a.s. for a.e. } t \in [0,T]. \end{aligned} \tag{2.11}$$

Since we only get an upper bound for a, we have to examine which choice of a yields the smallest upper bound regarding the classical problem. We claim that this holds, if we have equality in (2.11). In this case, we obtain the following SDE:

$$dY^{(y,v)}(t) = -Y^{(y,v)}(t)\big[\big(r(t)+\delta_K(v(t))\big)\,dt+\big(\theta(t)+\sigma^{-1}(t)v(t)\big)^\intercal\,dB(t)\big], \quad t\in[0,T], \tag{2.12}$$

with initial condition $Y^{(y,v)}(0) = y$. Note that we hereby obtain a family of SDEs which can be described by means of the inital condition y and a progressively measurable process v, as indicated by the notation. The process v, which will be called dual control process from now on, has to be chosen such that the above SDE admits a unique strong solution. This is clearly satisfied, if we introduce the set of all admissible pairs of initial values and dual control processes, similarly to (2.2), via

$$\mathcal{D} := \left\{(y,v)\in\mathbb{R}^+\times\mathcal{A}_{prog}\;\middle|\;\mathbb{E}\left[\int_0^T\big[|v(t)|^2+\delta_K(v(t))\big]\,dt\right]<\infty\right\}. \tag{2.13}$$

Let $(y,v)\in\mathcal{D}$ be fixed and a, so that we do not have equality in (2.11). This leads to an SDE with the same initial condition, whose diffusion part agrees with the corresponding structure in (2.12):

$$dY(t) = Y(t)\big[a(t)\,dt - \big(\theta(t)+\sigma^{-1}(t)v(t)\big)^\intercal\,dB(t)\big], \quad t\in[0,T], \quad Y(0)=y.$$

Therefore, by comparison of the explicit stochastic exponential representations of the solutions we conclude that $Y^{(y,v)}(t)\geq Y(t)$ holds almost surely for almost every $t\in[0,T]$. Hence, as $\widetilde{U}$ is decreasing, we obtain

$$\mathbb{E}\big[\widetilde{U}(Y^{(y,v)}(T))\big]\leq\mathbb{E}\big[\widetilde{U}(Y(T))\big],$$

which implies that the choice of a as in (2.12) is indeed optimal for our purpose. By the definition of $\widetilde{U}$ and the supermartingale property of $X^\pi Y^{(y,v)}$ we obtain the following result:

$$\mathbb{E}\big[U(X^\pi(T))\big]\leq\mathbb{E}\big[\widetilde{U}(Y^{(y,v)}(T))\big]+\mathbb{E}\big[X^\pi(T)Y^{(y,v)}(T)\big]\leq\mathbb{E}\big[\widetilde{U}(Y^{(y,v)}(T))\big]+x_0y. \tag{2.14}$$

Since this is true for any $\pi\in\mathcal{A}$ and $(y,v)\in\mathcal{D}$, we can take the supremum on the left-hand side and the infimum on the right-hand side. This leads to

$$V=\sup_{\pi\in\mathcal{A}}\mathbb{E}\big[U(X^\pi(T))\big]\leq\inf_{(y,v)\in\mathcal{D}}\big\{\mathbb{E}\big[\widetilde{U}(Y^{(y,v)}(T))\big]+x_0y\big\}, \tag{2.15}$$

where V is exactly the same as in (2.5). The right-hand side corresponds to the so-called dual problem:

Definition 2.11. Let a constrained utility maximization problem according to Definition 2.4 be given. The value of the corresponding dual problem is defined by

$$\widetilde{V} := \inf_{(y,v)\in\mathcal{D}}\big\{\mathbb{E}\big[\widetilde{U}(Y^{(y,v)}(T))\big]+x_0y\big\}, \tag{2.16}$$

where $\mathcal{D}$ is given by (2.13) and $Y^{(y,v)}$ denotes the unique solution to the SDE (2.12) for a fixed tuple $(y,v)\in\mathcal{D}$. A pair $(y^*,v^*)\in\mathcal{D}$ is called optimal, if it attains the infimum on the right-hand side of (2.16).

Note that we also have to optimize with respect to the initial condition here, whereas x_0 is a priori given in the original problem. A crucial advantage of the dual problem compared to the original one is that the set K is directly incorporated within the dynamics of the dual "wealth" process given by (2.12). Except for the integrability condition on $\delta_K(v)$, the set of all admissible dual pairs, as defined in (2.13), does not depend on K. Therefore, it is possible that the dual problem becomes unconstrained. Contrary to this, the dynamics of X^π does not explicitly depend on K. In this case, we implement the constraints by means of explicitly requiring $\pi \in K$ in the definition of $\mathcal{A}$. However, this is certainly a more difficult condition in view of solving the optimization problem by means of the algorithms which will we introduced later.

We conclude this section with three classical examples of closed, convex sets and their implications for the corresponding dual problems.

Example 2.12. We consider $K_1 := \mathbb{R}^m$, $K_2 := \overline{B}_r$, $r > 0$, and a closed convex cone K_3, e.g. the conical hull of finitely many vectors such that the origin does not lie in the convex hull of the original vectors. Here, the last requirement ensures the closedness of the conical hull. $\overline{B}_r$ denotes the closed ball of radius r centered at the origin.
Clearly, δ_{K_1} maps to infinity with the exception of $\delta_{K_1}(0) = 0$. Therefore, a dual control process v has to agree with the zero process (a.s for a.e $t \in [0, T]$) in order to be admissible according to (2.13). Hence, only the optimal initial dual wealth y^* needs to be found, which decisively simplifies the dual optimization problem.
Due to the Cauchy-Schwarz inequality, we know that the scalar product in the definition of δ_{K_2} attains its maximum absolute value, if the two vectors are linearly dependent. For any $(-z) \in \mathbb{R}^m \setminus \{0\}$, the vector from K_2 pointing in the same direction with maximum length is given by

$$\pi_{max} := -r\frac{z}{|z|}.$$

Hence, we obtain $\delta_{K_2}(z) = r|z|$, $z \in \mathbb{R}^m$, which simplifies the admissibility condition (2.13).
Finally, we study K_3. At first, we consider $z \in \mathbb{R}^m$ such that for any $\pi \in K_3$, $-\pi^\intercal z \leq 0$ holds. Since we have $0 \in K$ by assumption, we obtain $\delta_{K_3}(z) = 0$ in this case. For every other z, there exists $\pi \in K_3$ such that $-\pi^\intercal z$ is strictly greater than zero. As K_3 is a cone, $\lambda\pi$ lies in K_3 as well for every $\lambda \in \mathbb{R}^+$. Hence, the set $\{-\pi^\intercal z\}_{\pi \in K_3}$ is not bounded above, i.e. $\delta_{K_3}(z) = \infty$. Note that K_1 is a special case thereof.

3 The Deep Controlled 2BSDE Algorithm in a Markovian Setting

In this chapter, we discuss the special case of the market model introduced in Section 2.1, where r, μ and σ are deterministic processes, i.e. their paths do not depend on $\omega \in \Omega$. These assumptions imply that the risk-free bond and the m stocks become Markov processes due to the structure of the SDEs in (2.1). We refer to Chapter 7 of [18] for a proof in the time-homogeneous case, where the drift and diffusion coefficient at time t are given by measurable functions of the demanded process at t. It is claimed that if these coefficients also depend explicitly on t, this case can be reduced to the situation above via considering the process $(t, X(t))_{t\in[0,T]}$. Hence, the state processes in the utility maximization problem and the associated dual problem become controlled Markov diffusion processes, which are thoroughly discussed in [7].

As briefly mentioned in Remark 2.5, this controlled Markovian structure allows us to study a family of control problems with varying initial conditions. In accordance with this idea, we discuss the dynamic programming approach for a more general type of control problems, cf. Section 3.1, in the second section of this chapter. Moreover, we show how this principle motivates the derivation of the so-called Hamilton-Jacobi-Bellman equation. Furthermore, the stochastic maximum principle is presented in Section 3.3, whose first-order adjoint equation is also an essential ingredient for the ensuing formulation of the deep controlled 2BSDE algorithm in Section 3.4.

3.1 Formulation of a More General Control Problem

Since we intend to formulate the deep controlled 2BSDE algorithm such that it is applicable to the constrained utility maximization problem as well as the associated dual problem, we choose a more abstract notation in the following, which covers both cases. We shall see that the results of the following sections indeed hold for both problems and even for more general ones, if certain technical properties are satisfied.

Let the probability space from Section 2.1, $K \subseteq \mathbb{R}^m$ and two deterministic, measurable functions $a : [0,T] \times \mathbb{R} \times K \to \mathbb{R}$ and $b : [0,T] \times \mathbb{R} \times K \to \mathbb{R}^m$ be given, which are locally Lipschitz continuous in the second argument. This is important with respect to the measurability of the value function (cf. (3.4)). Furthermore, we consider all progressively measurable, K-valued processes π such that

$$dX^{(t,x),\pi}(s) = a\big(s, X^{(t,x),\pi}(s), \pi(s)\big)\, ds + b^\intercal\big(s, X^{(t,x),\pi}(s), \pi(s)\big)\, dB(s), \tag{3.1}$$

has a pathwise unique strong solution for all initial conditions $X^{(t,x),\pi}(t) = x$, where $(t,x) \in [0,T] \times \mathbb{R}$. We denote the subset consisting of all processes π with the above property which are considered for the specific control problem with $\mathcal{A}_M$. In the following, we usually restrict $\mathcal{A}_M$ to control processes which lie in $H^2(0,T;K)$ (cf. (3.21) below) since in the proof of the DPP (cf. Theorem 3.4) one intends to equip the control space with a metric.

Remark 3.1. Due to the choice of $\mathcal{A}$ and $\mathcal{D}$ in Chapter 2, the above requirements are satisfied by the SDEs of the form (2.3) and (2.12) for any $\pi \in \mathcal{A}$ and $(y,v) \in \mathcal{D}$, respectively. Clearly, since the processes r, μ and σ are assumed to be deterministic in this chapter, these SDEs admit a representation as in (3.1), while choosing $K = \widetilde{K} := \{z \in \mathbb{R}^m \mid \delta_K(z) < \infty\}$ for the dual problem. It is important to point out that, contrary to the classical setting as presented in [7] or [19], we do not require that a and b satisfy a Lipschitz condition in x which holds uniformly for all t and π. The main reason for imposing such assumptions is ensuring the existence of a unique strong solution to the SDE (3.1). However, we do not need those properties for this purpose due to the special structure of (2.3) and (2.12), which allows us to apply the corresponding results for stochastic exponentials of continuous semimartingales. We note that the aforementioned Lipschitz condition is satisfied for the primal problem, if K is bounded. For the dual problem, it would require the boundedness of $\delta_K(\widetilde{K})$ and $\widetilde{K}$, respectively.

Remark 3.2. One disadvantage of not requiring the uniform Lipschitz condition is that classical moment estimates for the solutions might no longer hold. We refer to Section 2.5 of [13] for detailed proofs of the estimates under Lipschitz and linear growth conditions. The proofs rely heavily on Grönwall's lemma and Hölder's inequality. The aforementioned estimates essentially state that the pth Moment of $\big(X^{(t,x),\pi}\big)^*(T) := \sup_{s\in[t,T]} \big|X^{(t,x),\pi}(s)\big|$ is bounded above by the pth moment of the initial random variable modulo additive and multiplicative constants. This upper bound is trivially finite, if one considers constant starting values as in the setting of our market model. The following example shows that, in general, we cannot draw this conclusion in our setup.

Example 3.3. We consider a utility maximization problem constrained to the closed, convex set $K = \mathbb{R}_0^+$ and $p = 2$. For the sake of simplicity, we remove the strong non-degeneracy condition here. Hence, we can choose $m = 1$, $T \in \mathbb{R}^+$, $x_0 = 1$, $\sigma \equiv r \equiv 0$ and $\mu \equiv 1$. Furthermore, we consider the strategy defined via $\pi := \mathbb{1}_{[T/2,T]}(\cdot)\exp\big(B(T/2) - T/4\big)$. The idea is to obtain the mgf of a log-normally distributed random variable as a lower bound for $\mathbb{E}\big[((X^\pi)^*(T))^2\big]$, where $X^\pi := X^{(0,1),\pi}$.
At first, we have to check that $\pi \in \mathcal{A}$ holds indeed. Obviously, π cannot take values outside of K. Furthermore, the progressive measurability of π follows from the right-continuity of its paths and the adaptedness of B. A short calculation shows

$$\mathbb{E}\bigg[\int_0^T |\pi(t)|^2\,dt\bigg] = \frac{T}{2}\,\mathbb{E}\Big[e^{2B(T/2)-\frac{T}{2}}\Big] = \frac{T}{2}e^{\frac{T}{2}} < \infty,$$

whence $\pi \in \mathcal{A}$ follows. However, $X^\pi(T)$ is not in L^2:

$$\mathbb{E}\big[(X^\pi(T))^2\big] = \mathbb{E}\Big[e^{Te^{B(T/2)-\frac{T}{4}}}\Big] = \infty, \tag{3.2}$$

where the last equality follows due to $\exp\big(B(T/2) - T/4\big)$ being non-degenerate log-normally distributed, $T > 0$ and the fact that the mgf of a non-degenerate log-normal distribution is infinite/not defined for positive numbers. This concludes the counterexample, since $|X^\pi(T)| \leq (X^\pi)^*(T)$ holds.

Now we turn to the formulation of a generalized optimization problem, which admits the constrained utility maximization problem from Section 2.2 as a special case. As indicated in Remark 2.5, we intend to map any initial pair $(t,x) \in [0,T] \times \mathbb{R}$ to the value of the corresponding optimization problem instead of merely considering one fixed pair $(0,x_0)$ as in Definition 2.4.

As in [19], we consider measurable functions $f : [0,T] \times \mathbb{R} \times K \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$. The function g corresponds to the terminal gain function. As discussed below Definition 2.4, g being bounded from below or satisfying a polynomial growth condition (if the respective moments of $X^{(t,x),\pi}$ are finite, cf. Remark 3.2) ensures that the expectation in (3.3) is well-defined. For example, let $g|_{\mathbb{R}^+} := U$ be a utility function, where g is set to zero on the remaining real line (and analogously for $\widetilde{U}$). Since $\widetilde{U}$ is decreasing, we obtain from Lemma 2.7 *(vi)* that U is bounded from below if and only if $\widetilde{U}$ has this property.

Contrary to the setup in Chapter 2, we introduce a running gains function f here, which enlarges the set of the considered control problems. The reason for allowing only control problems of a very specific type in Chapter 2 is that we encounter in general a non-Markovian setup there due to the potential path-dependency of r, μ and σ. As we shall see in Chapter 4, deriving optimality conditions becomes rather difficult in this case, since the approach of this chapter is in general invalid under those circumstances. For the same reason, we only consider utility functions in Chapter 2, which justifies the application of convex duality and optimality methods. In contrast to this, we note that we do not require g to satisfy any (strict) convexity/concavity properties here.

The function f has to be chosen such that the corresponding integral in (3.3) is well-defined. We assume that f is bounded until further notice. We shall see later that the dynamic programming approach and the classical stochastic maximum principle can be quite easily extended to problems with running gains. Hence, also the deep controlled 2BSDE algorithm can be applied to more general control problems than the ones discussed in Chapter 2.
Now we can define the gain function similarly to Definition 2.4 for any initial pair $(t,x) \in [0,T] \times \mathbb{R}$ and $\pi \in \mathcal{A}_M$ via

$$J(t,x,\pi) := \mathbb{E}\Big[\int_t^T f\big(s, X^{(t,x),\pi}(s), \pi(s)\big)\, ds + g\big(X^{(t,x),\pi}(T)\big)\Big]. \tag{3.3}$$

Again, we intend to maximize J with respect to π. This leads to the so-called value function:

$$v(t,x) := \sup_{\pi \in \mathcal{A}_M} J(t,x,\pi), \quad (t,x) \in [0,T] \times \mathbb{R}. \tag{3.4}$$

Even though the dual problem is a minimization problem, in particular also with respect to the initial value y (using the notation of Chapter 2), we can still write it in the form of (3.4), where we ignore the minimization with respect to y and the summand $x_0 y$ from (2.16) for the moment. This leads to the value function

$$\widetilde{v}(t,y) := \inf_{v \in \mathcal{D}_2} \mathbb{E}\big[\widetilde{U}(Y^{(t,y),v}(T))\big], \quad (t,y) \in [0,T] \times \mathbb{R}, \tag{3.5}$$

where $\mathcal{D}_2 := \{v \in \mathcal{A}_{prog} | \, (1, v) \in \mathcal{D}\}$ with $\mathcal{D}$ as defined in (2.13) and $\widetilde{U}$ is defined as the zero function on the negative real line. Furthermore, $Y^{(t,y),v}$ corresponds to the process with dynamics given by (2.12) and satisfying the initial condition $Y^{(t,y),v}(t) = y$. Note that the solvability of a stochastic exponential-type SDE with global integrability assumptions on the coefficients as in (2.13) does not depend on the choice of the deterministic initial pair (t, y). Hence, $-\widetilde{v}$ describes the value function of a classical controlled maximization problem as in (3.4) with $g = -\widetilde{U}\,\mathbb{1}_{\mathbb{R}^+}$ and $f \equiv 0$. Finally, we have to minimize for every $t \in [0, T]$ with respect to $y \in \mathbb{R}^+$:

$$\widetilde{V}(t) := \inf_{y \in \mathbb{R}^+} \{\widetilde{v}(t, y) + x_0 y\}, \quad t \in [0, T], \tag{3.6}$$

where x_0 denotes the starting value of the state process in the associated primal problem. For $t = 0$, this yields exactly $\widetilde{V}$ from (2.16). We conclude from (3.6) that if we consider dual problems, we have an additional variable which has to be optimized, namely y. Hence, we need to add another loss function to the deep controlled 2BSDE algorithm in this case, as we shall see in Section 3.4.

3.2 The Dynamic Programming Approach Leading to the Hamilton-Jacobi-Bellman Equation

Since, in the setting of this chapter, the state processes of the form (3.1) are controlled Markovian diffusion processes, we can apply the dynamic programming approach in order to analyze the dynamics of the associated value function. The first essential result for this purpose is the so-called dynamic programming principle (DPP), which is stated in the following:

Theorem 3.4. *Let a stochastic control problem as discussed in Section 3.1 be given with associated value function v. Then the following holds for every $(t, x) \in [0, T] \times \mathbb{R}$ and stopping time $\tau : \Omega \to [t, T]$:*

$$v(t, x) = \sup_{\pi \in \mathcal{A}_M} \mathbb{E}\left[\int_t^\tau f\big(s, X^{(t,x),\pi}(s), \pi(s)\big)\, ds + v\big(\tau, X^{(t,x),\pi}(\tau)\big)\right]. \tag{3.7}$$

There are many proofs for this result in the literature. Usually these require explicitly that a and b satisfy a Lipschitz condition in x, which holds uniformly for all t and π, and a boundedness condition. The main purpose of these requirements is ensuring that (3.1) admits a pathwise unique strong solution. However, this holds by assumption in our setting (cf. Remark 3.1). We refer to [19] for a great overview of the proof concept which involves proving both inequalities in (3.7) separately. The most delicate part of the proof is finding an admissible, thus in particular progressively measurable, ε-optimal control for the problem starting in $(\tau, X^{(t,x),\pi}(\tau))$ as τ is not necessarily deterministic. In [4], this problem is tackled by means of a measurable selection theorem. It is shown that there exists a Borel-measurable function which maps ν-almost every pair (t, x) to an ε-optimal control for the problem starting in (t, x), where ν is an arbitrary product measure formed by a measure on $[0, T]$ and a probability measure on $\mathbb{R}$. This result is then applied to

$\big(\tau(\omega), X^{(t,x),\pi}(\omega,\tau(\omega))\big)$, $\omega \in \Omega$, and the measure induced by $X^{(t,x),\pi}(\tau)$ leading to an ε-optimal control for the problem starting in $(\tau, X^{(t,x),\pi}(\tau))$.

It has to be pointed out that Theorem 3.4 holds in particular for constant stopping times, i.e. deterministic times $\tau \equiv s \in [t,T]$. (3.7) admits the following natural interpretation: If we want to invest optimally in $[t,T]$, we can subdivide the time interval into $[t,\tau]$ and $[\tau,T]$ and proceed as follows: Roughly speaking, we can choose for any $\pi \in \mathcal{A}_M$ an optimal trading strategy, if it exists, for the problem starting at time τ, where the corresponding initial wealth is exactly given by $X^{(t,x),\pi}(\tau)$ as we applied the strategy π for $[t,\tau]$. Additionally, the running gains generated by π during $[t,\tau]$ have to be added. Clearly, we are interested in the strategy π which maximizes the expected gains of this procedure. (3.7) states that this approach is equivalent to behaving optimally on $[t,T]$.

Since it is certainly very difficult to solve (3.7) directly in order to obtain v, we are interested in an infinitesimal version of it. This leads to the so-called Hamilton-Jacobi-Bellman (HJB) equation, which will be formally derived below. As these considerations rely heavily on Itô's formula, we have to assume that the value function is sufficiently smooth, i.e. we require $v \in C^{1,2}([0,T]\times\mathbb{R})$. We want to place emphasis on the fact that this assumption is non-trivial, as Example 3.7 shows.

Let $(t,x) \in [0,T]\times\mathbb{R}$ be fixed, $h \in (0,T-t]$ an arbitrary number and $\pi \in K$ an arbitrary, but fixed, constant control. Moreover, we assume that all such constant controls are admissible. Obviously, this is trivially satisfied by the constrained utility maximization problem. Since K is precisely chosen as the set $\widetilde{K}$ for the dual problem (cf. Remark 3.1), the above assumption also holds for the dual problem. Furthermore, we define the following stopping time:

$$\tau_{t,x,1} := \inf\big\{s \in [t,T] : \big|X^{(t,x),\pi}(s) - x\big| \geq 1\big\} \wedge T.$$

Since $X^{(t,x),\pi}$ is a continuous and adapted process, this defines indeed a stopping time. Hereby, we intend to keep the stopped process $\big(X^{(t,x),\pi}\big)^{\tau_{t,x,1}}$ close to the starting value x. By Theorem 3.4 applied to the stopping time $\tau_h := \tau_{t,x,1} \wedge (t+h)$, we obtain

$$v(t,x) \geq \mathbb{E}\bigg[\int_t^{\tau_h} f\big(s, X^{(t,x),\pi}(s), \pi\big)\,ds + v\big(\tau_h, X^{(t,x),\pi}(\tau_h)\big)\bigg]. \tag{3.8}$$

Since $v \in C^{1,2}([0,T]\times\mathbb{R})$ holds by assumption, we can apply Itô's formula to $v\big(\tau_h, X^{(t,x),\pi}(\tau_h)\big)$, which leads to

$$\begin{aligned}0 \geq \mathbb{E}\bigg[\int_t^{\tau_h} \Big(& f\big(s, X^{(t,x),\pi}(s),\pi\big) + a\big(s, X^{(t,x),\pi}(s),\pi\big)\frac{\partial v}{\partial x}\big(s, X^{(t,x),\pi}(s)\big) \\ &+ \frac{\partial v}{\partial t}\big(s, X^{(t,x),\pi}(s)\big) + \frac{1}{2}\big|b\big(s, X^{(t,x),\pi}(s),\pi\big)\big|^2 \frac{\partial^2 v}{\partial x^2}\big(s, X^{(t,x),\pi}(s)\big)\Big)\,ds \\ &+ \int_t^{\tau_h} b^{\intercal}\big(s, X^{(t,x),\pi}(s),\pi\big)\frac{\partial v}{\partial x}\big(s, X^{(t,x),\pi}(s)\big)\,dB(s)\bigg],\end{aligned} \tag{3.9}$$

where we subtracted $v(t,x)$ from both sides of the inequality. For notational convenience, we define for $\pi \in K$:

$$\mathcal{L}^{\pi} v(s,y) := a(s,y,\pi)\frac{\partial v}{\partial x}(s,y) + \frac{1}{2}\big|b(s,y,\pi)\big|^2\frac{\partial^2 v}{\partial x^2}(s,y), \quad (s,y) \in [0,T]\times\mathbb{R}. \tag{3.10}$$

We wish that the expectation of the above stochastic integral with respect to B is zero, which is certainly satisfied if the expectation of the corresponding covariation is finite, i.e.

$$\mathbb{E}\bigg[\int_t^{\tau_h} \big|b\big(s, X^{(t,x),\pi}(s), \pi\big)\big|^2 \left(\frac{\partial v}{\partial x}\right)^2 \big(s, X^{(t,x),\pi}(s)\big)\, ds\bigg] < \infty.$$

Clearly, the partial derivative part does not prevent us from achieving our goal, since $X^{(t,x),\pi}$ is bounded up to time τ_h and the partial derivative with respect to x is a continuous function by assumption. Hence, this factor is bounded by a constant C. However, it is non-trivial that the remaining integral has finite expectation, since we assumed no continuity properties for b and we can only conclude almost sure finiteness of the integral from $X^{(t,x),\pi}$ being a strong solution to (3.1). Hence, we have to assume that this property holds. At least, it is encouraging that this holds for the original and the dual state processes given by (2.3) and (2.12), respectively, due to the boundedness of $\big(X^{(t,x),\pi}\big)^{\tau_h}$, θ, σ and σ^{-1} and the respective choice of the set of admissible controls. Under this assumption, simplifying (3.9) and dividing by $h > 0$ leads to

$$0 \geq \mathbb{E}\bigg[\frac{1}{h}\int_t^{\tau_h} f\big(s, X^{(t,x),\pi}(s), \pi\big) + \frac{\partial v}{\partial t}\big(s, X^{(t,x),\pi}(s)\big) + \mathcal{L}^\pi v\big(s, X^{(t,x),\pi}(s)\big)\, ds\bigg]. \tag{3.11}$$

In the following, we would like to send h towards zero and apply the dominated convergence theorem to (3.11). As the paths of $X^{(t,x),\pi}$ are continuous, we can find for (almost) every ω a positive real number h_ω such that $t + h_\omega \leq \tau_{t,x,1}(\omega)$ holds, i.e. $\tau_h(\omega) = t + h$ for all $h \in (0, h_\omega]$. Hence, we can conclude from the fundamental theorem of calculus (for Lebesgue integrals), the integrability properties according to the concept of strong solutions and the boundedness of the occurring partial derivative terms that the expression within the expectation converges in particular almost surely to

$$f(t, x, \pi) + \frac{\partial v}{\partial t}(t, x) + \mathcal{L}^\pi v(t, x).$$

Since we assumed that f is bounded, we only have to consider conditions on a and b which guarantee that we find an integrable majorant. For example, requiring that a and b are continuous is sufficient. Alternatively, as π is a constant, it would also be sufficient for this purpose to require boundedness in t for all (x, π) and continuity in x, which reflects exactly the situation of the primal and the dual problem as discussed in Chapter 2. Therefore, under suitable conditions, an application of the dominated convergence theorem shows

$$0 \geq f(t, x, \pi) + \frac{\partial v}{\partial t}(t, x) + \mathcal{L}^\pi v(t, x). \tag{3.12}$$

Since $\pi \in K$ was chosen arbitrarily, the above inequality also holds, if we take the supremum on the right-hand side, which results in

$$0 \geq \frac{\partial v}{\partial t}(t, x) + \sup_{\pi \in K} \big\{ f(t, x, \pi) + \mathcal{L}^\pi v(t, x) \big\}. \tag{3.13}$$

Finally, it has to be justified why it is reasonable to require equality in (3.13). Under the assumption that there exists an optimal control π^* attaining the supremum in (3.7), we have equality in (3.8), where π is not necessarily a constant process anymore. By calculations analogous to above, we obtain "equality" in (3.12), i.e.

$$0 = \mathbb{E}\bigg[\frac{\partial v}{\partial t}(t,x) + f(t,x,\pi^*(t)) + \mathcal{L}^{\pi^*(t)}v(t,x)\bigg], \tag{3.14}$$

where the requirements for a and b which ensure that the above arguments work have to be adapted accordingly, as the influence of the control on the integrals is non-trivial anymore. This is closely related to the structure of a and b concerning π and potential integrability assumptions required in the definition of $\mathcal{A}_M$. We refer to Assumption 3.18 below for a set of conditions which guarantees that the HJB equation, i.e. equality in (3.13), is almost surely satisfied for the points of the form $\big(s, X^{(t,x),\pi^*}(s)\big)$, $s \in [t,T]$, where π^* is an optimal control. We shall see in Subsection 3.3.2 that this result suffices for our considerations, i.e. we do not need the HJB equation in its full strength. This is insofar encouraging as, in the case of the primal and the dual problem from Chapter 2 (with deterministic r, μ and σ), the aforementioned result holds, if v is sufficiently smooth, an optimal control π^* exists and certain growth and integrability conditions are fulfilled. The multiplicative structure of the coefficients and the integrability assumptions in (2.2) and (2.13), respectively, will be essential for this purpose (cf. Example 3.19 and Lemma 3.20 below).

Note that, contrary to (3.12), we still have the expectation operator in (3.14) since $\pi^*(t)$ is not necessarily a constant random variable. However, because $\pi^*(t)$ is K-valued, we can conclude from (3.13) that the expression within the expectation is nonpositive. Therefore, combining this with (3.14) leads to

$$0 = \frac{\partial v}{\partial t}(t,x) + f(t,x,\pi^*(t)) + \mathcal{L}^{\pi^*(t)}v(t,x) \quad \text{a.s.} \tag{3.15}$$

Hence, it is plausible to require equality in (3.13) with regard to the formulation of the HJB equation, which is lastly obtained by multiplying both sides with -1. Finally, we shall consider $t = T$ by deriving a reasonable terminal condition. (3.3) and (3.4) show immediately that requiring $v(T,\cdot) = g$ serves this purpose.

Remark 3.5. In contrast to the sufficient requirements above, which have been formulated after each step such that the respective argument works, we want to emphasize that a priori requiring additional properties for a, b, f and g, which are in general not fulfilled in the setting of Section 3.1, leads indeed to the value function v necessarily satisfiying the HJB equation, if $v \in C^{1,2}([0,T]\times\mathbb{R})$ holds. We refer to [22] for a result in this direction. There, a, b, f and g are supposed to be uniformly continuous. Furthermore, it is required that each function satisfies a Lipschitz condition with respect to x which holds uniformly in t and π, as well as for $x = 0$, these functions are supposed to be bounded in t and π. We notice that all constant controls are admissible under these assumptions. Furthermore, in contrast to our considerations above and essentially due to the uniform continuity, one does not have to presuppose the existence of an optimal control π^*, as the proof in [22] shows. Hence, whenever $v \in C^{1,2}([0,T]\times\mathbb{R})$ holds in this setting, v necessarily has to solve the HJB equation.

Now we are in position to define the HJB equation associated with the general stochastic control problem, as previously introduced.

Definition 3.6. Let a stochastic control problem be given as discussed in Section 3.1. Then the so-called Hamilton-Jacobi-Bellman (HJB) equation associated with this problem is defined via

$$-\frac{\partial v}{\partial t}(t,x) - \sup_{\pi \in K} \left\{ f(t,x,\pi) + \mathcal{L}^{\pi} v(t,x) \right\} = 0, \quad (t,x) \in [0,T) \times \mathbb{R}, \tag{3.16}$$

where we are interested in solutions satisfying the terminal condition

$$v(T,x) = g(x), \quad x \in \mathbb{R}. \tag{3.17}$$

Furthermore, the so-called Hamiltonian $H : [0,T] \times \mathbb{R} \times K \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ associated with this stochastic control problem is defined as

$$H(t,x,\pi,y,z) = a(t,x,\pi)y + \frac{1}{2}\left|b(t,x,\pi)\right|^2 z + f(t,x,\pi), \quad (t,x,\pi,y,z) \in [0,T] \times \mathbb{R} \times K \times \mathbb{R} \times \mathbb{R}.$$

Alternatively, we can write the HJB equation, which is a nonlinear PDE of second order, by means of the associated Hamiltonian as

$$-\frac{\partial v}{\partial t}(t,x) - \sup_{\pi \in K} H\left(t,x,\pi,\frac{\partial v}{\partial x}(t,x),\frac{\partial^2 v}{\partial x^2}(t,x)\right) = 0, \quad (t,x) \in [0,T) \times \mathbb{R}. \tag{3.18}$$

As the entire considerations above are performed under the premise that $v \in C^{1,2}([0,T] \times \mathbb{R})$ holds, which is also a necessary requirement for classical solutions to (3.16), it is natural to ask whether this is in general true for value functions. Unfortunately, this has to be unambiguously negated. The following example, which is based on [18] and [19], shows that the aforementioned condition can even be violated in a rather simple setting.

Example 3.7. We consider an unconstrained stochastic control problem as discussed in Section 3.1 with $m = 1$. Hence, we have $K = \mathbb{R}$. We choose $\mathcal{A}$ from (2.2) as the set of our admissible strategies. Furthermore, we consider state processes which satisfy the following SDE:

$$dX^{(t,x),\pi}(s) = \pi(s)\, dB(s), \quad s \in [t,T], \quad X^{(t,x),\pi}(t) = x,$$

where $\pi \in \mathcal{A}$, which greatly simplifies (3.1). Due to the definition of $\mathcal{A}$ we can even conclude that $X^{(t,x),\pi}$ is a martingale for every control process π and initial pair (t,x). Moreover, we choose $f \equiv 0$ and g as a function which satisfies a linear growth condition. Therefore, $\mathbb{E}\left[g\left(X^{(t,x),\pi}(T)\right)\right]$ is always finite due to the integrability of $X^{(t,x),\pi}(T)$.
Moreover, as all constant controls $\pi \equiv c \in \mathbb{R}$ lie in $\mathcal{A}$ and $a \equiv 0$ and $b \equiv c$ holds in this case, we can conclude that the above arguments leading to (3.12) are in fact applicable to this problem, if we assume $v \in C^{1,2}([0,T] \times \mathbb{R})$. Hence, we obtain

$$\forall (t,x) \in [0,T) \times \mathbb{R} : \quad 0 \geq \frac{\partial v}{\partial t}(t,x) + \frac{c^2}{2}\frac{\partial^2 v}{\partial x^2}(t,x).$$

Since the first summand on the right-hand side is obviously finite and the above inequality holds for every $c \in \mathbb{R}$, it is immediately clear that the second-order partial derivative has to be nonpositive, i.e. the function $v(t,\cdot)$ is concave for every $t \in [0,T)$.

In the following, we denote the upper concave envelope of g by g^c. It follows from $X^{(t,x),0} \equiv x$ and (3.4) that $v(t,\cdot) \geq g$ holds for every $t \in [0,T)$. Due to the concavity of $v(t,\cdot)$ and the minimality of g^c we can conclude that also $v(t,\cdot) \geq g^c$ holds for every $t \in [0,T)$.
Finally, we want to show that also the converse inequality is true. We convince ourselves thereof by performing the following short calculation, which uses $g^c \geq g$, Jensen's inequality for concave functions and $X^{(t,x),\pi}$ being a martingale:

$$\forall (t,x) \in [0,T) \times \mathbb{R} : \quad v(t,x) \leq \sup_{\pi \in \mathcal{A}} \mathbb{E}\big[g^c\big(X^{(t,x),\pi}(T)\big)\big] \leq g^c(x).$$

Hence, the value function in this example is explicitly given by

$$v(t,x) = g^c(x), \quad (t,x) \in [0,T) \times \mathbb{R}.$$

Therefore, v cannot lie in $C^{1,2}([0,T] \times \mathbb{R})$, if $g^c \notin C^2(\mathbb{R})$. For example, if we choose $g_1(x) := -\max\{K-x,0\} = \min\{x-K,0\}$ with $K \in \mathbb{R}^+$, which corresponds to shorting a put option on the value of the state process at T, we obtain the desired contradiction because $g_1^c = g_1$ is not even continuously differentiable.
Another way for finding a counterexample by means of the above results is considering functions g which differ from their upper concave envelope g^c. We can conclude for such a function that the corresponding value function v is not even continuous in t, which results from the jump at T. Therefore, $v \notin C^{1,2}([0,T] \times \mathbb{R})$ follows in particular. As an example, we consider $g_2(x) := |x|$, $x \in \mathbb{R}$, leading to $g_2^c \equiv \infty$. Moreover, we want to verify by hand that the corresponding value function is indeed infinite for $t \in [0,T)$. For constant controls $\pi \equiv c \in \mathbb{R}$ we obtain $\big|X^{(t,x),c}(T)\big| \overset{d}{=} |x + cY|$, where $Y \sim \mathcal{N}(0, T-t)$. Since Y is non-degenerate normally distributed and c can be chosen arbitrarily large, it follows that $v(t,\cdot) = \infty$ indeed holds true for every $t \in [0,T)$.

As outlined in the course of the formal derivation of the HJB equation and in Remark 3.5, the value function necessarily has to satisfy the HJB equation, if $v \in C^{1,2}([0,T] \times \mathbb{R})$ and certain additional properties hold. Hence, a reasonable approach for identifying the value function of the control problem is to analyze the solutions to the associated HJB equation. The aim of so-called verification theorems is to give sufficient conditions such that a particular solution to (3.16) and (3.17) is indeed the value function of the studied control problem. Even though there are many different versions of such theorems in the literature, they all agree with regard to the basic idea of the statement and the proof. For example, results in this direction can be found in [7], [18], where a control problem is studied for which only Markovian controls of the form $\pi(t) = h(t, X(t))$ are allowed, [19] and [22]. In the following, we cite a slightly generalized version of the result presented in [19], as this allows us to formulate a recipe for solving certain stochastic control problems.

Theorem 3.8. *Let a stochastic control problem as in Section 3.1 be given and let* $u \in C^{1,2}([0,T] \times \mathbb{R})$ *be a classical solution to* (3.16) *satisfying the terminal condition* (3.17). *Moreover, we suppose that there exists a measurable function* $\overline{\pi}^* : [0,T) \times \mathbb{R} \to K$*, such that for every pair* $(t,x) \in [0,T) \times \mathbb{R}$ *the supremum in* (3.16)*, or equivalently in* (3.18)*, is attained by* $\overline{\pi}^*(t,x)$*. Additionally, we assume that the following SDE, which is closely related to* (3.1)*, admits a unique strong solution for every initial pair* (t,x)*:*

$$dX^{(t,x)}(s) = a\big(s, X^{(t,x)}(s), \overline{\pi}^*\big(s, X^{(t,x)}(s)\big)\big)\, ds + b^\intercal\big(s, X^{(t,x)}(s), \overline{\pi}^*\big(s, X^{(t,x)}(s)\big)\big)\, dB(s).$$

We suppose that $\pi^*_{(t,x)} := \big(\overline{\pi}^*\big(s, X^{(t,x)}(s)\big)\big)_{s\in[t,T]} \in \mathcal{A}_M$ *holds. Furthermore, we assume that* u *satisfies a polynomial growth condition, i.e.* $\exists k \in \mathbb{N}^+$ *such that*

$$\exists C \in \mathbb{R}^+, \forall (t,x) \in [0,T]\times\mathbb{R} : \quad |u(t,x)| \le C\big(1+|x|^k\big), \tag{3.19}$$

and additionally that $\big(X^{(t,x),\pi}\big)^*(T), \big(X^{(t,x)}\big)^*(T) \in L^k$ *holds true for every pair* (t,x) *and* $\pi \in \mathcal{A}_M$ *with the same constant* k *as above. Then* u *agrees with the value function globally on* $[0,T]\times\mathbb{R}$ *and* $\pi^*_{(t,x)}$ *corresponds to an optimal control for the related optimization problem starting in* (t,x)*.*

In the following, as the arguments for proofing such statements are usually very similar, we briefly summarize the main idea of the applied proof concept. We refer to Section 3.5 of [19] for a detailed proof for the special case $k = 2$. At first, we are for every $(t,x) \in [0,T) \times \mathbb{R}$ and $\pi \in \mathcal{A}_M$ interested in a sequence of stopping times $(\tau_n)_{n\in\mathbb{N}}$ which satisfies $\tau_n \nearrow T$ as $n \to \infty$ and guarantees that the stochastic integral resulting from Itô's formula applied to the stopped process $u\big(\cdot\wedge\tau_n, X^{(t,x),\pi}(\cdot\wedge\tau_n)\big)$ is for each n even a true martingale, i.e. it has expectation 0. Moreover, by the HJB equation we can find an upper estimate containing f for the remaining integrand. Taking the expectation, removing the stopping times, which is justified by the growth and integrability assumptions and the dominated convergence theorem, and using the terminal condition leads to $v \le u$. However, if a function $\overline{\pi}^*$ satisfying the required properties exits, then the above estimate for the integrand becomes, in fact, equality for $\pi^*_{(t,x)}$. Therefore, one can conclude $u = v$.

Note that, contrary to [19], we have to explicitly impose integrability assumptions on the controlled supremum processes, as this is non-trivial in our setup according to Remark 3.2 and Example 3.3. Theorem 3.8 suggests the following procedure for solving a control problem:

- $\forall (t,x) \in [0,T)\times\mathbb{R}$: Determine maximizers of the Hamiltonian for the pair (t,x) (cf. (3.18)), where the partial derivatives serve as placeholders. This can be used for defining a function $\overline{\pi}^*$, once a solution is known.
- Solve the HJB equation (3.16) with terminal condition (3.17), if possible, by plugging in the maximizers from the previous step and finding an appropriate ansatz for the resulting PDE. Let u denote a solution.
- Finally, check whether u and $\overline{\pi}^*$ meet the requirements formulated in Theorem 3.8.

A successful application of this approach can be found in Section 3.6 of [19], where Merton's classical portfolio allocation problem with power utility function is studied. Here, the choice of the utility function greatly simplifies the first two steps of the above procedure (cf. also Examples 5.1 and 5.3 below).

We notice that requiring the existence of such a measurable function $\overline{\pi}^*$ is rather strong. However, by Theorem 3.8 we then get intriguing additional insights into the considered family of control problems. For instance, we obtain that these optimal controls even have a Markovian structure. Furthermore, the knowledge of $\overline{\pi}^*$ leads for all initial pairs (t,x) via

$\pi^*_{(t,x)}$ to an optimal control for the corresponding control problem. In contrast to this, the results in [7] and [22] give for a fixed initial pair (t,x) a condition on an admissible control π^* which ensures that π^* is optimal for this specific problem. To be more precise, π^* has to satisfy almost surely for almost every $s \in [t,T]$:

$$\begin{aligned}&H\left(s, X^{(t,x),\pi^*}(s), \pi^*(s), \frac{\partial u}{\partial x}\big(s, X^{(t,x),\pi^*}(s)\big), \frac{\partial^2 u}{\partial x^2}\big(s, X^{(t,x),\pi^*}(s)\big)\right)\\ &= \sup_{\pi\in K} H\left(s, X^{(t,x),\pi^*}(s), \pi, \frac{\partial u}{\partial x}\big(s, X^{(t,x),\pi^*}(s)\big), \frac{\partial^2 u}{\partial x^2}\big(s, X^{(t,x),\pi^*}(s)\big)\right),\end{aligned} \tag{3.20}$$

i.e. π^* maximizes the Hamiltonian in the above sense. Clearly, similar arguments as presented below Theorem 3.8 show that π^* is indeed optimal. We will return to (3.20) in the course of Section 3.3 below (cf. (3.34)).

Remark 3.9. It has to be pointed out that the above considerations regarding the verification step were performed under the premise that a classical, hence sufficiently smooth, solution to the HJB equation (3.16) exists, which satisfies the terminal condition (3.17). However, this is a non-trivial assumption. Fortunately, one can find sufficient, albeit strong, conditions such that a classical solution indeed exist. We refer to Section IV.4 of [7] for a summary of different results in this direction which rely in a large part on arguments thoroughly presented in Chapter 6 of [14]. In the following, we cite one possible collection of requirements which is sufficient for this purpose. The paramount assumption, which all the aforementioned results have in common, is that b is supposed to satisfy the following uniform parabolicity condition: There exists a constant $c \in \mathbb{R}^+$ such that

$$\forall (t,x,\pi) \in [0,T) \times \mathbb{R} \times K,\ \forall y \in \mathbb{R}: \quad |b(t,x,\pi)|^2 y^2 \geq c y^2,$$

holds, i.e. $|b|^2$ is bounded from below by $c \in \mathbb{R}^+$. Note that this simple structure results from the fact that the state processes are one-dimensional in our considerations. We notice that this property clearly does not hold for the dynamics (2.3) in the classical utility maximization problem due the multiplicative structure of the diffusion coefficient and $0 \in K$ by assumption. Furthermore, it is assumed that K is compact and a, $|b|^2$, f and g satisfy certain smoothness properties, namely $a, |b|^2, f \in C_b^{1,2}$ and $g \in C_b^3$. As usual, we denote by C_b^j the set of functions from C^j such that the function itself and its partial derivatives up to order j are bounded. In conclusion, if the above conditions hold, then we obtain the existence of a candidate function for our verification procedure as described above.

We conclude this section with two remarks on the notion of viscosity solutions of HJB equations, which represents a significantly weaker solution concept and allows to associate even not necessarily smooth value functions with (3.16) in an appropriate way. This is important, as value functions are in general not even for problems with a simple structure smooth enough (cf. Example 3.7) and sufficient conditions for obtaining the existence of a classical solution to (3.16) are very strong (cf. Remark 3.9).

Remark 3.10. We remember that $u \in C^0([0,T]\times\mathbb{R})$ is called a viscosity sub-/supersolution to (3.16) with terminal condition (3.17), if for every test function $\phi \in C^{1,2}([0,T] \times \mathbb{R})$ and all maximizers/minimizers $(\overline{t}, \overline{x}) \in [0,T) \times \mathbb{R}$ of $u - \phi$ the following holds:

$$-\frac{\partial \phi}{\partial t}(\overline{t}, \overline{x}) - \sup_{\pi \in K} H\left(\overline{t}, \overline{x}, \pi, \frac{\partial \phi}{\partial x}(\overline{t}, \overline{x}), \frac{\partial^2 \phi}{\partial x^2}(\overline{t}, \overline{x})\right) \leq / \geq 0, \quad \text{and}$$

$$\forall x \in \mathbb{R}: \quad u(T,x) \leq / \geq g(x).$$

Consequently, u is called a viscosity solution, if it is a viscosity sub- as well as a supersolution. By considering the smooth functions $\widetilde{\phi}(t,x) := \phi(t,x) + (u(\overline{t}, \overline{x}) - \phi(\overline{t}, \overline{x}))$, $(t,x) \in [0,T] \times \mathbb{R}$, for all maximizers/minimizers $(\overline{t}, \overline{x})$ we can require w.l.o.g that $u(\overline{t}, \overline{x}) = \phi(\overline{t}, \overline{x})$ holds in the above definition. Hence, the viscosity sub-/supersolution property can be interpreted as follows: Every $\phi \in C^{1,2}([0,T] \times \mathbb{R})$ which lies above/below u has to satisfy the classical sub-/supersolution property for all intersection points of ϕ and u. Obviously, it is reasonable to require certain properties which are closely related to (3.16) for arbitrary, sufficiently smooth test functions in the above notion, since the derivatives of u do not even have to exist. Clearly, if $u \in C^{1,2}([0,T] \times \mathbb{R})$ is a viscosity solution, then it is also a classical solution to the HJB equation, as $\phi = u$ shows. Since also the converse holds (see [7] or [22] for a proof), the concept of viscosity solution represents indeed a meaningful generalization of the classical notion of a solution to a PDE.

Remark 3.11. A crucial result in the setting of Remark 3.5 is that the value function v is the unique viscosity solution to the associated HJB equation in a class of functions which satisfy certain growth conditions. We refer to [22] for details and a proof. The proof of the viscosity solution property of v essentially relies on the application of the DPP (cf. Theorem 3.4) to v and Itô's formula to an arbitrary function ϕ with the same properties as above. By the maximality/minimality of $(\overline{t}, \overline{x})$ we get $v(\overline{t}, \overline{x}) - \phi(\overline{t}, \overline{x}) - v(t,x) + \phi(t,x) \geq / \leq 0$, in particular for pairs (t,x) close to $(\overline{t}, \overline{x})$ with $t > \overline{t}$. Finally, similar arguments as in the formal derivation of the HJB equation and Remark 3.5 lead to the desired result.
Furthermore, we refer to Chapter 4 of [19] for a more generalized notion of viscosity solutions, where the continuity assumption is replaced by local boundedness. However, one has to consider the so-called HJB variational inequality in this case.

3.3 The Stochastic Maximum Principle

The aim of this section is to present the statement of the so-called stochastic maximum principle (SMP) for stochastic control problems as introduced in Section 3.1. Hereby, we obtain a second tool for solving stochastic control problems, since the SMP is derived independently from the dynamic programming approach. The main statement of the SMP is that, under additional assumptions, an optimal control process necessarily maximizes a Hamiltonian-type function with respect to $\pi \in K$, where the other arguments are given by processes which solve certain forward and backward SDEs (BSDEs).

Furthermore, we will see that if we require additional properties which make the results from Section 3.2 applicable, we can even find exact solutions to the first-order adjoint equation ocurring in the SMP by means of the value function's partial derivatives. These results essentially form the motivation for the formulation of the deep controlled 2BSDE algorithm as in Section 3.4 below.

3.3.1 A General Formulation

At first, we recall the definition of BSDEs and cite some results which guarantee the existence of solutions. We refer to [19] and [22] for a detailed exposition. In contrast to the heretofore considered (forward) SDEs, the solutions to BSDEs are required to satisfy a terminal condition. However, as solutions to SDEs have to be adapted, finding solutions to BSDEs becomes rather delicate because a reversal of time as for ODEs is not necessarily helpful. Furthermore, unlike in the forward case, finding the diffusion part should also be a part of the solution as the following simple example shows (see [22] for details): At first, we define for every $j \in \mathbb{N}$ the following spaces of progressively measurable processes for later usage:

$$\begin{aligned} H^2(0,T;\mathbb{R}^j) &:= \left\{ X : \Omega \times [0,T] \to \mathbb{R}^j \;\middle|\; \text{prog.m.} \;\wedge\; \mathbb{E}\left[\int_0^T |X(t)|^2 \, dt \right] < \infty \right\} \quad \text{and} \\ S^2(0,T;\mathbb{R}^j) &:= \left\{ X : \Omega \times [0,T] \to \mathbb{R}^j \;\middle|\; \text{prog.m.} \;\wedge\; \mathbb{E}\left[\sup_{t\in[0,T]} |X(t)|^2 \right] < \infty \right\}. \end{aligned} \tag{3.21}$$

Let $\xi \in L^2(\Omega, \mathcal{F}_T, \mathbb{P}; \mathbb{R})$ be given such that ξ is not $\mathcal{F}_0$-measurable. Then the "simple" SDE with drift and diffusion coefficient being equal to the zero process and terminal condition ξ clearly does not admit an adapted solution, in contrast to the corresponding forward SDE combined with a measurable initial random variable. However, we can circumvent this issue by considering the martingale $Y := \mathbb{E}[\xi|\mathcal{F}_\cdot]$ instead and applying the martingale representation theorem, which is allowed as the filtration is Brownian, in order to obtain $Y = \mathbb{E}[\xi] + Z \bullet B$ for a process $Z \in H^2(0,T;\mathbb{R}^m)$. Hence, Y satisfies the SDE

$$dY(t) = Z^\intercal(t)\, dB(t), \quad t \in [0,T], \quad Y(T) = \xi \quad \text{a.s.}$$

As the process Z is a priori unknown, it has to be part of the solution. This motivates Definition 3.12 below. Using the above notation, we can define the following:

Definition 3.12. Consider the probability space from Section 2.1. Let $h : \Omega \times [0,T] \times \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}$ be a measurable function, called generator, and $\xi \in L^2(\Omega, \mathcal{F}_T, \mathbb{P}; \mathbb{R})$ the random variable representing the terminal condition. For notational convenience, we omit ω in the following. A pair of processes $(Y,Z) \in S^2(0,T;\mathbb{R}) \times H^2(0,T;\mathbb{R}^m)$, whose component processes are thus in particular adapted, is called solution to the BSDE

$$dY(t) = h(t, Y(t), Z(t))\, dt + Z^\intercal(t)\, dB(t), \quad t \in [0,T], \quad Y(T) = \xi \quad \text{a.s.,} \tag{3.22}$$

if the following holds almost surely for every $t \in [0,T]$:

$$Y(t) = \xi - \int_t^T h(s, Y(s), Z(s))\, ds - \int_t^T Z^\intercal(s)\, dB(s). \tag{3.23}$$

If, in addition, $\big(h(t,y,z)\big)_{t\in[0,T]}$ is progressively measurable for every $(y,z) \in \mathbb{R} \times \mathbb{R}^m$, $\big(h(t,0,0)\big)_{t\in[0,T]} \in H^2(0,T;\mathbb{R})$ and there exists a constant $C \in \mathbb{R}^+$ such that almost surely the following uniform Lipschitz condition holds:

$$\forall t \in [0,T],\ \forall y_1, y_2 \in \mathbb{R},\ \forall z_1, z_2 \in \mathbb{R}^m : \quad |h(t,y_2,z_2) - h(t,y_1,z_1)| \le C(|y_2 - y_1| + |z_2 - z_1|),$$

then the BSDE (3.22) admits a unique solution $(Y, Z) \in S^2(0, T; \mathbb{R}) \times H^2(0, T; \mathbb{R}^m)$. A proof of this statement relying on Banach's fixed point theorem for contraction maps can be found in [19] and [22]. Even though we will not need this existence and uniqueness result in the following, it provides the interested reader with an important insight with regard to the solvability of BSDEs.

As a preparation for the formulation of the adjoint equations, we define the so-called generalized Hamiltonian for a minimization problem first (cf. Remark 3.17):

Definition 3.13. Let a minimization problem as discussed in Section 3.1 be given. Then the function $\widetilde{\mathcal{H}} : [0, T] \times \mathbb{R} \times K \times \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}$ which maps every $(t, x, \pi, y, z) \in [0, T] \times \mathbb{R} \times K \times \mathbb{R} \times \mathbb{R}^m$ to

$$\widetilde{\mathcal{H}}(t, x, \pi, y, z) := a(t, x, \pi)\, y + b^\intercal(t, x, \pi)\, z - f(t, x, \pi) \tag{3.24}$$

is called the generalized Hamiltonian for the minimization problem.

Now we can define the first- and second-order adjoint equations:

Definition 3.14. Consider a minimization problem as introduced in Section 3.1 and fix $(t, x) \in [0, T] \times \mathbb{R}$ and $\pi \in \mathcal{A}_M$. If the functions a, b, f and g are sufficiently smooth, we can define the first-order adjoint equation via

$$dP_1(s) = -\frac{\partial \widetilde{\mathcal{H}}}{\partial x}\big(s, X^{(t,x),\pi}(s), \pi(s), P_1(s), Q_1(s)\big)\, ds + Q_1^\intercal(s)\, dB(s), \quad s \in [t, T], \tag{3.25}$$

with accompanying terminal condition $P_1(T) = -\frac{\partial g}{\partial x}\big(X^{(t,x),\pi}(T)\big)$ and the second-order adjoint equation via

$$\begin{aligned} dP_2(s) = -\Bigg[& 2\,\frac{\partial a}{\partial x}\big(s, X^{(t,x),\pi}(s), \pi(s)\big)\, P_2(s) + \left|\frac{\partial b}{\partial x}\big(s, X^{(t,x),\pi}(s), \pi(s)\big)\right|^2 P_2(s) \\ & + 2\left(\frac{\partial b}{\partial x}\right)^\intercal \big(s, X^{(t,x),\pi}(s), \pi(s)\big)\, Q_2(s) \\ & + \frac{\partial^2 \widetilde{\mathcal{H}}}{\partial x^2}\big(s, X^{(t,x),\pi}(s), \pi(s), P_1(s), Q_1(s)\big)\Bigg]\, ds + Q_2^\intercal(s)\, dB(s), \quad s \in [t, T], \end{aligned} \tag{3.26}$$

with terminal condition $P_2(T) = -\frac{\partial^2 g}{\partial x^2}\big(X^{(t,x),\pi}(T)\big)$, where $\frac{\partial b}{\partial x}$ has to be understood componentwise.

The SMP states that if there exists an optimal control $\pi^* \in \mathcal{A}_M$ with corresponding state process $X^{(t,x),\pi^*}$ and processes $P_1^*, P_2^* \in S^2(t, T; \mathbb{R})$ and $Q_1^*, Q_2^* \in H^2(t, T; \mathbb{R}^m)$ which satisfy the adjoint equations (cf. Definition 3.14), then π^* necessarily has to maximize a Hamiltonian-related function, as defined below, with respect to $\pi \in K$. However, this is not the most satisfying formulation, as it is not specified, when we can indeed expect that solutions to the adjoint equations exist. This is insofar a delicate question, as (3.1), (3.25) and (3.26) form a system of one forward and two backward SDEs, i.e. they are coupled. At least it is encouraging that (3.1) is independent of P_1^*, Q_1^*, P_2^* and Q_2^* and the generators of (3.25) and (3.26) are affine in Q_1^* and Q_2^*, respectively. In the following, we cite the assumptions made in [22] which guarantee the solvability of (3.25) and (3.26). In addition to the assumptions incorporated in the setting of Section 3.1 we require:

Assumption 3.15. The functions a, b, f and g are twice continuously differentiable in x and there exists a constant $L \in \mathbb{R}^+$ and moduli of continuity $w_1, w_2 : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ such that the aforementioned functions, each denoted by h, satisfy the following properties:

$$\begin{aligned}
&\forall s \in [0,T],\ x_1, x_2 \in \mathbb{R},\ \pi_1, \pi_2 \in K: \quad |h(s,0,\pi_1)| \le L,\\
&|h(s,x_2,\pi_2) - h(s,x_1,\pi_1)| \le L|x_2 - x_1| + w_1(|\pi_2 - \pi_1|),\\
&|h_x(s,x_2,\pi_2) - h_x(s,x_1,\pi_1)| \le L|x_2 - x_1| + w_2(|\pi_2 - \pi_1|) \quad \text{and}\\
&|h_{xx}(s,x_2,\pi_2) - h_{xx}(s,x_1,\pi_1)| \le w_2(|x_2 - x_1| + |\pi_2 - \pi_1|),
\end{aligned}$$

where h_x and h_{xx} denote the corresponding partial derivatives for notational convenience.

This enables us to formulate the version of the SMP presented in [22]:

Theorem 3.16. *Let a minimization problem (cf. Remark 3.17) in the setting of Section 3.1 be given and suppose that Assumption 3.15 holds and there exists an optimal control $\pi^* \in \mathcal{A}_M$. Then there exist pairs of processes (P_1^*, Q_1^*) and (P_2^*, Q_2^*) which satisfy the associated adjoint equations (cf. Definition 3.14) and it holds almost surely for almost every $s \in [t,T]$:*

$$\begin{aligned}
&\widetilde{\mathcal{H}}\big(s, X^{(t,x),\pi^*}(s), \pi^*(s), P_1^*(s), Q_1^*(s)\big) - \frac{1}{2}\big|b\big(s, X^{(t,x),\pi^*}(s), \pi^*(s)\big)\big|^2 P_2^*(s)\\
&= \sup_{\pi \in K} \Big\{ \widetilde{\mathcal{H}}\big(s, X^{(t,x),\pi^*}(s), \pi, P_1^*(s), Q_1^*(s)\big) + \frac{1}{2}\big|b\big(s, X^{(t,x),\pi^*}(s), \pi\big)\big|^2 P_2^*(s) \qquad (3.27)\\
&\qquad - b^\intercal\big(s, X^{(t,x),\pi^*}(s), \pi^*(s)\big)\, b\big(s, X^{(t,x),\pi^*}(s), \pi\big)\, P_2^*(s) \Big\},
\end{aligned}$$

which is equivalent to the so-called variational inequality: It holds for every $\pi \in K$:

$$\begin{aligned}
&\widetilde{\mathcal{H}}\big(s, X^{(t,x),\pi^*}(s), \pi^*(s), P_1^*(s), Q_1^*(s)\big) - \widetilde{\mathcal{H}}\big(s, X^{(t,x),\pi^*}(s), \pi, P_1^*(s), Q_1^*(s)\big)\\
&-\frac{1}{2}\Big|b\big(s, X^{(t,x),\pi^*}(s), \pi^*(s)\big) - b\big(s, X^{(t,x),\pi^*}(s), \pi\big)\Big|^2 P_2^*(s) \ge 0, \quad \textit{a.s. for a.e. } s \in [t,T].
\end{aligned}$$

Proof. We refer to Theorem 3.2 in Chapter 3 of [22] for a proof. □

Remark 3.17. Note that we explicitly formulated Theorem 3.16 for a minimization problem, since the statement in the above form serves as the formal starting point for our considerations in Section 4.1 with regard to the dual problem. As already discussed in connection with (3.5), a maximization problem described by (3.3) and (3.4) can easily be transformed into a minimization problem with "gain functions" $-f$ and $-g$, and vice versa. Hence, by defining the generalized Hamiltonian for a maximization problem $\mathcal{H}$ analogously to (3.24) via

$$\mathcal{H}(t,x,\pi,y,z) := a(t,x,\pi)\,y + b^\intercal(t,x,\pi)\,z + f(t,x,\pi), \qquad (3.28)$$

for all $(t,x,\pi,y,z) \in [0,T] \times \mathbb{R} \times K \times \mathbb{R} \times \mathbb{R}^m$, the first-order adjoint equation becomes

$$dP_1(s) = -\frac{\partial \mathcal{H}}{\partial x}\big(s, X^{(t,x),\pi}(s), \pi(s), P_1(s), Q_1(s)\big)\, ds + Q_1^\intercal(s)\, dB(s), \quad s \in [t,T], \qquad (3.29)$$

with associated terminal condition $P_1(T) = \frac{\partial g}{\partial x}\big(X^{(t,x),\pi}(T)\big)$ and similarly for the second-order case. (3.29) will play an important role in the following subsection.

3.3.2 Deriving a Maximum Principle by Means of the Dynamic Programming Approach

The aim of this subsection is to show that if certain additional requirements are fulfilled and π^* is an optimal control, then $\big(v\big(s, X^{(t,x),\pi^*}(s)\big)\big)_{s\in[t,T]}$ is a solution to a BSDE which is coupled with (3.1) and (3.29) and π^* maximizes the classical Hamiltonian in a suitable sense (cf. (3.44) below). Hence, solving control problems with the deep controlled 2BSDE algorithm essentially consists of solving a system of one forward and two backward SDEs and making sure that a maximum condition holds. At first, we formulate assumptions which guarantee that the generator of (3.29) is well-defined and which are necessary for the derivation of the HJB equation in the points $\big(s, X^{(t,x),\pi^*}(s)\big)$, $s \in [t,T]$, (cf. (3.33)).

Assumption 3.18. The generalized Hamiltonian of the considered control problem $\mathcal{H}$ is differentiable with respect to x, which is determined by the corresponding property of a, b and f. Furthermore, $\mathcal{A}_M$ is chosen such that for all $\pi \in \mathcal{A}_M$, $x \in \mathbb{R}$ and $n \in \mathbb{N}$:

$$\begin{aligned}&\Big(\sup_{t\in[0,T]}\sup_{y\in[x-n,x+n]}\big|a\big(t,y,\pi(s)\big)\big|\Big)_{s\in[0,T]} \in H^1(0,T;\mathbb{R}) \quad \text{and}\\ &\Big(\sup_{t\in[0,T]}\sup_{y\in[x-n,x+n]}\big|b\big(t,y,\pi(s)\big)\big|\Big)_{s\in[0,T]} \in H^2(0,T;\mathbb{R})\end{aligned} \tag{3.30}$$

hold, where $H^1(0,T;\mathbb{R})$ is defined analogously to (3.21). Moreover, all constant controls are required to be admissible.

Note that we assumed the boundedness of f in Section 3.1. If we intend to relax this assumption, we have to impose in Assumption 3.18 the same requirements on f as on a. In the following example, we briefly discuss Assumption 3.18 in the light of the utility maximization problem and its dual problem (with r, μ and σ being deterministic).

Example 3.19. Since the processes r, μ and σ are deterministic, we observe that (2.3) is of the structure (3.1) with $a_1(t,x,\pi) := x(r(t) + \pi^\intercal\sigma(t)\theta(t))$ and $b_1(t,x,\pi) := x\sigma^\intercal(t)\pi$ for every $(t,x,\pi) \in [0,T]\times\mathbb{R}\times K$. Therefore, the associated generalized Hamiltonian is for every $(t,x,\pi,y,z) \in [0,T]\times\mathbb{R}\times K\times\mathbb{R}\times\mathbb{R}^m$ given by

$$\mathcal{H}_1(t,x,\pi,y,z) := x\,(r(t) + \pi^\intercal\sigma(t)\theta(t))\,y + x\,\pi^\intercal\sigma(t)\,z. \tag{3.31}$$

The set of admissible strategies is defined by $\mathcal{A}_{M,1} := \mathcal{A}$, cf. (2.2). Since $f \equiv 0$ holds in this setup, the differentiability of $\mathcal{H}_1$ with respect to x follows directly from the respective properties of a_1 and b_1. Clearly, a_1 and b_1 are bounded in t, i.e. the functions $a_1(\cdot,x,\pi)$ and $b_1(\cdot,x,\pi)$ are bounded for every $(x,\pi)\in\mathbb{R}\times K$, as this also holds for r, θ and σ, respectively. It is an immediate consequence of (2.2) that all constant controls are admissible. Moreover, the integrability properties (3.30) hold due to a_1 and b_1 being continuous in x, the compactness of $[x-n,x+n]$, the boundedness of a_1 and b_1 in t, $\pi\in H^2(0,T;\mathbb{R}^m)$ according to (2.2) and the fact that $L^2 \subseteq L^1$ holds for finite measure spaces.
With arguments similar to above, we obtain that (2.12) from the dual problem is also of the form (3.1) with $a_2(t,x,\pi) := -x\,(r(t)+\delta_K(\pi))$ and $b_2(t,x,\pi) := -x\,(\theta(t)+\sigma^{-1}(t)\,\pi)$ for every $(t,x,\pi)\in[0,T]\times\mathbb{R}\times\widetilde{K}$ and $\mathcal{A}_{M,2} := \mathcal{D}$, cf. (2.13). It follows with arguments

analogous to above that all the requirements from Assumption 3.18 are also fulfilled by the dual problem. In particular, we can conclude that all constant control processes are admissible as they are required to map to the set $\widetilde{K}$. Therefore, the support function is necessarily finite for these processes. Since we have $f \equiv 0$ here, it follows that (3.24) and (3.28) agree. Hence, we can denote the generalized Hamiltonian by $\mathcal{H}_2$ in this case. For every $(t, x, \pi, y, z) \in [0, T] \times \mathbb{R} \times \widetilde{K} \times \mathbb{R} \times \mathbb{R}^m$ it is explicitly given by

$$\mathcal{H}_2(t, x, \pi, y, z) := -x\left(r(t) + \delta_K(\pi)\right) y - x\left(\theta(t) + \sigma^{-1}(t)\,\pi\right)^{\intercal} z. \tag{3.32}$$

Hence, in conclusion, Assumption 3.18 is not restrictive with respect to the constrained utility maximization problem and its associated dual problem. This is insofar satisfying as these problems are also the main focus of this thesis.

The next lemma presents conditions in connection with Assumption 3.18 which guarantee that the value function almost surely satisfies the HJB equation in the points $\left(s, X^{(t,x),\pi^*}(s)\right)$, $s \in [t, T]$. Moreover, it shows that an optimal control maximizes the Hamiltonian in the sense of (3.20).

Lemma 3.20. *Let a maximization problem in the setting of Section 3.1 with value function v be given and suppose that Assumption 3.18 holds and $v \in C^{1,2}([0, T] \times \mathbb{R})$. Then it follows that v satisfies* (3.13)*. Furthermore, consider the problem starting in (t, x) and assume that there exists an optimal control $\pi^* \in \mathcal{A}_M$. If v, in addition, satisfies a polynomial growth condition of order $k \in \mathbb{N}$, cf.* (3.19)*, and $\left(X^{(t,x),\pi^*}\right)^*(T) \in L^k$ holds for the corresponding state process then v necessarily satisfies the following equation which is closely related to the associated HJB equation (cf. Definition 3.6):*

$$\begin{aligned} 0 = &-\sup_{\pi \in K} H\left(s, X^{(t,x),\pi^*}(s), \pi, \frac{\partial v}{\partial x}\left(s, X^{(t,x),\pi^*}(s)\right), \frac{\partial^2 v}{\partial x^2}\left(s, X^{(t,x),\pi^*}(s)\right)\right) \\ &-\frac{\partial v}{\partial t}\left(s, X^{(t,x),\pi^*}(s)\right), \quad \text{a.s. for a.e. } s \in [t, T]. \end{aligned} \tag{3.33}$$

Moreover, π^* satisfies almost surely for almost every $s \in [t, T]$:

$$\begin{aligned} &H\left(s, X^{(t,x),\pi^*}(s), \pi^*(s), \frac{\partial v}{\partial x}\left(s, X^{(t,x),\pi^*}(s)\right), \frac{\partial^2 v}{\partial x^2}\left(s, X^{(t,x),\pi^*}(s)\right)\right) \\ &= \sup_{\pi \in K} H\left(s, X^{(t,x),\pi^*}(s), \pi, \frac{\partial v}{\partial x}\left(s, X^{(t,x),\pi^*}(s)\right), \frac{\partial^2 v}{\partial x^2}\left(s, X^{(t,x),\pi^*}(s)\right)\right). \end{aligned} \tag{3.34}$$

Proof. The proof of the first assertion follows from the fact that the assumptions have precisely been chosen such that the arguments presented in Section 3.2 leading to (3.13) work. In particular, (3.30) ensures that the occurring stochastic integral is even a true martingale and that the dominated convergence theorem is applicable.
For the proof of the other claims, we consider a sequence of stopping times $(\tau_n)_{n \in \mathbb{N}}$, where for each $n \in \mathbb{N}$, τ_n is defined by

$$\tau_n := \inf\left\{s \in [t, T] : \left|X^{(t,x),\pi^*}(s) - x\right| \geq n\right\} \wedge T.$$

Since $X^{(t,x),\pi^*}$ has continuous paths, we can conclude that $\tau_n \nearrow T$ holds almost surely as $n \to \infty$. We notice that the differentiability of b with respect to x implies its continuity in x. Therefore, (3.30) implies that the expectation of the integral with respect to B resulting from the application of Itô's formula to $v\big(\tau_n, X^{(t,x),\pi^*}(\tau_n)\big)$ is zero as $\big(X^{(t,x),\pi^*}\big)^{\tau_n}$ is bounded. Hence, by using the optimality of π^*, the definition of H, expanding additively and taking the expectation we obtain

$$\begin{aligned}
J(t,x,\pi^*) \overset{(1)}{=} v(t,x) &= \mathbb{E}\big[v\big(\tau_n, X^{(t,x),\pi^*}(\tau_n)\big)\big] + \mathbb{E}\bigg[\int_t^{\tau_n} f\big(s, X^{(t,x),\pi^*}(s), \pi^*(s)\big)\,ds\bigg] \\
&\quad - \mathbb{E}\bigg[\int_t^{\tau_n} H\Big(s, X^{(t,x),\pi^*}(s), \pi^*(s), \frac{\partial v}{\partial x}\big(s, X^{(t,x),\pi^*}(s)\big), \frac{\partial^2 v}{\partial x^2}\big(s, X^{(t,x),\pi^*}(s)\big)\Big) \\
&\qquad\qquad + \frac{\partial v}{\partial t}\big(s, X^{(t,x),\pi^*}(s)\big)\,ds\bigg] \\
&\overset{(2)}{\geq} \mathbb{E}\big[v\big(\tau_n, X^{(t,x),\pi^*}(\tau_n)\big)\big] + \mathbb{E}\bigg[\int_t^{\tau_n} f\big(s, X^{(t,x),\pi^*}(s), \pi^*(s)\big)\,ds\bigg] \\
&\quad - \mathbb{E}\bigg[\int_t^{\tau_n} \sup_{\pi \in K} H\Big(s, X^{(t,x),\pi^*}(s), \pi, \frac{\partial v}{\partial x}\big(s, X^{(t,x),\pi^*}(s)\big), \frac{\partial^2 v}{\partial x^2}\big(s, X^{(t,x),\pi^*}(s)\big)\Big) \\
&\qquad\qquad + \frac{\partial v}{\partial t}\big(s, X^{(t,x),\pi^*}(s)\big)\,ds\bigg] \\
&\overset{(3)}{\geq} \mathbb{E}\big[v\big(\tau_n, X^{(t,x),\pi^*}(\tau_n)\big)\big] + \mathbb{E}\bigg[\int_t^{\tau_n} f\big(s, X^{(t,x),\pi^*}(s), \pi^*(s)\big)\,ds\bigg].
\end{aligned}$$

where (3) follows from (3.13). As f is bounded by assumption, v satisfies the polynomial growth condition (3.19) and $\big(X^{(t,x),\pi^*}\big)^*(T) \in L^k$ holds, we can apply the dominated convergence theorem to the right-hand side of (3) for $n \to \infty$. Because v satisfies the terminal condition (3.17), the limit corresponds exactly to $J(t,x,\pi^*)$. Hence, the inequalities (2) and (3) above become, in fact, equalities as $n \to \infty$. Therefore, by subtracting the real-valued limit of the right-hand side of (3), i.e. $J(t,x,\pi^*) = v(t,x)$, and applying the monotone convergence theorem to the remaining integrals, which is justified by the fact that (3.13) prevents the integrands from changing sign, we get the following identities:

$$\begin{aligned}
0 &= \mathbb{E}\bigg[\int_t^{T} - \sup_{\pi \in K} H\Big(s, X^{(t,x),\pi^*}(s), \pi, \frac{\partial v}{\partial x}\big(s, X^{(t,x),\pi^*}(s)\big), \frac{\partial^2 v}{\partial x^2}\big(s, X^{(t,x),\pi^*}(s)\big)\Big) \\
&\qquad - \frac{\partial v}{\partial t}\big(s, X^{(t,x),\pi^*}(s)\big)\,ds\bigg], \quad \text{and} \\
0 &= \mathbb{E}\bigg[\int_t^{T} \sup_{\pi \in K} H\Big(s, X^{(t,x),\pi^*}(s), \pi, \frac{\partial v}{\partial x}\big(s, X^{(t,x),\pi^*}(s)\big), \frac{\partial^2 v}{\partial x^2}\big(s, X^{(t,x),\pi^*}(s)\big)\Big) \\
&\qquad - H\Big(s, X^{(t,x),\pi^*}(s), \pi^*(s), \frac{\partial v}{\partial x}\big(s, X^{(t,x),\pi^*}(s)\big), \frac{\partial^2 v}{\partial x^2}\big(s, X^{(t,x),\pi^*}(s)\big)\Big)\,ds\bigg].
\end{aligned}$$

The nonnegativity of both integrands implies (3.33) and (3.34), respectively, which concludes the proof. □

As already discussed in connection with (3.20), also the converse of the last statement is true, i.e. if (3.33) and (3.34) hold for $\pi^* \in \mathcal{A}_M$ then π^* is an optimal control. This can be shown by performing calculations and arguments which are analogous to above. Here, (2) and (3) yield equality by assumption, which shows that equality (1) holds indeed, i.e. π^* is optimal.

We refer to [22] for an alternative proof of the last statement of Lemma 3.20, which is performed in the setting of Remark 3.5. The idea is to use a corollary of the DPP in order to decompose $\overline{Y} := \big(v\big(s, X^{(t,x),\pi^*}(s)\big)\big)_{s\in[t,T]}$ into a martingale of the form $M := E[\xi|\mathcal{F}_\cdot]$ and an integral process containing f. The first part can be written as the sum of a random variable and an integral with respect to our Brownian motion by the martingale representation theorem. On the other hand, one can apply Itô's formula to $\overline{Y}$ as in our proof above. Comparing these two representations and using the HJB equation leads to (3.34), as well. Note that this proof concept relies on imposing growth conditions on f and g (cf. Remark 3.5) and the classical moment estimates for solutions to SDEs, whose coefficients satisfy certain Lipschitz conditions, in order to guarantee that M is indeed a martingale. As shown in [22], v always satisfies a polynomial growth condition in this case. Since the aforementioned results are not applicable to our general setting as introduced in Section 3.1 (cf. Example 3.3), we have to explicitly make assumptions in our formulation of Lemma 3.20 which point in this direction.

Note that the arguments presented in our proof of Lemma 3.20 show in particular that π^* is also optimal in the sense of (3.7). This can be easily seen by applying Itô's formula to $v\big(\tau\wedge\tau_n, X^{(t,x),\pi^*}(\tau\wedge\tau_n)\big)$ for an arbitrary stopping time $\tau:\Omega\to[t,T]$, using (3.33) and (3.34), taking the expectation and removing the localizing sequence as above.

The next result shows by means of Lemma 3.20 that if some technical properties hold (cf. (3.35)), $\overline{Y}$ solves a BSDE whose generator depends on the running gains function f, the control π^* and the corresponding solution to (3.1).

Lemma 3.21. *Suppose that the assumptions made in Lemma 3.20 hold. By π^* we denote again an optimal control for the control problem with initial pair $(t,x)\in[0,T]\times\mathbb{R}$. Moreover, we assume that the technical integrability conditions*

$$\begin{aligned} &\overline{Y} := \big(v\big(s, X^{(t,x),\pi^*}(s)\big)\big)_{s\in[t,T]} \in S^2(t,T;\mathbb{R}) \quad \textit{and} \\ &\overline{Z} := \left(b\big(s, X^{(t,x),\pi^*}(s), \pi^*(s)\big)\frac{\partial v}{\partial x}\big(s, X^{(t,x),\pi^*}(s)\big)\right)_{s\in[t,T]} \in H^2(t,T;\mathbb{R}^m) \end{aligned} \tag{3.35}$$

hold. Then these processes solve the BSDE

$$dY(s) = -f\big(s, X^{(t,x),\pi^*}(s), \pi^*(s)\big)\,ds + Z^\intercal(s)\,dB(s), \quad s\in[t,T], \tag{3.36}$$

with associated terminal condition $Y(T) = g\big(X^{(t,x),\pi^*}(T)\big)$.

Proof. At first, it has to be pointed out that the integrability conditions in (3.35) ensure that $\overline{Y}$ and $\overline{Z}$ satisfy at least the technical requirements for being a solution to a BSDE according to Definition 3.12. Since v is sufficiently smooth by assumption, we can apply Itô's formula to $\overline{Y}$, from which we obtain for $s\in[t,T]$:

$$d\overline{Y}(s) = \left[\frac{\partial v}{\partial t}\big(s, X^{(t,x),\pi^*}(s)\big) + \mathcal{L}^{\pi^*(s)}v\big(s, X^{(t,x),\pi^*}(s)\big)\right]ds + \overline{Z}^\intercal(s)\,dB(s),$$

where we used (3.10) and the definition of $\overline{Z}$. Expanding the finite variation part additively by $\pm f\big(s, X^{(t,x),\pi^*}(s), \pi^*(s)\big)\, ds$ and substituting for the partial derivative with respect to t according to (3.33), which is justified by Lemma 3.20, leads to

$$\begin{aligned} d\overline{Y}(s) = \Bigg[& H\bigg(s, X^{(t,x),\pi^*}(s), \pi^*(s), \frac{\partial v}{\partial x}\big(s, X^{(t,x),\pi^*}(s)\big), \frac{\partial^2 v}{\partial x^2}\big(s, X^{(t,x),\pi^*}(s)\big)\bigg) \\ & - \sup_{\pi \in K} H\bigg(s, X^{(t,x),\pi^*}(s), \pi, \frac{\partial v}{\partial x}\big(s, X^{(t,x),\pi^*}(s)\big), \frac{\partial^2 v}{\partial x^2}\big(s, X^{(t,x),\pi^*}(s)\big)\bigg)\Bigg]\, ds \\ & - f\big(s, X^{(t,x),\pi^*}(s), \pi^*(s)\big)\, ds + \overline{Z}^\intercal(s)\, dB(s), \quad s \in [t,T]. \end{aligned} \tag{3.37}$$

Again by Lemma 3.20, π^* maximizes the Hamiltonian in the sense of (3.34). Hence, the expression within the square brackets in (3.37) vanishes (a.s. for a.e. $s \in [t,T]$), which shows (3.36). Finally, we obtain $\overline{Y}(T) = v\big(T, X^{(t,x),\pi^*}(T)\big) = g\big(X^{(t,x),\pi^*}(T)\big)$ from the definition of $\overline{Y}$, where the last equality is an immediate consequence of v satisfying the terminal condition (3.17). Therefore, the pair $\big(\overline{Y}, \overline{Z}\big)$ indeed solves the above BSDE, which concludes the proof. $\square$

Note that we could again conclude the required integrability properties for $\overline{Y}$ and $\overline{Z}$, if v, $\frac{\partial v}{\partial x}$ and $|b|$ satisfy certain growth conditions and suitable moments of $\big(X^{(t,x),\pi^*}\big)^*(T)$ are finite, whose order should usually be at least twice as high as the order of the respective growth conditions.

The following lemma (cf. [19] and [22] for similar results) states that if we require v to be smoother than in the previous considerations and certain technical integrability conditions hold, we can explicitly describe solutions to the first-order adjoint BSDE (3.29) by means of partial derivatives of v, an optimal control π^* and the associated state process $X^{(t,x),\pi^*}$.

Lemma 3.22. *Suppose that the assumptions from Lemma 3.20 hold, v is even an element of $C^{1,3}([0,T] \times \mathbb{R})$ and $\frac{\partial^2 v}{\partial t \partial x}$ exists and is continuous. Let an optimal control $\pi^* \in \mathcal{A}_M$ for the control problem starting in $(t,x) \in [0,T] \times \mathbb{R}$ be given. If, in addition, the processes*

$$\begin{aligned} \overline{P} &:= \bigg(\frac{\partial v}{\partial x}\big(s, X^{(t,x),\pi^*}(s)\big)\bigg)_{s \in [t,T]} \quad \text{and} \\ \overline{Q} &:= \bigg(b\big(s, X^{(t,x),\pi^*}(s), \pi^*(s)\big)\frac{\partial^2 v}{\partial x^2}\big(s, X^{(t,x),\pi^*}(s)\big)\bigg)_{s \in [t,T]} \end{aligned} \tag{3.38}$$

are elements of $S^2(t,T;\mathbb{R})$ and $H^2(t,T;\mathbb{R}^m)$, respectively, then the pair $\big(\overline{P}, \overline{Q}\big)$ solves the first-order adjoint equation, which is given by (3.29) for a maximization problem.

Proof. At first, we notice that we can apply Itô's formula to $\overline{P}$, since $\frac{\partial v}{\partial x} \in C^{1,2}([0,T] \times \mathbb{R})$ holds by assumption. Hence, we obtain for $s \in [t,T]$:

$$\begin{aligned} d\overline{P}(s) = \bigg[& \frac{\partial^2 v}{\partial t \partial x}\big(s, X^{(t,x),\pi^*}(s)\big) + a\big(s, X^{(t,x),\pi^*}(s), \pi^*(s)\big)\frac{\partial^2 v}{\partial x^2}\big(s, X^{(t,x),\pi^*}(s)\big) \\ & + \frac{1}{2}\big|b\big(s, X^{(t,x),\pi^*}(s), \pi^*(s)\big)\big|^2 \frac{\partial^3 v}{\partial x^3}\big(s, X^{(t,x),\pi^*}(s)\big)\bigg]\, ds + \overline{Q}^\intercal(s)\, dB(s), \end{aligned} \tag{3.39}$$

where we used the definition of $\overline{Q}$. Therefore, it remains to show that the expression within the square brackets can indeed be described by means of $-\frac{\partial \mathcal{H}}{\partial x}$.
By using (3.12), (3.33) and (3.34), as shown in Lemma 3.20, pointwise for $\big(s, X^{(t,x),\pi^*}(s)\big)$, we obtain for every $y \in \mathbb{R}$ the following result almost surely for almost every $s \in [t,T]$:

$$\begin{aligned} 0 &= \frac{\partial v}{\partial t}\big(s, X^{(t,x),\pi^*}(s)\big) + H\Big(s, X^{(t,x),\pi^*}(s), \pi^*(s), \frac{\partial v}{\partial x}\big(s, X^{(t,x),\pi^*}(s)\big), \frac{\partial^2 v}{\partial x^2}\big(s, X^{(t,x),\pi^*}(s)\big)\Big) \\ &\geq \frac{\partial v}{\partial t}(s,y) + H\Big(s, y, \pi^*(s), \frac{\partial v}{\partial x}(s,y), \frac{\partial^2 v}{\partial x^2}(s,y)\Big). \end{aligned}$$

By viewing the right-hand side of the above inequality for fixed $s \in [t,T]$ and $\omega \in \Omega$ (except for some null sets) as a function of y, we can conclude that $X^{(t,x),\pi^*}(s)$ is almost surely a local maximum in the above sense, because $\mathbb{R}$ is an open set. These functions are in particular differentiable with respect to y due to our differentiability assumptions above (including Assumption 3.18). Hence, $X^{(t,x),\pi^*}(s)$ is a critical point in the above meaning. This observation together with an application of the chain and the product rule leads to

$$\begin{aligned} 0 =& \frac{\partial}{\partial x}\left(\frac{\partial v}{\partial t}(s,y) + H\Big(s, y, \pi^*(s), \frac{\partial v}{\partial x}(s,y), \frac{\partial^2 v}{\partial x^2}(s,y)\Big)\right)\bigg|_{y=X^{(t,x),\pi^*}(s)} \\ =& \frac{\partial^2 v}{\partial t \partial x}\big(s, X^{(t,x),\pi^*}(s)\big) + a\big(s, X^{(t,x),\pi^*}(s), \pi^*(s)\big)\, \frac{\partial^2 v}{\partial x^2}\big(s, X^{(t,x),\pi^*}(s)\big) \\ &+ \frac{1}{2}\big|b\big(s, X^{(t,x),\pi^*}(s), \pi^*(s)\big)\big|^2\, \frac{\partial^3 v}{\partial x^3}\big(s, X^{(t,x),\pi^*}(s)\big) \\ &+ \frac{\partial a}{\partial x}\big(s, X^{(t,x),\pi^*}(s), \pi^*(s)\big)\, \frac{\partial v}{\partial x}\big(s, X^{(t,x),\pi^*}(s)\big) + \frac{\partial f}{\partial x}\big(s, X^{(t,x),\pi^*}(s), \pi^*(s)\big) \\ &+ \left(\frac{\partial b}{\partial x}\right)^{\intercal}\big(s, X^{(t,x),\pi^*}(s), \pi^*(s)\big)\, b\big(s, X^{(t,x),\pi^*}(s), \pi^*(s)\big)\, \frac{\partial^2 v}{\partial x^2}\big(s, X^{(t,x),\pi^*}(s)\big), \end{aligned} \tag{3.40}$$

where $\frac{\partial b}{\partial x}$ has to be understood componentwise and $\frac{\partial}{\partial x}$ in the first line corresponds to differentiating with respect to y, i.e. the second component, with a slight abuse of notation. (Note that x is already assigned to the fixed initial value of the state process.) The terms in the last two rows of (3.40) correspond exactly to

$$\frac{\partial \mathcal{H}}{\partial x}\big(s, X^{(t,x),\pi^*}(s), \pi^*(s), \overline{P}(s), \overline{Q}(s)\big). \tag{3.41}$$

Since (3.40) holds almost surely for almost every $s \in [t,T]$, we obtain the following result from inserting (3.40) and (3.41) into (3.39):

$$d\overline{P}(s) = -\frac{\partial \mathcal{H}}{\partial x}\big(s, X^{(t,x),\pi^*}(s), \pi^*(s), \overline{P}(s), \overline{Q}(s)\big)\, ds + \overline{Q}^{\intercal}(s)\, dB(s), \quad s \in [t,T],$$

i.e. the pair $\big(\overline{P}, \overline{Q}\big)$ satisfies (3.29). Finally, the terminal condition described in Remark 3.17 is satisfied as well, since $v(T,\cdot) = g$ holds and we, therefore, obtain from the definition of the process $\overline{P}$:

$$\overline{P}(T) = \frac{\partial g}{\partial x}\big(X^{(t,x),\pi^*}(T)\big),$$

which completes the proof, as $\big(\overline{P}, \overline{Q}\big) \in S^2(t,T;\mathbb{R}) \times H^2(t,T;\mathbb{R}^m)$ holds by assumption. □

The following theorem combines the results of the previous three lemmata. It forms the mathematical foundation for the deep controlled 2BSDE algorithm, which shall be formulated in Section 3.4 below.

Theorem 3.23. *Let a maximization problem in the setting of Section 3.1 be given with value function v and suppose that Assumption 3.18 holds, $v \in C^{1,3}([0,T] \times \mathbb{R})$ and the partial derivative $\frac{\partial^2 v}{\partial t \partial x}$ exists and is even continuous.*
Fix $(t,x) \in [0,T] \times \mathbb{R}$. Assume that there exists an optimal control $\pi^* \in \mathcal{A}_M$ for the control problem starting in (t,x). As usual, we denote by $X^{(t,x),\pi^*}$ the corresponding solution to (3.1). Moreover, if v satisfies a polynomial growth condition, cf. (3.19), of order $k \in \mathbb{N}$, $\big(X^{(t,x),\pi^*}\big)^*(T) \in L^k$ holds and the technical integrability conditions (3.35) and (3.38) are satisfied, then the processes

$$\begin{aligned} &X^* := \big(X^{(t,x),\pi^*}(s)\big)_{s\in[t,T]}, \quad Y^* := \big(v\big(s, X^{(t,x),\pi^*}(s)\big)\big)_{s\in[t,T]}, \\ &Z^* := \left(\frac{\partial v}{\partial x}\big(s, X^{(t,x),\pi^*}(s)\big)\right)_{s\in[t,T]} \quad \text{and} \quad Q^* := \left(\frac{\partial^2 v}{\partial x^2}\big(s, X^{(t,x),\pi^*}(s)\big)\right)_{s\in[t,T]} \end{aligned} \tag{3.42}$$

solve the following coupled system consisting of one forward and two backward SDEs: For $s \in [t,T]$, it is defined via

$$\begin{aligned} dX(s) &= a\big(s, X(s), \pi^*(s)\big)\, ds + b^\intercal\big(s, X(s), \pi^*(s)\big)\, dB(s), \\ dY(s) &= -\, f\big(s, X(s), \pi^*(s)\big)\, ds + b^\intercal\big(s, X(s), \pi^*(s)\big)\, Z(s)\, dB(s), \\ dZ(s) &= -\,\frac{\partial \mathcal{H}}{\partial x}\Big(s, X(s), \pi^*(s), Z(s), b\big(s, X(s), \pi^*(s)\big)\, Q(s)\Big)\, ds \\ &\qquad + b^\intercal\big(s, X(s), \pi^*(s)\big)\, Q(s)\, dB(s), \end{aligned} \tag{3.43}$$

with initial condition $X(t) = x$ and terminal conditions $Y(T) = g(X(T))$ and $Z(T) = \frac{\partial g}{\partial x}(X(T))$. Moreover, π^* maximizes the Hamiltonian in the following sense almost surely for almost every $s \in [t,T]$:

$$H\big(s, X^*(s), \pi^*(s), Z^*(s), Q^*(s)\big) = \sup_{\pi\in K} H\big(s, X^*(s), \pi, Z^*(s), Q^*(s)\big). \tag{3.44}$$

Proof. This follows immediately from (3.1), Lemmata 3.20, 3.21 and 3.22 and observing that, using the terminology of Lemmata 3.21 and 3.22, $\overline{Z}(s) = b\big(s, X^{(t,x),\pi^*}(s), \pi^*(s)\big)\, \overline{P}(s)$ holds for $s \in [t,T]$. □

Note that Theorem 3.23 can also be interpreted as a “maximum principle” as the statement has a structure similar to Theorem 3.16: Roughly speaking, if an optimal control exists, it necessarily maximizes a Hamiltonian-type function, namely H, where the other arguments of H are given by a solution to a system of forward and backward SDEs. A solution can even be stated explicitly as the dynamic programming approach is applicable thanks to the smoothness assumptions on v. Clearly, this is a decisive difference compared to the statement of Theorem 3.16, where an abstract existence result is formulated.

Remark 3.24. Note that Theorem 3.23 is formulated explicitly for maximization problems. However, we can easily state the corresponding result for minimization problems in the setting of Section 3.1: Again, v denotes the value function. As discussed in connection with (3.5), $-v$ corresponds to the value function of a maximization problem with running gains function $-f$ and terminal gain function $-g$. Hence, we can apply Theorem 3.23 to this problem, if the prerequisites are met. Except for the different notion of optimality, i.e. sup vs. inf, the assumptions above are independent of the sign. By using the fact that $\sup(-F) = -\inf F$ holds for an arbitrary function F and the definition of the classical Hamiltonian, we can conclude that, in this case, (3.44) becomes

$$H\big(s, X^*(s), \pi^*(s), Z^*(s), Q^*(s)\big) = \inf_{\pi \in K} H\big(s, X^*(s), \pi, Z^*(s), Q^*(s)\big), \tag{3.45}$$

where H, Z^* and Q^* are defined with respect to v, f and g as in Theorem 3.23. The remaining results of Theorem 3.23 hold equally for the minimization problem, as the consideration of $-v$, $-f$ and $-g$ shows: By plugging the explicit solutions (with $-v$) into (3.43) (with $-f$ and $-g$), i.e. the system for the previously constructed maximization problem, and multiplying by -1, we obtain exactly (3.43) in its original form. Clearly, this also works for the terminal conditions. Hence, except for replacing (3.44) with (3.45), Theorem 3.23 holds equally for minimization problems.

Remark 3.25. The arguments presented in this chapter are given under the premise that the prerequisites, e.g. the sufficient smoothness of the value function, are met on the entire space $[0,T]\times\mathbb{R}$, for simplicity. However, these requirements can insofar be weakened as they only have to be satisfied on a set $[0,T]\times O$, where O is an open set which contains the image of every possible state process. For example, the state processes in the utility maximization problem and its dual problem can only attain positive values due to $x_0, y \in \mathbb{R}^+$ and the structures of their SDEs. Hence, we are only interested in the value function on $[0,T]\times\mathbb{R}^+$ with the above requirements adapted accordingly.

For notational convenience, all of the above results are presented and proved for one-dimensional state processes, as this is precisely the case for the utility maximization problem and its dual problem. However, the dynamic programming approach obviously also works for problems with d-dimensional state processes which leads to a generalized version of Theorem 3.23. This is an important observation with regard to our numerical experiments in Chapter 5, as it allows us to apply the deep controlled 2BSDE algorithm also to problems with random coefficients which satisfy their own SDEs. Even though those problems are per se non-Markovian, they can become Markovian, if the wealth process is expanded by these coefficient processes leading to a multi-dimensional state process. We refer to Section 5.3 for details.

In this setup, we consider an SDE of type (3.1) with deterministic, measurable functions $a : [0,T]\times\mathbb{R}^d\times K \to \mathbb{R}^d$ and $b : [0,T]\times\mathbb{R}^d\times K \to \mathbb{R}^{m\times d}$. Let $f : [0,T]\times\mathbb{R}^d\times K \to \mathbb{R}$ and $g : \mathbb{R}^d \to \mathbb{R}$ be our gain functions. Hence, the value function can be defined on $[0,T]\times\mathbb{R}^d$ similarly to (3.4). Motivated by Itô's formula, the classical Hamiltonian for this higher-dimensional setup is defined via

$$H(t,x,\pi,y,z) := a^\intercal(t,x,\pi)\,y + \frac{1}{2}\operatorname{tr}\big(b^\intercal(t,x,\pi)\,b(t,x,\pi)\,z\big) + f(t,x,\pi), \tag{3.46}$$

for every $(t, x, \pi, y, z) \in [0,T] \times \mathbb{R}^d \times K \times \mathbb{R}^d \times \mathbb{R}^{d\times d}$. Moreover, the generalized Hamiltonian is defined by

$$\mathcal{H}(t, x, \pi, y, z) := a^{\intercal}(t, x, \pi)\, y + \operatorname{tr}\big(b(t, x, \pi)\, z\big) + f(t, x, \pi), \tag{3.47}$$

for every $(t, x, \pi, y, z) \in [0,T] \times \mathbb{R}^d \times K \times \mathbb{R}^d \times \mathbb{R}^{d\times m}$. It follows under analogous conditions that if an optimal control for the problem starting in $(t,x) \in [0,T] \times \mathbb{R}^d$ exists, then one can find processes which solve the system

$$\begin{aligned} dX(s) &= a\big(s, X(s), \pi^*(s)\big)\, ds + b^{\intercal}\big(s, X(s), \pi^*(s)\big)\, dB(s), \\ dY(s) &= -\, f\big(s, X(s), \pi^*(s)\big)\, ds + Z^{\intercal}(s)\, b^{\intercal}\big(s, X(s), \pi^*(s)\big)\, dB(s), \\ dZ(s) &= -\, \nabla_x \mathcal{H}\big(s, X(s), \pi^*(s), Z(s), Q(s)\, b^{\intercal}\big(s, X(s), \pi^*(s)\big)\big)\, ds \\ &\qquad + Q(s)\, b^{\intercal}\big(s, X(s), \pi^*(s)\big)\, dB(s), \end{aligned} \tag{3.48}$$

for $s \in [t,T]$, with initial condition $X(t) = x$ and terminal conditions $Y(T) = g(X(T))$ and $Z(T) = \nabla_x g(X(T))$. As in Theorem 3.23, these processes can even be stated explicitly. In contrast to (3.42), the partial derivative operators are replaced by ∇_x and the Hessian operator with respect to x, i.e. H_x. Moreover, the Hamiltonian maximization condition (3.44) also holds in this case. The proof can be carried out by means of the same arguments as above and the multidimensional version of Itô's formula. Clearly, this result is equivalent to Theorem 3.23, if $d = 1$ holds.

3.4 Formulation of the Deep Controlled 2BSDE Algorithm

The deep controlled 2BSDE algorithm, which was originally formulated in [5], is essentially an extension of the algorithm proposed in [1] for solving second-order BSDEs. As the optimal control is in many cases a priori not known, one has to take the optimization with respect to the control space into consideration as well. As we shall see below, this is achieved by integrating the Hamiltonian maximization condition (3.44) from Theorem 3.23 (or its generalization, if $d > 1$) into the algorithm.

At first, we determine an equidistant time discretization $(t_i)_{i\in\{0,\dots,N\}}$ of $[0,T]$ with step size $\Delta t := T/N$, where $N \in \mathbb{N}$ is fixed. The processes X, Y and Z are simulated by means of a discrete-time forward scheme which is based on the Euler-Maruyama method. This leads to processes $(X_i)_{i\in\{0,\dots,N\}}$, $(Y_i)_{i\in\{0,\dots,N\}}$ and $(Z_i)_{i\in\{0,\dots,N\}}$ (cf. (3.51) below). Obviously, applying this method requires the knowledge of not necessarily optimal processes π and Q, as well. For every $i \in \{0, \dots, N-1\}$, we model $\pi(t_i)$ and $Q(t_i)$ as

$$\pi_i := \mathcal{N}_{\theta_{i,\pi}}(X_i) \quad \text{and} \quad Q_i := \mathcal{N}_{\theta_{i,Q}}(X_i), \tag{3.49}$$

respectively, where the neural networks $\mathcal{N}_{\theta_{i,\pi}} : \mathbb{R}^d \to \mathbb{R}^m$ and $\mathcal{N}_{\theta_{i,Q}} : \mathbb{R}^d \to \mathbb{R}^{d\times d}$ are parameterized by appropriate vectors $\theta_{i,\pi}$ and $\theta_{i,Q}$. We ensure that every π_i is a K-valued random variable by applying a surjective, almost everywhere differentiable function mapping to K to the outputs of the final dense layers of the neural networks $\mathcal{N}_{\theta_{i,\pi}}$. We recall that the set K is precisely given by $\widetilde{K}$ for the dual problem. (3.49) implies that we are only interested in finding a Markovian control. This simplification is not overly restrictive as a result from [18] shows, which states (under additional assumptions) that an investor cannot perform better by using more general progressively measurable control processes. Moreover, the process Q^* from Theorem 3.23 (and similarly its generalization for $d > 1$) suggests that it is plausible to model $Q(t_i)$ as a function of X_i.

Since we do not know the initial values of Y and Z, we model them by means of variables y_0 and z_0 which have to be optimized. Hence, the scheme starts with

$$X_0 := x_0, \quad Y_0 := y_0 \quad \text{and} \quad Z_0 := z_0. \tag{3.50}$$

Let $(\Delta B_i)_{i\in\{0,\dots,N-1\}}$ be a family of i.i.d. m-dimensional, centered normally distributed random vectors with covariance matrix $\Delta t \cdot I_m$. For every $i \in \{0, \dots, N-1\}$, we can then define inductively:

$$\begin{aligned} X_{i+1} :=& X_i + a(t_i, X_i, \pi_i)\,\Delta t + b^\intercal(t_i, X_i, \pi_i)\,\Delta B_i, \\ Y_{i+1} :=& Y_i - f(t_i, X_i, \pi_i)\,\Delta t + Z_i^\intercal\, b^\intercal(t_i, X_i, \pi_i)\,\Delta B_i, \\ Z_{i+1} :=& Z_i - \nabla\mathcal{H}_x\big(t_i, X_i, \pi_i, Z_i, Q_i\, b^\intercal(t_i, X_i, \pi_i)\big)\,\Delta t + Q_i\, b^\intercal(t_i, X_i, \pi_i)\,\Delta B_i, \end{aligned} \tag{3.51}$$

which corresponds to a discretized version of (3.48). As the processes Y and Z have to satisfy certain terminal conditions according to Theorem 3.23 and its generalization, we have to ensure that the parameters which are not used for the control, i.e. y_0, z_0 and $\theta_{i,Q}$ for $i \in \{0, \dots, N-1\}$, minimize the corresponding squared L^2-error

$$\mathcal{L}_{2BSDE}(y_0, z_0, \theta_{0,Q}, \dots, \theta_{N-1,Q}) := \mathbb{E}\Big[\big|Y_N - g(X_N)\big|^2 + \big|Z_N - \nabla_x g(X_N)\big|^2\Big]. \tag{3.52}$$

Furthermore, we are required to maximize the Hamiltonian (cf. (3.44)), i.e. we intend to minimize for every $i \in \{0, \dots, N-1\}$:

$$\mathcal{L}^i_{control}(\theta_{i,\pi}) := -\mathbb{E}\Big[H\big(t_i, X_i, \pi_i, Z_i, Q_i\big)\Big], \tag{3.53}$$

where we use the expectation operator for simplicity as we consider only a finite number of trajectories in each optimization step and the algorithm almost surely works with a trajectory at most once (cf. Algorithm 1). If the studied problem is a minimization problem, then we have to remove the negative sign on the right-hand side of (3.53) (cf. Remark 3.24).

In the case of the dual problem, the above procedure only solves the part problem (3.5) for a fixed parameter value $y \in \mathbb{R}^+$ which replaces x_0 in (3.50) above. However, in the spirit of (3.6), we can likewise model the initial value of $(X_i)_{i\in\{0,\dots,N\}}$ as a trainable variable which needs to be optimized with respect to the loss function

$$\mathcal{L}_{dual}(y) := \mathbb{E}\big[\widetilde{U}(X_N)\big] + x_0 y. \tag{3.54}$$

Note that X_N implicitly depends on the parameter y.

After randomly initializing the parameters, the deep controlled 2BSDE algorithm generates a fixed number of realizations of $(\Delta B_i)_{i\in\{0,\dots,N-1\}}$, i.e. a batch of size b_{size}, for every training step and updates the parameters with respect to the above loss functions, where the expectation operator is replaced by the sample mean. For each loss function, the updates are performed by means of one step of a stochastic gradient descent algorithm. In the following pseudocode (cf. Algorithm 1), we summarize the structure of the algorithm by describing a full training step. This procedure is supposed to be repeated until the parameters seem to have converged. Note that we already use the updated parameters from previous substeps for the optimization substeps in connection with (3.53) and (3.54), respectively. This should facilitate convergence. We close this section with a remark on the network architectures of $\mathcal{N}_{\theta_{0,\pi}}$ and $\mathcal{N}_{\theta_{0,Q}}$, respectively.

Remark 3.26. By the definition of our filtration, random variables which are measurable with respect to $\mathcal{F}_0$ have to be almost surely constant. Hence, it is sufficient to model $\pi(0)$ and $Q(0)$ by means of trivial neural networks which merely consist of a bias vector. This becomes even clearer, if we combine the ansatz (3.49) with X_0 being deterministic. Hence, a more complex network architecture would not improve the result while still increasing the dimensionality of the problem.

Algorithm 1 One training step of the deep controlled 2BSDE algorithm

```
1: Generate $b_{size}$ realizations of $(\Delta B_i)_{i\in\{0,\dots,N-1\}}$;
2: // Substep 1: Minimizing $\mathcal{L}_{2BSDE}$
3: Initialize according to (3.50) for every $j \in \{1,\dots,b_{size}\}$ (dual: $X_0^j = y$);
4: for $i = 0,1,\dots,N-1$ do
5:     for $j = 1,2,\dots,b_{size}$ do
6:         Calculate $\pi_i^j$ and $Q_i^j$ by means of (3.49);
7:         Use (3.51) in order to obtain $X_{i+1}^j$, $Y_{i+1}^j$ and $Z_{i+1}^j$;
8:     end for
9: end for
10: $loss1 \leftarrow \frac{1}{b_{size}} \sum_{j=1}^{b_{size}} \left( \left|Y_N^j - g(X_N^j)\right|^2 + \left|Z_N^j - \nabla_x g(X_N^j)\right|^2 \right)$;
11: Update $y_0, z_0, \theta_{0,Q}, \dots, \theta_{N-1,Q}$ with one step of an SGD algorithm w.r.t. $loss1$;
12: // Substep 2: Minimizing $\mathcal{L}_{control}^i$ for every $i \in \{0,\dots,N-1\}$
13: for $i = 0,1,\dots,N-1$ do
14:     for $j = 1,2,\dots,b_{size}$ do
15:         if i==0 then
16:             Initialize $X_0^j$ (dual: $X_0^j = y$), $Y_0^j$ and $Z_0^j$ according to (3.50);
17:         else
18:             Use (3.51) in order to obtain $X_i^j$, $Y_i^j$ and $Z_i^j$;
19:         end if
20:         Calculate $\pi_i^j$ and $Q_i^j$ by means of (3.49);
21:     end for
22:     if problemtype == max then
23:         $loss2_i \leftarrow -\frac{1}{b_{size}} \sum_{j=1}^{b_{size}} H\left(t_i, X_i^j, \pi_i^j, Z_i^j, Q_i^j\right)$;
24:     else
25:         $loss2_i \leftarrow \frac{1}{b_{size}} \sum_{j=1}^{b_{size}} H\left(t_i, X_i^j, \pi_i^j, Z_i^j, Q_i^j\right)$;
26:     end if
27:     Update $\theta_{i,\pi}$ with one step of an SGD algorithm w.r.t. $loss2_i$;
28: end for
29: // Substep 3: Minimizing $\mathcal{L}_{dual}$
30: if problem == dual then
31:     Initialize $X_0^j$ with $y$ for every $j \in \{1,\dots,b_{size}\}$;
32:     for $i = 0,1,\dots,N-1$ do
33:         for $j = 1,2,\dots,b_{size}$ do
34:             Calculate $\pi_i^j$ by means of (3.49);
35:             Use (3.51) in order to obtain $X_{i+1}^j$;
36:         end for
37:     end for
38:     $loss3 \leftarrow \frac{1}{b_{size}} \sum_{j=1}^{b_{size}} \widetilde{U}(X_N^j) + x_0 y$;
39:     Update $y$ with one step of an SGD algorithm w.r.t. $loss3$;
40: end if
```

4 The Deep SMP and the Deep Primal SMP Algorithm in a Non-Markovian Setting

In this chapter, we consider the utility maximization problem and its dual problem, as introduced in Chapter 2, in its full generality. Hence, in contrast to the setting of Chapter 3, r, μ and σ are allowed to be random here, i.e. dependent on ω. This implies that the dynamic programming approach is not necessarily applicable, as already mentioned in Chapter 3. However, considering whether we can prove stochastic maximum principles for both problems might be worthwhile as there are several such results in the literature which also work for random coefficients. For example, [3] finds an SMP for problems with random coefficients, where the drift and the diffusion coefficient of the state process X^π are affine in π and X^π. Unfortunately, this does not hold for our problem, as (2.3) shows.

The aim of the following considerations is proving stochastic maximum principles for the utility maximization problem and its dual problem, respectively, where a simplified version of Theorem 3.16 serves as the formal starting point. In [22] it is argued that the purpose of the adjustment of $\widetilde{\mathcal{H}}$ ($= \mathcal{H}$, since $f \equiv 0$) in the maximum condition (3.27) precisely is ensuring that the adjusted function is concave in π. Moreover, it is pointed out that if $\mathcal{H}$ is already concave in π, then the second-order adjoint equation (3.26) is superfluous. Clearly, this is the case for the utility maximization problem and its dual problem as (3.31), (3.32) and the concavity of $-\delta_K$ show. Hence, we intend to prove for both problems a result similar to Theorem 3.16 with $P_2 \equiv 0$ while also allowing random coefficients. (3.27) will, therefore, correspond to maximizing $\mathcal{H}$ with respect to the control space. Furthermore, we will see that also the reverse implication holds. We refer to Theorems 4.6 and 4.12 and Remarks 4.9 and 4.15 below for details. Note that we are going to weaken Definition 3.12 in the following insofar as we require from a potential solution (Y, Z) to a BSDE only that the right-hand side of (3.23) is well-defined. This is in line with [1, 6, 8, 21]. Alternatively, we could simply require the necessary integrability conditions as in Theorem 3.23.
Moreover, it follows from (3.31) and (3.32) that the corresponding first-order adjoint equations are for $t \in [0, T]$ given by

$$dp_1(t) = -\left[\big(r(t) + \pi^\intercal(t)\,\sigma(t)\,\theta(t)\big)\,p_1(t) + \pi^\intercal(t)\,\sigma(t)\,q_1(t)\right]dt + q_1^\intercal(t)\,dB(t), \tag{4.1}$$

with associated terminal condition $p_1(T) = -U'\big(X^\pi(T)\big)$ for the utility maximization problem and

$$dp_2(t) = \left[\big(r(t) + \delta_K(v(t))\big)\,p_2(t) + \big(\theta(t) + \sigma^{-1}(t)\,v(t)\big)^\intercal q_2(t)\right]dt + q_2^\intercal(t)\,dB(t), \tag{4.2}$$

with terminal condition $p_2(T) = -\widetilde{U}'\big(Y^{(y,v)}(T)\big)$ for the accompanying dual problem.

As motivated above, we begin this chapter by proving a stochastic maximum principle for

the dual problem in great detail. As we shall see below, Section 4.1 will pave the way for the formulation of the deep SMP algorithm in Section 4.4. Moreover, we derive a similar result for the primal problem in Section 4.2, namely Theorem 4.12. This result can be used on the one hand for formulating an algorithm made-to-measure for the primal problem (cf. Section 4.5) and on the other hand, as done in Section 4.3, for proving a relationship with the solution to the corresponding dual problem. This justifies applying the deep SMP algorithm also for solving the primal problem.
The first three sections are based on the results found in [16], which partially rely on arguments presented in [15] and [21], respectively. In the following, we place great emphasis on carrying out the proofs in great detail and removing mistakes from the presentation in [16]. In the case of Theorem 4.12, this leads to a formulation which differs fundamentally from the original result. We refer to Remarks 4.7, 4.8 and 4.13 for details and other major changes. The deep SMP algorithm was originally formulated in [5].

4.1 A Stochastic Maximum Principle for the Dual Problem

At first, we derive the aforementioned result for the dual problem as it is the key to the deep SMP algorithm. We are going to need the following technical assumption:

Assumption 4.1. Suppose that $\widetilde{U}\big(Y^{(y,v)}(T)\big) \in L^2$ holds for any admissible pair $(y, v) \in \mathcal{D}$, where $Y^{(y,v)}$ denotes the solution to the corresponding SDE (2.12), as usual.

Remark 4.2. The motivation for formulating Assumption 4.1 as above is a result from [15] which guarantees that there exists an optimal pair (y^*, v^*) for the dual problem, if, in addition, Assumption 2.2, $U(0) > -\infty$, $U(\infty) = \infty$ and $V \in \mathbb{R}$ (cf. (2.5)) hold. Alternatively, we could directly require the integrability condition which is needed for the second part of Lemma 4.3, as it is done for Lemma 4.11 in Section 4.2.

The first auxiliary result ensures that, in the setting of Theorem 4.6, there always exists a pair solving (4.2) and satisfying the prerequisites. Hence, it is guaranteed that Theorem 4.6 does not correspond to a statement on the empty set.

Lemma 4.3. *Consider a fixed pair $(y, v) \in \mathcal{D}$ and suppose that Assumptions 2.2 and 4.1 and either, using the notation of Lemma 2.7, $U(0) > -\infty$ or $U = \log$ hold. Then $Y^{(y,v)}(T)\widetilde{U}'\big(Y^{(y,v)}(T)\big) \in L^2$ and there exists a solution (p_2, q_2) to (4.2) such that $p_2 Y^{(y,v)}$ is a martingale.*

Proof. Following the argument presented in [16], we obtain for $z \in \mathbb{R}^+$ and the constants β, γ from Lemma 2.8 *(iii)*:

$$\begin{aligned}\widetilde{U}(z) - \widetilde{U}(\infty) &\geq \widetilde{U}(z) - \widetilde{U}(\beta^{-1}z) = -\int_z^{\beta^{-1}z} \widetilde{U}'(s)\,ds \geq -(\beta^{-1}z - z)\widetilde{U}'(\beta^{-1}z) \\ &\geq -\frac{1-\beta}{\beta\gamma} z\widetilde{U}'(z),\end{aligned} \tag{4.3}$$

where we used the monotonicity of $\widetilde{U}$ and $\widetilde{U}'$ from Lemma 2.7 *(ii)* in the first line. The last inequality follows from Lemma 2.8 *(iii)* and $-\widetilde{U} = I$ according to Lemma 2.7 *(iv)*. As the

right-hand side of (4.3) is nonnegative, the relation remains valid after squaring each side. Hence, the proof of the first claim under the first set of assumptions can be completed by plugging $Y^{(y,v)}(T)$ into the squared version of (4.3) and using $\widetilde{U}(\infty) = U(0)$ (cf. Lemma 2.7 *(vi)*) and Assumption 4.1. If $U = \log$, then $\widetilde{U}$ is given by $-\log -1$ according to Example 2.9. Hence, we obtain $Y^{(y,v)}(T)\,\widetilde{U}'\big(Y^{(y,v)}(T)\big) \equiv -1$, which, therefore, lies in L^2.
Due to our previous considerations, $M := \big(\mathbb{E}\big[-Y^{(y,v)}(T)\,\widetilde{U}'\big(Y^{(y,v)}(T)\big)\big|\mathcal{F}_t\big]\big)_{t\in[0,T]}$ defines a square-integrable martingale. We obtain from the martingale representation theorem that there exists a continuous version of this martingale which can be written as $V := M_0 + W \bullet B$, where $W \in H^2(0,T;\mathbb{R}^m)$ is unique. Hence, (V,W) is the unique solution to

$$dV(t) = W^\intercal(t)\,dB(t), \quad t \in [0,T], \quad V(T) = -Y^{(y,v)}(T)\,\widetilde{U}'\big(Y^{(y,v)}(T)\big).$$

Obviously, $V/Y^{(y,v)}$ satisfies the terminal condition of the first-order adjoint equation. Therefore, it remains to show that we can choose q_2 such that (4.2) holds as well. Since $Y^{(y,v)}$ is strictly positive, an application of Itô's formula to $p_2 := V/Y^{(y,v)}$ leads to

$$\begin{aligned} dp_2(t) = \Big[&p_2(t)\big(r(t) + \delta_K(v(t))\big) + p_2(t)\big|\theta(t) + \sigma^{-1}(t)v(t)\big|^2 \\ &+ \big(\theta(t) + \sigma^{-1}(t)v(t)\big)^\intercal \frac{W(t)}{Y^{(y,v)}(t)}\Big]\,dt + \Big[p_2(t)\big(\theta(t) + \sigma^{-1}(t)v(t)\big)^\intercal + \frac{W^\intercal(t)}{Y^{(y,v)}(t)}\Big]\,dB(t), \end{aligned}$$

$t \in [0,T]$. Hence, defining the process q_2 for $t \in [0,T]$ via

$$q_2(t) := p_2(t)\big(\theta(t) + \sigma^{-1}(t)v(t)\big) + \frac{W(t)}{Y^{(y,v)}(t)},$$

reduces the equation above exactly to (4.2), i.e. (p_2, q_2) solves the first-order adjoint equation. By construction, we have $p_2 Y^{(y,v)} = V$, which is a martingale. □

Furthermore, we are going to need an easy, albeit important, result on the concavity/convexity of the composition of concave and convex functions, respectively, if certain monotonicity properties hold.

Lemma 4.4. *Let $A, B \subseteq \mathbb{R}$ be convex sets and consider a convex, nonincreasing function $f : B \to \mathbb{R}$ and a concave function $g : A \to \mathbb{R}$, such that $g(A) \subseteq B$ holds. Then $f \circ g$ is a convex function on A. Moreover, if f is nondecreasing and concave and g is again concave, then $f \circ g$ is concave.*

Proof. Fix arbitrary numbers $a_1, a_2 \in A$ and $\lambda \in [0,1]$. By using the concavity of g and the monotonicity of f in (1) and the convexity of f in (2) we obtain

$$f\big(g\big(\lambda a_1 + (1-\lambda)a_2\big)\big) \overset{(1)}{\leq} f\big(\lambda g(a_1) + (1-\lambda)g(a_2)\big) \overset{(2)}{\leq} \lambda f(g(a_1)) + (1-\lambda)f(g(a_2)).$$

The second claim follows as the relations (1) and (2) are precisely reversed in this case. □

If f and g were supposed to be twice differentiable, the above result would follow immediately from differentiating. The next lemma shows that for a given progressively measurable process Y there exists another progressively measurable process Z such that Z and $\delta_K(Z)$ are bounded and $Z(t)$ attaining 0 characterizes $Y(t)$ being K-valued. Since we have $0 \in K$ by assumption, we obtain in particular that δ_K is bounded from below (by 0). Hence, the results presented in [11] are valid for our setting.

Lemma 4.5. *Let a closed, convex set $K \subseteq \mathbb{R}^m$ containing 0 be given and define δ_K as in (2.9). Consider a progressively measurable process $Y : \Omega \times [0,T] \to \mathbb{R}^m$. Then there exists a progressively measurable process $Z : \Omega \times [0,T] \to \mathbb{R}^m$ such that it holds almost surely for every $t \in [0,T]$:*

$$\begin{aligned} &\big|Z(t)\big| \le 1 \quad \text{and} \quad \big|\delta_K(Z(t))\big| \le 1, \\ &Y(t) \in K \iff Z(t) = 0 \quad \text{and} \\ &Y(t) \notin K \iff \delta_K(Z(t)) + Y^\intercal(t) Z(t) < 0. \end{aligned}$$

Proof. See Lemma 4.2 in Chapter 5 of [11]. □

Note that $Z(t) = 0$ implies $\delta_K(Z(t)) + Y^\intercal(t)Z(t) = 0$. In the following, we summarize the idea of the proof: The proof relies essentially on

$$y \in K \iff \big(\forall x \in \widetilde{K} : \ \delta_K(x) + y^\intercal x \ge 0\big), \tag{4.4}$$

where $\widetilde{K} := \{x \in \mathbb{R}^m \mid \delta_K(x) < \infty\}$, which is an immediate consequence of $-K$ corresponding exactly to the intersection of all closed half-spaces containing $-K$ (cf. [20]). Consider an exhaustion $(\widetilde{K}_n)_{n\in\mathbb{N}}$ of $\widetilde{K}$ by compact subsets. The crucial part of the proof is an application of a version of the Dubins-Savage measurable selection theorem for lower semi-continuous functions (cf. [2]) in order to obtain for all $n \in \mathbb{N}$ a Borel-measurable selection function $\varphi_n : \mathbb{R}^m \to \widetilde{K}_n$ such that $x_y^* := \varphi_n(y)$ minimizes $\delta_K(x) + y^\intercal x$ in $x \in \widetilde{K}_n$ for every $y \in \mathbb{R}^m$. Clearly, if $y \in K$ holds, then $\delta_K(\varphi_n(y)) + y^\intercal \varphi_n(y)$ is nonnegative for every $n \in \mathbb{N}$ according to (4.4), whereas, also by (4.4), there exists $n(y) \in \mathbb{N}$ such that this expression becomes negative, if $y \notin K$. Hence, these results can be used for defining a Borel-measurable function $\varphi : \mathbb{R}^m \to \widetilde{K}$ satisfying $\varphi(y) = 0$ for $y \in K$ and $\delta_K(\varphi(y)) + y^\intercal \varphi(y) < 0$ for $y \notin K$. Dividing $\varphi(Y)$ by the strictly positive process $1 + |\varphi(Y)| + \delta_K(\varphi(Y))$ defines a process Z with the required properties.

Now that we have these preliminary results at our disposal, we are in position to formulate and prove the main result which justifies the formulation of the deep SMP algorithm like in Section 4.4.

Theorem 4.6. *Let a dual problem according to Definition 2.11 be given such that Assumptions 2.2 and 4.1 and either $U(0) > -\infty$ or $U = \log$ hold and consider an admissible pair $(y^*, v^*) \in \mathcal{D}$. Moreover, let $Y^{(y^*,v^*)}$, p_2^* and q_2^* denote processes which solve the following FBSDE system and assume that $p_2^* Y^{(y^*,v^*)}$ is even a martingale. For $t \in [0,T]$:*

$$\begin{aligned} dY^{(y^*,v^*)}(t) &= -Y^{(y^*,v^*)}(t)\big[\big(r(t) + \delta_K(v^*(t))\big)\,dt + \big(\theta(t) + \sigma^{-1}(t)v^*(t)\big)^\intercal\,dB(t)\big], \\ dp_2^*(t) &= \big[\big(r(t) + \delta_K(v^*(t))\big)\,p_2^*(t) + \big(\theta(t) + \sigma^{-1}(t)\,v^*(t)\big)^\intercal\,q_2^*(t)\big]\,dt + q_2^{*\intercal}(t)\,dB(t), \end{aligned} \tag{4.5}$$

with initial condition $Y^{(y^*,v^*)}(0) = y^*$ and terminal condition $p_2^*(T) = -\widetilde{U}'\big(Y^{(y^*,v^*)}(T)\big)$, respectively. Then (y^*, v^*) is optimal in the sense of Definition 2.11 if and only if the *following properties are satisfied almost surely for almost every* $t \in [0,T]$*:*

$$p_2^*(0) = x_0, \tag{4.6}$$
$$p_2^{*\,-1}(t)\big[\sigma^{-1}\big]^{\intercal}(t)\, q_2^*(t) \in K \quad \textit{and} \tag{4.7}$$
$$p_2^*(t)\,\delta_K(v^*(t)) + \big(\sigma^{-1}(t)\, v^*(t)\big)^{\intercal}\, q_2^*(t) = 0. \tag{4.8}$$

Proof. Note that Lemma 4.3 ensures that there always exists a pair of processes (p_2^*, q_2^*) which solves (4.5) (together with the unique process $Y^{(y^*,v^*)}$ satisfying (2.12)) such that $p_2^* Y^{(y^*,v^*)}$ is a martingale. For reasons of clarity, we will subdivide the proof into seven steps. At first, we show that the above conditions are necessarily satisfied by an optimal control.

Step 1. Necessary Condition: (4.6) *is satisfied: Applying the dominated convergence theorem to a sequence of difference quotients of a function which maps, for fixed* v^**, possible initial conditions to their corresponding expected terminal "gains".*

Let $(y^*, v^*) \in \mathcal{D}$ be an optimal pair. Then we can define a function $h : \mathbb{R}^+ \to \mathbb{R}$ by $h(z) := x_0 z y^* + \mathbb{E}\big[\widetilde{U}\big(zY^{(y^*,v^*)}(T)\big)\big]$ for each $z \in \mathbb{R}^+$. Note that this function is well-defined due to $(zy^*, v^*) \in \mathcal{D}$ (cf. (2.13)), Assumption 4.1 and $zY^{(y^*,v^*)} = Y^{(zy^*,v^*)}$, which follows from the fact that solutions to (2.12) are stochastic exponentials. Since (y^*, v^*) is optimal, we can conclude that h has a global minimum in $z = 1$ which is also a local one due to the openness of $\mathbb{R}^+$. Hence, differentiability in $z = 1$, in particular for the second summand, would be favorable in order to obtain a stationary point. An application of the dominated convergence theorem shows that this holds indeed:
Let $\delta \in \mathbb{R}^+$ be small and fixed. It suffices to consider h in a neighborhood of 1, i.e. $h|_{(1-\delta,1+\delta)}$. Obviously, thanks to Lemma 2.7 *(ii)*, $\widetilde{U}\big(zY^{(y^*,v^*)}(T)\big)$ is differentiable in z for an arbitrary, but fixed $\omega \in \Omega$ with derivative $Y^{(y^*,v^*)}(T)\,\widetilde{U}'\big(zY^{(y^*,v^*)}(T)\big)$. Moreover, we obtain from the considerations above and Assumption 4.1 that $\widetilde{U}\big(zY^{(y^*,v^*)}(T)\big) \in L^1$ holds for any $z \in (1-\delta, 1+\delta)$. Finally, we can conclude from $\widetilde{U}' < 0$ and the convexity of $\widetilde{U}$ (cf. Lemma 2.7 *(ii)*) that the following estimate holds pointwise for any $z \in (1-\delta, 1+\delta)$:

$$\big|Y^{(y^*,v^*)}(T)\,\widetilde{U}'\big(zY^{(y^*,v^*)}(T)\big)\big| \le -Y^{(y^*,v^*)}(T)\,\widetilde{U}'\big((1-\delta)Y^{(y^*,v^*)}(T)\big), \tag{4.9}$$

where the majorant on the right-hand side lies in L^1 due to $\big((1-\delta)y^*, v^*\big) \in \mathcal{D}$ and Lemma 4.3. Note that finding a majorant in the sense of (4.9) suffices for applying the dominated convergence theorem to a sequence of difference quotients, as in our case, due to the mean value theorem. Hence, h is differentiable in $z = 1$, which, therefore, is a stationary point. As the above procedure additionally yields $h'(1)$ explicitly, we obtain

$$x_0\, y^* + \mathbb{E}\big[Y^{(y^*,v^*)}(T)\,\widetilde{U}'\big(Y^{(y^*,v^*)}(T)\big)\big] = 0.$$

As $p_2^* Y^{(y^*,v^*)}$ is a martingale by assumption (cf. Lemma 4.3) and $\mathcal{F}_0$ is $\mathbb{P}$-trivial, we can conclude that the second summand corresponds to $-p_2^*(0)\, y^*$. This proves (4.6).

Step 2. Necessary Condition: (4.7) *is satisfied:* $[0,1] \ni \varepsilon \mapsto \widetilde{U}\big(Y^{(y^*,v^*+\varepsilon(v-v^*))}(T)\big)$ *is convex for certain pairs* $(y^*, v) \in \mathcal{D}$ *and almost every* $\omega \in \Omega$*.*

Fix an arbitrary pair $(y^*, v) \in \mathcal{D}$ which satisfies $(y^*, v - v^*) \in \mathcal{D}$ (cf. Remark 4.7). Hence, the initial condition agrees in particular with the one associated with our optimal pair. We denote the function described in the summary of *Step 2* above as Φ_v, where ω is omitted for notational convenience. Clearly, Φ_v is well-defined as convex combinations of admissible control processes are again admissible. This is an immediate consequence of the convexity of δ_K (cf. Remark 2.10) and the integrability properties in the definition of $\mathcal{D}$. Hence, the corresponding SDE (2.12) admits a unique solution which is a stochastic exponential. At time T it is precisely given by

$$\begin{aligned}Y^{(y^*,v^*+\varepsilon(v-v^*))}(T) = y^* \exp\Big(&-\int_0^T \big(r(t)+\delta_K\big(v^*(t)+\varepsilon(v(t)-v^*(t))\big)\big)\,dt\\ &-\frac{1}{2}\int_0^T \big|\theta(t)+\sigma^{-1}(t)\big(v^*(t)+\varepsilon(v(t)-v^*(t))\big)\big|^2\,dt\\ &-\int_0^T \big(\theta(t)+\sigma^{-1}(t)\big(v^*(t)+\varepsilon(v(t)-v^*(t))\big)\big)^\intercal\,dB(t)\Big)\end{aligned}$$

This representation allows us to conclude that $g_v(\varepsilon) := \log\big(Y^{(y^*,v^*+\varepsilon(v-v^*))}(T)\big)$ is concave in ε due to the concavity of $\varepsilon \mapsto C\varepsilon^2$ with $C \leq 0$, the linearity of (stochastic) integrals and the convexity of δ_K, which is preserved, if the input is transformed by an affine function. This is certainly the case, if the integrals in the above representation are finite, which holds for almost every $\omega \in \Omega$ due to the definition of $\mathcal{D}$. Moreover, we obtain from (2.8) and Lemma 2.8 *(ii)* that $\widetilde{U} \circ \exp$ is nonincreasing and convex on $\mathbb{R}$. Therefore, Lemma 4.4 shows that $\Phi_v = \widetilde{U} \circ \exp \circ g_v$ is indeed a convex function.

Step 3. Necessary Condition: (4.7) *is satisfied: Find for each* $\varepsilon \in (0,1)$ *an upper estimate for the difference quotient of* Φ_v *over* $[0,\varepsilon]$ *such that the resulting sequence converges for* $\varepsilon \searrow 0$.

Let $\Delta_\varepsilon\Phi_v$ denote the difference quotient of Φ_v over $[0,\varepsilon]$. From the convexity of Φ_v we conclude that $\big(\Delta_\varepsilon\Phi_v\big)_{\varepsilon\in(0,1)}$ is increasing in ε, which will allow us to apply the reverse Fatou lemma in the following. Note that it is in general not easy to obtain an explicit expression for $\lim_{\varepsilon\searrow 0}\Delta_\varepsilon\Phi_v$ since it depends on δ_K', if it exists, which is linked to the existence of maximizers in (2.9). However, we can circumvent this issue by considering a suitable sequence which lies for every $\varepsilon \in (0,1)$ above $\Delta_\varepsilon\Phi_v$ and whose limit can be calculated rather easily. This suffices for proving (4.7), as we shall see later.
Similar to [16], define $H_{v,\varepsilon}$ as the logarithm of $Y^{(y^*,v^*)}/Y^{(y^*,v^*+\varepsilon(v-v^*))}$, i.e. for $t \in [0,T]$ we obtain from the explicit representation presented in *Step 2*:

$$\begin{aligned}H_{v,\varepsilon}(t) = &\int_0^t \delta_K\big(v^*(s)+\varepsilon(v(s)-v^*(s))\big) - \delta_K\big(v^*(s)\big)\,ds + \varepsilon\int_0^t \theta^\intercal(s)\,\sigma^{-1}(s)\,(v(s)-v^*(s))\,ds\\ &+\varepsilon\int_0^t v^{*\intercal}(s)\big[\sigma^{-1}\big]^\intercal(s)\,\sigma^{-1}(s)\,(v(s)-v^*(s))\,ds + \frac{1}{2}\varepsilon^2\int_0^t \big|\sigma^{-1}(s)\,(v(s)-v^*(s))\big|^2\,ds\\ &+\varepsilon\int_0^t \big(\sigma^{-1}(s)\,(v(s)-v^*(s))\big)^\intercal\,dB(s).\end{aligned}$$

Since δ_K is subadditive and positive homogeneous, it follows that an upper estimate for the integrand in the first integral is given by $\varepsilon\,\delta_K\big(v(s)-v^*(s)\big)$. Hence, because $(y^*, v-v^*) \in \mathcal{D}$

holds by assumption, this adjusted process can be differentiated at $\varepsilon = 0$ quite effortlessly (for fixed $\omega \in \Omega$, except for a null set). This leads to a process H_v which is defined by

$$H_v(t) := \int_0^t \big[\delta_K\big(v(s) - v^*(s)\big) + \theta^\intercal(s)\,\sigma^{-1}(s)\,(v(s) - v^*(s)) \\ + v^{*\intercal}(s)\,\big[\sigma^{-1}\big]^\intercal(s)\,\sigma^{-1}(s)\,(v(s) - v^*(s))\big]\,ds + \int_0^t \big(\sigma^{-1}(s)\,(v(s) - v^*(s))\big)^\intercal\,dB(s),$$

$t \in [0, T]$. By expanding the quotient in the definition of $\Delta_\varepsilon \Phi_v$ in the spirit of the chain rule we obtain for $\varepsilon \in (0, 1)$:

$$\Delta_\varepsilon \Phi_v = \frac{\widetilde{U}\big(Y^{(y^*, v^* + \varepsilon(v - v^*))}(T)\big) - \widetilde{U}\big(Y^{(y^*, v^*)}(T)\big)}{Y^{(y^*, v^* + \varepsilon(v - v^*))}(T) - Y^{(y^*, v^*)}(T)}\, Y^{(y^*, v^*)}(T)\, \frac{\exp\big(-H_{v,\varepsilon}(T)\big) - 1}{\varepsilon}. \tag{4.10}$$

Note that the aforementioned upper estimate of $H_{v,\varepsilon}(T)$ serves the same purpose in (4.10) as the first quotient is negative due to the monotonicity of $\widetilde{U}$ according to Lemma 2.7 *(ii)*. Therefore, we obtain that the estimate for the last quotient converges almost surely to $-H_v(T)$, if we send ε towards 0. As in our previous considerations, it is essential that the occurring integrals are almost surely finite, which is guaranteed by the admissibility of v^*, v and $v - v^*$ and the estimate $ab \le 0.5\,(a^2 + b^2)$, where a, b are arbitrary real numbers. Furthermore, it holds that $Y^{(y^*, v^* + \varepsilon(v - v^*))}(T)$ converges almost surely to $Y^{(y^*, v^*)}(T)$ for $\varepsilon \searrow 0$. We convince ourselves of this fact by considering the first integral in the explicit representation (cf. *Step 2*) as the result clearly holds for the remaining integrals. At first, we find bounds for the integrand which are independent of ε. For $t \in [0, T]$ we have

$$0 \overset{(1)}{\le} \delta_K\big(v^*(t) + \varepsilon(v(t) - v^*(t))\big) \overset{(2)}{\le} (1 - \varepsilon)\,\delta_K(v^*(t)) + \varepsilon\,\delta_K(v(t)) \overset{(1)}{\le} \delta_K(v^*(t)) + \delta_K(v(t)),$$

where (1) and (2) follow from the nonnegativity and the convexity of δ_K, respectively. (cf. Remark 2.10) Since the controls v and v^* are admissible, it follows that the right-hand side of the previous inequality is integrable with respect to $\lambda|_{[0,T]}$ for almost every $\omega \in \Omega$. Moreover, we obtain from [20] (cf. Theorem 13.2 therein) that δ_K is lower semicontinuous, which implies in particular that

$$\delta_K(v^*(t)) \le \liminf_{\varepsilon \searrow 0} \delta_K\big(v^*(t) + \varepsilon(v(t) - v^*(t))\big) \tag{4.11}$$

holds. On the other hand, we can conclude from the subadditivity property and the positive homogeneity of δ_K and the admissibility of $v - v^*$ that we have almost surely for almost every $t \in [0, T]$:

$$\limsup_{\varepsilon \searrow 0} \delta_K\big(v^*(t) + \varepsilon(v(t) - v^*(t))\big) \le \delta_K(v^*(t)) + \limsup_{\varepsilon \searrow 0} \varepsilon\,\delta_K\big(v(t) - v^*(t)\big) = \delta_K(v^*(t)).$$

Combining this with (4.11) shows that the considered integrand converges almost everywhere to $\delta_K(v^*(t))$. Hence, the dominated convergence theorem proves that also the integral converges (for almost every $\omega \in \Omega$).
Finally, this shows by means of Lemma 2.7 *(ii)* that the first quotient in (4.10) converges

almost surely towards $\widetilde{U}'\big(Y^{(y^*,v^*)}(T)\big)$. Taking the limit superior on both sides of (4.10) and estimating the right-hand side, as discussed earlier, leads to

$$\limsup_{\varepsilon\searrow 0} \Delta_\varepsilon \Phi_v \le -\widetilde{U}'\big(Y^{(y^*,v^*)}(T)\big)\, Y^{(y^*,v^*)}(T)\, H_v(T), \quad \text{a.s.,} \tag{4.12}$$

which is essential for *Step 4* below.

Step 4. Necessary Condition: (4.7) *is satisfied: Localization and an application of the reverse Fatou lemma.*

Consider an arbitrary pair $(y^*, v) \in \mathcal{D}$ which satisfies $(y^*, v - v^*) \in \mathcal{D}$. The aim of this step is finding a sequence of stopping times converging almost surely to T which guarantees on the one hand that H_v is square-integrable and on the other hand that a stochastic integral appearing in *Step 5* is even a true martingale up to each of these stopping times. For each $n \in \mathbb{N}$, we define the stopping time

$$\begin{aligned}\tau_n^v := \inf\left\{ t \ge 0 \;:\; \left| \int_0^t p_2^*(s) Y^{(y^*,v^*)}(s)\big(v(s) - v^*(s)\big)^\intercal \big[\sigma^{-1}\big]^\intercal(s)\, dB(s) \right| \ge n \right\} \\ \wedge\ \inf\big\{ t \ge 0 \;:\; |H_v(t)| \ge n \big\} \ \wedge\ T.\end{aligned} \tag{4.13}$$

Since the process $p_2^* Y^{(y^*,v^*)}$ is continuous, σ^{-1} is bounded and $v - v^*$ is L^2 with respect to $\mathbb{P} \otimes \lambda|_{[0,T]}$ thanks to its admissibility, it follows that the process within the norm in the first line of (4.13) is a well-defined continuous local martingale. Hence, the first line defines a localizing sequence for this process such that corresponding stopped process is bounded for each $n \in \mathbb{N}$. A similar argument shows that the stopping time in the second line localizes the well-defined continuous semimartingale H_v in the same sense as above, i.e. H_v is locally bounded. Note that the pointwise minimum of two localizing sequences is a localizing sequence for both processes. Clearly, the continuity is essential such that $\tau_n^v \nearrow T$ holds indeed almost surely.
Fix $n \in \mathbb{N}$ and define $\overline{v}_n := \mathbb{1}_{[0,\tau_n^v]}(v - v^*)$. Obviously, $\overline{v}_n$ is again admissible due to the positive homogeneity of δ_K and $\mathbb{1}_{[0,\tau_n^v]}$ mapping to $\{0,1\}$. Note that $\mathbb{1}_{[0,\tau_n^v]}$ is progressively measurable because its paths are left-continuous and $\{\omega \in \Omega \mid \mathbb{1}_{[0,\tau_n^v(\omega)]}(t) = 0\} = \{\tau_n^v < t\} \in \mathcal{F}_t$ holds for every $t \in [0,T]$ as τ_n^v is in particular a weak stopping time. Replacing $v - v^*$ with $\overline{v}_n$ in *Step 2* and *Step 3* shows that the original arguments work in this case as well, since the most crucial assumption is the admissibility of the process which is multiplied by ε. Note that the limit of the estimate for the rightmost quotient in (4.10) is precisely $H_v^{\tau_n^v}(T)$ (a.s.) here as we additionally obtain the factor $\mathbb{1}_{[0,\tau_n^v]}$ for each integrand. Clearly, the first quotient in (4.10) converges again almost surely towards $\widetilde{U}'\big(Y^{(y^*,v^*)}(T)\big)$ because (2) from *Step 3* holds again as $\varepsilon\mathbb{1}_{[0,\tau_n^v]}$ and $1 - \varepsilon\mathbb{1}_{[0,\tau_n^v]}$ (understood pointwise) are admissible weights for a convex combination and $\mathbb{1}_{[0,\tau_n^v]}$ is bounded. Moreover, define

$$\Delta_\varepsilon \Phi_v^n := \frac{\widetilde{U}\big(Y^{(y^*,v^*+\varepsilon\mathbb{1}_{[0,\tau_n^v]}(v-v^*))}(T)\big) - \widetilde{U}\big(Y^{(y^*,v^*)}(T)\big)}{\varepsilon}. \tag{4.14}$$

Hence, we can conclude from the previous considerations for any $n \in \mathbb{N}$:

$$\limsup_{\varepsilon\searrow 0} \Delta_\varepsilon \Phi_v^n \le -\widetilde{U}'\big(Y^{(y^*,v^*)}(T)\big)\, Y^{(y^*,v^*)}(T)\, H_v^{\tau_n^v}(T), \quad \text{a.s.} \tag{4.15}$$

Note that the right-hand side agrees with (4.12) except for the fact that H_v is now stopped by the stopping time τ_n^v.
Fix $n \in \mathbb{N}$. We recall from *Step 3* applied to the modification (4.14) that $\left(\Delta_\varepsilon \Phi_v^n\right)_{\varepsilon\in(0,1)}$ is increasing in ε. Therefore, this family is almost surely bounded above by

$$\widetilde{U}\left(Y^{(y^*,v^*+\mathbb{1}_{[0,\tau_n^v]}(v-v^*))}(T)\right) - \widetilde{U}\left(Y^{(y^*,v^*)}(T)\right),$$

which lies in L^1 due to the admissibility of $\mathbb{1}_{[0,\tau_n^v]}(v-v^*)$ and Assumption 4.1. Hence, we can apply the reverse Fatou lemma, which yields combined with (4.15):

$$\begin{aligned} 0 \;&\le \limsup_{\varepsilon\searrow 0} \mathbb{E}\big[\Delta_\varepsilon\Phi_v^n\big] \;\le\; \mathbb{E}\Big[\limsup_{\varepsilon\searrow 0} \Delta_\varepsilon\Phi_v^n\Big] \\ &\le \mathbb{E}\Big[-\widetilde{U}'\big(Y^{(y^*,v^*)}(T)\big)\, Y^{(y^*,v^*)}(T)\, H_v^{\tau_n^v}(T)\Big], \end{aligned} \tag{4.16}$$

where the first inequality results from the optimality of (y^*, v^*) as all the considered dual state processes start in y^*. Furthermore, it follows from Lemma 4.3 and (4.13) that the expectation on the right-hand side of (4.16) is finite since the integrand even lies in L^2.

Step 5. Necessary Condition: (4.7) *is satisfied: Applying Itô's formula to the right-hand side of* (4.16) *and, finally, Lemma 4.5.*

At first, we recall that the continuous process $p_2^*\, Y^{(y^*,v^*)}$ is even a martingale by assumption. Therefore, as p_2^* satisfies the terminal condition associated with (4.2), we can conclude for every $t \in [0,T]$:

$$p_2^*(t)\, Y^{(y^*,v^*)}(t) = \mathbb{E}\big[-\widetilde{U}'\big(Y^{(y^*,v^*)}(T)\big)\, Y^{(y^*,v^*)}(T)\big|\mathcal{F}_t\big], \quad \text{a.s.} \tag{4.17}$$

Hence, $p_2^*\, Y^{(y^*,v^*)}$ is a modification of the continuous process V^* discussed in the proof of Lemma 4.3. The continuity of both processes implies by means of a standard result, whose proof essentially relies on the fact that $[0,T]\cap\mathbb{Q}$ is countable and dense in $[0,T]$, that they are even indistinguishable. This observation enables us to apply Itô's formula more efficiently to the process $p_2^*\, Y^{(y^*,v^*)}\, H_v^{\tau_n^v}$ as we have already calculated the dynamics of $p_2^*\, Y^{(y^*,v^*)}$ in the course of the proof of Lemma 4.3.

$$\begin{aligned} d\big(p_2^*\, Y^{(y^*,v^*)}\, H_v^{\tau_n^v}\big)(t) &= p_2^*(t)\, Y^{(y^*,v^*)}(t)\, \mathbb{1}_{[0,\tau_n^v]}(t)\Big[\delta_K\big(v(t)-v^*(t)\big) + \theta^\intercal(t)\,\sigma^{-1}(t)\,(v(t)-v^*(t)) \\ &\quad + v^{*\intercal}(t)\,\big[\sigma^{-1}\big]^\intercal(t)\,\sigma^{-1}(t)\,(v(t)-v^*(t))\Big]\,dt + \mathbb{1}_{[0,\tau_n^v]}(t)\,\big(\sigma^{-1}(t)\,(v(t)-v^*(t))\big)^\intercal\big[q_2^*(t)Y^{(y^*,v^*)}(t) \\ &\quad - p_2^*(t)\, Y^{(y^*,v^*)}(t)\,\big(\theta(t)+\sigma^{-1}(t)\,v^*(t)\big)\big]\,dt + \Big[p_2^*(t)\, Y^{(y^*,v^*)}(t)\,\mathbb{1}_{[0,\tau_n^v]}(t)\,(v(t)-v^*(t))^\intercal\big[\sigma^{-1}\big]^\intercal(t) \\ &\quad + \big[q_2^*(t)Y^{(y^*,v^*)}(t) - p_2^*(t)\, Y^{(y^*,v^*)}(t)\,\big(\theta(t)+\sigma^{-1}(t)\,v^*(t)\big)\big]^\intercal\, H_v^{\tau_n^v}(t)\Big]\,dB(t), \end{aligned}$$

where $t\in[0,T]$. It follows immediately from the definition of τ_n^v in (4.13) that the stochastic integrals arising from the local martingale part above are even true martingales. In order to illustrate this for the last term, recall that $H_v^{\tau_n^v}$ is bounded due to (4.13) and for each $t\in[0,T]$:

$$q_2^*(t)Y^{(y^*,v^*)}(t) - p_2^*(t)\, Y^{(y^*,v^*)}(t)\,\big(\theta(t)+\sigma^{-1}(t)\,v^*(t)\big)$$

coincides with $W^*(t)$ from the proof of Lemma 4.3. Hence, the statement follows as $W^* \in H^2(0,T;\mathbb{R}^m)$ holds. Simplifying the remaining differentials and applying the identity $(CD)^\intercal = D^\intercal C^\intercal$ for real-valued matrices C, D shows that the finite variation part can be written as

$$\mathbb{1}_{[0,\tau_n^v]}(t)\, Y^{(y^*,v^*)}(t) \left[p_2^*(t)\, \delta_K\big(v(t)-v^*(t)\big) + q_2^{*\intercal}(t)\, \sigma^{-1}(t)\, (v(t)-v^*(t))\right] dt.$$

Combining this with the martingale property of the occuring stochastic integrals with respect to B, $H_v^{\tau_n^v}$ starting in 0 and (4.16) proves

$$0 \le \mathbb{E}\left[\int_0^{\tau_n^v} Y^{(y^*,v^*)}(t) \left[p_2^*(t)\, \delta_K\big(v(t)-v^*(t)\big) + q_2^{*\intercal}(t)\, \sigma^{-1}(t)\, (v(t)-v^*(t))\right] dt\right]. \tag{4.18}$$

We conclude the proof of (4.7) by showing that if (4.7) did not hold, we would have a contradiction to (4.18). Clearly, $p_2^{*\,-1}\left[\sigma^{-1}\right]^\intercal q_2^*$ is a well-defined, progressively measurable process since p_2^* is a strictly positive process (cf. (4.17)). Hence, we obtain from Lemma 4.5 that there exists a progressively measurable process $\overline{v}$ which characterizes $p_2^{*\,-1}\left[\sigma^{-1}\right]^\intercal q_2^*$ taking values in K in the following way:

$$\begin{aligned} p_2^{*\,-1}(t)\left[\sigma^{-1}\right]^\intercal(t)\, q_2^*(t) \in K &\iff \delta_K(\overline{v}(t)) + p_2^{*\,-1}(t)\, q_2^{*\intercal}(t)\, \sigma^{-1}(t)\, \overline{v}(t) = 0 \quad \text{and} \\ p_2^{*\,-1}(t)\left[\sigma^{-1}\right]^\intercal(t)\, q_2^*(t) \notin K &\iff \delta_K(\overline{v}(t)) + p_2^{*\,-1}(t)\, q_2^{*\intercal}(t)\, \sigma^{-1}(t)\, \overline{v}(t) < 0. \end{aligned} \tag{4.19}$$

As the processes $\overline{v}$ and $\delta_K(\overline{v})$ are almost everywhere bounded, it follows immediately that $(y^*,\overline{v}) \in \mathcal{D}$ holds (cf. (2.13)). Define $v' := v^* + \overline{v}$. Then (y^*, v') is also admissible due to the subadditivity property of δ_K. Since we have by construction that also the difference $v' - v^* = \overline{v}$ is an admissible dual control, it follows that our previous results are applicable to (y^*, v'). Hence, (4.18) becomes in this case

$$0 \le \mathbb{E}\left[\int_0^{\tau_n^{v'}} Y^{(y^*,v^*)}(t) \left[p_2^*(t)\, \delta_K\big(\overline{v}(t)\big) + q_2^{*\intercal}(t)\, \sigma^{-1}(t)\, \overline{v}(t)\right] dt\right]. \tag{4.20}$$

If $N := \left\{(\omega,t) \in \Omega \times [0,T] : p_2^{*\,-1}(t)\left[\sigma^{-1}\right]^\intercal(t)\, q_2^*(t) \notin K\right\}$ was not a $\mathbb{P} \otimes \lambda|_{[0,T]}$-null set, then there would exist a number $n^* \in \mathbb{N}$ such that

$$\mathbb{P} \otimes \lambda|_{[0,T]}\big(N \cap \{(\omega,t) \in \Omega \times [0,T] : \mathbb{1}_{[0,\tau_n^{v'}]}(t) = 1\}\big) > 0 \tag{4.21}$$

holds as well for every $n \in \mathbb{N}$ with $n \ge n^*$. This is an immediate consequence of $\tau_n^{v'} \nearrow T$, i.e. $\mathbb{1}_{[0,\tau_n^{v'}]} \nearrow \mathbb{1}_{[0,T]}$, almost surely as $n \to \infty$ and the fact that $\mathbb{P} \otimes \lambda|_{[0,T]}$ is continuous from below. Hence, combining (4.19), (4.21) and the fact that $p_2^* Y^{(y^*,v^*)}$ is a strictly positive process shows that (4.20) cannot hold for $n \ge n^*$, which yields the desired contradiction. Therefore, N has to be a $\mathbb{P} \otimes \lambda|_{[0,T]}$-null set. By Fubini's theorem, this is equivalent to $p_2^{*\,-1}(t)\left[\sigma^{-1}\right]^\intercal(t)\, q_2^*(t) \in K$ being satisfied almost surely for almost every $t \in [0,T]$.

Step 6. Necessary Condition: (4.8) *is satisfied: Refining the previous steps for the special case* $v \equiv 0$.

At first, we notice that the arguments presented above are in general not valid for $(y^*, 0) \in \mathcal{D}$

as $-v^*$ is possibly not admissible. However, we can adapt the proof such that a result similar to (4.18) holds in this case as well. The key observation for this purpose is that we have for every $\varepsilon \in [0,1]$ by the positive homogeneity of δ_K:

$$\delta_K\big(v^*(t) + \varepsilon(v(t) - v^*(t))\big) = \delta_K\big((1-\varepsilon)v^*(t)\big) = (1-\varepsilon)\,\delta_K\big(v^*(t)\big). \tag{4.22}$$

Obviously, the right-hand side defines a convex function of ε. Hence, combining this with the admissibility of v^* shows that the conclusion of *Step 2* holds. As the expression above is affine in ε, there is no need for an upper estimate of $H_{0,\varepsilon}$ as in *Step 3*. Except for this detail, we can proceed analogously to above. Hence, differentiating $H_{0,\varepsilon}$ at $\varepsilon = 0$ results in a process H_0 which is defined by

$$\begin{aligned} H_0(t) := &\int_0^t \big[-\delta_K\big(v^*(s)\big) + \theta^\intercal(s)\,\sigma^{-1}(s)\,(-v^*(s)) + v^{*\intercal}(s)\,\big[\sigma^{-1}\big]^\intercal(s)\,\sigma^{-1}(s)\,(-v^*(s))\big]\,ds \\ &+ \int_0^t \big(\sigma^{-1}(s)\,(-v^*(s))\big)^\intercal\,dB(s), \quad t \in [0,T]. \end{aligned}$$

Note that H_0 is well-defined due to the admissibility of v^*. Furthermore, thanks to (4.22), the proof of $Y^{(y^*,(1-\varepsilon)v^*)}(T) \to Y^{(y^*,v^*)}(T)$ almost surely as $\varepsilon \searrow 0$ does not require the admissibility of $-v^*$ as an assumption. Therefore, we obtain as in *Step 3*

$$\limsup_{\varepsilon\searrow 0} \Delta_\varepsilon \Phi_0 \le -\widetilde{U}'\big(Y^{(y^*,v^*)}(T)\big)\,Y^{(y^*,v^*)}(T)\,H_0(T), \quad \text{a.s.} \tag{4.23}$$

We define a sequence of stopping times $(\tau_n^0)_{n\in\mathbb{N}}$ as in (4.13) for $v = 0$ and H_0 as defined above. Clearly, $\tau_n^0 \nearrow T$ almost surely as $n \to \infty$ holds again due to the admissibility of v^*. Moreover, it follows from the fact that (4.22) is still true, if we replace ε with $\varepsilon \mathbb{1}_{[0,\tau_n^0]}$, that the localization procedure from *Step 4* also works in this case. In particular, we have that $(1 - \varepsilon\mathbb{1}_{[0,\tau_n^0]})\,v^*$ is admissible for every $\varepsilon \in [0,1]$ and $n \in \mathbb{N}$. Therefore, we obtain again by means of the reverse Fatou lemma (cf. (4.16)):

$$0 \le \mathbb{E}\Big[\limsup_{\varepsilon\searrow 0} \Delta_\varepsilon \Phi_0^n\Big] \le \mathbb{E}\Big[-\widetilde{U}'\big(Y^{(y^*,v^*)}(T)\big)\,Y^{(y^*,v^*)}(T)\,H_0^{\tau_n^0}(T)\Big]. \tag{4.24}$$

Finally, an application of Itô's formula as in *Step 5* shows that the structure of the dynamics of $p_2^*\,Y^{(y^*,v^*)}\,H_0^{\tau_n^0}$ agrees with the one obtained in the original argument, except for replacing $\delta_K\big(-v^*(t)\big)$ with $-\delta_K\big(v^*(t)\big)$. This observation is not surprising, if one compares the dynamics of H_0 with H_v from *Step 3*. Therefore, the analogue of (4.18) is given by

$$0 \le \mathbb{E}\bigg[\int_0^{\tau_n^0} Y^{(y^*,v^*)}(t)\big[-p_2^*(t)\,\delta_K\big(v^*(t)\big) - q_2^{*\intercal}(t)\,\sigma^{-1}(t)\,v^*(t)\big]\,dt\bigg]. \tag{4.25}$$

in this case. This is equivalent to

$$0 \ge \mathbb{E}\bigg[\int_0^{\tau_n^0} p_2^*(t)\,Y^{(y^*,v^*)}(t)\big[\delta_K\big(v^*(t)\big) + p_2^{*-1}(t)\,q_2^{*\intercal}(t)\,\sigma^{-1}(t)\,v^*(t)\big]\,dt\bigg]. \tag{4.26}$$

It follows from the strict positivity of $p_2^*\,Y^{(y^*,v^*)}$, (4.7), whose proof was carried out independently from our current considerations, and the definition of δ_K that the integrand in

(4.26) is nonnegative (up to a $\mathbb{P}\otimes\lambda|_{[0,T]}$-null set). However, since (4.26) holds as well, we necessarily obtain the following result for every $n\in\mathbb{N}$:

$$\begin{aligned}&\mathbb{P}\otimes\lambda|_{[0,T]}\big(M\cap\{(\omega,t)\in\Omega\times[0,T]:\mathbb{1}_{[0,\tau_n^0]}(t)=1\}\big)=0,\quad\text{where}\\&M:=\big\{(\omega,t)\in\Omega\times[0,T]:\delta_K\big(v^*(t)\big)+{p_2^*}^{-1}(t)\,{q_2^*}^{\intercal}(t)\,\sigma^{-1}(t)\,v^*(t)\neq 0\big\}.\end{aligned}\tag{4.27}$$

Combining this with $\tau_n^0\nearrow T$ almost surely as $n\to\infty$ and $\mathbb{P}\otimes\lambda|_{[0,T]}$ being continuous from below proves that M is a $\mathbb{P}\otimes\lambda|_{[0,T]}$-null set. Hence, Fubini's theorem applied to $\mathbb{1}_{M^c}$ shows that we have almost surely for almost every $t\in[0,T]$:

$$\delta_K\big(v^*(t)\big)+{p_2^*}^{-1}(t)\,{q_2^*}^{\intercal}(t)\,\sigma^{-1}(t)\,v^*(t)=0.\tag{4.28}$$

Finally, transposing the scalars on the left-hand side and multiplying (4.28) by $p_2^*(t)$ completes the proof of (4.8).

Step 7. Sufficient Condition: (y^*,v^*) is optimal: Proving that $p_2^*Y^{(y,v)}$ is a supermartingale for every $(y,v)\in\mathcal{D}$ and applying the convexity of $\widetilde{U}$.

We conclude the proof by showing that the three conditions together are even sufficient for deeming a dual pair optimal. Let $(y^*,v^*)\in\mathcal{D}$ and the corresponding (unique) state process $Y^{(y^*,v^*)}$ be given such that (4.6), (4.7) and (4.8) are satisfied together with a pair (p_2^*,q_2^*) which solves the dual adjoint equation and achieves that $p_2^*Y^{(y^*,v^*)}$ is a martingale. Consider an arbitrary, but fixed pair $(y,v)\in\mathcal{D}$ with associated state process $Y^{(y,v)}$. We obtain by means of the integration by parts formula for continuous semimartingales

$$\begin{aligned}p_2^*Y^{(y,v)}=\;&p_2^*(0)y+\int_0^{\cdot}Y^{(y,v)}(t)\big[\big(r(t)+\delta_K(v^*(t))\big)p_2^*(t)+\big(\theta(t)+\sigma^{-1}(t)\,v^*(t)\big)^{\intercal}q_2^*(t)\big]\,dt\\&-\int_0^{\cdot}Y^{(y,v)}(t)\,\big[p_2^*(t)\,\big(r(t)+\delta_K(v(t))\big)+\big(\theta(t)+\sigma^{-1}(t)v(t)\big)^{\intercal}q_2^*(t)\big]\,dt\\&+\int_0^{\cdot}Y^{(y,v)}(t)\,\big[-p_2^*(t)\,\big(\theta(t)+\sigma^{-1}(t)v(t)\big)^{\intercal}+{q_2^*}^{\intercal}(t)\big]\,dB(t).\end{aligned}$$

Applying (4.6) and (4.8) and noticing that there are two pairs of terms which cancel each other out yields

$$\begin{aligned}p_2^*Y^{(y,v)}=\;&x_0\,y+\int_0^{\cdot}-p_2^*(t)\,Y^{(y,v)}(t)\,\big[\delta_K(v(t))+v(t)^{\intercal}{p_2^*}^{-1}(t)\,\big[\sigma^{-1}\big]^{\intercal}(t)\,q_2^*(t)\big]\,dt\\&+\int_0^{\cdot}\big[-p_2^*(t)\,Y^{(y,v)}(t)\,\big(\theta(t)+\sigma^{-1}(t)v(t)\big)^{\intercal}+Y^{(y,v)}(t)\,{q_2^*}^{\intercal}(t)\big]\,dB(t).\end{aligned}\tag{4.29}$$

Since $p_2^*Y^{(y,v)}$ is a strictly positive (cf. (4.17) and the accompanying arguments for the strict positivity of p_2^*), continuous semimartingale, it admits a representation of the form $p_2^*Y^{(y,v)}=x_0y\,\mathcal{E}(Z)$, where Z is precisely the associated stochastic logarithm

$$\begin{aligned}Z=&-\int_0^{\cdot}\big[\delta_K(v(t))+v(t)^{\intercal}{p_2^*}^{-1}(t)\,\big[\sigma^{-1}\big]^{\intercal}(t)\,q_2^*(t)\big]\,dt\\&-\int_0^{\cdot}\big[\big(\theta(t)+\sigma^{-1}(t)v(t)\big)^{\intercal}-{p_2^*}^{-1}(t)\,{q_2^*}^{\intercal}(t)\big]\,dB(t)=:A+M.\end{aligned}\tag{4.30}$$

Hence, we obtain $p_2^* Y^{(y,v)} = x_0 y \exp(A) \exp(M - [M]/2)$. Moreover, it holds that $x_0 y\, \mathcal{E}(M) = x_0 y \exp(M - [M]/2)$ is a local martingale which is even a supermartingale as it is bounded from below and its starting value $x_0 y$ is integrable. Furthermore, it follows from (4.7) and the definition of δ_K that the integrand of A is nonpositive, i.e. A is decreasing.
Now we are in position to verify that $p_2^* Y^{(y,v)}$ is a supermartingale: Obviously, the process is adapted. The integrability follows from the boundedness of $\exp(A)$ by 1 and the remaining part being a supermartingale. Fix $s, t \in [0,T]$ with $s \leq t$. Then we obtain from $x_0 y > 0$, the monotonicity of A and the supermartingale property of the remaining part:

$$\mathbb{E}\big[p_2^*(t)\, Y^{(y,v)}(t) | \mathcal{F}_s\big] \leq x_0 y \exp(A(s))\, \mathbb{E}\big[\mathcal{E}(M)_t | \mathcal{F}_s\big] \leq p_2^*(s)\, Y^{(y,v)}(s).$$

Hence, $p_2^* Y^{(y,v)}$ is indeed a supermartingale. Since it holds by assumption that $p_2^* Y^{(y^*,v^*)}$ is even a martingale (starting in $x_0 y^*$ according to (4.6)) we can conclude

$$\begin{aligned} x_0 y^* - x_0 y &\leq \mathbb{E}\big[p_2^*(T)\, Y^{(y^*,v^*)}(T)\big] - \mathbb{E}\big[p_2^*(T)\, Y^{(y,v)}(T)\big] \\ &= \mathbb{E}\Big[\widetilde{U}'\big(Y^{(y^*,v^*)}(T)\big)\, \big(Y^{(y,v)}(T) - Y^{(y^*,v^*)}(T)\big)\Big] \\ &\leq \mathbb{E}\Big[\widetilde{U}\big(Y^{(y,v)}(T)\big) - \widetilde{U}\big(Y^{(y^*,v^*)}(T)\big)\Big], \end{aligned} \tag{4.31}$$

where the last inequality follows from the convexity of $\widetilde{U}$ according to Lemma 2.7 *(ii)*. Rearranging (4.31) yields

$$x_0 y^* + \mathbb{E}\Big[\widetilde{U}\big(Y^{(y^*,v^*)}(T)\big)\Big] \leq x_0 y + \mathbb{E}\Big[\widetilde{U}\big(Y^{(y,v)}(T)\big)\Big]. \tag{4.32}$$

As we considered an arbitrary pair $(y,v) \in \mathcal{D}$, (4.32) implies the optimality of (y^*, v^*) for the dual problem. Therefore, the proof is completed. □

Remark 4.7. In [16], it is claimed that the process $I_v := \int_0^\cdot \big|\delta_K(v(s) - v^*(s))\big|^2 ds$ is for every $(y^*, v) \in \mathcal{D}$ locally bounded, i.e. there exists a sequence of stopping times $(\tau_{v,n})_{n\in\mathbb{N}}$ with $\tau_{v,n} \nearrow T$ (a.s.) as $n \to \infty$ such that $I_v^{\tau_{v,n}}$ is bounded for each $n \in \mathbb{N}$. However this is in general not true for admissible dual strategies v_1, v_2 as the following example shows: Take $m = 1$ and $K = \mathbb{R}_0^+$ (no short selling of stocks). Then $v_1 :\equiv 1$ and $v_2 :\equiv 2$ are admissible dual controls due to $\sup \mathbb{R}_0^- = 0$. However, we obtain $\delta_K(v_1 - v_2) \equiv \delta_K(-1) = \infty$, whence $\int_0^\cdot \big|\delta_K(v_1(s) - v_2(s))\big|^2 ds$ is not locally bounded. Fortunately, as the arguments presented in *Step 5* show, it suffices to consider admissible dual controls v such that $v - v^*$ is again an admissible dual control process.

Remark 4.8. Besides that, we implemented the following major changes in comparison with [16]: In *Step 3*, we explicitly proved $Y^{(y^*, v^* + \varepsilon(v - v^*))}(T) \to Y^{(y^*,v^*)}(T)$ almost surely for $\varepsilon \searrow 0$ as this takes some care. We optimized the sequence of stopping times $(\tau_n^v)_{n\in\mathbb{N}}$ insofar as our definition in (4.13) makes it straightforward that the stochastic integrals in *Step 5* are in fact true martingales. Moreover, we explicitly consider the special case $v \equiv 0$ in *Step 6* as proving that also (4.8) is a necessary condition requires a different approach than in the previous steps. In *Step 7*, we placed great emphasis on showing that $p_2^* Y^{(y,v)}$ is a supermartingale for every pair $(y,v) \in \mathcal{D}$. Hence, this reminds us of a common concept in optimization theory: If a process from a family of supermartingales, which is indexed by the set of the admissible strategies, is even a true martingale, then the corresponding control is optimal. Similar changes are implemented in the proof of the SMP for the primal problem (cf. Theorem 4.12). Most importantly, we refer to Remark 4.13 for an argument, why our formulation differs fundamentally from the original result in [16].

Remark 4.9. It follows from (3.32) and the strict positivity of $Y^{(y^*,v^*)}$ that the Hamiltonian maximization condition

$$\mathcal{H}_2\big(t, Y^{(y^*,v^*)}(t), v^*(t), p_2^*(t), q_2^*(t)\big) = \sup_{v\in\widetilde{K}} \mathcal{H}_2\big(t, Y^{(y^*,v^*)}(t), v, p_2^*(t), q_2^*(t)\big), \tag{4.33}$$

is equivalent to

$$p_2^*(t)\,\delta_K\big(v^*(t)\big) + \big(\sigma^{-1}(t)\,v^*(t)\big)^\intercal q_2^*(t) = \inf_{v\in\widetilde{K}} \big\{p_2^*(t)\,\delta_K(v) + \big(\sigma^{-1}(t)\,v\big)^\intercal q_2^*(t)\big\}, \tag{4.34}$$

being satisfied almost surely for almost every $t \in [0,T]$. Clearly, (4.8) yields that the left-hand side is 0. Moreover, we obtain from (4.7), p_2^* being strictly positive and the definition of δ_K that the expression within the curly brackets is nonnegative for every $v \in \widetilde{K}$. Hence, (4.33) holds indeed for an optimal dual control process v^*. Furthermore, as *Step 1* shows, (4.6) results from considering the minimization problem $\inf_{y\in\mathbb{R}^+}\big\{x_0 y + \mathbb{E}\big[\widetilde{U}\big(Y^{(y,v^*)}(T)\big)\big]\big\}$ with v^* being fixed. Consequently, Theorem 4.6 can be regarded as a stochastic maximum principle for the dual problem which also takes the optimization with respect to y into account.

4.2 An Analogous Result for the Primal Problem

The aim of this section is deriving a result for characterizing the optimality of a control π^* for the primal problem which is conceptually similar to Theorem 4.6. It will allow us to derive an insightful relationship between solutions to the primal problem and solutions to its associated dual problem (cf. Section 4.3 below). At first, we formulate technical assumptions which we are going to need in the following.

Assumption 4.10. Suppose that we have $U\big(X^\pi(T)\big) \in L^1$ and $X^\pi(T)\,U'\big(X^\pi(T)\big) \in L^2$ for every admissible strategy $\pi \in \mathcal{A}$, where X^π denotes the unique solution to (2.3).

Note that the second condition is trivially satisfied by $U = \log$. This assumption guarantees in the spirit of the second part of Lemma 4.3 that there exists a solution to (4.1).

Lemma 4.11. *Consider $\pi \in \mathcal{A}$ and suppose that Assumption 4.10 is in place. Then there exists a pair (p_1, q_1) solving (4.1) such that $p_1 X^\pi$ is a martingale.*

Proof. As in the proof of Lemma 4.3, we obtain by means of the martingale representation theorem that there exists a continuous version V of the square-integrable martingale

$$M := \Big(\mathbb{E}\big[-X^\pi(T)\,U'\big(X^\pi(T)\big)\big|\mathcal{F}_t\big]\Big)_{t\in[0,T]},$$

which can be written as $M_0 + W \bullet B$, where the process $W \in H^2(0,T;\mathbb{R}^m)$ is unique. Clearly, $p_1 := V/X^\pi$ is well-defined and satisfies the terminal condition associated with

(4.1). Hence, an application of Itô's formula to p_1 shows

$$\begin{aligned} dp_1(t) =& \Big[- p_1(t)\big(r(t) + \pi^\intercal(t)\,\sigma(t)\,\theta(t) - \big|\pi^\intercal(t)\,\sigma(t)\big|^2\big) - \frac{\pi^\intercal(t)\,\sigma(t)\,W(t)}{X^\pi(t)}\Big]\,dt \\ &+ \Big[\frac{W^\intercal(t)}{X^\pi(t)} - p_1(t)\,\pi^\intercal(t)\,\sigma(t)\Big]\,dB(t), \end{aligned} \tag{4.35}$$

$t \in [0,T]$. Therefore, defining a process q_1 for $t \in [0,T]$ by

$$q_1(t) := \frac{W(t)}{X^\pi(t)} - p_1(t)\,\sigma^\intercal(t)\,\pi(t)$$

reduces (4.35) to (4.1). Hence, (p_1, q_1) solves the primal adjoint equation and $p_1 X^\pi = V$ is indeed a martingale, which concludes the proof. □

Now we are in position to formulate the main result of this section.

Theorem 4.12. *Let a utility maximization problem in the setting of Definition 2.4 be given and suppose that Assumption 4.10 is satisfied. Consider an admissible control $\pi^* \in \mathcal{A}$: Let X^{π^*}, p_1^* and q_1^* denote processes which satisfy $X^{\pi^*}(0) = x_0$, $p_1^*(T) = -U'\big(X^{\pi^*}(T)\big)$ and for $t \in [0,T]$:*

$$\begin{aligned} dX^{\pi^*}(t) &= X^{\pi^*}(t)\big[\big(r(t) + \pi^{*\intercal}(t)\,\sigma(t)\,\theta(t)\big)\,dt + \pi^{*\intercal}(t)\,\sigma(t)\,dB(t)\big], \\ dp_1^*(t) &= -\big[\big(r(t) + \pi^{*\intercal}(t)\,\sigma(t)\,\theta(t)\big)\,p_1^*(t) + \pi^{*\intercal}(t)\,\sigma(t)\,q_1^*(t)\big]\,dt + q_1^{*\intercal}(t)\,dB(t), \end{aligned} \tag{4.36}$$

such that $p_1^* X^{\pi^*}$ is even a martingale. Moreover, suppose that at least one of the following statements is true:

(i) $\mathrm{id}_{\mathbb{R}^+} \cdot U'$ *is a nonincreasing function on $\mathbb{R}^+$.*

(ii) Define $\Theta := \big\{(\overline{\pi} - \pi^*)\mathbb{1}_C \,\big|\, \overline{\pi} \in K, C \in \Sigma_p^{\mathbb{F},[0,T]}\big\}$, where $\Sigma_p^{\mathbb{F},[0,T]}$ denotes the progressive σ-algebra accompanying our filtered probability space and $\overline{\pi}$ also stands for the constant control process mapping to $\overline{\pi}$, for notational convenience. For every $\theta \in \Theta$, we have the uniform integrability of the family

$$\big(\Delta_\varepsilon^\theta\big)_{\varepsilon \in (0,1)} := \left(\frac{U\big(X^{\pi^* + \varepsilon\theta}(T)\big) - U\big(X^{\pi^*}(T)\big)}{\varepsilon}\right)_{\varepsilon \in (0,1)}.$$

(iii) For every $\theta \in \Theta$, there exists a random variable ξ_θ with $(\xi_\theta)^- \in L^1$ such that $\Delta_\varepsilon^\theta$ is (a.s.) bounded below by ξ_θ for every $\varepsilon \in (0,1)$.

Then π^* is optimal in the sense of Definition 2.4 if and only if it holds almost surely for almost every $t \in [0,T]$:

$$\pi^{*\intercal}(t)\,\big[-\sigma(t)\,\big(p_1^*(t)\,\theta(t) + q_1^*(t)\big)\big] = \sup_{\pi \in K} \Big\{\pi^\intercal\,\big[-\sigma(t)\,\big(p_1^*(t)\,\theta(t) + q_1^*(t)\big)\big]\Big\}. \tag{4.37}$$

Proof. At first, we recall that Lemma 4.11 guarantees the existence of processes X^{π^*}, p_1^* and q_1^* which meet the requirements.

Step 1. Necessary Condition: (4.37) *is satisfied:* $\Phi_\pi(\varepsilon) := U\big(X^{\pi^*+\varepsilon(\pi-\pi^*)}(T)\big)$, $\varepsilon \in [0,1]$, *is right differentiable at* $\varepsilon = 0$ *and the corresponding derivative is almost surely given by* $U'\big(X^{\pi^*}(T)\big)\, X^{\pi^*}(T)\, H_\pi(T)$, *for every* $\pi \in \mathcal{A}$.

Let $\pi^* \in \mathcal{A}$ be an optimal control and consider an arbitrary, but fixed, control $\pi \in \mathcal{A}$. Since K is a convex set, it follows that the convex combination $\pi^* + \varepsilon(\pi - \pi^*)$ is for every $\varepsilon \in [0,1]$ again an admissible control. Hence, Φ_π is well-defined for almost every $\omega \in \Omega$. Moreover, due to the structure of (2.3), $X^{\pi^*+\varepsilon(\pi-\pi^*)}(T)$ is explicitly given by

$$
\begin{aligned}
X^{\pi^*+\varepsilon(\pi-\pi^*)}(T) = x_0 \exp\Bigg(\int_0^T &\big(r(t) + \big(\pi^*(t) + \varepsilon(\pi(t) - \pi^*(t))\big)^\intercal \sigma(t)\,\theta(t)\big)\, dt \\
&- \frac{1}{2}\int_0^T \big|\big(\pi^*(t) + \varepsilon(\pi(t) - \pi^*(t))\big)^\intercal \sigma(t)\big|^2\, dt \\
&+ \int_0^T \big(\pi^*(t) + \varepsilon(\pi(t) - \pi^*(t))\big)^\intercal \sigma(t)\, dB(t)\Bigg).
\end{aligned} \tag{4.38}
$$

Fix $\delta \in \mathbb{R}^-$. We consider a function g_π on $(\delta, 1]$ which is defined by $\log\big(X^{\pi^*+\varepsilon(\pi-\pi^*)}(T)\big) =: g_\pi(\varepsilon)$ for $\varepsilon \in [0,1]$ and for the remaining points of the domain by the logarithm of the right-hand side of (4.38), which is also meaningful for negative ε. As g_π is a polynomial in ε, it follows immediately that it is differentiable with respect to ε. Its derivative at $\varepsilon = 0$ is given by $H_\pi(T)$, where the process H_π is defined as follows:

$$
\begin{aligned}
H_\pi := &\int_0^\cdot \big[\big(\pi(t) - \pi^*(t)\big)^\intercal \sigma(t)\,\theta(t) - \big(\pi(t) - \pi^*(t)\big)^\intercal \sigma(t)\,\sigma^\intercal(t)\,\pi^*(t)\big]\, dt \\
&+ \int_0^\cdot \big(\pi(t) - \pi^*(t)\big)^\intercal \sigma(t)\, dB(t).
\end{aligned} \tag{4.39}
$$

Therefore, we obtain from the chain rule and $\Phi_\pi = (U \circ \exp \circ g_\pi)\big|_{[0,1]}$:

$$
\lim_{\varepsilon \searrow 0} \Delta_\varepsilon \Phi_\pi = U'\big(X^{\pi^*}(T)\big)\, X^{\pi^*}(T)\, H_\pi(T), \quad \text{a.s.}, \tag{4.40}
$$

where $\Delta_\varepsilon\Phi_\pi$ denotes the difference quotient of Φ_π over $[0,\varepsilon]$. It is important to place emphasis on the fact that, in contrast to the situation for the dual problem, g_π is already differentiable with respect to ε. Hence, there is no necessity for searching for a converging estimate as in *Step 3* from the proof of Theorem 4.6.

Step 2. Necessary Condition: (4.37) *is satisfied: Localization by appropriate stopping times.*

For every $\pi \in \mathcal{A}$, we define a sequence of stopping times $(\tau_n^\pi)_{n\in\mathbb{N}}$ via

$$
\begin{aligned}
\tau_n^\pi := \inf\bigg\{t \ge 0 \;:\; &\bigg|\int_0^t p_1^*(s)\, X^{\pi^*}(s)\, \big(\pi(s) - \pi^*(s)\big)^\intercal \sigma(s)\, dB(s)\bigg| \ge n\bigg\} \\
&\wedge \inf\big\{t \ge 0 \;:\; |H_\pi(t)| \ge n\big\} \;\wedge\; T,
\end{aligned} \tag{4.41}
$$

for $n \in \mathbb{N}$. These stopping times serve exactly the same purpose as in the proof of Theorem 4.6 (cf. (4.13)). As the processes within the norm $|\cdot|$ are continuous semimartingales, we

can conclude again that $\tau_n^\pi \nearrow T$ holds almost surely as $n \to \infty$.
Fix $n \in \mathbb{N}$. Clearly, $\pi^* + \mathbb{1}_{[0,\tau_n^\pi]}\,\varepsilon\,(\pi - \pi^*)$ defines an admissible control as well, since it corresponds pointwise to convex combinations of elements of K. Hence, the arguments presented in *Step 1* are also applicable, if we replace $\pi^* + \varepsilon(\pi - \pi^*)$ with the previously introduced control process. Note that the derivative at $\varepsilon = 0$ of the adjusted right-hand side of (4.38) is exactly given by $H_\pi^{\tau_n^\pi}(T)$ since the occurring integrands are, in comparison with the original result, precisely multiplied by $\mathbb{1}_{[0,\tau_n^\pi]}$. Moreover, we define the function $\Phi_\pi^n(\varepsilon) := U\big(X^{\pi^*+\mathbb{1}_{[0,\tau_n^\pi]}\,\varepsilon\,(\pi-\pi^*)}(T)\big)$, $\varepsilon \in [0,1]$, and

$$\Delta_\varepsilon \Phi_\pi^n := \frac{U\big(X^{\pi^*+\mathbb{1}_{[0,\tau_n^\pi]}\,\varepsilon\,(\pi-\pi^*)}(T)\big) - U\big(X^{\pi^*}(T)\big)}{\varepsilon}, \tag{4.42}$$

for each $\varepsilon \in (0,1]$. Hence, it follows from our previous considerations for any $n \in \mathbb{N}$:

$$\lim_{\varepsilon \searrow 0} \Delta_\varepsilon \Phi_\pi^n = U'\big(X^{\pi^*}(T)\big)\, X^{\pi^*}(T)\, H_\pi^{\tau_n^\pi}(T), \quad \text{a.s.} \tag{4.43}$$

Step 3. Necessary Condition: (4.7) *is satisfied: Proving* $\mathbb{E}\big[U'\big(X^{\pi^*}(T)\big)\, X^{\pi^*}(T)\, H_\pi^{\tau_n^\pi}(T)\big] \le 0$ *under the premise of (i), (ii) or (iii), respectively.*

(i): At first, we notice that $g_\pi^n(\varepsilon) := \log\big(X^{\pi^*+\mathbb{1}_{[0,\tau_n^\pi]}\,\varepsilon\,(\pi-\pi^*)}(T)\big)$, $\varepsilon \in [0,1]$, is a concave function for almost every $\omega \in \Omega$ (cf. an adjusted version of (4.38)). Furthermore, it is an immediate consequence of Condition *(i)* that $\exp \cdot (U' \circ \exp)$ is nonincreasing as well. Additionally, we have that $U \circ \exp$ is a nondecreasing function on $\mathbb{R}$. Hence, it follows that $U \circ \exp$ is a concave and nondecreasing function. Lemma 4.4, therefore, guarantees that $\Phi_\pi^n = U \circ \exp \circ g_\pi^n$ is a concave function. This observation shows that $(\Delta_\varepsilon \Phi_\pi^n)_{\varepsilon \in (0,1]}$ is nonincreasing. Hence, this family is almost surely bounded from below by the integrable (cf. Assumption 4.10) random variable $U\big(X^{\pi^*+\mathbb{1}_{[0,\tau_n^\pi]}\,(\pi-\pi^*)}(T)\big) - U\big(X^{\pi^*}(T)\big)$. Therefore, we obtain from the optimality of π^*, the monotone convergence theorem and (4.43):

$$0 \ge \lim_{\varepsilon \searrow 0} \mathbb{E}\big[\Delta_\varepsilon \Phi_\pi^n\big] = \mathbb{E}\big[U'\big(X^{\pi^*}(T)\big)\, X^{\pi^*}(T)\, H_\pi^{\tau_n^\pi}(T)\big], \tag{4.44}$$

for every $n \in \mathbb{N}$. Clearly, this implies

$$\mathbb{E}\big[U'\big(X^{\pi^*}(T)\big)\, X^{\pi^*}(T)\, H_\pi^{\tau_n^\pi}(T)\big] \le 0, \tag{4.45}$$

where the integral on the left-hand side is finite due to (4.41) and Assumption 4.10.
(ii): As we shall see in *Step 4* below, it is sufficient to consider controls of the form $\pi = \overline{\pi}\,\mathbb{1}_C + \pi^*\,\mathbb{1}_{C^c}$ with a constant control $\overline{\pi}$ and $C \in \Sigma_p^{\mathbb{F},[0,T]}$. Hence, we restrict ourselves to these controls in the following. For a control π with the above structure, we obtain $\mathbb{1}_{[0,\tau_n^\pi]}\,(\pi - \pi^*) = \mathbb{1}_{[0,\tau_n^\pi]}\,\mathbb{1}_C\,(\overline{\pi} - \pi^*)$. We notice that $\mathbb{1}_{[0,\tau_n^\pi]}$ is progressively measurable because its paths are left-continuous and $\{\omega \in \Omega \mid \mathbb{1}_{[0,\tau_n^\pi(\omega)]}(t) = 0\} = \{\tau_n^\pi < t\} \in \mathcal{F}_t$ holds for every $t \in [0,T]$ as τ_n^π is in particular a weak stopping time. Hence, it follows from Condition *(ii)* that $(\Delta_\varepsilon \Phi_\pi^n)_{\varepsilon \in (0,1)}$ is uniformly integrable for every $n \in \mathbb{N}$. As $\mathbb{P}$ is in particular a finite measure, the convergence in (4.43) also holds in probability. Combining this with the optimality of π^* and Vitali's convergence theorem proves the validity of (4.44) and (4.45) also in this case.

(iii): We recall from the previous paragraph that $\mathbb{1}_{[0,\tau_n^\pi]}(\pi - \pi^*) \in \Theta$ holds for controls of the form $\pi = \overline{\pi}\,\mathbb{1}_C + \pi^*\,\mathbb{1}_{C^c}$ as specified above. Hence, Condition *(iii)* guarantees the existence of a random variable ξ with $\xi^- \in L^1$ which is (a.s.) a lower bound for the elements of $(\Delta_\varepsilon \Phi_\pi^n)_{\varepsilon\in(0,1)}$. Therefore, we obtain from the optimality of π^*, Fatou's lemma and (4.43):

$$0 \geq \liminf_{\varepsilon\searrow 0} \mathbb{E}\big[\Delta_\varepsilon \Phi_\pi^n\big] \geq \mathbb{E}\Big[\liminf_{\varepsilon\searrow 0} \Delta_\varepsilon \Phi_\pi^n\Big] = \mathbb{E}\big[U'\big(X^{\pi^*}(T)\big)\,X^{\pi^*}(T)\,H_\pi^{\tau_n^\pi}(T)\big], \tag{4.46}$$

which implies again (4.45).

Step 4. Necessary Condition: (4.37) *is satisfied: Applying Itô's formula to the left-hand side of* (4.45) *and concluding the proof by considering certain strategies $\pi \in \mathcal{A}$.*

By means of a similar argument as in *Step 5* from the proof of Theorem 4.6, we obtain that $p_1^* X^{\pi^*}$ is indistinguishable from the process V^* which we encountered in the course of the proof of Lemma 4.11. On the one hand this observation facilitates the application of Itô's formula below and on the other hand it implies that $p_1^* X^{\pi^*}$ (and, therefore, also p_1^*) is a strictly negative process. The latter will play a decisive role below, especially in *Step 5*, where this property is also applicable since we did not use the optimality of π^* for its proof. We obtain for every $n \in \mathbb{N}$ by means of Itô's formula

$$\begin{aligned}
d\big(p_1^* X^{\pi^*} H_\pi^{\tau_n^\pi}\big)(t) =& p_1^*(t)\,X^{\pi^*}(t)\,\mathbb{1}_{[0,\tau_n^\pi]}(t)\,\big(\pi(t) - \pi^*(t)\big)^\intercal \sigma(t)\,\big(\theta(t) - \sigma^\intercal(t)\,\pi^*(t)\big)\,dt \\
&+ \mathbb{1}_{[0,\tau_n^\pi]}(t)\,\big(\pi(t) - \pi^*(t)\big)^\intercal \sigma(t)\,\big(p_1^*(t)\,X^{\pi^*}(t)\,\sigma^\intercal(t)\,\pi^*(t) + X^{\pi^*}(t)\,q_1^*(t)\big)\,dt \\
&+ p_1^*(t)\,X^{\pi^*}(t)\,\mathbb{1}_{[0,\tau_n^\pi]}(t)\,\big(\pi(t) - \pi^*(t)\big)^\intercal \sigma(t)\,dB(t) \\
&+ \big(p_1^*(t)\,X^{\pi^*}(t)\,\pi^{*\intercal}(t)\,\sigma(t) + X^{\pi^*}(t)\,q_1^{*\intercal}(t)\big)\,H_\pi^{\tau_n^\pi}(t)\,dB(t),
\end{aligned}$$

$t \in [0,T]$. We observe that the integrand of the finite variation part can be simplified to

$$X^{\pi^*}(t)\,\mathbb{1}_{[0,\tau_n^\pi]}(t)\,\big(\pi(t) - \pi^*(t)\big)^\intercal \sigma(t)\,\big(p_1^*(t)\,\theta(t) + q_1^*(t)\big), \quad t \in [0,T]. \tag{4.47}$$

Note that the process which is multiplied by $H_\pi^{\tau_n^\pi}$ in the dynamics above corresponds exactly to $W^{*\intercal}$ from the proof of Lemma 4.11, which lies in $H^2(0,T;\mathbb{R}^m)$. Hence, it follows from (4.41) that the local martingale part defines in fact a true martingale. Combining this with $H_\pi^{\tau_n^\pi}(0) = 0$ and (4.45) shows for every $\pi \in \mathcal{A}$ for which the conclusion of *Step 3* holds:

$$0 \geq \mathbb{E}\bigg[\int_0^{\tau_n^\pi} -X^{\pi^*}(t)\,\big(\pi(t) - \pi^*(t)\big)^\intercal \sigma(t)\,\big(p_1^*(t)\,\theta(t) + q_1^*(t)\big)\,dt\bigg], \tag{4.48}$$

for every $n \in \mathbb{N}$. We conclude this segment of the proof by arguing that (4.48) necessarily implies (4.37). Fix $\overline{\pi} \in K$ and consider the set

$$N^{\overline{\pi}} := \big\{(\omega,t) \in \Omega \times [0,T] : -\big(\overline{\pi} - \pi^*(t)\big)^\intercal \sigma(t)\,\big(p_1^*(t)\,\theta(t) + q_1^*(t)\big) > 0\big\}. \tag{4.49}$$

Moreover, we define the control process $\pi^{\overline{\pi}} := \overline{\pi}\,\mathbb{1}_{N^{\overline{\pi}}} + \pi^*\,\mathbb{1}_{(N^{\overline{\pi}})^c}$. The admissibility of $\pi^{\overline{\pi}}$ results from $\pi^* \in \mathcal{A}$ and $N^{\overline{\pi}}$ being measurable with respect to the progressive σ-algebra. It is an immediate consequence of the special structure of $\pi^{\overline{\pi}}$ that *Step 3* (and, therefore, also (4.48)) is applicable to $\pi^{\overline{\pi}}$ under each of the three Conditions *(i)*, *(ii)* and *(iii)*. As

X^{π^*} is a strictly positive process, we necessarily obtain from (4.48) applied to $\pi^{\overline{\pi}}$ that, for every $n \in \mathbb{N}$, the set

$$N^{\overline{\pi}} \cap \{(\omega, t) \in \Omega \times [0,T] : \mathbb{1}_{[0,\tau_n^{\pi^{\overline{\pi}}}]}(t) = 1\} \tag{4.50}$$

has to be a $\mathbb{P} \otimes \lambda|_{[0,T]}$-null set. Hence, as $\mathbb{1}_{[0,\tau_n^{\pi^{\overline{\pi}}}]} \nearrow \mathbb{1}_{[0,T]}$ holds almost surely for $n \to \infty$ and $\mathbb{P} \otimes \lambda|_{[0,T]}$ is continuous from below, we can conclude that also $N^{\overline{\pi}}$ is a $\mathbb{P} \otimes \lambda|_{[0,T]}$-null set. Moreover, we obtain from the subadditivity property that even

$$N := \bigcup_{\overline{\pi} \in K \cap \mathbb{Q}^m} N^{\overline{\pi}}$$

is a $\mathbb{P} \otimes \lambda|_{[0,T]}$-null set. It follows from $K \cap \mathbb{Q}^m$ being dense in K and the continuity of the Euclidean inner product in every component that we have for $\mathbb{P} \otimes \lambda|_{[0,T]}$-almost every $(\omega, t) \in \Omega \times [0,T]$ and every $\overline{\pi} \in K$:

$$-\left(\overline{\pi} - \pi^*(t)\right)^\intercal \sigma(t) \left(p_1^*(t)\, \theta(t) + q_1^*(t)\right) \leq 0, \tag{4.51}$$

which is equivalent to (4.37).

Step 5. Sufficient Condition: π^* is optimal: Proving that $p_1^* X^\pi$ is a submartingale for every $\pi \in \mathcal{A}$ and applying the concavity of U.

Finally, we show by means of an argument, which is similar to *Step 7* from the proof of Theorem 4.6, that (4.37) is also sufficient for the optimality of π^*. Let $\pi^* \in \mathcal{A}$ be given and assume that (4.37) is satisfied. We obtain for every $\pi \in \mathcal{A}$ by means of the integration by parts formula:

$$\begin{aligned} p_1^* X^\pi = {} & p_1^*(0)\, x_0 - \int_0^\cdot X^\pi(t) \left[\left(r(t) + \pi^{*\intercal}(t)\, \sigma(t)\, \theta(t)\right) p_1^*(t) + \pi^{*\intercal}(t)\, \sigma(t)\, q_1^*(t)\right] dt \\ & + \int_0^\cdot X^\pi(t) \left[\left(r(t) + \pi^\intercal(t)\, \sigma(t)\, \theta(t)\right) p_1^*(t) + \pi^\intercal(t)\, \sigma(t)\, q_1^*(t)\right] dt \\ & + \int_0^\cdot X^\pi(t) \left(p_1^*(t)\, \pi^\intercal(t)\, \sigma(t) + q_1^{*\intercal}(t)\right) dB(t). \end{aligned} \tag{4.52}$$

As the process $p_1^* X^\pi$ is strictly negative (cf. *Step 4* for the strict negativity of p_1^*), it follows that it can be written as $p_1^*(0)\, x_0\, \mathcal{E}(Z)$, where the continuous semimartingale Z is given by

$$\begin{aligned} Z = {} & \int_0^\cdot \left(\pi^\intercal(t) - \pi^{*\intercal}(t)\right) \sigma(t) \left(\theta(t) + {p_1^*}^{-1}(t)\, q_1^*(t)\right) dt \\ & + \int_0^\cdot \left[\pi^\intercal(t)\, \sigma(t) + {p_1^*}^{-1}(t)\, q_1^{*\intercal}(t)\right] dB(t) =: A + M. \end{aligned} \tag{4.53}$$

Therefore, we obtain the explicit representation $p_1^*(0)\, x_0\, \mathcal{E}(M) \exp(A)$. Clearly, $\mathcal{E}(M)$ is a supermartingale as it is a nonnegative local martingale with integrable starting value. Hence, it is an immediate consequence of $p_1^*(0) < 0$ that $p_1^*(0)\, x_0\, \mathcal{E}(M)$ is a submartingale. Moreover, it follows from p_1^* being strictly negative and (4.37) that the integrand of A is nonpositive, i.e. the process $\exp(A)$ is decreasing. Obviously, $p_1^* X^\pi$ is adapted and L^1.

The second property essentially results from the above representation and $\exp(A)$ being bounded by 1. Fix $s, t \in [0,T]$ with $s \leq t$. The submartingale property follows from the monotonicity of $\exp(A)$ and $p_1^*(0)\, x_0\, \mathcal{E}(M)$ being a negative submartingale:

$$\mathbb{E}\big[p_1^*(t)\, X^\pi(t) | \mathcal{F}_s\big] \geq \exp(A(s))\, \mathbb{E}\big[p_1^*(0)\, x_0\, \mathcal{E}(M)_t | \mathcal{F}_s\big] \geq p_1^*(s)\, X^\pi(s). \tag{4.54}$$

Finally, as $p_1^*\, X^{\pi^*}$ is even a martingale, we obtain from the concavity of U:

$$\begin{aligned} 0 = p_1^*(0)\, x_0 - p_1^*(0)\, x_0 &\geq \mathbb{E}\big[p_1^*(T)\, X^{\pi^*}(T)\big] - \mathbb{E}\big[p_1^*(T)\, X^\pi(T)\big] \\ &= \mathbb{E}\big[U'\big(X^{\pi^*}(T)\big)\, \big(X^\pi(T) - X^{\pi^*}(T)\big)\big] \\ &\geq \mathbb{E}\big[U\big(X^\pi(T)\big) - U\big(X^{\pi^*}(T)\big)\big]. \end{aligned} \tag{4.55}$$

Since $\pi \in \mathcal{A}$ was arbitrary, we can conclude that π^* is indeed optimal. $\square$

Note that our set of assumptions differs fundamentally from [16].

Remark 4.13. In [16], the proof of (4.37) being a necessary optimality condition is carried out under Assumption 2.2. However, in contrast to *Step 4* of Theorem 4.6, this is not sufficient for concluding that the family of difference quotients $(\Delta_\varepsilon \Phi_\pi^n)_{\varepsilon \in (0,1]}$ is monotone. The key difference is that $\widetilde{U} \circ \exp$ is decreasing, whereas $U \circ \exp$ is increasing. Note that both functions are convex under Assumption 2.2 (cf. Lemma 2.8). Furthermore, g_π^n as defined in *Step 3* is always concave. In the case of $U \circ \exp$ it is, therefore, in general not true that Φ_π^n is either concave or convex as the inequalities (1) and (2) from the proof of Lemma 4.4 cannot be combined by the transitive property anymore.
However, as the second statement of Lemma 4.4 shows, it is possible to resolve this issue, if $U \circ \exp$ is concave. Hence, in contrast to Assumption 2.2 (an important assumption for Theorem 4.6), we have to require that $\mathrm{id}_{\mathbb{R}^+} \cdot U'$ is a nonincreasing function. Note that this assumption is also formulated in [21]. Since the only utility functions which satisfy the aforementioned condition and Assumption 2.2 at the same time are given by $c \cdot \log + d$, where $c \in \mathbb{R}^+$ and $d \in \mathbb{R}$ are fixed, we formulate by *(ii)* and *(iii)* alternative conditions which justify interchanging limit and expectation operator in *Step 3*. Hence, other utility functions U, in particular those where $\mathrm{id}_{\mathbb{R}^+} \cdot U'$ is strictly increasing, are a priori not excluded from Theorem 4.12.

Remark 4.14. Note that (4.37) is equivalent to $-\sigma(t)\, \big(p_1^*(t)\, \theta(t) + q_1^*(t)\big)$ being an element of the normal cone of K at $\pi^*(t)$ almost surely for almost every $t \in [0,T]$, i.e.

$$\forall \pi \in K: \quad \big(\pi - \pi^*(t)\big)^\intercal \big[-\sigma(t)\, \big(p_1^*(t)\, \theta(t) + q_1^*(t)\big)\big] \leq 0. \tag{4.56}$$

Remark 4.15. It is an immediate consequence of (3.31) and the strict positivity of X^{π^*} that (4.37) is equivalent to

$$-\mathcal{H}_1\big(t, X^{\pi^*}(t), \pi^*(t), p_1^*(t), q_1^*(t)\big) = -\inf_{\pi \in K} \mathcal{H}_1\big(t, X^{\pi^*}(t), \pi, p_1^*(t), q_1^*(t)\big), \tag{4.57}$$

almost surely for almost every $t \in [0,T]$, i.e. π^* minimizes $\mathcal{H}_1$. Hence, as the primal problem is a maximization problem and Subsection 3.3.1, in particular Theorem 3.16, is devoted to minimization problems, it is hereby justified to view Theorem 4.12 as a stochastic maximum principle. Note that (4.1) actually corresponds to the adjoint equation in a minimization setting. Since $\mathcal{H}_1$ is linear in y and z, it follows that $(-p_1^*, -q_1^*)$ solves (3.29). Hence, (4.37) would become a maximization condition for $\mathcal{H}_1$, if we replaced (4.1) with (3.29) in the formulation of Theorem 4.12.

Remark 4.16. We observe that neither of the Conditions *(i)*, *(ii)* and *(iii)* is necessary for performing the arguments in *Step 5* above. These were just required in order to ensure that we can interchange limit and expectation operator in *Step 3*. This is an important observation with regard to Theorem 4.17 below. Clearly, the same applies to the requirement that $\mathrm{id}_{\mathbb{R}+} \cdot U'$ is nondecreasing in the light of *Step 7* from Theorem 4.6.

4.3 Constructing a Solution to the Primal Problem by Means of a Solution to Its Dual Problem

As the deep SMP algorithm (cf. Section 4.4 below) is formulated for solving the dual problem by means of Theorem 4.6, it is natural to ask whether the obtained results are also meaningful for the corresponding primal problem.
The following theorem (cf. [16] for a similar result) proves by means of Theorems 4.6 and 4.12 and the technical assumption (4.58) that we can construct a solution to the primal problem, if there exists an optimal pair for the associated dual problem.

Theorem 4.17. *Let a dual problem according to Definition 2.11 be given such that Assumptions 2.2 and 4.1 and either $U(0) > -\infty$ or $U = \log$ hold. Assume that there exists an optimal pair $(y^*, v^*) \in \mathcal{D}$. Let $Y^{(y^*,v^*)}$, p_2^* and q_2^* denote processes which solve (4.5) and suppose that $p_2^* Y^{(y^*,v^*)}$ is a martingale (cf. Theorem 4.6). Moreover, assume*

$$\pi^* := p_2^{*\,-1} \left[\sigma^{-1}\right]^\intercal q_2^* \in H^2(0,T;\mathbb{R}^m). \tag{4.58}$$

Then it follows that π^* is an optimal control for the corresponding primal problem. Furthermore, a triple of processes solving (4.36) for π^* is explicitly given by

$$X^{\pi^*} := p_2^*, \quad p_1^* := -Y^{(y^*,v^*)} \quad \textit{and} \quad q_1^* := Y^{(y^*,v^*)}(\sigma^{-1} v^* + \theta). \tag{4.59}$$

Proof. Since (y^*, v^*) is optimal and the requirements of Theorem 4.6 are fulfilled, we obtain that (4.6), (4.7) and (4.8) hold for p_2^* and q_2^*. Hence, combining (4.58) with (4.7) proves $\pi^* \in \mathcal{A}$. Plugging π^* and X^{π^*} according to (4.58) and (4.59), respectively, into the right-hand side of (2.3) and simplifying the obtained expression shows

$$\begin{aligned} &p_2^*(t)\left[\big(r(t) + p_2^{*\,-1}(t)\, q_2^{*\intercal}(t)\, \sigma^{-1}(t)\, \sigma(t)\, \theta(t)\big)\, dt + p_2^{*\,-1}(t)\, q_2^{*\intercal}(t)\, \sigma^{-1}(t)\, \sigma(t)\, dB(t)\right] \\ &= \big(r(t)\, p_2^*(t) + q_2^{*\intercal}(t)\, \theta(t)\big)\, dt + q_2^{*\intercal}(t)\, dB(t) \\ &= dp_2^*(t) = dX^{\pi^*}(t), \end{aligned} \tag{4.60}$$

where the second to last equality follows from combining (4.2) and (4.8). Moreover, plugging π^*, p_1^* and q_1^* as defined in (4.58) and (4.59), respectively, into the right-hand side of

(4.1) and applying (4.8) and (2.12) leads to

$$\begin{aligned}
&- \Big[\big(r(t) + {p_2^*}^{-1}(t)\, {q_2^*}^{\intercal}(t)\, \sigma^{-1}(t)\, \sigma(t)\, \theta(t)\big)\, \big(- Y^{(y^*,v^*)}(t)\big) \\
&\qquad + {p_2^*}^{-1}(t)\, {q_2^*}^{\intercal}(t)\, \sigma^{-1}(t)\, \sigma(t)\, Y^{(y^*,v^*)}(t)\, \big(\sigma^{-1}(t)\, v^*(t) + \theta(t)\big)\Big]\, dt \\
&+ Y^{(y^*,v^*)}(t)\, \big(\sigma^{-1}(t)\, v^*(t) + \theta(t)\big)^{\intercal}\, dB(t) \qquad (4.61)\\
&= Y^{(y^*,v^*)}(t)\, \Big[\big(r(t) - {p_2^*}^{-1}(t)\, {q_2^*}^{\intercal}(t)\, \sigma^{-1}(t)\, v^*(t)\big)\, dt + \big(\sigma^{-1}(t)\, v^*(t) + \theta(t)\big)^{\intercal}\, dB(t)\Big] \\
&= d\big(- Y^{(y^*,v^*)}\big)(t) = dp_1^*(t).
\end{aligned}$$

It is an immediate consequence of (4.6) that X^{π^*} starts in x_0. Furthermore, it can be shown by means of Lemma 2.7 *(iv)* that p_1^* satisfies the required terminal condition:

$$p_1^*(T) = -Y^{(y^*,v^*)}(T) = -U'\big(- \widetilde{U}'\big(Y^{(y^*,v^*)}(T)\big)\big) = -U'\big(p_2^*(T)\big) = -U'\big(X^{\pi^*}(T)\big). \qquad (4.62)$$

Hence, combining this with (4.60) and (4.61) proves that the processes defined in (4.59) solve the FBSDE system (4.36) for π^* as given in (4.58).
Note that $p_1^* X^{\pi^*} = -p_2^* Y^{(y^*,v^*)}$ is a martingale since $p_2^* Y^{(y^*,v^*)}$ is a martingale by assumption. Moreover, we obtain from (4.59):

$$\begin{aligned}
-\sigma(t)\, \big(p_1^*(t)\, \theta(t) + q_1^*(t)\big) &= -\sigma(t)\, \big[- Y^{(y^*,v^*)}(t)\, \theta(t) + Y^{(y^*,v^*)}(t)\, \big(\sigma^{-1}(t)\, v^*(t) + \theta(t)\big)\big] \\
&= -Y^{(y^*,v^*)}(t)\, v^*(t), \quad t \in [0,T].
\end{aligned}$$

Therefore, the strict positivity of $Y^{(y^*,v^*)}$ implies that (4.37) is equivalent to

$$\forall \pi \in K: \quad -\pi^{\intercal}\, v^*(t) \leq -{\pi^*}^{\intercal}(t)\, v^*(t) = -{p_2^*}^{-1}(t)\, {q_2^*}^{\intercal}(t)\, \sigma^{-1}(t)\, v^*(t), \qquad (4.63)$$

being satisfied almost surely for almost every $t \in [0,T]$. We observe that the right-hand side of (4.63) corresponds precisely to $\delta_K(v^*(t))$ (cf. (4.8)). Hence, the validity of the aforementioned statement is an immediate consequence of (2.9) and (4.7).
In conclusion, we obtain from Theorem 4.12 that π^* is an optimal control as we have proved that the requirements for (4.37) being a sufficient optimality condition are fulfilled. Note that neither of the Conditions *(i)*, *(ii)* and *(iii)* is necessary for this part of the statement of Theorem 4.12 (cf. Remark 4.16). □

By means of the above result, we can quite easily prove that the so-called strong duality property, i.e. $V = \widetilde{V}$, holds. This means that the value of the primal problem agrees with the one of the associated dual problem.

Corollary 4.18. *Consider a utility maximization problem and its dual problem and suppose that the requirements of Theorem 4.17 are met. Then the strong duality property holds.*

Proof. Due to our choice of the prerequisites we obtain from Theorem 4.17 that there also exists an optimal control for the primal problem whose associated state process is explicitly given by p_2^*. Hence, we are required to prove

$$\mathbb{E}\big[U\big(p_2^*(T)\big)\big] = x_0 y^* + \mathbb{E}\big[\widetilde{U}\big(Y^{(y^*,v^*)}(T)\big)\big]. \qquad (4.64)$$

Combining $p_2^*(T) = -\widetilde{U}'\big(Y^{(y^*,v^*)}(T)\big)$ with Lemma 2.7 *(iii)* and *(iv)* on the left-hand side of (4.64) shows the equivalence with

$$\mathbb{E}\big[\widetilde{U}\big(Y^{(y^*,v^*)}(T)\big) + p_2^*(T)\,Y^{(y^*,v^*)}(T)\big] = x_0 y^* + \mathbb{E}\big[\widetilde{U}\big(Y^{(y^*,v^*)}(T)\big)\big]. \tag{4.65}$$

As $p_2^* Y^{(y^*,v^*)}$ is a martingale and $p_2^*(0) = x_0$ according to (4.6) from Theorem 4.6, we obtain

$$\mathbb{E}\big[p_2^*(T)\,Y^{(y^*,v^*)}(T)\big] = x_0 y^*, \tag{4.66}$$

which concludes the proof as (4.66) is equivalent to (4.65). □

In conclusion, we gather from the previous results that it is indeed reasonable to utilize the results of the deep SMP algorithm also for a study of the original problem. Furthermore, (4.64) suggests a method for obtaining high quality upper and lower estimates for the true value of the dual problem (cf. (4.73)). This is important as, in contrast to the deep controlled 2BSDE algorithm, the deep SMP algorithm does not provide a natural estimate for the value of the dual problem by means of the initial values of the processes from (4.5). As we shall see below, the same applies to the deep primal SMP algorithm.

4.4 Formulation of the Deep SMP Algorithm

In this section, we are going to formulate the deep SMP algorithm (cf. [5]) which combines the BSDE solver from [6] with the optimality conditions on the dual control (y^*, v^*) from Theorem 4.6.

As in Section 3.4, we fix an equidistant time discretization $(t_i)_{i\in\{0,\dots,N\}}$ of $[0,T]$ with step size $\Delta t := T/N$, where $N \in \mathbb{N}$ is fixed. We consider the system (4.5) from Theorem 4.6. In the following, we simulate this system by means of a forward scheme in discrete time leading to processes $(p_i)_{i\in\{0,\dots,N\}}$ and $(Y_i)_{i\in\{0,\dots,N\}}$. Moreover, the random variables $v(t_i)$ and $q_2(t_i)$ are in our scheme, for every $i \in \{0,\dots,N-1\}$, replaced by

$$v_i := \mathcal{N}_{\theta_{i,v}}(Y_i) \quad\text{and}\quad q_i := p_i\,\sigma^\intercal(t_i)\,s_K\big(\mathcal{N}_{\theta_{i,q}}(Y_i)\big), \tag{4.67}$$

where the neural networks $\mathcal{N}_{\theta_{i,v}} : \mathbb{R} \to \mathbb{R}^m$ and $\mathcal{N}_{\theta_{i,q}} : \mathbb{R} \to \mathbb{R}^m$ are parameterized by vectors $\theta_{i,v}$ and $\theta_{i,q}$, respectively. The map $s_K : \mathbb{R}^m \to K$ is a fixed surjective and almost everywhere differentiable function which, therefore, ensures that (4.7) is satisfied. Moreover, like in the deep controlled 2BSDE algorithm, hard constraints guarantee via the final layer of $\mathcal{N}_{\theta_{i,v}}$ that every v_i is $\widetilde{K}$-valued (cf. Chapter 5 for details). For $i = 0$, we simplify the network architectures as discussed in Remark 3.26. The initial values of our scheme are given by

$$Y_0 := y \quad\text{and}\quad p_0 := x_0, \tag{4.68}$$

where the second assignment is motivated by (4.6). Hence, in contrast to the deep controlled 2BSDE algorithm, the initial value of the BSDE part is known. Let $(\Delta B_i)_{i\in\{0,\dots,N-1\}}$ be a family of i.i.d. m-dimensional, centered normally distributed random vectors with covariance matrix $\Delta t \cdot I_m$. In the spirit of (4.5) and (4.8), this allows us to define

$$\begin{aligned} Y_{i+1} &:= Y_i - Y_i\big(r(t_i) + \delta_K(v_i)\big)\,\Delta t - Y_i\big(\theta(t_i) + \sigma^{-1}(t_i)\,v_i\big)^\intercal\,\Delta B_i,\\ p_{i+1} &:= p_i + \big(r(t_i)\,p_i + \theta^\intercal(t_i)\,q_i\big)\,\Delta t + q_i^\intercal\,\Delta B_i, \end{aligned} \tag{4.69}$$

for every $i \in \{0, \dots, N-1\}$. Since $(p_i)_{i\in\{0,\dots,N\}}$ has to satisfy a terminal condition, we wish to find parameters which minimize the corresponding L^2-error

$$\mathcal{L}_{BSDE}(\theta_{0,q}, \dots, \theta_{N-1,q}) := \mathbb{E}\big[\big|p_N + \widetilde{U}'(Y_N)\big|^2\big]. \tag{4.70}$$

Moreover, the parameters for the dual control process have to be chosen such that (4.8) holds. Hence, we intend to minimize for every $i \in \{0, \dots, N-1\}$:

$$\mathcal{L}^i_{control}(\theta_{i,v}) := \mathbb{E}\big[\big|p_i\, \delta_K(v_i) + q_i^\intercal\, \sigma^{-1}(t_i)\, v_i\big|^2\big]. \tag{4.71}$$

Finally, as in the construction of the dual problem, we want to find y such that it minimizes the loss function

$$\mathcal{L}_{dual}(y) := \mathbb{E}\big[\widetilde{U}(Y_N)\big] + x_0 y. \tag{4.72}$$

Similarly to the deep controlled 2BSDE algorithm, the deep SMP algorithm initializes the parameters randomly and then repeats a training step procedure until the parameters seem to have converged. One such training step is described in Algorithm 2 below. For every training step, we draw b_{size} realizations of $(\Delta B_i)_{i\in\{0,\dots,N-1\}}$, i.e. we sample $\{\omega_j\}_{j\in\{1,\dots,b_{size}\}} \subseteq \Omega$ as we study random coefficients.

Note that the deep SMP algorithm does not have a parameter which describes the value of the control problem, as opposed to y_0 in the deep controlled 2BSDE algorithm. However, motivated by Theorem 4.17, we can formulate a lower and an upper bound, respectively:

$$\widetilde{V}_l := \mathbb{E}\big[U(p_N)\big] \quad \text{and} \quad \widetilde{V}_u := \mathbb{E}\big[\widetilde{U}(Y_N)\big] + x_0 y. \tag{4.73}$$

Corollary 4.18 suggests that the gap between those bounds becomes small, if the technical requirements are satisfied. In practice, we calculate $\widetilde{V}_l$ and $\widetilde{V}_u$ by drawing a sample of size $N_{MC} \gg b_{size}$ from Ω, simulating the processes according to the above scheme and calculating the corresponding sample means. As this Monte Carlo method is computationally expensive, the bounds should not be calculated after each training step.

4.5 Formulation of the Novel Deep Primal SMP Algorithm

As (4.37) corresponds to minimizing the generalized Hamiltonian $\mathcal{H}$ in the control argument (cf. Remark 4.15), it is natural to ask whether Theorem 4.12 can be used for formulating an algorithm which solves the primal problem directly. We aim at integrating (4.37) into the novel algorithm, which will be called deep primal SMP algorithm in the following, in a way similar to how (3.44) was utilized in Section 3.4.

Let us fix an equidistant time discretization $(t_i)_{i\in\{0,\dots,N\}}$ of $[0,T]$ with step size $\Delta t := T/N$, where $N \in \mathbb{N}$. We are going to simulate the system (4.36) from Theorem 4.12 by means of a discrete-time forward scheme which yields two processes $(p_i)_{i\in\{0,\dots,N\}}$ and $(X_i)_{i\in\{0,\dots,N\}}$. For every $i \in \{0, \dots, N-1\}$, we substitute $\pi(t_i)$ and $q_1(t_i)$ with

$$\pi_i := \mathcal{N}_{\theta_{i,\pi}}(X_i) \quad \text{and} \quad q_i := \mathcal{N}_{\theta_{i,q}}(X_i), \tag{4.74}$$

respectively, where the neural networks $\mathcal{N}_{\theta_{i,\pi}} : \mathbb{R} \to \mathbb{R}^m$ and $\mathcal{N}_{\theta_{i,q}} : \mathbb{R} \to \mathbb{R}^m$ are parameterized by appropriate vectors $\theta_{i,\pi}$ and $\theta_{i,q}$. Moreover, like in both previously introduced

Algorithm 2 One training step of the deep SMP algorithm

```
1:  Generate $b_{size}$ realizations of $(\Delta B_i)_{i\in\{0,\dots,N-1\}}$, i.e. $\{\omega_j\}_{j\in\{1,\dots,b_{size}\}} \subseteq \Omega$;
2:  // Substep 1: Minimizing $\mathcal{L}_{BSDE}$
3:  Initialize according to (4.68) for every $j \in \{1,\dots,b_{size}\}$;
4:  for $i = 0,1,\dots,N-1$ do
5:      for $j = 1,2,\dots,b_{size}$ do
6:          Calculate $v_i^j$ and $q_i^j$ by means of (4.67);
7:          Use (4.69) in order to obtain $Y_{i+1}^j$ and $p_{i+1}^j$;
8:      end for
9:  end for
10: $loss1 \leftarrow \frac{1}{b_{size}} \sum_{j=1}^{b_{size}} \left|p_N^j + \widetilde{U}'(Y_N^j)\right|^2$;
11: Update $\theta_{0,q},\dots,\theta_{N-1,q}$ with one step of an SGD algorithm w.r.t. $loss1$;
12: // Substep 2: Minimizing $\mathcal{L}^i_{control}$ for every $i \in \{0,\dots,N-1\}$
13: for $i = 0,1,\dots,N-1$ do
14:     for $j = 1,2,\dots,b_{size}$ do
15:         if i==0 then
16:             Initialize $Y_0^j$ and $p_0^j$ according to (4.68);
17:         else
18:             Use (4.69) in order to obtain $Y_i^j$ and $p_i^j$;
19:         end if
20:         Calculate $v_i^j$ and $q_i^j$ by means of (4.67);
21:     end for
22:     $loss2_i \leftarrow \frac{1}{b_{size}} \sum_{j=1}^{b_{size}} \left|p_i^j\, \delta_K(v_i^j) + \left(q_i^j\right)^\intercal \sigma^{-1}(\omega_j, t_i)\, v_i^j\right|^2$;
23:     Update $\theta_{i,v}$ with one step of an SGD algorithm w.r.t. $loss2_i$;
24: end for
25: // Substep 3: Minimizing $\mathcal{L}_{dual}$
26: Initialize $Y_0^j$ with $y$ for every $j \in \{1,\dots,b_{size}\}$;
27: for $i = 0,1,\dots,N-1$ do
28:     for $j = 1,2,\dots,b_{size}$ do
29:         Calculate $v_i^j$ by means of (4.67);
30:         Use (4.69) in order to obtain $Y_{i+1}^j$;
31:     end for
32: end for
33: $loss3 \leftarrow \frac{1}{b_{size}} \sum_{j=1}^{b_{size}} \widetilde{U}(Y_N^j) + x_0 y$;
34: Update $y$ with one step of an SGD algorithm w.r.t. $loss3$;
```

algorithms, hard constraints guarantee via the final layer of $\mathcal{N}_{\theta_{i,\pi}}$ that every π_i is K-valued (cf. Chapter 5 for details). For $i = 0$, we simplify the network architectures according to Remark 3.26. We start our scheme with

$$X_0 := x_0 \quad \text{and} \quad p_0 := p, \tag{4.75}$$

where p is a trainable variable. Let $(\Delta B_i)_{i\in\{0,\dots,N-1\}}$ be a family of i.i.d. m-dimensional, centered normally distributed random vectors with covariance matrix $\Delta t \cdot I_m$. Hence, we can inductively define for every $i \in \{0, \dots, N-1\}$ (cf. (4.36)):

$$\begin{aligned} X_{i+1} &:= X_i + X_i \big(r(t_i) + \pi_i^\intercal \sigma(t_i)\, \theta(t_i)\big)\, \Delta t + X_i\, \pi_i^\intercal \sigma(t_i)\, \Delta B_i, \\ p_{i+1} &:= p_i - \big[\big(r(t_i) + \pi_i^\intercal \sigma(t_i)\, \theta(t_i)\big)\, p_i + \pi_i^\intercal \sigma(t_i)\, q_i\big]\, \Delta t + q_i^\intercal \Delta B_i. \end{aligned} \tag{4.76}$$

As $(p_i)_{i\in\{0,\dots,N\}}$ is supposed to fulfill a terminal condition, we aim at finding parameters which minimize

$$\mathcal{L}_{BSDE}(p, \theta_{0,q}, \dots, \theta_{N-1,q}) := \mathbb{E}\big[\big|p_N + U'(X_N)\big|^2\big]. \tag{4.77}$$

Furthermore, also the parameters for the control process have to be optimized. This is achieved by minimizing

$$\mathcal{L}^i_{control}(\theta_{i,\pi}) := \mathbb{E}\big[\pi_i^\intercal \sigma(t_i)\, \big(p_i\, \theta(t_i) + q_i\big)\big], \tag{4.78}$$

for every $i \in \{0, \dots, N-1\}$. In contrast to (4.37), we use the expectation operator here as the algorithm only considers a finite number of realizations. Moreover, it works with the same realization almost surely at most once (cf. also (3.53)).

Like the deep SMP algorithm, the deep primal SMP algorithm starts with randomly initialized parameters. It then repeats the training step procedure (cf. Algorithm 3 below) until the updates become sufficiently small. For every training step, the algorithm studies b_{size} realizations of $(\Delta B_i)_{i\in\{0,\dots,N-1\}}$, i.e. it samples $\{\omega_j\}_{j\in\{1,\dots,b_{size}\}} \subseteq \Omega$.

As in the deep SMP algorithm, there is no parameter which naturally describes the value of the control problem. However, motivated by (4.59), we can formulate a lower and an upper bound in a similar manner:

$$V_l := \mathbb{E}\big[U(X_N)\big] \quad \text{and} \quad V_u := \mathbb{E}\big[\widetilde{U}(-p_N)\big] - x_0 p. \tag{4.79}$$

Again, Corollary 4.18 suggests that $V_u - V_l$ becomes small, if the technical requirements are satisfied. In Chapter 5, we calculate V_l and V_u by drawing a large sample of size $N_{MC} \gg b_{size}$ from Ω, simulating the processes according to the above scheme and calculating the corresponding sample means.

Algorithm 3 One training step of the deep primal SMP algorithm

```
 1: Generate $b_{size}$ realizations of $(\Delta B_i)_{i\in\{0,\dots,N-1\}}$, i.e. $\{\omega_j\}_{j\in\{1,\dots,b_{size}\}} \subseteq \Omega$;
 2: // Substep 1: Minimizing $\mathcal{L}_{BSDE}$
 3: Initialize according to (4.75) for every $j \in \{1,\dots,b_{size}\}$;
 4: for $i = 0, 1, \dots, N-1$ do
 5:     for $j = 1, 2, \dots, b_{size}$ do
 6:         Calculate $\pi_i^j$ and $q_i^j$ by means of (4.74);
 7:         Use (4.76) in order to obtain $X_{i+1}^j$ and $p_{i+1}^j$;
 8:     end for
 9: end for
10: $loss1 \leftarrow \frac{1}{b_{size}} \sum_{j=1}^{b_{size}} \left| p_N^j + U'(X_N^j) \right|^2$;
11: Update $p, \theta_{0,q}, \dots, \theta_{N-1,q}$ with one step of an SGD algorithm w.r.t. $loss1$;
12: // Substep 2: Minimizing $\mathcal{L}_{control}^i$ for every $i \in \{0,\dots,N-1\}$
13: for $i = 0, 1, \dots, N-1$ do
14:     for $j = 1, 2, \dots, b_{size}$ do
15:         if i==0 then
16:             Initialize $X_0^j$ and $p_0^j$ according to (4.75);
17:         else
18:             Use (4.76) in order to obtain $X_i^j$ and $p_i^j$;
19:         end if
20:         Calculate $\pi_i^j$ and $q_i^j$ by means of (4.74);
21:     end for
22:     $loss2_i \leftarrow \frac{1}{b_{size}} \sum_{j=1}^{b_{size}} \left(\pi_i^j\right)^\intercal \sigma(\omega_j, t_i) \left(p_i^j\, \theta(\omega_j, t_i) + q_i^j\right)$;
23:     Update $\theta_{i,\pi}$ with one step of an SGD algorithm w.r.t. $loss2_i$;
24: end for
```

5 Numerical Experiments

In this chapter, we aim at providing numerical examples which illustrate the performance of the three previously introduced algorithms for specific utility maximization problems. All experiments have been performed in Python by means of the popular machine learning library TensorFlow (cf. Appendix A for the Python codes used in Example 5.5). Moreover, Adam will serve as our SGD algorithm in the following.
Our examples will show that, even after a relatively small number of training steps, the algorithms provide reliable estimates of the control problem's value. This holds in particular for high-dimensional problems (cf. Examples 5.4, 5.5 and 5.12 below).

5.1 Markovian Utility Maximization Problems

At first, we study problems where the processes r, μ and σ are deterministic. Hence, the state processes are in particular of the form (3.1). Therefore, all of the studied algorithms can be applied without any restrictions.

Example 5.1. We consider Merton's classical portfolio allocation problem with $K = \mathbb{R}$, i.e. we study the unconstrained problem with $m = 1$. We choose $x_0 = 1$, $T = 1$, $r \equiv 0.1$, $\mu \equiv 0.15$ and $\sigma \equiv 0.2$. Our utility function is given by $U(x) = 2\sqrt{x}, x \in \mathbb{R}^+$, whose Legendre-Fenchel transform has already been calculated in Example 2.9. As previously mentioned in connection with Theorem 3.8, one can explicitly find the value function for this problem. In [19] the solution is found by means of a separation of variables ansatz containing U which leads to an ODE in t with known solution. Moreover, it is observed that the supremum in (3.18) is attained by the same constant for every $(t, x) \in [0, T) \times \mathbb{R}^+$. In our unconstrained case, this constant is given by

$$\pi^* := 2\,\frac{\mu - r}{\sigma^2} = 2.5. \tag{5.1}$$

Furthermore, it is shown by means of a verification result (cf. Theorem 3.8) that the candidate function

$$v(t, x) := \exp(\rho(1 - t))\, U(x), \quad (t, x) \in [0, 1] \times \mathbb{R}^+, \tag{5.2}$$

with $\rho := (\mu - r)^2/(2\sigma^2) + r/2 = 0.0813$, corresponds indeed to the value function with the constant control π^* being optimal for all initial pairs. Hence, the value of our control problem is given by $V = 2.1693$. For our numerical experiments, we choose $N = 30$ and $b_{size} = 64$. Moreover, the network architectures for $t_i \neq 0$ and the piecewise constant learning rate schedules are given in Table 5.1. We place a batch normalization layer in front of every dense layer and ReLU serves as our activation function (cf. also Appendix A, where a very similar network architecture is implemented for Example 5.5).

Algorithm	LR $\mathcal{L}_{(2)BSDE}$	LR $\mathcal{L}^i_{control}/\mathcal{L}_{dual}$	Network Architecture
2BSDE	$10^{-2} \overset{2000}{\rightarrow} 10^{-3}$	$10^{-3} \overset{2000}{\rightarrow} 10^{-4}$	hidden: 2, neurons/hidden: 11
2BSDE_dual	$10^{-2} \overset{2000}{\rightarrow} 10^{-3}$	$10^{-3} \overset{2000}{\rightarrow} 10^{-4}$	hidden: 2, neurons/hidden: 11
SMP	$10^{-2} \overset{2000}{\rightarrow} 10^{-3}$	$10^{-3} \overset{2000}{\rightarrow} 10^{-4}$	hidden: 2, neurons/hidden: 11
SMP_primal	$10^{-2} \overset{2000}{\rightarrow} 10^{-3}$	$10^{-3} \overset{2000}{\rightarrow} 10^{-4}$	hidden: 2, neurons/hidden: 11

Table 5.1: Learning rate schedules and network architectures for our numerical experiments using TensorFlow.

Note that we do not have to train the dual control process as it necessarily has to be projected onto zero. We ran our algorithms for 5000 steps each. Figure 5.1 shows the evolution of the value approximations during the training procedure. For both SMP-based algorithms we calculated the bounds every 200 steps by means of a batch of size $N_{MC} = 10^5$.

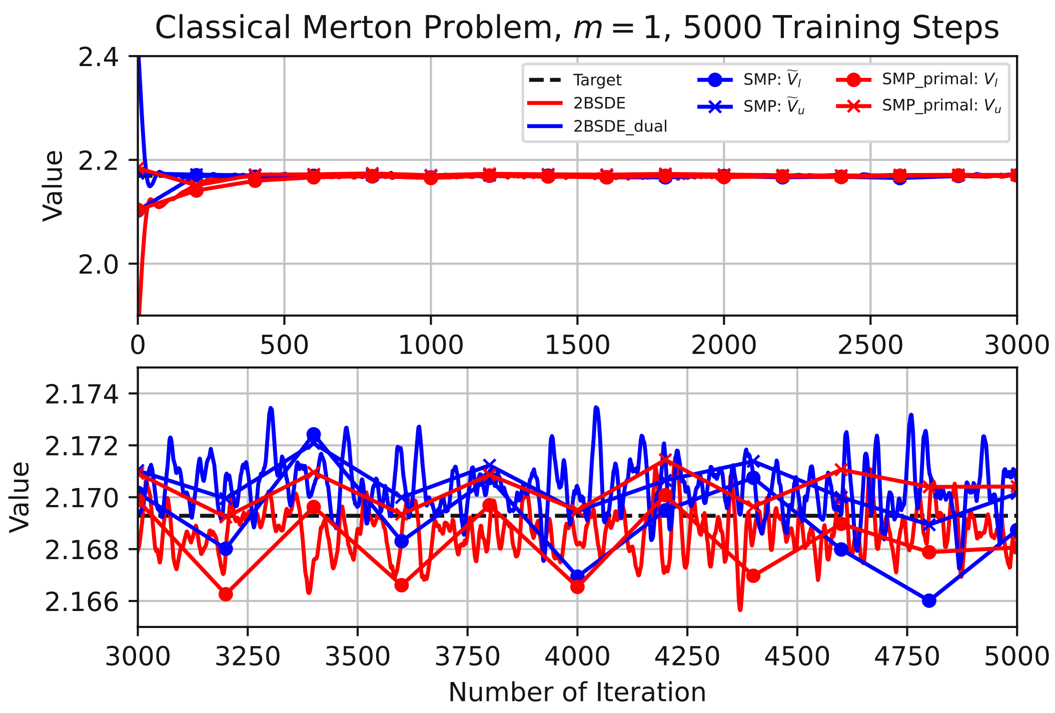

Figure 5.1: Value approximations for all four of our methods in the classical unconstrained Merton problem in the course of 5000 training steps.

The above figure shows in particular that the bounds from (4.73) and (4.79) do not necessarily have to hold for the entire training process. This is insofar not surprising as (4.59) and (4.64) only hold for optimal controls. Hence, we can expect this behavior merely from close to optimal parameters and larger values for N_{MC}.
Moreover, Figure 5.1 illustrates that we already arrive at good value approximations after

only a few hundred training steps in this low-dimensional example. The second subplot shows that the obtained values fluctuate in a neighborhood of the target value once they have reached it. However, this is not surprising as we have intentionally chosen learning rate schedules which feature only one reduction in order to demonstrate this phenomenon. In our subsequent examples we are, therefore, going to weaken this effect significantly by introducing a third (and sometimes even a fourth) learning rate level.
Finally, Figure 5.2 displays the neural networks $\mathcal{N}_{\theta_{i,\pi}}$ for several indices i from both algorithms which address the primal problem directly.

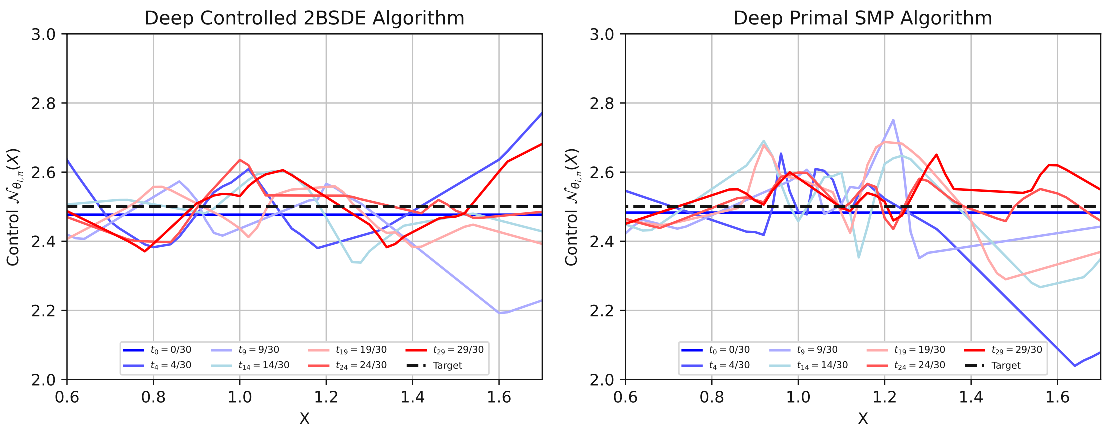

Figure 5.2: Neural networks modeling $\pi^*(t_i)$ for seven equidistant points in time at the end of the training procedure.

As the supremum norm of the errors is approximately bounded by 0.3 in both cases (except for t_4) and no initial guesses for the neural networks were used, we consider the approximation of π^* as good. Clearly, we have to assess the quality of the approximations for smaller points in time t_i in the light of X_i taking values close to $x_0 = 1$ with a high probability. Hence, the algorithms do not "see" a lot of training data from regions which are farther away from 1. An extreme example for this phenomenon would be $\mathcal{N}_{\theta_{0,\pi}}$, which always receives $x_0 = 1$ as input. However, we avoided this effect by modeling π_0 as a trainable variable, i.e. a trivial neural network that consists just of a bias vector.

Remark 5.2. During our numerical experiments, especially those which follow below, we observed that the batch normalization layers' parameter ε is an essential hyperparameter for the studied algorithms. As we shall see below, using the default value might lead to NaN-values, whereas choosing a larger value, e.g. $\varepsilon = 100$, resolves this issue and allows the algorithm to converge astonishingly fast (cf. Examples 5.3, 5.4 and 5.5). Moreover, small values for ε may also cause that the bounds of both SMP-based algorithms explode at the beginning of the training procedure (cf. Examples 5.6 and 5.7 for details). Hence, tuning ε for each problem is of paramount importance for the training success.

In the following, we slightly modify the problem studied in Example 5.1.

Example 5.3. Let us consider Merton's classical portfolio allocation problem again. We choose the same parameters as in Example 5.1. However, we require that admissible control processes have to take values in $[0,1]$, i.e. $K = [0,1]$. Hence, neither short selling nor borrowing money from the bank account is permitted. It can easily be seen that the support function is given by $\delta_K(z) = (-z)^+$, $z \in \mathbb{R}$. The exact solution to this problem can be obtained by using the same verification machinery as in Example 5.1. A separation of variables ansatz for the value function shows for our particular parameter choice that maximizing the Hamiltonian in (3.18) is equivalent to finding the global maximum of the quadratic function

$$[0,1] \ni \pi \mapsto (-0.01\pi^2 + 0.05\pi + 0.1)/2. \tag{5.3}$$

Clearly, it is attained by $\pi^* := 1$ since the stationary point given by (5.1) does not lie in K. As (5.3) maps π^* to 0.07, we obtain from an analogue verification procedure as in Example 5.1 that the constant control π^* is optimal for all $(t,x) \in [0,1] \times \mathbb{R}^+$ and the value function is given by

$$v(t,x) := \exp\big(0.07\,(1-t)\big)\,U(x), \quad (t,x) \in [0,1] \times \mathbb{R}^+. \tag{5.4}$$

Therefore, the value of the studied control problem is given by 2.14502. As in Example 5.1, we choose $N = 30$, $b_{size} = 64$ and $N_{MC} = 10^5$. Moreover, we select the same network architecture as above. In both algorithms which tackle the primal problem directly, the constraints are incorporated by applying the function $x \mapsto \exp(-x^2)$ to the outputs of the neural networks which model the control process. Clearly, one could also choose other functions which project the controls onto K. One decisive advantage of the above function is that the gradient never vanishes. In contrast to this, our experiments with the natural choice $x \mapsto \min\{x^+, 1\}$ led to a significantly larger approximation error since some π_i reached 0 and never recovered. As $\widetilde{K} = \mathbb{R}$ holds, the dual control does not require a projection function. Furthermore, we selected $s_K = (1 + |\cdot|)^{-1}$. We ran each algorithm for 10000 steps while using the following piecewise constant learning rate schedules:

Algorithm	LR $\mathcal{L}_{(2)BSDE}$	LR $\mathcal{L}^i_{control}/\mathcal{L}_{dual}$
2BSDE	$10^{-2} \overset{2000}{\rightarrow} 10^{-3} \overset{3000}{\rightarrow} 10^{-4} \overset{3000}{\rightarrow} 10^{-5}$	$10^{-3} \overset{2000}{\rightarrow} 10^{-4} \overset{3000}{\rightarrow} 10^{-5} \overset{3000}{\rightarrow} 10^{-6}$
2BSDE_dual	$10^{-2} \overset{1000}{\rightarrow} 10^{-3} \overset{2000}{\rightarrow} 10^{-4} \overset{3000}{\rightarrow} 10^{-5}$	$10^{-3} \overset{1000}{\rightarrow} 10^{-4} \overset{2000}{\rightarrow} 10^{-5} \overset{3000}{\rightarrow} 10^{-6}$
SMP	$10^{-2} \overset{2000}{\rightarrow} 10^{-3} \overset{3000}{\rightarrow} 10^{-4} \overset{3000}{\rightarrow} 10^{-5}$	$10^{-2} \overset{2000}{\rightarrow} 10^{-3} \overset{3000}{\rightarrow} 10^{-4} \overset{3000}{\rightarrow} 10^{-5}$
SMP_primal	$10^{-2} \overset{2000}{\rightarrow} 10^{-3} \overset{3000}{\rightarrow} 10^{-4} \overset{3000}{\rightarrow} 10^{-5}$	$10^{-3} \overset{2000}{\rightarrow} 10^{-4} \overset{3000}{\rightarrow} 10^{-5} \overset{3000}{\rightarrow} 10^{-6}$

Table 5.2: Piecewise constant learning rate schedules for our numerical experiments using TensorFlow.

Figure 5.3 depicts the value approximations obtained during our training procedure. We observe that all value approximations, with the exception of $\widetilde{V}_u$, are even after only a few hundred training steps highly accurate. The bounds of both SMP-based algorithms show a greater variability due to potential errors resulting from the Monte Carlo method. However, the results from the deep SMP algorithm have to be treated with caution as we modified the algorithm by projecting v onto 0 in order to obtain faster convergence. In its original form (cf. Section 4.4), it required 40000 training steps for $\widetilde{V}_l$ to reach the same accuracy as

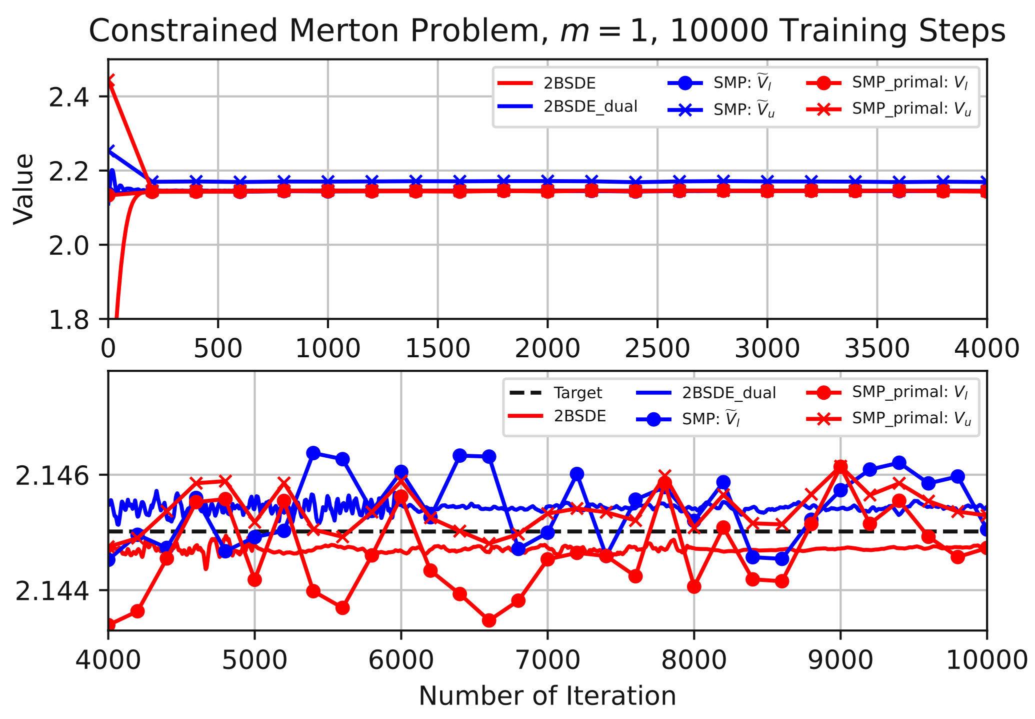

Figure 5.3: Value approximations for the studied constrained Merton problem in the course of 10000 training steps.

above, whereas $\widetilde{V}_u$ remained close to the unconstrained value as in Figure 5.3 above. The reason for this behavior is that the obtained dual control was very close to the zero process. This illustrates a weakness of the deep SMP algorithm which has also been observed in [5]: Since q_i is modeled without taking the potential dependency on v_i (cf. Theorem 3.23 for Markovian problems) into account (cf. (4.67)), the zero control always minimizes the loss function (4.71). Hence, the algorithm tends to mistake the zero control for the optimal dual control, which leads to an upper bound as in the unconstrained case. This might be the reason, why in [5] the deep SMP algorithm was applied exclusively to unconstrained problems or problems with positive market price of risk and $K = (\mathbb{R}_0^+)^m$. The lower bound is not exposed to this issue as it does not depend explicitly on the dual control (cf. (4.69) and (4.73)). Here the constraints are incorporated by the function s_K.
Moreover, we initially observed NaN-values while applying the deep controlled 2BSDE algorithm for the dual problem. Setting $\varepsilon = 100$ for all batch normalization layers resolved this issue. At the end of training, we obtain

$$V = 2.14468, \quad \widetilde{V} = 2.14544, \quad \widetilde{V}_l = 2.14505, \quad V_l = 2.14473 \quad \text{and} \quad V_u = 2.14529. \tag{5.5}$$

$\widetilde{V}_u$ is given by 2.17062. Finally, Table 5.3 shows the approximation quality of the neural networks modeling the control process for both algorithms which solve the primal problem directly. We use the supremum norm for measuring the quality since π^* is a Markovian control as well. For reasons of clarity, we present the results by means of a table, as the approximations are significantly better than in Example 5.1 (cf. Figure 5.2). The reason for this performance disparity is that the one-dimensional set K is bounded here with the optimal constant control mapping to a point in ∂K.

Algorithm	$i = 4$	$i = 9$	$i = 14$	$i = 19$	$i = 24$	$i = 29$
2BSDE	0.165721	0.000142	0.000036	0.001166	0.000005	0.000055
SMP_primal	0.114749	0.021831	0.001304	0.000062	0.000029	0.000649

Table 5.3: Maximal deviations of the neural networks $\mathcal{N}_{\theta_{i,\pi}}$ from the optimal control on $[0.6, 1.7]$, i.e. $\|1 - \mathcal{N}_{\theta_{i,\pi}}\|_\infty$, for several indices i.

We observe that our neural networks $\mathcal{N}_{\theta_{i,\pi}}$ are, especially for large i, uniformly close to $\pi^*(t_i) \equiv 1$. Again, the values for $i = 4$ have to be assessed considering the fact that the probability distribution of X_4 has most of its mass concentrated close to 1. In a smaller neighborhood of 1, we observe the same approximation quality as for larger indices i. Due to our construction of π_0 (cf. Remark 3.26) we do not encounter this issue here. Both algorithms determine $\pi_0 \equiv 1$ which corresponds exactly to the theoretical benchmark.

In the following example, we consider a high-dimensional Markovian problem with logarithmic utility function.

Example 5.4. We consider an unconstrained problem with $m = 30$, i.e. $K = \mathbb{R}^{30}$. The natural logarithm serves as our utility function. We recall from Example 2.9 that $\widetilde{U} = -\log -1$ holds. Moreover, we choose $x_0 = 10$ and $T = 0.5$. For every $t \in [0, 0.5]$ and $i \in \{1, \dots, 30\}$, $r(t)$ and $\mu_i(t)$ are given by $0.06 \exp(t/2)$ and $0.07 + 0.02 \sin(4\pi t + \pi i/15)$, respectively. Furthermore, the diagonal elements $\sigma_{i,i}(t)$ are given by $0.3\big(1 + \sqrt{t}\big)$ whereas the remaining entries are chosen to be 0.1. We consider two possible approaches for finding a theoretical benchmark for the value of our problem. On the one hand, we can directly use the well-known (cf. [16], for example) optimal control

$$\pi^*(t) = \big(\sigma(t)\,\sigma^\intercal(t)\big)^{-1}\big[\mu(t) - \big(r(t), \dots, r(t)\big)^\intercal\big], \quad t \in [0, 0.5], \tag{5.6}$$

and calculate the expectation by means of the Monte Carlo method. The same approach works for the dual problem as well, since $v^* \equiv 0$ holds and $y^* = x_0^{-1} = 0.1$ can be concluded from $\widetilde{U} = -\log -1$ and the dual state process being a stochastic exponential. We simulated the state processes by means of a discrete-time forward scheme as above with $N = 50$ and $N_{MC} = 10^6$. We obtained $V = 2.35069$ and $\widetilde{V} = 2.35067$, respectively. Clearly, this method requires quite some time and still produces an error caused by the discretization and the Monte Carlo method. Fortunately, as the logarithm satisfies some desirable properties, we can even find the exact dual value quite easily (cf. [16]). As a preparation for Example 5.5, we consider a general closed, convex set K which contains 0. As $\widetilde{U} = -\log -1$ holds, we obtain $y^* = x_0^{-1} = 0.1$ as well in this general case. Hence, we can conclude (cf. (3.5) and the integrability assumptions in the definition of $\mathcal{D}$):

$$\widetilde{V} = \log(x_0) + \inf_{v \in \mathcal{D}_2} \mathbb{E}\bigg[\int_0^{0.5} \Big(r(t) + \delta_K(v(t)) + \frac{1}{2}\big|\theta(t) + \sigma^{-1}(t)v(t)\big|^2\Big)\,dt\bigg]. \tag{5.7}$$

The infimum is attained by the deterministic process v^* which is, for every $t \in [0, 0.5]$, defined via

$$v^*(t) := \arg\min_{v \in \widetilde{K}} \left\{ \delta_K(v) + \frac{1}{2} \left| \theta(t) + \sigma^{-1}(t) v \right|^2 \right\}. \tag{5.8}$$

Obviously, we obtain $v^* \equiv 0$ for the unconstrained problem. Combining this with (5.7) and calculating the occurring integral by means of an equidistant time discretization consisting of 1000 points yields 2.35058. We use this as our benchmark in the following as this method eliminates the error caused by the Monte Carlo method. Moreover, we can use a finer time grid for this method since the computation time is significantly reduced by not having to draw large samples as in the first approach.
For our numerical experiments, we choose the same network architectures as above. Moreover, we select $N = 10$, $b_{size} = 64$, $N_{MC} = 10^5$ and the following piecewise constant learning rate schedules are used:

Algorithm	LR $\mathcal{L}_{(2)BSDE}$	LR $\mathcal{L}^i_{control}$/ LR $\mathcal{L}_{dual}$
2BSDE	$10^{-2} \overset{2000}{\to} 10^{-3} \overset{3000}{\to} 10^{-4} \overset{3000}{\to} 10^{-5}$	$10^{-3} \overset{2000}{\to} 10^{-4} \overset{3000}{\to} 10^{-5} \overset{3000}{\to} 10^{-6}$
2BSDE_dual	$10^{-2} \overset{1000}{\to} 10^{-3} \overset{2000}{\to} 10^{-4} \overset{3000}{\to} 10^{-5}$	$10^{-2} \overset{200}{\to} 10^{-4} \overset{800}{\to} 10^{-5} \overset{7000}{\to} 10^{-6}$
SMP	$10^{-2} \overset{2000}{\to} 10^{-3} \overset{3000}{\to} 10^{-4} \overset{3000}{\to} 10^{-5}$	$10^{-2} \overset{200}{\to} 10^{-4} \overset{800}{\to} 10^{-5} \overset{7000}{\to} 10^{-6}$
SMP_primal	$10^{-2} \overset{1000}{\to} 10^{-3} \overset{2000}{\to} 10^{-4} \overset{5000}{\to} 10^{-5}$	$10^{-3} \overset{1000}{\to} 10^{-4} \overset{2000}{\to} 10^{-5} \overset{5000}{\to} 10^{-6}$

Table 5.4: Piecewise constant learning rate schedules for our numerical experiments using TensorFlow.

We quickly reduce the learning rate for the parameter y, since it converges at an extremely high speed to y^* and its value significantly influences the implied value approximation. Potential issues with NaN-values were again resolved by choosing ε sufficiently large for all batch normalization layers. We ran the algorithms for 10000 training steps. The results are depicted in Figure 5.4 below. Even though we study a high-dimensional example, we observe that we already obtain good estimates during the first 2000 training steps. At the end of training, we get

$$\begin{aligned} &V = 2.34866, \quad \widetilde{V} = 2.35226, \quad \widetilde{V}_l = 2.35162, \quad \widetilde{V}_u = 2.35196, \\ &V_l = 2.35003 \quad \text{and} \quad V_u = 2.35067. \end{aligned} \tag{5.9}$$

It is remarkable that the deep primal SMP algorithm yields the most accurate results despite the fact that the optimal dual control is a priori known for unconstrained problems, which significantly reduces the dimensionality of the dual problem.

In the following, we reconsider the above problem. However, we introduce constraints which further increases the complexity of the problem.

Example 5.5. Let us consider the setting of Example 5.4 again. We define $\kappa := 1/m = 1/30$ and choose $K = [-\kappa, \infty)^{30}$. Hence, short selling is permitted, but only to a certain extent for each stock. This choice of K implies in particular that the overall short selling

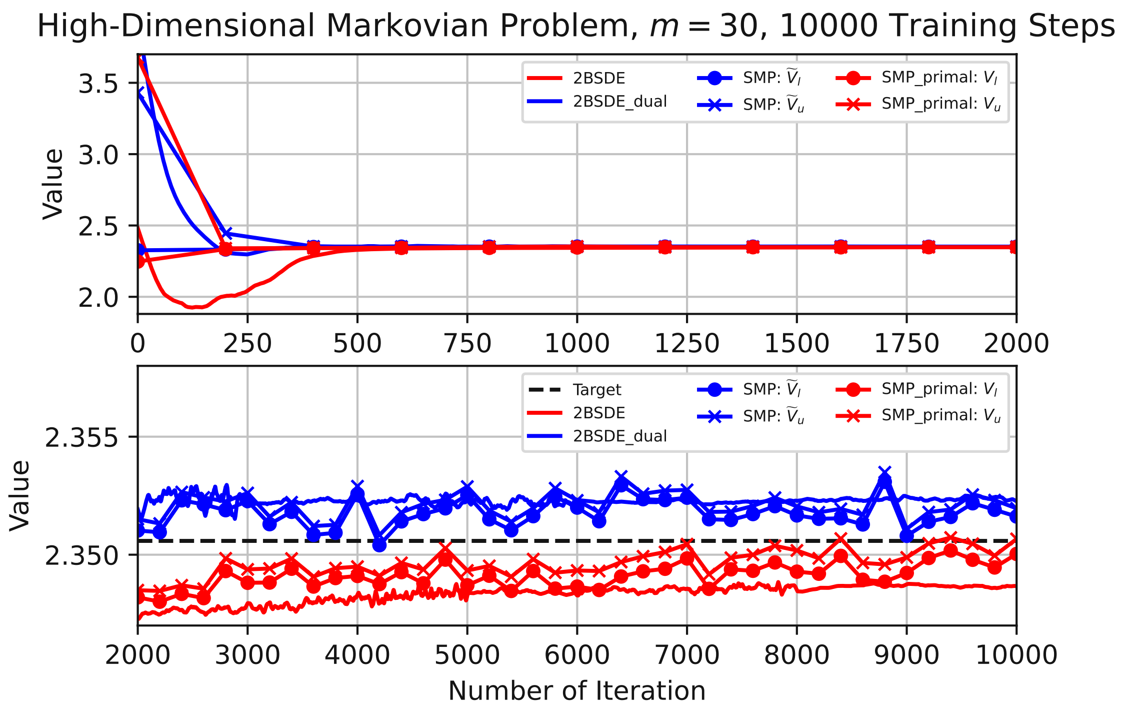

Figure 5.4: Value approximations for the studied unconstrained Markovian problem with logarithmic utility function in the course of 10000 training steps.

volume of stocks must not surpass the portfolio value of an investor. One can easily see that $\widetilde{K} = (\mathbb{R}_0^+)^{30}$ and $\delta_K(v) = \kappa \sum_{i=1}^{30} v_i$, $v \in \widetilde{K}$, hold. We obtain a theoretical benchmark for this problem by means of the approach described in Example 5.4 above. We approximate the integral in (5.7) using an equidistant time discretization consisting of 1000 points. For each of these points, we solved the deterministic convex optimization problem (5.8) by means of the CVXPY package in Python. After only a few seconds we obtained 2.34335 as our benchmark, which illustrates again that this method, if applicable to a specific problem, should be preferred over Monte Carlo methods.
We choose the same network architectures and values for N, b_{size} and N_{MC} as in Example 5.4 above. For both algorithms which tackle the primal problem directly we implemented hard constraints by applying $x \mapsto (x^2 - \kappa)$ componentwise to the outputs of $\mathcal{N}_{\theta_{i,\pi}}$ for every $i \in \{1, \ldots, N-1\}$. Our numerical experiments have shown that it is favorable to use $x \mapsto \max\{-\kappa, x\}$ for π_0 instead. For the dual control process, we project onto $\widetilde{K}$ in a similar manner by means of the function $x \mapsto x^2$. Moreover, s_K in the deep SMP algorithm is defined as the function which applies $x \mapsto \max\{-\kappa, x\}$ to every component. Our numerical experiments have shown that also $x \mapsto (x^2 - \kappa)$ or $x \mapsto (|x| - \kappa)$ yield similar results. In the following, we apply the same learning rate schedules as given in Table 5.4. However, we delay the final learning rate level for the dual version of the deep controlled 2BSDE algorithm by 2000 training steps as the algorithm yielded the largest approximation errors

in the middle of the training procedure. For the deep SMP algorithm, we choose the same learning rate schedule for the dual control as for the BSDE part, whereas, for the dual version of the deep controlled 2BSDE algorithm, we use the schedule for the 2BSDE part with learning rates divided by ten. The Python codes for this specific problem are provided in Appendix A below.

We ran each algorithm for 10000 training steps. The training progress is depicted in Figure 5.5 below. We observe fast convergence towards the theoretical benchmark. Moreover, we witness again (cf. Example 5.3) that the deep SMP algorithm tends to mistake $v \equiv 0$ for the optimal dual control as $\widetilde{V}_u$ agrees approximately with the value of the corresponding unconstrained problem (cf. (5.10) below). However, as argued in Example 5.3, we can still use $\widetilde{V}_l$ as a reliable estimate. In this example, it is even more accurate than the value determined by the dual version of the deep controlled 2BSDE algorithm, as Figure 5.5 and (5.10) show.

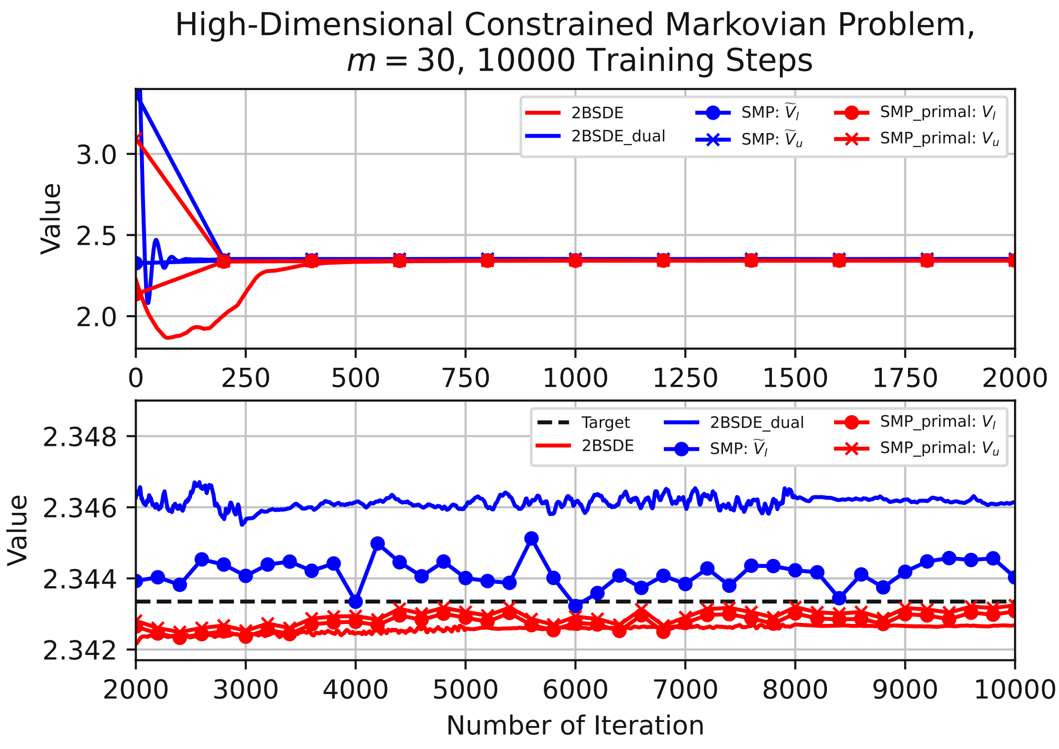

Figure 5.5: Value approximations for the studied constrained Markovian problem with logarithmic utility function in the course of 10000 training steps.

At the end of the training procedure, we obtain

$$\begin{aligned} &V = 2.34268, \quad \widetilde{V} = 2.34614, \quad \widetilde{V}_l = 2.34403, \quad \widetilde{V}_u = 2.35192, \\ &V_l = 2.34308 \quad \text{and} \quad V_u = 2.34324. \end{aligned} \tag{5.10}$$

Except for $\widetilde{V}$, $\widetilde{V}_u$ and V_u we achieve even smaller relative errors than in the unconstrained case. Note that the value of $\widetilde{V}_u$ is essentially the same as in (5.9). In conclusion, we gather from the above observations that the studied algorithms also provide highly accurate value approximations for high-dimensional Markovian problems with non-trivial constraints. Moreover, both bounds implied by the deep primal SMP algorithm remain highly accurate for constrained problems, whereas the upper bound determined by the deep SMP algorithm tends to attain the value of the associated unconstrained problem. As in Example 5.4, the deep primal SMP algorithm outperforms even both versions of the deep controlled 2BSDE algorithm quite significantly (cf. (5.9) and (5.10)).

5.2 Non-Markovian Utility Maximization Problems: Path Dependent Coefficients

In this section, we study utility maximization problems with path dependent coefficients. Hence, we cannot apply the deep controlled 2BSDE algorithm. However, we can still use both SMP-related algorithms and compare their outcomes. At first, we consider a problem with path dependent μ.

Example 5.6. We consider an unconstrained problem with five stocks, i.e. $K = \mathbb{R}^5$. Moreover, we select $U = 2\sqrt{\cdot}$, $x_0 = 1$, $T = 0.5$, $r \equiv 0.1$ and σ as a constant matrix with diagonal elements equal to 0.2 and the remaining entries are set to 0.05. For every $i \in \{1, \dots, 5\}$ and $t \in (0, 0.5]$, we define

$$\mu_i(t) := \begin{cases} \mu_{high} & \text{for } S_i(t) \geq \frac{1}{t}\int_0^t S_i(s)\,ds, \\ \mu_{low} & \text{else.} \end{cases} \tag{5.11}$$

Note that the expression on the right-hand side is well-defined due to the continuity of S_i. Its limit for $t \searrow 0$ exists almost surely and is given by $S_i(0)$, which follows again from the fact that S_i has continuous paths. Hence, we can define $\mu_i(0) := \mu_{high}$. For our numerical experiments, we choose $\mu_{low} = 0.08$ and $\mu_{high} = 0.12$. The motivation of (5.11) is that many private investors nowadays simply make trade decisions based on trends. For example, if a chart looks “good”, i.e. the stock is rising, but not straight out of a historic low, then the stock is considered a buy. Hence, the demand rises which strengthens the upward trend. We assume that the opposite happens, if the stock price is lower than its historic mean. Moreover, we consider the historic mean over $[0, t]$ instead of a moving average for a fixed time length as $T = 0.5$ is rather small.
For our numerical implementation, we take $N = 10$, $b_{size} = 64$ and $N_{MC} = 10^5$. Again, the bounds are calculated every 200 steps. Except for the dimension of the output layers, we choose the same network architectures as in Example 5.1 (cf. Table 5.1). Moreover, we use the following learning rate schedules:

Algorithm	LR $\mathcal{L}_{BSDE}$	LR $\mathcal{L}^i_{control}$	LR $\mathcal{L}_{dual}$
SMP	$10^{-2} \overset{300}{\to} 10^{-3} \overset{1700}{\to} 10^{-4}$	-	$10^{-2} \overset{300}{\to} 10^{-3} \overset{1700}{\to} 10^{-4}$
SMP_primal	$10^{-2} \overset{300}{\to} 10^{-3} \overset{1700}{\to} 10^{-4}$	$10^{-3} \overset{300}{\to} 10^{-4} \overset{1700}{\to} 10^{-5}$	-

Table 5.5: Piecewise constant learning rate schedules for our numerical experiments using TensorFlow.

Figure 5.6 shows the results of running both algorithms for 20000 steps, where we chose $\varepsilon = 100$. This was necessary in order to prevent the lower bound provided by the deep primal SMP algorithm from becoming very large during the first half of the training procedure (cf. Remark 5.2).

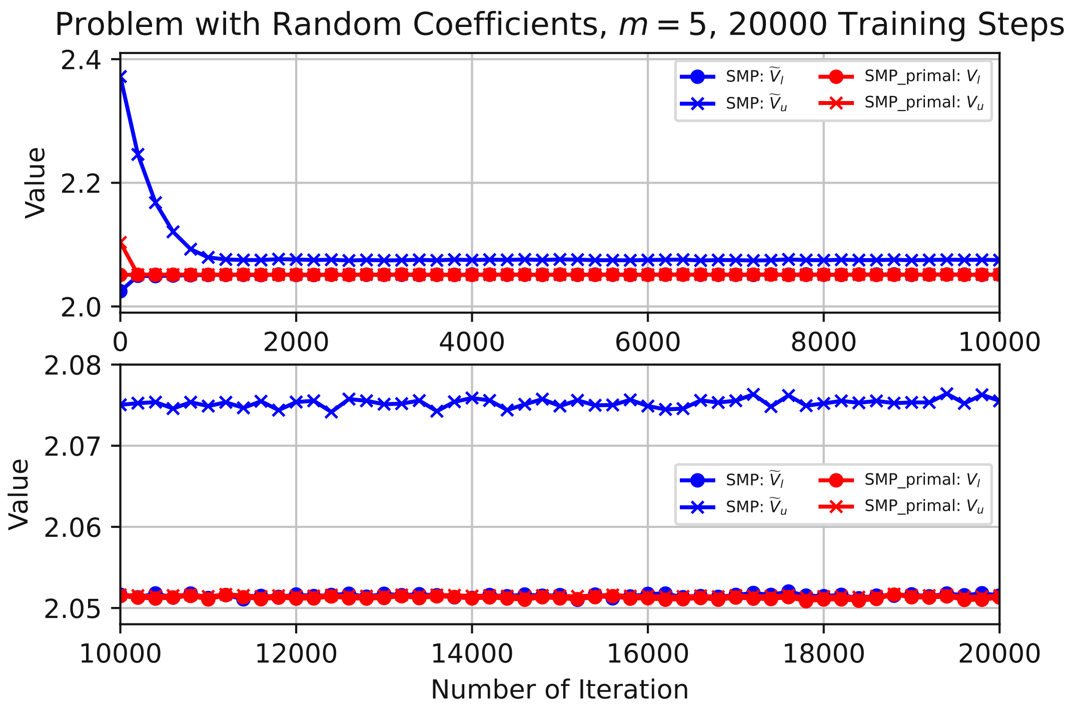

Figure 5.6: Value approximations for both SMP-based algorithms in the studied control problem with path dependent coefficient μ in the course of 20000 training steps.

Since the non-Markovian structure of the problem complicates finding an explicit solution significantly, we content ourselves with comparing the bounds implied by the algorithms. As in the above examples, we observe that most bounds only need a relatively low amount of training steps for arriving at reasonable estimates. This is in particular the case for both bounds implied by the deep primal SMP algorithm. After just 400 training steps, we obtain $V_l = 2.0513$ and $V_u = 2.0516$, which almost agrees with the final result (cf. (5.12) below). At the end of the training procedure, our four estimates are given by

$$\widetilde{V}_l = 2.0515, \quad \widetilde{V}_u = 2.0755, \quad V_l = 2.0513 \quad \text{and} \quad V_u = 2.0515. \tag{5.12}$$

It is remarkable that there is a steady duality gap between $\widetilde{V}_l$ and $\widetilde{V}_u$. We investigated this result further by dividing the learning rates by 10 and running the deep SMP algorithm for 30000 additional steps. We obtained $\widetilde{V}_l = 2.0514$ and $\widetilde{V}_u = 2.0749$, i.e. approximately the same values as in (5.12), which supports the above observation. Moreover, we repeated the above experiment for $N = 50$ and 20000 training steps, which resulted in $\widetilde{V}_l = 2.0505$ and $\widetilde{V}_u = 2.0758$, respectively, i.e. leading to the same result. Furthermore, working with

$\mu_{high} = 0.14$ instead increases the duality gap determined by the deep SMP algorithm to 0.0532, where the obtained bounds are given by $\widetilde{V}_l = 2.0566$ and $\widetilde{V}_u = 2.1098$. In contrast to this, the deep primal SMP algorithm yields $V_l = 2.0565$ and $V_u = 2.0568$, which agrees again with $\widetilde{V}_l$. Moreover, if we set $\mu_{low} = \mu_{high}$, the duality gap vanishes. Hence, the duality gap produced by the deep SMP algorithm might have a non-trivial connection with $\mu_{high} - \mu_{low}$. However, as the duality gap is negligible in the deep primal SMP algorithm, this behavior should not result from a potential violation of the strong duality property. In conclusion, we gather from (5.12) and the ensuing considerations that the value of our control problem should be approximately given by 2.0515 as all bounds except $\widetilde{V}_u$ suggest.

In the following, we apply the deep primal SMP algorithm to a problem with path dependent volatility which is also studied in [5], where the problem is tackled by means of the deep SMP algorithm. Hence, we will be able to compare the results.

Example 5.7. We consider a two-dimensional problem where selling stocks short is prohibited, i.e. $K = (\mathbb{R}_0^+)^2$. Moreover, we choose $U = 2\sqrt{\cdot}$, $x_0 = 1$, $r \equiv 0.05$, $\mu \equiv (0.06, 0.06)^\intercal$ and, for every $t \in [0, T]$, $\sigma(t)$ is given by a diagonal matrix, where the diagonal entry $\sigma_{i,i}(t)$ is for each $i \in \{1, 2\}$ determined by

$$\sigma_{i,i}(t) := \begin{cases} \sigma_{low} & \text{for } S_i(t) < \sup_{s\in[0,t]} S_i(s), \\ \sigma_{high} & \text{else.} \end{cases} \tag{5.13}$$

As in [5], we choose $\sigma_{low} = 0.3$ and $\sigma_{high} = 0.2$. Clearly, the path dependency in (5.13) makes the problem non-Markovian, which is why we cannot apply the deep controlled 2BSDE algorithm. Furthermore, it complicates finding a theoretical benchmark for the value of the problem. Therefore, the authors of [5] chose to simply compare the bounds implied by the deep SMP algorithm. Since we now have a second method for tackling such problems, namely the novel deep primal SMP algorithm, we can use this algorithm in order to verify the results of [5] for several pairs (T, N). For our numerical experiments we choose $b_{size} = 64$ and $N_{MC} = 10^5$. We select the same network architectures as in Example 5.1, except for the output dimensions. By applying ReLU right after each final dense layer of our control process networks $\mathcal{N}_{\theta_{i,\pi}}$, we ensure that the control process maps to K. Moreover, we work with the following learning rate schedules:

Algorithm	LR $\mathcal{L}_{BSDE}$	LR $\mathcal{L}^i_{control}$
SMP_primal	$10^{-2} \overset{300}{\to} 10^{-3} \overset{1700}{\to} 10^{-4}$	$10^{-3} \overset{300}{\to} 10^{-4} \overset{1700}{\to} 10^{-5}$

Table 5.6: Piecewise constant learning rate schedules for our numerical experiments using TensorFlow.

In accordance with Remark 5.2, we tried different choices for ε. We observed that the default value for ε worked, except for the experiments with $N = 50$. Here we witnessed exploding bounds in the beginning and a relatively low convergence speed. However, setting ε to 1 resolved both issues, which is why we use $\varepsilon = 1$ for all pairs (T, N) with $N = 50$ in the following.

We ran the deep primal SMP algorithm for 10000 training steps and for several pairs (T, N). Table 5.7 provides the results of our numerical experiments and compares them with the results obtained in [5].

(T, N)	$\widetilde{V}_l$	$\widetilde{V}_u$	$\widetilde{V}_u - \widetilde{V}_l$	V_l	V_u	$V_u - V_l$
$(0.2, 5)$	2.01057	2.01066	0.00009	2.01058	2.01067	0.00009
$(0.2, 10)$	2.01054	2.01063	0.00009	2.01061	2.01063	0.00002
$(0.2, 20)$	2.01057	2.01063	0.00006	2.01051	2.01059	0.00008
$(0.2, 50)$	2.01061	2.01062	0.00001	2.01051	2.01058	0.00007
$(0.5, 5)$	2.02652	2.02682	0.00030	2.02624	2.02664	0.00040
$(0.5, 10)$	2.02654	2.02676	0.00022	2.02633	2.02651	0.00018
$(0.5, 20)$	2.02648	2.02665	0.00017	2.02602	2.02615	0.00013
$(0.5, 50)$	2.02637	2.02650	0.00013	2.02628	2.02645	0.00017
$(1, 5)$	2.05333	2.05421	0.00088	2.05349	2.05391	0.00042
$(1, 10)$	2.05337	2.05397	0.00060	2.05322	2.05372	0.00050
$(1, 20)$	2.05327	2.05368	0.00041	2.05286	2.05315	0.00029
$(1, 50)$	2.05307	2.05339	0.00032	2.05295	2.05332	0.00037

Table 5.7: Results of our numerical experiments using the deep primal SMP algorithm in comparison with the results of [5], where the deep SMP algorithm was used.

We observe that we obtain slightly better results for small N, i.e. $N = 5$ and $N = 10$. However, this conclusion has to be handled with great care as the gaps are usually determined by the fourth and fifth decimal place, which are certainly exposed to a potential approximation error resulting from the Monte Carlo method. During all of the above experiments we experienced that the gap size already falls below 0.001 during the first 2000 training steps (cf. in particular Figure 5.7 below). For example, we obtained $V_l = 2.05278$ and $V_u = 2.05339$, respectively, after the initial 2000 steps for $(T, N) = (1, 20)$. Moreover, we conclude from Table 5.7 that increasing the parameter N does not necessarily have to improve the results. This is illustrated by the pairs with $N = 50$. Furthermore, a training step for the example with $(0.2, 10)$ lasts 0.023 seconds, whereas it takes 0.056 seconds for the one with $(0.2, 50)$. Those two observations are the reason why we focus on medium scale values for N in the remainder of Chapter 5.
The aforementioned immense convergence speed of the deep primal SMP algorithm turns out to be a decisive advantage over the deep SMP algorithm for this particular problem. For this purpose, we applied the latter algorithm to the problem corresponding to the pair $(0.5, 10)$. Note that $\delta_K \equiv 0$ holds on $\widetilde{K} = (\mathbb{R}_0^+)^2$ according to Example 2.12. Since the deep SMP algorithm had not converged during the first 10000 steps, we increased the number of training steps for both algorithms to 20000. The evolution of the bounds is depicted in Figure 5.7 below. As in Example 5.3, projecting v onto 0 might be a worthwhile approach as repeating the above experiment while implementing this adjustment shows: We obtained $\widetilde{V}_l = 2.0262$ and $\widetilde{V}_u = 2.0267$ with a similar accuracy already achieved after 2500 training steps. It is not surprising that also $\widetilde{V}_u$ is accurate here, since the market price of risk is strictly positive and we only have short selling constraints.

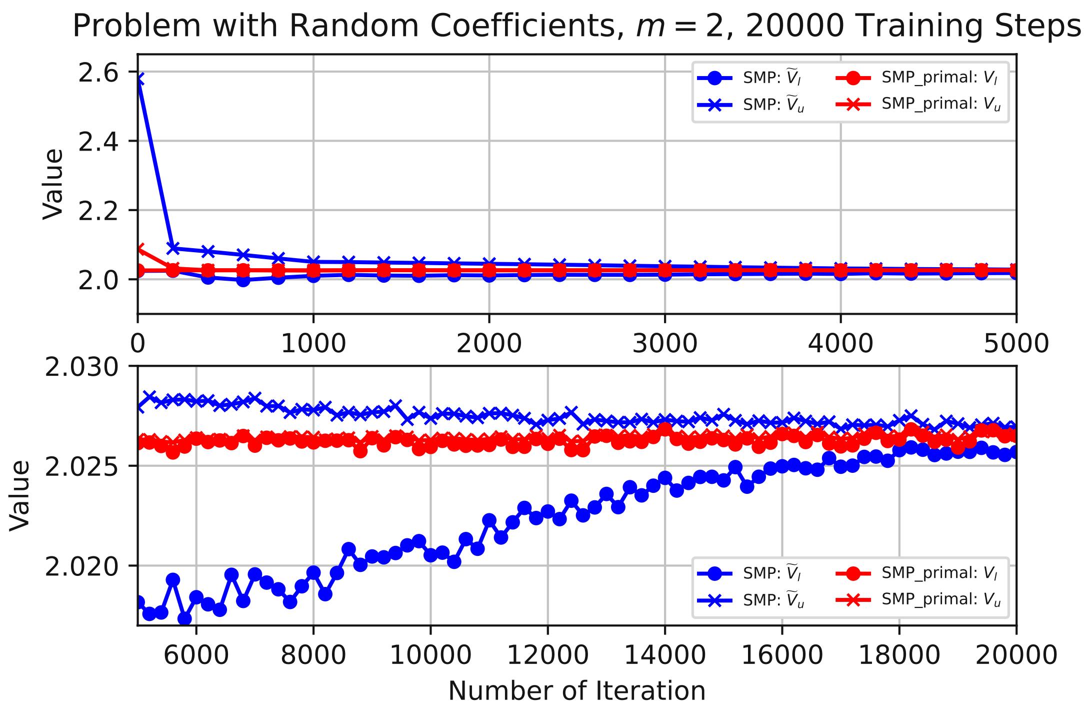

Figure 5.7: Value approximations for both SMP-based algorithms in our second control problem with random coefficients for $T = 0.5$ and $N = 10$ in the course of 20000 training steps.

We observe that the deep primal SMP algorithm arrives almost immediately at the solution, whereas its dual counterpart needs significantly more iterations. However, both algorithms yield relatively small duality gaps. At the end of training the four bounds are given by

$$\widetilde{V}_l = 2.02569, \quad \widetilde{V}_u = 2.02695, \quad V_l = 2.02649 \quad \text{and} \quad V_u = 2.02652. \tag{5.14}$$

It is remarkable that the deep primal SMP algorithm already yields $V_l = 2.02603$ and $V_u = 2.02658$ after only 800 training steps.
During a repetition of the above experiment we noticed that the lower bound of the deep SMP algorithm can become very large at the beginning of the training process (cf. Example 5.6 for a similar phenomenon caused by the deep primal SMP algorithm). However, we did not increase ε, as suggested by Remark 5.2, for comparability reasons since we used its default value for the deep primal SMP algorithm. We conclude that the deep primal SMP algorithm should be used as the primary algorithm for constrained problems with path dependent volatility as in (5.13). The results can then be verified by means of the deep SMP algorithm. A reason for this performance disparity might be that, unlike in the unconstrained case, the dual control optimization procedure is non-trivial. Hence, a training step consists of three substeps which might initially interfere in a negative way as they aim at minimizing different loss functions.

5.3 Non-Markovian Utility Maximization Problems: Coefficients Satisfying Their Own SDEs

In this section, we consider problems with coefficients which satisfy their own SDEs. As in Section 5.2, this implies that the wealth process is not of the form (3.1). However, as we shall see below, we can still apply the deep controlled 2BSDE algorithm. The problem can be converted into a Markovian problem by considering higher-dimensional state processes, where the additional components are precisely the random coefficients.

At first, we consider Heston's stochastic volatility model with one traded, risky asset. Here the dynamics of the stock price is given by

$$dS(t) = S(t)\,(r + A\nu(t))\,dt + S(t)\sqrt{\nu(t)}\,dB^1(t), \quad t \in [0,T], \tag{5.15}$$

where the parameters $r, A \in \mathbb{R}^+$ can be interpreted as the risk-free interest rate, which is assumed to be constant, and the market price of risk, respectively. Note that we have $\theta = A\sqrt{\nu}$ and $\sigma = \sqrt{\nu}$ following the notation from Chapter 2. The process ν satisfies

$$d\nu(t) = \kappa(\theta_\nu - \nu(t))\,dt + \xi\sqrt{\nu(t)}\,dB^\nu(t), \quad t \in [0,T], \tag{5.16}$$

with initial condition $\nu(0) = \nu_0 \in \mathbb{R}^+$. The Brownian motions B^ν and B^1 have correlation parameter $\rho \in [-1,1]$. Hence, we can write B^ν as $\rho B^1 + \sqrt{1-\rho^2}\,B^2$, where B^2 is a standard Brownian motion which is independent from B^1. Therefore, $\big(B^1, B^2\big)^\intercal$ can be viewed as a standard two-dimensional Brownian motion. We assume that the parameters $\kappa, \theta_\nu, \xi \in \mathbb{R}^+$ satisfy the so-called Feller condition $2\kappa\theta_\nu > \xi^2$, which ensures that ν is strictly positive. Since ν is not necessarily progressively measurable with respect to the filtration generated by B^1, this market is not yet of the form discussed in Section 2.1. Hence, we introduce an artificial stock, whose local martingale part is driven by B^2:

$$dS^2(t) = S^2(t)\,r\,dt + S^2(t)\,dB^2(t), \quad t \in [0,T], \tag{5.17}$$

which is excluded from trading by requiring that portfolio processes have to take values in $K \times \{0\}$. Moreover, we have $\theta = (A\sqrt{\nu}, 0)^\intercal$ and $\sigma = \left(\begin{smallmatrix}\sqrt{\nu} & 0\\ 0 & 1\end{smallmatrix}\right)$ in this two-dimensional setup. Since the second component of admissible controls necessarily has to be 0, we obtain that B^2 influences the wealth process only through ν. For notational convenience, we identify every control process with its first component. Hence, for each $\pi \in \mathcal{A}$, the wealth process X^π satisfies

$$dX^\pi(t) = X^\pi(t)\,(r + \pi(t)A\nu(t))\,dt + X^\pi(t)\,\pi(t)\sqrt{\nu(t)}\,dB^1(t), \quad t \in [0,T], \tag{5.18}$$

where the initial wealth is given by a fixed number $x_0 \in \mathbb{R}^+$. Note that the unboundedness of ν is not an issue here because its invertibility is ensured by the Feller condition. Hence, the deep primal SMP algorithm is applicable to this problem.

From the derivation of (2.12) and the above considerations we conclude that for every $(y, v_1, v_2) \in \mathcal{D}$, $(v_1, v_2) := v$, the associated dual state process satisfies $Y^{(y,v_1,v_2)}(0) = y$ and

$$\begin{aligned} dY^{(y,v_1,v_2)}(t) = -Y^{(y,v_1,v_2)}(t)\big[\big(r + \delta_K(v_1(t))\big)\,dt + \big(A\sqrt{\nu(t)} + v_1(t)/\sqrt{\nu(t)}\big)\,dB^1(t)\\ + v_2(t)\,dB^2(t)\big], \quad t \in [0,T]. \end{aligned} \tag{5.19}$$

Following the argument in [5], it can be shown by means of Theorem 4.6 that also the deep SMP algorithm only has to learn a one-dimensional control process. It is an immediate consequence of (4.7), p_2^* being strictly positive and the form of σ that the second component of q_2^* is the zero process. Since $\delta_{K\times\{0\}}$ does not depend on the second component of its input as well, we conclude that (4.8) is independent from v_2^*. Hence, the second component can be set to zero and does not need to be trained. As the function s_K guarantees (4.7), it is sufficient that also the neural networks $\mathcal{N}_{\theta_{i,q}}$ map to a one-dimensional set.

As already mentioned above, also the deep controlled 2BSDE algorithm can be applied here by increasing the dimension of the state processes. To be more precise, we consider the processes $\big(X^\pi, \nu\big)^\intercal$ and $\big(Y^{(y,v_1,v_2)}, \nu\big)^\intercal$ instead of just X^π and $Y^{(y,v_1,v_2)}$, respectively. Obviously, the value function, therefore, has three arguments. This adjustment converts the problems into Markovian problems since the resulting two-dimensional SDE systems satisfy the two-dimensional analogue of (3.1). For the primal problem, we conclude from (5.16) and (5.18) that the functions a and b are given by

$$a(t,x,\nu,\pi) = \big(x(r+\pi A\nu),\ \kappa(\theta_\nu-\nu)\big)^\intercal,\ b^\intercal(t,x,\nu,\pi) = \begin{pmatrix} x\pi\sqrt{\nu} & 0 \\ \rho\xi\sqrt{\nu} & \sqrt{1-\rho^2}\xi\sqrt{\nu}\end{pmatrix}, \tag{5.20}$$

where $(t,x,\nu,\pi)\in[0,T]\times\mathbb{R}^+\times\mathbb{R}^+\times K$. The terminal gain function is defined by $g(x,\nu) = U(x)$, $(x,\nu)\in(\mathbb{R}^+)^2$. For the dual problem, we obtain from (5.16) and (5.19):

$$\begin{aligned}\widetilde{a}(t,y,\nu,v_1,v_2) &= \big(-y(r+\delta_K(v_1)),\ \kappa(\theta_\nu-\nu)\big)^\intercal,\\ \widetilde{b}^\intercal(t,y,\nu,v_1,v_2) &= \begin{pmatrix} -y\big(A\sqrt{\nu}+v_1/\sqrt{\nu}\big) & -yv_2 \\ \rho\xi\sqrt{\nu} & \sqrt{1-\rho^2}\xi\sqrt{\nu}\end{pmatrix},\end{aligned} \tag{5.21}$$

where $(t,y,\nu,v_1,v_2)\in[0,T]\times\mathbb{R}^+\times\mathbb{R}^+\times\widetilde{K}\times\mathbb{R}$. Furthermore, the terminal gain function is defined by $\widetilde{g}(y,\nu) = \widetilde{U}(y)$, $(y,\nu)\in(\mathbb{R}^+)^2$.

In the following example, we study an unconstrained problem with power utility function for various terminal times T. This choice is insofar appealing, as explicit solutions are known in this case, which allows us to assess the quality of the obtained estimates.

Example 5.8. We consider Heston's stochastic volatility model, as introduced above. Moreover, we assume that trading is not restricted, i.e. $K=\mathbb{R}$. Motivated by Example 5.1 from [17], we choose $r=0.05$, $A=0.5$, $\kappa=10$, $\theta_\nu=0.05$, $\xi=0.5$, $\rho=-0.5$, $x_0=1$, $v_0=0.5$ and the power utility function with parameter $p=0.5$. Clearly, the Feller condition is satisfied. However, as the process can still become negative due to the SDE discretization error, we truncate $(\nu_i)_{i\in\{0,\dots,N\}}$ at $\delta := 10^{-5}$. We consider three different values for T, namely 0.2, 0.5 and 1. As indicated above, the value of an unconstrained power utility maximization problem in a Heston stochastic volatility setting can be found explicitly. We refer to [17] for details, where the HJB equation of the dual problem is solved by means of an appropriate ansatz which yields a system of two Riccati equations. Moreover, it is proved that the duality gap is indeed zero. We refer to Table 5.9 for the exact benchmark values for all three parameter configurations.
For our numerical experiments, we select $b_{size}=64$, $N_{MC}=10^5$ and the same network

architecture as above. Moreover, we choose N such that T/N is constant for all three configurations (cf. Table 5.9). The learning rate schedules are given in Table 5.8 below. Note that we chose rather conservative learning rate schedules for both algorithms, which tackle the primal problem directly, as they converged exceptionally fast. This holds except for the deep controlled 2BSDE algorithm and $T = 1$, where too high learning rates even led to the divergence of the algorithm.

Algorithm	LR $\mathcal{L}_{(2)BSDE}$	LR $\mathcal{L}^i_{control}$/ LR $\mathcal{L}_{dual}$
2BSDE	$10^{-2} \overset{500}{\to} 10^{-3} \overset{1500}{\to} 10^{-4} \overset{4000}{\to} 10^{-5}$	$10^{-3} \overset{500}{\to} 10^{-4} \overset{1500}{\to} 10^{-5} \overset{4000}{\to} 10^{-6}$
2BSDE_dual	$10^{-2} \overset{1000}{\to} 10^{-3} \overset{2000}{\to} 10^{-4} \overset{3000}{\to} 10^{-5}$	$10^{-3} \overset{1000}{\to} 10^{-4} \overset{2000}{\to} 10^{-5} \overset{3000}{\to} 10^{-6}$
SMP	$10^{-2} \overset{1000}{\to} 10^{-3} \overset{2000}{\to} 10^{-4} \overset{3000}{\to} 10^{-5}$	$10^{-3} \overset{1000}{\to} 10^{-4} \overset{2000}{\to} 10^{-5} \overset{3000}{\to} 10^{-6}$
SMP_primal	$10^{-2} \overset{200}{\to} 10^{-3} \overset{800}{\to} 10^{-4} \overset{4000}{\to} 10^{-5}$	$10^{-3} \overset{200}{\to} 10^{-4} \overset{800}{\to} 10^{-5} \overset{4000}{\to} 10^{-6}$

Table 5.8: Piecewise constant learning rate schedules for our numerical experiments using TensorFlow.

Choosing values for the hyperparameter ε which are larger than its default value was again highly favorable (cf. Remark 5.2). For both algorithms tackling the primal problem we selected $\varepsilon = 1$ and for the remaining two methods $\varepsilon = 100$. We ran each algorithm for 10000 training steps. The final results are provided in the following table.

(T, N)	V	$\widetilde{V}$	$\widetilde{V}_l$	$\widetilde{V}_u$	V_l	V_u	Benchmark
$(0.2, 6)$	2.02269	2.02278	2.02246	2.02285	2.02228	2.02296	2.02225
$(0.5, 15)$	2.04237	2.04284	2.04249	2.04301	2.04283	2.04298	2.04268
$(1, 30)$	2.06609	2.07904	2.07431	2.07525	2.07474	2.07705	2.07484

Table 5.9: Value approximations for the studied unconstrained problem with stochastic volatility at the end of the training procedure for various pairs (T, N) in comparison with the corresponding theoretical benchmarks.

We observe that all value approximations, except V for $(T, N) = (1, 30)$, are highly accurate. This outlier is most likely not a consequence of unfortunate initial guesses as repeating the experiment several times yielded approximately the same value. Interestingly enough, reducing N to 5 resulted in $V = 2.07498$ which would correspond to the second best approximation in the last row of Table 5.9. Hence, this result could be used for verifying the results of the other algorithms, if no theoretical benchmark was known. A trader can still use the control process determined by the deep primal SMP algorithm. This illustrates the luxury of having four independent methods for solving utility maximization problems. Moreover, we conclude from Table 5.9, in particular from the last row, that increasing T while keeping T/N constant usually increases the approximation error. In [5], a similar observation is made for the deep controlled 2BSDE algorithm and the classical unconstrained Merton problem.
Motivated by Remark 5.9 below, we refined the deep primal SMP algorithm for the above

experiments by also using the current volatility as an input for the neural networks $\mathcal{N}_{\theta_{i,q}}$, $i \in \{1, \ldots, N-1\}$. Table 5.9 suggests that this is a worthwhile refinement, as V_l corresponds to the best value approximation for all of the considered pairs (T, N).
Figure 5.8 depicts the evolution of the value approximations during the training procedure for $(T, N) = (0.5, 15)$.

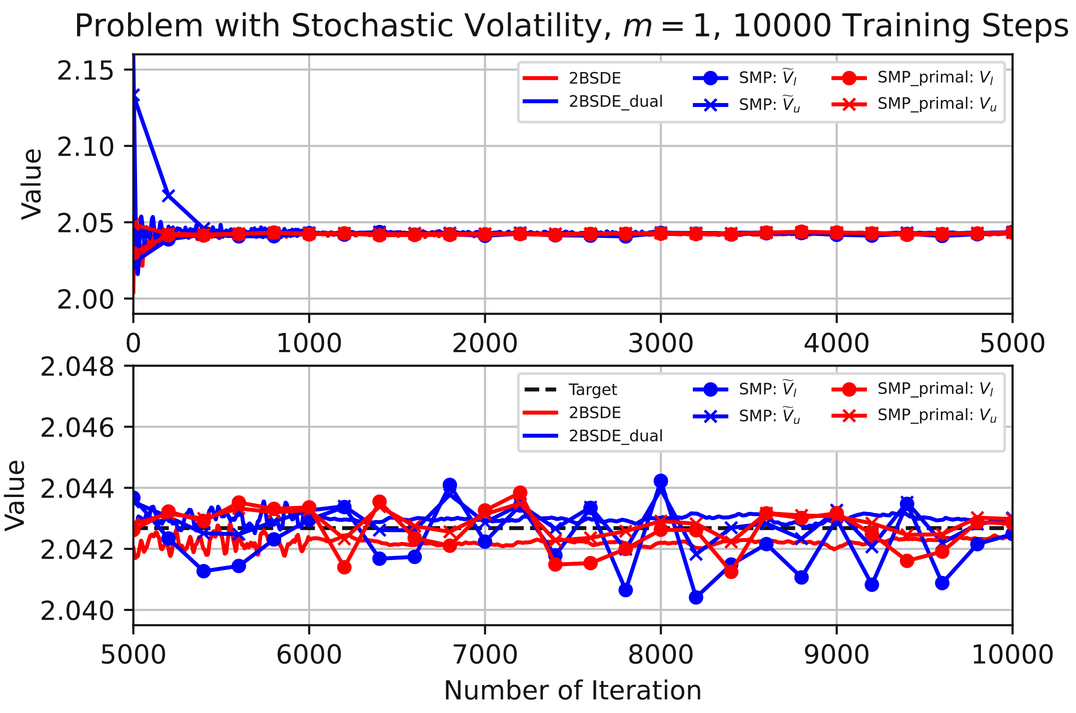

Figure 5.8: Value approximations for the studied unconstrained problem with stochastic volatility in the course of 10000 training steps for $(T, N) = (0.5, 15)$.

As in many of the previous examples, we observe that it only takes 1000 training steps for arriving at reasonable estimates. Finally, we intend to assess the control approximation quality for both algorithms tackling the primal problem directly. We choose again $(T, N) = (0.5, 15)$, as the deep controlled 2BSDE algorithm did not even yield a satisfying value approximation for $(T, N) = (1, 30)$. In contrast to this, we observed that the control process determined by the deep primal SMP algorithm boasts a similar accuracy as the one obtained for the problem with $(T, N) = (0.5, 15)$.
Figure 5.9 illustrates the control processes determined by the deep controlled 2BSDE algorithm and the deep primal SMP algorithm, respectively. For this purpose, the neural networks $\mathcal{N}_{\theta_{i,\pi}}$ are depicted on $[0.7, 1.4] \times [0, 0.25]$ for several indices i. We recall that the neural networks $\mathcal{N}_{\theta_{i,\pi}}$ take only a one-dimensional input in the deep primal SMP algorithm. Hence, the corresponding surfaces in Figure 5.9 are constant in ν. For each subplot and corresponding t_i, the domain of the color map is centered around the deterministic random variable $\pi^*(t_i)$. We refer to [17] for a derivation of the optimal control process

π^*, which proves in particular that $\pi^*(t_i)$ is indeed constant. For our parameter choice, $\{\pi^*(t_i)\,|\,i \in \{0,\dots,14\}\}$ lies in $[0.9938, 0.9983]$. Hence, all the surfaces should lie in a similar region, as we shall see below.

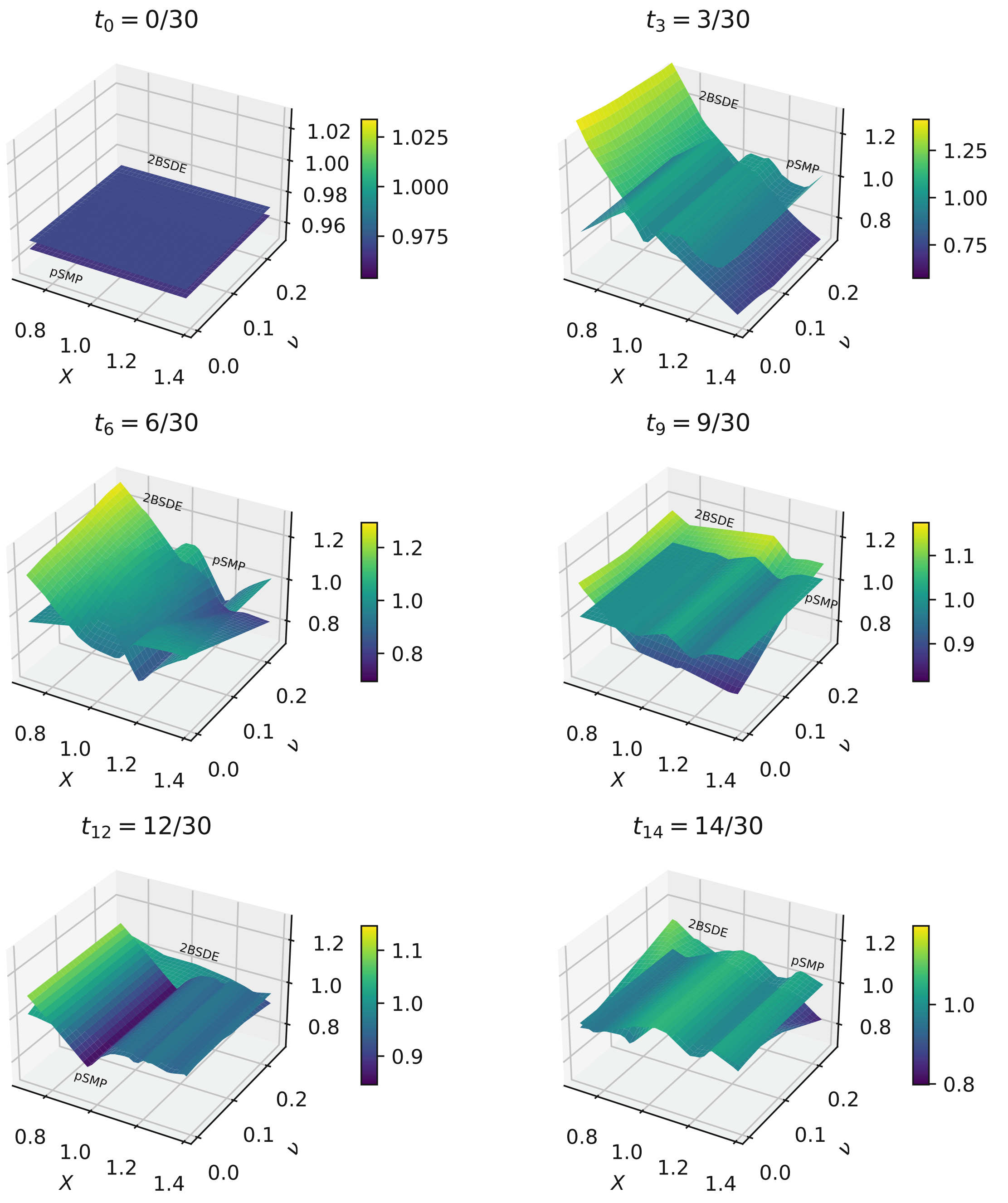

Figure 5.9: Neural networks $\mathcal{N}_{\theta_{i,\pi}}$ for $(T, N) = (0.5, 15)$, several indices i and both algorithms tackling the primal problem directly.

We observe that the approximation quality is good for both algorithms. This holds in particular for large indices i. Again, the relatively large approximation errors for small indices and pairs whose first component is farther away from 1 have to be assessed considering the fact that the wealth at time t_i is close to $x_0 = 1$ with high probability. Note, however, that the deep primal SMP algorithm achieves a significantly better approximation for these indices. Finally, we compare the maximal absolute errors with respect to the exact solution on $[0.7, 1.4] \times [0, 0.25]$.

Algorithm	$i = 0$	$i = 3$	$i = 6$	$i = 9$	$i = 12$	$i = 14$
2BSDE	0.02232	0.41483	0.28734	0.15769	0.08373	0.16496
SMP_primal	0.02771	0.09208	0.14719	0.03757	0.14283	0.06164

Table 5.10: Maximal deviations of the neural networks $\mathcal{N}_{\theta_{i,\pi}}$ from the deterministic optimal control process at t_i on $[0.7, 1.4] \times [0, 0.25]$, i.e. $\|\pi^*(t_i) - \mathcal{N}_{\theta_{i,\pi}}\|_\infty$, for the same indices i as in Figure 5.9.

As in Figure 5.9, we observe that the deep primal SMP algorithm yields a more accurate control process than the deep controlled 2BSDE algorithm.

Remark 5.9. Extending the input vectors of the neural networks $\mathcal{N}_{\theta_{i,q}}$ in the deep primal SMP algorithm by $\sqrt{\nu_i}$ is motivated by (3.48) and the definition of the function b in (5.20). This refinement improved the results in all of our numerical experiments.

In the following example, we are going to calibrate the Heston model to market data, which puts the corresponding real-life portfolio optimization problem into the scope of the studied algorithms. All the data was collected on the 23rd of April 2021.

Example 5.10. We consider the Dow Jones Industrial Average (DJIA). At first, we wish to find parameters which describe its dynamics sufficiently well. As we consider a problem with $T = 0.5$, we choose r as the current yield of a U.S. 6 Month Treasury Bill, i.e. 0.0003. The parameters κ, θ_ν, ξ, ρ and ν_0 were calibrated by minimizing the error between model and market values for 55 European options whose expiration dates lie within the next 8 months. Note that the underlying for the most common European call options on the DJIA is actually one-hundredth of its value process (DJX). However, it is an immediate consequence of (5.15) and (5.16) that the above parameters agree for DJIA and DJX, respectively. We considered options whose strike prices range from 290 to 390. The calibration procedure was implemented in Python by means the pricing engine provided by QuantLib. Moreover, Scipy's implementation of the differential evolution algorithm served as our optimizer. We obtained $\kappa = 14.133$, $\theta_\nu = 0.039$, $\xi = 2.687$, $\rho = -0.544$ and $\nu_0 = 0.022$. Note that the Feller condition is violated, which is not surprising as this phenomenon is quite common in the market. In theory, this means that ν, which still exists, can attain 0. However, as we discretize (5.16) for all algorithms according to the Euler-Maruyama scheme, this event (and even negative values) were also possible, if the Feller condition held. Hence, truncating $(\nu_i)_{i\in\{0,\dots,N\}}$ at a small positive number δ as in Example 5.8 is necessary in both cases. Therefore, the violation of the Feller condition merely implies for the discretization procedure that the process might fall below δ more often, which leads to a potentially larger approximation error. Figure 5.10 illustrates the quality of the calibrated parameters as the model's implied volatility surface bears great resemblance to the one observed in the market.

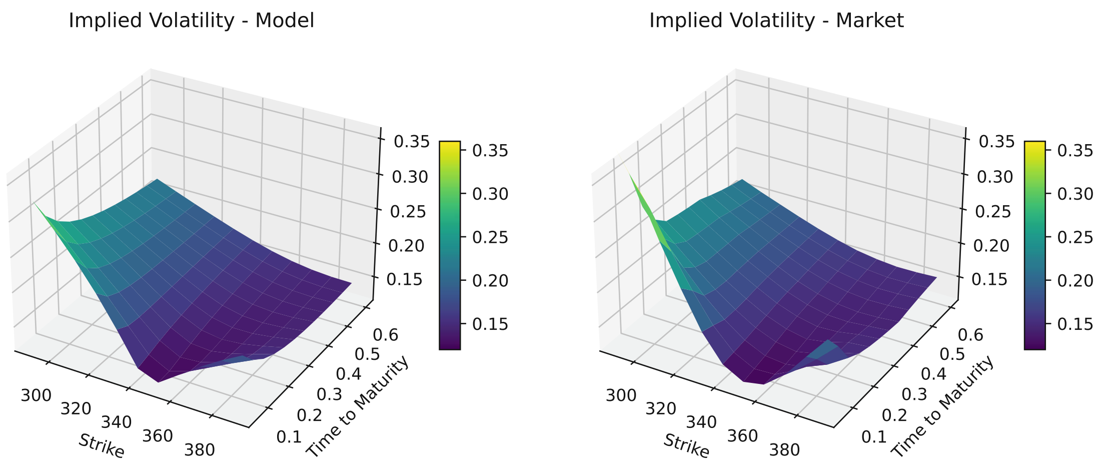

Figure 5.10: Implied volatility surface for our calibrated Heston model in comparison with the observed surface in the market.

Finally, we have to estimate A. This is a delicate task as option pricing models, i.e. the resulting formulas, do not include the original drift parameter due to the fact that pricing is performed under a risk-neutral measure. For simplicity, we assume that the drift coefficient of dS/S is constant and the parameter θ_ν was the same in the past. By calculating the arithmetic mean of the daily logarithmic returns of the last 20 years and multiplying it with the number of trading days per year we obtain $\widetilde{\mu}$. Our simple, but certainly not optimal, estimator is then given by $(\widetilde{\mu} + \theta_\nu/2 - r)/\theta_\nu$, i.e. $A = 2.027$. We consider an investor with initial wealth $x_0 = 1$ who selects the power utility function with parameter $p = 1/3$. Hence, our investor is more risk averse than in our previous examples with power utility. Moreover, he intends to invest in the following six months, i.e. $T = 0.5$. We assume that selling stocks short is not permitted and the amount of money the investor can borrow is limited by twice his current wealth, i.e. $K = [0, 3]$. It follows immediately that $\widetilde{K} = \mathbb{R}$ and $\delta_K(z) = (-3z)^+$, $z \in \mathbb{R}$, hold. For our numerical experiments, we choose $N = 30$, $b_{size} = 64$ and $N_{MC} = 10^5$. The learning rate schedules are given in Table 5.11.

Algorithm	LR $\mathcal{L}_{(2)BSDE}$	LR $\mathcal{L}^i_{control}$/ LR $\mathcal{L}_{dual}$
2BSDE	$10^{-2} \overset{500}{\to} 10^{-3} \overset{500}{\to} 10^{-4} \overset{2000}{\to} 10^{-5}$	$10^{-3} \overset{500}{\to} 10^{-4} \overset{500}{\to} 10^{-5} \overset{2000}{\to} 10^{-6}$
2BSDE_dual	$10^{-2} \overset{500}{\to} 10^{-3} \overset{500}{\to} 10^{-4} \overset{3000}{\to} 10^{-5}$	$10^{-3} \overset{500}{\to} 10^{-4} \overset{500}{\to} 10^{-5} \overset{3000}{\to} 10^{-6}$
SMP	$10^{-2} \overset{1000}{\to} 10^{-3} \overset{2000}{\to} 10^{-4} \overset{3000}{\to} 10^{-5}$	$10^{-3} \overset{1000}{\to} 10^{-4} \overset{2000}{\to} 10^{-5} \overset{3000}{\to} 10^{-6}$
SMP_primal	$10^{-2} \overset{500}{\to} 10^{-3} \overset{500}{\to} 10^{-4} \overset{4000}{\to} 10^{-5}$	$10^{-3} \overset{500}{\to} 10^{-4} \overset{500}{\to} 10^{-5} \overset{4000}{\to} 10^{-6}$

Table 5.11: Piecewise constant learning rate schedules for our numerical experiments using TensorFlow.

We choose to reduce the learning rates rather quickly as this has a beneficial stabilization effect. For the deep SMP algorithm, we select a more refined network architecture (cf. Remark 5.11) and $s_K = 3\,(1 + |\cdot|)^{-1}$. Moreover, the same function serves as projection function for the controls in both algorithms which tackle the primal problem directly. For all algorithms, except the dual version of the deep controlled 2BSDE algorithm, we choose $\varepsilon = 1$ for all batch normalization layers. In contrast to this, we select $\varepsilon = 100$ in the latter algorithm as the previous choice might lead to NaN-values. We ran each of the studied algorithms for 10000 training steps. The training progress is depicted in Figure 5.11 below.

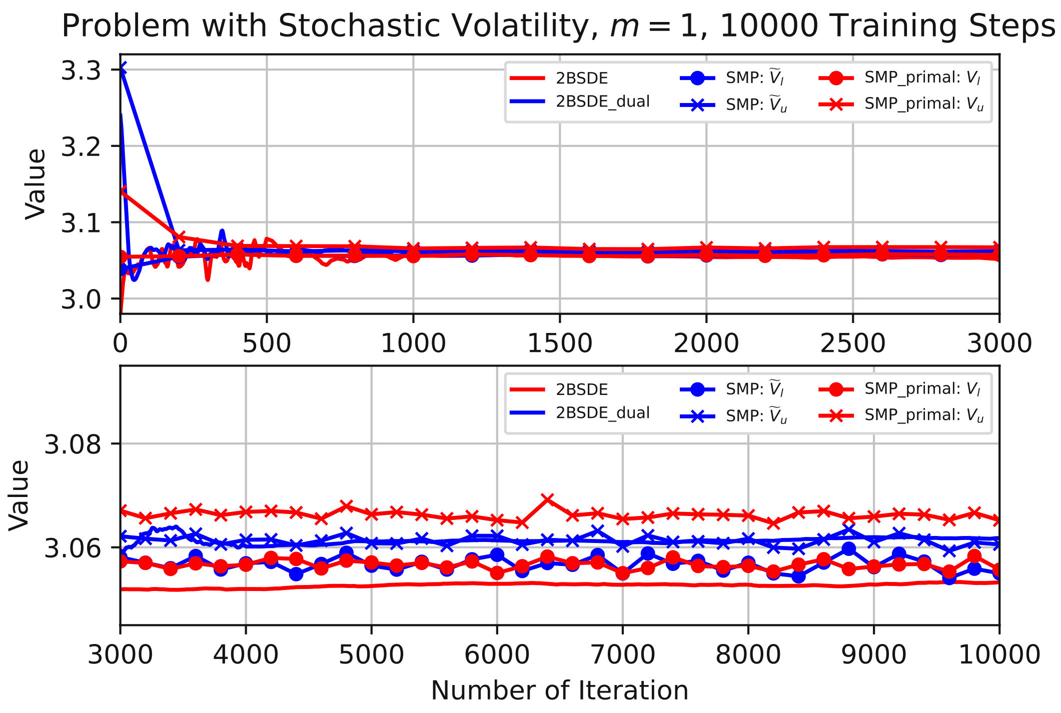

Figure 5.11: Value approximations for the studied constrained problem with stochastic volatility and parameters calibrated to market data in the course of 10000 training steps.

At the end of the training procedure, we obtain

$$\begin{aligned} &V = 3.05326, \quad \widetilde{V} = 3.06176, \quad \widetilde{V}_l = 3.05501, \quad \widetilde{V}_u = 3.06055, \\ &V_l = 3.05569 \quad \text{and} \quad V_u = 3.06519. \end{aligned} \tag{5.22}$$

As deriving a theoretical benchmark for this constrained problem is certainly very complex, we content ourselves with comparing the results to 3.05373, i.e. the value of the corresponding unconstrained problem. This is justified since the (deterministic) optimal control π^* of the latter problem lies for almost every t_i, $i \in \{0, \dots, N-1\}$, in K. Hence, the value of the constrained problem should be slightly lower than the above benchmark. We observe that all the obtained approximations lie within the same region. However, it has to be noted that the values are not as concentrated as in Example 5.8. This may be caused on the one hand by the constraints and on the other hand, and perhaps most importantly, by the discretization error for the process ν as the probability of ν_i being replaced by δ is significant. The key difference is that ξ is larger and ν_0 significantly smaller than in Example 5.8. One possible approach to weaken this effect is choosing a more complex simulation scheme for $(\nu_i)_{i\in\{0,\dots,N\}}$. Moreover, we observed that increasing N substantially reduces the gaps which is why we chose $N = 30$. Finally, our investor can apply the obtained strategy in the future by adjusting the portfolio in accordance with the neural networks $\mathcal{N}_{\theta_{i,\pi}}$, $i \in \{0,\dots,N-1\}$.

Remark 5.11. In a similar manner to Remark 5.9 and motivated by (3.48) and the definition of the function $\widetilde{b}$ in (5.21), we extend the input vectors of the neural networks $\mathcal{N}_{\theta_{i,q}}$ by $\sqrt{\nu_i}$ in the deep SMP algorithm. While this refinement was not necessary for the unconstrained problem in Example 5.8, it improved the value approximation for the constrained problem in Example 5.10 quite significantly. This adjustment reduced the duality gap approximately by a factor of three.

Finally, we consider the utility maximization setup from Chapter 2, where the short rate r is, as in the Vasicek model, given by an Ornstein-Uhlenbeck process. Hence, the dynamics of r is given by

$$dr(t) = \alpha(\beta - r(t))\,dt + \gamma\,dB^{m+1}(t), \quad t \in [0,T], \tag{5.23}$$

with initial condition $r(0) = r_0 \in \mathbb{R}$ and (positive) real-valued parameters α, β and γ. Moreover, we assume that the Brownian motions B^{m+1} and B are independent. Hence, $\overline{B} := (B, B^{m+1})^\intercal$ can be viewed as a standard $(m+1)$-dimensional Brownian motion. Since r is not progressively measurable with respect to the filtration generated by B, we introduce an additional stock (cf. (5.17)), whose local martingale part is driven by B^{m+1}:

$$dS^{m+1}(t) = S^{m+1}(t)\,r(t)\,dt + S^{m+1}(t)\,dB^{m+1}(t), \quad t \in [0,T]. \tag{5.24}$$

As in our considerations for the Heston model, this stock cannot be traded by the investor, which ensures that the value of the problem remains unchanged under this generalization. Hence, we have a market of the form as introduced in Chapter 2 which is driven by $\overline{B}$. Therefore, both SMP-based algorithms are applicable. Following the argument presented for the Heston model, it can be shown that also the dual control process is essentially m-dimensional. For notational convenience we, therefore, identify the processes π and v with their first m components in the following.

Like in the stochastic volatility setting, also the deep controlled 2BSDE algorithm can be applied here by increasing the dimension of the state processes, i.e. we consider the processes $\big(X^\pi, r\big)^\intercal$ and $\big(Y^{(y,v)}, r\big)^\intercal$ instead of X^π and $Y^{(y,v)}$, respectively. For the primal problem, we obtain from (2.3) and (5.23) that the functions a and b are given by

$$\begin{aligned} a(t,x,r,\pi) &= \big(x(r + \pi^\intercal(\mu(t) - r(1,\dots,1)^\intercal)),\ \alpha(\beta - r)\big)^\intercal, \\ b^\intercal(t,x,r,\pi) &= \begin{pmatrix} x\pi^\intercal\sigma(t) & 0 \\ 0 & \gamma \end{pmatrix}, \end{aligned} \tag{5.25}$$

where $(t,x,r,\pi) \in [0,T] \times \mathbb{R}^+ \times \mathbb{R} \times K$. For the dual problem, we conclude from (2.12) and (5.23):

$$\begin{aligned} \widetilde{a}(t,y,r,v) &= \big(-y(r+\delta_K(v)),\ \alpha(\beta - r)\big)^\intercal, \\ \widetilde{b}^\intercal(t,y,r,v) &= \begin{pmatrix} -y\big(\sigma^{-1}(t)\big(\mu(t) - r(1,\dots,1)^\intercal + v\big)\big)^\intercal & 0 \\ 0 & \gamma \end{pmatrix}, \end{aligned} \tag{5.26}$$

where $(t,y,r,v) \in [0,T] \times \mathbb{R}^+ \times \mathbb{R} \times \widetilde{K}$. Furthermore, the terminal gain functions are given by $g(x,r) = U(x)$ and $\widetilde{g}(y,r) = \widetilde{U}(y)$, for all $(x,r),(y,r) \in \mathbb{R}^+ \times \mathbb{R}$.

Example 5.12. We consider a 30-dimensional problem, where short selling is not permitted, i.e. $K = (\mathbb{R}_0^+)^{30}$. Moreover, we choose $U = \log$, $x_0 = 10$ and $T = 0.5$. The parameters for the short rate process are selected as $r_0 = 0.05$, $\alpha = 5$, $\beta = 0.05$ and $\gamma = 0.05$. The processes μ and σ are assumed to be deterministic. For every $t \in [0,0.5]$ and $i,j \in \{1,\dots,30\}$, we choose $\mu_i(t) = 0.06 + 0.01\sin(4\pi t + \pi i/15)$, $\sigma_{i,i}(t) = 0.3/(1+t)$ and $\sigma_{i,j}(t) = 0.05$, if $j \neq i$ holds. According to the above considerations, all of the studied algorithms are applicable to this specific problem.
For our numerical experiments, we select $N = 20$, $b_{size} = 64$ and $N_{MC} = 10^5$. We ensure that the control processes map to K by applying the function $x \mapsto x^2$ componentwise to the outputs of the final dense layers of the corresponding neural networks. Moreover, the same function serves as projection s_K in the deep SMP algorithm. Since $\widetilde{K} = K$ holds for this particular problem, we choose the same projection function for the dual control processes. The learning rate schedules are given in Table 5.12 below.

Algorithm	LR $\mathcal{L}_{(2)BSDE}$	LR $\mathcal{L}^i_{control}$/ LR $\mathcal{L}_{dual}$
2BSDE	$10^{-2} \overset{1000}{\to} 10^{-3} \overset{2000}{\to} 10^{-4} \overset{2000}{\to} 10^{-5}$	$10^{-3} \overset{1000}{\to} 10^{-4} \overset{2000}{\to} 10^{-5} \overset{2000}{\to} 10^{-6}$
2BSDE_dual	$10^{-2} \overset{1000}{\to} 10^{-3} \overset{2000}{\to} 10^{-4} \overset{2000}{\to} 10^{-5}$	$10^{-3} \overset{1000}{\to} 10^{-4} \overset{2000}{\to} 10^{-5} \overset{2000}{\to} 10^{-6}$
SMP	$10^{-2} \overset{1000}{\to} 10^{-3} \overset{2000}{\to} 10^{-4} \overset{2000}{\to} 10^{-5}$	$10^{-2} \overset{1000}{\to} 10^{-3} \overset{2000}{\to} 10^{-4} \overset{2000}{\to} 10^{-5}$
SMP_primal	$10^{-2} \overset{1000}{\to} 10^{-3} \overset{2000}{\to} 10^{-4} \overset{2000}{\to} 10^{-5}$	$10^{-3} \overset{1000}{\to} 10^{-4} \overset{2000}{\to} 10^{-5} \overset{2000}{\to} 10^{-6}$

Table 5.12: Piecewise constant learning rate schedules for our numerical experiments using TensorFlow.

As in many of our previous examples, choosing larger values for ε, e.g. $\varepsilon = 1$ or $\varepsilon = 100$, is highly favorable with regards to the convergence of the studied algorithms. We ran each algorithm for 10000 training steps. The evolution of the implied value approximations in the course of this procedure is depicted in Figure 5.12 below. We observe that most bounds converge astonishingly fast. The initial variability of V_u can be explained by the fact that $-p$ is used as an estimator for the parameter y from the dual problem (cf. (4.79)). Since p is optimized with respect to $\mathcal{L}_{BSDE}$, its initial learning rate is 10^{-2} according to Table 5.12, whereas the learning rate schedule for y starts at 10^{-3} for both algorithms tackling the dual problem. Once the learning rate is reduced at training step 1000, V_u closes the duality gap with respect to V_l. In contrast to this, the deep SMP algorithm yields a steady duality gap which does not decrease as the training procedure progresses. While $\widetilde{V}_l$ remains close to the other value approximations, $\widetilde{V}_u$ yields a significantly higher value. Interestingly

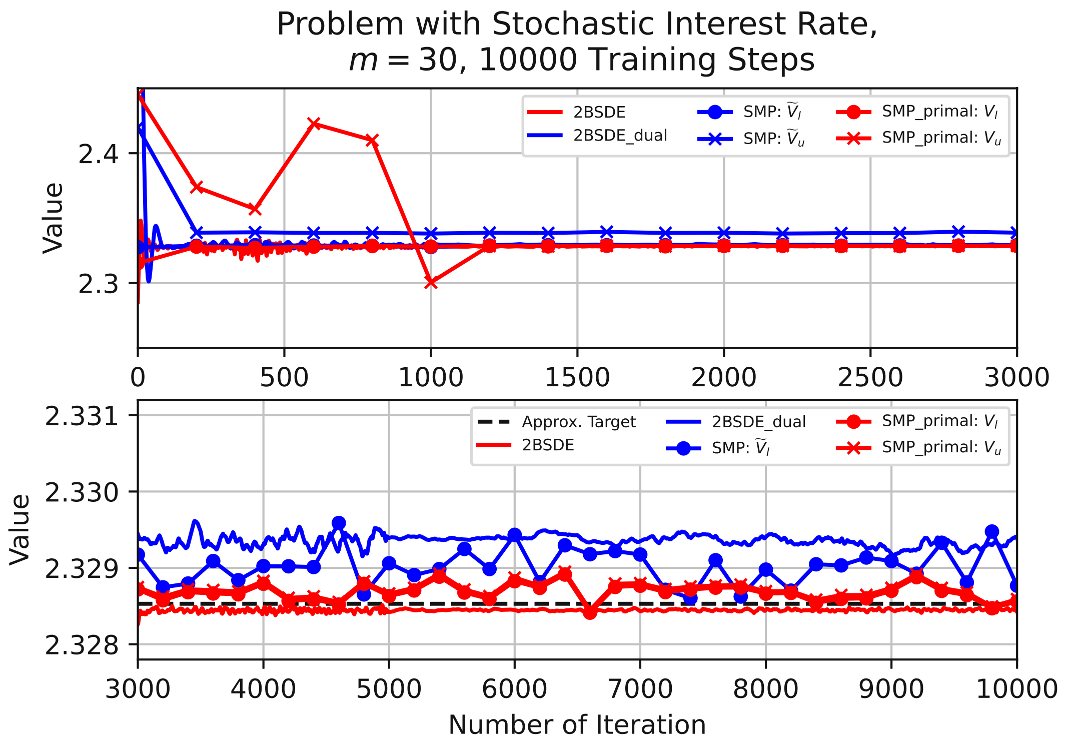

Figure 5.12: Value approximations for the studied high-dimensional, constrained problem with stochastic interest rate in the course of 10000 training steps.

enough, applying the other algorithms to the corresponding unconstrained problem yields values which are very close to $\widetilde{V}_u$. Hence, this is another example, where the deep SMP algorithm mistakes $v \equiv 0$ for the optimal dual control. At the end of the training procedure, we obtain the following estimates for our constrained problem:

$$\begin{aligned} &V = 2.32842, \quad \widetilde{V} = 2.32939, \quad \widetilde{V}_l = 2.32877, \quad \widetilde{V}_u = 2.33868, \\ &V_l = 2.32856 \quad \text{and} \quad V_u = 2.32859. \end{aligned} \tag{5.27}$$

As all values except $\widetilde{V}_u$ lie within an interval of length 10^{-3}, we are highly confident that also the true value of the problem lies within this region. Moreover, we solved the corresponding problem with deterministic short rate $r \equiv r_0$ by means of the machinery described in Examples 5.4 and 5.5, which led to 2.32853. While this is certainly not an exact benchmark, it emphasizes that the values in (5.27) are plausible.

6 Conclusion and Future Work

In the course of this thesis, we studied the constrained utility maximization problem and its dual problem with regard to theoretical results which allow the formulation deep learning based, algorithmic solvers. If the processes r, μ and σ are either deterministic or satisfy their own SDEs, as in the Heston or Vasicek model, we can apply the deep controlled 2BSDE algorithm by potentially increasing the dimension of the state processes. We observed that the underlying theoretical result essentially holds, if the value function is sufficiently smooth and satisfies a growth condition. Moreover, it is clear that this algorithm could even be used for more general control problems with running gains and a not necessarily concave/convex terminal gain function. In contrast to this, the strict concavity of U is essential for our considerations leading to the formulation of the deep SMP algorithm and the deep primal SMP algorithm, respectively. For this purpose, we derived a stochastic maximum principle for both problems while also proving the reverse implications. One decisive advantage of these algorithms is that they can also handle path dependent random coefficients. Hence, the deep SMP algorithm and the novel deep primal SMP algorithm are suitable for solving a larger class of utility maximization problems.
Moreover, we conclude from the results of the studied constrained problems that also the upper bound implied by the deep primal SMP algorithm serves as a reasonable value approximation. This is an essential advantage over the deep SMP algorithm as the latter method tends to produce the value of the corresponding unconstrained problem as the upper bound. Combining this with the fact that the deep primal SMP algorithm outperformed its dual counterpart in Section 5.2 (cf. Figures 5.6 and 5.7) and that it produced more accurate results than both versions of the deep controlled 2BSDE algorithm in both high-dimensional examples with deterministic coefficients (cf. Examples 5.4 and 5.5) illustrates and underscores the power of our novel algorithm.
If one is interested in the control processes determined by the algorithms, e.g. in order to apply them in practice, we recommend choosing for every t_i, $i \in \{0, \dots, N-1\}$, the neural network $\mathcal{N}_{\theta_{i,\pi}}$ from one algorithm which solves the primal problem directly, whereas both algorithms tackling the dual problem can be used for verifying the obtained value of the problem. We cannot use the primal control process implied by the dual problem (cf. (4.58)) for this purpose, as the dual state process, i.e. -1 times the primal adjoint process according to Theorem 4.17, is not directly observable in the market.

We recall from Section 5.3 that extending the input vectors for the neural networks by the current value of the volatility/short rate process significantly reduced the duality gap. Clearly, this generalizes the original definition of both SMP-based algorithms. Hence, depending on the specific problem, more complex network architectures might lead to even better results. We leave the systematic study of possible generalizations and their effect on the approximation quality for our future work.

A Python Codes for the Studied Algorithms Used in Example 5.5

In the following, we provide the Python codes, adapted to the specific problem discussed in Example 5.5, for all three of the algorithms which have been studied in the course of this thesis. For the deep controlled 2BSDE algorithm, we present its primal version as well as its dual version (cf. Sections A.1 and A.2). The source codes for both SMP-based algorithms (cf. Sections A.3 and A.4) can be quite easily adapted to the setting of random coefficient problems by adding the dynamics, which are required for determining the coefficients, to the loops used for the forward simulation of the processes. For example, we included the stock prices in Examples 5.6 and 5.7, whereas we added the process ν (cf. (5.16)) in our examples covering the Heston model. Hence, these processes are simulated by means of the same discrete-time Euler-Maruyama scheme.
At the end of training, it suffices in most cases to save just the neural networks $\mathcal{N}_{\theta_{i,\pi}}, i \in \{0, \dots, N-1\}$, as these are the most crucial results for a trader wishing to apply the obtained strategy in practice. This can be done by means of the inherited save-method of the PartNetwork class.

A.1 Python Implementation of the Primal Version of the Deep Controlled 2BSDE Algorithm

```python
import tensorflow as tf
from tensorflow import keras
import numpy as np
import time
from scipy.stats import multivariate_normal

class Deepcontrolled2BSDE(keras.Model):

    def __init__(self, **kwargs):
        super(Deepcontrolled2BSDE, self).__init__(**kwargs)
        self.m = 30
        self.BBdim = 30
        self.d = 1
        self.T = 0.5
        self.N = 10
        self.dt = self.T/self.N
        self.layers_num = 4 # number of hidden + 2
        self.nodes = [11, 11] # nodes in hidden layers
        self.batch_size = 64
        self.x0 = 10
        self.schedule1 = keras.optimizers.schedules.PiecewiseConstantDecay([2000, 5000, 8000],
                                           [1e−2, 1e−3, 1e−4, 1e−5])
        self.schedule2 = [keras.optimizers.schedules.PiecewiseConstantDecay([2000, 5000, 8000],
                                           [1e−3, 1e−4, 1e−5, 1e−6]) for _ in range(self.N)]
        self.optimizer1 = keras.optimizers.Adam(learning_rate=self.schedule1)
        self.optimizer2 = [keras.optimizers.Adam(learning_rate=self.schedule2[i])
                           for i in range(self.N)]
        self.y0 = tf.Variable(np.random.uniform(size=(1, 1), low=2.1, high=2.3), trainable=True,
                              dtype=tf.float32)
        self.z0 = tf.Variable(np.random.uniform(size=(1, self.d), low=−0.1, high=0.1),
                              trainable=True, dtype=tf.float32)
        self.pi0 = tf.Variable(np.random.uniform(size=(1, self.m), low=0.0, high=0.2), trainable=True,
                         dtype=tf.float32, constraint=lambda x: tf.where(x<−1/self.m, −1/self.m, x))
        self.Q0 = tf.Variable(np.random.uniform(size=(self.d, self.d), low=−0.1, high=0.1),
                              trainable=True, dtype=tf.float32)
        self.train_startvalues = [self.y0, self.z0]
        self.ModelQ = [self.Q0] + [PartNetwork(self.m, self.d, self.layers_num, self.nodes,
                                           isQ=True) for _ in range(self.N−1)]
        self.varforloss1 = self.train_startvalues + self.ModelQ
        self.Modelpi = [self.pi0] + [PartNetwork(self.m, self.d, self.layers_num, self.nodes,
                                            isQ=False) for _ in range(self.N−1)]
        self.history_time = []
        self.history_y0 = []

    def build(self):
        for i in range(self.N−1):
            self.Modelpi[i+1](tf.zeros(shape=(1, self.d)), training = False)
            self.ModelQ[i+1](tf.zeros(shape=(1, self.d)), training = False)
        return

    # The following four methods depend essentially on the studied problem. (also d=1 here)
    def aSDE(self, t, x, pi):
        helper = tf.expand_dims(tf.range(1, limit=self.m+1, dtype = tf.float32), axis=0)
        mu = 0.07+0.02*tf.sin(4*np.pi*t+2*np.pi*helper/self.m) \
              *tf.ones(shape=(self.batch_size, self.m))
        r = 0.06*tf.exp(0.5*t)
        return x*(r + tf.reduce_sum(pi*(mu − r), axis = 1, keepdims = True))

    def aSDEx(self, t, x, pi, z):
        helper = tf.expand_dims(tf.range(1, limit=self.m+1, dtype = tf.float32), axis=0)
        mu = 0.07+0.02*tf.sin(4*np.pi*t+2*np.pi*helper/self.m) \
              *tf.ones(shape=(self.batch_size, self.m))
        r = 0.06*tf.exp(0.5*t)
        return (r + tf.reduce_sum(pi*(mu − r), axis = 1, keepdims = True))*z

    # has to be (d, BBdim)−valued in the last two dimensions
    def bSDE(self, t, x, pi):
        sigma = (0.3*(1+tf.sqrt(t))−0.1)*tf.eye(self.m) + 0.1*tf.ones(shape=(self.m, self.m))
        return tf.expand_dims(x*tf.matmul(pi, sigma), axis = 1)

    def bSDEx(self, t, x, pi, q):
        sigma = (0.3*(1+tf.sqrt(t))−0.1)*tf.eye(self.m) + 0.1*tf.ones(shape=(self.m, self.m))
        return (tf.expand_dims(tf.linalg.trace(tf.matmul(tf.expand_dims(
                 tf.matmul(pi, sigma), axis = −1), q)), axis = −1))

    def f(self, t, x, pi):
        return tf.zeros(shape=(self.batch_size, 1))

    def fx(self, t, x, pi):
        return tf.zeros(shape=(self.batch_size, self.d))

    def g(self, x):
        # such that the gradients of both branches exist, cf. documentation of tf.where
        return tf.where(x>0, tf.math.log(tf.where(x>0, x, 1)), 0)

    def gx(self, x):
        return tf.where(x>0, tf.pow(tf.where(x>0, x, 1), -1), 0)

    def classicalHam(self, t, x, pi, z, q):
        return (tf.reduce_sum(self.aSDE(t, x, pi)*z, axis = 1, keepdims = True)
                + tf.expand_dims(0.5*tf.linalg.trace(tf.matmul(tf.matmul(self.bSDE(t, x, pi),
                self.bSDE(t, x, pi), transpose_b = True), q)), axis = -1) + self.f(t, x, pi))

    def genHamx(self, t, x, pi, z, q):
        return self.aSDEx(t, x, pi, z) + self.bSDEx(t, x, pi, q) + self.fx(t, x, pi)

    def loss1(self, x, y, z):
        helper1 = tf.reduce_mean(tf.square(y - self.g(x)))
        helper2 = tf.reduce_mean(tf.reduce_sum(tf.square(z - self.gx(x)), axis = 1))
        return helper1 + helper2

    def loss2(self, t, x, pi, z, q):
        return -tf.reduce_mean(self.classicalHam(t, x, pi, z, q))

    @ tf.autograph.experimental.do_not_convert
    def simulate1(self, dW):

        X = self.x0 * tf.ones(shape=(self.batch_size, self.d))
        Y = self.y0 * tf.ones(shape=(self.batch_size, 1))
        Z = self.z0 * tf.ones(shape=(self.batch_size, self.d))

        pi = self.pi0 * tf.ones(shape=(self.batch_size, self.m))
        Q = self.Q0 * tf.ones(shape=(self.batch_size, self.d, self.d))

        Y = Y - self.f(0.0, X, pi) * self.dt \
                + tf.reduce_sum(tf.squeeze(tf.matmul(self.bSDE(0.0, X, pi),
                dW[:, :, :, 0] * tf.sqrt(self.dt)), axis = -1) * Z, axis = 1, keepdims = True)
        Z = Z - self.genHamx(0.0, X, pi, Z, tf.matmul(Q, self.bSDE(0.0, X, pi))) * self.dt \
                + tf.squeeze(tf.matmul(tf.matmul(Q, self.bSDE(0.0, X, pi)),
                dW[:, :, :, 0] * tf.sqrt(self.dt)), axis = -1)
        X = X + self.aSDE(0.0, X, pi) * self.dt \
                + tf.squeeze(tf.matmul(self.bSDE(0.0, X, pi), dW[:, :, :, 0] \
                * tf.sqrt(self.dt)), axis = -1)

        for i in range(self.N - 1):

            pi = self.Modelpi[i+1](X, training = True)
            Q = self.ModelQ[i+1](X, training = True)

            Y = Y - self.f((i+1)*self.dt, X, pi) * self.dt \
                    + tf.reduce_sum(tf.squeeze(tf.matmul(self.bSDE((i+1)*self.dt, X, pi),
                    dW[:, :, :, i+1] * tf.sqrt(self.dt)), axis = −1) * Z, axis = 1, keepdims = True)
            Z = Z − self.genHamx((i+1)*self.dt, X, pi, Z, tf.matmul(Q,
                    self.bSDE((i+1)*self.dt, X, pi))) * self.dt \
                    + tf.squeeze(tf.matmul(tf.matmul(Q, self.bSDE((i+1)*self.dt, X, pi)),
                    dW[:, :, :, i+1] * tf.sqrt(self.dt)), axis = −1)
            X = X + self.aSDE((i+1)*self.dt, X, pi) * self.dt \
                    + tf.squeeze(tf.matmul(self.bSDE((i+1)*self.dt, X, pi),
                    dW[:, :, :, i+1] * tf.sqrt(self.dt)), axis = −1)

        return X, Y, Z

    @ tf.function
    def optimize1(self, dW):
        with tf.GradientTape(watch_accessed_variables = False) as tape:
            tape.watch(self.varforloss1.trainable_variables)

            X, Y, Z = self.simulate1(dW)
            loss1 = self.loss1(X, Y, Z)

        grad1 = tape.gradient(loss1, self.varforloss1.trainable_variables)
        self.optimizer1.apply_gradients(zip(grad1, self.varforloss1.trainable_variables))
        return

    @ tf.function
    def optimize2_0(self, optimizer):
        with tf.GradientTape(watch_accessed_variables = False) as tape:
            tape.watch(self.pi0)

            X = self.x0 * tf.ones(shape=(self.batch_size, self.d))
            # Y = self.y0 * tf.ones(shape=(self.batch_size, 1))
            Z = self.z0 * tf.ones(shape=(self.batch_size, self.d))

            pi = self.pi0 * tf.ones(shape=(self.batch_size, self.m))
            Q = self.Q0 * tf.ones(shape=(self.batch_size, self.d, self.d))

            loss = self.loss2(0.0, X, pi, Z, Q)

        grad = tape.gradient(loss, self.pi0)
        optimizer.apply_gradients([(grad, self.pi0)])
        return

    @ tf.autograph.experimental.do_not_convert
    def simulate2(self, dW, N, X, Y, Z, pi, Q):

        Y = Y − self.f((N−1)*self.dt, X, pi) * self.dt \
                + tf.reduce_sum(tf.squeeze(tf.matmul(self.bSDE((N−1)*self.dt, X, pi),
                dW[:, :, :, N−1] * tf.sqrt(self.dt)), axis = −1) * Z, axis = 1, keepdims = True)
        Z = Z − self.genHamx((N−1)*self.dt, X, pi, Z, tf.matmul(Q,
                self.bSDE((N−1)*self.dt, X, pi))) * self.dt \
                + tf.squeeze(tf.matmul(tf.matmul(Q, self.bSDE((N−1)*self.dt, X, pi)),
                dW[:, :, :, N−1] * tf.sqrt(self.dt)), axis = −1)
        X = X + self.aSDE((N−1)*self.dt, X, pi) * self.dt \
                + tf.squeeze(tf.matmul(self.bSDE((N−1)*self.dt, X, pi),
                dW[:, :, :, N−1] * tf.sqrt(self.dt)), axis = −1)

```

```python
        pi = self.Modelpi[N](X, training = True)
        Q = self.ModelQ[N](X, training = True)

        return X, Y, Z, pi, Q

    def optimize2(self, dW, N, X, Y, Z, pi, Q, optimizer, Model):
        with tf.GradientTape(watch_accessed_variables = False) as tape:
            tape.watch(Model.trainable_variables)

            X, Y, Z, pi, Q = self.simulate2(dW, N, X, Y, Z, pi, Q)
            loss = self.loss2(N*self.dt, X, pi, Z, Q)

        grad = tape.gradient(loss, Model.trainable_variables)
        optimizer.apply_gradients(zip(grad, Model.trainable_variables))
        return X, Y, Z, pi, Q

    def optimize2prepare(self):
        self.optimizecontrol = [tf.function(self.optimize2).get_concrete_function(
            tf.TensorSpec(shape=[self.batch_size, self.BBdim, 1, self.N], dtype=tf.float32), i+1,
            tf.TensorSpec(shape=[self.batch_size, self.d], dtype=tf.float32),
            tf.TensorSpec(shape=[self.batch_size, 1], dtype=tf.float32),
            tf.TensorSpec(shape=[self.batch_size, self.d], dtype=tf.float32),
            tf.TensorSpec(shape=[self.batch_size, self.m], dtype=tf.float32),
            tf.TensorSpec(shape=[self.batch_size, self.d, self.d], dtype=tf.float32),
            self.optimizer2[i+1], self.Modelpi[i+1]) for i in range(self.N-1)]
        return

    def train(self, steps):
        time_start = time.time()
        for k in range(steps):
            # dW does not have the desired shape, if BBdim==1
            dW = tf.constant(multivariate_normal.rvs(size=[self.batch_size, self.BBdim, self.N]),
                             dtype=tf.float32)
            if self.BBdim == 1:
                dW = tf.expand_dims(dW, axis = 1)
            dW = tf.expand_dims(dW, axis = 2)

            self.optimize1(dW)

            optimizer = self.optimizer2[0]
            self.optimize2_0(optimizer)

            X = self.x0 * tf.ones(shape=(self.batch_size, self.d))
            Y = self.y0 * tf.ones(shape=(self.batch_size, 1))
            Z = self.z0 * tf.ones(shape=(self.batch_size, self.d))

            pi = self.pi0 * tf.ones(shape=(self.batch_size, self.m))
            Q = self.Q0 * tf.ones(shape=(self.batch_size, self.d, self.d))

            for i in range(self.N - 1):
                X, Y, Z, pi, Q = self.optimizecontrol[i](dW, X, Y, Z, pi, Q)

            self.history_time.append(time.time()-time_start)
            self.history_y0.append(self.y0.numpy())

```

```
                if k%50 == 0:
                    print("Step: %d, Time: %.2f, y0: %.4f"
                          % (k, self.history_time[-1], self.history_y0[-1]))
        return


class PartNetwork(keras.Model):

    def __init__(self, m, d, layers_num, nodes, isQ, **kwargs):
        super(PartNetwork, self).__init__(**kwargs)
        self.d = d
        self.m = m
        self.layers_num = layers_num
        self.nodes = nodes
        self.isQ = isQ
        if self.isQ == False:
            self.outdim = self.m
        else:
            self.outdim = (self.d)**2

        self.bnorm_layers = [keras.layers.BatchNormalization(epsilon=100)
                             for _ in range(self.layers_num-1)]
        self.dense_layers = [keras.layers.Dense(nodes[i], use_bias=False, activation=None)
                             for i in range(self.layers_num-2)]
        self.dense_layers.append(keras.layers.Dense(self.outdim, activation=None))
        if self.isQ == True:
            self.reshape_layer = keras.layers.Reshape((self.d, self.d))

    def call(self, x, training):
        x = self.bnorm_layers[0](x, training)
        for i in range(self.layers_num-2):
            x = self.dense_layers[i](x)
            x = self.bnorm_layers[i+1](x, training)
            x = tf.nn.relu(x)
        x = self.dense_layers[self.layers_num-2](x)
        if self.isQ == True:
            x = self.reshape_layer(x)
        if self.isQ == False:
            x = x*x-1/self.m
        return x


Model = Deepcontrolled2BSDE()
Model.build()
Model.optimize2prepare()
Model.train(10000)
```

Code A.1: Python code for the primal version of the deep controlled 2BSDE algorithm in the setting of Example 5.5.

A.2 Python Implementation of the Dual Version of the Deep Controlled 2BSDE Algorithm

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import time
from scipy.stats import multivariate_normal

class Deepcontrolled2BSDE_dual(keras.Model):

    def __init__(self, **kwargs):
        super(Deepcontrolled2BSDE_dual, self).__init__(**kwargs)
        self.m = 30
        self.BBdim = 30
        self.d = 1
        self.T = 0.5
        self.N = 10
        self.dt = self.T/self.N
        self.layers_num = 4 # number of hidden + 2
        self.nodes = [11, 11]   # nodes in hidden layers
        self.batch_size = 64
        self.x0_primal = 10
        self.x0 = tf.Variable(np.random.uniform(size=(1, self.d), low=0.2, high=0.4),
                              trainable=True, dtype=tf.float32)
        self.schedule1 = keras.optimizers.schedules.PiecewiseConstantDecay([1000, 3000, 8000],
                                                                  [1e-2, 1e-3, 1e-4, 1e-5])
        self.schedule2 = [keras.optimizers.schedules.PiecewiseConstantDecay([1000, 3000, 8000],
                                                                  [1e-3, 1e-4, 1e-5, 1e-6]) for _ in range(self.N)]
        self.schedule3 = keras.optimizers.schedules.PiecewiseConstantDecay([200, 1000, 8000],
                                                                  [1e-2, 1e-4, 1e-5, 1e-6])
        self.optimizer1 = keras.optimizers.Adam(learning_rate = self.schedule1)
        self.optimizer2 = [keras.optimizers.Adam(learning_rate = self.schedule2[i])
                           for i in range(self.N)]
        self.optimizer3 = keras.optimizers.Adam(learning_rate = self.schedule3)
        self.y0 = tf.Variable(np.random.uniform(size=(1, 1), low=1.4, high=1.6), trainable=True,
                              dtype=tf.float32)
        self.z0 = tf.Variable(np.random.uniform(size=(1, self.d), low=-0.1, high=0.1),
                              trainable=True, dtype=tf.float32)
        self.pi0 = tf.Variable(np.random.uniform(size=(1, self.m), low=0.1, high=0.4),
                               trainable=True, dtype=tf.float32, constraint=lambda z: z**2)
        self.Q0 = tf.Variable(np.random.uniform(size=(self.d, self.d), low=-0.1, high=0.1),
                              trainable=True, dtype=tf.float32)
        self.train_startvalues = [self.y0, self.z0]
        self.ModelQ = [self.Q0] + [PartNetwork(self.m, self.d, self.layers_num, self.nodes,
                                              isQ=True) for _ in range(self.N-1)]
        self.varforloss1 = self.train_startvalues + self.ModelQ
        self.Modelpi = [self.pi0] + [PartNetwork(self.m, self.d, self.layers_num, self.nodes,
                                                isQ=False) for _ in range(self.N-1)]
        self.history_time = []
        self.history_value = []

    def build(self):
        for i in range(self.N-1):
                self.Modelpi[i+1](tf.zeros(shape=(1, self.d)), training = False)
                self.ModelQ[i+1](tf.zeros(shape=(1, self.d)), training = False)
        return

    def delta_K(self, pi):
        kappa = 1/self.m
        return kappa*tf.reduce_sum(pi, axis = 1, keepdims = True)

    # The following four methods depend essentially on the studied problem. (also d=1 here)
    def aSDE(self, t, x, pi):
        r = 0.06*tf.exp(0.5*t)
        return -x*(r + self.delta_K(pi))

    def aSDEx(self, t, x, pi, z):
        r = 0.06*tf.exp(0.5*t)
        return -(r + self.delta_K(pi))*z

    def bSDE(self, t, x, pi):
        sigma_inv = tf.linalg.inv((0.3*(1+tf.sqrt(t))-0.1)*tf.eye(self.m) \
                                  + 0.1*tf.ones(shape=(self.m, self.m)))
        helper = tf.expand_dims(tf.range(1, limit=self.m+1, dtype = tf.float32), axis=0)
        mu = 0.07+0.02*tf.sin(4*np.pi*t+2*np.pi*helper/self.m) \
             *tf.ones(shape=(self.batch_size, self.m))
        r = 0.06*tf.exp(0.5*t)
        theta = tf.transpose(tf.matmul(sigma_inv, mu-r, transpose_b = True))
        return tf.expand_dims(-x*(theta + tf.transpose(tf.matmul(sigma_inv, pi,
                                                   transpose_b = True))), axis = 1)

    def bSDEx(self, t, x, pi, q):
        sigma_inv = tf.linalg.inv((0.3*(1+tf.sqrt(t))-0.1)*tf.eye(self.m) \
                                  + 0.1*tf.ones(shape=(self.m, self.m)))
        helper = tf.expand_dims(tf.range(1, limit=self.m+1, dtype = tf.float32), axis=0)
        mu = 0.07+0.02*tf.sin(4*np.pi*t+2*np.pi*helper/self.m) \
             *tf.ones(shape=(self.batch_size, self.m))
        r = 0.06*tf.exp(0.5*t)
        theta = tf.transpose(tf.matmul(sigma_inv, mu-r, transpose_b = True))
        helper2 = tf.expand_dims(-(theta + tf.transpose(tf.matmul(sigma_inv, pi,
                                                   transpose_b = True))), axis = -1)
        return tf.expand_dims(tf.linalg.trace(tf.matmul(helper2, q)), axis = -1)

    def f(self, t, x, pi):
        return tf.zeros(shape=(self.batch_size, 1))

    def fx(self, t, x, pi):
        return tf.zeros(shape=(self.batch_size, self.d))

    def g(self, x):
        return tf.where(x>0, -1-tf.math.log(tf.where(x>0, x, 1)), 0)

    def gx(self, x):
        return tf.where(x>0, -tf.pow(tf.where(x>0, x, 1), -1), 0)

    def classicalHam(self, t, x, pi, z, q):
        return (tf.reduce_sum(self.aSDE(t, x, pi)*z, axis = 1, keepdims = True)
                + tf.expand_dims(0.5*tf.linalg.trace(tf.matmul(tf.matmul(self.bSDE(t, x, pi),
                self.bSDE(t, x, pi), transpose_b = True), q)), axis = -1) + self.f(t, x, pi))

    def genHamx(self, t, x, pi, z, q):
        return self.aSDEx(t, x, pi, z) + self.bSDEx(t, x, pi, q) + self.fx(t, x, pi)

    def loss1(self, x, y, z):
        helper1 = tf.reduce_mean(tf.square(y - self.g(x)))
        helper2 = tf.reduce_mean(tf.reduce_sum(tf.square(z - self.gx(x)), axis = 1))
        return helper1 + helper2

    def loss2(self, t, x, pi, z, q):
        return tf.reduce_mean(self.classicalHam(t, x, pi, z, q))

    def loss3(self, x):
        # Note that the dual problem is only meaningful for a one-dimensional process
        return tf.reduce_mean(self.g(x)) + self.x0*self.x0_primal

    @ tf.autograph.experimental.do_not_convert
    def simulate1(self, dW):

        X = self.x0 * tf.ones(shape=(self.batch_size, self.d))
        Y = self.y0 * tf.ones(shape=(self.batch_size, 1))
        Z = self.z0 * tf.ones(shape=(self.batch_size, self.d))

        pi = self.pi0 * tf.ones(shape=(self.batch_size, self.m))
        Q = self.Q0 * tf.ones(shape=(self.batch_size, self.d, self.d))

        Y = Y - self.f(0.0, X, pi) * self.dt \
                + tf.reduce_sum(tf.squeeze(tf.matmul(self.bSDE(0.0, X, pi),
                dW[:, :, :, 0] * tf.sqrt(self.dt)), axis = -1) * Z, axis = 1, keepdims = True)
        Z = Z - self.genHamx(0.0, X, pi, Z, tf.matmul(Q, self.bSDE(0.0, X, pi))) * self.dt \
                + tf.squeeze(tf.matmul(tf.matmul(Q, self.bSDE(0.0, X, pi)),
                dW[:, :, :, 0] * tf.sqrt(self.dt)), axis = -1)
        X = X + self.aSDE(0.0, X, pi) * self.dt \
                + tf.squeeze(tf.matmul(self.bSDE(0.0, X, pi),
                dW[:, :, :, 0] * tf.sqrt(self.dt)), axis = -1)

        for i in range(self.N-1):

            pi = self.Modelpi[i+1](X, training = True)
            Q = self.ModelQ[i+1](X, training = True)

            Y = Y - self.f((i+1)*self.dt, X, pi) * self.dt \
                    + tf.reduce_sum(tf.squeeze(tf.matmul(self.bSDE((i+1)*self.dt, X, pi),
                    dW[:, :, :, i+1] * tf.sqrt(self.dt)), axis = -1) * Z, axis = 1, keepdims = True)
            Z = Z - self.genHamx((i+1)*self.dt, X, pi, Z, tf.matmul(Q,
                    self.bSDE((i+1)*self.dt, X, pi))) * self.dt \
                    + tf.squeeze(tf.matmul(tf.matmul(Q, self.bSDE((i+1)*self.dt, X, pi)),
                    dW[:, :, :, i+1] * tf.sqrt(self.dt)), axis = -1)
            X = X + self.aSDE((i+1)*self.dt, X, pi) * self.dt \
                    + tf.squeeze(tf.matmul(self.bSDE((i+1)*self.dt, X, pi),
                    dW[:, :, :, i+1] * tf.sqrt(self.dt)), axis = -1)

        return X, Y, Z

```

```
@ tf.function
def optimize1(self, dW):
    with tf.GradientTape(watch_accessed_variables = False) as tape:
        tape.watch(self.varforloss1.trainable_variables)

        X, Y, Z = self.simulate1(dW)
        loss1 = self.loss1(X, Y, Z)

    grad1 = tape.gradient(loss1, self.varforloss1.trainable_variables)
    self.optimizer1.apply_gradients(zip(grad1, self.varforloss1.trainable_variables))
    return

@ tf.function
def optimize2_0(self, optimizer):
    with tf.GradientTape(watch_accessed_variables = False) as tape:
        tape.watch(self.pi0)

        X = self.x0 * tf.ones(shape=(self.batch_size, self.d))
        # Y = self.y0 * tf.ones(shape=(self.batch_size, 1))
        Z = self.z0 * tf.ones(shape=(self.batch_size, self.d))

        pi = self.pi0 * tf.ones(shape=(self.batch_size, self.m))
        Q = self.Q0 * tf.ones(shape=(self.batch_size, self.d, self.d))

        loss = self.loss2(0.0, X, pi, Z, Q)

    grad = tape.gradient(loss, self.pi0)
    optimizer.apply_gradients([(grad, self.pi0)])
    return

@ tf.autograph.experimental.do_not_convert
def simulate2(self, dW, N, X, Y, Z, pi, Q):

    Y = Y - self.f((N-1)*self.dt, X, pi) * self.dt \
          + tf.reduce_sum(tf.squeeze(tf.matmul(self.bSDE((N-1)*self.dt, X, pi),
          dW[:, :, :, N-1] * tf.sqrt(self.dt)), axis = -1) * Z, axis = 1, keepdims = True)
    Z = Z - self.genHamx((N-1)*self.dt, X, pi, Z, tf.matmul(Q,
          self.bSDE((N-1)*self.dt, X, pi))) * self.dt \
          + tf.squeeze(tf.matmul(tf.matmul(Q, self.bSDE((N-1)*self.dt, X, pi)),
          dW[:, :, :, N-1] * tf.sqrt(self.dt)), axis = -1)
    X = X + self.aSDE((N-1)*self.dt, X, pi) * self.dt \
          + tf.squeeze(tf.matmul(self.bSDE((N-1)*self.dt, X, pi),
          dW[:, :, :, N-1] * tf.sqrt(self.dt)), axis = -1)

    pi = self.Modelpi[N](X, training = True)
    Q = self.ModelQ[N](X, training = True)

    return X, Y, Z, pi, Q

def optimize2(self, dW, N, X, Y, Z, pi, Q, optimizer, Model):
    with tf.GradientTape(watch_accessed_variables = False) as tape:
        tape.watch(Model.trainable_variables)

        X, Y, Z, pi, Q = self.simulate2(dW, N, X, Y, Z, pi, Q)
        loss = self.loss2(N*self.dt, X, pi, Z, Q)

    grad = tape.gradient(loss, Model.trainable_variables)
    optimizer.apply_gradients(zip(grad, Model.trainable_variables))

    return X, Y, Z, pi, Q

def optimize2prepare(self):
    self.optimizecontrol = [tf.function(self.optimize2).get_concrete_function(
        tf.TensorSpec(shape=[self.batch_size, self.BBdim, 1, self.N], dtype=tf.float32), i+1,
        tf.TensorSpec(shape=[self.batch_size, self.d], dtype=tf.float32),
        tf.TensorSpec(shape=[self.batch_size, 1], dtype=tf.float32),
        tf.TensorSpec(shape=[self.batch_size, self.d], dtype=tf.float32),
        tf.TensorSpec(shape=[self.batch_size, self.m], dtype=tf.float32),
        tf.TensorSpec(shape=[self.batch_size, self.d, self.d], dtype=tf.float32),
        self.optimizer2[i+1], self.Modelpi[i+1]) for i in range(self.N-1)]
    return

@tf.autograph.experimental.do_not_convert
def simulate3(self, dW):

    X = self.x0 * tf.ones(shape=(self.batch_size, self.d))
    pi = self.pi0 * tf.ones(shape=(self.batch_size, self.m))

    X = X + self.aSDE(0.0, X, pi) * self.dt \
          + tf.squeeze(tf.matmul(self.bSDE(0.0, X, pi),
          dW[:, :, :, 0] * tf.sqrt(self.dt)), axis = -1)

    for i in range(self.N-1):

        pi = self.Modelpi[i+1](X, training = True)
        X = X + self.aSDE((i+1)*self.dt, X, pi) * self.dt \
                + tf.squeeze(tf.matmul(self.bSDE((i+1)*self.dt, X, pi),
                dW[:, :, :, i+1] * tf.sqrt(self.dt)), axis = -1)

    return X

@tf.function
def optimize3(self, dW):
    with tf.GradientTape(watch_accessed_variables = False) as tape:
        tape.watch(self.x0)

        X = self.simulate3(dW)
        loss3 = self.loss3(X)

    grad = tape.gradient(loss3, self.x0)
    self.optimizer3.apply_gradients([(grad, self.x0)])
    return

def train(self, steps):
    time_start = time.time()
    for k in range(steps):
        # dW does not have the desired shape, if BBdim==1
        dW = tf.constant(multivariate_normal.rvs(size=[self.batch_size, self.BBdim, self.N]),
                                dtype=tf.float32)
        if self.BBdim == 1:
          dW = tf.expand_dims(dW, axis = 1)
        dW = tf.expand_dims(dW, axis = 2)

        self.optimize1(dW)

        optimizer = self.optimizer2[0]
        self.optimize2_0(optimizer)

        X = self.x0 * tf.ones(shape=(self.batch_size, self.d))
        Y = self.y0 * tf.ones(shape=(self.batch_size, 1))
        Z = self.z0 * tf.ones(shape=(self.batch_size, self.d))

        pi = self.pi0 * tf.ones(shape=(self.batch_size, self.m))
        Q = self.Q0 * tf.ones(shape=(self.batch_size, self.d, self.d))

        for i in range(self.N-1):
          X, Y, Z, pi, Q = self.optimizecontrol[i](dW, X, Y, Z, pi, Q)

        self.optimize3(dW)

        self.history_time.append(time.time()-time_start)
        self.history_value.append(self.y0.numpy() + self.x0.numpy()*self.x0_primal)

        if k%50 == 0:
          print("Step: %d, Time: %.2f, Value: %.4f"
                % (k, self.history_time[-1], self.history_value[-1]))
    return


class PartNetwork(keras.Model):

def __init__(self, m, d, layers_num, nodes, isQ, **kwargs):
    super(PartNetwork, self).__init__(**kwargs)
    self.d = d
    self.m = m
    self.layers_num = layers_num
    self.nodes = nodes
    self.isQ = isQ
    if self.isQ == False:
        self.outdim = self.m
        # if unconstrained: self.projection = keras.layers.Lambda(lambda x: x*0)
    else:
        self.outdim = (self.d)**2

    self.bnorm_layers = [keras.layers.BatchNormalization(epsilon=100)
                         for _ in range(self.layers_num-1)]
    self.dense_layers = [keras.layers.Dense(nodes[i], use_bias=False, activation=None)
                         for i in range(self.layers_num-2)]
    self.dense_layers.append(keras.layers.Dense(self.outdim, activation=None))
    if self.isQ == True:
        self.reshape_layer = keras.layers.Reshape((self.d, self.d))

def call(self, x, training):
    x = self.bnorm_layers[0](x, training)
    for i in range(self.layers_num-2):
        x = self.dense_layers[i](x)
        x = self.bnorm_layers[i+1](x, training)
        x = tf.nn.relu(x)
    x = self.dense_layers[self.layers_num-2](x)
    if self.isQ == True:
        x = self.reshape_layer(x)
    if self.isQ == False:
        x = x*x
    return x


Model = Deepcontrolled2BSDE_dual()
Model.build()
Model.optimize2prepare()
Model.train(10000)
```

Code A.2: Python code for the dual version of the deep controlled 2BSDE algorithm in the setting of Example 5.5.

A.3 Python Implementation of the Deep SMP Algorithm

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import time
from scipy.stats import multivariate_normal

class DeepSMP(keras.Model):

    def __init__(self, **kwargs):
        super(DeepSMP, self).__init__(**kwargs)
        self.m = 30
        self.T = 0.5
        self.N = 10
        self.dt = self.T/self.N
        self.layers_num = 4  # number of hidden + 2
        self.nodes = [11, 11]  # nodes in hidden layers
        self.batch_size = 64
        self.x0_primal = 10
        self.y0 = tf.Variable(np.random.uniform(low = 0.2, high = 0.4), trainable = True)
        self.schedule1 = keras.optimizers.schedules.PiecewiseConstantDecay([2000, 5000, 8000],
                                                    [1e-2, 1e-3, 1e-4, 1e-5])
        self.schedule2 = [keras.optimizers.schedules.PiecewiseConstantDecay([2000, 5000, 8000],
                                                    [1e-2, 1e-3, 1e-4, 1e-5]) for _ in range(self.N)]
        self.schedule3 = keras.optimizers.schedules.PiecewiseConstantDecay([200, 1000, 8000],
                                                    [1e-2, 1e-4, 1e-5, 1e-6])
        self.optimizer1 = keras.optimizers.Adam(learning_rate = self.schedule1)
        self.optimizer2 = [keras.optimizers.Adam(learning_rate = self.schedule2[i])
                           for i in range(self.N)]
        self.optimizer3 = keras.optimizers.Adam(learning_rate = self.schedule3)
        self.p0 = self.x0_primal  # Condition 2 from SMP theorem
        self.pi0 = tf.Variable(np.random.uniform(size = (1, self.m), low = 0.1, high = 0.4),
                               trainable=True, dtype=tf.float32, constraint=lambda z: z*z)
        self.Q0 = tf.Variable(np.random.uniform(size = (1, self.m), low = −0.1, high = 0.1),
                              trainable=True, dtype=tf.float32)
        self.ModelQ = [self.Q0] + [PartNetwork(self.m, self.layers_num, self.nodes,
                                               isQ=True) for _ in range(self.N − 1)]
        self.Modelpi = [self.pi0] + [PartNetwork(self.m, self.layers_num, self.nodes,
                                                 isQ=False) for _ in range(self.N − 1)]
        self.history_time = []
        self.history_y0 = []
        self.mc_size = 100000
        self.history_bound_u = []
        self.history_bound_l = []

    def build( self ):
        for i in range(self.N−1):
            self.Modelpi[i+1](tf.zeros(shape=(1, 1)), training = False)
            self.ModelQ[i+1](tf.zeros(shape=(1, 1)), training = False)
        return

    def delta_K( self , v):
        kappa = 1/self.m
        return kappa∗tf.reduce_sum(v, axis = 1, keepdims = True)

    def projection_K( self , x):
        return tf.where(x<−1/self.m, −1/self.m, x)

    def sigma(self, t):
        sigma = (0.3∗(1+tf.sqrt(t))−0.1)∗tf.eye(self.m) + 0.1∗tf.ones(shape=(self.m, self.m))
        return sigma

    def sigma_inv( self , t):
        sigma_inv = tf.linalg.inv(self.sigma(t))
        return sigma_inv

    def mu(self, t, size):
        helper = tf.expand_dims(tf.range(1, limit=self.m+1, dtype = tf.float32), axis=0)
        mu = 0.07+0.02∗tf.sin(4∗np.pi∗t+2∗np.pi∗helper/self.m)
        return mu∗tf.ones(shape=(size, self.m))

    def r( self , t, size):
        r = 0.06∗tf.exp(0.5∗t)
        return r∗tf.ones(shape=(size, 1))

    def theta( self , t, size):
        theta = tf.transpose(tf.matmul(self.sigma_inv(t), self.mu(t, size)−self.r(t, size),
                                       transpose_b = True))
        return theta

    def U(self, x):
        x = tf.expand_dims(x, axis = −1)
        return tf.where(x>0, tf.math.log(tf.where(x>0, x, 1)), 0)

    def g( self , x):
        x = tf.expand_dims(x, axis = −1)
        return tf.where(x>0, −1−tf.math.log(tf.where(x>0, x, 1)), 0)

    def gx(self, x):
        x = tf.expand_dims(x, axis = -1)
        return tf.where(x>0, -tf.pow(tf.where(x>0, x, 1), -1), 0)

    def loss1(self, y, p):
        return tf.reduce_mean(tf.square(tf.expand_dims(p, axis = -1) + self.gx(y)))

    def loss2(self, t, p, v, q):
        return (tf.reduce_mean(tf.square(tf.expand_dims(p, axis = -1)*self.delta_K(v)
                + tf.reduce_sum(q*tf.transpose(tf.matmul(self.sigma_inv(t), v,
                transpose_b = True)), axis = 1, keepdims = True))))

    def loss3(self, y):
        return tf.reduce_mean(self.g(y)) + self.y0*self.x0_primal

    def bounds(self):
        dW = tf.constant(multivariate_normal.rvs(size=[self.mc_size, self.m, self.N]),
                         dtype=tf.float32)
        if self.m == 1:
            dW = tf.expand_dims(dW, axis = 1)
        Y, P = self.simulate1(dW, size = self.mc_size, training = False)

        bound_l = tf.reduce_mean(self.U(P))
        bound_u = tf.reduce_mean(self.g(Y)) + self.y0*self.x0_primal

        return [bound_l, bound_u]

    @ tf.autograph.experimental.do_not_convert
    def simulate1(self, dW, size, training):

        Y = self.y0 * tf.ones(size)
        P = self.p0 * tf.ones(size)

        pi = self.pi0 * tf.ones(shape=(size, self.m))
        Q = self.Q0 * tf.ones(shape=(size, self.m))

        P = P + tf.squeeze((self.r(0.0, size) * tf.expand_dims(P, axis = -1) + tf.reduce_sum(
                Q * self.theta(0.0, size), axis = 1, keepdims = True)) * self.dt, axis = 1) \
                + tf.reduce_sum(dW[:, :, 0] * Q * tf.sqrt(self.dt), axis = 1)
        Y = Y - tf.squeeze(tf.expand_dims(Y, axis = -1) * (self.r(0.0, size) + self.delta_K(pi))
                * self.dt, axis = 1) \
                - tf.reduce_sum(dW[:, :, 0] * tf.expand_dims(Y, axis = -1) * (self.theta(0.0, size)
                + tf.transpose(tf.matmul(self.sigma_inv(0.0), pi, transpose_b = True)))
                * tf.sqrt(self.dt), axis = 1)

        for i in range(self.N-1):

            pi = self.Modelpi[i+1](tf.expand_dims(Y, axis = -1), training)
            Q = tf.expand_dims(P, axis = -1) * tf.matmul(self.projection_K(self.ModelQ[i+1](
                 tf.expand_dims(Y, axis = -1), training)), self.sigma((i+1)*self.dt))

            P = P + tf.squeeze((self.r((i+1)*self.dt, size) * tf.expand_dims(P, axis = -1)
                    + tf.reduce_sum(Q * self.theta((i+1)*self.dt, size), axis = 1, keepdims = True))
                    * self.dt, axis = 1) \
                    + tf.reduce_sum(dW[:, :, i+1] * Q * tf.sqrt(self.dt), axis = 1)
            Y = Y − tf.squeeze(tf.expand_dims(Y, axis = −1) ∗ (self.r((i+1)∗self.dt,  size)
                    + self.delta_K(pi)) ∗ self.dt,  axis  = 1) \
                    − tf.reduce_sum(dW[:, :, i+1] ∗ tf.expand_dims(Y, axis = −1)
                    ∗ (self.theta((i+1)∗self.dt, size)  + tf.transpose(tf.matmul(self.sigma_inv(
                    (i+1)∗self.dt), pi,  transpose_b = True))) ∗ tf.sqrt(self.dt),  axis  = 1)

        return Y, P

    @ tf.function
    def optimize1(self,  dW):
        with tf.GradientTape(watch_accessed_variables = False) as tape:
            tape.watch(self.ModelQ.trainable_variables)

            Y, P = self.simulate1(dW, size = self.batch_size,  training  = True)
            loss1  = self.loss1(Y, P)

        grad1 = tape.gradient(loss1,  self.ModelQ.trainable_variables)
        self.optimizer1.apply_gradients(zip(grad1, self.ModelQ.trainable_variables))
        return

    @ tf.function
    def optimize2_0(self,  optimizer):
        with tf.GradientTape(watch_accessed_variables = False) as tape:
            tape.watch(self.pi0)

            # Y = self.y0 ∗ tf.ones(self.batch_size)
            P = self.p0 ∗ tf.ones(self.batch_size)

            pi  = self.pi0  ∗  tf.ones(shape=(self.batch_size,  self.m))
            Q = self.Q0 ∗ tf.ones(shape=(self.batch_size,  self.m))

            loss2  = self.loss2(0.0,  P,  pi,  Q)

        grad = tape.gradient(loss2,  self.pi0)
        optimizer.apply_gradients([(grad,  self.pi0)])
        return

    @ tf.autograph.experimental.do_not_convert
    def simulate2(self,  dW, N, Y, P, pi, Q):

        P = P + tf.squeeze((self.r((N−1)∗self.dt, self.batch_size)∗tf.expand_dims(P, axis = −1)
                + tf.reduce_sum(Q∗self.theta((N−1)∗self.dt, self.batch_size),  axis  = 1,
                keepdims = True)) ∗ self.dt, axis = 1) \
                + tf.reduce_sum(dW[:, :, N−1] ∗ Q ∗ tf.sqrt(self.dt),  axis  = 1)
        Y = Y − tf.squeeze(tf.expand_dims(Y, axis = −1) ∗ (self.r((N−1)∗self.dt, self.batch_size)
                + self.delta_K(pi)) ∗ self.dt,  axis  = 1) \
                − tf.reduce_sum(dW[:, :, N−1] ∗ tf.expand_dims(Y, axis = −1)
                ∗ (self.theta((N−1)∗self.dt, self.batch_size)  + tf.transpose(tf.matmul(self.sigma_inv(
                (N−1)∗self.dt),  pi,  transpose_b = True))) ∗ tf.sqrt(self.dt),  axis  = 1)

        pi  = self.Modelpi[N](tf.expand_dims(Y, axis = −1), training = True)
        Q = tf.expand_dims(P, axis = −1) ∗ tf.matmul(self.projection_K(self.ModelQ[N](
              tf.expand_dims(Y, axis = −1), training = True)), self.sigma(N∗self.dt))

        return Y, P, pi,  Q

    def optimize2(self, dW, N, Y, P, pi, Q, optimizer, Model):
        with tf.GradientTape(watch_accessed_variables = False) as tape:
            tape.watch(Model.trainable_variables)

            Y, P, pi, Q = self.simulate2(dW, N, Y, P, pi, Q)
            loss = self.loss2(N*self.dt, P, pi, Q)

        grad = tape.gradient(loss, Model.trainable_variables)
        optimizer.apply_gradients(zip(grad, Model.trainable_variables))
        return Y, P, pi, Q

    def optimize2prepare(self):
        self.optimizecontrol = [tf.function(self.optimize2).get_concrete_function(
            tf.TensorSpec(shape=[self.batch_size, self.m, self.N], dtype=tf.float32), i+1,
            tf.TensorSpec(shape=[self.batch_size], dtype=tf.float32),
            tf.TensorSpec(shape=[self.batch_size], dtype=tf.float32),
            tf.TensorSpec(shape=[self.batch_size, self.m], dtype=tf.float32),
            tf.TensorSpec(shape=[self.batch_size, self.m], dtype=tf.float32),
           self.optimizer2[i+1], self.Modelpi[i+1]) for i in range(self.N-1)]
        return

    @ tf.autograph.experimental.do_not_convert
    def simulate3(self, dW):

        Y = self.y0 * tf.ones(self.batch_size)
        pi = self.pi0 * tf.ones(shape=(self.batch_size, self.m))

        Y = Y - tf.squeeze(tf.expand_dims(Y, axis = -1) * (self.r(0.0, self.batch_size)
                + self.delta_K(pi)) * self.dt, axis = 1) \
                - tf.reduce_sum(dW[:, :, 0] * tf.expand_dims(Y, axis = -1) * (self.theta(0.0,
                 self.batch_size) +  tf.transpose(tf.matmul(self.sigma_inv(0.0), pi,
                 transpose_b = True))) * tf.sqrt(self.dt), axis = 1)

        for i in range(self.N-1):

            pi = self.Modelpi[i+1](tf.expand_dims(Y, axis = -1), training = True)
            Y = Y - tf.squeeze(tf.expand_dims(Y, axis = -1) * (self.r((i+1)*self.dt, self.batch_size)
                    + self.delta_K(pi)) * self.dt, axis = 1) \
                    - tf.reduce_sum(dW[:, :, i+1] * tf.expand_dims(Y, axis = -1)
                    * (self.theta((i+1)*self.dt, self.batch_size) + tf.transpose(tf.matmul(
                    self.sigma_inv((i+1)*self.dt), pi, transpose_b = True)))
                    * tf.sqrt(self.dt), axis = 1)

        return Y

    @ tf.function
    def optimize3(self, dW):
        with tf.GradientTape(watch_accessed_variables = False) as tape:
            tape.watch(self.y0)

            Y = self.simulate3(dW)
            loss3 = self.loss3(Y)

        grad = tape.gradient(loss3, self.y0)
        self.optimizer3.apply_gradients([(grad, self.y0)])
        return

    def train(self, steps):
        time_start = time.time()
        for k in range(steps):
            # dW does not have the desired shape, if BBdim==1
            dW = tf.constant(multivariate_normal.rvs(size=[self.batch_size, self.m, self.N]),
                             dtype=tf.float32)
            if self.m == 1:
                dW = tf.expand_dims(dW, axis = 1)

            self.optimize1(dW)

            optimizer = self.optimizer2[0]
            self.optimize2_0(optimizer)

            Y = self.y0 * tf.ones(self.batch_size)
            P = self.p0 * tf.ones(self.batch_size)

            pi = self.pi0 * tf.ones(shape=(self.batch_size, self.m))
            Q = self.Q0 * tf.ones(shape=(self.batch_size, self.m))

            for i in range(self.N-1):
                Y, P, pi, Q = self.optimizecontrol[i](dW, Y, P, pi, Q)

            self.optimize3(dW)

            self.history_time.append(time.time()-time_start)
            self.history_y0.append(self.y0.numpy())

            if k%50 == 0:
                print("Step: %d, Time: %.2f, y: %.4f"
                      % (k, self.history_time[-1], self.history_y0[-1]))

            if k%200 == 0:
                helper = self.bounds()
                self.history_bound_l.append(helper[0].numpy())
                self.history_bound_u.append(helper[1].numpy())
                print("Step: %d, Bound_l: %.4f, Bound_u: %.4f"
                      % (k, self.history_bound_l[-1], self.history_bound_u[-1]))
        return


class PartNetwork(keras.Model):

    def __init__(self, m, layers_num, nodes, isQ, **kwargs):
        super(PartNetwork, self).__init__(**kwargs)
        self.d = 1
        self.m = m
        self.layers_num = layers_num
        self.nodes = nodes
        self.isQ = isQ
        if self.isQ == False:
            self.outdim = self.m
            # if unconstrained: self.projection = keras.layers.Lambda(lambda x: x*0)
        else:
            self.outdim = self.m

        self.bnorm_layers = [keras.layers.BatchNormalization(epsilon=100)
                             for _ in range(self.layers_num-1)]
        self.dense_layers = [keras.layers.Dense(nodes[i], use_bias=False, activation=None)
                             for i in range(self.layers_num-2)]
        self.dense_layers.append(keras.layers.Dense(self.outdim, activation=None))

    def call(self, x, training):
        x = self.bnorm_layers[0](x, training)
        for i in range(self.layers_num-2):
            x = self.dense_layers[i](x)
            x = self.bnorm_layers[i+1](x, training)
            x = tf.nn.relu(x)
        x = self.dense_layers[self.layers_num-2](x)
        if self.isQ == False:
            x = x*x
        return x


Model = DeepSMP()
Model.build()
Model.optimize2prepare()
Model.train(10000)
```

Code A.3: Python code for the deep SMP algorithm in the setting of Example 5.5.

A.4 Python Implementation of the Deep Primal SMP Algorithm

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import time
from scipy.stats import multivariate_normal

class DeepSMP_primal(keras.Model):

    def __init__(self, **kwargs):
        super(DeepSMP_primal, self).__init__(**kwargs)
        self.m = 30
        self.T = 0.5
        self.N = 10
        self.dt = self.T/self.N
        self.layers_num = 4 # number of hidden + 2
        self.nodes = [11, 11]  # nodes in hidden layers
        self.batch_size = 64
        self.x0 = 10
        self.schedule1 = keras.optimizers.schedules.PiecewiseConstantDecay([1000, 3000, 8000],
                                                                            [1e-2, 1e-3, 1e-4, 1e-5])
        self.schedule2 = [keras.optimizers.schedules.PiecewiseConstantDecay([1000, 3000, 8000],
                                                                            [1e-3, 1e-4, 1e-5, 1e-6]) for _ in range(self.N)]
        self.optimizer1 = keras.optimizers.Adam(learning_rate = self.schedule1)
        self.optimizer2 = [keras.optimizers.Adam(learning_rate = self.schedule2[i])
                           for i in range(self.N)]
        self.p0 = tf.Variable(np.random.uniform(low=-0.4, high=-0.2), trainable=True)
        self.pi0 = tf.Variable(np.random.uniform(size=(1, self.m), low=0, high=0.2),
                               trainable=True, dtype=tf.float32,
                               constraint=lambda x: tf.where(x<-1/self.m, -1/self.m, x))
        self.Q0 = tf.Variable(np.random.uniform(size=(1, self.m), low=-0.1, high=0.1),
                              trainable=True, dtype=tf.float32)
        self.ModelQ = [self.Q0] + [PartNetwork(self.m, self.layers_num, self.nodes, isQ=True)
                                   for _ in range(self.N-1)]
        self.varforloss1 = [self.p0] + self.ModelQ
        self.Modelpi = [self.pi0] + [PartNetwork(self.m, self.layers_num, self.nodes, isQ=False)
                                     for _ in range(self.N-1)]
        self.history_time = []
        self.history_p0 = []
        self.mc_size = 100000
        self.history_bound_u = []
        self.history_bound_l = []

    def build(self):
        for i in range(self.N-1):
            self.Modelpi[i+1](tf.zeros(shape=(1, 1)), training = False)
            self.ModelQ[i+1](tf.zeros(shape=(1, 1)), training = False)
        return

    def sigma(self, t):
        sigma = (0.3*(1+tf.sqrt(t))-0.1)*tf.eye(self.m) + 0.1*tf.ones(shape=(self.m, self.m))
        return sigma

    def sigma_inv(self, t):
        sigma_inv = tf.linalg.inv(self.sigma(t))
        return sigma_inv

    def mu(self, t, size):
        helper = tf.expand_dims(tf.range(1, limit=self.m+1, dtype = tf.float32), axis=0)
        mu = 0.07 + 0.02*tf.sin(4*np.pi*t+2*np.pi*helper/self.m)
        return mu*tf.ones(shape=(size, self.m))

    def r(self, t, size):
        r = 0.06*tf.exp(0.5*t)
        return r*tf.ones(shape=(size, 1))

    def theta(self, t, size):
        theta = tf.transpose(tf.matmul(self.sigma_inv(t), self.mu(t, size)-self.r(t, size),
                                       transpose_b = True))
        return theta

    def U_transform(self, x):
        x = tf.expand_dims(x, axis = -1)
        return tf.where(x>0, -1-tf.math.log(tf.where(x>0, x, 1)), 0)

    def g(self, x):
        x = tf.expand_dims(x, axis = -1)
        return tf.where(x>0, tf.math.log(tf.where(x>0, x, 1)), 0)
```

```python

    def gx(self, x):
        x = tf.expand_dims(x, axis = -1)
        return tf.where(x>0, tf.pow(tf.where(x>0, x, 1), -1), 0)

    def loss1(self, x, p):
        return tf.reduce_mean(tf.square(tf.expand_dims(p, axis = -1) + self.gx(x)))

    def loss2(self, t, p, pi, q, size):
        return (tf.reduce_mean(tf.reduce_sum(pi * tf.transpose(tf.matmul(self.sigma(t),
                tf.expand_dims(p, axis = -1) * self.theta(t, size) + q, transpose_b = True)),
                axis = 1, keepdims = True)))

    def bounds(self):
        dW = tf.constant(multivariate_normal.rvs(size=[self.mc_size, self.m, self.N]),
                         dtype=tf.float32)
        if self.m == 1:
            dW = tf.expand_dims(dW, axis = 1)
        X, P = self.simulate1(dW, size = self.mc_size, training = False)

        bound_l = tf.reduce_mean(self.g(X))
        bound_u = tf.reduce_mean(self.U_transform(-P)) - self.p0*self.x0

        return [bound_l, bound_u]

    @ tf.autograph.experimental.do_not_convert
    def simulate1(self, dW, size, training):

        X = self.x0 * tf.ones(size)
        P = self.p0 * tf.ones(size)

        pi = self.pi0 * tf.ones(shape=(size, self.m))
        Q = self.Q0 * tf.ones(shape=(size, self.m))

        P = P - tf.squeeze(((self.r(0.0, size) + tf.reduce_sum(tf.matmul(pi, self.sigma(0.0))
                * self.theta(0.0, size), axis = 1, keepdims = True)) * tf.expand_dims(P, axis = -1)
                + tf.reduce_sum(Q * tf.matmul(pi, self.sigma(0.0)), axis = 1, keepdims = True))
                * self.dt, axis = 1) \
                + tf.reduce_sum(dW[:, :, 0] * Q * tf.sqrt(self.dt), axis = 1)
        X = X + tf.squeeze(tf.expand_dims(X, axis = -1) * (self.r(0.0, size) + tf.reduce_sum(
                tf.matmul(pi, self.sigma(0.0)) * self.theta(0.0, size), axis = 1, keepdims = True))
                * self.dt, axis = 1) \
                + tf.reduce_sum(dW[:, :, 0] * tf.expand_dims(X, axis = -1)
                * tf.matmul(pi, self.sigma(0.0)) * tf.sqrt(self.dt), axis = 1)

        for i in range(self.N-1):

            pi = self.Modelpi[i+1](tf.expand_dims(X, axis = -1), training)
            Q = self.ModelQ[i+1](tf.expand_dims(X, axis = -1), training)

            P = P - tf.squeeze(((self.r(self.dt*(i+1), size) + tf.reduce_sum(tf.matmul(pi,
                    self.sigma(self.dt*(i+1))) * self.theta(self.dt*(i+1), size), axis = 1,
                    keepdims = True)) * tf.expand_dims(P, axis = -1)
                    + tf.reduce_sum(Q * tf.matmul(pi, self.sigma(self.dt*(i+1))), axis = 1,
                    keepdims = True)) * self.dt, axis = 1) \
                    + tf.reduce_sum(dW[:, :, i+1] * Q * tf.sqrt(self.dt), axis = 1)
            X = X + tf.squeeze(tf.expand_dims(X, axis = -1) * (self.r(self.dt*(i+1), size)
                    + tf.reduce_sum(tf.matmul(pi, self.sigma(self.dt*(i+1))) * self.theta(
                    self.dt*(i+1), size), axis = 1, keepdims = True)) * self.dt, axis = 1) \
                    + tf.reduce_sum(dW[:, :, i+1] * tf.expand_dims(X, axis = -1)
                    * tf.matmul(pi, self.sigma(self.dt*(i+1))) * tf.sqrt(self.dt), axis = 1)

        return X, P

    @ tf.function
    def optimize1(self, dW):
        with tf.GradientTape(watch_accessed_variables = False) as tape:
            tape.watch(self.varforloss1.trainable_variables)

            X, P = self.simulate1(dW, size = self.batch_size, training = True)
            loss1 = self.loss1(X, P)

        grad1 = tape.gradient(loss1, self.varforloss1.trainable_variables)
        self.optimizer1.apply_gradients(zip(grad1, self.varforloss1.trainable_variables))
        return

    @ tf.function
    def optimize2_0(self, optimizer):
        with tf.GradientTape(watch_accessed_variables = False) as tape:
            tape.watch(self.pi0)

            # X = self.x0 * tf.ones(self.batch_size)
            P = self.p0 * tf.ones(self.batch_size)

            pi = self.pi0 * tf.ones(shape=(self.batch_size, self.m))
            Q = self.Q0 * tf.ones(shape=(self.batch_size, self.m))

            loss2 = self.loss2(0.0, P, pi, Q, size = self.batch_size)

        grad = tape.gradient(loss2, self.pi0)
        optimizer.apply_gradients([(grad, self.pi0)])
        return

    @ tf.autograph.experimental.do_not_convert
    def simulate2(self, dW, N, X, P, pi, Q):

        P = P - tf.squeeze(((self.r(self.dt*(N-1), self.batch_size) + tf.reduce_sum(tf.matmul(pi,
                self.sigma(self.dt*(N-1))) * self.theta(self.dt*(N-1), self.batch_size), axis = 1,
                keepdims = True)) * tf.expand_dims(P, axis = -1)
                + tf.reduce_sum(Q * tf.matmul(pi, self.sigma(self.dt*(N-1))), axis = 1,
                keepdims = True)) * self.dt, axis = 1) \
                + tf.reduce_sum(dW[:, :, N-1] * Q * tf.sqrt(self.dt), axis = 1)
        X = X + tf.squeeze(tf.expand_dims(X, axis = -1) * (self.r(self.dt*(N-1), self.batch_size)
                + tf.reduce_sum(tf.matmul(pi, self.sigma(self.dt*(N-1))) * self.theta(
                self.dt*(N-1), self.batch_size), axis = 1, keepdims = True)) * self.dt, axis = 1) \
                + tf.reduce_sum(dW[:, :, N-1] * tf.expand_dims(X, axis = -1)
                * tf.matmul(pi, self.sigma(self.dt*(N-1))) * tf.sqrt(self.dt), axis = 1)

        pi = self.Modelpi[N](tf.expand_dims(X, axis = -1), training = True)
        Q = self.ModelQ[N](tf.expand_dims(X, axis = -1), training = True)

        return X, P, pi, Q

    def optimize2(self, dW, N, X, P, pi, Q, optimizer, Model):
        with tf.GradientTape(watch_accessed_variables = False) as tape:
            tape.watch(Model.trainable_variables)

            X, P, pi, Q = self.simulate2(dW, N, X, P, pi, Q)
            loss = self.loss2(N*self.dt, P, pi, Q, size = self.batch_size)

        grad = tape.gradient(loss, Model.trainable_variables)
        optimizer.apply_gradients(zip(grad, Model.trainable_variables))
        return X, P, pi, Q

    def optimize2prepare(self):
        self.optimizecontrol = [tf.function(self.optimize2).get_concrete_function(
            tf.TensorSpec(shape=[self.batch_size, self.m, self.N], dtype=tf.float32), i+1,
            tf.TensorSpec(shape=[self.batch_size], dtype=tf.float32),
            tf.TensorSpec(shape=[self.batch_size], dtype=tf.float32),
            tf.TensorSpec(shape=[self.batch_size, self.m], dtype=tf.float32),
            tf.TensorSpec(shape=[self.batch_size, self.m], dtype=tf.float32),
            self.optimizer2[i+1], self.Modelpi[i+1]) for i in range(self.N-1)]
        return

    def train(self, steps):
        time_start = time.time()
        for k in range(steps):
            # dW does not have the desired shape, if BBdim==1
            dW = tf.constant(multivariate_normal.rvs(size=[self.batch_size, self.m, self.N]),
                             dtype=tf.float32)
            if self.m == 1:
                dW = tf.expand_dims(dW, axis = 1)

            self.optimize1(dW)

            optimizer = self.optimizer2[0]
            self.optimize2_0(optimizer)

            X = self.x0 * tf.ones(self.batch_size)
            P = self.p0 * tf.ones(self.batch_size)

            pi = self.pi0 * tf.ones(shape=(self.batch_size, self.m))
            Q = self.Q0 * tf.ones(shape=(self.batch_size, self.m))

            for i in range(self.N-1):
                X, P, pi, Q = self.optimizecontrol[i](dW, X, P, pi, Q)

            self.history_time.append(time.time()-time_start)
            self.history_p0.append(self.p0.numpy())

            if k%50 == 0:
                    print("Step: %d, Time: %.2f, p_0: %.4f"
                          % (k, self.history_time[-1], self.history_p0[-1]))

            if k%200 == 0:
```

```python
                helper = self.bounds()
                self.history_bound_l.append(helper[0].numpy())
                self.history_bound_u.append(helper[1].numpy())
                print("Step: %d, Bound_l: %.4f, Bound_u: %.4f"
                      % (k, self.history_bound_l[-1], self.history_bound_u[-1]))
        return


class PartNetwork(keras.Model):

    def __init__(self, m, layers_num, nodes, isQ, **kwargs):
        super(PartNetwork, self).__init__(**kwargs)
        self.d = 1
        self.m = m
        self.layers_num = layers_num
        self.nodes = nodes
        self.isQ = isQ
        if self.isQ == False:
            self.outdim = self.m
            # One could also define projection here.
        else:
            self.outdim = self.m

        self.bnorm_layers = [keras.layers.BatchNormalization(epsilon=100)
                             for _ in range(self.layers_num-1)]
        self.dense_layers = [keras.layers.Dense(nodes[i], use_bias=False, activation=None)
                             for i in range(self.layers_num-2)]
        self.dense_layers.append(keras.layers.Dense(self.outdim, activation=None))

    def call(self, x, training):
        x = self.bnorm_layers[0](x, training)
        for i in range(self.layers_num-2):
            x = self.dense_layers[i](x)
            x = self.bnorm_layers[i+1](x, training)
            x = tf.nn.relu(x)
        x = self.dense_layers[self.layers_num-2](x)
        if self.isQ == False:
            x = x*x-1/self.m
        return x


Model = DeepSMP_primal()
Model.build()
Model.optimize2prepare()
Model.train(10000)
```

Code A.4: Python code for the deep primal SMP algorithm in the setting of Example 5.5.

Bibliography

[1] Christian Beck, Weinan E, and Arnulf Jentzen. Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. *Journal of Nonlinear Science*, 29:1563–1619, 2019.

[2] Dimitri P. Bertsekas and Steven E. Shreve. *Stochastic Optimal Control: The Discrete–Time Case*. Academic Press, 1978.

[3] Abel Cadenillas and Ioannis Karatzas. The Stochastic Maximum Principle for Linear, Convex Optimal Control with Random Coefficients. *SIAM J. Control Optim.*, 33(2):590–624, 1995.

[4] Roy Cerqueti. Dynamic Programming via Measurable Selection. *Pacific Journal of Optimization*, 5(1):169–181, 2009.

[5] Ashley Davey and Harry Zheng. Deep Learning for Constrained Utility Maximisation. *arXiv:2008.11757*, 2020.

[6] Weinan E, Jiequn Han, and Arnulf Jentzen. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380, 2017.

[7] Wendell H. Fleming and Halil Mete Soner. *Controlled Markov Processes and Viscosity Solutions*. Springer, second edition, 2006.

[8] Ulrich Horst, Ying Hu, Peter Imkeller, Anthony Reveillac, and Jianing Zhang. Forward-backward systems for expected utility maximization. *Stochastic Processes and their Applications*, 124(5):1813–1848, 2014.

[9] Ioannis Karatzas, John P. Lehoczky, Steven E. Shreve, and Gan-Lin Xu. Martingale and Duality Methods for Utility Maximization in an Incomplete Market. *SIAM J. Control Optim.*, 29(3):702–730, 1991.

[10] Ioannis Karatzas and Steven E. Shreve. *Brownian Motion and Stochastic Calculus*. Springer, second edition, 1991.

[11] Ioannis Karatzas and Steven E. Shreve. *Methods of Mathematical Finance*. Springer, 1998.

[12] Dmitry Kramkov and Walter Schachermayer. The Asymptotic Elasticity of Utility Functions and Optimal Investment in Incomplete Markets. *Ann. Appl. Probab.*, 9(3):904–950, 1999.

[13] Nicolai V. Krylov. *Controlled Diffusion Processes*. Springer, 1980.

[14] Nicolai V. Krylov. *Nonlinear Elliptic and Parabolic Equations of the Second Order*. D. Reidel Publishing Company, 1987.

[15] Chantal Labbé and Andrew J. Heunis. Conjugate duality in problems of constrained utility maximization. *Stochastics*, 81(6):545–565, 2009.

[16] Yusong Li and Harry Zheng. Dynamic Convex Duality in Constrained Utility Maximization. *Stochastics*, 90(8):1145–1169, 2018.

[17] Jingtang Ma, Wenyuan Li, and Harry Zheng. Dual control Monte Carlo method for tight bounds of value function under Heston stochastic volatility model. *arXiv:1710.10487*, 2017.

[18] Bernt Øksendal. *Stochastic Differential Equations: An Introduction with Applications*. Springer, fifth edition, 1998.

[19] Huyên Pham. *Continuous-time Stochastic Control and Optimization with Financial Applications*. Springer, 2009.

[20] Tyrrell R. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

[21] Marina Santacroce and Barbara Trivellato. Forward Backward Semimartingale Systems for Utility Maximization. *SIAM J. Control Optim.*, 52(6):3517–3537, 2014.

[22] Jiongmin Yong and Xun Yu Zhou. *Stochastic Controls: Hamiltonian Systems and HJB Equations*. Springer, 1999.