QLBS: Q-Learner in the Black-Scholes(-Merton) Worlds

Igor Halperin

NYU Tandon School of Engineering

e-mail: *igor.halperin@nyu.edu*

December 16, 2017

Abstract:

This paper presents a discrete-time option pricing model that is rooted in Reinforcement Learning (RL), and more specifically in the famous Q-Learning method of RL. We construct a risk-adjusted Markov Decision Process for a discrete-time version of the classical Black-Scholes-Merton (BSM) model, where the option price is an optimal Q-function, while the optimal hedge is a second argument of this optimal Q-function, so that both the price and hedge are parts of the *same* formula. Pricing is done by *learning to dynamically optimize risk-adjusted returns* for an option replicating portfolio, as in the Markowitz portfolio theory. Using Q-Learning and related methods, once created in a parametric setting, the model is able to go *model-free* and learn to price and hedge an option *directly from data* generated from a dynamic replicating portfolio which is rebalanced at discrete times. If the world is according to BSM, our risk-averse Q-Learner converges, given enough training data, to the true BSM price and hedge ratio of the option in the continuous time limit $\Delta t \to 0$, even if hedges applied at the stage of data generation are completely random (i.e. it can learn the BSM model itself, too!), because Q-Learning is an *off-policy* algorithm. If the world is *different* from a BSM world, the Q-Learner will find it out as well, because Q-Learning is a *model-free* algorithm. For finite time steps Δt, the Q-Learner is able to efficiently calculate both the optimal hedge and optimal price for the option *directly from trading data, and without an explicit model of the world*. This suggests that RL may provide efficient *data-driven and model-free* methods for optimal pricing and hedging of options, once we depart from the academic continuous-time limit $\Delta t \to 0$, and vice versa, option pricing methods developed in Mathematical Finance may be viewed as special cases of model-based Reinforcement Learning. Further, due to simplicity and tractability of our model which only needs basic linear algebra (plus Monte Carlo simulation, if we work with synthetic data), and its close relation to the original BSM model, we suggest that our model could be used for benchmarking of different RL algorithms for financial trading applications.

I would like to thank my students for their interest in this work and stimulating discussions that challenged me to look for simple explanations of complex topics. I thank Tom N.L. for an initial implementation of a time-discretized BSM model. This work is dedicated to my wife Lola on the occasion of her birthday and receiving a doctoral degree.

1 Introduction

Reinforcement Learning (RL) is one of the most fascinating sub-fields of Machine Learning (ML) [1]. While being more than 50 years old, it seems to attract ever-growing interest of Machine Learning (ML) researchers. In particular, in the area of Finance, many tasks in trading or investment decisions seem well suited for RL approaches. At present, literature on the topic is sparse, and moreover scattered among different financial applications areas.

This paper suggests to try Reinforcement Learning (RL) in a well-understood financial setting that could be thought of as a financial analogue of the classical inverted pole problem, a veritable test case for generations of Reinforcement Learning models [1]. This environment and other physics-inspired simulated environments are available, in particular, in the popular RL research platform OpenAI Gym [2]. Likewise, a similar "simplest possible, but not a simpler one" setting could be used for financial applications of Reinforcement Learning as a testing laboratory for exploration and benchmarking of different RL algorithms for financial applications.

Here we propose such a testing environment for Reinforcement Learning in Finance. As we will show below, it is very flexible and extensible. In particular, it allows a researcher to benchmark both discrete-action and continuous-action Reinforcement Learning algorithms. On the side of Finance, it offers a look into the problems of hedging/trading and pricing in financial markets - all the main elements of many financial tasks, but in a controllable and well understood environment.

A model developed in this course may also be of an independent interest in the context of financial modeling research, as a data-driven derivatives pricing model rooted in RL, unlike previous derivatives pricing models rooted in continuous-time Stochastic Optimal Control (SOC) and Ito's calculus, or a discrete-time SOC. Last but not least, it can be extended to more practically interesting settings including early exercises, multiple factors, transaction costs, and so on.

A model presented here is based on the celebrated Black-Scholes-Merton (BSM) model, also known as the Black-Scholes (BS) model, a cornerstone of modern Quantitative Finance [3, 4]. In a nutshell, the BSM model established the fact that even though the option price can (and will) change in the future because it depends on a future stock price which is also unknown, a *unique* fair option price can be found by using the principle of one price for identical goods, alongside with the method of pricing by replication, for a special choice of dynamics of the stock price.

In the BSM model, the stock dynamics is chosen to follow the law of a Geometric Brownian motion (with a drift). It turns out that this choice leads to a closed-form expression for a *unique* option price for a European call or put option on a stock, given by the celebrated Black-Scholes formula. *Simultaneously*, the classical BSM model leads to a somewhat paradoxical conclusion that options are altogether completely *redundant*, as they can always be perfectly replicated by a simple portfolio made of a stock and a bond.

If this were indeed the case in the real life, i.e. options were totally redundant, nobody would ever trade them, except possibly for very bored traders. Yet, option trading is a multi-billion business where people make and lose money daily. Traders use options and other financial derivatives both as investment vehicles and as hedging instruments. Options are *not* redundant.

The reason that options are not redundant is that they carry a substantial *risk*, notwithstanding the proposition of the classical BSM model. Nobody in the market trades options at their BS prices, and differences between BS prices and traded prices reflect dealers' perception of actual risk embedded in options. Financial professionals are well aware of the fact that the classical BSM model completely eliminates any risk in options for the sake of analytical tractability, by making three strong assumptions/approximations that do not hold in practice, though they

do not have a universally acceptable formula that would replace the BSM price for any option by a different uniquely-defined value.

The assumptions of the classical BSM model that make it totally miss risk in options are market completeness, continuous re-hedging, and zero transactions costs, see e.g. Wilmott's book [5]. None of these three assumptions hold in practice, which is why all *practical* uses of the BSM model involve some modifications of either dynamics or hedging and pricing methods [5].

Here we propose to use a discrete-time version of the BSM model to study financial trading and pricing problems using RL. On the one hand, this is a well-understood extension of the BSM model, which brings back some realism of actual option trading by considering re-hedging at discrete times, as opposed to a continuous re-hedging obtained in the BS limit of infinitesimal time steps. On the other hand, keeping re-hedging frequency finite (similar to how a re-hedging is done in reality), allows one to focus on the key objective of option trading and pricing, which is *risk minimization by hedging in a sequential decision-making process*, which is exactly the problem studied by Reinforcement Learning. As we will argue below, once properly formulated, option pricing and hedging amounts to RL.

Various extensions of the BSM model to a discrete time setting are well studied in the literature. We take one such formulation, and re-formulate it as a risk-adjusted Markov Decision Process (MDP) problem, where we model an option seller as an agent that hedges its risk in the option by trading in the underlying stock at discrete times. This model is referred to as the QLBS model in this paper, in recognition of the fact that the famous Q-Learning method of Watkins [6, 7], one of the most important RL algorithms according to Satton and Barto [1], turns out to be exactly *the* algorithm that computes *both* the optimal price and optimal hedge in such time-discretized BS(M) model *directly from data*. This means that once originated within the BSM model, the QLBS approach can now just bypass any (not only the BSM!) modeling framework altogether, and become *model-free*, all due to the fact that Q-Learning is a *model-free* algorithm [1].

On the other hand, from the modeling perspective, what we show in this paper is that the classical BSM model [3, 4] can be obtained as a continuous-time limit $\Delta t \to 0$ of a multi-period version of the Markowitz portfolio theory [8], in a special setting when the world is lognormal, and the investment portfolio is an option replicating portfolio made of a stock and a bank cash account. As is known at least since the pioneering work of Föllmer and Schweizer [9], [10], the limit $\Delta t \to 0$ is a *wrong* limit to take.

We derive a Bellman optimality equation for an action-value function for the QLBS model, and then present its solution using a backward-recursion Value Iteration Dynamic Programming (DP) method, which can be used when model parameters are *known*. As we show below, this procedure produces both the optimal hedge and optimal option price of the QLBS model that converge to their respective BSM values in the strict continuous-time limit $\Delta t \to 0$, while providing easily calculable corrections to the BSM values for a finite time step case $\Delta t > 0$. In particular, it turns out that the (normally, a labor-intensive) policy optimization step of the DP backward recursion method is done *analytically* in the QLBS model, due to a particularly simple (quadratic) form of a proper objective function arising in this optimization problem.

On the other hand, when the dynamics of the QLBS model is *unknown*, we can try methods of *Reinforcement Learning* to learn the optimal hedge and price directly from *trading data* for a dynamic hedge portfolio. We can apply methods of value-based Reinforcement Learning [1] to *approximately* solve the Bellman optimality equation without any knowledge of model's dynamics, but relying solely on *sample data*. For the purpose of *benchmarking* of different RL models, as the correct solution for a known model dynamics is already *known* from a DP approach, this seems to offer an ideal theoretical laboratory for testing RL models for Finance,

where all the needed data can be just received on demand, and in any size needed, using Monte Carlo simulation.

On the other hand, the QLBS model can also be used with *real* data, given that there is enough such data. While a setting developed below involves only one factor (one stock price), the model is straightforward to extend to multiple factors, different option payoffs, etc. Such possible extensions, which would add computational complexity but leave the conceptual framework intact, are left here for a future work.

Note that our model can be formulated either as a continuous-state or a discrete-state model, the latter being a finite-state approximation for the former. While continuous-state financial models are more practically relevant, especially in a multi-factor case, discrete-state models are easier to understand or explain. Just to be on a safe side, we present both formulations. Our presentation generally assumes a continuous-state formulation, but we specifically point out simplifications, whenever appropriate, that arise if we choose a discrete-state approximation to the model.

For continuous-state models, RL has to rely on function approximations to compactly represent either a value function, or a policy function, or both [1]. One simple choice is to use *linear architectures* that express a value function or a policy function as a linear combination $\sum_k \theta_k \phi_k(x)$ of some fixed basis functions $\{\phi_k(x)\}_{k=1}^K$. This replaces a general function approximation problem in a non-parametric setting by a linearly parametrized function approximation $\sum_k \theta_k \phi_k(x)$, which makes it easier on a computational side, as model estimation now amounts to fitting parameters $\{\theta_k\}$ of such linear function approximation.

Expansions in sets of basis functions have been used in a financial setting, in particular, for pricing American options with Monte Carlo in the popular Least Square Monte Carlo (LSMC) method of Longstaff and Schwartz [11]. A method developed in this paper bears some *mathematical* similarities to the LSMC method of Longstaff and Schwartz, and has a similarly low computational cost, though in our case the *financial* setting is very different from theirs (pricing *and* hedging of a European option under a physical measure $\mathbb{P}$ in our problem, versus pricing an American option under the so-called "risk-neutral" measure $\mathbb{Q}$ in the LSMC method). Some further comments on differences and similarities between the method of this paper and the LSMC method of Longstaff and Schwartz [11] will be presented below in a due course.

A flip side of picking linear architectures is that it is not always easy to find a good set of basis functions and an optimal number of these functions. In general, a good choice of basis functions for linear architectures is a topic of active research in the ML literature. In contrast, for discrete-state models, both a value function and a policy function can be stored in a tabulated form [1]. While a discrete-state case can be considered as a special case of a continuous-state formulation, it makes the question of an optimal choice of a functional basis for function approximation completely trivial.

This implies that we can simplify things even further, and discretize the state space, in order to map the problem onto a *finite-state* MDP. This lets us try simple RL algorithms such as Q-Learning [7], that are *proved* to converge, in the limit of a large data sample, for finite-state MDP problems, and not worry about additional complexity due to the need for a function approximation that would be necessary within a continuous-space formulation [1]. To this end, the continuous-time/continuous-space BSM model can be approximated by a discrete-time/discrete-space Markov Chain approximation that has a smooth continuous-time limit coinciding with the BSM model.

In a nutshell, for benchmarking tasks, we can generate data by simulation of stock price history, alongside with actions (i.e. re-hedges) that implement a risk-minimization strategy, and risk-adjusted returns obtained with such strategy. Then we can give these data to a Q-Learner,

and ask it to find the best hedging (risk minimization) strategy *directly* from these data, without knowing anything about the dynamics and actual hedge strategy that generated these data. But because we actually *know* the best strategy, we can continuously monitor the progress of the Q-Learner towards its goals.

We can also randomize actions at the data generation stage (for example, by intentionally doing sub-optimal hedges from time to time, or even all the time), and again ask the Q-Learner to find the best hedging strategy, by looking at data collected under such a sub-optimal strategy. This would be a simple prototype of how RL could be used in a real trading environment: take the history of the market and own trading strategy, give it to a RL agent, and ask it to improve the strategy by keeping the same goals.

On the other hand, such a problem setting is quite standard for Q-Learning which is an *off-policy* algorithm that is able to learn an optimal policy even when the data used for training is produced using a sub-optimal policy [1]. All such questions of direct relevance for financial applications become answerable in our setting. Moreover, our framework is also quite general, and can be extended to more complex portfolios than a portfolio of a stock and bank cash account, including in particular more practically interesting cases involving multi-asset portfolios.

The paper is organized as follows. In Sect. 2, we present a discrete-time, continuous-space version of the BSM model. An approach based on the formalism of Markov Decision Processes (MDP) is developed in Sect. 3 where we introduce the QLBS model and obtain a DP-based solution for pricing and hedging in this model, that applies when the model is *known*. A RL setting is discussed in Sect. 4, where we introduce a finite-horizon Q-Learning and Fitted Q-Iteration (FQI) for our model, and derive RL-based pricing and hedging formulae that apply when the QLBS model goes *model-free* and purely data driven. In Sect. 5, we briefly discuss possible numerical examples. Finally, we conclude in Sect. 6.

2 Discrete-time Black-Scholes-Merton model

We start with a discrete-time version of the BSM model. As is well known, the problem of option hedging and pricing in this formulation amounts to a sequential risk minimization. The main open question is *how* to define risk in an option. In this part, we follow a local risk minimization approach pioneered in the work of Föllmer and Schweizer [9], Schweizer [10], and Cerný and Kallsen [14]. A similar method was developed by physicists Potters and Bouchaud [15], see also the work by Kapoor et. al. [16]. We use a version of this approach suggested in a Ph.D. thesis by Grau [17].

In this approach, we take the view of a seller of a European option (e.g. a put option) with maturity T and the terminal payoff of $H_T(S_T)$ at maturity, that depends on the final stock price S_T at that time. To hedge the option, the seller use the proceeds of the sale to set up a replicating (hedge) portfolio Π_t made of the stock S_t and a risk-free bank deposit B_t. The value of hedge portfolio at any time $t \leq T$ is

$$\Pi_t = u_t S_t + B_t \tag{1}$$

where u_t is a position in the stock at time t, taken to hedge risk in the option.

2.1 Hedge portfolio evaluation

As usual, the replicating portfolio tries to exactly match the option price in all possible future states of the world. If we start at maturity T when the option position is closed, the hedge u_t

should be closed at the same time, thus we set $u_t = 0$ and therefore

$$\Pi_T = B_T = H_T(S_T) \tag{2}$$

which sets a terminal condition for B_T that should hold in all future states of the world at time T.

To find an amount needed to be held in the bank account at previous times $t < T$, we impose the self-financing constraint which requires that all future changes in the hedge portfolio should be funded from an initially set bank account, without any cash infusions or withdrawals over the lifetime of the option. This implies the following relation that ensures conservation of the portfolio value by a re-hedge at time t:

$$u_t S_{t+1} + e^{r\Delta t} B_t = u_{t+1} S_{t+1} + B_{t+1} \tag{3}$$

This can be expressed as a recursive relation that can be used to calculate the amount of money to keep at the bank account to hedge the option at any time $t < T$ using its value at the next time instance:

$$B_t = e^{-r\Delta t} \left[B_{t+1} + (u_{t+1} - u_t) S_{t+1} \right] , \quad t = T-1, \ldots, 0 \tag{4}$$

Plugging this into Eq.(1) produces a recursive relation for Π_t in terms of its values at later times, which can therefore be solved backward in time, starting from $t = T$ with the terminal condition (2), and continued all the way to the current time $t = 0$:

$$\Pi_t = e^{-r\Delta t} \left[\Pi_{t+1} - u_t \Delta S_t \right] , \quad \Delta S_t = S_{t+1} - e^{r\Delta t} S_t , \quad t = T-1, \ldots, 0 \tag{5}$$

Note that Eqs.(4) and (5) imply that both B_t and Π_t are not measurable at any $t < T$, as they depend on the future. Respectively, their values today B_0 and Π_0 will be random quantities with some distributions. For any given hedging strategy $\{u_t\}_{t=0}^T$, these distributions can be estimated using Monte Carlo simulation, which first simulates N paths of the underlying $S_1 \to S_2 \to \ldots \to S_N$, and then evaluates Π_t going backward on each path. Note that because the choice of a hedge strategy does not affect the evolution of the underlying, such simulation of forward paths should only be performed once, and then re-used for future evaluations of the hedge portfolio under difference hedge strategy scenarios. Alternatively, the distribution of the hedge portfolio value Π_0 can be estimated using real historical data for stock prices, together with a pre-determined hedging strategy $\{u_t\}_{t=0}^T$ and a terminal condition (2).

To summarize, the forward pass of Monte Carlo simulation is done by simulating the process $S_1 \to S_2 \to \ldots \to S_N$, while the backward pass is performed using the recursion (5) that takes a prescribed hedge strategy $\{u_t\}_{t=0}^T$ and back-propagates uncertainty in the future into uncertainty today, via the self-financing constraint (3) [17] which serves as a 'time machine for risk'.

This is exactly what the dealer (seller of the option) needs, as she has to set the price of the option *today*. This can be done e.g. by setting the option price to be the mean of the distribution of Π_0, plus some premium for risk. But all this should obviously come only *after* the seller decides on a *hedging strategy* $\{u_t\}_{t=0}^T$ to be used in the future, that would be applied in the same way (as a mapping) for any future value $\{\Pi_t\}_{t=0}^T$. The choice of an *optimal* hedge strategy $\{u_t\}_{t=0}^T$ will therefore be discussed next.

2.2 Optimal hedging strategy

Unlike the recursive calculation of the hedge portfolio value (5) which is done *path-wise*, optimal hedges are computed using a *cross-sectional* analysis that operates simultaneously on all Monte

Carlo paths. This is because we need to learn a *strategy* $\{u_t\}_{t=0}^T$ that would apply to *all* states that might be encountered in the future, but each given MC path only produces *one* value S_t at time t. Therefore, to compute an optimal hedge $u_t(S_t)$ for a given time step t, we need a *cross-sectional* information on *all* Monte Carlo paths at this time.

Similar to the portfolio value calculation, the optimal hedges $\{u_t\}_{t=0}^T$ are computed backward in time, starting from $t = T$. However, because we cannot know the future when we compute a hedge, for each time t, any calculation of an optimal hedge u_t can only condition on the information $\mathcal{F}_t$ available at time t. This is similar to the American Monte Carlo method of Longstaff and Schwartz [11] (see Appendix B for a short overview).

The optimal hedge $u^\star(S_t)$ in this model is obtained from the requirement that the variance of Π_t across all simulated MC paths at time t is minimized when conditioned on the currently available *cross-sectional* information $\mathcal{F}_t$, i.e.

$$\begin{aligned} u_t^\star(S_t) &= arg\min_u Var\left[\Pi_t|\mathcal{F}_t\right] \\ &= arg\min_u Var\left[\Pi_{t+1} - u_t\Delta S_t|\mathcal{F}_t\right] , \quad t = T-1,\ldots,0 \end{aligned} \tag{6}$$

Note the the first expression in Eq.(6) implies that all uncertainty in Π_t is due to uncertainty regarding the amount B_t needed to be held at the bank account at time t in order to be able to cover the future obligations at the option maturity T. This means that an optimal hedge should minimize the cost of hedge capital at each time step t.

The optimal hedge can be found analytically by setting the derivative of (6) to zero. This gives

$$u_t^\star(S_t) = \frac{Cov\left(\Pi_{t+1}, \Delta S_t|\,\mathcal{F}_t\right)}{Var\left(\Delta S_t|\,\mathcal{F}_t\right)} , \quad t = T-1,\ldots,0 \tag{7}$$

This expression involves one-step expectations of quantities at time $t+1$, conditional on time t. How they can be computed depends on whether we deal with a continuous or a discrete state space. If the state space is discrete, then such one-step conditional expectations are simply finite sums involving transition probabilities of a MDP model. If, on the other hand, we work in a continuous-state setting, these conditional expectations can be calculated in a Monte Carlo setting by using expansions in basis functions, similarly to the LSMC method of Longstaff and Schwartz [11], or real-measure MC methods of Refs. [15, 17, 16].

In our exposition below, we use a general notation as in Eq.(7) to denote similar conditional expectations where $\mathcal{F}_t$ stands for cross-sectional information set at time t, which lets us keep the formalism general enough to handle both cases of a continuous and a discrete state spaces, and discuss simplifications that arise in a special case of a discrete-state formulation separately, whenever appropriate.

2.3 Option pricing in discrete time

We start with the notion of a *fair* option price $\hat{C}_t$ defined as a time-t expected value of the hedge portfolio Π_t:

$$\hat{C}_t = \mathbb{E}_t\left[\Pi_t|\,\mathcal{F}_t\right] \tag{8}$$

Using Eq.(5) and the tower law of conditional expectations, we obtain

$$\begin{aligned} \hat{C}_t &= \mathbb{E}_t\left[e^{-r\Delta t}\Pi_{t+1}\Big|\,\mathcal{F}_t\right] - u_t(S_t)\,\mathbb{E}_t\left[\Delta S_t|\,\mathcal{F}_t\right] \\ &= \mathbb{E}_t\left[e^{-r\Delta t}\mathbb{E}_{t+1}\left[\Pi_{t+1}|\,\mathcal{F}_{t+1}\right]\Big|\,\mathcal{F}_t\right] - u_t(S_t)\,\mathbb{E}_t\left[\Delta S_t|\,\mathcal{F}_t\right] \\ &= \mathbb{E}_t\left[e^{-r\Delta t}\hat{C}_{t+1}\Big|\,\mathcal{F}_t\right] - u_t(S_t)\,\mathbb{E}_t\left[\Delta S_t|\,\mathcal{F}_t\right] , \quad t = T-1,\ldots,0 \end{aligned} \tag{9}$$

Note that we can similarly use the tower law of conditional expectations to express the optimal hedge in terms of $\hat{C}_{t+1}$ instead of Π_{t+1}:

$$u_t^\star(S_t) = \frac{Cov\left(\Pi_{t+1}, \Delta S_t \middle| \mathcal{F}_t\right)}{Var\left(\Delta S_t \middle| \mathcal{F}_t\right)} = \frac{Cov\left(\hat{C}_{t+1}, \Delta S_t \middle| \mathcal{F}_t\right)}{Var\left(\Delta S_t \middle| \mathcal{F}_t\right)} \tag{10}$$

If we now substitute (10) into (9) and re-arrange terms, we can put the recursive relation for $\hat{C}_t$ in the following form:

$$\hat{C}_t = e^{-r\Delta t}\mathbb{E}^{\hat{\mathbb{Q}}}\left[\hat{C}_{t+1}\middle| \mathcal{F}_t\right], \quad t = T-1, \ldots, 0 \tag{11}$$

where $\hat{\mathbb{Q}}$ is a signed measure with transition probabilities

$$\tilde{q}\left(S_{t+1}\middle| S_t\right) = p\left(S_{t+1}\middle| S_t\right)\left[1 - \frac{\left(\Delta S_t - \mathbb{E}_t\left[\Delta S_t\right]\right)\mathbb{E}_t\left[\Delta S_t\right]}{Var\left(\Delta S_t \middle| \mathcal{F}_t\right)}\right] \tag{12}$$

where $p\left(S_{t+1}\middle| S_t\right)$ are transition probabilities under the physical measure $\mathbb{P}$. Note that for large enough moves ΔS_t this expression may become negative, which exactly means that $\hat{\mathbb{Q}}$ is only a *signed* measure rather than a genuine probability measure.

A potential for a negative fair option price $\hat{C}_t$ is a well-known property of quadratic risk minimization schemes [9, 10, 14, 15, 17]. We do *not* view this as a problem for our model, because the "fair" (expected) option price (8) is *not* a price a seller of the option should charge. The actual fair *risk-adjusted* price is given by Eq.(13) below, which can alway be made non-negative by a proper level of risk aversion λ which is defined by the seller's risk preferences. In any case, our model does *not* rely on the no-arbitrage principle because it works directly with raw data. If data respect the no-arbitrage principle, our model will respect it too, by construction.

Still, for the sake of completeness, an alternative approach that avoids negative option prices for arbitrary market/hedging scenarios is presented in Appendix A. Instead of a quadratic risk minimization, this approach relies on an exponential utility $U(X) = -\exp(-\gamma X)$. As shown in Appendix A, the hedges and prices corresponding to the quadratic risk minimization scheme can be obtained with the exponential utility in the limit of a small risk aversion $\gamma \to 0$, alongside calculable corrections via an expansion in powers of γ. As our focus here is on a RL agent that learns from *data* generated from an option trading strategy, in what follows we stick to a simple quadratic hedge (7). If so desired, the whole scheme for optimal hedging and pricing can instead be constructed as described in Appendix A.

Coming back to pricing of the option, the dealer cannot just ask the fair price $\hat{C}_0$ when selling the option, as she has to compensate for risk of exhausting the bank account B_t some time in the future, which would require cash infusions into the hedge portfolio, after any fixed amount $\hat{B}_0 = \mathbb{E}_0\left[B_0\right]$ is put in the bank account at time $t = 0$ right after selling the option. One possible specification of a risk premium that the dealer has to add on top of the fair option price to come up with her own optimal ask price is to add the cumulative expected discounted variance of the hedge portfolio along all time steps $t = 0, \ldots, N$, with a risk-aversion parameter λ:

$$C_0^{(ask)}(S, u) = \mathbb{E}_0\left[\Pi_0 + \lambda\sum_{t=0}^{T} e^{-rt} Var\left[\Pi_t\middle| \mathcal{F}_t\right]\middle| S_0 = S, u_0 = u\right] \tag{13}$$

Note that while the idea of adding an option price premium proportional to the variance of the hedge portfolio as done in Eq.(13) was initially suggested on the intuitive grounds by Potters and Bouchaud [15], a utility-based approach presented in Appendix A actually *derives* it as a

quadratic approximation to a utility-based option price, which also establishes an approximate relation between a risk aversion parameter λ of the quadratic risk optimization, and a parameter γ of the exponential utility $U(X) = -\exp(-\gamma X)$:

$$\lambda \simeq \frac{1}{2}\gamma \tag{14}$$

For what follows, we note that the problem of *minimization* of a fair (to the dealer) option price (13) can be equivalently expresses as the problem of *maximization* of its negative $V_t = -C_t^{(ask)}$, where

$$V_t(S_t) = \mathbb{E}_t\left[-\Pi_t - \lambda \sum_{t'=t}^{T} e^{-r(t'-t)} Var\left[\Pi_{t'} \middle| \mathcal{F}_{t'}\right] \middle| \mathcal{F}_t\right] \tag{15}$$

2.4 Hedging and pricing in the BS limit

The framework presented above provides a smooth transition to the strict BS limit $\Delta t \to 0$. In this limit, the BSM model dynamics under the physical measure $\mathbb{P}$ is described by a continuous-time Geometric Brownian motion with a drift μ and volatility σ:

$$\frac{dS_t}{S_t} = \mu dt + \sigma dW_t \tag{16}$$

where W_t is a standard Brownian motion.

Consider first the optimal hedge strategy (10) in the BS limit $\Delta t \to 0$. Using the first-order Taylor expansion

$$\hat{C}_{t+1} = C_t + \frac{\partial C_t}{\partial S_t}\Delta S_t + O\left(\Delta t\right) \tag{17}$$

in (10), we obtain

$$u_t^{BS}\left(S_t\right) = \lim_{\Delta t \to 0} u_t^{\star}(S_t) = \frac{\partial C_t}{\partial S_t} \tag{18}$$

which is the correct optimal hedge in the continuous-time BSM model.

To find the continuous-time limit of the option price, we first compute the limit of the second term in Eq.(9):

$$\lim_{\Delta t \to 0} u_t(S_t)\, \mathbb{E}_t\left[\Delta S_t \middle| \mathcal{F}_t\right] = \lim_{\Delta t \to 0} u_t^{BS} S_t(\mu - r)dt = \lim_{\Delta t \to 0}(\mu - r) S_t \frac{\partial C_t}{\partial S_t} dt \tag{19}$$

To evaluate the first term in Eq.(9), we use the second-order Taylor expansion:

$$\begin{aligned} \hat{C}_{t+1} &= C_t + \frac{\partial C_t}{\partial t}dt + \frac{\partial C_t}{\partial S_t}dS_t + \frac{1}{2}\frac{\partial^2 C_t}{\partial S_t^2}\left(dS_t\right)^2 + \ldots \\ &= C_t + \frac{\partial C_t}{\partial t}dt + \frac{\partial C_t}{\partial S_t}S_t\left(\mu dt + \sigma dW_t\right) + \frac{1}{2}\frac{\partial^2 C_t}{\partial S_t^2}S_t^2\left(\sigma^2 dW_t^2 + 2\mu\sigma dW_t dt\right) + O\left(dt^2\right) \end{aligned} \tag{20}$$

Plugging Eqs.(19) and (20) into Eq.(9), using $\mathbb{E}\left[dW_t\right] = 0$ and $\mathbb{E}\left[dW_t^2\right] = dt$, and simplifying, we find that the stock drift μ under the physical measure $\mathbb{P}$ drops out from the problem, and Eq.(9) becomes the celebrated Black-Scholes equation in the limit $dt \to 0$:

$$\frac{\partial C_t}{\partial t} + rS_t\frac{\partial C_t}{\partial S_t} + \frac{1}{2}\sigma^2 S_t^2\frac{\partial^2 C_t}{\partial S_t^2} - rC_t = 0 \tag{21}$$

Therefore, if the world is lognormal, both our hedging and pricing formulae become the original formulae of the Black-Scholes-Merton model in the strict limit $\Delta t \to 0$.

3 QLBS

Now we will re-formulate the hedging and pricing method presented in Sect. 2 using the language of Markov Decision Processes (MDP). By keeping in line with formulae presented in Sect. 2 that are valid for both a discrete-state and continuous-state cases, the MDP formulation to be presented here applies for both discrete-state and continuous-state spaces.

Our discrete-time MDP model provides a discrete-time approximation to the hedging and pricing strategy of the BSM model, but by working directly in the physical measure $\mathbb{P}$, and viewing the problem of option pricing and hedging as a problem of stochastic optimal control in discrete time, where the system being controlled is a hedge portfolio, and control is a stock position in this hedge portfolio. The problem is then solved by a sequential maximization of "rewards". These "rewards" are negatives of hedge portfolio one-step variances times the risk-aversion λ, plus a drift term.

As will be shown in detail below, when transition probabilities and a reward function are *known*, a Bellman optimality equation for this model can be solved using a recursive backward Value Iteration method. It turns out that a policy optimization step in this method can be performed *analytically* within our model, as this step amounts to a quadratic optimization problem. The whole calculation is therefore semi-analytical, and only involves matrix linear algebra (linear regression) for a numerical implementation.

On the other hand, we might know only the general *structure* of a MDP model, but *not* its specifications such as transition probability and reward function. In this case, we should solve a backward recursion for the Bellman optimality equation relying only on *samples* of data. This is a setting of *Reinforcement Learning*.

It turns out that a Bellman optimality equation for our MDP model without knowing model dynamics by relying only on *data* can be easily solved (also semi-analytically, due to a quadratic reward function!) by the celebrated *Q-Learning* method of Watkins [6, 7]. As will be shown below, a Q-Learner is *guaranteed* (by construction) to converge to the correct optimal hedge for any time step size, given enough training data, and is guaranteed to converge to the classical BSM results for option price and hedge ratio, if dynamics are lognormal, in the continuous time limit $\Delta t \to 0$. The only important question in this framework is a *numerical speed* of convergence under different model regimes.

In recognition of the fact that the optimal Q-function, learned by a *Q-Learner*, gives both the optimal price and optimal hedge in such time-discretized version of the Black-Scholes(-Merton) model (and learns all this directly from *trading data* for a dynamic hedge portfolio!), we call the model presented below the *QLBS model*.

Viewed as a financial model, our QLBS model feeds on the literature on hedging and pricing in incomplete markets [9, 10, 14, 15, 16, 17] (see also Appendix A). Unlike many previous models of this sort, it ensures a full consistency of hedging and pricing at each time step, all within an efficient and data-driven Q-Learning algorithm. In addition, our MDP formulation generalizes the formalism presented above in Sect. 2 by incorporating a drift in a risk/return analysis of the option's hedge portfolio, following the ideas of the Markowitz portfolio theory [8], but in a multi-period setting. Therefore, it also *extends* a discrete-time BSM model.

This allows one to consider both *hedging and speculation* with options in a consistent way within the *same* model, which is a challenge for the standard BSM model or its "phenomenological" generalizations, see e.g. [5]. As will be shown below, the standard BSM model is obtained from our discrete-time QLBS model by taking a continuous-time limit for multi-period version of a Markowitz-optimal *dynamic* hedge portfolio made of a stock and a cash deposit, if the world is log-normal.

To the extent that option pricing in discrete time amounts to either DP, also known as a model-based RL, if a model is known, and RL if a model is unknown, we may say that the classical continuous-time BSM model corresponds to an extreme version (the continuous-time limit, taken for the sake of a full analytical tractability) of *model-based* Reinforcement Learning, where all data requirements are reduced to just *two* numbers - the current stock price and volatility. In such purely academic BSM world, called the world of a *risk-neutral* valuation in Mathematical Finance, all the drill (risk) is *gone*, and the Q-Learner becomes *jobless*. Its job of a *risk manager* is being eliminated, and replaced by a low cost/low skill job of a mindless calculator that just spits out the price and delta of the classical BSM model, and gives them to a trader to somehow make sense out of these two numbers[1].

Further explorations of these observations could be interesting topics for a derivatives pricing research. Another potential line of research could be to extend this framework for numerical pricing of complex derivatives and derivatives portfolios using Monte Carlo simulation with Q-Learning. Both these topics go beyond the scope of this paper where we focus on the mathematical and algorithmic aspects of our model.

3.1 State variables

We first define a new variable X_t by the following relation:

$$X_t = -\left(\mu - \frac{\sigma^2}{2}\right) t + \log S_t \tag{22}$$

This implies that

$$dX_t = -\left(\mu - \frac{\sigma^2}{2}\right) dt + d\log S_t = \sigma dW_t \tag{23}$$

Therefore, X_t is a standard Brownian motion, scaled by volatility σ. If we know the value of X_t in a given MC scenario, the corresponding value of S_t is given by the formula

$$S_t = e^{X_t + \left(\mu - \frac{\sigma^2}{2}\right)t} \tag{24}$$

Note that as long as X_t is a martingale, i.e. $\mathbb{E}\left[dX_t\right] = 0$, on average it should not run too far away from an initial value X_0 during the lifetime of an option. The state variable X_t is time-uniform, unlike the stock price S_t that has a drift. But the relation (24) can always be used in order to map non-stationary dynamics of S_t into stationary dynamics of X_t. The martingale property of X_t is also helpful for numerical lattice approximations, as it implies that a lattice should not be too large to capture possible future variations of the stock price.

[1] Incidentally, a straightforward extension of this statement could also be applied to *all* of continuous-time "risk-neutral" Mathematical Finance literature. Continuous-time risk-neutral derivatives pricing models, commonly used in the financial industry for trading and pricing options, might be a "simplest possible, and even simpler" approach, paraphrasing the words of Einstein, to the original problem of option pricing and *hedging* by dynamic replication. This is because this *original* task, as formulated in the original BSM approach, and re-interpreted as a local risk minimization problem in discrete time by Föllmer and Schweizer [9, 10], becomes degenerate in the strict limit $\Delta t \to 0$. Indeed, in this limit, and thanks to Ito's calculus, it is only the *pricing* part that remains non-trivial (and amounts to solving the famous Black-Scholes equation, or its extensions to more complex risk-neutral valuation models), while the *risk* (hedging) part becomes trivial, and completely driven by the first part, i.e. by the *pricing*. This order of operations (first pricing, and then hedging) is an exact *opposite* of the actual financial practice of *hedging and pricing*, where hedging comes first, and pricing comes second. The Q-Learner becomes redundant in such counter-factual and irrational world, as there is no more reward (negative risk) to maximize. On the other hand, a *market* in such risk-neutral world is a *fake market*: it should not even *exist* in the first place, for the exact same reason of absence of any risk, and thus any *incentives to trade* in such market. The Q-Learner is no longer needed in such fake market, so it leaves the degenerate BSM world. For a related critique of a risk-neutral approach to Quantitative Finance, see Kapoor *et. al.* [16].

3.2 Finite-state approximation

Though most of our formalism to be presented below applies in both a continuous-state and discrete-state MDP settings, for benchmarking of discrete-state RL algorithms that are simpler than continuous-state algorithms, we may want to apply our method for a discrete-state setting at later stages. This can be done by discretizing the set of admissible values of state variables X_t defined in Eq.(22), while keeping the relation (24) that expresses the stock price S_t in terms of the (now discretized) state variable X_t. A simple method to approximate a continuous-state BSM dynamics by a discrete-time, discrete-state Markov Chain model was developed by Duan and Simonato [19], see Appendix C for a brief summary.

As was mentioned in the introduction, a state-space discretization is *not* necessary within the general methodology that we propose in this paper. A formalism presented below assumes a general continuous-state case in a Monte Carlo setting. This provides a very convenient way to research or benchmark *continuous-state* RL methods for financial applications. On the other hand, if we want to test *discrete-state* RL methods, a Markov Chain approximation to the BSM model can serve as a good benchmark model.

Note that as long as the objective is to benchmark discrete-state RL algorithms, we do *not* necessarily have to worry about a numerical match between a discrete-time Markov Chain approximation and the original continuous-space BSM model. We can always simulate data *directly* from a Markov Chain model, with optimal hedges obtained in *this* model, and benchmark RL algorithms directly against *this model*, rather than against the original continuous-time BSM model.

Also note that in addition to providing a MDP-enabled formulation of the problem, a discrete-state approximation enables an easy computation of various one-step expectations entering our hedging and pricing formulas (7) and (9). Indeed, in this case these expectations would be given by finite sums over all feature discrete states reachable in one step from a given discrete state. In a continuous-space formulation with a MC setting as in Refs. [15, 16, 17], one has instead to rely on function approximations, using e.g. expansions in basis functions, similar to the American Monte Carlo method of Longstaff and Schwartz [11], see also below.

3.3 Value function and Bellman equation

We start with re-stating the risk minimization procedure outlined above in Sect. 2.2 in a language of MDP problems. In particular, time-dependent state variables S_t are expressed in terms of time-homogeneous variables X_t using Eq.(24). In addition, we will use the notation $a_t = a_t(X_t)$ to denote actions expressed as functions of time-homogeneous variables X_t. Actions $u_t = u_t(S_t)$ in terms of stock prices are then obtained by the substitution

$$u_t\left(S_t\right) = a_t\left(X_t\left(S_t\right)\right) = a_t\left(\log S_t - \left(\mu - \frac{\sigma^2}{2}\right)t\right) \tag{25}$$

where we used Eq.(22).

To differentiate between the actual hedging decisions $a_t(x_t)$ where x_t is a particular realization of a random state X_t at time t, and a hedging *strategy* that applies for any state X_t, we introduce the notion of a time-dependent *policy* $\pi\left(t, X_t\right)$. We consider deterministic policies, i.e.

$$\pi : \{0, \ldots, T-1\} \times \mathcal{X} \to \mathcal{A} \tag{26}$$

is a deterministic policy that maps the time t and the current state $X_t = x_t$ into the action $a_t \in \mathcal{A}$:

$$a_t = \pi(t, x_t) \tag{27}$$

We start with the value maximization problem of Eq.(15), which we re-write here in terms of a new state variable X_t, and with an upper index to denote its dependence on the policy π:

$$\begin{aligned} V_t^{\pi}(X_t) &= \mathbb{E}_t\left[-\Pi_t(X_t) - \lambda \sum_{t'=t}^{T} e^{-r(t'-t)} Var\left[\Pi_{t'}(X_{t'})\middle|\mathcal{F}_{t'}\right]\middle|\mathcal{F}_t\right] \\ &= \mathbb{E}_t\left[-\Pi_t(X_t) - \lambda\, Var\left[\Pi_t\right] - \lambda \sum_{t'=t+1}^{T} e^{-r(t'-t)} Var\left[\Pi_{t'}(X_{t'})\middle|\mathcal{F}_{t'}\right]\middle|\mathcal{F}_t\right] \end{aligned} \tag{28}$$

The last term in this expression that involves a sum from $t' = t+1$ to $t' = T$ can be expressed in terms of V_{t+1} using the definition of the value function with a shifted time argument:

$$-\lambda \mathbb{E}_{t+1}\left[\sum_{t'=t+1}^{T} e^{-r(t'-t)} Var\left[\Pi_{t'}\middle|\mathcal{F}_{t'}\right]\right] = \gamma\left(V_{t+1} + \mathbb{E}_{t+1}\left[\Pi_{t+1}\right]\right), \quad \gamma \equiv e^{-r\Delta t} \tag{29}$$

Note that parameter γ introduced in the last relation is a discrete-time discount factor which in our framework is fixed in terms of a continuous-time risk-free interest rate r of the original BSM model.

Substituting this into (28), re-arranging terms and using the portfolio process Eq.(5), we obtain the Bellman equation for the QLBS model:

$$V_t^{\pi}(X_t) = \mathbb{E}_t^{\pi}\left[R(X_t, a_t, X_{t+1}) + \gamma V_{t+1}^{\pi}\left(X_{t+1}\right)\right] \tag{30}$$

where the one-step time-dependent random reward is defined as follows[2]:

$$\begin{aligned} R_t(X_t, a_t, X_{t+1}) &= \gamma a_t \Delta S_t\left(X_t, X_{t+1}\right) - \lambda\, Var\left[\Pi_t\middle|\mathcal{F}_t\right], \quad t = 0, \ldots, T-1 \\ &= \gamma a_t \Delta S_t\left(X_t, X_{t+1}\right) - \lambda\gamma^2 \mathbb{E}_t\left[\hat{\Pi}_{t+1}^2 - 2a_t \Delta\hat{S}_t \hat{\Pi}_{t+1} + a_t^2\left(\Delta\hat{S}_t\right)^2\right] \end{aligned} \tag{31}$$

where we used Eq.(5) in the second line, and $\hat{\Pi}_{t+1} \equiv \Pi_{t+1} - \bar{\Pi}_{t+1}$, where $\bar{\Pi}_{t+1}$ is the sample mean of all values of Π_{t+1}, and similarly for $\Delta\hat{S}_t$. For $t = T$, we have $R_T = -\lambda\, Var\left[\Pi_T\right]$ where Π_T is determined by the terminal condition (2).

Note that Eq.(31) implies that the expected reward R_t at time step t is *quadratic* in the action variable a_t:

$$\mathbb{E}_t\left[R_t\left(X_t, a_t, X_{t+1}\right)\right] = \gamma a_t \mathbb{E}_t\left[\Delta S_t\right] - \lambda\gamma^2 \mathbb{E}_t\left[\hat{\Pi}_{t+1}^2 - 2a_t\Delta\hat{S}_t\hat{\Pi}_{t+1} + a_t^2\left(\Delta\hat{S}_t\right)^2\right] \tag{32}$$

As we will see shortly, this proves very useful for a solution of the MDP dynamics in this model. Also note that when $\lambda \to 0$, the expected reward is *linear* in a_t, so it does not have a maximum.

Another comment that is due here is that a similar construction of a risk-sensitive MDP by adding one-step variance penalties to a finite-horizon risk-neutral MDP problem was suggested in a different context by Gosavi [20][3]. Here we rely on a similar idea, and incorporate a negative quadratic reward (i.e. risk) into a standard risk-neutral MDP formulation.

The *optimal policy* $\pi_t^{\star}(\cdot|X_t)$ is determined as a policy that maximizes the value function $V_t^{\pi}\left(X_t\right)$:

$$\pi_t^{\star}(X_t) = arg\, \max_{\pi} V_t^{\pi}(X_t) \tag{33}$$

[2]Note that with our definition of the value function Eq.(28), it is *not* equal to a discounted sum of future rewards.

[3]Incidentally, Gosavi quoted the Markowitz portfolio theory as an inspiration of his version of a risk-sensitive MDP.

The optimal value function satisfies the Bellman optimality equation

$$V_t^\star(X_t) = \mathbb{E}_t^{\pi^\star}\left[R_t(X_t, u_t = \pi_t^\star(X_t), X_{t+1}) + \gamma V_{t+1}^\star\left(X_{t+1}\right)\right] \tag{34}$$

If the system dynamics are known, the Bellman optimality equation can be solved using methods of Dynamic Programming such as Value Iteration. If, on the other hand, dynamics are unknown and the optimal policy should be computed using *samples*, which is a setting of Reinforcement Learning, then a formalism based on an action-value function, to be presented next, provides a better framework for Value Iteration methods.

3.4 Action-value function

The action-value function, or Q-function, is defined by an expectation of the same expression as in the definition of the value function (28), but conditioned on both the current state X_t *and* the initial action $a = a_t$, while following a policy π afterwards:

$$\begin{aligned} Q_t^\pi(x,a) &= \mathbb{E}_t\left[-\Pi_t(X_t)\middle|\, X_t = x, a_t = a\right] \\ -\lambda\, \mathbb{E}_t^\pi &\left[\sum_{t'=t}^T e^{-r(t'-t)} Var\left[\Pi_{t'}(X_{t'})\middle|\, \mathcal{F}_{t'}\right]\middle|\, X_t = x, a_t = a\right] \end{aligned} \tag{35}$$

Note that the first expectation involves only averaging over the next time step and is therefore independent of the policy π, while the second expectation depends on π.

In a similar way to the derivation of the Bellman equation for the value function (30), we can obtain the Bellman equation for the Q-function:

$$Q_t^\pi(x,a) = \mathbb{E}_t\left[R_t(X_t, a_t, X_{t+1})\middle|\, X_t = x, a_t = a\right] + \gamma \mathbb{E}_t^\pi\left[V_{t+1}^\pi\left(X_{t+1}\right)\middle|\, X_t = x\right] \tag{36}$$

An optimal action-value function $Q_T^\star(x,a)$ is obtained when (35) is evaluated with an optimal policy $\pi_t^\star$:

$$\pi_t^\star = \arg\max_\pi Q_t^\pi(x,a) \tag{37}$$

The optimal value- and state-value functions are connected by the following equations

$$\begin{aligned} V_t^\star(x) &= \max_a Q_t^\star(x,a) \\ Q_t^\star(x,a) &= \mathbb{E}_t\left[R_t(x,a,X_{t+1})\right] + \gamma \mathbb{E}\left[V_{t+1}^\star(X_{t+1})\middle|\, X_t = x\right] \end{aligned} \tag{38}$$

The Bellman Optimality equation for the action-value function is obtained by substituting the first of Eqs.(38) into the second one:

$$Q_t^\star(x,a) = \mathbb{E}_t\left[R_t\left(X_t, a_t, X_{t+1}\right) + \gamma \max_{a_{t+1}\in\mathcal{A}} Q_{t+1}^\star\left(X_{t+1}, a_{t+1}\right)\middle|\, X_t = x, a_t = a\right], \quad t = 0, \ldots, T-1 \tag{39}$$

with a terminal condition at $t = T$ given by

$$Q_T^\star(X_T, a_T = 0) = -\Pi_T\left(X_T\right) - \lambda\, Var\left[\Pi_T\left(X_T\right)\right] \tag{40}$$

where Π_T is determined by the terminal condition (2). Recall that $Var\left[\cdot\right]$ here means variance with respect to all Monte Carlo paths that end up in a given state. A paired equation defines a "greedy" policy $\pi^\star$ that always seeks an action that maximizes the action-value function in the current state:

$$\pi_t^\star(X_t) = arg\max_{a_t\in\mathcal{A}} Q_t^\star(X_t, a_t) \tag{41}$$

3.5 Backward recursion for the Q-function and the QLBS option price

If transition probabilities to compute the expectation in the right-hand side of the Bellman optimality equation (39) are *known*, then the Bellman equation (39) can be solved, jointly with the optimal policy (41), using backward recursion starting from $t = T - 1$ and the terminal condition (40). This can be used for benchmarking in our test environment where we *do* know these probabilities, and know the rewards function (31).

Substituting the expected reward (32) into the Bellman optimality equation (39) we find that $Q_t^\star(X_t, a_t)$ is *quadratic* in the action variable a_t:

$$\begin{aligned} Q_t^\star(X_t, a_t) &= \gamma \mathbb{E}_t \left[Q_{t+1}^\star \left(X_{t+1}, a_{t+1}^\star \right) + a_t \Delta S_t \right] \\ &- \lambda \gamma^2 \, \mathbb{E}_t \left[\hat{\Pi}_{t+1}^2 - 2 a_t \hat{\Pi}_{t+1} \Delta \hat{S}_t + a_t^2 \left(\Delta \hat{S}_t \right)^2 \right], \quad t = 0, \ldots, T-1 \end{aligned} \tag{42}$$

Note that in the limit of zero risk aversion $\lambda \to 0$, this formula becomes

$$Q_t^\star(X_t, a_t) = \gamma \mathbb{E}_t \left[Q_{t+1}^\star \left(X_{t+1}, a_{t+1}^\star \right) + a_t \Delta S_t \right] \tag{43}$$

As in this limit $Q_t^\star(X_t, a_t) = -\Pi(X_t, a_t)$, using the the fair option price definition (8), we obtain

$$\hat{C}_t = \gamma \mathbb{E}_t \left[\hat{C}_{t+1} - a_t \Delta S_t \right] \tag{44}$$

This coincides with Eq.(9). This means that the recursive formula (42) correctly rolls back the *BS fair* option price $\hat{C}_t = \mathbb{E}_t[\Pi_t]$, which corresponds to first taking the limit $\lambda \to 0$, and then taking the limit $\Delta t \to 0$ of the QLBS price (while using the BS delta for a_t in Eq.(44), see below).

As in a general case $\lambda > 0$, $Q_t^\star(X_t, a_t)$ is a *quadratic* function of a_t, the optimal action (i.e. the hedge) $a_t^\star(S_t)$ that maximizes $Q_t^\star(X_t, a_t)$ is computed analytically:

$$a_t^\star(X_t) = \frac{\mathbb{E}_t \left[\Delta \hat{S}_t \hat{\Pi}_{t+1} + \frac{1}{2\gamma\lambda} \Delta S_t \right]}{\mathbb{E}_t \left[\left(\Delta \hat{S}_t \right)^2 \right]} \tag{45}$$

If we now take the limit of this expression as $\Delta t \to 0$ by using Taylor expansions around time t as in Sect. 2.4, we obtain

$$\lim_{\Delta t \to 0} a_t^\star = \frac{\partial \hat{C}_t}{\partial S_t} + \frac{\mu - r}{2\lambda\sigma^2} \frac{1}{S_t} \tag{46}$$

Note that if we set $\mu = r$, or alternatively if we take the limit $\lambda \to \infty$, it becomes identical to the BS delta, while the finite-Δt delta in Eq.(45) coincides in these cases with a local-risk minimization delta given by Eq.(7). Both these facts have related interpretations. The quadratic hedging that approximates option delta (see Sect. 2.4) only looks at *risk* of a hedge portfolio, while here we extend it by adding a *drift term* $\mathbb{E}_t[\Pi_t]$ to the objective function, see Eq.(15), in the spirit of Markowitz risk-adjusted portfolio return analysis [8]. This produces a linear first term in the quadratic expected reward (32). Resulting hedges are therefore different from hedges obtained by only minimizing risk. Clearly, a pure risk-focused quadratic hedge corresponds to either taking the limit of *infinite* risk aversion rate in a Markowitz-like risk-return analysis, or setting $\mu = r$ in the above formula, to achieve the same effect. Both factors appearing in Eq.(46) show these two possible ways to obtain pure risk-minimizing hedges from our more general hedges. Such hedges can be applied when an option is considered for investment/speculation, rather than only as a hedge instrument.

To summarize, the local risk-minimization hedge and fair price formulae of Sect. 2 are recovered from Eqs.(45) and (42), respectively, if we first set $\mu = r$ in Eq.((45), and then set $\lambda = 0$ in Eq.(42). After that, the continuous-time BS formulae for these expressions are reproduced in the final limit $\Delta t \to 0$ in these resulting expressions, as discussed in Sec. 2. Note that the order of taking the limits is to start with the hedge ratio (46), set there $\mu = r$, then plug this into the price equation (42), and take the limit $\lambda \to 0$ there, leading to Eq.(44). The latter relation yields the Black-Scholes equation in the limit $\Delta t \to 0$ as shown in Eq.(21). This order of taking the BS limit is consistent with the principle of hedging first and pricing second, which is implemented in the QLBS model, as well as consistent with market practices of working with illiquid options.

Plugging Eq.(45) back into Eq.(42), we obtain an explicit recursive formula for the *optimal* action-value function:

$$Q_t^\star(X_t, a_t^\star) = \gamma \mathbb{E}_t \left[Q_{t+1}^\star(X_{t+1}, a_{t+1}^\star) - \lambda \gamma \hat{\Pi}_{t+1}^2 + \lambda \gamma \left(a_t^\star\left(X_t\right)\right)^2 \left(\Delta \hat{S}_t\right)^2 \right], \quad t = 0, \ldots, T-1 \quad (47)$$

where $a_t^\star\left(X_t\right)$ is defined in Eq.(45). Note that this relation does *not* have the right risk-neutral limit when we set $\lambda \to 0$ in it. The reason is that setting $\lambda \to 0$ in Eq.(47) is equivalent to setting $\lambda \to 0$ in Eq.(45), but, as we just discussed, this would *not* be the right way to reproduce the BS option price equation (21). The correct procedure of taking the limit $\lambda \to 0$ in the recursion for the Q-funciton is given by Eq.(43) which implies that action a_t used there is obtained as explained above by setting $\mu = r$ in Eq.(46).

The backward recursion given by Eqs. (45) and (47) proceeds all the way backward starting at $t = T - 1$ to the present $t = 0$. At each time step, the problem of maximization over possible actions amounts to convex optimization which is done analytically using Eq.(45), which is then substituted into Eq.(47) for the current time step. Note that such simplicity of action optimization in the Bellman optimality equation is not encountered very often in other SOC problems. As Eq.(47) provides the backward recursion directly for the *optimal* Q-function, neither continuous nor discrete action space representation is required in our setting, as the action in this equation is always just one *optimal* action. If we deal with a finite-state QLBS model, then the values of the optimal time-t Q-function for each node are obtained directly from sums of values of the next-step expectation in various states at time $t + 1$, times one-step probabilities to reach these states.

The end result of the backward recursion for the action-value function is its current value. According to our definition of the option price (13), it is exactly the negative of the optimal Q-function. We therefore obtain the following expression for the fair ask option price in our approach, which we can refer to as the *QLBS option price*:

$$C_t^{(QLBS)}(S_t, ask) = -Q_t\left(S_t, a_t^\star\right) \quad (48)$$

It is interesting to note that while in the original BSM model the price and the hedge for an option are given by two separate expressions, in the QLBS model, they are parts of the *same* expression (48) - simply because our option price is the (negative of the) optimal Q-function, whose second argument is by definition the optimal *action*, that is, the *hedge*, in the setting of our model.

Eqs.(48) and (45) that give, respectively, the optimal price and the optimal hedge for the option, jointly provide a complete solution of the QLBS model (when the dynamics are *known*) that generalizes the classical BSM model towards a non-asymptotic case $\Delta t > 0$, while reducing to the latter in the strict BSM limit $\Delta t \to 0$.

3.6 Monte Carlo implementation

In this section, we assume a general continuous-space scenario within a Monte Carlo simulation, and derive general recursive formulae for optimal hedge and optimal Q-function that work for either continuous-state or discrete-state formulation of our model. Solutions are presented in terms of coefficients in expansions of these functions over some set of basis functions. This approach does not offer a choice of a good set of basis functions, which is a question left for a researcher to decide upon. However, if our MC simulation is done for a *discrete-state* version of the QLBS model, then the problem of choice of good basis function becomes trivial, as they can be simply taken as "one-hot" representations for all nodes (see below).

Similar to Refs. [11, 15, 17], our model uses *all* Monte Carlo (or historical) paths for the replicating portfolio simultaneously. Note a difference between such approach and a lattice-based approach that could alternatively be used for a finite-state version of the model in the backward recursion for the Q-function described in Sect. 3.5 . With a lattice approach, for each time step t, expectations with respect to future scenarios at time $t+1$ are computed for each node X_t separately. While such approach is fine when the model is known and a state space is finite and small, in a setting of RL, when the model is *unknown*, this amounts to asynchronous updates of both the policy and the action-value function, which might substantially slow down the learning process.

However, in a Monte Carlo scheme, we average over all scenarios at time t and $t+1$ simultaneously, by taking an empirical mean of different path-wise Monte Carlo scenarios. Therefore there is no need in explicit conditioning on the market state X_t at time t. State variable values at both times t and $t+1$ for *all* MC paths are used *simultaneously* to find the optimal action and optimal Q-function at time t, for all states X_t. Learning optimal actions for all states simultaneously means learning a *policy*, which is exactly our objective.

Assume we have a set of basis functions $\{\Phi_n(x)\}$. We can then expand the optimal action (hedge) $a_t^\star(X_t)$ and optimal Q-function $Q_t^\star(X_t, a_t^\star)$ in basis functions, with time-dependent coefficients:

$$\begin{aligned} a_t^\star(X_t) &= \sum_n^M \phi_{nt}\Phi_n(X_t) \\ Q_t^\star(X_t, a_t^\star) &= \sum_n^M \omega_{nt}\Phi_n(X_t) \end{aligned} \tag{49}$$

Coefficients ϕ_{nt} and ω_{nt} are computed recursively backward in time for $t = T-1, \ldots, 0$. First, we find coefficients ϕ_{nt} of the optimal action expansion. This is found by minimization of the following quadratic functional that is obtained by replacing the expectation in Eq.(42) by a MC estimate, dropping all a_t-independent terms, substituting the expansion (49) for a_t, and changing the overall sign to convert a maximization problem into a minimization problem:

$$G_t(\phi) = \sum_{k=1}^{N_{MC}} \left(-\sum_n \phi_{nt}\Phi_n\left(X_t^k\right)\Delta S_t^k + \gamma\lambda\left(\hat{\Pi}_{t+1}^k - \sum_n \phi_{nt}\Phi_n\left(X_t^k\right)\hat{S}_t^k\right)^2\right) \tag{50}$$

This formulation automatically takes care of averaging over market scenarios at time t.

Minimization of Eq,(50) with respect to coefficients ϕ_{nt} produces a set of linear equations:

$$\sum_m^M A_{nm}^{(t)}\phi_{mt} = B_n^{(t)}, \quad n = 1, \ldots, M \tag{51}$$

where

$$\begin{aligned} A_{nm}^{(t)} &= \sum_{k=1}^{N_{MC}} \Phi_n\left(X_t^k\right) \Phi_m\left(X_t^k\right) \left(\Delta \hat{S}_t^k\right)^2 \\ B_n^{(t)} &= \sum_{k=1}^{N_{MC}} \Phi_n\left(X_t^k\right) \left[\hat{\Pi}_{t+1}^k \Delta \hat{S}_t^k + \frac{1}{2\gamma\lambda} \Delta S_t^k\right] \end{aligned} \tag{52}$$

which produces the solution for the coefficients of expansions of the optimal action $a_t^\star\left(X_t\right)$ in a vector form:

$$\phi_t^\star = \mathbf{A}_t^{-1} \mathbf{B}_t \tag{53}$$

where $\mathbf{A_t}$ and $\mathbf{B_t}$ are a matrix and vector, respectively, with matrix elements given by Eq.(52). Note a similarity between this expression and the general relation (45) for the optimal action.

Once the optimal action $a_t^\star$ at time t is found in terms of its coefficients (53), we turn to the problem of finding coefficients ω_{nt} of the basis function expansion (49) for the optimal Q-function. To this end, the one-step Bellman optimality equation (39) for $a_t = a_t^\star$ is interpreted as regression of the form

$$R_t\left(X_t, a_t^\star, X_{t+1}\right) + \gamma \max_{a_{t+1} \in \mathcal{A}} Q_{t+1}^\star\left(X_{t+1}, a_{t+1}\right) = Q_t^\star(X_t, a_t^\star) + \varepsilon_t \tag{54}$$

where ε_t is a random noise at time t with mean zero. Clearly, taking expectations of both sides of (54), we recover Eq.(39) with $a_t = a_t^\star$, therefore Eqs. (54) and (39) are equivalent in expectations when $a_t = a_t^\star$.

Coefficients ω_{nt} are therefore found by solving the following least-square optimization problem:

$$F_t(\omega) = \sum_{k=1}^{N_{MC}} \left(R_t\left(X_t, a_t^\star, X_{t+1}\right) + \gamma \max_{a_{t+1} \in \mathcal{A}} Q_{t+1}^\star\left(X_{t+1}, a_{t+1}\right) - \sum_n^M \omega_{nt} \Phi_n\left(X_t^k\right) \right)^2 \tag{55}$$

Introducing another pair of a matrix $\mathbf{C_t}$ and a vector $\mathbf{D_t}$ with elements

$$\begin{aligned} C_{nm}^{(t)} &= \sum_{k=1}^{N_{MC}} \Phi_n\left(X_t^k\right) \Phi_m\left(X_t^k\right) \\ D_n^{(t)} &= \sum_{k=1}^{N_{MC}} \Phi_n\left(X_t^k\right) \left(R_t\left(X_t, a_t^\star, X_{t+1}\right) + \gamma \max_{a_{t+1} \in \mathcal{A}} Q_{t+1}^\star\left(X_{t+1}, a_{t+1}\right) \right) \end{aligned} \tag{56}$$

we obtain the vector-valued solution for optimal weights ω_t defining the optimal Q-function at time t:

$$\omega_t^\star = \mathbf{C}_t^{-1} \mathbf{D}_t \tag{57}$$

Equations (53) and (57), computed jointly and recursively for $t = T-1, \ldots, 0$ provide a practical implementation of the backward recursion scheme of Sect. 3.5 in a continuous-space setting using expansions in basis functions. This approach can be used to find optimal price and optimal hedge when the dynamics are *known*.

Note that our numerical scheme is similar to the Least Square Monte Carlo (LSMC) method of Longstaff and Schwartz (LS) [11], as well as methods of Hedged Monte Carlo (HMC) of Potters and Bouchaud (PB) [15], and Simulation-Based Hedging of Grau [17]. Similar to the LSMC method of Longstaff and Schwartz [11]) and the method of Grau [17] (but different from the

HMC method of Potters and Bouchaud [15]), possible inaccuracies incurred in rolling back the optimal Q-function do *not* impact the resulting optimal hedge (53). This is similar to the LSMC method of Longstaff and Schwartz, where inaccuracies in determining the continuation value for an option do not impact the quality of a computed exercise boundary. But none of these previous Monte Carlo based approaches is able to do what the QLBS model can - namely, go *model-free* in a consistent way for both the price and the hedge. This is because in the QLBS model, unlike these previous approaches, the price is identified with the Q-function, which immediately brings all the benefits of Q-Learning, and most importantly, the fact the Q-Learning is model-free.

As mentioned above, if we work with a discrete-state version of QLBS, then Equations (53) and (57) simplify even further, as for a discrete-state MDP, we can simply use "one-hot" basis functions, that are equal one for a given node, and zero otherwise:

$$\Phi_n^{(finite-state)}\left(X_t\right) = \delta_{X_t,X_n} \tag{58}$$

where $\delta_{a,b} = 1$ if $a = b$, and zero otherwise , and X_n is the n-th node on a grid of discretized X-values. These basis functions would serve as accumulators of contributions of individual grid states into sums in Eqs.(52) and (56). A smoothed-out choice of such localized basis for a continuous-state version of our model could be provided by B-splines or Gaussian kernels, while for a multi-dimensional continuous-state case one could use multivariate B-splines or Radial Basis Functions (RBF's).

3.7 Discussion

As we have shown above, the correct order of taking the BS limit amounts to first setting $\mu = r$ (or $\lambda \to \infty$) in Eq.(45), then taking the limit $\lambda \to 0$ in Eq.(42), and then taking the final limit $\Delta t \to 0$.

On the other hand, note that because the local reward (31) has a drift and a variance penalty term, our problem can also be viewed as a multi-period version of the Markowitz portfolio model where an investment portfolio is an option replicating portfolio made of a stock and a bank cash account.

The above convergence of our QLBS model to the BSM model as described above implies that for a specific investment portfolio made of a stock and a bank cash account, this particular multi-period extension of the Markowitz theory converges to the BSM price and delta of the option. Such possible link between the Markowitz portfolio model and the BSM model appears new, at least to the author.

The second comment we want to make is that our Bellman optimality equation (39) for the optimal action-value function is *guaranteed* by construction, at least for a finite-state formulation of the QLBS model, to converge to the correct (negative of the) option price and option hedge of the BSM model in the joint limit $\lambda \to 0$, $\Delta t \to 0$, while providing calculable corrections for a pre-asymptotic regime $\Delta t > 0$, $\lambda > 0$ by using a backward recursion of the Q-function (39), along with a greedy policy choice (41). We will leave a detailed investigation of empirical behavior of option prices and hedges in this pre-asymptotic regime to a future work, while concentrating in this paper on a mathematical framework.

In what comes to a finite Δt, finite λ regime corresponding to a general setting of the QLBS model (35), we have shown that the approach of a Bellman optimality equation for the action-value Q-function provides a very natural and easily computable link between a particular local-risk minimization hedging strategy (an optimal policy, in the language of MDP models) and a corresponding price for the option that should be a functional of this strategy. In our model, we suggested using a discounted expected cumulative variances of a hedge portfolio from

all future hedge rebalance periods as a measure of expected risk in the option, that should be compensated to the dealer in accord with her risk tolerance λ. While this choice of a risk-premium for option *pricing* is indeed non-unique, in complete agreement with the principle 'hedging first, pricing next', as we saw above, it leads to a very tractable MDP formulation that can be solved semi-analytically using the backward recursion for the optimal Q-function as described above, if both the dynamics and reward function are known.

Coming back to the issue of possible non-uniqueness of option prices even under the same hedging strategies, we note that while our approach is identical (if we set $\mu = r$ in Eq.(45)) to the approach of Grau [17] in the *hedging* part, our model differs from the method of Grau in the *pricing* part. While Grau suggested to use a CVAR-adjusted price as a compensation for risk in the option [17], it is in a sense an "after-thought" of his scheme. Another risk-related premium (for example, VAR) could be used there in place of CVAR, and the hedging method would still be the same. In other words, there is no direct link between the hedging and pricing in the approach of [17]. Grau's approach is a pure "direct policy search", in the language of Reinforcement Learning [1], that does not rely on any value function in optimization of a hedge strategy.

On the other hand, we suggest a different definition of a fair ask option price, that is entirely consistent with a prescribed local-risk minimization hedging strategy. This price is given by a negative of the Q-function $Q_0(X_0, a_0^\star)$ at time $t = 0$. If the state dynamics and reward functions are *known*, then the Bellman optimality equation for the Q-function can be solved recursively backward in time, starting from $t = T - 1$. The optimization step, performed T times in our scheme, amounts to a quadratic optimization task which can be performed analytically. No discretization of the action space is required, due to a quadratic dependence of the action-value function Q_t at time t on the action a_t for this time step. This implies, in particular, that we can benchmark both continuous-state and discrete-state RL algorithms with either a continuous-state (discrete-time) QLBS model, or a finite-state QLBS model which is obtained by a state-discretization of the continuous-state MDP model.

On the other hand, in most real-world situations the dynamics of a system is *unknown*, therefore the Bellman optimality equation (39) should be solved using only *samples* from data, without any pre-specified model of the world. This is a topic for Reinforcement Learning methods that we will discuss next.

4 Q-Learning and Fitted Q Iteration in QLBS

Reinforcement Learning (RL) solves the same problem as Dynamic Programming (DP), i.e. it finds an optimal policy. But unlike DP, Reinforcement Learning does *not* assume that transition probabilities and reward function are known. Instead, it relies on samples to find an optimal policy. We will focus on value-based RL methods that work with the same objects (the value function and action-value function) as DP [1].

Our setting assumes a *batch-mode* learning, when we only have access to some historically collected data. No access to a real-time environment, or a simulator of such environment, is assumed available. The data available is given by a set of N trajectories for the underlying stock S_t (expressed as a function of X_t using Eq.(24)), hedge position a_t , instantaneous reward R_t, and the next-time value X_{t+1}:

$$\mathcal{F}_t^{(n)} = \left\{ \left(X_t^{(n)}, a_t^{(n)}, R_t^{(n)}, X_{t+1}^{(n)} \right) \right\}_{t=0}^{T-1} , \quad n = 1, \ldots, N \tag{59}$$

Moreover, as long as the dynamics are Markov, a collection of N trajectories of length T each is

equivalent, for the purpose of training a RL model, to a dataset of $N \times T$ single-step transitions. We assume that such dataset is available either as a simulated data, or as a real historical stock price data, combined with some artificial data that would track a performance of a hypothetical stock-and-cash replicating portfolio for a given option.

Neither the dynamics nor the true reward distribution are assumed known in the RL approach. All that is given is a set of one-step transitions. Such off-line setting for RL is known as batch-mode Reinforcement Learning [12]. The paper of Ernst *et. al.* proposed to use a *fitted Q-iteration* method for such setting [12]. This method combines the celebrated Q-Learning method for model-free RL [7] with function approximations to treat large discrete or continuous state spaces. While methods presented by Ernst *et. al.* deal with infinite-horizon batch mode RL, this setting was extended to a finite-horizon case by Murphy [13], see also a Ph.D. thesis by Fonteneau [21]. We will start with a pure Q-Learning, and then discuss Fitted Q Iteration.

4.1 Stochastic approximations

When the model is unknown, we try to *approximately* solve the Bellman optimality equation (39) by replacing expectations entering this equation by their empirical averages. Stochastic approximations such as the Robbins-Monro algorithm [22] take this idea one step further, and estimate the mean without directly summing the samples, but instead adding data points one by one, and iteratively updating the running estimation of the mean $\hat{x}^k$ (where k is the number of iteration, or the number of data points in a dataset):

$$\hat{x}^{k+1} = (1 - \alpha^k)\hat{x}^k + \alpha^k x^k \tag{60}$$

where x^k is the k-th data point, and $\alpha^k < 1$ denotes the step size (learning rate) that should satisfy the following conditions:

$$\lim_{K\to\infty} \sum_{k=1}^{K} \alpha^k = \infty\,, \quad \lim_{K\to\infty} \sum_{k=1}^{K} \left(\alpha^k\right)^2 < \infty \tag{61}$$

Robbins and Monro have shown that under these constraints, an iterative method of computing the mean (60) converges to a true mean with probability one [22]. In general, the optimal choice of a (step-dependent) learning rate α^k is not universal but specific to a problem, and may require some experimentation. The Robbins-Monro algorithm (60) is an *on-line* algorithm, which explains its popularity in real-time ML and RL applications, see e.g. [23]. Alternatively, stochastic approximations can be used to pick a *chunk* of data, instead of a single data point, to iteratively update model parameters. Such approaches are useful when working with large datasets, and are used e.g. in a mini-batch Stochastic Gradient Descent method, as a way to more efficiently train a model by feeding it mini-batches of data.

4.2 Q-Learning

Here we assume a discrete-state/discrete-action version of the QLBS model, as the Q-Learning in its original form suggested by Watkins in 1989 [6] applies only for such setting. In this case, Q-Learning converges to the true optimal action-value function with probabilty one, given enough data. Note that the original Q-Learning method of Watkins [6, 7] refers to an infinite-horizon MDP problem, while we deal with a *finite-horizon* MDP problem. But stochastic approximation works the same way in this setting as for infinite-horizon problems: we just have to apply it *separately* at each step of a backward recursion for the Bellman optimality equation (39) for the Q-function, and asymptotic convergence is still guaranteed [13].

The classical Q-Learning method implements a single-step update of a current value of the Q-function, or in fact, a Q-*value* for each combination of a state and action, as both the state space and action space are discrete. In other words, the Q-function for an infinite-horizon discrete-state/discrete-action MDP is represented in a *tabulated form*, as a two-dimensional matrix/tensor. For a *time-dependent* problem with a *finite* time horizon, which is our case here, a time argument is added to a Q-function/table, so it is represented by a *three*-dimensional tensor for a discrete state-action space.

Q-Learning is obtained by using the Robbins-Monro stochastic approximation (60) to estimate the unknown expectation in Eq.(39) which we repeat here for convenience:

$$Q_t^\star(x,a) = \mathbb{E}_t\left[R_t\left(X_t, a_t, X_{t+1}\right) + \gamma \max_{a_{t+1}\in\mathcal{A}} Q_{t+1}^\star\left(X_{t+1}, a_{t+1}\right)\middle| X_t = x, a_t = a\right], \quad t = 0, \ldots, T-1 \tag{62}$$

with a terminal condition at $t = T$ given by Eqs.(40) and (2). Recall that $Var\left[\cdot\right]$ here means variance with respect to all Monte Carlo paths that end up in a given state.

Using the Robbins-Monro update (60) , we obtain an update rule for the optimal Q-function with online Q-Learning after observing one datapoint $\left(X_t^{(n)}, a_t^{(n)}, R_t^{(n)}, X_{t+1}^{(n)}\right)$:

$$Q_t^{\star,k+1}(X_t, a_t) = (1-\alpha^k)Q_t^{\star,k}(X_t, a_t) + \alpha^k\left[R_t\left(X_t, a_t, X_{t+1}\right) + \gamma \max_{a_{t+1}\in\mathcal{A}} Q_{t+1}^{\star,k}\left(X_{t+1}, a_{t+1}\right)\right] \tag{63}$$

In addition to an online undate of the Q-function, similar Robbins-Monro updates should be applied to estimate expectations that enter Eq.(45).

The Q-iteration rule of Eq.(63) shows why Q-Learning is both a *model-free* and *off-policy* algorithm: it *does not make any assumptions* on a true data-generating process that produced the observation $\left(X_t^{(n)}, a_t^{(n)}, R_t^{(n)}, X_{t+1}^{(n)}\right)$. It simply takes it as *given*, and updates the action-value function for a given state-action node $\left(X_t^{(n)}, a_t^{(n)}\right)$. Q-iteration (63) is guaranteed to asymptotically converge to a true optimal value function for all state-action pairs, given that each such pair is encountered infinitely many times in data [6]. For more on convergence of Q-Learning algorithms see e.g. [23].

Given the results obtained in Sect. 2, it should be evident that even though the classical online Q-Learning algorithm is *guaranteed* to asymptotically converge, it might take it too long for practical purposes. The reason for this is that optimal hedges are obtained using *cross-sectional* information across all Monte Carlo paths. Such cross-sectional information would be masked for any on-line method of updating Q-values and optimal hedge ratios $a_t^\star\left(X_t\right)$ on a space grid.

However, the way out in our case is also clear - we simply have to update Q-values and values of $a_t^\star\left(X_t\right)$ at all points on the grid *simultaneously*, by looking, for each time t, at a time-t snapshot of *all* Monte Carlo paths. Because we are in a setting of *batch-mode* Reinforcement Learning, the data are *already there*, and there is no need to take data points one by one, as in done in the classical Q-Learning.

4.3 Batch Q-Learning: Fitted Q Iteration

Fortunately, extensions of Q-Learning for a batch-mode RL are readily available. Again, we take a most popular extension of Q-Learning to a batch RL setting called Fitted Q Iteration (FQI) [12, 13]. The model formulation is now back to a general continuous-state space case, as in a Monte Carlo setting used in this paper, the only difference between a continuous-state and a discrete-state specification amounts to the choice of basis functions to use.

In a similar way to a Monte Carlo method of Sect. 3.6 that was used above for the case of known dynamics, the FQI method in our model works using *all* Monte Carlo (or historical) paths for the replicating portfolio simultaneously. In this method, we average over all scenarios at time t and $t+1$ simultaneously, by taking an empirical mean of different path-wise Monte Carlo scenarios. Therefore there is no need in explicit conditioning on the market state X_t at time t.

The only thing that needs to be changed in such Monte Carlo setting to make it amenable to batch-mode RL is the structure of input-output data. In the setting of Sect. 3.6, when the model is *known*, the inputs are simulated (or real) forward paths of the state variable X_t, and the outputs are the optimal action (hedge) policy and optimal Q-function (i.e. the negative option price). The optimal policy $\pi_t^\star$ is defined in terms of optimization problem of Eq.(41), and instantaneous rewards are known from Eq.(31) as they are *computed* in the course of backward recursion for the optimal action and optimal Q-function.

Now, in the setting of batch-model RL, *neither* transitions probabilities, nor policy or reward functions are known. The required output is the same as before, i.e. the option price and hedge. However, the input is now richer than in the previous case, as now we have two more observations per each observed state vector X_t: the action a_t and the reward R_t. Instead of functions $\pi_t^\star(\cdot)$ and $R_t(\cdot)$, we now have only *samples* obtained from these functions in different states X_t.

This is the setting of Batch Reinforcement Learning, where the state variable values at both times t and $t+1$ along with rewards received for *all* MC paths are used *simultaneously* to find the optimal action and optimal Q-function at time t, for all states X_t and all actions $a_t(X_t)$. This amounts to learning a *policy*. With either simulated MC data or real data, batch-mode RL works the same way.

We use the same set of basis functions $\{\Phi_n(x)\}$ as we used above in Sect. 3.6 . As the optimal Q-function $Q_t^\star(X_t, a_t)$ is a quadratic function of a_t, we can represent it as an expansion in basis functions, with time-dependent coefficients parametrized by a matrix $\mathbf{W}_t$:

$$\begin{aligned} Q_t^\star(X_t, a_t) &= \left(1, a_t, \frac{1}{2}a_t^2\right) \begin{pmatrix} W_{11}(t) & W_{12}(t) & \cdots & W_{1M}(t) \\ W_{21}(t) & W_{22}(t) & \cdots & W_{2M}(t) \\ W_{31}(t) & W_{32}(t) & \cdots & W_{3M}(t) \end{pmatrix} \begin{pmatrix} \Phi_1(X_t) \\ \vdots \\ \Phi_M(X_t) \end{pmatrix} \\ &\equiv \mathbf{A}_t^T \mathbf{W}_t \mathbf{\Phi}(X_t) = \mathbf{A}_t^T \, \mathbf{U}_W(t, X_t) \end{aligned} \tag{64}$$

Here we defined

$$\mathbf{A}_t = \begin{pmatrix} 1 \\ a_t \\ \frac{1}{2}a_t^2 \end{pmatrix} \ , \ \ \mathbf{U}_W(t, X_t) \equiv \mathbf{W}_t \mathbf{\Phi}(X_t) \tag{65}$$

Terminal conditions for components of vector $\mathbf{U}_W(t, X_t)$ are obtained from Eq. (40):

$$\mathbf{U}_W^{(0)}(T, X_T) = -\Pi_T(X_T) - \lambda \, Var\left[\Pi_T(X_T)\right] \ , \ \ \mathbf{U}_W^{(1)}(T, X_T) = 0 \ , \ \ \mathbf{U}_W^{(2)}(T, X_T) = 0 \tag{66}$$

where Π_T is determined by the terminal condition (2).

Note that when the matrix of weights $\mathbf{W}_t$ is *known*, then both the optimal action and optimal Q-function with the optimal action are found easily in terms of components of vector $\mathbf{U}_W(t, X_t)$:

$$\begin{aligned} a_t^\star(X_t) &= -\frac{\mathbf{U}_W^{(1)}(t, X_t)}{\mathbf{U}_W^{(2)}(t, X_t)} \\ Q_t^\star(X_t, a_t^\star) &= \mathbf{U}_W^{(0)}(t, X_t) - \frac{1}{2}(a_t^\star)^2 \, \mathbf{U}_W^{(2)}(t, X_t) \end{aligned} \tag{67}$$

On the other hand, if instead the optimal action $a_t^\star(X_t)$ and optimal action-value function $Q_t^\star(X_t, a_t^\star)$ are known, then these relations could be read from the right to the left as two equations for three unknown functions $\mathbf{U}_W^{(k)}(t, X_t)$ where $k = 1, 2, 3$, such that the k-th function is parametrized by the k-th row of matrix $\mathbf{W_t}$. This suggests that a right solution, one that would be found by the DP Monte Carlo method of Sect. 3.6 if the dynamics are *known*, should be in the space of solutions parametrized in Eq.(64) by a time dependent matrix $\mathbf{W}_t$.

To learn the matrix $\mathbf{W}_t$, we follow a method to learn a matrix W for a bilinear Q-function of the form $Q(X, a) = X^T W a$ when X and a are multi-dimensional real-valued states values and actions, respectively, suggested in an elegant paper by Elkan [24]. Note that while Elkan uses the bilinear Q-function, in our case the Q-function is quadratic in action variables. But a basic simple trick is still the same.

The idea is to re-arrange terms in Eq.(64) to convert it into a product of a parameter vector and a vector that depends on both the state and the action:

$$\begin{aligned} Q_t^\star(x, a) &= \mathbf{A}_t^T \mathbf{W}_t \mathbf{\Phi}(X) = \sum_{i=1}^{3} \sum_{j=1}^{M} \left(\mathbf{W}_t \odot \left(\mathbf{A}_t \otimes \mathbf{\Phi}^T(X) \right) \right)_{ij} \\ &= \vec{\mathbf{W}}_t \cdot vec\left(\mathbf{A}_t \otimes \mathbf{\Phi}^T(X) \right) \equiv \vec{\mathbf{W}}_t \vec{\mathbf{\Psi}}(X_t, a_t) \end{aligned} \tag{68}$$

Here $\odot$ stands for an element-wise (Hadamard) product of two matrices. The vector of time-dependent parameters $\vec{\mathbf{W}}_t$ is obtained by concatenating columns of matrix $\mathbf{W_t}$, and similarly, $\vec{\mathbf{\Psi}}(X_t, a_t) = vec\left(\mathbf{A}_t \otimes \mathbf{\Phi}^T(X) \right)$ stands for a vector obtained by concatenating columns of the outer product of vectors $\mathbf{A}_t$ and $\mathbf{\Phi}(X)$.

The parameter vector $\vec{\mathbf{W}}_t$ therefore has $3M$ components, and vector $\vec{\mathbf{\Psi}}_t(X_t, a_t)$ has the same dimension, while the number of data records for a RL problem per time step is (59) is $3N$. This can be compared with the DP setting of Sect. 3.6 where we have $2M$ parameters defining the optimal policy and action value function in Eq.(49), but only N values of X_t as input data.

Coefficients $\mathbf{W_t}$ can now be computed recursively backward in time for $t = T-1, \ldots, 0$. To this end, the one-step Bellman optimality equation (39) is interpreted as regression of the form

$$R_t(X_t, a_t, X_{t+1}) + \gamma \max_{a_{t+1} \in \mathcal{A}} Q_{t+1}^\star(X_{t+1}, a_{t+1}) = \vec{\mathbf{W}}_t \vec{\mathbf{\Psi}}(X_t, a_t) + \varepsilon_t \tag{69}$$

where ε_t is a random noise at time t with mean zero, similar to Eq.(54). This is similar to Eq.(54) but this time we use it for arbitrary action a_t, not the optimal action $a_t^\star$, and in addition, use function approximation (64) to specify values of the optimal Q-function $Q_t^\star(X_t, a_t)$ for arbitary values of state-action pairs. Furthermore, while in regression of Eq.(54) we *computed* the reward $R_t(X_t, a_t^\star, X_{t+1})$ for the *optimal* action $a_t^\star$, in Eq.(69) we have *observed* rewards $R_t(X_t, a_t, X_{t+1})$. Again, (69) and (39) are equivalent in expectations, as taking the expectation of both sides of (69), we recover (39) with function approximation (64) used for the optimal Q-function $Q_t^\star(x, a)$.

Coefficients $\mathbf{W_t}$ are therefore found by solving the following least-square optimization problem:

$$\mathcal{L}_t(\mathbf{W}_t) = \sum_{k=1}^{N_{MC}} \left(R_t(X_t, a_t, X_{t+1}) + \gamma \max_{a_{t+1} \in \mathcal{A}} Q_{t+1}^\star(X_{t+1}, a_{t+1}) - \vec{\mathbf{W}}_t \vec{\mathbf{\Psi}}(X_t, a_t) \right)^2 \tag{70}$$

Note that this relation holds for a general *off-model*, *off-policy* setting of the Fitted Q Iteration method of RL.

Before proceeding to a solution to (70) in such general case, it is useful to consider first a special case of *on-policy* learning. In this case, we know that data always correspond to an

optimal action performed for a given market scenario, so the general *off-policy* problem (70) is replaced by an *on-policy* problem

$$\mathcal{L}_t^{(op)}\left(\mathbf{W}_t\right) = \sum_{k=1}^{N_{MC}} \left(R_t\left(X_t, a_t^\star, X_{t+1}\right) + \gamma Q_{t+1}^\star\left(X_{t+1}, a_{t+1}^\star\right) - \vec{\mathbf{W}}_t \vec{\mathbf{\Psi}}\left(X_t, a_t^\star\right)\right)^2 \tag{71}$$

Now compare this with Eq.(55). Note that these two equation becomes identical if the expression $\vec{\mathbf{W}}_t \vec{\mathbf{\Psi}}\left(X_t, a_t\right)$ for the optimal Q-function for *on-policy*, *model-free* learning corresponding to Eq.(71 is point-wise equal to the expression $\sum_n^M \omega_{nt} \Phi_n\left(X_t^k\right)$ that stands in Eq.(55) for an optimal *model-based* Q-function. Of course, this is only possible when a model (*any* model, in fact, because we did *not* specify it) implied in a DP solution of Sect. 3.6 is a true model of the world corresponding to data actually collected. For this case, we obtain

$$\lim_{N_{MC}\to\infty} Q_t^{\star, on-policy} = \lim_{N_{MC}\to\infty} Q_t^{\star, model} \tag{72}$$

Because our optimization problem (70) is convex and hence has a unique solution, and because Q-Learning is an *off-policy* algorithm, it means that a solution of Eq.(70) converges to the same limit given by the right-hand side of Eq.(72), if only a model used by the DP solution of Sect. 3.6 is the true model of the world. While these are general properties of Q-Learning, in our case they can be seen directly from equations, because the model is so simple.

The asymptotic relation (72) can also be re-stated in the case of a large number N_{MC} of Monte Carlo simulations as a least-square equality between the RL solution and the DP solution:

$$\mathcal{L}_t^{(DP)}\left(\mathbf{W}_t\right) = \sum_{k=1}^{N_{MC}} \left(\sum_n^M \omega_{nt} \Phi_n\left(X_t^k\right) - \vec{\mathbf{W}}_t \vec{\mathbf{\Psi}}\left(X_t, a_t\right)\right)^2 \tag{73}$$

Solving this for the vector $\vec{\mathbf{W}}_t$, we obtain

$$\vec{\mathbf{W}}_t = \left[\sum_{k=1}^{N_{MC}} \vec{\mathbf{\Psi}}\left(X_t^k, a_t^k\right) \vec{\mathbf{\Psi}}^T\left(X_t^k, a_t^k\right)\right]^{-1} \sum_{k=1}^{N_{MC}} \vec{\mathbf{\Psi}}\left(X_t^k, a_t^k\right) \sum_n^M \omega_{nt} \Phi_n\left(X_t^k\right) \tag{74}$$

This relation, which becomes exact in the limit $N_{MC} \to \infty$, allows one to express the solution for the RL method in terms of a solution of the DP method, assuming that the latter is known, and the model of the DP approach is the true model corresponding to data used by the RL approach. This corresponds to the *on-model*, *on-policy* special case for the general Q-Learning settting.

Coming back to a general *model-free* and *off-policy* Q-Learning given by Eq.(70), it is easy to solve, too. Similar to Eq.(55), the quadratic optimization problem of Eq.(70) is solved in terms of another pair of a matrix $\mathbf{S_t}$ and a vector $\mathbf{M_t}$ with elements

$$\begin{aligned} S_{nm}^{(t)} &= \sum_{k=1}^{N_{MC}} \Psi_n\left(X_t^k, a_t^k\right) \Psi_m\left(X_t^k, a_t^k\right) \\ M_n^{(t)} &= \sum_{k=1}^{N_{MC}} \Psi_n\left(X_t^k, a_t^k\right) \left(R_t\left(X_t, a_t, X_{t+1}\right) + \gamma \max_{a_{t+1}\in\mathcal{A}} Q_{t+1}^\star\left(X_{t+1}, a_{t+1}\right)\right) \end{aligned} \tag{75}$$

The optimal weights ω_t defining the optimal Q-function (69) at time t are then given by the following expression:

$$\mathbf{W}_t^\star = \mathbf{S}_t^{-1} \mathbf{M}_t \tag{76}$$

Equations (76), (75) and (69), computed recursively for $t = T-1, \ldots, 0$ provide a practical implementation of the backward recursion scheme of Sect. 3.5 when the dynamics are *unknown*, which is the setting of batch-mode Reinforcement Learning. Note that we can use Eq.(67) to perform the maximization step in the second equation in (75) analytically, because coefficients $\mathbf{W}_{t+1}$ and hence vectors $\mathbf{U}_W(t+1, X_{t+1}) \equiv \mathbf{W}_{t+1}\mathbf{\Phi}(X_{t+1})$ (see Eq.(65)) are known from the previous step.

Equation (76) gives the solution for the QLBS model in a *model-free* and *off-policy* setting. This is because in our model, the optimal option price is the negative of the optimal Q-function, while Eq.(76) that gives this price in terms of a simple recursive formula implements Fitted Q Iteration for our MDP problem, which *is* a model-free and off-policy algorithm [12, 13, 24].

This solution obliviates any need for model estimation/calibration. It even skips the problem of defining the dynamics of the world. The (in)famous "volatility smile problem" (see e.g. [5]) is solved too, because our approach is distribution-free and works directly with raw trading data. The only thing our model needs is a set of basis functions and little linear algebra (linear regression).

Computational complexity is the same as in other linear regression-based approaches such as the quadratic risk minimization of Föllmer and Schweizer [9, 10] or the Least Square Monte Carlo of Longstaff and Schwartz [11]. Only this time, our model does it *without making any assumptions on the true dynamics of the stock price*. Linear regressions in this model take actual *trading data* from a dynamic portfolio of a stock and a bank cash account, and convert it *directly* into the optimal price and optimal hedge for the option. Parameters found by these linear regressions are parameters defining the optimal price and optimal hedge strategy themselves, there are *no other parameters* beyond these parameters that serve as weights of basis functions.

Under the *only* modeling assumption that local rewards (negative losses) are *quadratic* a-lá Markowitz [8], Eq.(76) is a FQI-based *model-free* solution for the Bellman optimality equation (39) in a pure data-driven approach of batch-mode Reinforcement Learning. It only needs training data from an option trading desk in the form of tuples (59), plus basic linear algebra that can be implemented using GPU architectures, if a computational speed becomes critical. No volatility surface models or jump-diffusion models with integro-differential equations for option pricing are ever needed, and not even canonical $N(d_1)$ and $N(d_2)$ of the BSM model (see e.g. [5]) are nowhere to be seen. As long as rewards are quadratic, the Q-Learner goes *model-free* and gives the model-independent answer (76) that defines the optimal Q-function (68) and hence both the optimal hedge and optimal price.

As long the solution in our model is given explicitly by Eq.(76), it converges to the correct solution in the limit of asymptotically large trading data, for both discrete-state and continuous-state formulations of our model. Therefore, the Q-Learner of our model is *guaranteed* to compute the optimal hedge and price of the option directly from trading data, if only there is enough such data.

Most interestingly, actions in recorded data can be completely random, and the Q-Learner will still learn the right price and hedge of an option if there is enough *market data*. This is guaranteed by the property of Q-Learning and Fitted Q Iteration of being *off-policy* algorithms. This creates an ability to create artificial trading histories by combining real market data (stock prices) with some pre-defined trading strategy in stocks.

5 Possible numerical experiments

The main result of this paper is given by by a simple recursive relation Eq.(76), implementable via linear regression, that provides a way to price and hedge a stock option by learning *directly*

from the past trading data. Our model is backed by the celebrated Q-Learning method of Watkins [6, 7] and the Fitted Q Iteration method of Ernst *et. al.* [12] and Murphy [13].

As the latter methods are *model-free* and converging, and the explicit *model-free* solution of the model is provided by Eq.(76), it was the author's feeling that this also makes *him* free to concentrate in this paper on the mathematical solution of the model, and leave numerical examples and simple extensions for another publication.

As hedges in our model are the same as in the model of Grau [17], if we set $\mu = r$ to focus only on risk of an option seller, some numerical examples showing the pricing and hedging performance when the model is known can be found in a Ph.D. thesis of Dr. A. Grau [17].

6 Summary

In this paper we presented the QLBS model - a model for derivatives prices that is rooted in Reinforcement Learning. It was designed with an idea of building a model for derivatives pricing that would implement the principle of hedging first and pricing second in a consistent way for a discrete-time version of the classical Black–Scholes(-Merton) model. By definition, an optimal action-value Q-function [1] does such *hedging and pricing by learning*, if we have an option pricing model where the option price is the negative of a Q-function, and the option hedge is its second argument. Once we construct such a model, we can use Q-Learning, one of the most powerful algorithms of Reinforcement Learning, to learn both the optimal option price and hedge *directly from trading data* for a replicating portfolio of a stock and a bank cash account. Therefore, by stepping *aside* from the academic limit $\Delta t \to 0$, our model gains model independence, because it is solved by Q-Learning, and even gets somewhat simplified numerically, as an added bonus!

Author's original intention at the start of this project was to have a toy "BSM-like" simulated financial environment in a simple discrete-time/discrete-space setting, that would be used for pedagogical purposes to explain simple RL algorithms such as Q-Learning. We hope that, at the very least, the QLBS model did achieve this goal, and therefore could potentially be used, along with its straightforward extensions, for benchmarking different RL algorithms in a simulated financial environment. Due to a high degree of tractability of our model (linear algebra plus Monte Carlo simulation) and its extensibility, both the discrete-space and continuous-space versions of the model can be used to benchmark different RL algorithms including policy gradient methods, actor-critic algorithms, Deep Reinforcement Learning, and so on - all with known optimal policies and action-value functions easily computable by means of Dynamic Programming.

However, as the initial idea of this project, to have "something like the BSM" for RL in Finance, has been progressing, different elements of this scheme started to come together as a *real* financial model, especially after recognizing the fact that Q-Learning gives, by its definition, *both* the price and the hedge in one easily computable formula, if only the option price is defined properly. Given both the simplicity and generality of this emerging model, it started to appear that the Q-Learner was there all the time, but just on the side, keeping its distance $\Delta t > 0$ and quietly waiting to get noticed, while the vast majority[4] of continuous-time Mathematical Finance literature kept fighting demons of "risk-neutral" $\Delta t \to 0$ worlds (both within the BSM model itself and its many "risk-neutral" extensions). The model presented in this paper is in some way even simpler than the continuous-time BSM model which involves non-elementary functions such as cumulative normal distributions of composite arguments, which correspond to infinite series in terms of elementary functions. Our model involves only Linear Algebra and

[4]Except the literature on hedging and pricing in incomplete markets.

finite sums, and both hedge and price are contained in *one* formula, not two as in the BSM model, though the formula itself is not given as a closed-form expression.

If the famous Black-Scholes *formula* for the option price was somehow *not* known till today, it could be re-discovered by the QLBS model, by analyzing data collected with purely random strategies, combined with a Deep Neural Network to approximate the value function, in a limit of very small time steps. This is because Q-Learning is an *off-policy* method that can learn even from data with purely random actions. The only question is how much data and how many GPUs or TPUs are needed. On the other hand, *no knowledge* of Partial Differential Equations would be required with such approach! Experiments of this kind will possibly be presented elsewhere.

Besides mathematical simplicity, and probably more importantly, because Q-Learning is a model-free method, our approach can be used to price and hedge options *directly from data, and without any model at all*, going "the RL way" by extending this framework to a multi-dimensional case, and using more advanced versions of Q-Learning.

On the financial side, our model demonstrates that one very classical financial model (the continuous-time BSM model) can be obtained from another classical financial model (the Markowitz portfolio theory), if applied in a special dynamic setting of a BSM replicating portfolio under BSM lognormal dynamics, taken to the limit $\Delta t \to 0$. As is known since Föllmer and Schweizer [9, 10], this limit washes out any *risk*. But *markets only exist when there is risk to trade*. Therefore, the very last step of going to the limit $\Delta t \to 0$ in the classical BSM model, while making perfect sense *mathematically*, makes no sense *financially*. While this latter fact is *also* well known in the literature, our model is the first one that not only provides an easy *consistent* pricing and hedging alternative for a discrete-time case $\Delta t > 0$, but also does it in a data-driven and model independent way, effectively reducing *all* of option pricing to Reinforcement Learning.

The infamous volatility smile problem of the classical BSM model (see e.g. [5]) is solved *automatically*, due to Q-Learning and reliance on data instead of a model. It simply *does not exist* for the Q-Learner - it is just a name that humans gave to a specific pattern in data.

The QLBS model provides a *simultaneous and consistent* model-independent and RL-based hedging and pricing of a European option (the method is straightforward to extend to other options e.g. American, Asian, basket, etc.). This is unlike many other incomplete market models where a link between pricing and hedging is more *ad hoc*. The QLBS model extends the BSM model by adding three more quite intuitive parameters: the stock drift μ, the time step Δt, and the Markowitz risk-aversion parameter λ[5].

Unlike the classical BSM model rooted in Ito's calculus, our QLBS model is rooted in Dynamic Programming and Reinforcement Learning, and capitalizes on convergence results for Q-learning to establish its own convergence to the classical BSM model in the academic limit $\Delta t \to 0$ (if the world is lognormal), while producing corrections outside this limit, by finding them *directly from trading data* using Q-Learning algorithms. Like the Cheshire Cat from Carrol's "Alice's Adventures in Wonderland", the risk-averse Q-Learner of our model disappears in the strict limit $\Delta t \to 0$, leaving only its grin (i.e. the BS price and delta of the option) behind. As all risk is gone in this limit, and markets are *fake*, its services are no longer needed, and it leaves the degenerate $\Delta t \to 0$ BSM world, in its quest for more *purposeful*, rational, BSM-like or otherwise, *risky* worlds. We know that it will be just fine out there, because it is *model-free*!

[5]Note that in the natural sciences, models with four or five parameters are often able to capture even very complex dynamic systems [25].

References

[1] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction".

[2] https://github.com/openai/gym.

[3] F. Black and M. Scholes, "The Pricing of Options and Corporate Liabilities", Journal of Political Economy, Vol. 81(3), 637-654, 1973.

[4] R. Merton, "Theory of Rational Option Pricing", Bell Journal of Economics and Management Science, Vol.4(1), 141-183, 1974.

[5] P. Wilmott, *Derivatives: The Theory and Practive of Financial Engineering*, Wiley 1998.

[6] C.J. Watkins, *Learning from Delayed Rewards*. Ph.D. Thesis, Kings College, Cambridge, England, May 1989.

[7] C.J. Watkins and P. Dayan, "Q-Learning", Machine Learning, 8(3-4), 179-192, 1992.

[8] H. Markowitz, *Portfolio Selection: efficient diversification of investment*, John Wiley, 1959.

[9] H. Föllmer and M. Schweizer, "Hedging by Sequential Regression: an Introduction to the Mathematics of Option Trading", ASTIN *Bulletin* 18 , 147-160, 1989.

[10] M. Schweizer, "Variance-Optimal Hedging in Discrete Time", Mathematics of Operations Research, 20, 1-32, 1995.

[11] F.A. Longstaff and E.S. Schwartz, "Valuing American Options by Simulation - a Simple Least-Square Approach", The Review of Financial Studies, Vol. 14(1), 113-147, 2001.

[12] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-Based Batch Model Reinforcement Learning", Journal of Machine Learning Research, 6, 405-556, 2005.

[13] S.A. Murphy, "A Generalization Error for Q-Learning", Journal of Machine Learning Research, 6, 1073-1097, 2005.

[14] A. Cerný and J. Kallsen, "Hedging by Sequential Regression Revisited", Working paper, City University London and TU München, 2007.

[15] M. Potters, J.P. Bouchaud, and D. Sestovic, "Hedged Monte Carlo: Low Variance Derivative Pricing with Objective Probabilities", Physica A, vol. 289, 517-525, 2001.

[16] A.Petrelli et al, "Optimal Dynamic Hedging of Equity Options: Residual-Risks, Transaction-Costs, & Conditioning", *working paper*, 2010.

[17] A.J. Grau, "Applications of Least-Square Regressions to Pricing and Hedging of Financial Derivatives", PhD. thesis, Technische Universit"at München, 2007.

[18] J. Lim, "A Numerical Algorithm for Indifference Pricing in Incomplete Markets", working paper, University of Texas, 2005.

[19] J.C. Duan and J.G. Simonato, "American Option Pricing Under GARCH by a Markov Chain Approximation", Journal of Economic Dynamics and Control, Vol. 25, (2001), pp. 1689-1718.

[20] A. Gosavi, "Finite Horizon Markov Control with One-Step Variance Penalties", Conference Proceedings of the Allerton Conferences, Allerton, IL, 2010.

[21] R. Fonteneau, "Contributions to Batch Mode Reinforcement Learning", Ph.D. thesis, University of Li'ege, 2011.

[22] H. Robbins and S. Monro, "A Stochastic Approximation Method", Ann. Math. Statistics, 22, 400-407, 1951.

[23] A. Gosavi, "Solving Markov Decision Processes via Simulation", in *Handbook of Simulation Optimization*, Springer, 2014.

[24] C. Elkan, "Reinforcement Learning with a Bilinear Q Function", working paper (2011).

[25] "With four parameters I can fit an elephant, and with five I can make him wiggle his trunk." (John von Neumann).

Appendix A: Indifference Hedging and Pricing

In what comes to option hedging, the framework presented in Sect. 2 largely follows the Hedged Monte Carlo (HMC) method of Potters and Bouchaud [15], with modifications suggested by Grau [17]. This approach is based on a quadratic loss (utility) function. Here we outline an alternative approach to the pricing and hedging under market incompleteness.

We use the expected utility maximization approach with a specific choice of the exponential utility function $U(X) = -\exp(-\gamma X)$. As will be shown below, as long as the risk aversion parameter γ is small, the only optimization that is needed in this approach is convex optimization arising at the stage of finding the Minimum Entropy Martingale Measure (MEMM) $\mathbb{Q}$.

The latter optimization can still be computationally costly as it should be done at each node at each time slice. However, if the model for the underlying is simple enough (e.g. an arithmetic or geometric Brownian motion), the transition from the real measure $\mathbb{P}$ to a risk-neutral martingale measure $\mathbb{Q}$ is known analytically in closed form. In this case, assuming that we have already computed the risk-neutral measure and moreover we simulate from this measure, no additional numerical optimization is needed in the formalism to be presented below, as long as the risk aversion parameter γ is small. Instead, we use a low-order perturbative expansion in powers of γ to approximate all quantities of interest (i.e. hedges and indifference prices of options)[6]. In this case, the whole calculation to the zero order in γ proceeds as in the quadratic risk formulation, the difference being that we now compute with the MEMM $\mathbb{Q}$ rather than a real-world measure $\mathbb{P}$. In addition, to calculate first-order corrections $O(\gamma)$, all we need is to compute two additional $\mathbb{Q}$-expectations per node[7].

Exponential utility and duality

Here we provide a brief overview of a related method developed by Lim [18] in the setting of optimal option hedging and pricing with an exponential utility. For details, proofs, and references to original publications, please see Ref. [18].

[6] Note that the usual Minimum Entropy (MinEnt) model for option prices is recovered in this approach as the zero-order term of this expansion. In other words, MinEnt corresponds to a risk-neutral limit $\gamma \to 0$ of the result of expected utility maximization for the exponential utility.

[7] If desired, this scheme can be easily extended to an arbitrary finite order in γ. The cost of calculating each additional correction in expansion in powers of γ is again two additional $\mathbb{Q}$-expectations per node.

To price and hedge a contingent claim with payoff B, we consider two optimal investment problems, with and without the derivative B:

$$\begin{aligned} V(x,t) &= \sup_{u\in\mathcal{A}} \mathbb{E}^P\left[-e^{-\gamma X^{x,u}(T)}\Big| X^{x,u}(t)=x,\mathcal{F}_t\right] \\ V^B(x,t) &= \sup_{u\in\mathcal{A}} \mathbb{E}^P\left[-e^{-\gamma (X^{x,u}(T)-B)}\Big| X^{x,u}(t)=x,\mathcal{F}_t\right] \end{aligned} \tag{A.1}$$

Here $u=u_t$ is the investment strategy and x is the initial wealth at time t. We concentrate on the option price for the writer of the option.

The indifference price $h_t(B)$ for claim B is defined as an extra wealth that makes the option writer indifferent between writing and not writing the option under the optimal investment strategy. It is found as a solution to the equation

$$V(x,t) = V^B\left(x+h_t(B),t\right) \tag{A.2}$$

The key relation is the duality formula that replaces the real-measure maximization wrt strategies $u\in\mathcal{A}$ by maximization wrt equivalent martingale measures $\mathcal{M}_e$ of $\mathbb{P}$:

$$\sup_{u\in\mathcal{A}} \mathbb{E}^P\left[-e^{-\gamma (X^{x,u}(T)-B)}\Big|\mathcal{F}_t\right] = -e^{-\gamma x}\exp\left(\sup_{M\in\mathcal{M}_e}\left\{-H_t(M|P)+\gamma\mathbb{E}^M[B|\mathcal{F}_t]\right\}\right) \tag{A.3}$$

where $H_t(M|P)$ stands for the conditional entropy of measures M and P. Using (A.2) and (A.3), the indifference price reads

$$\begin{aligned} h_t(B) &= \frac{1}{\gamma}\left[\sup_{M\in\mathcal{M}_e}\left\{-H_t(M|P)+\mathbb{E}^M\left[\gamma B|\mathcal{F}_t\right]\right\} - \sup_{M\in\mathcal{M}_e}\left\{-H_t(M|P)\right\}\right] \\ &\equiv \frac{1}{\gamma}\left[v_t^{P,\gamma}(B)-v_t^P(0)\right] \end{aligned} \tag{A.4}$$

The indifference pricing therefore reduces to two functional optimizations for two terms in (A.4). These optimizations are done using the Lagrange multiplier method. Let $S_i=(S_{i1},\ldots,S_{id})$ be the node for the underlying at time t (here d is the number of components), $\mathcal{I}_t$ be the set of all nodes at can be reached starting from node i at t, and ΔS_{ij} is a vector of changes of the underlying from node i at t to node j at time $t+1$. We have

$$\begin{aligned} v^{P,\gamma}(t,i,B) &= \min_{\mu,\lambda}\max_{m}\left\{-\sum_{j\in\mathcal{I}_t} m_j\left[\log\frac{m_j}{p_j}-\gamma b_j\right]+\mu\left(\sum_{j\in\mathcal{I}_t} m_j-1\right)+\lambda_B^T\sum_{j\in\mathcal{I}_t} m_j\Delta S_{ij}\right\} \\ &= \min_{\lambda}\left[\log Z_{ti}(\lambda_B)\right] \end{aligned} \tag{A.5}$$

where $\lambda_B=(\lambda_B^{(1)},\ldots,\lambda_B^{(d)})$ is a vector of Lagrange multipliers, and

$$Z_{ti}=\sum_j p_j e^{\gamma b_j+\lambda_B^T\Delta S_{ij}} \tag{A.6}$$

The remaining minimization wrt Lagrange multipliers λ_B amounts to convex optimization in d dimensions. The second functional $v_t^P(0)$ is computed similarly with the substitution $b_j\to 0$, $\lambda_B\to\lambda_0$.

Note that Eqs.(A.6) and (A.4) indicate that the indifference price is non-linear in the payoff. This is in contrast to the linear pricing rule of the Minimum Entropy method. However, it

is easy to check that the latter is recovered from the indifference pricing method in the limit $\gamma \to 0$. This can be seen by expanding $v_t^{P,\gamma}$ to the first order around $v_t^{P,0}$ and then taking the limit $\gamma \to 0$ in (A.4), see the next section.

The optimal hedge ratios u_k $(k = 1, \ldots, d)$ are obtained as a rescaled difference of two optimal Lagrange multipliers calculated with and without the option:

$$u_k = -\frac{1}{\gamma}\left(\lambda_{B,k} - \lambda_{0,k}\right) \tag{A.7}$$

The above formulae refer to a single-period setting, and involve two Lagrangian optimizations with two set of Lagrange multipliers. A multi-period recursive version involving only the MEMM $\mathbb{Q}$ was worked out by Lim [18]:

$$h_t = \frac{1}{\gamma}\log \mathbb{E}^Q_{(t)}\left[e^{\gamma(h_{t+1} - u^T_{t+1}\Delta S_t)}\right] = u^T_{t+1}S_t + \frac{1}{\gamma}\log \mathbb{E}^Q_{(t)}\left[e^{\gamma(h_{t+1} - u^T_{t+1}S_{t+1})}\right] \tag{A.8}$$

for $t = 0, \ldots, T-1$ and $h_{T,j} = B_j$, where the optimal hedge is defined as follows

$$u_{t+1} = \arg\min_u \mathbb{E}^Q_{(t)}\left[e^{\gamma(h_{t+1} - u^T \Delta S_t)}\right] \tag{A.9}$$

Expansions for small γ

While equations (A.8) and (A.9) are valid for arbitrary values $\gamma \geq 0$, in practice their use can be costly for the calculation of optimal hedges, as the latter amounts to a transcendent equation that should be solved numerically at each node. Instead, we invoke a low-order expansion in γ in order to calculate approximate optimal hedges and option prices.

We start with the optimal hedge formula (A.9). We look for the optimal hedge ratio in terms of an expansion in powers of γ:

$$u_{t+1} = u^{(0)}_{t+1} + \gamma u^{(1)}_{t+1} + \gamma^2 u^{(2)}_{t+1} + \ldots \tag{A.10}$$

Plugging this into (A.9) and equating the like powers of γ, we find

$$\begin{aligned} u^{(0)}_{t+1} &= \frac{Cov^Q_t\left[h_{t+1}, \Delta S_t\right]}{Var^Q_t\left[\Delta S_t\right]} \\ u^{(1)}_{t+1} &= \frac{1}{2}\frac{\mathbb{E}^Q_t\left[\left(h_{t+1} - u^{(0)T}_{t+1}\Delta S_t\right)^2 \Delta S_t\right]}{Var^Q_t\left[\Delta S_t\right]} \end{aligned} \tag{A.11}$$

If desired, this expansion can be continued.

Next we turn to the pricing formula. It is convenient to introduce the following short notation:

$$\tilde{h}_{t+1} = h_{t+1} - u^{(0)T}_{t+1}\Delta S_t \;\;,\;\; \bar{\tilde{h}}_{t+1} = \mathbb{E}^Q_t\left[\tilde{h}_{t+1}\right] \tag{A.12}$$

Expanding the second term in (A.8) in powers of γ, we find

$$\begin{aligned} h_t &= \mathbb{E}^Q_t\left[\tilde{h}_{t+1}\right] + \frac{1}{2}\gamma Var^Q_t\left(\tilde{h}_{t+1}\right) + \frac{1}{3!}\gamma^2 \mathbb{E}^Q_t\left[\left(\tilde{h}_{t+1} - \bar{\tilde{h}}_{t+1}\right)^3\right] + \ldots \\ &= \mathbb{E}^Q_t\left[h_{t+1}\right] + \frac{1}{2}\gamma Var^Q_t\left(\tilde{h}_{t+1}\right) + \frac{1}{3!}\gamma^2 \mathbb{E}^Q_t\left[\left(\tilde{h}_{t+1} - \bar{\tilde{h}}_{t+1}\right)^3\right] + \ldots \end{aligned} \tag{A.13}$$

where in the second line we used the definition (A.12) and the fact that the Q-expectation of ΔS_t is zero.

Eq.(A.13) is an expansion in moments of the slippage distribution. Note that the first two terms of this formula are very similar to the recursive pricing formula used in the local quadratic risk formulation, except that the conditional expectation is calculated under the risk-neutral measure $\mathbb{Q}$ rather than the physical measure $\mathbb{P}$. Also note that the resulting expression is non-negative provided contributions from higher moments (the third and above) are small. Therefore, this approach solves the problem with negative option prices in the quadratic risk formulation[8].

Next consider correction terms. The leading correction is proportional to the variance of the slippage distribution, with the coefficient fixed by the risk aversion parameter γ. The next correction is proportional to the third moment, etc. Therefore, this approach gives support to the intuitive idea of Potters and Bouchaud [15] that a risk premium in the option price should be driven by the second moment of the P&L distribution. It is exactly what happens in our approach, though the measure is fixed to be $\mathbb{Q}$ rather than $\mathbb{P}$, and the coefficient is fixed in terms of γ.

Appendix B: Least Square Monte Carlo (LSM) for American Options

While the objective of the American Monte Carlo method of Longstaff and Schwartz [11] is altogether different from the problem addressed in this paper (a risk-neutral valuation of an American option vs a real-measure discrete-time hedging/pricing of a European option), a *mathematical* setting is similar. Both problems look for an optimal strategy and solve this problem by a backward recursion in a combination with a forward Monte Carlo simulation. Here we provide a brief outline of their method.

The main idea of the LSM approach of Longstaff and Schwartz [11] is to treat the backward-looking stage of the security evaluation as a regression problem formulated in a forward-looking manner which is more suited for a Monte Carlo (MC) setting. The starting point is the (backward-looking) Bellman equation, the most fundamental equation of the stochastic optimal control (otherwise known as stochastic optimization). For an American option on a financial underlying, control variable takes a simple binary form ("exercise" or "not exercise"). The Bellman equation for this particular case produces the continuation value $C_t(S_t)$ at time t as a function of the current underlying value S_t:

$$C_t(S_t) = \mathbb{E}\left[e^{-r\Delta t} \max\left(h_{t+\Delta t}(S_{t+\Delta t}), C_{t+\Delta t}(S_{t+\Delta t})\right) \middle| \mathcal{F}_t \right] \tag{B.1}$$

Here $h_\tau(S_\tau)$ is the option payoff at time τ. For example, for an American put option $h_\tau(S_\tau) = (K - S_\tau)^+$.

Note that for American options, the continuation value should be estimated as a *function* $C_t(x)$ of the value $x = X_t$, as long we we want to know whether it is larger or smaller than the intrinsic value $H(X_t)$ for a particular *realization* $X_t = x$ (obtained e.g. with Monte Carlo simulation) of the process X_t at time t. The problem is, of course, that each Monte Carlo path has exactly *one* value of X_t at time t. One way to estimate a function $C_t(S_t)$ is to use all Monte

[8] If a truncated expansion produces a negative option price due to, say, a negative third moment, this can always be fixed by either adding more moments to the expansion (A.13), or by doing a full-blown numerical hedge optimization, where option prices are guaranteed to stay positive.

Carlo paths, i.e. use the *cross-sectional information*. To this end, the one-step Bellman equation (B.1) is interpreted as regression of the form

$$\max\left(h_{t+\Delta t}(S_{t+\Delta t}), C_{t+\Delta t}(S_{t+\Delta t})\right) = e^{r\Delta t} C_t(S_t) + \varepsilon_t(S_t) \tag{B.2}$$

where $\varepsilon_t(S_t)$ is a random noise at time t with mean zero, which may in general depend on the underlying value S_t at that time. Clearly (B.2) and (B.1) are equivalent in expectations, as taking the expectation of both sides of (B.2), we recover (B.1). Next the unknown function $C_t(S_t)$ is expanded in a set of basis functions:

$$C_t(x) = \sum_n a_n(t)\phi_n(x) \tag{B.3}$$

for some particular choice of the basis $\{\phi_n(x)\}$, and the coefficients $a_n(t)$ are then calculated using the least-squared regression of $\max\left(h_{t+\Delta t}(S_{t+\Delta t}), C_{t+\Delta t}(S_{t+\Delta t})\right)$ on the value S_t of the underlying at time t across all Monte Carlo paths.

Appendix C: Markov Chain approximation for the BSM model

Here we present a simple method that approximates a continuous-time, continuous-space BSM dynamics by a discrete-time, discrete-space Markov Chain model, following an approach developed in a different context by Duan and Simonato [19].

The space discretization proceeds as follows. First we define a grid of N (where N is an odd number) discrete points in the X-space by the following relation:

$$p_i = \log S_0 + \frac{2i - N - 1}{N - 1} I_p, \quad i = 1, \ldots, N \tag{C.1}$$

Here $I_p = \delta(N)\sigma\sqrt{T\Delta t}$, where $\delta(N)$ is a scaling factor which is chosen to be $\delta(N) = 2 + \log\log N$ in Ref.[19]. With this specification, for any i, we have $p_i \in [p_1, p_N]$, with the end points $p_1 = \log S_0 - I_p$ and $p_N = \log S_0 + I_p$. In addition, assuming that N is an odd number, we have $p_{\frac{N+1}{2}} = \log S_0$.

Mid-points between these grid points define separations between different cells for the continuous variable X_t:

$$C_1 = (c_1, c_2)\,,\ C_i = [c_i, c_{i+1}),\ i = 2, \ldots, N \tag{C.2}$$

where

$$c_1 = -\infty,\ c_i = \frac{p_i + p_{i+1}}{2},\ i = 2, \ldots, N,\ c_{N+1} = +\infty \tag{C.3}$$

The probability p_{ij} of transition from cell i to cell j in a time step Δt is given by the following expression [19]:

$$p_{ij} = \mathcal{N}\left(\frac{c_{j+1} - p_i - \left(\mu - 0.5\sigma^2\right)\Delta t}{\sigma\sqrt{\Delta t}}\right) - \mathcal{N}\left(\frac{c_j - p_i - \left(\mu - 0.5\sigma^2\right)\Delta t}{\sigma\sqrt{\Delta t}}\right) \tag{C.4}$$

Here $\mathcal{N}(\cdot)$ stands for the cumulative normal distribution function.

A *time-homogeneous* discrete-time Markov Chain with transition probabilities (C.4), defined on a discretized set of states (C.1) for values of (time-stationary) variable X_t defined in Eq.(22), is our final *controllable* discrete-time/discrete-space approximation of the stock price dynamics in the BSM model, which will be used in the next section to build a corresponding finite-state MDP problem for optimal pricing and hedging of a European stock option.