

Policy gradient learning methods for stochastic control with exit time and applications to share repurchase pricing

Mohamed HAMDOUNE^{*} Pierre HENRY-LABORDERE[†] Huy  n PHAM[‡]

Abstract

We develop policy gradients methods for stochastic control with exit time in a model-free setting. We propose two types of algorithms for learning either directly the optimal policy or by learning alternately the value function (critic) and the optimal control (actor). The use of randomized policies is crucial for overcoming notably the issue related to the exit time in the gradient computation. We demonstrate the effectiveness of our approach by implementing our numerical schemes in the application to the problem of share repurchase pricing. Our results show that the proposed policy gradient methods outperform PDE or other neural networks techniques in a model-based setting. Furthermore, our algorithms are flexible enough to incorporate realistic market conditions like e.g. price impact or transaction costs.

1 Introduction

Let us consider a controlled Markov state process $X = (X^\alpha)_t$ valued in $\mathcal{X} \subset \mathbb{R}^d$ with a control process $\alpha = (\alpha_t)$ valued in $A \subset \mathbb{R}^m$. Given an open set \mathcal{O} of \mathcal{X} , we denote by $\tau = \tau^\alpha$ the exit time of the domain \mathcal{O} before a terminal horizon $T < \infty$, i.e.,

$$\tau = \inf\{t \geq 0 : X_t \notin \mathcal{O}\} \wedge T,$$

with the usual convention that $\inf \emptyset = \infty$. The objective is then to maximize over control process α a criterion in the form

$$J(\alpha) = \mathbb{E}[g(X_\tau^\alpha)], \quad \rightarrow \quad V_0 = \sup_{\alpha} J(\alpha), \quad (1.1)$$

for some terminal reward function g on \mathbb{R}^d . In typical examples, X is modelled by a controlled diffusion process as

$$dX_t = \mu(X_t, \alpha_t)dt + \sigma(X_t, \alpha_t)dW_t, \quad (1.2)$$

and we can also consider jump-diffusion processes, which is in particular relevant for insurance/reinsurance problem with minimization of the ruin probability in finite time.

^{*}LPSM, Universit   Paris Cit  , hamdouche at lpsm.paris

[†]Qube Research and Technologies, pierre.henrylabordere at qube-rt.com

[‡]LPSM, Universit   Paris Cit  , pham at lpsm.paris

Remark 1.1. Notice that there is no loss of generality to focus on the above Mayer form, as the case of Bolza criterion with running reward:

$$J(\alpha) = \mathbb{E}\left[\int_0^\tau f(X_t^\alpha, \alpha_t)dt + g(X_\tau^\alpha)\right],$$

can be reduced to the Mayer form by considering as usual the additional component $(Y_t)_t$ of the state process, driven by

$$dY_t = f(X_t, \alpha_t)dt,$$

and the corresponding terminal reward function $\tilde{g}(x, y) = y + g(x)$.

The control problem (1.1) with exit time can be solved in a model-based setting, e.g. when the coefficients μ, σ in (1.2), and the analytical form of g are known, by PDE methods with splitting scheme as described in appendix B, and eventually by backward SDE methods, see [4]. In the last years, there has been an important literature about the use of deep learning techniques in the numerical resolution of stochastic control problems and PDEs, which have shown success for notably overcoming the curse of dimensionality, and we refer the reader to the recent surveys by [3] and [6]. However, these methods do not work well for our class of control problem with exit time. Indeed, for example, when trying to apply the global method of [9] by approximating the policy by a neural network with parameters θ , the differentiation of the associated gain function would lead to a Dirac function due to the presence of an indicator function related to the exit time, hence the gradient is ill-posed, which prevents an efficient implementation of the stochastic gradient ascent algorithm.

In this paper, we propose two types of algorithms based on reinforcement learning for estimating the solution to the control problem (1.1) in a model-free setting, i.e., without *a priori* knowledge of the model coefficients. We develop policy gradient methods for learning approximate optimal control and value function based on samples of state and rewards. A key feature is to consider parametrized randomized policies, notably for overcoming the issue related to exit time in the policy gradient representation. Our first algorithm learns directly the optimal policy, while the second type of algorithm is of actor-critic nature by learning alternately the policy and the value function. This can be done either in an offline setting with updates rules based on the whole trajectories of the state, or in an online setting with update rules in real-time incrementally. Our algorithms can be viewed as extensions to controlled processes with exit time of policy gradients in reinforcement learning usually designed for infinite or finite horizon, see [12].

The main application that we develop in this paper for stochastic control in the form (1.1) concerns the pricing of buyback options in Stock Repurchase Programs (in short SRPs). Those are defined as transactions initiated by companies to re-buy their proper stocks for various reasons including the raising of the debt-to-equity ratio or the improvement of earnings per share by reducing the number of outstanding shares. SRPs are also an alternative way to distribute the dividends to the shareholders, see [11]. For more details about SRPs and its regulatory issues and associated tools, the reader can consult this report [1].

There exist several mechanisms for SRPs with complex contracts involving investment banks, where the company mandates a bank to repurchase its shares through a derivative product. A well-known example often used by practitioners is Accelerated Share Repurchases (ASRs), where at time $t = 0$ the bank borrows a quantity B of shares required by the company from shareholders, and then purchases progressively from the open market the quantity B to give it back to shareholders. In addition, the bank becomes in a long position of an American option where at some exercise time τ , the company should pay the bank the average price between 0 and τ for each share.

The valuation of ASRs has recently attracted attention in the literature. Guéant et al. [8] consider in a discrete time/space model the pricing of ASRs, which leads to a tree based algorithm. Jaimungal et al. [10] investigate the same problem in continuous time/space setting by additionally taking into consideration temporary and long-term market impact, and characterize the execution frontier. Guéant et al. [7] use deep learning algorithms in the spirit of [5] and [2] for the pricing of ASRs contracts and buyback contract called VWAP-minus profit-sharing. In such contract, the exercise time τ is chosen by the bank once the amount of shares requested by the company is redeemed. In this paper, we consider a buyback contract where the exercise time τ is entirely characterized by the execution strategy and can not be chosen by any party. We shall call such a buyback product as Barrier VWAP-minus. Actually, one can show (see Appendix A) that in absence of market impact, the price of the Barrier VWAP-minus is equal to the price of the VWAP-minus.

The pricing of barrier VWAP-minus leads to a stochastic control formulation as in (1.1) where the exit time is defined as the first stopping time when the controlled inventory exceeds the quantity of shares to be purchased by the bank within a finite time interval. We implement our algorithms to this pricing problem: since they are model-free, they are robust to model misspecifications, and are valid notably for general model for the stock price including market impact and transaction costs.

We first compare our numerical results with those obtained by PDE methods with splitting scheme as detailed in Appendix B. Our validation test consists in approximating the optimal policy and then computing the price using Monte Carlo: it provides then by definition a lower bound to the true price of the constrained VWAP-minus contract. We show that our model-free policy gradient algorithms yield accurate results similar to PDE schemes designed in a specific model-based setting. It is also less costly and more stable than methods performed in [7] in a model-based setting, where the control and the stopping time are parametrized by two distinct neural networks. Moreover, it has the advantage to be easily implemented in general factor models including market impact. We illustrate notably the impact of market impact on the optimal trading policies.

The rest of the paper is structured as follows. We develop in Section 2 the policy gradient approach with randomized policies, and present our two types of algorithms. Section 3 is devoted to the application to valuation of SRP, including the case with market impact and transaction costs, with numerical results illustrating the convergence and accuracy of our algorithms, and comparison with other methods.

2 Policy gradient methods

We consider a time discretization of the stochastic control problem (1.1). Let $\mathbb{T} = \{t_0 = 0 < \dots < t_i < \dots < t_N = T\}$ be a subdivision of $[0, T]$ of size N with time steps $\Delta t_i = t_{i+1} - t_i$, $i = 0, \dots, N-1$. By misuse of notation, we denote by $(X_{t_i})_{i \in \llbracket 0, N \rrbracket}$ the Markov decision process (MDP) arising from the time discretization of the controlled state process $(X_t)_t$, and it is characterized by an initial distribution p_0 for X_{t_0} , and the transition kernel function $p(\cdot | t_i, x_i, a)$ representing the probability of the next state $X_{t_{i+1}}$ given the current state $X_{t_i} = x_i \in \mathcal{X}$, and an action $a \in A$ at time t_i . Notice that in a model-free setting, this transition kernel is unknown.

A randomized policy in this discretized time setting is a measurable transition kernel function $\pi : (t_i, x_i) \in \mathbb{T} \times \mathcal{X} \mapsto \pi(\cdot | t_i, x_i) \in \mathcal{P}(A)$ (the set of probability measures on A), and we say that $\alpha = (\alpha_{t_i})_{i \in \llbracket 0, N-1 \rrbracket}$ is a randomized feedback control generated from the stochastic policy π , written as $\alpha \sim \pi$, when α_{t_i} is drawn from $\pi(\cdot | t_i, X_{t_i})$ at any time t_i .

The exit time of the Markov decision process $(X_{t_i})_{i \in \llbracket 0, N \rrbracket}$ is given by

$$\tau = \inf\{t_i \in \mathbb{T} : X_{t_i} \notin \mathcal{O}\} \wedge t_N,$$

and the gain functional associated to the Markov decision process with exit time and randomized feedback control $\alpha \sim \pi$ is given by

$$J(\pi) = \mathbb{E}_{\alpha \sim \pi}[g(X_\tau)].$$

Here the notation $\mathbb{E}_{\alpha \sim \pi}[\cdot]$ means that the expectation is taken when the Markov decision process (X_{t_i}) is controlled by the randomized feedback control α generated from the stochastic policy π .

We now consider stochastic policies $\pi = \pi_\theta$ with parameters $\theta \in \mathbb{R}^D$, and which admit densities with respect to some measure ν on A : $\pi_\theta(\mathrm{d}a | t_i, x_i) = \rho_\theta(t_i, x_i, a)\nu(\mathrm{d}a)$, for some parametrized measurable functions $\rho_\theta : \mathbb{T} \times \mathcal{X} \times A \rightarrow (0, \infty)$.

- when A is a finite space, say $A = \{a_1, \dots, a_M\}$, we take ν as the counting measure, and choose softmax policies, i.e.,

$$\rho_\theta(t_i, x_i, a_m) = \frac{\exp(\phi_{\theta_m}(t_i, x_i))}{\sum_{\ell=1}^M \exp(\phi_{\theta_\ell}(t_i, x_i))}, \quad m = 1, \dots, M, \quad (2.1)$$

where ϕ_{θ_m} are neural networks on $[0, T] \times \mathbb{R}^d$, and $\theta = (\theta_1, \dots, \theta_M)$ gathers all the parameters of the M neural networks. In this case, the score function is given by

$$\nabla_{\theta_\ell} \log \rho_\theta(t_i, x_i, a_m) = (\delta_{m\ell} - \rho_\theta(t_i, x_i, a_\ell)) \nabla_{\theta_\ell} \phi_{\theta_\ell}(t_i, x_i).$$

- when A is a continuous space of \mathbb{R}^m , we can choose typically a Gaussian distribution on \mathbb{R}^m for the stochastic policy, with mean parametrized by neural network $\mu_\theta(t, x)$ valued on A , and variance a positive definite matrix Σ on $\mathbb{R}^{m \times m}$ to encourage

exploration, e.g. $\Sigma = \varepsilon I_m$. In this case, ν is the Lebesgue measure on \mathbb{R}^m , and the density is

$$\rho_\theta(t_i, x_i, a) = \frac{1}{(2\pi)^{m/2} \det(\Sigma)^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(a - \mu_\theta(t_i, x_i))^\top \Sigma^{-1}(a - \mu_\theta(t_i, x_i))\right).$$

In this case, the score function is given by

$$\nabla_\theta \log \rho_\theta(t_i, x_i, a) = \nabla_\theta \mu_\theta(t_i, x_i)^\top \Sigma^{-1}(a - \mu_\theta(t_i, x_i)).$$

We then denote, by abuse of notation, $J(\theta) = J(\pi_\theta)$, the performance function viewed as a function of the parameter θ of the stochastic policy, and the principle of policy gradient method is to maximize over θ this function by stochastic gradient ascent algorithm. In a model-free setting, the purpose is then to derive a suitable expectation representation of the gradient function $\nabla_\theta J(\theta)$ that does not involve unknown model coefficients and transition kernel $p(\cdot|t, x, a)$ of the state process, but only sample observations of the states X_{t_i} , $i = 0, \dots, N$, hence of the exit time τ , when taking decisions $\alpha \sim \pi_\theta$, with known chosen family of densities ρ_θ .

2.1 Policy gradient representation

Our first main result is to provide a stochastic policy gradient representation for the performance function J by adapting arguments in the infinite or finite horizon case.

Theorem 2.1. *We have*

$$\nabla_\theta J(\theta) = \mathbb{E}_{\alpha \sim \pi_\theta} \left[g(X_\tau) \sum_{i=0}^{N-1} \nabla_\theta \log \rho_\theta(t_i, X_{t_i}, \alpha_{t_i}) 1_{t_i < \tau} \right]. \quad (2.2)$$

Proof. For a path $(x_0, \dots, x_N) \in \mathcal{X}^{N+1}$, we denote by

$$\iota(x_0, \dots, x_N) = \inf\{i \in \llbracket 0, N \rrbracket : x_i \notin \mathcal{O}\} \wedge N,$$

so that the exit time of $(X_{t_i})_{i \in \llbracket 0, N \rrbracket}$ is written as $\tau = t_{\iota(x_0, \dots, x_N)}$. Let us then introduce the function G defined on \mathcal{X}^{N+1} by $G(x_0, \dots, x_N) = g(x_{\iota(x_0, \dots, x_N)})$, so that

$$\begin{aligned} J(\theta) &= \mathbb{E}_{\alpha \sim \pi_\theta} [G(X_{t_0}, \dots, X_{t_N})] \\ &= \int_{\mathcal{X}^{N+1}} \int_{A^N} G(x_0, \dots, x_N) p_0(dx_0) \prod_{i=0}^{N-1} \pi_\theta(da_i|t_i, x_i) p(dx_{i+1}|t_i, x_i, a_i) \\ &= \int_{\mathcal{X}^{N+1}} \int_{A^N} G(\mathbf{x}) p_0(dx_0) \boldsymbol{\rho}_\theta^N(\mathbf{x}, \mathbf{a}) \prod_{i=0}^{N-1} p(dx_{i+1}|t_i, x_i, a_i) \nu(da_i), \end{aligned} \quad (2.3)$$

where we set $\mathbf{x} = (x_0, \dots, x_N)$, $\mathbf{a} = (a_0, \dots, a_{N-1})$, and

$$\boldsymbol{\rho}_\theta^N(\mathbf{x}, \mathbf{a}) = \prod_{i=0}^{N-1} \rho_\theta(t_i, x_i, a_i).$$

By using the classical log-likelihood trick: $\nabla_{\theta} \rho_{\theta}^N(\mathbf{x}, \mathbf{a}) = (\nabla_{\theta} \log \rho_{\theta}^N(\mathbf{x}, \mathbf{a})) \rho_{\theta}^N(\mathbf{x}, \mathbf{a})$, and noting that

$$\nabla_{\theta} \log \rho_{\theta}^N(\mathbf{x}, \mathbf{a}) = \sum_{i=0}^{N-1} \nabla_{\theta} \log \rho_{\theta}(t_i, x_i, a_i),$$

we deduce by differentiating (2.3) that

$$\begin{aligned} \nabla_{\theta} J(\theta) &= \int_{\mathcal{X}^{N+1}} \int_{A^N} G(\mathbf{x}) \nabla_{\theta} \log \rho_{\theta}^N(\mathbf{x}, \mathbf{a}) p_0(dx_0) \prod_{i=0}^{N-1} \pi_{\theta}(da_i | t_i, x_i) p(dx_{i+1} | t_i, x_i, a_i) \\ &= \mathbb{E}_{\alpha \sim \pi_{\theta}} \left[G(X_{t_0}, \dots, X_{t_N}) \sum_{i=0}^{N-1} \nabla_{\theta} \log \rho_{\theta}(t_i, X_{t_i}, \alpha_{t_i}) \right]. \end{aligned}$$

Finally, observe that for any $i \in \llbracket 0, N-1 \rrbracket$, we have

$$\begin{aligned} &\mathbb{E}_{\alpha \sim \pi_{\theta}} \left[G(X_{t_0}, \dots, X_{t_N}) 1_{t_i \geq \tau} \nabla_{\theta} \log \rho_{\theta}(t_i, X_{t_i}, \alpha_{t_i}) \right] \\ &= \mathbb{E}_{\alpha \sim \pi_{\theta}} \left[g(X_{\tau}) 1_{t_i \geq \tau} \nabla_{\theta} \log \rho_{\theta}(t_i, X_{t_i}, \alpha_{t_i}) \right] \\ &= \mathbb{E}_{\alpha \sim \pi_{\theta}} \left[g(X_{\tau}) 1_{t_i \geq \tau} \mathbb{E}_{\alpha \sim \pi_{\theta}} [\nabla_{\theta} \log \rho_{\theta}(t_i, X_{t_i}, \alpha_{t_i}) | X_{t_i}] \right] \\ &= \mathbb{E}_{\alpha \sim \pi_{\theta}} \left[g(X_{\tau}) 1_{t_i \geq \tau} \underbrace{\nabla_{\theta} \left(\int_A \rho_{\theta}(t_i, X_{t_i}, a) \nu(da) \right)}_{=0} \right] = 0, \end{aligned} \tag{2.4}$$

which yields the required result. \square

Alternately, we now provide a second representation formula for the gradient of the performance function by exploiting the dynamic programming. Let us introduce the dynamic version of J . For $i \in \llbracket 0, N \rrbracket$, and $x \in \mathcal{X}$, we define the value (performance) function associated to the policy π_{θ}

$$V_i^{\theta}(x) := \mathbb{E}_{\alpha \sim \pi_{\theta}} [g(X_{\tau_i}) | X_{t_i} = x],$$

where $\tau_i = \inf\{t_j \in \mathbb{T}, t_j \geq t_i : X_{t_j} \notin \mathcal{O}\} \wedge t_N$, so that $J(\theta) = \mathbb{E}[V_0^{\theta}(X_0)]$. We notice that $V_N^{\theta}(x) = g(x)$, for all $x \in \mathcal{X}$, and $V_i^{\theta}(x) = g(x)$, for all $i \in \llbracket 0, N-1 \rrbracket$, and $x \notin \mathcal{O}$. Moreover, by the dynamic programming (which is here simply reduced to the law of conditional expectations), we have for $i \in \llbracket 0, N-1 \rrbracket$:

$$V_i^{\theta}(x) = \mathbb{E}_{\alpha \sim \pi_{\theta}} [V_{i+1}^{\theta}(X_{t_{i+1}}) | X_{t_i} = x], \quad \text{for } x \in \mathcal{O}. \tag{2.5}$$

Theorem 2.2. *We have*

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\alpha \sim \pi_{\theta}} \left[\sum_{i=0}^{N-1} V_{i+1}^{\theta}(X_{t_{i+1}}) \nabla_{\theta} \log \rho_{\theta}(t_i, X_{t_i}, \alpha_{t_i}) 1_{t_i < \tau} \right]. \tag{2.6}$$

Proof. From (2.5), we have for $(i, x_i) \in \llbracket 0, N-1 \rrbracket \times \mathcal{O}$

$$V_i^\theta(x_i) = \int_{\mathcal{X}} \int_A V_{i+1}^\theta(x_{i+1}) \rho_\theta(t_i, x, a) \nu(da) p(dx_{i+1}|t_i, x_i, a).$$

By differentiating with respect to θ , and using again the log-likelihood trick, we get

$$\begin{aligned} \nabla_\theta V_i^\theta(x_i) &= \int_{\mathcal{X}} \int_A \nabla_\theta [V_{i+1}^\theta(x_{i+1})] \rho_\theta(t_i, x_i, a) \nu(da) p(dx_{i+1}|t_i, x_i, a) \\ &\quad + \int_{\mathcal{X}} \int_A V_{i+1}^\theta(x_{i+1}) \nabla_\theta [\log \rho_\theta(t_i, x_i, a)] \rho_\theta(t_i, x_i, a) \nu(da) p(dx_{i+1}|t_i, x_i, a) \\ &= \int_{\mathcal{O}} \int_A \nabla_\theta [V_{i+1}^\theta(x_{i+1})] \pi_\theta(da|t_i, x_i) p(dx_{i+1}|t_i, x_i, a) \\ &\quad + \mathbb{E}_{\alpha \sim \pi_\theta} \left[V_{i+1}^\theta(X_{t_{i+1}}) \nabla_\theta \log \rho_\theta(t_i, X_{t_i}, \alpha_{t_i}) | X_{t_i} = x_i \right], \quad i \in \llbracket 0, N-1 \rrbracket, \end{aligned}$$

for all $x_i \in \mathcal{O}$, by noting that $\nabla_\theta V_{i+1}^\theta(x) = 0$ for $x \notin \mathcal{O}$, and $V_{i+1}^\theta(x) = V_{i+1}^\theta(x)$ for $x \in \mathcal{O}$. By iterating over i , and noting that $\nabla_\theta V_N^\theta(\cdot) \equiv 0$, we deduce that for all $x_0 \in \mathcal{O}$

$$\nabla_\theta V_0^\theta(x_0) = \mathbb{E}_{\alpha \sim \pi_\theta} \left[\sum_{i=0}^{N-1} V_{i+1}^\theta(X_{t_{i+1}}) \nabla_\theta \log \rho_\theta(t_i, X_{t_i}, \alpha_{t_i}) \prod_{j=1}^i 1_{X_{t_j} \in \mathcal{O}} | X_{t_0} = x_0 \right]$$

Since $\prod_{j=1}^i 1_{X_{t_j} \in \mathcal{O}} = 1_{t_i < \tau}$ and $\nabla_\theta V_0^\theta(\cdot) = 0$ on $\mathcal{X} \setminus \mathcal{O}$, we get the required representation formula. \square

Remark 2.3. *It is known that stochastic gradient policy algorithms suffer from high variance, and a good alternative is to use a baseline. For instance, in the representation (2.6), we can subtract to $V_{i+1}^\theta(X_{t_{i+1}})$ the term $V_i^\theta(X_{t_i})$ without biasing the gradient, i.e.*

$$\nabla_\theta J(\theta) = \mathbb{E}_{\alpha \sim \pi_\theta} \left[\sum_{i=0}^{N-1} (V_{i+1}^\theta(X_{t_{i+1}}) - V_i^\theta(X_{t_i})) \nabla_\theta \log \rho_\theta(t_i, X_{t_i}, \alpha_{t_i}) 1_{t_i < \tau} \right], \quad (2.7)$$

by the same trick as in (2.4):

$$\begin{aligned} &\mathbb{E}_{\alpha \sim \pi_\theta} \left[V_i^\theta(X_{t_i}) \nabla_\theta \log \rho_\theta(t_i, X_{t_i}, \alpha_{t_i}) 1_{t_i < \tau} \right] \\ &= \mathbb{E}_{\alpha \sim \pi_\theta} \left[V_i^\theta(X_{t_i}) 1_{t_i < \tau} \mathbb{E}_{\alpha \sim \pi_\theta} [\nabla_\theta \log \rho_\theta(t_i, X_{t_i}, \alpha_{t_i}) | X_{t_i}] \right] \\ &= \mathbb{E}_{\alpha \sim \pi_\theta} \left[V_i^\theta(X_{t_i}) 1_{t_i < \tau} \underbrace{\nabla_\theta \left(\int_A \rho_\theta(t_i, X_{t_i}, a) \nu(da) \right)}_{=0} \right] = 0. \end{aligned}$$

2.2 Algorithms

We now propose policy gradient algorithms which are based on the representation of the previous section. They do not require necessarily the knowledge of model coefficients

and transition kernel $p(\cdot|t, x, a)$ of the state process, but only sample observations of the states X_{t_i} , $i = 0, \dots, N$, when taking decisions α according to the chosen family of randomized policies, via e.g. an environment simulator (blackbox), hence of the exit time τ . They do neither require the knowledge of the analytical form of the reward function g , and instead, we can consider that given an input/observation of a state x , the associated output/reward $g(x)$ is evaluated via e.g. a blackbox simulator.

Our first algorithm (see pseudo-code in Algorithm 1) is based on the gradient representation (2.2).

Algorithm 1: Stochastic gradient policy

Input data: Number of episodes E , mini-batch size K , learning rate η for policy gradient estimation; Parametrized family of randomized policies π_θ with densities ρ_θ ;

Initialization: parameter θ ;

for each episode $e = 1, \dots, E$ **do**

 select a random path $k = 1, \dots, K$;

 Initialize state $X_0^{(k)} \in \mathcal{O}$;

for $i = 0, \dots, N - 1$ **do**

 Generate action $\alpha_{t_i}^{(k)} \sim \pi_\theta(\cdot|t_i, X_{t_i}^{(k)})$

 Simulate by a model or observe (e.g. by blackbox) state $X_{t_{i+1}}^{(k)}$

 If $X_{t_{i+1}}^{(k)} \notin \mathcal{O}$ or $t_{i+1} = T$, store the exit time $\tau^{(k)} = t_{i+1}$, compute or

 observe by blackbox $G^{(k)} := g(X_{\tau^{(k)}}^{(k)})$, and close the loop;

 Otherwise $i \leftarrow i + 1$;

end

 Compute for path k

$$\Gamma_\theta^{(k)} := G^{(k)} \sum_{t_i < \tau^{(k)}} \nabla_\theta \log \rho_\theta(t_i, X_{t_i}^{(k)}, \alpha_{t_i}^{(k)})$$

 Update parameters of the policies: $\theta \leftarrow \theta + \eta \frac{1}{K} \sum_{k=1}^K \Gamma_\theta^{(k)}$;

end

Return: π_θ

Our second type of algorithm is based on the gradient representation (2.7), and is of actor-critic type: it consists in estimating simultaneously via fixed-point iterations the randomized optimal policy (the actor) by policy gradient (PG), and the value function (critic) by performance evaluation relying on the martingale property relation (2.5). More precisely, in addition to the parametrized family π_θ of randomized policies, we are given a family of functions \mathcal{V}_ϕ on $[0, T] \times \mathcal{X}$, with parameter ϕ , e.g. neural network, aiming to approximate the value function. The parameters (θ, ϕ) are then updated alternately as follows: given a current estimation $(\theta^{(n)}, \phi^{(n)})$, the parameter θ is updated

according to the PG (2.7) by replacing V by $\mathcal{V}_{\phi^{(n)}}$:

$$\theta^{(n+1)} = \theta^{(n)} + \eta \mathbb{E}_{\alpha \sim \pi_{\theta^{(n)}}} \left[\sum_{t_i < \tau} (\mathcal{V}_{\phi^{(n)}}(t_{i+1}, X_{t_{i+1}}) - \mathcal{V}_{\phi^{(n)}}(t_i, X_{t_i})) \nabla_{\theta} \log \rho_{\theta^{(n)}}(t_i, X_{t_i}, \alpha_{t_i}) \right]$$

while ϕ is updated by minimizing the square regression error:

$$\mathbb{E} \left[\left| \mathcal{V}_{\phi^{(n)}}(t_{i+1}, X_{t_{i+1}}) - \mathcal{V}_{\phi}(t_i, X_{t_i}) \right|^2 1_{X_{t_i} \in \mathcal{O}} \right].$$

Notice that we only need to learn the value function on the domain \mathcal{O} by sampling the state process until the exit time τ , as it is extended on $\mathcal{X} \setminus \mathcal{O}$ by the reward g .

The pseudo-code of our Actor-Critic algorithm is described in Algorithm 2.

Algorithm 2: Actor-Critic (offline)

Input data: Number of episodes E , mini-batch size K , learning rates η^G, η^V for policy and value function estimation; Parametrized family π_{θ} with densities ρ_{θ} for randomized policies, and \mathcal{V}_{ϕ} for value function;

Initialization: parameter θ, ϕ ;

for each episode $e = 1, \dots, E$ **do**

select a random path $k = 1, \dots, K$;

Initialize state $X_0^{(k)} \in \mathcal{O}$;

for $i = 0, \dots, N-1$ **do**

Generate action $\alpha_{t_i}^{(k)} \sim \pi_{\theta}(\cdot | t_i, X_{t_i}^{(k)})$

Simulate by a model or observe (e.g. by blackbox) state $X_{t_{i+1}}^{(k)}$

If $X_{t_{i+1}}^{(k)} \notin \mathcal{O}$ or $t_{i+1} = T$, set $\tau^{(k)} = t_{i+1}$, $\mathcal{V}_{\phi}(t_{i+1}, X_{t_{i+1}}^{(k)}) = g(X_{t_{i+1}}^{(k)})$ computed e.g. by blackbox, and close the loop;

Otherwise $i \leftarrow i + 1$;

end

Compute for path k

$$\Gamma_{\theta}^{(k)} := \sum_{t_i < \tau^{(k)}} (\mathcal{V}_{\phi}(t_{i+1}, X_{t_{i+1}}^{(k)}) - \mathcal{V}_{\phi}(t_i, X_{t_i}^{(k)})) \nabla_{\theta} \log \rho_{\theta}(t_i, X_{t_i}^{(k)}, \alpha_{t_i}^{(k)})$$

$$\Delta_{\phi}^{(k)} := \sum_{t_i < \tau^{(k)}} (\mathcal{V}_{\phi}(t_{i+1}, X_{t_{i+1}}^{(k)}) - \mathcal{V}_{\phi}(t_i, X_{t_i}^{(k)})) \nabla_{\phi} \mathcal{V}_{\phi}(t_i, X_{t_i}^{(k)})$$

Actor update: $\theta \leftarrow \theta + \eta^G \frac{1}{K} \sum_{k=1}^K \Gamma_{\theta}^{(k)}$;

Critic update: $\phi \leftarrow \phi + \eta^V \frac{1}{K} \sum_{k=1}^K \Delta_{\phi}^{(k)}$;

end

Return: $\pi_{\theta}, \mathcal{V}_{\phi}$.

In the above actor-critic algorithm, the parameters are updated once the whole state trajectories are sampled. We can design an online version where the parameters are updated in real-time incrementally, see pseudo-code in Algorithm 3.

Algorithm 3: Actor-Critic (online)

Input data: Number of episodes E , mini-batch size K , learning rates η^G, η^V for policy and value function estimation; Parametrized family π_θ with densities ρ_θ for randomized policies, and \mathcal{V}_ϕ for value function;

Initialization: parameter θ, ϕ ;

for each episode $e = 1, \dots, E$ **do**

- select a random path $k = 1, \dots, K$;
- Initialize state $X_0^{(k)} \in \mathcal{O}$;
- for** $i = 0, \dots, N - 1$ **do**

 - Generate action $\alpha_{t_i}^{(k)} \sim \pi_\theta(\cdot | t_i, X_{t_i}^{(k)})$
 - Simulate by a model or observe (e.g. by blackbox) state $X_{t_{i+1}}^{(k)}$
 - If $X_{t_{i+1}}^{(k)} \notin \mathcal{O}$ or $t_{i+1} = T$, set $\tau^{(k)} = t_{i+1}$, $\mathcal{V}_\phi(t_{i+1}, X_{t_{i+1}}^{(k)}) = g(X_{t_{i+1}}^{(k)})$ computed e.g. by blackbox
 - Actor update:
$$\theta \leftarrow \theta + \eta^G (\mathcal{V}_\phi(t_{i+1}, X_{t_{i+1}}^{(k)}) - \mathcal{V}_\phi(t_i, X_{t_i}^{(k)})) \nabla_\theta \log \rho_\theta(t_i, X_{t_i}^{(k)}, \alpha_{t_i}^{(k)})$$
 - Critic update:
$$\phi \leftarrow \phi + \eta^V (\mathcal{V}_\phi(t_{i+1}, X_{t_{i+1}}^{(k)}) - \mathcal{V}_\phi(t_i, X_{t_i}^{(k)})) \nabla_\phi \mathcal{V}_\phi(t_i, X_{t_i}^{(k)})$$
 - If $X_{t_{i+1}}^{(k)} \notin \mathcal{O}$ or $t_{i+1} = T$, close the loop; Otherwise $i \leftarrow i + 1$;

- end**

end

Return: $\pi_\theta, \mathcal{V}_\phi$.

3 Application to Share Repurchase Programs Pricing

3.1 Problem formulation

We consider a company/client with stock price S . This client mandates a bank to buy a quantity B of shares of stock within a period $[0, T]$. At early termination date τ or at maturity T if no early termination has appeared, the client pays to the bank the Volume Weighted Average Price (in short VWAP) defined as $V_\tau := \frac{1}{\tau} \int_0^\tau S_t dt$, discounted by the number of shares, i.e., the amount $B V_\tau$. The bank gives to the client the quantity B of shares, and its value at τ is $B S_\tau$. From the bank perspective, it is equivalent to being long an option with payoff $B(V_\tau - S_\tau)$ at τ . If the bank fails to collect the quantity B before T for the company, it must pay a penalty to the client. For the sake of simplicity, we have not included rate, dividends and repo, although this can be easily incorporated.

We denote by $(Q_t)_{t \in [0, T]}$ the quantity of shares (inventory) hold by the trader of the bank, and governed by

$$dQ_t = \alpha_t dt,$$

where α represents the trading speed, valued in $[0, \bar{a}]$, for some constant $\bar{a} \in (0, \infty)$. The underlying stock price S is a continuous time process, possibly controlled by α in presence of permanent market impact. The dynamics of the VWAP process $(V_t)_t$ and of the cumulated cost process $(C_t)_t$ are given by

$$dV_t = \left(\frac{S_t - V_t}{t} \right) dt, \quad 0 < t \leq T, \quad V_0 = S_0, \quad dC_t = \alpha_t S_t dt, \quad C_0 = 0.$$

The profit and loss (PnL) of the bank at execution time $\tau \leq T$ is then given by

$$\text{PnL}_\tau^\alpha = B(V_\tau - S_\tau) - \lambda(B - Q_\tau)_+ - \beta BC_\tau,$$

where $\lambda > 0$ is a penalization parameter, effective when $\tau = T$, and $Q_T < B$, and $\beta \geq 0$ is a transaction cost parameter. The price of the barrier VWAP-minus contract is determined by the following stochastic control problem

$$P_{BV} := \sup_{\alpha \in \mathcal{A}} \mathbb{E}[\text{PnL}_{\tau^\alpha}^\alpha],$$

where \mathcal{A} is the set of admissible trading strategies, and $\tau^\alpha := \inf\{t > 0 \mid Q_t \geq B\} \wedge T$ is the early termination time of the contract, defined as the first time when the inventory exceeds the required quantity B of shares. This fits into the form (1.1) with state variables $X = (S, V, Q, C)$.

Remark 3.1. *In this context, the price of the ASR is given by*

$$P_{ASR} := \sup_{\alpha \in \mathcal{A}} \sup_{\bar{\tau} \in \mathcal{T}_{0,T}} \mathbb{E}[\text{PnL}_{\bar{\tau}}^\alpha],$$

while the price of the VWAP-minus contract as considered in [7] is given by

$$P_V := \sup_{\alpha \in \mathcal{A}} \sup_{\bar{\tau} \in \mathcal{T}_{\tau^\alpha, T}} \mathbb{E}[\text{PnL}_{\bar{\tau}}^\alpha],$$

where $\mathcal{T}_{t,T}$ is the set of stopping times valued in $[t, T]$. The prices of these contracts have been computed in [7] by using two distinct neural networks for approximating the policy α and the stopping time $\bar{\tau}$, and by definition, we should have $P_{ASR} \geq P_V \geq P_{BV}$. Actually, one can show that $P_V = P_{BV}$ in absence of market impact and transaction costs, see Appendix A. In other words, the pricing problem for the VWAP-minus can be reduced to a stochastic control with exit time, and there is no need to consider an additional optimization over stopping times $\bar{\tau}$, which is quite advantageous from a numerical point of view.

The algorithm proposed in [7] considers two neural networks: p_θ for the randomized stopping time and a_ξ for trading rate to estimate the optimal strategy leading to P_V . The optimisation is performed by a stochastic gradient ascent with the loss function

$$\mathcal{L}(\theta, \xi) = \mathbb{E} \left[\sum_{i=0}^{N-1} \prod_{j=0}^{i-1} (1 - p_\theta(t_j, X_{t_j})) p_\theta(t_i, X_{t_i}) \text{PnL}_{t_i} + \prod_{j=0}^{N-1} (1 - p_\theta(t_j, X_{t_j})) \text{PnL}_{t_N} \right].$$

Here $\prod_{j=0}^{i-1} (1 - p_\theta(t_j, X_{t_j})) p_\theta(t_i, X_{t_i})$ represents the probability to exercise at t_i , for a given path of the state variables. For the profit and loss PnL, $(B - Q_{t_i})^+$ is replaced by $|B - Q_{t_i}|$ to prevent the agent from buying once the barrier is reached. Notice that the computation of the gradient of \mathcal{L} with respect to θ and ξ is extremely costly. Furthermore, the numerical experiments show highly unstable results. Instead, our policy gradient algorithms is less costly and show stable results.

3.2 Numerical results

For the numerical results and comparison with other methods, we consider a price process with linear permanent price impact, governed by

$$dS_t = S_t(\gamma\alpha_t dt + \sigma dW_t), \quad 0 \leq t \leq T,$$

where $\gamma \geq 0$ is a constant market impact parameter. The value function $P(t, x)$ with $t \in [0, T]$, $x = (s, v, q, c) \in \mathbb{R}_+^* \times \mathbb{R}_+^* \times \mathbb{R}_+ \times \mathbb{R}_+$, is solution to the Bellman equation:

$$\begin{aligned} & \partial_t P + \bar{a}(\gamma s \partial_s P + s \partial_c P + \partial_q P)^+ \\ & + \frac{s-v}{t} \partial_v P + \frac{1}{2} \sigma^2 s^2 \partial_s^2 P = 0, \quad t \in (0, T), (s, v, q, c) \in \mathbb{R}_+^* \times \mathbb{R}_+^* \times [0, B) \times \mathbb{R}_+, \end{aligned} \quad (3.1)$$

with the boundary conditions:

$$\begin{cases} P(t, x) &= B(v - s) - \beta Bc, \quad t \in [0, T], (s, v, q, c) \in \mathbb{R}_+^* \times \mathbb{R}_+^* \times [B, \infty) \times \mathbb{R}_+, \\ P(T, x) &= B(v - s) - \lambda(B - q)_+ - \beta Bc, \quad (s, v, q, c) \in \mathbb{R}_+^* \times \mathbb{R}_+^* \times \mathbb{R}_+ \times \mathbb{R}_+. \end{cases}$$

Notice that the optimal feedback control is of bang-bang type, namely:

$$\hat{a}(t, x) = \begin{cases} 0 & \text{if } \gamma s \partial_s P + s \partial_c P + \partial_q P \leq 0, \\ \bar{a} & \text{otherwise,} \end{cases}$$

and therefore, we shall consider a softmax randomized policy as in (2.1) with two possible values in $\{0, \bar{a}\}$.

For numerical experiments of our algorithms to the pricing of Barrier VWAP-minus, we neglect transaction costs $\beta = 0$, and take the following parameters: $T = 60$ days, $S_0 = 1$, $B = 1$, and \bar{a} ranging from 5.04 to 25.2, $\lambda = 5$, $\Delta t = 1/252$, number of Monte-Carlo simulations: $N_{MC} = 10^5$.

For the architecture of the neural networks for the randomized policies and the value function (for the actor-critic AC algorithm), we have used neural networks with 2 hidden layers of dimension 8 (linear output and Relu as intermediate activation function). The SGD is an Adam algorithm with standard hyper-parameters and 64 as mini-batch size for SGP and 32 for AC¹. We first compute the price $P_{BV} \times 10^4$ in absence of market impact $\gamma = 0$, and compare with the results obtained by HJB solver² (see Appendix B). We fix $\sigma = 0.2$, and vary the maximal trading rate \bar{a} , and display the associated prices

¹The algorithm has been written from scratch in C_{++} .

²We thank A. Conze and J. Adrien for their contributions to the PDE implementation of this project.

in Figure 1. By construction, as we compute the expectation for a sub-optimal control, we obtain a lower bound. In particular, as the underlying price process is a martingale, note that using a constant control, we get 0 bp. The graph of convergence in terms of the number of episodes of the algorithm for two pairs of parameters of (\bar{a}, σ) , is reported in Figure 2.

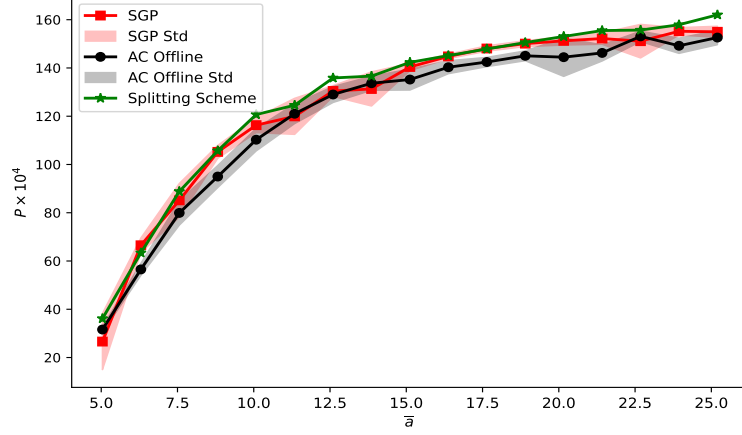


Figure 1: $P_{BV} \times 10^4$ in absence of market impact and transaction costs for different values of \bar{a} computed with stochastic gradient policy and actor critic compared to splitting scheme (HJB solver).

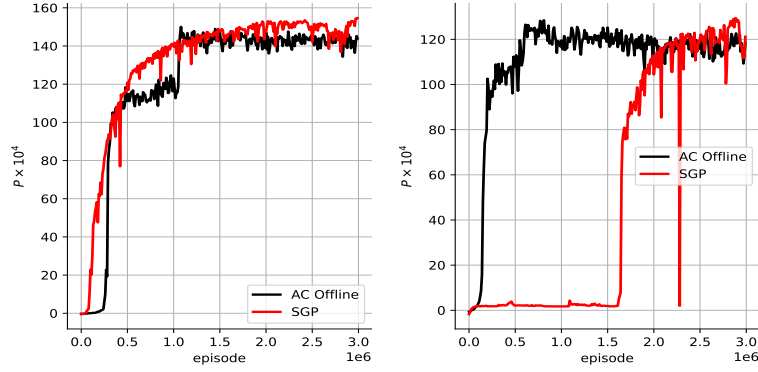


Figure 2: Convergence as a function of iterations for $P_{BV} \times 10^4$ (without market impact and transaction costs) for $\bar{a} = 36.5, \sigma = 0.2$ (left) and $\bar{a} = 9, \sigma = 0.25$ (right)

The two algorithms (SGP and AC) produce results that are similar to those obtained using splitting scheme in terms of price. Furthermore, the execution time of these algorithms is also found to be comparable to that of HJB solver, with both methods taking about two minutes to converge, indicating that they are computationally efficient and

capable of solving the problem in a timely manner. However, when the number of state variables increases, the PDE method becomes computationally very costly in comparison to our proposed methods. This means that for problems involving a large number of state variables, our method becomes the only viable option. Overall, the results of this study demonstrate that our proposed algorithms are a reliable and cost-effective alternative to the PDE method for solving this class of problems.

Next, we display the surface of the optimal randomized policy for fixed spot price S , for two different values of t ($t = T/2$ and t near maturity T), and as a function of the VWAP and inventory. Figure 3 shows the results in absence of market impact while Figure 4 considers the case with market impact. We observe that when we are close to the maturity, the probability of choosing the maximal trading rate is equal to one for almost all states of the VWAP and inventory with or without market impact: this is due to the fact that the trader has to achieve the goal of repurchasing the requested quantity of shares as he would be penalized otherwise. When we are in the midterm of the program, the optimal policy consists in choosing the maximal trading rate only when the VWAP is larger than some threshold, say V^* , as he has enough time to complete his repurchasing goal. In absence of market impact, this threshold V^* is approximately equal to the spot price, while in presence of market impact, this threshold decreases with the market impact and also with the inventory. In other words, the trader will buy more quickly some fraction of the total shares B as the market impact is more penalizing when approaching maturity.

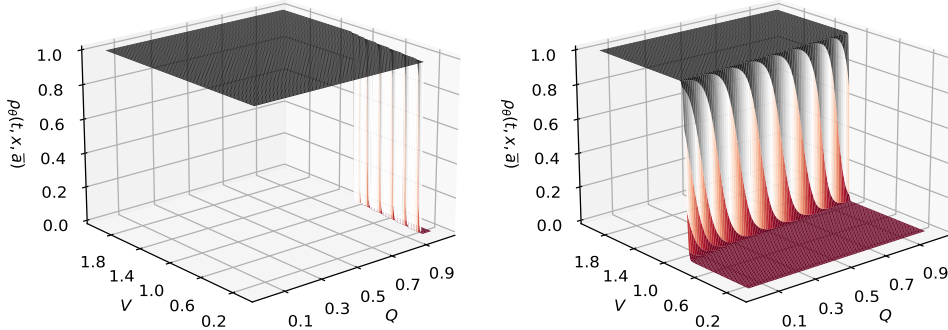


Figure 3: Optimal policy $\rho_\theta(t, x, \bar{a})$ in absence of market impact and transaction costs for $Q \in [0, 1]$, $V \in [0.1, 2]$, $S = 1$, $\sigma = 0.2$, $t = T - dt$ (left) and $t = \frac{T}{2}$ (right).

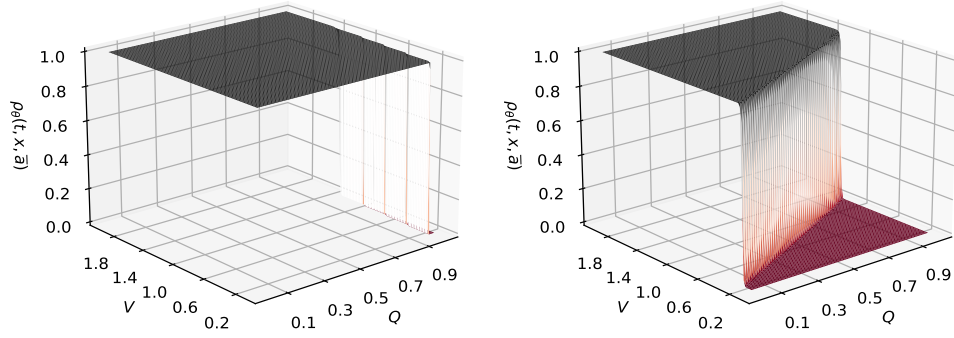


Figure 4: Optimal policy $\rho_\theta(t, x, \bar{a})$ when market impact is included ($\gamma = 0.1$) for $Q \in [0, 1]$, $V \in [0.1, 2]$, $S = 1$, $\sigma = 0.2$, $t = T - dt$ (left) and $t = \frac{T}{2}$ (right).

Finally, we represent the evolution of the optimal inventory for two price realizations, in the case without market impact (see Figure 5) and with market impact (see Figure 6) The trader starts by purchasing some fraction of the total shares B (and this is done more quickly and with a higher fraction in presence of market impact), then do not trade for a while until the time when the spot price falls below the VWAP, where he purchases the remaining shares to complete the buy-back programme.

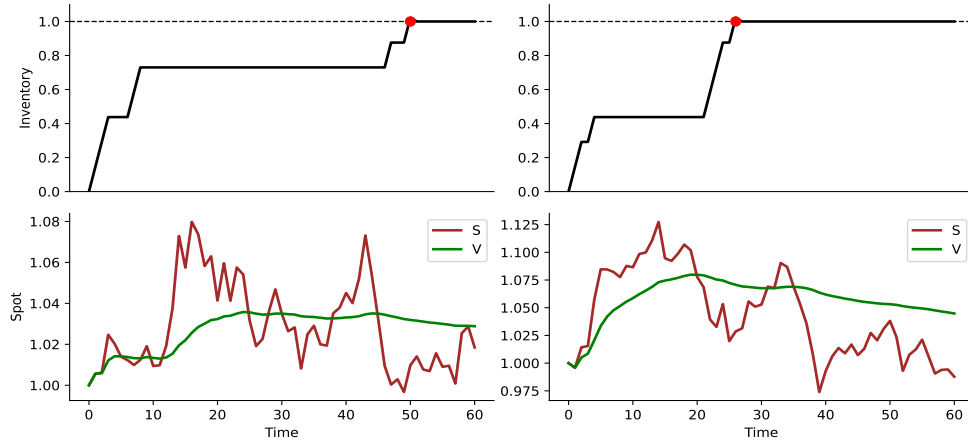


Figure 5: Optimal repurchase strategy evolution for two price realizations ($\sigma = 0.2$) in absence of market impact and transaction costs.

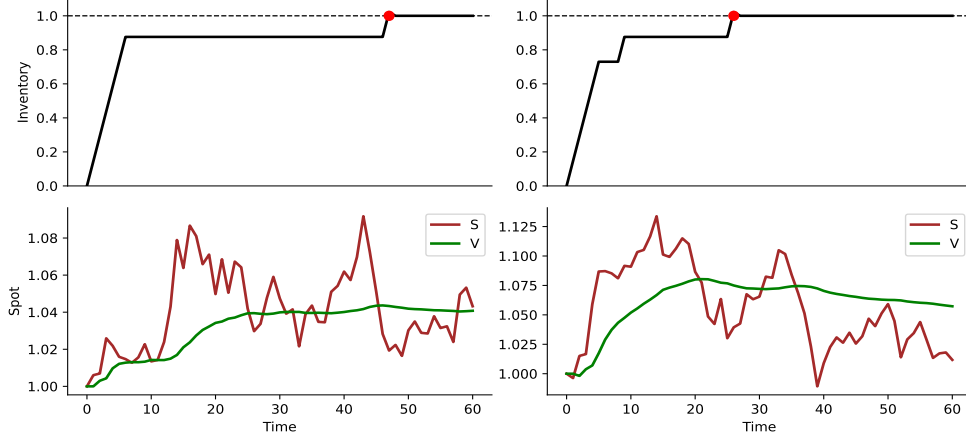


Figure 6: Optimal repurchase strategy evolution for two price realizations ($\sigma = 0.2$) with market impact ($\gamma = 0.1$)

A Barrier VWAP-minus vs VWAP-minus

Given a trading strategy $\alpha \in \mathcal{A}$, valued in $A = [0, \bar{a}]$, we denote by τ^α the first time when the inventory $Q_t^\alpha = \int_0^t \alpha_s ds$ reaches B , and we consider the price of the VWAP-minus and Barrier VWAP-minus given by

$$P_V = \sup_{\alpha \in \mathcal{A}} \sup_{\bar{\tau} \in \mathcal{T}_{\tau^\alpha, T}} \mathbb{E}[\text{PNL}_{\bar{\tau}}^\alpha] \quad P_{BV} = \sup_{\alpha \in \mathcal{A}} \mathbb{E}[\text{PNL}_{\tau^\alpha}^\alpha],$$

where the PNL, in absence of transaction costs, is given by

$$\text{PNL}_t^\alpha = B \left(\frac{1}{t} \int_0^t S_s ds - S_t \right) - \lambda(B - Q_t^\alpha)_+, \quad 0 \leq t \leq T.$$

The price process S is a general continuous semimartingale process without market impact, and satisfying

$$\mathbb{E} \left[\max_{t \in [0, T]} |S_t| \right] < \infty. \quad (\text{A.1})$$

Notice that by Doob's inequality, such condition (A.1) is satisfied whenever the drift and the volatility of the asset price S are bounded.

Proposition A.1. *Under (A.1), and in absence of market impact and transaction costs, we have $P_{BV} = P_V$.*

Proof. Fix some arbitrary $\alpha \in \mathcal{A}$, and $\bar{\tau} \in \mathcal{T}_{\tau^\alpha, T}$. For $\varepsilon > 0$, denote by $\tau_\varepsilon^\alpha = \inf\{t \geq 0 : Q_t^\alpha = B - \varepsilon\} \wedge T$, which is smaller than τ^α , and converges a.s. to τ^α when ε goes to

zero. Let us then define trading strategy $\alpha^\varepsilon \in \mathcal{A}$ by

$$\alpha_t^\varepsilon = \begin{cases} \alpha_t & \text{for } 0 \leq t \leq \tau_\varepsilon^\alpha \\ 0 & \text{for } \tau_\varepsilon^\alpha < t \leq \bar{\tau} \\ \bar{a} & \text{for } \bar{\tau} < t \leq T, \end{cases}$$

which leads to an associated inventory Q^{α^ε} given by

$$Q_t^{\alpha^\varepsilon} = \begin{cases} Q_t^\alpha & \text{for } 0 \leq t \leq \tau_\varepsilon^\alpha \\ B - \varepsilon & \text{for } \tau_\varepsilon^\alpha < t \leq \bar{\tau} \\ B - \varepsilon + \bar{a}(t - \bar{\tau}) & \text{for } \bar{\tau} < t \leq T. \end{cases}$$

Notice that $\tau^{\alpha^\varepsilon}$ (the first time when Q^{α^ε} reaches B) is lower-bounded by $\bar{\tau}$, decreases with ε , and converges a.s. to $\bar{\tau}$ when ε goes to zero.

By definition, we have $P_{BV} \geq \mathbb{E}[\text{PNL}_{\tau^{\alpha^\varepsilon}}^{\alpha^\varepsilon}]$. Let us check that $\text{PNL}_{\tau^{\alpha^\varepsilon}}^{\alpha^\varepsilon}$ converges a.s. to $\text{PNL}_{\bar{\tau}}^\alpha$ when ε goes to zero. We distinguish two cases:

- If $\tau^\alpha < T$. Then, $Q_{\tau^\alpha}^\alpha = B \leq Q_{\bar{\tau}}^\alpha$, and $Q_{\tau^{\alpha^\varepsilon}}^{\alpha^\varepsilon} = B - \varepsilon + \bar{a}(\tau^{\alpha^\varepsilon} - \bar{\tau})$ converges to B when ε goes to zero. It follows that

$$\begin{aligned} \text{PNL}_{\tau^{\alpha^\varepsilon}}^{\alpha^\varepsilon} &= B \left(\frac{1}{\tau^{\alpha^\varepsilon}} \int_0^{\tau^{\alpha^\varepsilon}} S_s ds - S_{\tau^{\alpha^\varepsilon}} \right) - \lambda(B - Q_{\tau^{\alpha^\varepsilon}}^{\alpha^\varepsilon})_+ \\ &\rightarrow B \left(\frac{1}{\bar{\tau}} \int_0^{\bar{\tau}} S_s ds - S_{\bar{\tau}} \right) = \text{PNL}_{\bar{\tau}}^\alpha, \end{aligned}$$

as ε goes to zero.

- If $\tau^\alpha = T$. Then $\bar{\tau} = T = \tau^{\alpha^\varepsilon}$, and α_t^ε converges to α_t , for $0 \leq t < T$, when ε goes to zero. It follows that $Q_T^{\alpha^\varepsilon}$ converges to Q_T^α . Therefore,

$$\begin{aligned} \text{PNL}_{\tau^{\alpha^\varepsilon}}^{\alpha^\varepsilon} &= B \left(\frac{1}{T} \int_0^T S_s ds - S_T \right) - \lambda(B - Q_T^{\alpha^\varepsilon})_+ \\ &\rightarrow B \left(\frac{1}{T} \int_0^T S_s ds - S_T \right) - \lambda(B - Q_T^\alpha)_+ = \text{PNL}_T^\alpha, \end{aligned}$$

as ε goes to zero.

Moreover, by noting that $|\text{PNL}_{\tau^{\alpha^\varepsilon}}^{\alpha^\varepsilon}| \leq B(2 \max_{t \in [0, T]} |S_t| + \lambda)$, and under (A.1), we can apply dominated convergence theorem to deduce that

$$\mathbb{E}[\text{PNL}_{\tau^{\alpha^\varepsilon}}^{\alpha^\varepsilon}] \rightarrow \mathbb{E}[\text{PNL}_{\bar{\tau}}^\alpha], \quad \text{when } \varepsilon \text{ goes to zero,}$$

and so $P_{BV} \geq \mathbb{E}[\text{PNL}_{\bar{\tau}}^\alpha]$. Since this holds true for any $\alpha \in \mathcal{A}$, and $\bar{\tau} \in \mathcal{T}_{\tau^\alpha, T}$, we conclude that $P_{BV} \geq P_V$, hence the equality since it is clear that $P_V \geq P_{BV}$. \square

B PDE Implementation by splitting scheme

We solve the Bellman (HJB) equation (3.1) by backward induction. We know P at T (*Terminal condition*). Now, we assume that we know P at t and we want to compute P at a previous date $t - \Delta t$. We use the approximation:

$$\bar{a}1_{\{(\gamma s \partial_s + s \partial_c + \partial_q)P(t-\Delta t, x) \geq 0\}} \approx \bar{a}1_{\{(\gamma s \partial_s + s \partial_c + \partial_q)P(t, x) \geq 0\}} := \tilde{a}^*(t, x)$$

for all $x = (s, v, q, c) \in \mathcal{O} = \mathbb{R}_+^* \times \mathbb{R}_+^* \times (0, B) \times \mathbb{R}_+$. The HJB equation becomes

$$\partial_t P|_{\mathcal{O}} + \mathcal{L}P|_{\mathcal{O}} + \mathcal{D}P|_{\mathcal{O}} = 0 \quad (\text{B.1})$$

where $P|_{\mathcal{O}}$ is the restriction of P to \mathcal{O} , \mathcal{L} is a diffusion operator and \mathcal{D} is a transport operator defined over \mathcal{O} as

$$\begin{aligned} \mathcal{L} \cdot &= \frac{1}{2} \sigma^2 s^2 \partial_{ss}^2 \cdot \\ \mathcal{D} \cdot &= \frac{s-v}{t} \partial_v \cdot - \tilde{a}^*(t, x) \partial_q \cdot \end{aligned}$$

where $x = (s, v, q, c) \in \mathcal{O}$. One can verify that \mathcal{L} , \mathcal{D} and $\mathcal{L} + \mathcal{D}$ generate a C^0 semi-groups, thus, the solution of (B.1) at $t - \Delta t$ can be represented as

$$P(t - \Delta t, x) = e^{\Delta t(\mathcal{L} + \mathcal{D})} P(t, x)$$

where $e^{\Delta t(\mathcal{L} + \mathcal{D})}$ denotes the semi-group associated to the parabolic linear PDE (B.1). A first order approximation of the solution operator is obtained using Baker–Campbell–Hausdorff formula and Lie–Trotter splitting (see [13])

$$e^{\Delta t(\mathcal{L} + \mathcal{D})} P(t, x) = e^{\Delta t \mathcal{D}} e^{\Delta t \mathcal{L}} P(t, x) + O(\Delta t) \quad (\text{B.2})$$

One can also use Strang splitting $e^{\frac{\Delta t}{2} \mathcal{D}} e^{\Delta t \mathcal{L}} e^{\frac{\Delta t}{2} \mathcal{D}}$ to get a second order approximation. The splitting (B.2) corresponds to solving the parabolic PDE first with generator \mathcal{L} and then the first-order transport PDE corresponding to the operator \mathcal{D} . By using the method of characteristics, the solution corresponding to \mathcal{D} is explicitly given by

$$e^{\Delta t \mathcal{D}} Q(t, x) = Q(t, s, v + \frac{s-v}{t} \Delta t, q + \tilde{a}^*(t, x) \Delta t, c + \tilde{a}^*(t, x) s \Delta t)$$

where $x = (s, v, q, c) \in \mathcal{O}$ and $Q(t, x) = e^{\Delta t \mathcal{L}} P(t, x)$. Finally, we extend $P(t, \cdot)$ to $\mathbb{R}_+^* \times \mathbb{R}_+^* \times \mathbb{R} \times \mathbb{R}$ using boundary conditions.

References

- [1] Technical committee of the international organization of securities commissions. Technical report, Report On Stock Repurchase Programs, 2004.

- [2] C. Beck, P. Cheridito, and A. Jentzen. Deep optimal stopping. *Journal of Machine Learning Research*, 74(1):1–25, 2019.
- [3] C. Beck, M. Hutzenthaler, A. Jentzen, and B. Kuckuck. An overview on deep learning-based approximation methods for partial differential equations. *arXiv preprint: 2012.12348*, 2020.
- [4] B. Bouchard and S. Menozzi. Strong approximations of BSDE in a domain. *Bernoulli*, 15(4):1117–1147, 2009.
- [5] H. Buehler, L. Gonon, J. Teichmann, and B. Wood. Deep hedging. *Quantitative Finance*, 19(8), 2019.
- [6] M. Germain, H. Pham, and X. Warin. Neural networks-based algorithms for stochastic control and PDEs in finance, 2021.
- [7] O. Guéant, I. Manziuk, and J. Pu. Accelerated share repurchase and other buyback programs: what neural networks can bring. *Quantitative Finance*, 20(8), 2020.
- [8] O. Guéant, J. Pu, and G. Royer. Accelerated share repurchase: pricing and execution strategy. *International Journal of Theoretical and Applied Finance*, 18(3), 2015.
- [9] J. Han and W. E. Deep learning approximation for stochastic control problems. *NIPS*, 2016.
- [10] S. Jaimungal, D. Kinzebulatov, and D. Rubisov. Optimal accelerated share repurchase. *Applied Mathematical Finance*, 24(3), 2017.
- [11] M. Miller and F. Modigliani. Dividend policy, growth, and the valuation of shares. *Journal of Business*, 34(411), 1961.
- [12] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018, 2nd edition.
- [13] H. F. Trotter. On the product of semi-groups of operators. *Amer. Math. Soc.*, 10:545–551, 1959.