arXiv:2408.09242v3 [math.OC] 11 Aug 2025

Learning to Optimally Stop Diffusion Processes, with Financial Applications*

Min Dai † Yu Sun ‡ Zuo Quan Xu § Xun Yu Zhou ¶

August 11, 2025

Abstract

We study optimal stopping for diffusion processes with unknown model primitives within the continuous-time reinforcement learning (RL) framework developed by Wang et al. (2020), and present applications to option pricing and portfolio choice. By penalizing the corresponding variational inequality formulation, we transform the stopping problem into a stochastic optimal control problem with two actions. We then randomize controls into Bernoulli distributions and add an entropy regularizer to encourage exploration. We derive a semi-analytical optimal Bernoulli distribution, based on which we devise RL algorithms using the martingale approach established in Jia and Zhou (2022a). We establish a policy improvement theorem and prove the fast convergence of the resulting policy iterations. We demonstrate the effectiveness of the algorithms in pricing finite-horizon American put options, solving Merton's problem with transaction costs, and scaling to high-dimensional optimal stopping problems. In particular, we show that both the offline and online algorithms achieve high accuracy in learning the value functions and characterizing the associated free boundaries.

Keywords: Optimal stopping, reinforcement learning, policy improvement, option pricing, portfolio choice.

*We thank the participants in the First INFORMS Conference on Financial Engineering and FinTech for their comments.

†Department of Applied Mathematics and School of Accounting and Finance, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, China. Email: mindai@polyu.edu.hk

‡Department of Applied Mathematics, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, China. Email: yusun017@gmail.com

§Department of Applied Mathematics, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, China. Email: maxu@polyu.edu.hk

¶Department of Industrial Engineering and Operations Research & Data Science Institute, Columbia University, New York, USA, NY 10027. Email: xz2574@columbia.edu

1 Introduction

Optimal stopping involves making decisions on the best timing to exit from (or enter into) an endeavor with an immediate payoff when waiting for a higher payoff in the future is no longer beneficial. Such problems abound in many applications, including finance, physics, and engineering. A prominent example is American-type options where one needs to decide when to exercise the right to buy or sell an underlying stock.

Optimal stopping can be regarded as a control problem in which there are only two control actions: stop (exit) or wait (continue). Yet historically, the optimal stopping theory has been developed fairly separately from that of the general optimal control, owing to the very specific nature of the former, even though both are based on the fundamentally same approaches, namely martingale theory and dynamic programming (DP). For optimal stopping with a continuous-time diffusion process, DP leads to a variational inequality problem (also known as a free-boundary PDE) that, *if* solved, can be used to characterize the stopping/continuation regions in the time–state space and to obtain the optimal value of the problem.

However, there are two issues with the classical approach to optimal stopping, one conceptual and one technical. The first one is that the variational inequality formulation is available only when the model primitives are known and given, while in many applications model primitives are often hard to estimate to the desired accuracy or are simply unavailable. The second issue is that even if the model is completely known and the variational inequality formulation is available, solving it by traditional numerical PDE methods (e.g., the finite difference method) in high dimensions is notoriously difficult, if not impossible, due to the curse of dimensionality.

This is where continuous-time reinforcement learning (RL) comes to the rescue. RL has been developed to solve dynamic optimization in partially or completely unknown environments. This is achieved through interacting with and exploring the environment: the RL agent tries different actions strategically and observes responses from the environment, based on which she learns to improve her action plans. A key to the success of this process is to carefully balance exploration (learning) and exploitation (optimization). RL is also a powerful way to deal with the curse of dimensionality, because it no longer attempts to solve dynamic programming equations; rather it employs Monte Carlo and stochastic gradient descent to compute policies and values, which is less prone to the dimensionality complexity (e.g., Han et al., 2018).

While RL for discrete-time Markov decision processes (MDPs) has been developed for a long time (Sutton and Barto, 1999), recently the study of RL for controlled continuous-time diffusion processes has drawn increasing attention, starting from Wang et al. (2020). Continuous RL problems are important because many practical problems are naturally continuous in time (and in space). While one can discretize time upfront and turn a

continuous-time problem into a discrete-time MDP, it has been experimentally shown (e.g., Munos, 2006; Tallec et al., 2019) that this approach is very sensitive to time discretization and performs poorly with small time steps. Equally important, there are more mathematical tools available for the continuous setting that make it possible to establish an overarching and rigorous theoretical foundation for RL algorithms, as demonstrated in a series of papers Jia and Zhou (2022a,b, 2023). Indeed, the study on continuous RL may provide new perspectives even for discrete-time MDPs.[1] The general theory has already found applications in finance, including dynamic mean–variance analysis (Wang and Zhou, 2020; Dai et al., 2023; Wu and Li, 2024) and Merton's problem (Jiang et al., 2022; Dai et al., 2023; Dai and Dong, 2024), to name but a few.

This paper studies optimal stopping in finite time horizons for (potentially high-dimensional) diffusion processes with unknown model parameters. Instead of developing an RL theory for optimal stopping from the ground up parallel to the control counterpart (Wang et al., 2020; Jia and Zhou, 2022a,b, 2023), we first transform the stopping problem into a stochastic control problem, inspired by Dai et al. (2007). Indeed this transformation is motivated by the fact that the variational inequality problem can be approximated by a "penalized" PDE, whose solution converges to that of the original variational inequality problem as the penalty parameter approaches infinity (Friedman, 1982; Forsyth and Vetzal, 2002; Liang et al., 2007; Peng et al., 2024). Moreover, the penalized PDE is actually the Hamilton–Jacobi–Bellman (HJB) equation of a stochastic bang-bang control problem.

Once we have turned the problem into an RL problem for stochastic control, we can apply/adapt all the available results, methods, and algorithms developed for the latter right away. In particular, we take the exploratory formulation of Wang et al. (2020) involving control randomization and entropy regularization. Because the control set now contains only two actions (corresponding to stop and continuation), the randomized control is a Bernoulli distribution. To wit, for exploring the environment there is no longer a clear-cut stop or continuation decision. Rather, the agent designs a (biased) coin for each time–state pair and makes the stopping decision based on the outcome of a coin toss.[2] Moreover, we derive a semi-analytical formula for the optimal Bernoulli distribution, which is crucial for policy parameterization and policy improvement in designing our RL algorithms. Built upon the martingale approach established by Jia and Zhou (2022a), we

[1] For instance, a theoretical underpinning of Jia and Zhou (2022a,b, 2023) is the martingale property of certain stochastic processes. It turns out that there is an analogous, and almost trivial, martingality for discrete-time MDPs; see Jia and Zhou (2022a, Appendix C), but such a property had never been mentioned let alone exploited in the literature per our best knowledge.

[2] Interestingly, in a seemingly different context (a casino gambling model originally proposed by Barberis (2012) featuring behavioral finance theory), He et al. (2017) and Hu et al. (2023) show that introducing coin-toss for making stopping decisions can on one hand strictly increase the cumulative prospect theory preference value and on the other hand make the problem analytically tractable. However, these papers stop short of explaining why people actually toss coins in reality. Exploration *à la* RL probably offers one explanation.

design two RL algorithms: an offline martingale loss (ML) algorithm and an online TD(0) algorithm. Moreover, we provide a theoretical guarantee – a fast convergence result of the policy improvement/iteration.

We then conduct an extensive simulation study, applying these algorithms to several problems, including pricing finite-horizon American put options and solving Merton's portfolio choice problem with transaction costs. More importantly, we experiment with two *high*-dimensional problems: American geometric average put options and optimal stopping for fractional Brownian motions. Through these experiments we demonstrate the effectiveness and efficiency of our reinforcement learning methods and algorithms, particularly their scalability to high-dimensional settings.

This work is closely related to Dong (2024), the first paper in the literature to study optimal stopping within the continuous-time exploratory framework of Wang et al. (2020). However, there are stark differences. First, we turn the stopping problem into a stochastic control problem using the penalty method, while Dong (2024) treats the stopping problem directly by randomizing stopping times with a Cox process. The advantage of our approach, as already noted earlier, is that once in the control formulation, we can immediately apply all the available RL theories for controlled diffusions developed so far, instead of having to develop the corresponding theories for stopping from the get-go. Second, the entropy regularizers are chosen differently. We adopt the usual Shannon differential entropy, also used by Wang et al. (2020) and its sequels, while Dong (2024) employs the so-called unnormalized negentropy. See Section 4.6 for a more detailed discussion on this point. Finally, in our experiments, we examine the performance of our methods with both online and offline algorithms, while Dong (2024) only shows his offline results.

Another recent relevant paper is Dianetti et al. (2024).[3] That paper is also significantly different from ours in several fronts. First, Dianetti et al. (2024) use randomized stopping times to turn their optimal stopping into a *singular* control problem, and then apply the exploratory formulation to the latter which is probably more complex than the former. By contrast, we do quite the opposite: in solving Merton's problem with transaction costs which is a singular control problem, we turn it into optimal stopping and then into regular control. The reason for us doing so is that singular control involves gradient constraints, which are always hard to treat. In contrast, optimal stopping, along with regular control, is much easier to deal with. That said, it is important to develop an RL theory for singular control (beyond the motivation for solving optimal stopping). In this regard, Dianetti et al. (2024) obtain many results including those of theoretical convergence. Second, in terms of algorithm design, the value function in Dianetti et al. (2024) is approximated by Monte Carlo thus offline in nature, and their policy improvement involves the *gradient* of

[3]The paper came to our attention shortly after the previous version of our paper was completed and posted.

the value function that requires very fine grids and a large amount of data to learn. In comparison, we have both online and offline algorithms, which do not demand computing the aforementioned gradient. Finally, Dianetti et al. (2024) study stopping in infinite time horizon, while we deal with finite horizons. Besides the consideration of practical applicability, optimal stopping in a finite horizon is significantly harder than its infinite horizon counterpart.

This work is also related to a strand of papers on machine learning for optimal stopping (e.g., Becker et al., 2019, 2020, 2021; Reppen et al., 2025; Herrera et al., 2024; Peng et al., 2024). These papers focus on using deep neural nets to approximate the free boundaries and/or optimal stopping, significantly outside the entropy-regularized, exploratory framework of Wang et al. (2020) to which the present paper belongs. The RL algorithms developed in this paper can also be applied to optimal stopping problems beyond the field of finance. For instance, the Stefan problem in physics, which describes the evolution of the boundary between two phases of a material undergoing a phase change (Wang and Perdikaris, 2021), can be formulated as an optimal stopping problem (Grigelionis and Shiryaev, 1966). In medicine, optimal stopping is relevant to problems such as radiotherapy treatment, where the goal is to determine when to stop the current treatment scheme (Ajdari et al., 2019). Many other engineering problems also involve optimal stopping, such as deciding the best time to stop the production of a natural resource (Pham, 2009), early stopping of model training in machine learning using Bayesian methods (Dai et al., 2019), and classifying urban reports by deciding when to stop adding more features once they no longer enhance classification accuracy (Liyanage et al., 2019).

The remainder of the paper is organized as follows. Section 2 transforms the optimal stopping problem with a finite horizon into a bang-bang stochastic control problem and then presents its exploratory formulation. In Section 3, we propose both the online and offline RL algorithms and provide a policy improvement theorem with a theoretical convergence rate. Section 4 is devoted to simulation studies on pricing an American put option and on an investment problem with transaction costs, while Section 5 is on two high-dimensional examples. Finally, Section 6 concludes. All the proofs and additional materials are presented in the appendices.

2 Optimal stopping in finite time horizon

We fix a filtered probability space $(\Omega, \mathcal{F}, \mathbb{P}; \{\mathcal{F}_t\}_{t\geqslant 0})$ in which a standard d-dimensional Brownian motion $W = (W^{(1)}, W^{(2)}, \cdots, W^{(d)})$ is defined, along with a fixed finite time $T > 0$. Consider the following optimal stopping problem with a finite horizon

$$\max_{\tau\in[0,T]} \mathbb{E}[g(\tau, X_\tau)] \tag{2.1}$$

for an n-dimensional diffusion process $X = (X^{(1)}, X^{(2)}, \cdots, X^{(n)})$ driven by

$$\mathrm{d}X_t^{(i)} = b_i(t, X_t)\,\mathrm{d}t + \sum_j \sigma_{i,j}(t, X_t)\,\mathrm{d}W_t^{(j)}. \tag{2.2}$$

It is well-known that the HJB equation of the problem (2.1) is given by

$$\begin{cases} \max\{v_t + \mathscr{L}v, g - v\} = 0, \ (t,x) \in [0,T) \times \mathbb{R}^n, \\ v(T,x) = g(T,x), \end{cases} \tag{2.3}$$

where

$$\mathscr{L}v(t,x) = \frac{1}{2}\sum_{i,j,k} \sigma_{i,k}(t,x)\sigma_{j,k}(t,x)\frac{\partial^2 v}{\partial x_i \partial x_j} + \sum_i b_i(t,x)\frac{\partial v}{\partial x_i}. \tag{2.4}$$

The equation (2.3) is equivalent to a set of two inequalities and one equality, referred to as a *variational inequality* problem.

Throughout this paper, we assume all the coefficients are continuous in t and Lipschitz continuous in x. Moreover, $\sigma\sigma'$ is uniformly bounded and positive definite, where $\sigma = (\sigma_{i,j})_{i,j}$.

2.1 Penalty approximation

Consider the following penalty approximation problem for (2.3):

$$\begin{cases} v_t + \mathscr{L}v + K(g - v)^+ = 0, \ (t,x) \in [0,T) \times \mathbb{R}^n, \\ v(T,x) = g(T,x), \end{cases} \tag{2.5}$$

where K is a positive scalar, called the *penalty factor*. The following result states the relationship between (2.5) and (2.3).

Lemma 2.1. *The solution of* (2.5) *converges to that of* (2.3) *as* $K \to +\infty$.

A proof of this lemma can be found in the literature such as Friedman (1982), Liang et al. (2007), and Peng et al. (2024). Indeed, one can obtain the following estimate:

$$0 \leqslant v^{\mathrm{os}}(t,x) - v^K(t,x) \leqslant \frac{C}{K}, \quad \forall\ (t,x,K) \in [0,T] \times \mathbb{R}^n \times (0,\infty) \tag{2.6}$$

under some technical conditions, where v^{os} and v^K are respectively the solutions to (2.3) and (2.5), and C is a positive constant independent of K. For reader's convenience we provide a proof of Lemma 2.1 in Appendix A.1.

Lemma 2.1 suggests that solving (2.3) reduces to solving (2.5) with a sufficiently large K. In particular, numerically one can simply solve the latter to get an approximated solution to the former. We can rewrite (2.5) as the following PDE involving a binary optimization term

$$0 = v_t + \mathscr{L}v + K(g - v)^+ = v_t + \max_{u \in \{0,1\}} \{\mathscr{L}v - Kuv + Kg(t,x)u\}. \tag{2.7}$$

We are now going to show in the next subsection that this PDE corresponds to some (unconventional) stochastic control problem.

2.2 Transformation to optimal control

Before we further analyze (2.7), let us first consider the following general stochastic control problem

$$v(s,x) = \max_{u \in \mathscr{U}} J(s,x;u), \tag{2.8}$$

where the controlled n-dimensional process X_t follows

$$\mathrm{d}X_t = \tilde{b}(t, X_t, u_t)\,\mathrm{d}t + \tilde{\sigma}(t, X_t, u_t)\,\mathrm{d}W_t, \tag{2.9}$$

and the objective functional (also called the reward function in the RL literature) is defined as

$$J(s,x;u) = \mathbb{E}\left[\int_s^T e^{-\int_s^t f(r,X_r,u_r)\,\mathrm{d}r} G(t,X_t,u_t)\,\mathrm{d}t + e^{-\int_s^T f(r,X_r,u_r)\,\mathrm{d}r} H(X_T) \,\middle|\, X_s = x\right].$$

Here $\mathscr{U}$ denotes the set of admissible controls $u : [0,T] \times \Omega \to U$ which are progressively measurable such that (2.9) admits a unique strong solution and $J(s,x;u)$ is finite, while $U \subseteq \mathbb{R}^m$ is a given closed control constraint set. In addition, $G : [0,T] \times \mathbb{R}^n \times U \to \mathbb{R}$ and $H : \mathbb{R}^n \to \mathbb{R}$ represent the running reward and terminal reward terms, respectively, in the objective functional J of the control problem (2.8), while $f : [0,T] \times \mathbb{R}^n \times U \to \mathbb{R}$ serves as a discount factor.

Note that this is not a conventional stochastic control problem (e.g., Yong and Zhou, 1999) as it involves control-dependent discount factors. However, we still have the following HJB equation and verification theorem.

Theorem 2.2 (Verification theorem). *Suppose v is a classical solution of*

$$\begin{cases} v_t + \max_{u \in U} \{\mathscr{L}_u v - vf + G\} = 0, \quad (t,x) \in [0,T) \times \mathbb{R}^n, \\ v(T,x) = H(x), \end{cases} \tag{2.10}$$

where

$$\mathscr{L}_u v = \frac{1}{2} \sum_{i,j,k} \tilde{\sigma}_{i,k}(t,x,u)\tilde{\sigma}_{j,k}(t,x,u) \frac{\partial^2 v}{\partial x_i \partial x_j} + \sum_i \tilde{b}_i(t,x,u) \frac{\partial v}{\partial x_i}.$$

Then it is the optimal value function of problem (2.8).

Remark 2.1. *If f is a constant, then the above result reduces to the classical verification theorem. One can even establish a similar result when a term like $e^{-\int_s^t f(r,X_r,u_r)\,\mathrm{d}r - \int_s^t g(r,X_r,u_r)\,\mathrm{d}W_r}$ is in place of $e^{-\int_s^t f(r,X_r,u_r)\,\mathrm{d}r}$ in the reward functional J.*

By Theorem 2.2, the PDE (2.7) is indeed the HJB equation of the following stochastic control problem:

$$v(s,x) = \max_{u_t \in \{0,1\}} J(s,x;u), \tag{2.11}$$

where the (uncontrolled) process X_t is driven by (2.2) and

$$J(s,x;u)=\mathbb{E}\left[\int_s^T Ke^{-K\int_s^t u_r\,\mathrm{d}r}g(t,X_t)u_t\,\mathrm{d}t+e^{-K\int_s^T u_r\,\mathrm{d}r}g(T,X_T)\,\middle|\,X_s=x\right]. \tag{2.12}$$

Applying Theorem 2.2 yields

Lemma 2.3. *Suppose v is a classical solution of (2.7) with terminal condition $v(T,x)=g(T,x)$. Then it is the value function of problem (2.11).*

Even though we have established the theoretical dynamic programming result for the non-conventional control problem (2.11), it is beneficial (to our subsequent analysis) to present an alternative conventional formulation. This is achieved by introducing an additional state.

Recalling that X_t is driven by (2.2), we introduce a new controlled state equation $\mathrm{d}R_t=-KR_tu_t\,\mathrm{d}t$ or $R_t=R_se^{-K\int_s^t u_r\,\mathrm{d}r}$ for $t\geqslant s$. Let

$$\bar{v}(s,x,r)=\max_{u_t\in\{0,1\}}\bar{J}(s,x,r;u), \tag{2.13}$$

where

$$\bar{J}(s,x,r;u)=\mathbb{E}\left[\int_s^T KR_tg(t,X_t)u_t\,\mathrm{d}t+R_Tg(T,X_T)\,\middle|\,X_s=x,R_s=r\right].$$

Then clearly problem (2.11) has the optimal value $v(s,x)=\bar{v}(s,x,1)$.

As is well known, the HJB equation for the classical control problem (2.13) is given by

$$\begin{cases}\bar{v}_t+\max\limits_{u\in\{0,1\}}\left[\mathscr{L}\bar{v}-Kru\dfrac{\partial\bar{v}}{\partial r}+Krg(t,x)u\right]=0,\ (t,x,r)\in[0,T)\times\mathbb{R}^n\times(0,\infty),\\ \bar{v}(T,x,r)=rg(T,x),\quad \bar{v}(t,x,0)=0.\end{cases} \tag{2.14}$$

Clearly, for $r>0$, $\bar{J}(s,x,r;u)=r\bar{J}(s,x,1;u)$; so $\bar{v}(s,x,r)=r\bar{v}(s,x,1)=rv(s,x)$. Substituting this into (2.14), we get, for $(t,x)\in[0,T)\times\mathbb{R}^n$,

$$\begin{cases}v_t+\max\limits_{u\in\{0,1\}}\left[\frac{1}{2}\sum_{i,j,k}\sigma_{i,k}(t,x)\sigma_{j,k}(t,x)\frac{\partial^2 v}{\partial x_i\partial x_j}+\sum_i b_i(t,x)\frac{\partial v}{\partial x_i}-Kuv+Kg(t,x)u\right]=0,\\ v(T,x)=g(T,x),\end{cases} \tag{2.15}$$

which recovers (2.7). In this sense, the two problems (2.13) and (2.11) are equivalent.

Finally it is clear from (2.7) that the control/action $u=1$ (resp. 0) for (2.13) corresponds to stop (resp. continue) for the original stopping problem.

2.3 RL formulation

Now we are ready to present the RL formulation for problem (2.13) by considering the entropy-regularized, exploratory version of the problem, which was first introduced by Wang et al. (2020) and then studied in many continuous-time RL examples (Wang and Zhou, 2020; Jiang et al., 2022; Jia and Zhou, 2022b, 2023; Dong, 2024; Dai et al., 2023). A special feature of the current problem is that, because the action set has only two points, 0 and 1, the exploratory control process π is a Bernoulli-distribution valued process with $\mathbb{P}(u_t = 1) = \pi_t = 1 - \mathbb{P}(u_t = 0)$. When exercising such a control, at any t the agent flips a coin having head with probability π_t and tail with probability $1 - \pi_t$, and stops if head appears and continues otherwise.

With a slight abuse of notation, henceforth we still use (X, R) to denote the exploratory states. Since X is not controlled, it is still driven by (2.2) but R now satisfies

$$\mathrm{d}R_t = -KR_t\pi_t\,\mathrm{d}t. \tag{2.16}$$

Then the entropy-regularized exploratory counterpart of problem (2.13) is

$$\bar{v}^{\mathbf{\Pi}}(s, x, r) = \max_{\pi} \bar{J}^{\pi}(s, x, r), \tag{2.17}$$

where the superscript $\mathbf{\Pi}$ is a reminder that we are currently considering an exploratory problem with control randomization and entropy regularization, and the maximization is over all Bernoulli-distribution valued processes taking values in $\{0, 1\}$. The value function is

$$\bar{J}^{\pi}(s, x, r) = \mathbb{E}\left[\int_s^T R_t\Big[Kg(t, X_t)\pi_t - \lambda\mathscr{H}(\pi_t)\Big]\,\mathrm{d}t + R_T g(T, X_T)\,\Big|\, X_s = x, R_s = r\right], \tag{2.18}$$

where

$$\mathscr{H}(\pi) = \pi\log\pi + (1-\pi)\log(1-\pi). \tag{2.19}$$

In the above, $-\mathscr{H}(\pi)$ is the differential entropy of the Bernoulli distribution π, and $\lambda > 0$ is the *exploration weight* or *temperature parameter* representing the trade-off between exploration and exploitation.

Note that the formulation (2.18) does not follow strictly the general formulation in Wang et al. (2020) and those in the subsequent works, because the running entropy regularization term in (2.18) is $-R_t\mathscr{H}(\pi_t)$ instead of $-\mathscr{H}(\pi)$. There are two reasons behind this modification. First, $\bar{v}(t, x, 0) = 0$ in (2.14) for both current and future time–state pairs (t, x) motivates us to expect that $\bar{v}^{\mathbf{\Pi}}(t, x, 0) = 0$ holds as well, which would be untrue if we took $-\mathscr{H}(\pi)$ as the regularizer. Second, with the modification we can reduce the dimension of the problem as follows. Taking the expression of R_t into $\bar{J}^{\pi}(s, x, r)$ to get

$$v^{\mathbf{\Pi}}(s, x) = \max_{\pi_t \in [0,1]} J^{\pi}(s, x), \tag{2.20}$$

where $J^\pi(s,x)$ defined below does not rely on R_t anymore:

$$J^\pi(s,x)=\mathbb{E}\left[\int_s^T e^{-K\int_s^t \pi_v\,\mathrm{d}v}\Big[Kg(t,X_t)\pi_t-\lambda\mathscr{H}(\pi_t)\Big]\,\mathrm{d}t+e^{-K\int_s^T \pi_v\,\mathrm{d}v}g(T,X_T)\;\Big|\; X_s=x\right]. \tag{2.21}$$

Note that we have the following relationships:

$$\bar{J}^\pi(s,x,r)=rJ^\pi(s,x)\;\text{ and }\;v^{\mathbf{\Pi}}(s,x)=\frac{1}{r}\bar{v}^{\mathbf{\Pi}}(s,x,r)=\bar{v}^{\mathbf{\Pi}}(s,x,1)\;\text{ for all }\;r>0. \tag{2.22}$$

Meanwhile, the optimal strategies of (2.17) and (2.20) are identical, which will be shown shortly. Consequently, in the RL algorithms in the next section, to learn the optimal value function $v^{\mathbf{\Pi}}(s,x)$ of the problem (2.20), we will start from the problem (2.17) with initial state $R_0=1$.

By Theorem 2.2, the HJB equation of the problem (2.20) is

$$v_t+\mathscr{L}v+\max_{\pi\in[0,1]}\Big\{-K\pi v+Kg\pi-\lambda(\pi\log\pi+(1-\pi)\log(1-\pi))\Big\}=0,\;\;(t,x)\in[0,T)\times\mathbb{R}^n. \tag{2.23}$$

Since $\pi\mapsto -K\pi v+Kg\pi-\lambda(\pi\log\pi+(1-\pi)\log(1-\pi))$ is strictly concave, the unique maximizer π^* in the above is determined by the first order condition: $-Kv+Kg-\lambda(\log\pi^*-\log(1-\pi^*))=0$, or

$$\pi^*=\frac{1}{1+\exp(\lambda^{-1}K(v-g))}. \tag{2.24}$$

Using this, the HJB equation (2.23) reduces to

$$\exp(-\lambda^{-1}(v_t+\mathscr{L}v))=\exp(\lambda^{-1}(-Kv+Kg))+1. \tag{2.25}$$

By Theorem 2.2, we have

Lemma 2.4. *Suppose v is a classical solution of (2.25) satisfying the terminal condition $v(T,x)=g(T,x)$. Then it is the value function of the problem (2.20).*

Following the same procedure, the HJB equation of (2.17) is

$$\bar{v}_t+\mathscr{L}\bar{v}+\max_{\bar{\pi}\in[0,1]}r\Big\{-K\bar{\pi}\bar{v}_r+Kg\bar{\pi}-\lambda(\bar{\pi}\log\bar{\pi}+(1-\bar{\pi})\log(1-\bar{\pi}))\Big\}=0, \tag{2.26}$$

where $(t,x,r)\in[0,T)\times\mathbb{R}^n\times[0,\infty)$, and the unique maximizer in above is

$$\bar{\pi}^*=\frac{1}{1+\exp(\lambda^{-1}K(\bar{v}_r-g))}. \tag{2.27}$$

By (2.22), $\bar{v}_r^{\mathbf{\Pi}}(s,x,r)=v^{\mathbf{\Pi}}(s,x)$, therefore, $\bar{\pi}^*(s,x,r)=\pi^*(s,x)$, which shows that the optimal strategies of the problems (2.17) and (2.20) are the same.

We end this section by providing an error analysis arising from the penalty approximation (in terms of K) and the exploratory framework (in terms of λ). It is immediate

that the optimal value function v^K of the penalty approximation (which satisfies (2.7)) also satisfies (2.23) with $\lambda = 0$, while the optimal value function $v^{\mathbf{\Pi}}$ of the exploratory control problem (2.20) satisfies (2.23) with a positive λ. By the maximum principle, it is straightforward that $v^K \leqslant v^{\mathbf{\Pi}}$.

Next, we provide a lower bound of v^K. Consider the function $\tilde{v}^\lambda := v^K + \lambda(T-t)\log 2$, which coincides with $v^{\mathbf{\Pi}}$ when $t = T$. Substituting $\tilde{v}^\lambda$ into the LHS of (2.23) yields:

$$\begin{aligned}
&\tilde{v}_t^\lambda + \mathcal{L}\tilde{v}^\lambda + \max_{\pi\in[0,1]}\Big\{-K\pi\tilde{v}^\lambda + Kg\pi - \lambda\Big(\pi\log\pi + (1-\pi)\log(1-\pi)\Big)\Big\} \\
&= v_t^K + \mathcal{L}v^K - \lambda\log 2 \\
&\quad + \max_{\pi\in[0,1]}\Big\{-K\pi v^K - K\pi\lambda(T-t)\log 2 + Kg\pi - \lambda\Big(\pi\log\pi + (1-\pi)\log(1-\pi)\Big)\Big\} \\
&\leqslant v_t^K + \mathcal{L}v^K + \max_{\pi\in[0,1]}\Big\{-K\pi v^K + Kg\pi\Big\} + \max_{\pi\in[0,1]}\{-K\pi\lambda(T-t)\log 2\} \\
&\quad + \max_{\pi\in[0,1]}\Big\{-\lambda\Big(\pi\log\pi + (1-\pi)\log(1-\pi)\Big) - \lambda\log 2\Big\} \leqslant\ 0,
\end{aligned}$$

where we make use of the facts that $v_t^K + \mathcal{L}v^K + \max_{\pi\in[0,1]}\Big\{-K\pi v^K + Kg\pi\Big\} = 0$ and $-\Big(\pi\log\pi + (1-\pi)\log(1-\pi)\Big) \leqslant \log 2$. Thus, again by the maximum principle, we have that $\tilde{v}^\lambda \geqslant v^{\mathbf{\Pi}}$, i.e., $v^K \geqslant v^{\mathbf{\Pi}} - \lambda(T-t)\log 2$.

Theorem 2.5. *For any fixed penalty factor K, we have*

$$v^{\mathbf{\Pi}} - \lambda(T-t)\log 2 \leqslant v^K \leqslant v^{\mathbf{\Pi}},$$

where v^K and $v^{\mathbf{\Pi}}$ satisfy (2.7) and (2.23), respectively. Combined with (2.6), we have

$$-\lambda(T-t)\log 2 \leqslant v^{os} - v^{\mathbf{\Pi}} \leqslant \frac{C}{K},$$

where v^{os} satisfying (2.3) is the optimal value function of the original optimal stopping problem (2.1), and C is a positive constant independent of λ and K.

As a consequence, as we increase the penalty factor K to infinity and decrease the temperature parameter λ to zero, $v^{\mathbf{\Pi}}$ converges uniformly to v^{os}.

3 Reinforcement learning algorithms

We discretize the time horizon $[0, T]$ into a series of equally spaced time intervals $[t_{l-1}, t_l]$, $1 \leqslant l \leqslant L$ with $t_l = l\Delta t$. Given a control policy π, approximate its value function of problem (2.20) at time t_l, i.e., $J^\pi(t_l, \cdot)$, by a neural network (NN) with parameter $\theta_l \in \mathbb{R}^{N_{\theta_l}}$, where N_{θ_l} represents the dimension of θ_l, for $l = 0, \ldots, L-1$.[4] We

[4]For problems with special structures, we can exploit them to directly parameterize the value function instead of resorting to the NN, which could reduce the complexity greatly. For example, Wang and Zhou (2020) and Jiang et al. (2022) use analytical forms of the value functions for parameterizing a pre-committed dynamic mean-variance problem and a dynamic terminal log utility problem, respectively.

denote by $V_{\theta_l}(\cdot)$ this approximated value function at time t_l. Besides, at $t_L = T$ the value function is simply $V_{\theta_L}(\cdot) = g(T, \cdot)$. Finally, (2.22) yields the corresponding approximation of the value function of the problem (2.17).

A typical RL task consists of two steps: policy evaluation and policy improvement. Starting from an initial (parameterized) policy $\pi^{(0)}$, we need to learn the value function under $\pi^{(0)}$ to get an evaluation of this policy. The next step is to find a direction to update the policy to $\pi^{(1)}$ such that the value function under $\pi^{(1)}$ is improved compared to that under $\pi^{(0)}$. We repeat this procedure until optimality or near-optimality is achieved.

The second step, policy improvement, is based on the following theoretical result (which is analogous to (2.24)).

Theorem 3.1 (Policy improvement theorem). *Given an admissible policy π together with its value function $J^\pi(\cdot,\cdot)$, let*

$$\tilde{\pi}(t,x) = \frac{1}{1+\exp\left(\lambda^{-1}K\left(J^\pi(t,x) - g(t,x)\right)\right)}. \tag{3.1}$$

Then $J^\pi(t,x) \leqslant J^{\tilde{\pi}}(t,x)$ for all $(t,x) \in [0,T]\times\mathbb{R}^n$.

So for the current problem, the policy improvement can be computed *analytically* provided that $J^\pi(\cdot,\cdot)$ is accessed for any π. Thus, solving the problem boils down to the first step, policy evaluation.[5] For that step, we take the martingale approach for continuous-time RL developed in Jia and Zhou (2022a).

First, we consider an offline algorithm. Following Jia and Zhou (2022a), denote

$$\mathcal{M}_t := \bar{J}^\pi(t, X_t, R_t) + \int_0^t R_s\Big[Kg(s,X_s)\pi_s - \lambda\mathscr{H}(\pi_s)\Big]\,\mathrm{d}s,$$

which is a martingale. Noting $\bar{J}^\pi(T, X_T, R_T) = R_T g(T, X_T)$, (2.22) and the NN approximation, we get

$$\begin{aligned}
\mathcal{M}_T - \mathcal{M}_t &= \bar{J}^\pi(T,X_T,R_T) - \bar{J}^\pi(t,X_t,R_t) + \int_t^T R_s\Big[Kg(s,X_s)\pi_s - \lambda\mathscr{H}(\pi_s)\Big]\,\mathrm{d}s\\
&= R_T g(T,X_T) - R_t J^\pi(t,X_t) + \int_t^T R_s\Big[Kg(s,X_s)\pi_s - \lambda\mathscr{H}(\pi_s)\Big]\,\mathrm{d}s\\
&= R_T g(T,X_T) - R_t V_\theta(X_t) + \int_t^T R_s\Big[Kg(s,X_s)\pi_s - \lambda\mathscr{H}(\pi_s)\Big]\,\mathrm{d}s.
\end{aligned}$$

The martingale loss function is thus

$$\mathrm{ML}\left(\{\theta_l\}_{l=0}^{L-1}\right) = \frac{1}{2}\mathbb{E}\left[\int_0^T |\mathcal{M}_T - \mathcal{M}_t|^2 dt\right] \approx: \frac{1}{2}\mathbb{E}\left[\sum_{l=0}^{L-1} G_l^2 \Delta t\right], \tag{3.2}$$

[5] Note that this shortcut is specific to the current problem and the approach we take.

where

$$G_l = R_T g(T, X_T) - R_{t_l} V_{\theta_l}(X_{t_l}) + \sum_{j=l}^{L-1} R_{t_j} \Big[K g(t_j, X_{t_j}) \pi_{t_j} - \lambda \mathscr{H}(\pi_{t_j}) \Big] \Delta t. \tag{3.3}$$

For states at discretized time points, we adopt the forward Euler scheme as follows,

$$X_{t_{l+1}} = X_{t_l} + b(t_l, X_{t_l}) \Delta t + \sigma(t_l, X_{t_l}) \Delta W \quad \text{with } \Delta W \sim \mathcal{N}(0, \Delta t), \tag{3.4}$$

and

$$R_{t_{l+1}} = R_{t_l} \left(1 - K \pi_{t_l} \Delta t\right) \quad \text{for } l = 0, \ldots, L-1. \tag{3.5}$$

It is obvious that to minimize the martingale loss, the state trajectories in the whole time horizon are required, which leads to the *offline* RL Algorithm 1. During training, as long as we can collect the realized outcomes of the payoff function g, referred to as the *reward signals* from the environment, the algorithm is applicable even if we do not know the functional form of g.[6]

For *online* learning, we apply the martingale orthogonality condition $\mathbb{E}\left[\int_0^T \xi_t \mathrm{d}\mathcal{M}_t\right] = 0$ for any test function $\xi \in L^2_{\mathcal{F}}([0,T], \mathcal{M})$, again proposed by Jia and Zhou (2022a), where

$$\begin{aligned} &L^2_{\mathcal{F}}([0,T], \mathcal{M}) \\ &:= \left\{ \kappa = \{\kappa_t, 0 \leqslant t \leqslant T\} : \kappa \text{ is } \mathcal{F}_t\text{-progressively measurable and } \mathbb{E} \int_0^T |\kappa_t|^2 \mathrm{d}\langle \mathcal{M} \rangle_t < \infty \right\}. \end{aligned}$$

In particular, a TD(0)-type algorithm results from choosing $\xi_t = \frac{\partial \bar{J}^\pi(t, X_t, R_t)}{\partial \theta}$. The corresponding online updating rule is

$$\theta \leftarrow \theta + \alpha \frac{\partial \bar{J}^\pi(t, X_t, R_t)}{\partial \theta} \mathrm{d}\mathcal{M}_t,$$

where α is the learning rate. For actual implementation where time is discretized, we have at time t_l:

$$\begin{aligned} &\frac{\partial \bar{J}^\pi(t_l, X_{t_l}, R_{t_l})}{\partial \theta_l} \mathrm{d}\mathcal{M}_{t_l} \\ \approx\ &\frac{\partial \left(R_{t_l} V_{\theta_l}(X_{t_l})\right)}{\partial \theta_l} \left\{ R_{t_{l+1}} V_{\theta_{l+1}}(X_{t_{l+1}}) - R_{t_l} V_{\theta_l}(X_{t_l}) + R_{t_l} \left[K g\left(t_l, X_{t_l}\right) \pi_{t_l} - \lambda \mathscr{H}(\pi_{t_l}) \right] \Delta t \right\} \\ \propto\ &\frac{\partial V_{\theta_l}(X_{t_l})}{\partial \theta_l} \left\{ \frac{R_{t_{l+1}}}{R_{t_l}} V_{\theta_{l+1}}(X_{t_{l+1}}) - V_{\theta_l}(X_{t_l}) + \left[K g(t_l, X_{t_l}) \pi_{t_l} - \lambda \mathscr{H}(\pi_{t_l}) \right] \Delta t \right\}. \end{aligned}$$

[6] In some situations, we may also be able to learn g through parameterizations. For instance, in the next section we will show that when g is the payoff function corresponding to the early exercise premium of an American put option, whose exact value is dependent on the unknown market volatility, an accurate approximation of g is available upon training by the martingale approach.

Consequently, the updating rule in our online algorithm is

$$\theta_l \leftarrow \theta_l + \alpha \frac{\partial V_{\theta_l}(X_{t_l})}{\partial \theta_l} \left\{ \frac{R_{t_{l+1}}}{R_{t_l}} V_{\theta_{l+1}}(X_{t_{l+1}}) - V_{\theta_l}(X_{t_l}) + \left[Kg(t_l, X_{t_l})\pi_{t_l} - \lambda \mathscr{H}(\pi_{t_l}) \right] \Delta t \right\} \tag{3.6}$$

for $l = 0, \ldots, L-1$. Algorithm 2 presents the online learning procedure.

Algorithm 1: Offline ML

Inputs: initial states x_0 and $r_0 = 1$, penalty factor K, time horizon T, time step Δt, number of time intervals L, training batch size M, number of training steps N, learning rate α, temperature parameter λ and a good representation of the payoff function $g(\cdot, \cdot)$.

Learning procedure:

Initialize the neural networks $V_{\theta_l}(\cdot)$ for $l = 0, \ldots, L-1$.

for $n = 1, \ldots, N$ **do**

- Generate M state trajectories of $\{X_{t_l}\}_{l=0}^{L}$ with $X_0 = x_0$ by (3.4).
- Calculate the strategy $\{\pi(t_l, X_{t_l})\}_{l=0}^{L}$ according to $\pi(t_l, X_{t_l}) = \frac{1}{1+\exp\left(\frac{K}{\lambda}\left(V_{\theta_l}(X_{t_l}) - g(t_l, X_{t_l})\right)\right)}$.
- Generate M state trajectories of $\{R_{t_l}\}_{l=0}^{L}$ with $R_0 = r_0$ by (3.5).
- Calculate the martingale loss (3.2).
- Update $\{\theta_l\}_{l=0}^{L-1}$ by taking one step stochastic gradient descent with respect to the martingale loss with learning rate α.

end

Finally, we provide some convergence result as a theoretical guarantee of the proposed RL algorithm, assuming that the policy evaluation can be carried out precisely.[7] Starting from a given policy π^0 and its value function J^{π^0}, consider a sequence of policies $\{\pi^n\}_{n=1}^{\infty}$ and the corresponding value functions $\left\{J^{\pi^n}\right\}_{n=1}^{\infty}$, where

$$\pi^{n+1}(t, x) = \frac{1}{1 + \exp\left(\lambda^{-1} K \left(J^{\pi^n}(t, x) - g(t, x)\right)\right)}. \tag{3.7}$$

It follows from the Feynman–Kac formula that

$$\partial_t J^{\pi^n} + \mathscr{L} J^{\pi^n} + H\left(t, x, \pi^n, J^{\pi^n}\right) = 0, \quad J^{\pi^n}(T, x) = g(T, x), \tag{3.8}$$

where $H(t, x, \pi, J^{\pi}) = K\pi(t, x)\left(g(t, x) - J^{\pi}(t, x)\right) - \lambda \mathscr{H}(\pi(t, x))$ is as defined in the proof of Theorem 3.1.

Theorem 3.2 (Convergence rate of policy iteration). *Given any initial guess $J^{\pi^0} \in C_b(\mathbb{R})$ for the optimal exploratory control problem (2.20) under an initial admissible policy*

[7] Here we ignore the impact of time discretization in the presented convergence result to avoid distraction from the central ideas of the paper. For a convergence analysis involving the effects of time discretization and sampling in randomization, see Section 3.2.2 in Dai et al. (2025) for Merton's problem with power utility in a Black-Scholes market, where there is only one unknown model primitive. For more general problems with more complex structures, see a recent paper Jia et al. (2025). This is still an on-going active research topic in the realm of RL for controlled diffusion processes.

Algorithm 2: Online TD(0)

Inputs: initial states x_0 and $r_0 = 1$, penalty factor K, time horizon T, time step Δt, number of time intervals L, training batch size M, number of training steps N, learning rate α, temperature parameter λ and a good representation of the payoff function $g(\cdot, \cdot)$.

Learning procedure:

Initialize the neural networks $V_{\theta_l}(\cdot)$ for $l = 0, \ldots, L-1$.

for $n = 1, \ldots, N$ **do**

  Generate M state trajectories with the identical initial states $X_0 = x_0$ and $R_0 = r_0$.

  for $l = 0, \ldots, L-1$ **do**

    Generate $X_{t_{l+1}}$ by (3.4) for all M state trajectories.

    Calculate the strategy $\pi(t_l, X_{t_l})$ according to $\pi(t_l, X_{t_l}) = \frac{1}{1+\exp\left(\frac{K}{\lambda}\left(V_{\theta_l}(X_{t_l}) - g(t_l, X_{t_l})\right)\right)}$.

    Generate $R_{t_{l+1}}$ by (3.5) for all M state trajectories.

    Update θ_l by stochastic approximation (3.6) with learning rate α.

  end

end

π^0, generate the sequence $\left\{\pi^n, J^{\pi^n}\right\}_{n=1}^{\infty}$ alternatingly by (3.7) and (3.8). Suppose the payoff function g is uniformly bounded. Then, for any fixed penalty factor K and temperature parameter λ, we have

$$\left\|v^{\mathbf{\Pi}} - J^{\pi^{n+1}}\right\|_\infty \leqslant \frac{(2KT)^n}{n!} 2[(K\|g\|_\infty + \lambda \log 2)T + \|g\|_\infty].$$

In particular, it follows that J^{π^n} converges to $v^{\mathbf{\Pi}}$ as $n \to \infty$.

4 Simulation studies: American put options

In this section, we present simulation experiments in which an RL agent learns how to exercise finite-horizon American put options. A put option can be formulated as the following optimal stopping problem under the risk-neutral measure

$$\max_{\tau \in [0,T]} \mathbb{E}\left[e^{-\rho\tau}(\Gamma - X_\tau)^+\right], \tag{4.1}$$

where the stock price follows a one-dimensional geometric Brownian motion: $\mathrm{d}X_t = \rho X_t \,\mathrm{d}t + \sigma X_t \,\mathrm{d}W_t$ with ρ being the risk-free rate and σ being the volatility of the stock price, and the (time-invariant) payoff function is $g(x) = (\Gamma - x)^+$ with Γ being the strike price. In the RL setting, the volatility σ is unknown to the agent.

Due to the presence of the exponential time discounting in the objective function, (3.3) and (3.6) in Algorithms 1 and 2 need to be respectively modified to (refer to Section 5.1

in Jia and Zhou, 2022a):

$$G_l = e^{-\rho T} R_T g(X_T) - e^{-\rho t_l} R_{t_l} V_{\theta_l}(X_{t_l}) + \sum_{j=l}^{L-1} e^{-\rho t_j} R_{t_j} \Big[K g(X_{t_j}) \pi_{t_j} - \lambda \mathscr{H}(\pi_{t_j}) \Big] \Delta t,$$

and

$$\theta_l \leftarrow \theta_l + \alpha \frac{\partial V_{\theta_l}(X_{t_l})}{\partial \theta_l} \left\{ \frac{R_{t_{l+1}}}{R_{t_l}} V_{\theta_{l+1}}(X_{t_{l+1}}) - V_{\theta_l}(X_{t_l}) + \left[K g(X_{t_l}) \pi_{t_l} - \lambda \mathscr{H}(\pi_{t_l}) \right] \Delta t - \rho V_{\theta_l}(X_{t_l}) \right\}.$$

In our experiments, we use the same parameters as in Becker et al. (2021) and Dong (2024) for the simulator, i.e., the initial stock price $x_0 = 40$, strike price $\Gamma = 40$, risk-free rate $\rho = 0.06$, volatility $\sigma = 0.4$ (but unknown to the agent), and maturity time $T = 1$. We use the implicit finite difference method to solve the penalty approximation of the HJB equation of the optimal stopping problem, namely (2.5), and obtain the omniscient option price which equals 5.317, as well as the free boundary $X_f(t)$ which is shown as the blue line in Figure 2.

Finally, Table 1 lists some key (hyper)parameter values in our experiments.

Table 1: Benchmark values in RL training.

	Notation	Benchmark value
Number of time mesh grids	L	50
Penalty factor	K	10
Temperature parameter	λ	1
Learning rate	α	0.01
Training batch size	M	2^{10}
Testing batch size	M_{test}	2^{18}
Number of training steps	N	1000 or 5000

4.1 Learning early exercise premium

Early exercise premium is the difference in value between an American option and its European counterpart. In our setting, it is defined as

$$\tilde{v}(t,x) := v(t,x) - V_E(t,x), \tag{4.2}$$

where $V_E(t,x)$ is the value of the European put option with the same strike. The associated payoff function is

$$\tilde{g}(t,x) := g(x) - V_E(t,x). \tag{4.3}$$

Instead of learning the value of the American put option, we set out to learn the value of the early exercise premium. An advantage of doing so is that it removes the singularity of the terminal condition, which improves the learning performance around the maturity time.

In the following subsections, we will learn the early exercise premium $\tilde{v}$ by solving the related entropy-regularized problem with payoff $\tilde{g}$. However, the unknown model primitive σ renders an unknown V_E embedded in $\tilde{g}$; so the problem has an unknown payoff function. We will apply policy evaluation to learn an approximated $\tilde{g}^{\phi}$; refer to Appendix B for details.

The architecture of the fully connected feedforward NNs used in our study is the same as that in Dong (2024): At each decision time point $t \in \{t_l : l = 0, 1, \ldots, L-1\}$, we construct one neural network, which consists of one input layer, two hidden layers with 21 neurons followed by an ReLU activation function and one output layer. Following Becker et al. (2021), at time t_l we take the inputs to be the state and the approximated payoff value, i.e., X_{t_l} and $\tilde{g}^{\phi}(t_l, X_{t_l})$. The output w_{θ_l} of the NN at time t_l is the difference between the early exercise premium and the associated payoff function, i.e., $\tilde{v}(t_l, X_{t_l}) - \tilde{g}(t_l, X_{t_l})$, which is equal to $v(t_l, X_{t_l}) - g(t_l, X_{t_l})$ in theory by (4.2) and (4.3). Write

$$w_{\theta_l}\left(X_{t_l}, \tilde{g}^{\phi}(t_l, X_{t_l})\right) := V_{\theta_l}(X_{t_l}) - \tilde{g}^{\phi}(t_l, X_{t_l}), \tag{4.4}$$

where $V_{\theta_l}(\cdot)$ is the NN approximator for the value function of the exploratory problem regarding $\tilde{v}$, namely problem (B.6) at time t_l. Finally, after the input layer and the output layer and before every activation layer, we add a batch normalization layer to facilitate training.

4.2 Learned value functions

With the parameter values given in Table 1, we implement training for 1000 steps by the offline ML Algorithm 1 and for 5000 steps by the online TD(0) Algorithm 2. These are carried out within the PyTorch framework. Each algorithm spends about 0.5s for one step training on a Windows 11 PC with an Intel Core i7-12700 2.10 GHz CPU and 32GB RAM by CPU computing only.

Figure 1 shows the learning results by Algorithms 1 and 2 regarding the exploratory value functions at six selected decision time points $t \in \{t_l : l = 0, 10, 20, 30, 49\} = \{0, 0.2, 0.4, 0.6, 0.8, 0.98\}$. In each panel, we plot five lines with the stock price ranging from zero to twice the initial stock price. The solid red line (with label "Option") shows the theoretical (omniscient) values of $v(t,x) - g(x)$ derived by finite difference. The dotted solid blue, orange and green lines (with label "Dong (2024)", "ML" and "TD(0)") are respectively the outputs $v - g$ from the NN by the TD error algorithm in Dong (2024) and by the ML and TD(0) algorithms in this paper. The dotted yellow vertical line (with label "S^*") refers to the specific stock price that separates the stopping and continuation regions at each time, which is also derived by finite difference.

As observed, all the algorithms result in values close to the oracle value, especially when the stock price is above S^*. The results of Dong (2024) have relatively high errors

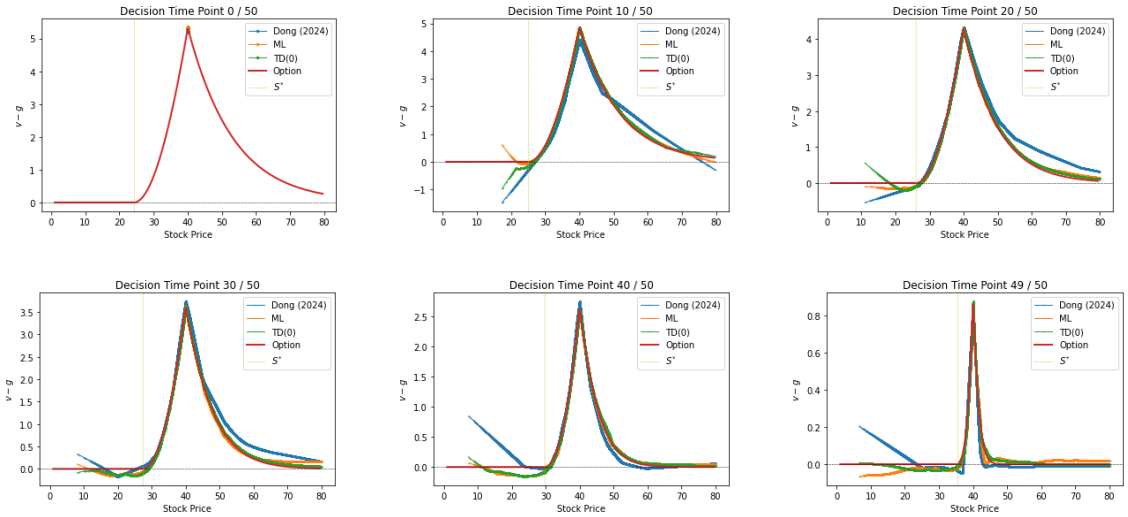


Figure 1: Learning results of $v - g$ under the TD algorithm in Dong (2024), the offline ML algorithm and the online TD(0) algorithm in this paper at six selected decision time points: $\{0, 10, 20, 30, 40, 49\}$-th discretized time points. The solid red lines are the theoretical values of $v - g$ and the dotted yellow vertical lines specify the stock price on the theoretical free boundary, both derived by finite difference. The training parameters of Dong (2024) are specified in Section 4.6.

when approaching the terminal time, which is the main motivation for learning early exercise premium instead in this paper. Except for extremely low stock prices, which are rarely reached by our simulated stock price trajectories leading to too few data for adequate learning, the yellow dotted vertical lines well separate the positive values from the non-positive ones. This result is desired when we construct the execution policy in the next subsection.

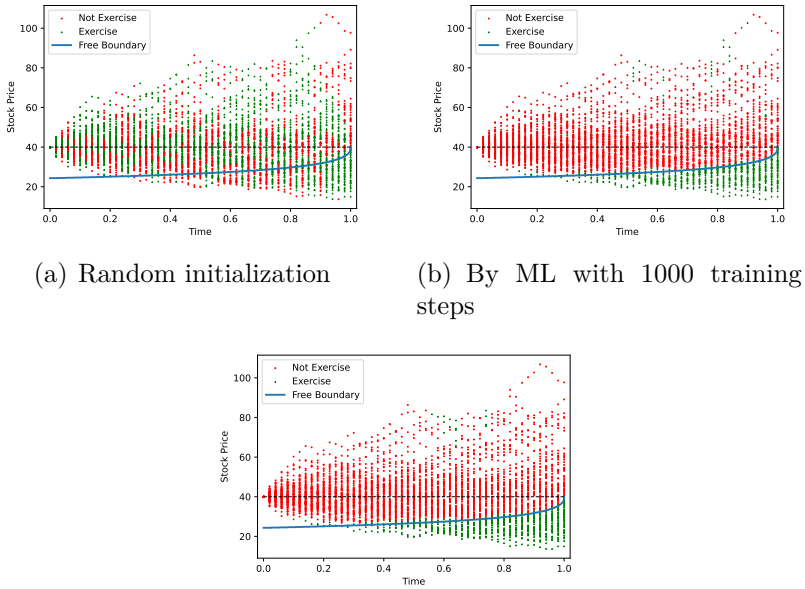
4.3 Execution policy

After having learned the exploratory value functions with well-trained NNs, we can further obtain the stopping probabilities by (2.24). However, when it comes to *real* execution regarding stopping or continuing, it does not seem reasonable to actually toss a biased coin. To wit, randomization is necessary for *training*, but not for *execution*. For the latter, one way is to use the mean of the randomized control (Dai et al., 2023), which is, however, inapplicable in the current 0–1 optimal stopping context. For example, a mean of 0.8 is outside of the control value set $\{0, 1\}$.

Recall that for the classical (non-RL) optimal stopping problem of the American put option, the stopping region is characterized by $\mathcal{S} = \{(t, x) : v(t, x) = g(x)\}$ while the continuation region is characterized by $\mathcal{C} = \{(t, x) : v(t, x) > g(x)\}$. Thus, it is reasonable to base the sign of $w_{\theta_l} = V_{\theta_l} - \tilde{g}^{\phi}$ in (4.4) to make an exercise decision at time t_l. Theoretically, by (2.24), when the penalty factor K goes to infinity, π^* tends to 1 and

0 when $w_{\theta_l} < 0$ and $w_{\theta_l} > 0$, respectively. However, K is finite in our implementation, π^* in (2.24) is more likely to be neither 0 nor 1; but 0.5 is an appropriate threshold to divide between $w_{\theta_l} < 0$ and $w_{\theta_l} > 0$. In Figure 2, we randomly choose 100 stock price trajectories from the testing dataset, and classify the learned stopping strategies into two groups based on either $\pi^* \geqslant 0.5$ or $\pi^* < 0.5$. When the former occurs the agent chooses to exercise the option, which is represented by the green triangle; otherwise the agent does not exercise, represented by the red circle. Besides, the blue line corresponds to the free boundary of this American put option $X_f(t)$. As Figure 2 shows, after sufficient training, both algorithms effectively classify almost all the exercise/non-exercise time–price pairs.

Next, we calculate the accuracy rate of classification at each time point as $\frac{\#\{\text{correctly classified sample points}\}}{M_{\text{test}}}$. It turns out that both algorithms achieve accuracy rates of higher than 95% for all discretized time points.[8]



(a) Random initialization

(b) By ML with 1000 training steps

(c) By TD(0) with 5000 training steps

Figure 2: Stopping strategies learned from the two algorithms. The blue line is the theoretical free boundary of the American put option derived by finite difference. In each panel, there are 100 stock price trajectories with 50 discretized time points. The red circles correspond to the cases for which the stopping probability is less than half, while the green triangles to the cases for which the stopping probability is larger than or equal to half.

[8]One never exercises an American put option when the stock price is greater than the strike price; so actually we only need to study the points below the black dotted lines in Figure 2. With this in mind the accuracy rates achieved are even higher.

4.4 Option price

With the execution policy determined in Section 4.3, how to compute the option price? In this subsection, we introduce two ways to do it based on the learned exploratory value functions in Section 4.2.

The first way is to calculate the value of the original optimal stopping problem with the stopping time determined by the learned exploratory value functions in Section 4.2, namely

$$\mathcal{P}_{\text{stopping}} := \mathbb{E}\left[e^{-\rho\tau} g(X_\tau)\right], \tag{4.5}$$

where

$$\tau := \inf\left\{t_l : V_{\theta_l}(X_{t_l}) \leqslant \tilde{g}^{\phi}(t_l, X_{t_l}),\ l \in \{0, 1, ..., L\}\right\} \wedge T \tag{4.6}$$

with the convention that the infimum of an empty set is infinity.

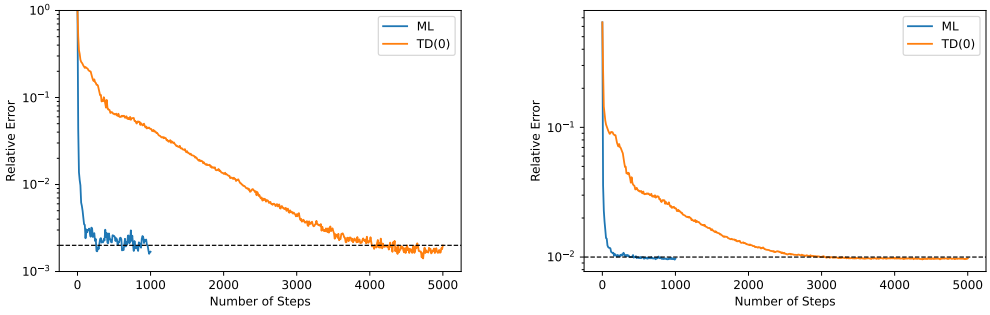
The second way is to calculate the value (2.12) at the initial state $s = 0, x = x_0$ under the execution policy determined in Section 4.3, namely

$$\mathcal{P}_{\text{control}} := \mathbb{E}\left[\int_s^T K e^{-K\int_s^t u_r \,\mathrm{d}r} g(t, X_t) u_t \,\mathrm{d}t + e^{-K\int_s^T u_r \,\mathrm{d}r} g(T, X_T) \,\middle|\, X_s = x\right], \tag{4.7}$$

where

$$u(t, x) = 1 \text{ if } V_{\theta_l}(x) \leqslant \tilde{g}^{\phi}(t, x), \text{ and 0 otherwise, for } (t, x) \in [0, T) \times \mathbb{R}. \tag{4.8}$$

Recalling that the omniscient option price is 5.317, we define the relative error of a learned price $\mathcal{P}$ to be $\frac{|\mathcal{P}-5.317|}{5.317}$. We calculate the relative error on the testing dataset every 10 training steps. The learning curves of option price from the two methods described above are plotted in Figure 3.



(a) $\mathcal{P}_{\text{stopping}}$ (b) $\mathcal{P}_{\text{control}}$

Figure 3: Learning curves of option price $\mathcal{P}_{\text{stopping}}$ defined in (4.5) and $\mathcal{P}_{\text{control}}$ defined in (4.7) by offline ML (blue line) and online TD(0) (orange line) algorithms. We train the ML algorithm for 1000 steps and TD(0) algorithm for 5000 steps. Values of model parameters used for training are specified in Table 1.

First, we find that the ML algorithm has faster convergence than the online TD(0). This is because the latter needs to update parameters at every small time steps. Second,

both algorithms eventually achieve very low relative errors (about 0.2% and 1% on average for $\mathcal{P}_{\text{stopping}}$ and $\mathcal{P}_{\text{control}}$, respectively). Thus, the RL algorithms prove to be effective in terms of learning the option price without knowing the true value of the volatility σ.

We observe that the relative errors in the second method converge to the level about 1%. This is due to the finite value of the penalty factor $K = 10$ chosen in Table 1. In the next subsection, we will show that as K increases, the relative error can be further reduced.

4.5 Sensitivity analysis

In this subsection, we carry out a sensitivity analysis on the impacts of the values of the penalty factor K and the temperature parameter λ on learning.

4.5.1 Penalty factor

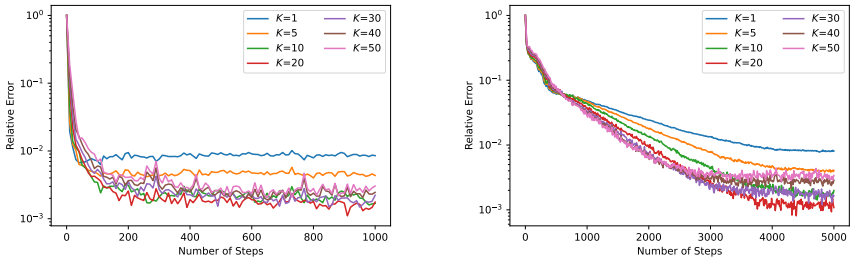
We examine the effect of penalty factor on the learned option price in this subsection. First, from a numerical implementation perspective, it follows from (3.5) that $0 < K \leqslant \frac{1}{\Delta t} = 50$ in our setting, since the process R_t cannot become negative. We pick $K \in \{1, 5, 10, 20, 30, 40, 50\}$ and keep the other model parameter values unchanged as in Table 1. Figure 4 shows the learning curves of option prices with a combination of the two algorithms and the two alternative ways. As expected, the higher the value of K, the less the relative error achieved by both algorithms. However, when K increases to a certain level, a further increase no longer helps decrease the error, in which case the sampling error dominates.

4.5.2 Temperature parameter

The temperature parameter controls the weight on exploration, and thus is important for learning. Figure 5 shows the learning curves of option prices when $\lambda \in \{0.1, 0.5, 1, 3, 5, 10\}$, while keeping the other model parameter values as in Table 1.[9] We observe that in general, the smaller the temperature parameter, the more accurate but the lower the learning speed for convergence, which is especially evident with the TD(0) algorithm.[10] An implication of this observation is that if the computing resource is limited so the number of training steps is small, we can increase λ to accelerate convergence. Intriguingly, in Figure 5, the errors are not monotonic in λ when $\lambda \leqslant 1$ in the two cases using $\mathcal{P}_{\text{stopping}}$. We conjecture that it is because the effect of λ is dominated by the sampling error when λ is sufficiently small. However, overall, the differences are around 0.3% for $\lambda \in \{0.1, 0.5, 1\}$, which are sufficiently small.

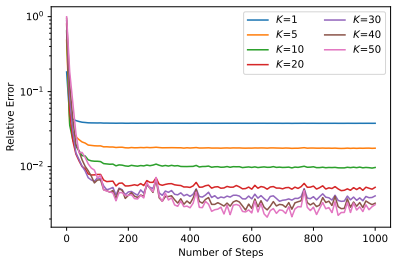
[9] In Appendix C, we also present the results for a higher penalty factor $K = 50$.

[10] Similar results are also noted in Dong (2024).

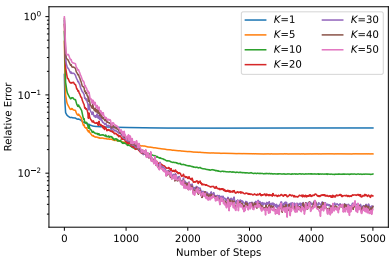


(a) $\mathcal{P}_{\text{stopping}}$ by ML

(b) $\mathcal{P}_{\text{stopping}}$ by TD(0)



(c) $\mathcal{P}_{\text{control}}$ by ML



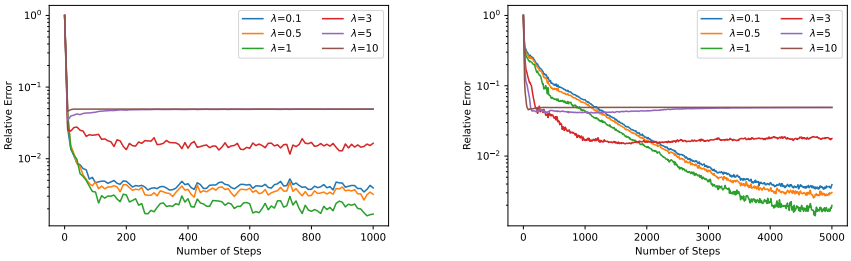
(d) $\mathcal{P}_{\text{control}}$ by TD(0)

Figure 4: Learning curves of option prices $\mathcal{P}_{\text{stopping}}$ defined in (4.5) and $\mathcal{P}_{\text{control}}$ defined in (4.7) by offline ML and online TD(0) algorithms. In each panel, the penalty factor K takes value from the set $\{1, 5, 10, 20, 30, 40, 50\}$ while other parameter values are set in Table 1. We train the ML algorithm for 1000 steps and TD(0) algorithm for 5000 steps. In all cases, the relative errors decrease in K.

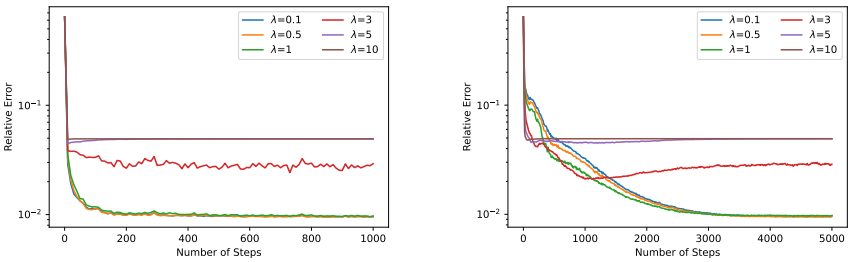
4.6 Comparison with Dong (2024)

Both Dong (2024) and this paper study RL for optimal stopping, but there are major differences as discussed in the introduction section. In particular, there is a considerable technical difference in terms of the entropy terms introduced in the respective RL objectives. It turns out that, when formulated in our setting with the penalty factor K, the decision variable $\tilde{\pi}$ in Dong (2024) is the product of K and the probability of stopping in our paper, i.e., $\tilde{\pi} = K\pi$. The entropy used in Dong (2024) is $R(\tilde{\pi}) := \tilde{\pi} - \tilde{\pi}\log\tilde{\pi} = R(K\pi) = K\pi - K\pi\log(K\pi)$, whereas we take the differential entropy $-\mathscr{H}(\pi) = -\pi\log\pi - (1-\pi)\log(1-\pi)$. Dong (2024) explains that $R(\tilde{\pi})$ is proposed by assuming $\tilde{\pi}\Delta t$ to be the probability of stopping in a Δt-length time interval and then integrating the differential entropy $-\mathscr{H}(\tilde{\pi}\Delta t) = -\tilde{\pi}\Delta t\log\Delta t + (\tilde{\pi} - \tilde{\pi}\log\tilde{\pi})\,\Delta t + o(\Delta t)$ over the whole time horizon while dropping the first divergence term. In other words, it is not the exact differential entropy of the randomized strategy as in our paper.

Next, we compare the performances of the algorithms between the two papers. We use $\mathcal{P}_{\text{stopping}}$ defined in (4.5) for computing option price, as it is the one taken by Dong (2024). For a fair comparison, we use the TD error employed in Dong (2024) as the criterion for performance and we learn the option price instead of the early exercise premium. We



(a) $\mathcal{P}_{\text{stopping}}$ by ML (b) $\mathcal{P}_{\text{stopping}}$ by TD(0)



(c) $\mathcal{P}_{\text{control}}$ by ML (d) $\mathcal{P}_{\text{control}}$ by TD(0)

Figure 5: Learning curves of $\mathcal{P}_{\text{stopping}}$ defined in (4.5) and $\mathcal{P}_{\text{control}}$ defined in (4.7) by offline ML and online TD(0) algorithms. In each panel, the penalty factor K is fixed to 10 and the temperature parameter λ takes value from the set $\{0.1, 0.5, 1, 3, 5, 10\}$ while other parameter values are chosen by Table 1. We train the ML algorithm for 1000 steps and TD(0) algorithm for 5000 steps.

implement training for 1000 steps for both algorithms and repeat the training for 10 times. Because the scales of the entropy terms used in the two papers are considerably different, we train the two algorithms for different combinations of temperature parameter $\lambda \in \{10, 1, 0.1, 0.01, 0.001, 0.0001\}$ and learning rate $\alpha \in \{0.1, 0.01, 0.001, 0.0001\}$. The other parameters are kept the same as in Table 1 and the simulated stock price samples are the same for both algorithms. We then check the learning speeds of the two algorithms. Since the theoretical option price is 5.317, we examine the learning speeds for all the cases of (λ, α) in which the learned option prices are at least 5.29 for a meaningful comparison. We find that our algorithm is always faster than Dong (2024)'s. Figure 6 is an illustration with $(\lambda, \alpha) = (0.001, 0.01)$ (these are the model parameters corresponding to the solid blue line in Figure 1) in Dong's algorithm and $(\lambda, \alpha) = (1, 0.1)$ in our algorithm. The figure shows that both algorithms achieve very low relative errors (0.39% for our method and 0.44% for Dong's method) after 1000 training steps, but our algorithm is around 200 steps faster in convergence.

The simulation study reported so far is for an American option, whose price is evaluated under the *risk-neutral* probability. However, the underlying stock price is in general *not* observable in the risk-neural world; hence pricing American options is not the best example for illustrating a data-driven approach such as RL. Here, we present another

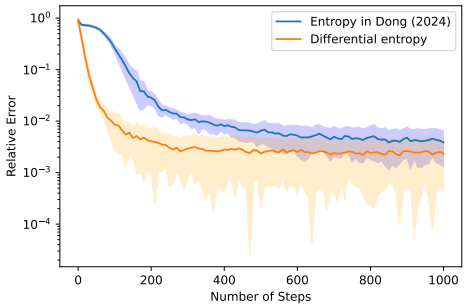
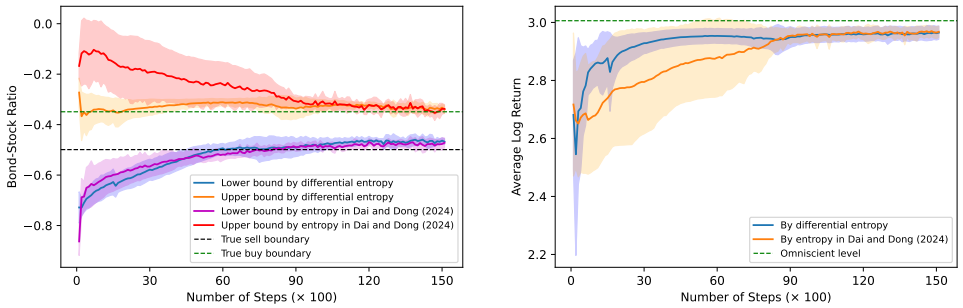


Figure 6: Learning curves of option price from our algorithm and Dong (2024)'s. We repeat the training for 10 times to calculate the standard deviations of the learned option prices, which are represented as the shaded areas. The width of each shaded area is twice the corresponding standard deviation. Note that the vertical axis uses a logarithmic scale.

application to demonstrate the effectiveness of our method in which the physical probability measure instead of the risk-neutral one is used (and thus all the relevant data are observable). Specifically, Dai and Dong (2024) consider the problem of learning an optimal investment policy to maximize a portfolio's terminal log return in the presence of proportional transaction costs, and turn it into an equivalent Dynkin game which involves optimal stopping decisions of two players. The latter is naturally formulated under the physical measure, and suitable for RL study under the frameworks of both Dong (2024) and this paper; refer to Appendix D for details. Dai and Dong (2024) extend the RL algorithm of Dong (2024) to learn an optimal investment policiy with transaction costs. With the same model parameters as in Dai and Dong (2024) and a penalty factor $K = 5$ — which is an appropriate choice as specified in Section 4.5.1 when the time horizon $[0, T]$ with $T = 1$ is equally divided into 10 time intervals (so that $\Delta t = 1/10$, and there are two stopping times in the Dynkin game) — we learn how to trade in the Black–Scholes market under the penalty formulation. We then compare the learning results for the free boundaries and the average log return when applying the learned strategies over a 20-year horizon. Similar to the case of the American put option, we choose temperature parameter $\lambda \in \{10^{-2}, 10^{-4}, 10^{-6}\}$ and learning rate $\alpha \in \{0.1, 0.05, 0.01, 0.005\}$, and keep the other training parameters and structures of neural networks the same as in Dai and Dong (2024), with 15000 training steps for both algorithms. As noted, in this example the stock price samples are simulated under the physical measure. Our experiments show that among the 12 combinations of λ and α, Dai and Dong (2024)'s algorithm achieves higher average log return in the last training step than our algorithm for 2 times, while we achieve better results for 10 times. For comparing the learning speeds, note that the omniscient level of average log return in the 20-year horizon under the testing dataset is around 3.00; so we choose the cases of (λ, α) in which the two algorithms achieve average log return values of greater than 2.95. We find that $(\lambda, \alpha) = (10^{-6}, 0.05)$ and $(\lambda, \alpha) = (10^{-2}, 0.01)$ result in the fastest convergence of average log return for Dai and

Dong (2024)'s algorithm and ours, respectively. Figure 7 plots the corresponding learning curves of the initial free boundaries and average log returns. Figure 7(a) shows that the convergence of the sell boundary is almost identical for the two algorithms, while the convergence of the buy boundary is faster under our algorithm. Figure 7(b) indicates that the convergence of average log return is around 5000 steps faster under our algorithm. It is interesting to note that the trivial buy-and-hold strategy leads to an average log return value of 2.5; so both the RL algorithms indeed help the agent "beat the market".



(a) Free boundaries at the initial time

(b) Log return

Figure 7: Learning curves of free boundaries and average log return from our algorithm and Dai and Dong (2024)'s. We repeat the training for 10 times to calculate the standard deviations of the learned free boundaries and average log return, which are represented as the shaded areas. The width of each shaded area is twice the corresponding standard deviation.

One reason for the aforementioned difference in performance may be that compared to the differential entropy in our algorithm, the scale of the entropy term in Dong (2024) is larger, causing greater biases in learning the optimal stopping value. Specifically, the unnormalized negentropy $R(\tilde{\pi}) \to -\infty$ as $\tilde{\pi} \to \infty$, whereas the differential entropy $-\mathscr{H}(\pi) \in [0, \log 2]$. The unboundedness of $R(\tilde{\pi})$ results in a convergence rate of $\frac{(CTe^{M/\lambda})^n}{n!}$ for policy iteration (see Theorem 3.4 in Dong (2024)), where the proof implies that C is a constant typically set to 2, and M depends on the temperature parameter λ. By contrast, Theorem 3.2 shows a convergence rate of $\frac{(2KT)^n}{n!}$ for our algorithm. To achieve sufficient accuracy, the penalty factor K in our algorithm is typically chosen as 50, while the temperature parameter in Dong's algorithm must be sufficiently small, e.g., $\lambda \leqslant 10^{-3}$. The dependence of M on λ in Dong (2024) is complex, as the unnormalized negentropy $R(\tilde{\pi})$ can take both positive and negative values. Nevertheless, based on the early-stage learning results shown in Figures 6 and 7(b), as well as the theoretical values—approximately 5.32 and 3.00, respectively, for the two problems—we have $M > 1$. Consequently, for small n, $\frac{(CTe^{M/\lambda})^n}{n!} \gg \frac{(2KT)^n}{n!}$, which partially explains the faster learning exhibited by our algorithm in the early training steps. As n increases, the factorial in the denominator dominates, and both algorithms eventually converge.

In addition, in Dong (2024)'s algorithm, the use of the unnormalized negentropy yields

an optimal intensity $\tilde{\pi} = \exp\left(-\frac{V_\theta - g}{\lambda}\right)$, which takes values in $[0, \infty)$. However, for a given time step size Δt, $\tilde{\pi}$ must be truncated at $\frac{1}{\Delta t}$ if $\exp\left(-\frac{V_\theta - g}{\lambda}\right) > \frac{1}{\Delta t}$, since $\tilde{\pi}\Delta t$ is interpreted as the probability of stopping in his formulation. This situation arises when the θ-parameterized neural network generates a negative $V_\theta - g$ and the temperature parameter is small, such as $\lambda \leqslant 10^{-3}$ in the numerical examples. As a result, the loss function minimized by stochastic gradient descent is not the exact TD error in his formulation (see equation (4.4) in Dong (2024)), introducing a bias into the learning process. By contrast, our algorithm employs differential entropy resulting in an optimal probability $\pi \in [0, 1]$, and the loss function minimized by stochastic gradient descent corresponds exactly to the TD error in our formulation. This may be another reason why our algorithm exhibits faster convergence.

5 Simulation studies: High-dimensional examples

Other than the model-free approach that mitigates the model estimation errors, another significant advantage of RL is its ability to scale to high-dimensional problems. We demonstrate this with two examples in this section.

5.1 American geometric average put-type option

The following example is adapted from Section 4.3.2.1 in Becker et al. (2021) and concerns the pricing of an American geometric average put-type option on up to 200 distinguishable stocks in a Black–Scholes market.

As in Becker et al. (2021), we set the dimension (the number of stocks) $d \in \{40, 80, 120, 160, 200\}$, the initial stock price $x_0 = 100^{1/\sqrt{d}}$ for all the stocks, the time horizon $T = 1$, the strike price $\Gamma = 95$, the risk-free rate $\rho = 0.6$, the volatility $\sigma_i = \min\{0.04[(i-1) \bmod 40], 1.6 - 0.04[(i-1) \bmod 40]\}$ and dividend yield $\delta_i = \rho - \frac{\sum_{i=1}^d \sigma_i^2}{d^2}(i - \frac{1}{2}) - \frac{1}{5\sqrt{d}}$ for each $i \in \{1, 2, \ldots, d\}$. Denote by $X_t^{(k)}$ the price of the k-th stock at time t. The optimal stopping problem is to

$$\max_{\tau \in [0,T]} \mathbb{E}\left[e^{-\rho\tau}\left(\Gamma - \left[\prod_{k=1}^{d} |X_\tau^{(k)}|^{1/\sqrt{d}}\right]\right)^+\right].$$

In our simulation studies, we set the penalty factor K to be 100 (which corresponds to the time mesh grid $\Delta t = 1/100$ as discussed in Section 4.5.1), the temperature parameter $\lambda = 1$, the training batch size to be 2^{10}, and the testing batch size to be 2^{17}. The initial learning rate is set to be 0.05 and is decreased by a factor 1/10 every 300 training steps. We train 1000 steps under the offline ML Algorithm 1. In Table 2, we show the learning results for different numbers d of underlying stocks. We test the learned price every 10 training steps; so we have 100 observations in total for each d. The final observation is

Dimension d	Last learned price by our method	In Becker et al. (2021)	Runtime in sec.
40	6.512	6.512	1838
80	6.523	6.509	2956
120	6.504	6.507	4094
160	6.507	6.504	6042
200	6.493	6.501	7394

Table 2: Comparison of learned prices and runtime for different values of dimension d. The initial stock price $x_0 = 100^{1/\sqrt{d}}$, the time horizon $T = 1$, the strike price $\Gamma = 95$, the risk-free rate $\rho = 0.6$, the volatility $\sigma_i = \min\{0.04[(i-1) \bmod 40], 1.6 - 0.04[(i-1) \bmod 40]$ and dividend yield $\delta_i = \rho - \frac{\sum_{i=1}^d \sigma_i^2}{d^2}(i - \frac{1}{2}) - \frac{1}{5\sqrt{d}}$ for each $i \in \{1, 2, \ldots, d\}$. The payoff function when exercising the option is $g(x) = \left(\Gamma - \left[\prod_{k=1}^d |x^{(k)}|^{1/\sqrt{d}}\right]\right)^+$. The benchmark value for the price of this option is 6.545.

listed in the second column of Table 2.

It is worth noting that pricing this high-dimensional American geometric average put-type option is equivalent to pricing a one-dimensional American put option that is independent of the dimension.[11] Hence we can easily obtain the corresponding omniscient values: The theoretical price of its European counterpart is 3.565, and the benchmark value for the price of this one-dimensional American put option calculated using the binomial tree method on Smirnov's website (https://www.math.columbia.edu/~smirnov/options13.html) with 20000 nodes is 6.545. It turns out that our RL algorithm achieves high early exercise premium and is effective for pricing the high-dimensional option with comparable high accuracy compared to the benchmarks in Becker et al. (2021). Moreover, our algorithm is scalable, as Table 2 shows a sublinear growth in runtime with respect to the dimension.

In the above experiment, the risk-free rate $\rho = 0.6$ taken from Becker et al. (2021) is unrealistically large. We now select a more reasonable value and adjust other parameters so that the equivalent one-dimensional American put option remains independent of $d \in \{40, 80, 120, 160, 200\}$. Specifically, we set the risk-free rate $\rho = 0.06$, the volatility $\sigma_i = \frac{1}{\sqrt{10}} \min\{0.04[(i-1) \bmod 40], 1.6 - 0.04[(i-1) \bmod 40]\}$ and dividend yield $\delta_i = \rho - \frac{\sum_{i=1}^d \sigma_i^2}{d^2}(i - \frac{1}{2}) - \frac{1}{50\sqrt{d}}$ for each $i \in \{1, 2, \ldots, d\}$. With these modifications, the equivalent one-dimensional American put option has a risk-free rate of 0.06, a dividend yield of 0.02932, and a volatility of 0.14615. We set the strike price to $\Gamma = 110$. The theoretical price of the corresponding European option is 9.649 according to the Black-

[11] Specifically, for the equivalent one-dimensional American put-type option problem, the stock price process in a Black–Scholes market setting has an initial price of 100, a risk-free rate of ρ, a dividend yield of $\rho - \frac{\sum_{i=1}^d \sigma_i^2}{2d} - \frac{1}{5}$, and a volatility of $\frac{1}{\sqrt{d}}\sqrt{\sum_{i=1}^d \sigma_i^2}$ under the particular choices of σ_i and δ_i in the example. Notably, $\frac{\sum_{i=1}^d \sigma_i^2}{d}$ remains constant when the dimension is a multiple of 40; see Section 4.3.2.1 and Proposition 4.3 in Becker et al. (2021).

Dimension d	Last learned price by our method	Runtime in sec.
40	10.828	1916
80	10.803	2948
120	10.813	4141
160	10.809	5665
200	10.795	7732

Table 3: Comparison of learned prices and runtime for different values of dimension d. The initial stock price $x_0 = 100^{1/\sqrt{d}}$, the time horizon $T = 1$, the strike price $\Gamma = 110$, the risk-free rate $\rho = 0.06$, the volatility $\sigma_i = \frac{1}{\sqrt{10}} \min\{0.04[(i-1) \bmod 40], 1.6 - 0.04[(i-1) \bmod 40]$ and dividend yield $\delta_i = \rho - \frac{\sum_{i=1}^d \sigma_i^2}{d^2}(i - \frac{1}{2}) - \frac{1}{50\sqrt{d}}$ for each $i \in \{1, 2, \ldots, d\}$. The payoff function when exercising the option is $g(x) = \left(\Gamma - \left[\prod_{k=1}^d |x^{(k)}|^{1/\sqrt{d}}\right]\right)^+$. The benchmark value for the price of this option is 10.816.

Scholes formula, and the benchmark value for the one-dimensional American put option, calculated using the binomial tree method on Smirnov's website with 20000 nodes, is 10.816. Table 3 presents the learning results obtained by our ML algorithm for different dimensions, which again demonstrate the high accuracy and scalability of the algorithm.

5.2 Optimally stopping a fractional Brownian motion

This example is adapted from Section 4.3 in Becker et al. (2019) and Section 4.2.3 in Dong (2024), and is an optimal stopping problem for a fractional Brownian motion with Hurst parameter $H \in (0, 1]$. By augmenting the state variable to include the entire history of the process, the problem becomes Markovian, and numerically a high-dimensional optimal stopping problem.

Specifically, a fractional Brownian motion with Hurst parameter $H \in (0, 1]$ is a continuous, centered Gaussian process $(W_t^H)_{t \geqslant 0}$ with covariance structure

$$\mathbb{E}[W_t^H W_s^H] = \frac{1}{2}\left(t^{2H} + s^{2H} - |t-s|^{2H}\right), \quad t, s \geqslant 0.$$

When $H = 1/2$, W^H is a standard Brownian motion, and when $H \neq 1/2$, W^H is neither a martingale nor a Markov process. The objective is to

$$\max_{0 \leqslant \tau \leqslant 1} \mathbb{E}[W_\tau^H]$$

over all W^H-stopping times $\tau \in [0, 1]$. This problem is non-Markovian and therefore falls outside the primary scope of this paper. However, we can approximate the optimal stopping value by discretizing the process. When combined with state space augmentation, this discretization transforms the problem into a Markovian one, thereby enabling the application of the martingale-based approach developed in this work.

To do this, denote $t_n = n/100$, for $n = 0, 1, \ldots, 100$. For details on simulating trajectories of fractional Brownian motion, see Becker et al. (2019). Define the following 101-dimensional state process:

$$X_0 = \left(W^H_{t_0}, W^H_{t_0}, W^H_{t_0}, \ldots, W^H_{t_0}\right), \qquad X_1 = \left(W^H_{t_1}, W^H_{t_0}, W^H_{t_0}, \ldots, W^H_{t_0}\right),$$
$$X_2 = \left(W^H_{t_2}, W^H_{t_1}, W^H_{t_0}, \ldots, W^H_{t_0}\right), \quad \ldots, \quad X_{100} = \left(W^H_{t_{100}}, W^H_{t_{99}}, W^H_{t_{98}}, \ldots, W^H_{t_0}\right),$$

and consider the following optimal stopping problem:

$$\max_{\tau \in \mathcal{T}} \mathbb{E}[g(X_\tau)],$$

where $g(x) = x^{(1)}$ for a multi-dimensional vector x with first component $x^{(1)}$, and $\mathcal{T}$ denotes the set of all X-stopping times. The multi-dimensional state process X_0, X_1, $\cdots$, X_{100} is now Markovian, and the martingale-based approach applies.

In our simulation, we set the penalty factor K to be 100 (corresponding to a time mesh grid $\Delta t = 1/100$ as specified in Section 4.5.1), the temperature parameter λ to be 0.1, the training batch size to be 2^{10}, the testing batch size to be 2^{15}, and the initial learning rate to 0.01 which is halved every 200 training steps. We train for 3000 steps using Algorithm 1.

Table 4 presents the learning results for different Hurst parameter values. The learned value is evaluated every 10 training steps, resulting in 300 observations for each case. The final observation is reported in the second column of Table 4. Note that in this example, if the agent does not stop until the terminal time, the stopping value is 0 in theory. Thus, our RL algorithm can achieve a high stopping value. Our learning results are comparable to benchmarks in the literature (Becker et al., 2019; Dong, 2024), demonstrating high accuracy in learning the stopping decisions in a non-Markovian (or equivalently a high-dimensional) setting.

6 Conclusions

In this paper, we study optimal stopping for diffusion processes in the exploratory RL framework of Wang et al. (2020). Different from Dong (2024) and Dai and Dong (2024) which directly randomize stopping times for exploration through mixed strategies characterized by Cox processes, we turn the problem into stochastic control by approximating the variational inequality with the penalized PDE. This enables us to apply all the available results and methods developed recently in RL for controlled diffusion processes (Jia and Zhou, 2022a,b, 2023). Our simulation study shows that the resulting RL algorithms achieve comparable, and sometimes better, performance in terms of accuracy and speed than those in Dong (2024) and Dai and Dong (2024). More importantly, we show experimentally that the proposed algorithms also work well for high-dimensional

H	Last learned value by our method	In Dong (2024)	In Becker et al. (2019)	Runtime in sec.
0.01	1.496	1.513	1.519	3013
0.05	1.270	1.283	1.293	3008
0.10	1.027	1.043	1.049	3039
0.15	0.821	0.825	0.839	3476
0.20	0.641	0.651	0.658	3539
0.25	0.490	0.494	0.503	3519
0.30	0.360	0.364	0.370	3525
0.35	0.246	0.243	0.255	3532
0.40	0.148	0.144	0.156	3519
0.45	0.059	0.061	0.071	3519
0.50	0.002	0.000	0.002	3526
0.55	0.056	0.052	0.061	3528
0.60	0.110	0.108	0.117	3516
0.65	0.158	0.155	0.164	3528
0.70	0.201	0.192	0.207	3352
0.75	0.240	0.233	0.244	3080
0.80	0.274	0.261	0.277	3009
0.85	0.306	0.299	0.308	2454
0.90	0.337	0.331	0.337	2536
0.95	0.366	0.354	0.366	2370

Table 4: Comparison of stopping values and runtime for different values of Hurst parameter H in the example of optimally stopping a fractional Brownian motion. The horizon length $T = 1$.

problems.

There is a notable limitation of the current approach especially to high-dimensional problems. In our implementation, we use neural networks to approximate value functions, which enables us to overcome the curse of dimensionality. However, a challenge may arise if the value functions—such as discontinuous ones—cannot be well approximated by neural networks. Overcoming the difficulty is an interesting yet important question.

We take option pricing as an application of the general theory in our experiments. There is a caveat in this: the expectation in the pricing formula is with respect to the risk-neutral probability under which data are *not observable*. While we can use simulation to test the learning efficiency of an RL algorithm, for real applications we need to take a formulation for option pricing that is under the physical measure due to the data-driven nature of RL. One example of such a formulation is the mean–variance hedging.

There are many other interesting problems to investigate along the lines of this paper, including a more substantial convergence and regret analysis, choice of the temperature parameter, and more complex underlying processes to stop.

A Proofs of statements

A.1 Proof of Lemma 2.1

Suppose there is a positive sequence $\{K_n\}_{n\geqslant 1}$ that increases to infinity, and $v^{(n)}$ solves (2.5) with $K = K_n$. Denote $a_n = v_t^{(n)} + \mathscr{L} v^{(n)}$, $b_n = g - v^{(n)}$, $a_\infty = v_t + \mathscr{L} v$, and $b_\infty = g - v$, where v is the limit of $v^{(n)}$. From (2.5) we have $a_n = -K_n (b_n)^+ \leqslant 0$. Because (a_n, b_n) converges to (a_∞, b_∞) as $n \to \infty$, we get $a_\infty \leqslant 0$. On the other hand, on any compact region, $\{a_n\}_{n\geqslant 1}$ is locally bounded; so $(b_n)^+ = -\frac{a_n}{K_n} \to 0$, as $n \to \infty$, yielding $b_\infty \leqslant 0$.

If $b_\infty < 0$, then there exists a large number N such that $b_n < 0$ for all $n > N$, leading to $a_n = 0$ for all $n > N$. Thus, $a_\infty = 0$ when $b_\infty < 0$.

Hence, we have $a_\infty \leqslant 0$, $b_\infty \leqslant 0$ and $a_\infty b_\infty = 0$, which implies that v solves (2.3).

A.2 Proof of Theorem 2.2

For notational simplicity, without loss of generality we only consider the case when X and W are both one-dimensional.

Write $\overline{R}_t = e^{-\int_s^t f(r, X_r, u_r)\,\mathrm{d}r}$, $t \geqslant s$. Then $\mathrm{d}\overline{R}_t = -\overline{R}_t f(t, X_t, u_t)\,\mathrm{d}t$, $\overline{R}_s = 1$. Itô's

lemma gives

$$
\begin{aligned}
&\mathrm{d}\left(\overline{R}_t v(t,X_t)\right)\\
&= \overline{R}_t\,\mathrm{d}v(t,X_t) + v(t,X_t)\,\mathrm{d}\overline{R}_t\\
&= \overline{R}_t\Big[\left(v_t(t,X_t) + \mathscr{L}_u v(t,X_t)\right)\mathrm{d}t + v_x(t,X_t)\sigma(t,X_t,u_t)\,\mathrm{d}W_t\Big] - v(t,X_t)\overline{R}_t f(t,X_t,u_t)\,\mathrm{d}t\\
&= \overline{R}_t\Big[\left(v_t(t,X_t) + \mathscr{L}_u v(t,X_t) - v(t,X_t)f(t,X_t,u_t)\right)\mathrm{d}t + v_x(t,X_t)\sigma(t,X_t,u_t)\,\mathrm{d}W_t\Big]\\
&\leqslant -\overline{R}_t G(t,X_t,u_t)\,\mathrm{d}t + \overline{R}_t v_x(t,X_t)\sigma(t,X_t,u_t)\,\mathrm{d}W_t,
\end{aligned}
$$

where the last inequality is due to (2.10) and $\overline{R}_t > 0$. Integrating both sides from s to T and then taking conditional expectation yield

$$
\mathbb{E}\Big[\overline{R}_T v(T,X_T) - \overline{R}_s v(s,X_s)\;\Big|\;X_s = x\Big] \leqslant -\,\mathbb{E}\left[\int_s^T \overline{R}_t G(t,X_t,u_t)\,\mathrm{d}t\;\middle|\;X_s = x\right].
$$

As $\overline{R}_s = 1$, the above leads to

$$
\begin{aligned}
v(s,x) &\geqslant \mathbb{E}\left[\int_s^T \overline{R}_t G(t,X_t,u_t)\,\mathrm{d}t + \overline{R}_T v(T,X_T)\;\middle|\;X_s = x\right]\\
&= \mathbb{E}\left[\int_s^T e^{-\int_s^t f(r,X_r,u_r)\,\mathrm{d}r} G(t,X_t,u_t)\,\mathrm{d}t + e^{-\int_s^T f(r,X_r,u_r)\,\mathrm{d}r} H(X_T)\;\middle|\;X_s = x\right].
\end{aligned}
$$

This means that v is an upper bound of the optimal value function of problem (2.8). All the inequalities above become equalities if we take $u \in U$ that solves the optimization problem in (2.10). Therefore, we conclude v is the optimal value function of (2.8).

A.3 Proof of Theorem 3.1

By the Feynman–Kac formula, for any (t,x),

$$
\partial_t J^\pi(t,x) + \mathscr{L} J^\pi(t,x) + H\left(t,x,\pi,J^\pi\right) = 0,
$$

where

$$
H\left(t,x,\pi,J^\pi\right) := K\pi(t,x)\left(g(t,x) - J^\pi(t,x)\right) - \lambda\mathscr{H}(\pi(t,x)).
$$

Similarly, for any (t,x),

$$
\partial_t J^{\tilde{\pi}}(t,x) + \mathscr{L} J^{\tilde{\pi}}(t,x) + H\left(t,x,\tilde{\pi},J^{\tilde{\pi}}\right) = 0,
$$

where $\tilde{\pi}$ is defined in (3.1). It turns out that

$$
\tilde{\pi}(t,x) = \operatorname*{argmax}_{\pi\in[0,1]} H\left(t,x,\pi,J^\pi\right).
$$

Denote $\Delta(t,x) := J^{\tilde{\pi}}(t,x) - J^\pi(t,x)$. Then

$$
\partial_t \Delta(t,x) + \mathscr{L}\Delta(t,x) + H\left(t,x,\tilde{\pi},J^{\tilde{\pi}}\right) - H\left(t,x,\pi,J^\pi\right) = 0.
$$

Because

$$\begin{aligned}
&H\left(t,x,\tilde{\pi},J^{\tilde{\pi}}\right)-H\left(t,x,\pi,J^{\pi}\right)\\
=&H\left(t,x,\tilde{\pi},J^{\tilde{\pi}}\right)-H\left(t,x,\tilde{\pi},J^{\pi}\right)+H\left(t,x,\tilde{\pi},J^{\pi}\right)-H\left(t,x,\pi,J^{\pi}\right)\\
=&-K\tilde{\pi}(t,x)\Delta(t,x)+H\left(t,x,\tilde{\pi},J^{\pi}\right)-H\left(t,x,\pi,J^{\pi}\right),
\end{aligned}$$

we have

$$\partial_t\Delta(t,x)+\mathscr{L}\Delta(t,x)-K\tilde{\pi}(t,x)\Delta(t,x)=H\left(t,x,\pi,J^{\pi}\right)-H\left(t,x,\tilde{\pi},J^{\pi}\right)\leqslant 0,$$

where the last inequality is from the definition of $\tilde{\pi}$. Besides, $\Delta(T,x)=0$ at the terminal time T. Therefore, 0 is a sub-solution of $-\partial_t u(t,x)-\mathscr{L}u(t,x)+K\tilde{\pi}(t,x)u(t,x)+H\left(t,x,\pi,J^{\pi}\right)-H\left(t,x,\tilde{\pi},J^{\pi}\right)=0$. By the maximum principle, we have $\Delta(t,x)\geqslant 0$, resulting in $J^{\tilde{\pi}}(t,x)\geqslant J^{\pi}(t,x)$.

A.4 Proof of Theorem 3.2

Recall that $v^{\mathbf{\Pi}}$ is the optimal value function of the exploratory control problem (2.20) with the optimal strategy (2.24). By the same procedure as in the proof of Theorem 3.1, we can show that $v^{\mathbf{\Pi}}\geqslant\cdots\geqslant J^{\pi^{n+1}}\geqslant J^{\pi^n}\geqslant\cdots\geqslant J^{\pi^0}$. Next, we show the uniform convergence of the sequence $\left\{J^{\pi^n}\right\}_{n=1}^{\infty}$. To proceed, write $\|g\|_{\infty}=C$ for some positive constant C due to the assumption that g is uniformly bounded.

For any $n\geqslant 1$, denote

$$\begin{aligned}
f^n(t)&:=\sup_x\left\{v^{\mathbf{\Pi}}(t,x)-J^{\pi^n}(t,x)\right\}\geqslant 0,\\
\phi^n(t)&:=2K\int_t^T f^n(s)ds,\\
W(t,x)&:=v^{\mathbf{\Pi}}(t,x)-J^{\pi^{n+1}}(t,x)-\phi^n(t).
\end{aligned}$$

Using $\partial_t J^{\pi^{n+1}}+\mathscr{L}J^{\pi^{n+1}}+H\left(t,x,\pi^{n+1},J^{\pi^{n+1}}\right)=0$ and $\partial_t v^{\mathbf{\Pi}}+\mathscr{L}v^{\mathbf{\Pi}}+H\left(t,x,\pi^*,v^{\mathbf{\Pi}}\right)=0$, we have

$$\begin{aligned}
\partial_t W+\mathscr{L}W&=\partial_t v^{\mathbf{\Pi}}+\mathscr{L}v^{\mathbf{\Pi}}-\left(\partial_t J^{\pi^{n+1}}+\mathscr{L}J^{\pi^{n+1}}\right)-\partial_t\phi^n\\
&=H\left(t,x,\pi^{n+1},J^{\pi^{n+1}}\right)-H\left(t,x,\pi^*,v^{\mathbf{\Pi}}\right)-\partial_t\phi^n\\
&=H\left(t,x,\pi^{n+1},J^{\pi^{n+1}}\right)-H\left(t,x,\pi^{n+1},v^{\mathbf{\Pi}}\right)\\
&\quad+H\left(t,x,\pi^{n+1},v^{\mathbf{\Pi}}\right)-H\left(t,x,\pi^*,v^{\mathbf{\Pi}}\right)-\partial_t\phi^n.
\end{aligned}$$

Noticing that

$$\begin{aligned}
H\left(t,x,\pi,J^{\pi}\right) &= K\pi(t,x)\left(g(t,x)-J^{\pi}(t,x)\right)-\lambda\mathscr{H}(\pi(t,x)),\\
\mathscr{H}(\pi^{n+1}(t,x)) &= \pi^{n+1}\log\pi^{n+1}+\left(1-\pi^{n+1}\right)\log\left(1-\pi^{n+1}\right)\\
&= \lambda^{-1}K\left(J^{\pi^n}-g\right)\left(1-\pi^{n+1}\right)-\log\left(1+\exp\left(\lambda^{-1}K\left(J^{\pi^n}-g\right)\right)\right),
\end{aligned}$$

and

$$\begin{aligned}
\mathscr{H}(\pi^{*}(t,x)) &= \pi^{*}\log\pi^{*}+\left(1-\pi^{*}\right)\log\left(1-\pi^{*}\right)\\
&= \lambda^{-1}K\left(v^{\mathbf{\Pi}}-g\right)\left(1-\pi^{*}\right)-\log\left(1+\exp\left(\lambda^{-1}K\left(v^{\mathbf{\Pi}}-g\right)\right)\right),
\end{aligned}$$

we obtain

$$\begin{aligned}
H\left(t,x,\pi^{n+1},v^{\mathbf{\Pi}}\right) &= K\pi^{n+1}\left(g-v^{\mathbf{\Pi}}\right)-K\left(J^{\pi^n}-g\right)\left(1-\pi^{n+1}\right)\\
&\quad+\lambda\log\left(1+\exp\left(\lambda^{-1}K\left(J^{\pi^n}-g\right)\right)\right)
\end{aligned}$$

and

$$\begin{aligned}
H\left(t,x,\pi^{*},v^{\mathbf{\Pi}}\right) &= K\pi^{*}\left(g-v^{\mathbf{\Pi}}\right)-K\left(v^{\mathbf{\Pi}}-g\right)\left(1-\pi^{*}\right)\\
&\quad+\lambda\log\left(1+\exp\left(\lambda^{-1}K\left(v^{\mathbf{\Pi}}-g\right)\right)\right)\\
&= K\left(g-v^{\mathbf{\Pi}}\right)+\lambda\log\left(1+\exp\left(\lambda^{-1}K\left(v^{\mathbf{\Pi}}-g\right)\right)\right).
\end{aligned}$$

Straightforward calculations show that

$$H\left(t,x,\pi^{n+1},J^{\pi^{n+1}}\right)-H\left(t,x,\pi^{n+1},v^{\mathbf{\Pi}}\right)=K\pi^{n+1}\left(v^{\mathbf{\Pi}}-J^{\pi^{n+1}}\right)\geqslant 0$$

and

$$\begin{aligned}
&H\left(t,x,\pi^{n+1},v^{\mathbf{\Pi}}\right)-H\left(t,x,\pi^{*},v^{\mathbf{\Pi}}\right)\\
&=K\left(v^{\mathbf{\Pi}}-J^{\pi^n}\right)\frac{1}{1+\exp\left(-\lambda^{-1}K\left(J^{\pi^n}-g\right)\right)}-\lambda\log\left(\frac{1+\exp\left(\lambda^{-1}K\left(v^{\mathbf{\Pi}}-g\right)\right)}{1+\exp\left(\lambda^{-1}K\left(J^{\pi^n}-g\right)\right)}\right).
\end{aligned}$$

It follows from the general inequality $|\log(1+\exp(x))-\log(1+\exp(y))|\leqslant|x-y|$ that

$$\left|H\left(t,x,\pi^{n+1},v^{\mathbf{\Pi}}\right)-H\left(t,x,\pi^{*},v^{\mathbf{\Pi}}\right)\right|\leqslant 2K\left(v^{\mathbf{\Pi}}-J^{\pi^n}\right)\leqslant 2Kf^{n}(t).$$

Thus,

$$\begin{aligned}
&H\left(t,x,\pi^{n+1},v^{\mathbf{\Pi}}\right)-H\left(t,x,\pi^{*},v^{\mathbf{\Pi}}\right)-\partial_t\phi^{n}\\
&=H\left(t,x,\pi^{n+1},v^{\mathbf{\Pi}}\right)-H\left(t,x,\pi^{*},v^{\mathbf{\Pi}}\right)+2Kf^{n}(t)\geqslant 0.
\end{aligned}$$

Consequently,

$$\partial_t W + \mathscr{L} W \geqslant 0,$$

which together with $W(T,x) = 0$ implies $W \leqslant 0$ by the maximum principle. Thus, we conclude

$$v^{\mathbf{\Pi}}(t,x) - J^{\pi^{n+1}}(t,x) \leqslant \phi^n(t) = 2K \int_t^T f^n(s) ds.$$

The definition of f^n yields

$$f^{n+1}(t) \leqslant 2K \int_t^T f^n(s) ds.$$

By induction, we have

$$f^{n+1}(t) \leqslant (2K)^n \frac{(T-t)^n}{n!} \sup_t f^1(t) \leqslant \frac{(2KT)^n}{n!} \sup_t f^1(t).$$

Recall

$$J^{\pi}(s,x) = \mathbb{E}\left[\int_s^T e^{-K\int_s^t \pi_v \,\mathrm{d}v} \left[Kg(t, X_t)\pi_t - \lambda \mathscr{H}(\pi_t) \right] \mathrm{d}t + e^{-K\int_s^T \pi_v \,\mathrm{d}v} g(T, X_T) \,\middle|\, X_s = x \right],$$

$\|g\|_\infty = C$, $e^{-K\int_s^t \pi_v \,\mathrm{d}v} \leqslant 1$ for any $0 \leqslant t \leqslant T$, and $0 \leqslant \pi_t \leqslant 1$ and

$$|\mathscr{H}(\pi_t)| \leqslant \sup_{x \in (0,1)} |x \log x + (1-x)\log(1-x)| = \log 2.$$

We thus have

$$|J^{\pi}(s,x)| \leqslant \mathbb{E}\left[\int_s^T \left[KC + \lambda \log 2 \right] \mathrm{d}t + C \,\middle|\, X_s = x \right] \leqslant (KC + \lambda \log 2)T + C$$

for any admissible π. It hence follows

$$\sup_t f^1(t) = \sup_{t,x} \left\{ v^{\mathbf{\Pi}}(t,x) - J^{\pi^1}(t,x) \right\} \leqslant 2((KC + \lambda \log 2)T + C).$$

Consequently,

$$\left\| v^{\mathbf{\Pi}} - J^{\pi^{n+1}} \right\|_\infty \leqslant \frac{(2KT)^n}{n!} 2((KC + \lambda \log 2)T + C),$$

which converges to 0 as n increases to infinity when K and λ are fixed. This concludes the proof.

Finally, let us complement the proof by numerically examining the theoretical convergence rate. As shown in Figure 8, the NN approximation for the stopping value of $(t_0, x_0) = (0, 40)$ converges to its theoretical value 5.333, which is calculated using the finite difference method, and the relative error decreases exponentially. In implementation, we use stochastic gradient descent to update the NN parameters at each training

step, which prevents us from performing the policy evaluation in (3.8) with a complete accuracy. Consequently, in the experiment we do not achieve the $n!$ rate of decay in relative error as predicted by Theorem 3.2. Nevertheless, exponential decay is sufficient to ensure rapid convergence.

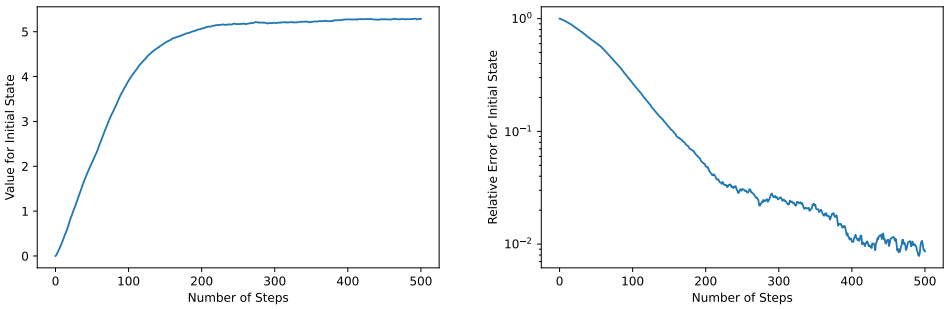


Figure 8: Numerical diagnostics for the convergence of the RL algorithms. We use the offline ML Algorithm 1 to learn the stopping value of $(t_0, x_0) = (0, 40)$, approximated by an NN, for the American put example in Section 4 with $\lambda = 0.01$ and $K = 50$. The left panel displays the learning curve of the value and the right panel shows that of the relative error, both with respect to training steps. The y-axis of the right panel is plotted on a logarithmic scale.

B Learning early exercise premium

In this section, we discuss how to learn the early exercise premium of an American put option.

Recall we have an explicit pricing formula for a European put option:

$$\begin{cases} V_E(t,x) = \Gamma e^{-\rho(T-t)} N\left(-d_-(t,x)\right) - xN\left(-d_+(t,x)\right), \ t \in [0,T), \\ V_E(T,x) = g(x), \end{cases} \tag{B.1}$$

where $d_\pm(t,x) = \frac{1}{\sigma\sqrt{T-t}}\left[\log\left(\frac{x}{\Gamma}\right) + \left(\rho \pm \frac{1}{2}\sigma^2\right)(T-t)\right]$, and $N(\cdot)$ is the cumulative standard normal distribution function $N(y) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{y} e^{-\frac{z^2}{2}}\,\mathrm{d}z = \frac{1}{\sqrt{2\pi}}\int_{-y}^{\infty} e^{-\frac{z^2}{2}}\,\mathrm{d}z$.

The early exercise premium of the American put option is

$$\tilde{v}(t,x) := v(t,x) - V_E(t,x), \tag{B.2}$$

where $v(t,x)$ is the option value at the time–state pair (t,x). The associated payoff function is

$$\tilde{g}(t,x) := g(x) - V_E(t,x). \tag{B.3}$$

By (2.3) and the fact that V_E satisfies $\frac{\partial}{\partial t}V_E + \mathscr{L}V_E = 0$ with terminal condition $V_E(T,x) =$

$g(x)$, we have

$$\begin{cases} \max\{\tilde{v}_t + \mathscr{L}\tilde{v},\ \tilde{g} - \tilde{v}\} = 0,\ (t,x) \in [0,T) \times \mathbb{R}, \\ \tilde{v}(T,x) = 0. \end{cases} \tag{B.4}$$

Thus, the terminal condition is smooth now, and the related PDE by penalty approximation is

$$\begin{cases} \tilde{v}_t + \max_{u\in\{0,1\}} \{\mathscr{L}\tilde{v} - Ku\tilde{v} + K\tilde{g}(t,x)u\} = 0,\ (t,x) \in [0,T) \times \mathbb{R}, \\ \tilde{v}(T,x) = 0. \end{cases}$$

Consequently, by Theorem 2.2 the corresponding control problem is

$$\max_{u\in\{0,1\}} \mathbb{E}\left[\int_s^T Ke^{-K\int_s^t u_r \,\mathrm{d}r}\tilde{g}(t,X_t)u_t \,\mathrm{d}t \,\middle|\, X_s = x\right], \tag{B.5}$$

and the entropy-regularized exploratory problem is

$$\max_{\pi\in[0,1]} \mathbb{E}\left[\int_s^T e^{-K\int_s^t \pi_v \,\mathrm{d}v}\left[K\tilde{g}(t,X_t)\pi_t - \lambda\mathscr{H}(\pi_t)\right]\mathrm{d}t \,\middle|\, X_s = x\right]. \tag{B.6}$$

While we could use the algorithms in Section 3 to learn $\tilde{v}$, $\tilde{g}$ is not fully known to us since σ (which appears in V_E) is an unknown model parameter.

To deal with the unknown payoff function $\tilde{g}$, recall

$$\tilde{g}(t,x) = g(x) - V_E(t,x) = (\Gamma - x)^+ - V_E(t,x). \tag{B.7}$$

Hence, in the formula of V_E, namely (B.1), we replace σ by ϕ, which is a parameter to learn with a properly chosen initial value. Denote by V_E^ϕ and $\tilde{g}^\phi$ the parameterized European option value function and payoff function, respectively. Note that

$$V_E(t,x) = \mathbb{E}\left[e^{-\rho(T-t)}g(X_T) \,\middle|\, X_t = x\right] = \mathbb{E}\left[e^{-\rho(T-t)}\left(\Gamma - X_T\right)^+ \,\middle|\, X_t = x\right]. \tag{B.8}$$

This is a policy evaluation problem. Thanks to the martingale approach developed in Jia and Zhou (2022a), we can learn/update ϕ by minimizing the martingale loss w.r.t. ϕ:

$$\begin{aligned} \mathrm{ML}(\phi) &\approx \frac{1}{2}\mathbb{E}\left[\sum_{l=0}^{L-1}\left(e^{-\rho T}V_E(T,X_T) - e^{-\rho t}V_E^\phi(t_l,X_l)\right)^2\Delta t\right] \\ &= \frac{1}{2}\mathbb{E}\left[\sum_{l=0}^{L-1}\left(e^{-\rho T}\left(\Gamma - X_T\right)^+ - e^{-\rho t}V_E^\phi(t_l,X_l)\right)^2\Delta t\right]. \end{aligned} \tag{B.9}$$

In each episode, we simulate a batch of stock price trajectories and then use the stochastic gradient descent method to minimize the above loss function to update ϕ. Figure 9 shows the efficacy of this policy evaluation algorithm, with an initial guess of the unknown σ to be 0.8, twice the true value.

We emphasize that in this particular example of European option, due to the avail-

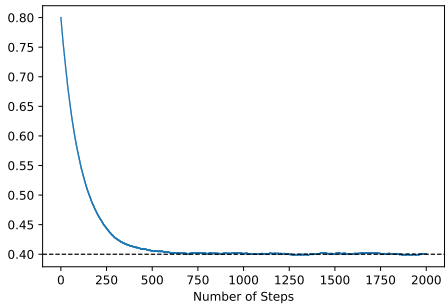


Figure 9: Learning curve of ϕ, which is used to parameterize the payoff function (B.7) by minimizing the martingale loss (B.9). The initialization of ϕ is 0.8, and the true value of σ is 0.4. The value of ϕ after 2000 steps is 0.4006.

ability of the explicit formula of the option price, the critic parameter ϕ *happens* to be a model parameter σ. In general, for more complicated problems with little structure, the actor/critic parameters may be very different from the model primitives and may themselves have no physical/practical meanings at all. Estimating model parameters is *not* a goal of RL in our framework.

C Additional results for sensitivity tests

In Figure 10, we show a sensitivity test when the penalty factor approaches its maximum possible value, i.e., $K = 50$, under our time discretization setting. In this case, the two ways of calculating the option price have almost the same results, which is consistent with Lemma 2.1. The trade-off between learning accuracy and speed observed previously still holds.

D RL for Dynkin games

Theorem 1 in Dai and Dong (2024) reveals that Merton's problem with proportional transaction costs is equivalent to a Dynkin game that is essentially an optimal stopping problem. Here, we focus on the Black–Scholes market where the stock price follows

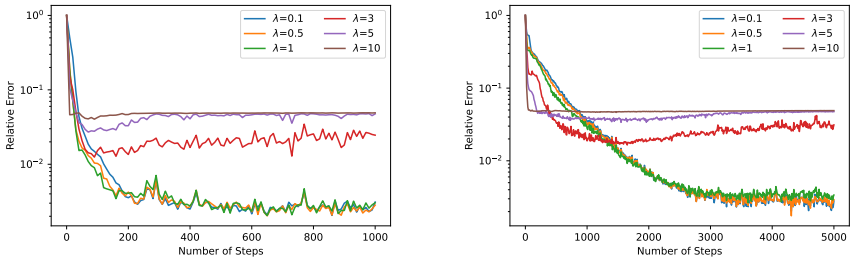
$$\mathrm{d}S_t = \alpha S_t \,\mathrm{d}t + \sigma S_t \,\mathrm{d}W_t,$$

where α and σ are constants but unknown to the agent. The value function of the Dynkin game satisfies the following variational inequality problem

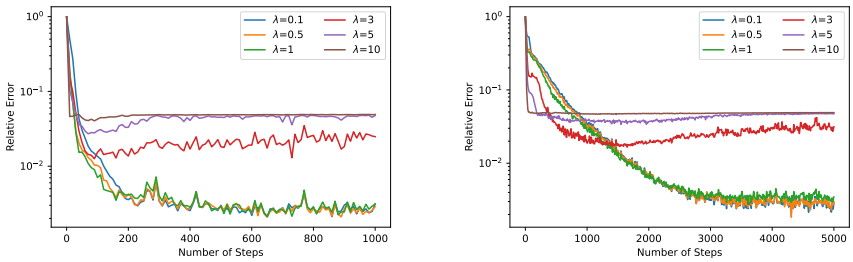
$$\max\left\{\min\left\{-w_t - \mathcal{L}w, w - L(x)\right\}, w - U(x)\right\} = 0 \tag{D.1}$$

with terminal condition

$$w(T,x) = \frac{x}{x+1-\theta}, \tag{D.2}$$



(a) $\mathcal{P}_{\text{stopping}}$ by ML (b) $\mathcal{P}_{\text{stopping}}$ by TD(0)



(c) $\mathcal{P}_{\text{control}}$ by ML (d) $\mathcal{P}_{\text{control}}$ by TD(0)

Figure 10: Learning curves of option prices $\mathcal{P}_{\text{stopping}}$ defined in (4.5) and $\mathcal{P}_{\text{control}}$ defined in (4.7) by offline ML and online TD(0) algorithms. In each panel, the penalty factor K is fixed to 50 and the temperature parameter λ takes value from the set $\{0.1, 0.5, 1, 3, 5, 10\}$ while the other parameter values are set as in Table 1. We train the ML algorithm for 1000 steps and TD(0) algorithm for 5000 steps.

where $\theta \in (0, 1)$ is the proportional transaction cost rate, x is the bond–stock position ratio, $\mathcal{L}$ is the generator of this ratio process, and $L(x)$ and $U(x)$ are respectively the lower and upper obstacles of $w(t, x)$.

It is proved in Zhang et al. (2022) that (D.1) can be approximated by the following penalized PDE:

$$-w_t - \mathcal{L}w - K\left(w - L(x)\right)^- + K\left(w - U(x)\right)^+ = 0, \tag{D.3}$$

where $K > 0$ is a large penalty factor. This PDE can be equivalently written as

$$w_t + \max_{a\in\{0,1\}} \min_{b\in\{0,1\}} \left\{\mathcal{L}w - wf(t, x, a, b) + G(t, x, a, b)\right\} = 0, \tag{D.4}$$

where

$$f(t, x, a, b) = K(a + b), \quad G(t, x, a, b) = K\left(L(x)a + U(x)b\right).$$

Similar to the setup in Section 2.2, we consider a constrained differential game:

$$w(s, x) = \max_{a\in\mathscr{U}} \min_{b\in\mathscr{U}} J(s, x; a, b), \tag{D.5}$$

where

$$J(s,x;a,b)=\mathbb{E}\Bigg[\int_s^T e^{-\int_s^t f(r,X_r,a_r,b_r)\,\mathrm{d}r}G(t,X_t,a_t,b_t)\,\mathrm{d}t + e^{-\int_s^T f(r,X_r,a_r,b_r)\,\mathrm{d}r}H(X_T)\,\Big|\,X_s=x\Bigg],$$

and the controlled n-dimensional process X follows

$$\mathrm{d}X_t=b(t,X_t,a_t,b_t)\,\mathrm{d}t+\sigma(t,X_t,a_t,b_t)\,\mathrm{d}W_t. \tag{D.6}$$

We have the following result.

Theorem D.1 (Verification theorem). *Suppose w is a classical solution of*

$$\begin{cases} w_t+\max_{a\in U}\min_{b\in U}\left\{\mathscr{L}_{a,b}w-wf+G\right\}=0, \quad (t,x)\in[0,T)\times\mathbb{R}^n,\\ w(T,x)=H(x), \end{cases} \tag{D.7}$$

where

$$\mathscr{L}_{a,b}w=\frac{1}{2}\sum_{i,j,k}\sigma_{i,k}(t,x,a,b)\sigma_{j,k}(t,x,a,b)\frac{\partial^2 w}{\partial x_i\partial x_j}+\sum_i b_i(t,x,a,b)\frac{\partial w}{\partial x_i}.$$

Then it is the value function of problem (D.5).

Proof. Let $\overline{R}_t=e^{-\int_s^t f(r,X_r,a_r,b_r)\,\mathrm{d}r}$, $t\geqslant s$. Then $\mathrm{d}\overline{R}_t=-\overline{R}_t f(t,X_t,a_t,b_t)\,\mathrm{d}t$ and $\overline{R}_s=1$.

For each fixed $a\in U$, let $b_t^*=b_t^*(x,a)$ denote the minimizer of $\mathscr{L}_{a,b}w-wf+G$. By (D.7),

$$w_t+\max_{a\in U}\left\{\mathscr{L}_{a,b^*}w-wf(t,x,a,b^*)+G(t,x,a,b^*)\right\}=0.$$

Then, similar to the proof in Appendix A.2, we get for any admissible process $a\in\mathscr{U}$,

$$\begin{aligned} w(s,x)&\geqslant\mathbb{E}\left[\int_s^T\overline{R}_t^{a,b^*}G(t,X_t,a_t,b_t^*)\,\mathrm{d}t+\overline{R}_T^{a,b^*}w(T,X_T)\,\Big|\,X_s=x\right]\\ &=\mathbb{E}\left[\int_s^T e^{-\int_s^t f(r,X_r,a_r,b_r^*)\,\mathrm{d}r}G(t,X_t,a_t,b_t^*)\,\mathrm{d}t+e^{-\int_s^T f(r,X_r,a_r,b_r^*)\,\mathrm{d}r}H(X_T)\,\Big|\,X_s=x\right]\\ &=J(s,x;a,b^*)\\ &\geqslant\min_{b\in\mathscr{U}}J(s,x;a,b). \end{aligned}$$

Since $a\in\mathscr{U}$ is arbitrarily chosen, the above implies

$$w(s,x)\geqslant\max_{a\in\mathscr{U}}\min_{b\in\mathscr{U}}J(s,x;a,b).$$

Hence, w is an upper bound of the value function of problem (D.5).

On the other hand, if we choose a pair $(a^*,b_t^*(x,a^*))$ to solve

$$\max_{a\in U}\min_{b\in U}\left\{\mathscr{L}_{a,b}w-wf+G\right\},$$

all the inequalities in the above become equalities, yielding that w is the value function of (D.5). □

Following the same procedure as in Section 2.2, we find that the solution to (D.4) satisfying the terminal condition is the value function of the following stochastic differential game (Fleming and Soner, 2006):

$$w(s,x) = \max_{a_t\in\{0,1\}} \min_{b_t\in\{0,1\}} J(s,x;a,b), \tag{D.8}$$

where

$$J(s,x;a,b) = \mathbb{E}\Big[\int_s^T Ke^{-K\int_s^t (a_r+b_r)\,\mathrm{d}r}\left(L(X_t)a_t + U(X_t)b_t\right)\mathrm{d}t + e^{-K\int_s^T (a_r+b_r)\,\mathrm{d}r}\frac{X_T}{X_T+1-\theta}\,\Big|\, X_s = x\Big].$$

Next, we randomize $\{a_t\}_{t\in[0,T]}$ and $\{b_t\}_{t\in[0,T]}$ respectively by $\{\nu_t\}_{t\in[0,T]}$ and $\{\mu_t\}_{t\in[0,T]}$, where

$$\nu_t = \mathbb{P}(a_t = 1) = 1 - \mathbb{P}(a_t = 0), \quad \mu_t = \mathbb{P}(b_t = 1) = 1 - \mathbb{P}(b_t = 0).$$

It turns out that the exploratory value function of the Dynkin game is

$$w^\lambda(s,x) = \max_{\nu_t\in[0,1]} \min_{\mu_t\in[0,1]} J^\lambda(s,x;\nu,\mu), \tag{D.9}$$

where

$$J^\lambda(s,x;\nu,\mu) = \mathbb{E}\Big[\int_s^T Ke^{-K\int_s^t (\nu_r+\mu_r)\,\mathrm{d}r}\left[L(X_t)\nu_t + U(X_t)\mu_t - \lambda\left(\mathscr{H}(\nu_t) - \mathscr{H}(\mu_t)\right)\right]\mathrm{d}t + e^{-K\int_s^T (\nu_r+\mu_r)\,\mathrm{d}r}\frac{X_T}{X_T+1-\theta}\,\Big|\, X_s = x\Big].$$

The HJB equation of problem (D.9) is thus

$$w_t^\lambda + \mathcal{L}w^\lambda + \max_{\nu\in[0,1]} \min_{\mu\in[0,1]} \left\{-w^\lambda K(\nu+\mu) + K\left(L(x)\nu + U(x)\mu\right) - \lambda\left(\mathscr{H}(\nu) - \mathscr{H}(\mu)\right)\right\} = 0, \tag{D.10}$$

and the optimal solutions are

$$\begin{aligned}
\nu^* &= \frac{\exp\left(\frac{K}{\lambda}(L(x) - w^\lambda)\right)}{\exp\left(\frac{K}{\lambda}(L(x) - w^\lambda)\right) + 1} = \frac{1}{1 + \exp\left(-\frac{K}{\lambda}(L(x) - w^\lambda)\right)}, \\
\mu^* &= \frac{\exp\left(\frac{K}{\lambda}(w^\lambda - U(x))\right)}{\exp\left(\frac{K}{\lambda}(w^\lambda - U(x))\right) + 1} = \frac{1}{1 + \exp\left(-\frac{K}{\lambda}(w^\lambda - U(x))\right)}.
\end{aligned} \tag{D.11}$$

Consequently, if we choose the TD error criterion for policy evaluation, the RL algorithm is essentially the same as Algorithm 1 in Dai and Dong (2024), except that ν, μ, ν^*, μ^* in their algorithm should be replaced by $K\nu, K\mu$, ν^*, μ^* in (D.11) in this section, and their entropy term should be replaced by the differential entropy in this paper.

E Finite difference method

In this section, we present (classical, model-based) algorithms for pricing a finite-horizon American put option and obtaining its stopping boundary. We use the penalty method in Forsyth and Vetzal (2002). The payoff function is $g(x) = (\Gamma - x)^+$ with x being the stock price and Γ the strike price. We consider the PDE (2.5) and let $u(t, y) := v(t, x)$ with $x = e^y$. In the finite difference scheme below, the time horizon $[0, T]$ is divided to N_t equally spaced intervals with $t_i = i\Delta t$, for $i = 0, \ldots, N_t$, and the range for y is chosen to be $[-N, N]$ with N being a large positive number, which is further divided into N_y equally spaced intervals with $y_j = -N + j\Delta y$ for $j = 0, \ldots, N_y$. Thus, $u_{i,j}$ denotes the numerical value at the mesh point (t_i, y_j), namely, $u_{i,j} := u(t_i, y_j)$ for $i = 0, \ldots, N_t, j = 0, \ldots, N_y$.

In the following, let $u_{i,j}^k$ denotes the k-th iteration value of $u_{i,j}$. The terminal condition is

$$u_{N_t,j} = g(e^{y_j}) \tag{E.1}$$

for $j = 0, \ldots, N_y$. Similarly, the boundary conditions at y_0 and y_{N_y} are set to be

$$u_{i,0} = g(e^{y_0}), \text{ and } u_{i,N_y} = g(e^{y_{N_y}}) \tag{E.2}$$

for $i = 0, \ldots, N_t$. The Black–Scholes operator $v_t(t, x) + \frac{1}{2}\sigma^2 x^2 v_{xx}(t, x) + \rho x v_x(t, x) - \rho v(t, x)$ changes to

$$\mathcal{L}_{BS} u(t, y) := u_t(t, y) + \frac{1}{2}\sigma^2 u_{yy}(t, y) + \left(\rho - \frac{1}{2}\sigma^2\right) u_y(t, y) - \rho u(t, y).$$

By the fully implicit finite difference scheme,

$$\begin{aligned}
\frac{\partial}{\partial t} u(t_i, y_j) &= \frac{u_{i+1,j} - u_{i,j}}{\Delta t}, \\
\frac{\partial}{\partial y} u(t_i, y_j) &= \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y}, \\
\frac{\partial^2}{\partial y^2} u(t_i, y_j) &= \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta y)^2},
\end{aligned}$$

for $j = 1, \ldots, N_y - 1$ and $i = 0, \ldots, N_t - 1$. The discretized Black–Scholes operator is therefore

$$\mathcal{L}_{BS} u_{i,j} = \frac{u_{i+1,j} - u_{i,j}}{\Delta t} + \beta u_{i,j+1} - (\alpha + \beta + \rho) u_{i,j} + \alpha u_{i,j-1}, \tag{E.3}$$

where

$$\alpha = \frac{\sigma^2}{2(\Delta y)^2} - \frac{\rho - \frac{1}{2}\sigma^2}{2\Delta y}, \text{ and } \beta = \frac{\sigma^2}{2(\Delta y)^2} + \frac{\rho - \frac{1}{2}\sigma^2}{2\Delta y}.$$

The PDE in (2.5) becomes

$$-\mathcal{L}_{BS} u(t, y) = K\left(g(e^y) - u(t, y)\right)^+. \tag{E.4}$$

For each mesh point (t_i, y_j), we use the following approximation by Newton iteration:

$$\begin{aligned}\left(g(e^{y_j}) - u_{i,j}^{k+1}\right)^+ &= \left(g(e^{y_j}) - u_{i,j}^k + u_{i,j}^k - u_{i,j}^{k+1}\right)^+ \\ &\approx \left(g(e^{y_j}) - u_{i,j}^k\right)^+ + 1_{g(e^{y_j})-u_{i,j}^k>0}\left(u_{i,j}^k - u_{i,j}^{k+1}\right) \\ &= \left(g(e^{y_j}) - u_{i,j}^{k+1}\right) 1_{g(e^{y_j})-u_{i,j}^k>0},\end{aligned}$$

where 1_A is the indicator function of a set A. By (E.3) and (E.5), it turns out that (E.4) reads

$$-\frac{u_{i+1,j} - u_{i,j}^{k+1}}{\Delta t} - \beta u_{i,j+1}^{k+1} + (\alpha + \beta + \rho)\, u_{i,j}^{k+1} - \alpha u_{i,j-1}^{k+1} = K\left(g(e^{y_j}) - u_{i,j}^{k+1}\right) 1_{g(e^{y_j})-u_{i,j}^k>0},$$

which simplifies to

$$-\beta u_{i,j+1}^{k+1} + \left(\frac{1}{\Delta t} + \alpha + \beta + \rho + K 1_{g(e^{y_j})-u_{i,j}^k>0}\right) u_{i,j}^{k+1} - \alpha u_{i,j-1}^{k+1} = \frac{u_{i+1,j}}{\Delta t} + K g(e^{y_j}) 1_{g(e^{y_j})-u_{i,j}^k>0}. \tag{E.5}$$

In each iterative step, the quantities in the RHS of (E.5) regarding future values $u_{i+1,j}$ for all j and previous estimates $u_{i,j}^k$ for all i and j are known, and the current estimates $u_{i,j}^{k+1}$ are obtained by solving the equation (E.5). See Algorithm 3 for details.

Algorithm 3: Finite difference scheme for (2.5)

Inputs: $\Delta t, \Delta y, N, N_t, N_y, K$ and the tolerance level $\varepsilon > 0$.
Learning procedure:
Initialize $u_{N_t,j}$ for all j by (E.1) as well as $u_{i,0}$ and u_{i,N_y} for all i by (E.2).
for $i = N_t - 1, \ldots, 0$ **do**
  Set $u_{i,j}^0 = u_{i+1,j}$ for $j = 1, \ldots, N_y - 1$.
  for $k = 0, 1, \ldots$ **do**
    Solve the equation (E.5) to get $u_{i,j}^{k+1}$ for $j = 1, \ldots, N_y - 1$.
    if $||u_i^{k+1} - u_i^k||_\infty < \varepsilon * \max\left\{1,\ ||u_i^k||_\infty\right\}$ **then**
      Quit the loop for k.
    end
  end
  Set $u_{i,j} = u_{i,j}^{k+1}$ for $j = 1, \ldots, N_y - 1$.
end

References

Ajdari, A., Niyazi, M., Nicolay, N. H., Thieke, C., Jeraj, R., and Bortfeld, T. (2019). Towards optimal stopping in radiation therapy. *Radiotherapy and Oncology*, 134:96–100.

Barberis, N. (2012). A model of casino gambling. *Management Science*, 58(1):35–51.

Becker, S., Cheridito, P., and Jentzen, A. (2019). Deep optimal stopping. *Journal of Machine Learning Research*, 20(74):1–25.

Becker, S., Cheridito, P., and Jentzen, A. (2020). Pricing and hedging American-style options with deep learning. *Journal of Risk and Financial Management*, 13(7):158.

Becker, S., Cheridito, P., Jentzen, A., and Welti, T. (2021). Solving high-dimensional optimal stopping problems using deep learning. *European Journal of Applied Mathematics*, 32(3):470–514.

Dai, M. and Dong, Y. (2024). Learning an optimal investment policy with transaction costs via a randomized Dynkin game. *Available at SSRN 4871712*.

Dai, M., Dong, Y., and Jia, Y. (2023). Learning equilibrium mean-variance strategy. *Mathematical Finance*, 33(4):1166–1212.

Dai, M., Dong, Y., Jia, Y., and Zhou, X. Y. (2023). Learning Merton's strategies in an incomplete market: Recursive entropy regularization and biased Gaussian exploration. *arXiv preprint arXiv:2312.11797*.

Dai, M., Dong, Y., Jia, Y., and Zhou, X. Y. (2025). Data-driven Merton's strategies via policy randomization. *arXiv preprint arXiv:2312.11797*.

Dai, M., Kwok, Y. K., and You, H. (2007). Intensity-based framework and penalty formulation of optimal stopping problems. *Journal of Economic Dynamics and Control*, 31(12):3860–3880.

Dai, Z., Yu, H., Low, B. K. H., and Jaillet, P. (2019). Bayesian optimization meets Bayesian optimal stopping. *International Conference on Machine Learning*, PMLR: 1496–1506.

Dianetti, J., Ferrari, G., and Xu, R. (2024). Exploratory optimal stopping: A singular control formulation. *arXiv preprint arXiv:2408.09335*.

Dong, Y. (2024). Randomized optimal stopping problem in continuous time and reinforcement learning algorithm. *SIAM Journal on Control and Optimization*, 62(3):1590–1614.

Fleming, W. H. and Soner, H. M. (2006). *Controlled Markov Processes and Viscosity Solutions*, (Vol. 25). Springer Science & Business Media.

Forsyth, P. A. and Vetzal, K. R. (2002). Quadratic convergence for valuing American options using a penalty method. *SIAM Journal on Scientific Computing*, 23(6):2095–2122.

Friedman, A. (1982). *Variational Principles and Free Boundary Problems*, Wiley, New York.

Gao, X., Xu, Z. Q., and Zhou, X. Y. (2022). State-dependent temperature control for Langevin diffusions. *SIAM Journal on Control and Optimization*, 60(3):1250–1268.

Grigelionis, B. I., and Shiryaev, A. N. (1966). On Stefan's problem and optimal stopping rules for Markov processes. *Theory of Probability and Its Applications*, 11(4):541–558.

Han, J., Jentzen, A., and E, W. (2018). Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510.

He, X. D., Hu, S., Obłój, J., and Zhou, X. Y. (2017). Path-dependent and randomized strategies in Barberis' casino gambling model. *Operations Research*, 65(1):97–103.

Hu, S., Obłój, J., and Zhou, X. Y. (2023). A casino gambling model under cumulative prospect theory: Analysis and algorithm. *Management Science*, 69(4):2474–2496.

Herrera, C., Krach, F., Ruyssen, P., and Teichmann, J. (2024). Optimal stopping via randomized neural networks. *Frontiers of Mathematical Finance*, 3(1):31–77.

Jia, Y., Ouyang, D., and Zhang, Y. (2025). Accuracy of discretely sampled stochastic policies in continuous-time reinforcement learning. *arXiv preprint arXiv:2503.09981.*

Jia, Y. and Zhou, X. Y. (2022a). Policy evaluation and temporal-difference learning in continuous time and space: A martingale approach. *Journal of Machine Learning Research*, 23(154):1–55.

Jia, Y. and Zhou, X. Y. (2022b). Policy gradient and actor-critic learning in continuous time and space: Theory and algorithms. *Journal of Machine Learning Research*, 23(1):12603–12652.

Jia, Y. and Zhou, X. Y. (2023). q-learning in continuous time. *Journal of Machine Learning Research*, 24(161):1–61.

Jiang, R., Saunders, D., and Weng, C. (2022). The reinforcement learning Kelly strategy. *Quantitative Finance*, 22(8):1445–1464.

Liang, J., Hu, B., Jiang, L., and Bian, B. (2007). On the rate of convergence of the binomial tree scheme for American options. *Numerische Mathematik*, 107(2):333–352.

Liyanage, Y. W., Zois, D. S., Chelmis, C., and Yao, M. (2019) Automating the classification of urban issue reports: An optimal stopping approach. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3137–3141.

Munos, R. (2006). Policy gradient in continuous time. *Journal of Machine Learning Research*, 7:771–791.

Peng, Y., Wei, P., and Wei, W. (2024). Deep penalty methods: A class of deep learning algorithms for solving high dimensional optimal stopping problems. *arXiv preprint arXiv:2405.11392*.

Pham, H. (2009). *Continuous-time Stochastic Control and Optimization with Financial Applications*, Springer Science & Business Media.

Reppen, A. M., Soner, H. M., and Tissot-Daguette, V. (2025). Neural optimal stopping boundary. *Mathematical Finance*, 35(2):441–469.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*, MIT press.

Tallec, C., Blier, L., and Ollivier, Y. (2019). Making deep q-learning methods robust to time discretization. *Proceedings of the 36th International Conference on Machine Learning*, PMLR 97:6096–6104.

Wang, S., and Perdikaris, P. (2021). Deep learning of free boundary and Stefan problems. *Journal of Computational Physics*, 428:109914.

Wang, H., Zariphopoulou, T., and Zhou, X. Y. (2020). Reinforcement learning in continuous time and space: A stochastic control approach. *Journal of Machine Learning Research*, 21(1):8145–8178.

Wang, H. and Zhou, X. Y. (2020). Continuous-time mean–variance portfolio selection: A reinforcement learning framework. *Mathematical Finance*, 30(4):1273–1308.

Wu, B. and Li, L. (2024). Reinforcement learning for continuous-time mean-variance portfolio selection in a regime-switching market. *Journal of Economic Dynamics and Control*, 158:104787.

Yong, J. and Zhou, X. Y. (1999). *Stochastic Controls : Hamiltonian Systems and HJB Equations*, New York : Springer.

Zhang, K., Yang, X., and Wang, S. (2022). Solution method for discrete double obstacle problems based on a power penalty approach. *Journal of Industrial and Management Optimization*, 18(2):1261–1274.