Journal of Industrial and Management Optimization
Vol. 21, No. 9, September 2025, pp. 5687-5715
doi:10.3934/jimo.2025110

PONTRYAGIN-GUIDED DIRECT POLICY OPTIMIZATION FOR CONTINUOUS-TIME PORTFOLIO PROBLEM

Jeonggyu Huh[1], Seungwon Jeong[2] and Jaegi Jeon[*3]

[1]Department of Mathematics, Sungkyunkwan University, Republic of Korea

[2]Global-Learning & Academic research institution for Master's·PhD students, and Postdocs, Chonnam National University, Republic of Korea

[3]Graduate School of Data Science, Chonnam National University, Republic of Korea

(Communicated by Phillip Yam)

Abstract. We present Pontryagin-Guided Direct Policy Optimization (PG-DPO), a framework for solving continuous-time portfolio optimization problems involving both consumption and investment decisions. Integrating Pontryagin's Maximum Principle (PMP) within a neural network pipeline, PG-DPO bypasses traditional value function approximation and directly optimizes policy parameters using adjoint processes associated with the current policy, computed via automatic differentiation. An optional alignment penalty, explicitly derived from PMP conditions, significantly accelerates convergence and improves policy stability during training. Numerical experiments validate the framework's efficacy: PG-DPO accurately recovers the closed-form solution for the classical Merton problem and, crucially, demonstrates its capability to handle more realistic, state-dependent dynamics involving stochastic factors, effectively capturing intertemporal hedging demands. These results highlight that the PMP-guided deep learning approach offers an effective and potentially efficient pathway for direct policy optimization in complex continuous-time stochastic control settings within finance.

1. **Introduction.** Merton's portfolio optimization problem [18] remains a cornerstone of mathematical finance, guiding optimal investment and consumption decisions in continuous time. Classical treatments (e.g., [13, 30, 21]) derived seminal closed-form solutions under specific assumptions like constant market coefficients and complete markets. However, real-world markets often exhibit more complex dynamics, such as time-varying or stochastic investment opportunities driven by external factors, motivating numerous extensions (e.g., [14, 28, 17, 19, 27]). While analytical solutions exist for certain classes of these extended models (e.g., affine or quadratic models, [14, 17]), many realistic scenarios involving intricate dynamics, multiple factors, or constraints lack tractable closed-form solutions, necessitating the use of numerical and data-driven methods.

2020 *Mathematics Subject Classification.* Primary: 91G10, 93E20, 68T07.

Key words and phrases. Merton portfolio problem, Consumption-Investment Problem, Pontryagin's Maximum Principle, Direct Policy Optimization, Stochastic Control, Neural Network.

*Corresponding author: Jaegi Jeon.

Parameterizing investment and consumption policies using neural networks and optimizing via gradient-based methods offers a promising data-driven approach. This approach is particularly effective for problems involving high-dimensional state spaces, path dependence, or market frictions [9, 10, 1, 4, 2, 31, 24, 25]. Nevertheless, purely empirical or black-box gradient methods may overlook the underlying continuous-time optimal control structure. Without grounding in classical control principles, convergence to a theoretically sound policy is not guaranteed (cf. [24, 25]). Furthermore, some approaches, like model-free reinforcement learning (RL), often require extensive exploration and may simplify the problem by omitting consumption [6].

In response to these limitations, a prominent alternative is the deep BSDE methodology (e.g., [10, 29, 12]), which focuses on approximating the value function by solving the associated backward stochastic differential equation (BSDE) derived from the Hamilton-Jacobi-Bellman (HJB) equation. The optimal policy is then inferred from the estimated value function. In contrast, our proposed Pontryagin-Guided Direct Policy Optimization (PG-DPO) framework is fundamentally adjoint-based and pursues direct policy optimization. Instead of explicitly approximating the value function, our framework leverages Pontryagin's Maximum Principle (PMP) by computing adjoint (costate) processes associated with current suboptimal policies directly through automatic differentiation (AD) and backpropagation-through-time (BPTT). We then use these adjoint sensitivities to explicitly guide and update the policy network parameters. This distinction in learning targets (adjoint process vs. value function) and update mechanisms (direct policy gradient vs. inferred policy) potentially offers different trade-offs in terms of interpretability, computational efficiency, and direct control over the policy structure.

Despite significant advances in deep learning methods for finance, existing neural network approaches (e.g., Deep BSDE, Physics-Informed Neural Networks (PINNs), RL) have primarily tackled simplified or investment-only portfolio problems. Consequently, developing a unified, theoretically grounded neural framework capable of simultaneously addressing consumption and investment decisions remains a challenging open problem. Our work addresses this gap by integrating PMP with a neural network solver designed to handle the dual controls of consumption and investment simultaneously and coherently.

We develop the PG-DPO framework, treating the portfolio problem as a continuous-time stochastic control system solved via direct neural network optimization. By embedding PMP principles within the training loop, interpreting gradient steps through the lens of adjoint processes, we provide a control-theoretically motivated approach to policy improvement that retains the flexibility of deep learning and AD. A key innovation is the optional alignment penalty derived directly from PMP's first-order conditions (FOCs). By explicitly regularizing the policy towards PMP-derived controls, this penalty significantly enhances convergence speed and training stability, as confirmed by our numerical experiments.

Critically, our numerical experiments validate the PG-DPO framework on two benchmarks: the classical Merton problem with constant coefficients and, importantly, the Kim & Omberg model involving stochastic investment opportunities. This second experiment explicitly demonstrates the framework's ability to effectively capture intertemporal hedging demands and state-dependent dynamics. The

results demonstrate reliable convergence to policies strongly aligned with PMP conditions, achieved within computationally practical training times.

The main contributions of this paper can be summarized as follows: (i) We propose the PG-DPO scheme, integrating PMP into a neural direct policy optimization framework for continuous-time portfolio problems. (ii) We introduce an adjoint-based alignment penalty to improve training dynamics. (iii) We demonstrate the method's effectiveness in solving the challenging dual-control (consumption-investment) problem and, significantly, its capability to handle state-dependent dynamics arising from stochastic factors. (iv) We explicitly extend the PG-DPO framework beyond purely Markovian settings to accommodate non-Markovian, path-dependent dynamics through Markovization, specifically illustrating how common momentum factors can be seamlessly integrated via state augmentation and corresponding adjoint sensitivities. This extension substantially broadens the applicability and flexibility of our approach, which also holds promise for scenarios involving constraints and multiple assets.

The remainder of the paper is structured as follows: Section 2 details the generalized problem formulation including stochastic factors. Section 3 revisits PMP for this generalized setting. Section 4 explains the PG-DPO algorithm, including discretization, gradient computation, and the alignment penalty. Section 5 presents numerical results for both the classical Merton problem and the Kim & Omberg model. Section 5.2 illustrates the extension of PG-DPO to non-Markovian momentum dynamics via Markovization. Finally, Section 6 concludes the paper and discusses future research directions.

2. Problem formulation.

We consider a general continuous-time portfolio optimization problem set within a filtered probability space $(\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{t\in[0,T]}, \mathbb{P})$ supporting a standard two-dimensional Brownian motion $\mathbf{W}_t = (W_{X,t}, W_{Y,t})^\top$. Here, $W_{X,t}$ drives the risky asset price impacting wealth X_t, $W_{Y,t}$ drives the external stochastic factor Y_t, and their instantaneous correlation is given by $\mathbb{E}[dW_{X,t}dW_{Y,t}] = \rho_{XY}dt$, where $\rho_{XY} \in [-1, 1]$. An investor allocates wealth between a risk-free asset and a single risky asset, where the market environment can be influenced by the external factor. The investor's goal is to maximize the expected utility derived from intermediate consumption and terminal wealth over the finite horizon $[0, T]$.

The market consists of a risk-free asset B_t providing a constant interest rate r, whose price evolves according to $dB_t = rB_t dt$ with $B_0 = 1$, and a risky asset S_t. The price dynamics of the risky asset are given by the stochastic differential equation (SDE)

$$\frac{dS_t}{S_t} = \mu(t, Y_t)dt + \sigma(t, Y_t)dW_{X,t}, \tag{1}$$

where the drift $\mu(t, Y_t)$ and volatility $\sigma(t, Y_t)$ (>0) are adapted processes potentially depending on time t and the stochastic state process Y_t. This process Y_t represents external factors influencing the investment opportunity set, such as stochastic risk premiums or volatility levels, and is assumed to follow a general Itô process

$$dY_t = b(t, Y_t)dt + a(t, Y_t)dW_{Y,t}, \quad Y_0 = y_0, \tag{2}$$

with drift $b(t, y)$ and diffusion $a(t, y)$. For clarity and to establish the core methodology, the theoretical exposition and numerical examples in this paper focus on cases with at most one stochastic factor (Y_t). Extending the framework and analysis to settings with multi-dimensional factors remains an important direction for

future research. Similarly, incorporating non-Markovian features arising from path-dependencies, such as financial momentum, represents another significant extension. As detailed in Section 5.2, such path-dependent dynamics can often be effectively handled within the PG-DPO framework by augmenting the state space to construct a Markovian representation, to which our PMP-guided methodology directly applies.

Let X_t denote the investor's total wealth at time t, with $X_0 = x_0 > 0$. The investor controls the proportion π_t of wealth invested in the risky asset and the rate of consumption C_t. The resulting wealth dynamics are governed by the SDE

$$dX_t = [rX_t + \pi_t(\mu(t, Y_t) - r)X_t - C_t]\, dt + \pi_t \sigma(t, Y_t) X_t dW_{X,t}. \tag{3}$$

The control processes $(\pi_t, C_t)_{t\in[0,T]}$ must be admissible, meaning they are adapted, satisfy integrability conditions for the existence and uniqueness of a strong solution to (3), and adhere to any problem-specific constraints such as $C_t \geq 0$ and maintaining X_t within the domain of the utility function.

The investor seeks to maximize the objective functional $J(\pi, C)$, defined as the expected total discounted utility from consumption and terminal wealth:

$$J(\pi, C) \coloneqq \mathbb{E}\left[\int_0^T e^{-\rho t} U(C_t) dt + \kappa e^{-\rho T} U(X_T)\right], \tag{4}$$

where $U(\cdot)$ is a strictly increasing and strictly concave utility function ($U' > 0, U'' < 0$), $\rho > 0$ is the discount rate, and $\kappa \geq 0$ weights the bequest motive. Specific forms, often from the Hyperbolic Absolute Risk Aversion (HARA) class such as CRRA or CARA, will be employed in our numerical examples.

In our proposed framework, we approximate the optimal policies using neural networks parameterized by θ and ϕ. These networks take the current state information (t, X_t, Y_t) as input to determine the investment proportion and consumption amount:

$$\pi_t = \pi_\theta(t, X_t, Y_t), \quad C_t = C_\phi(t, X_t, Y_t). \tag{5}$$

Network architectures can be designed to inherently satisfy constraints like $C_t \geq 0$ using appropriate activation functions.

For theoretical context, the optimal policies (π^*, C^*) can often be characterized using dynamic programming. Let $V(t, x, y)$ denote the value function, representing the maximum achievable expected utility from state (t, x, y). Under sufficient regularity, V satisfies the Hamilton-Jacobi-Bellman (HJB) partial differential equation:

$$\begin{aligned} &-\rho V + V_t + b(t, y)V_y + \frac{1}{2}a(t, y)^2 V_{yy} \\ &\quad + \sup_{\pi, C\geq 0}\Big\{\big[rx + \pi(\mu(t, y) - r)x - C\big]V_x \\ &\qquad + \frac{1}{2}\pi^2\sigma(t, y)^2 x^2 V_{xx} + \rho_{XY}\pi\sigma(t, y)a(t, y)xV_{xy} + U(C)\Big\} = 0 \end{aligned} \tag{6}$$

subject to the terminal condition $V(T, x, y) = \kappa U(x)$. The optimal controls that attain the supremum in (6) typically take the form:

$$U'(C^*) = V_x(t, x, y), \tag{7}$$

$$\pi^*(t,x,y) = \underbrace{-\frac{(\mu(t,y)-r)V_x}{\sigma(t,y)^2 x V_{xx}}}_{\text{Myopic Demand}} \underbrace{-\frac{\rho_{XY}a(t,y)V_{xy}}{\sigma(t,y)xV_{xx}}}_{\text{Hedging Demand}}. \tag{8}$$

These expressions reveal that the optimal policies depend intrinsically on the value function and its partial derivatives. While the structure of the optimal policy can be characterized via the HJB equation (6), obtaining an analytical or numerical solution for the value function V or its derivatives V_x, V_{xx}, V_{xy} can still be challenging, requiring specific techniques tailored to the problem structure. Our PG-DPO approach, detailed in the subsequent sections, offers an alternative pathway. It aims to directly learn the policy networks π_θ and C_ϕ by leveraging the necessary conditions derived from PMP within a computational learning framework, thus bypassing the explicit solution of the HJB equation. This approach provides a unified methodology, as demonstrated by its application to both the classical Merton problem and the Kim & Omberg model presented later.

3. Pontryagin's maximum principle for the portfolio problem.

In this section, we detail the application of PMP to the general portfolio optimization problem formulated in Section 2. PMP provides the necessary conditions for optimality and introduces the concept of adjoint processes, which are central to our gradient computation methodology. We first outline the PMP framework for our specific problem setting (Section 3.1) and then discuss the policy-fixed adjoint processes used for iterative policy improvement (Section 3.2).

3.1. PMP formulation and necessary conditions.

We now apply PMP to the general portfolio optimization problem defined in Section 2. PMP provides necessary conditions for optimality by introducing adjoint processes alongside the state dynamics defined by (3) and (2).

Recall the system state is the vector $\mathbf{X}_t = (X_t, Y_t)^\top$, the control is the vector $\mathbf{u}_t = (\pi_t, C_t)^\top$, and the dynamics $d\mathbf{X}_t = \mathbf{f}(t,\mathbf{X}_t,\mathbf{u}_t)dt + \boldsymbol{\sigma}(t,\mathbf{X}_t,\mathbf{u}_t)d\mathbf{W}_t$ are driven by the 2D Brownian motion $\mathbf{W}_t = (W_{X,t}, W_{Y,t})^\top$ with correlation ρ_{XY}. The drift vector $\mathbf{f}$ and diffusion matrix $\boldsymbol{\sigma}$ are explicitly given by:

$$\mathbf{f}(t,(x,y),(\pi,C)) = \begin{pmatrix} rx + \pi(\mu(t,y)-r)x - C \\ b(t,y) \end{pmatrix},$$

$$\boldsymbol{\sigma}(t,(x,y),\pi) = \begin{pmatrix} \pi\sigma(t,y)x & 0 \\ 0 & a(t,y) \end{pmatrix}.$$

The objective is to maximize the functional $J(\pi, C)$ given by (4).

PMP introduces the costate vector $\boldsymbol{\lambda}_t = (\lambda_t^X, \lambda_t^Y)^\top$ and the associated 2×2 matrix process $\mathbf{Z}_t$, explicitly given by

$$\mathbf{Z}_t = \begin{pmatrix} Z_{X,t}^X & Z_{Y,t}^X \\ Z_{X,t}^Y & Z_{Y,t}^Y \end{pmatrix},$$

where Z_B^A denotes the sensitivity of $d\lambda_t^A$ to $dW_{B,t}$. The Hamiltonian $\mathcal{H}(t,\mathbf{X},\mathbf{u},\boldsymbol{\lambda},\mathbf{Z})$ is defined as:

$$\begin{aligned}\mathcal{H} \coloneqq\ & e^{-\rho t}U(C) + \lambda^X\left[rx + \pi(\mu(t,y)-r)x - C\right] \\ & + \lambda^Y b(t,y) + Z_X^X \pi\sigma(t,y)x + Z_Y^X a(t,y).\end{aligned} \tag{9}$$

Let (π_t^*, C_t^*) be the optimal controls and $(\mathbf{X}_t^*, \boldsymbol{\lambda}_t^*, \mathbf{Z}_t^*)$ the corresponding optimal processes. The optimal adjoint processes satisfy the vector BSDE system:

$$d\boldsymbol{\lambda}_t^* = -\nabla_{\mathbf{x}}\mathcal{H}(t, \mathbf{X}_t^*, \mathbf{u}_t^*, \boldsymbol{\lambda}_t^*, \mathbf{Z}_t^*)dt + \mathbf{Z}_t^* d\mathbf{W}_t, \tag{10}$$

with the terminal condition $\boldsymbol{\lambda}_T^* = (\kappa e^{-\rho T} U'(X_T^*), 0)^\top$, derived from the terminal payoff term in the objective function (4). The Hamiltonian gradient $\nabla_{\mathbf{x}}\mathcal{H}^* = (\partial\mathcal{H}^*/\partial x, \partial\mathcal{H}^*/\partial y)^\top$ has components:

$$\frac{\partial \mathcal{H}^*}{\partial x} = \lambda^{X,*}[r + \pi^*(\mu - r)] + Z_X^{X,*}\pi^*\sigma,$$

$$\frac{\partial \mathcal{H}^*}{\partial y} = \lambda^{X,*}[\pi^* x(\partial\mu/\partial y)] + \lambda^{Y,*}(\partial b/\partial y) + Z_X^{X,*}[\pi^* x(\partial\sigma/\partial y)] + Z_Y^{Y,*}(\partial a/\partial y).$$

The PMP condition requires that the optimal controls $\mathbf{u}_t^* = (\pi_t^*, C_t^*)$ must maximize the Hamiltonian (9) pointwise:

$$\mathcal{H}(t, ..., \pi_t^*, C_t^*, ...) = \sup_{\pi, C \geq 0} \mathcal{H}(t, ..., \pi, C, ...).$$

The FOC with respect to consumption C_t yields the familiar optimality condition:

$$\left.\frac{\partial \mathcal{H}}{\partial C_t}\right|_* = e^{-\rho t}U'(C_t^*) - \lambda_t^{X,*} = 0 \quad \Longrightarrow \quad U'(C_t^*) = e^{\rho t}\lambda_t^{X,*}. \tag{11}$$

Similarly, the FOC with respect to the investment proportion π_t, assuming an interior solution, is given by the equation:

$$\left.\frac{\partial \mathcal{H}}{\partial \pi_t}\right|_* = \lambda_t^{X,*}(\mu(t, Y_t^*) - r)X_t^* + Z_{X,t}^{X,*}\sigma(t, Y_t^*)X_t^* = 0, \tag{12}$$

$$\Longrightarrow \quad Z_{X,t}^{X,*} = -\lambda_t^{X,*}\frac{\mu(t, Y_t^*) - r}{\sigma(t, Y_t^*)}.$$

These FOCs implicitly define the optimal controls (π_t^*, C_t^*). While the condition for C_t^* (11) is straightforward, deriving an explicit expression for π_t^* from the FOC (12) requires utilizing the relationship between the adjoint diffusion component $Z_{X,t}^{X,*}$ and the derivatives of the costate $\lambda_t^{X,*}$, which arises from applying Itô's lemma. As derived in Appendix A , this connection allows for the explicit decomposition of the optimal investment proportion π_t^* into two components:

$$\pi_t^* = \underbrace{\frac{\mu(t, Y_t^*) - r}{\sigma(t, Y_t^*)^2}\left(\frac{-\lambda_t^{X,*}}{X_t^*(\partial\lambda_t^{X,*}/\partial x)}\right)}_{\text{Myopic Demand}} \underbrace{-\frac{\rho_{XY}a(t, Y_t^*)(\partial\lambda_t^{X,*}/\partial y)}{\sigma(t, Y_t^*)X_t^*(\partial\lambda_t^{X,*}/\partial x)}}_{\text{Hedging Demand}}. \tag{13}$$

This decomposition into myopic and intertemporal hedging demands aligns perfectly with the structure derived from the HJB approach (8), highlighting the consistency between the methodologies. Understanding this theoretical structure derived from PMP informs the interpretation of the learned policy and the role of the adjoint processes in our PG-DPO method.

3.2. **Policy-fixed adjoint processes and iterative policy improvement.** While PMP characterizes the optimal policy (π_t^*, C_t^*) using adjoint processes $(\boldsymbol{\lambda}_t^*, \mathbf{Z}_t^*)$ derived under optimality, we can define analogous adjoint processes for any fixed, potentially suboptimal policy parameterized by (θ, ϕ), as given by $\pi_t = \pi_\theta(t, X_t, Y_t)$ and $C_t = C_\phi(t, X_t, Y_t)$. This concept of policy-fixed adjoints is crucial for our gradient-based optimization.

For a given policy (π_θ, C_ϕ), we define the corresponding policy-fixed adjoint processes $(\boldsymbol{\lambda}_t^{\theta,\phi}, \mathbf{Z}_t^{\theta,\phi})$ as the solution to the BSDE:

$$d\boldsymbol{\lambda}_t = -\nabla_{\mathbf{x}}\mathcal{H}(t, \mathbf{X}_t, \mathbf{u}_t^{\theta,\phi}, \boldsymbol{\lambda}_t, \mathbf{Z}_t)dt + \mathbf{Z}_t d\mathbf{W}_t, \quad \boldsymbol{\lambda}_T = (\kappa e^{-\rho T}U'(X_T), 0)^\top, \tag{14}$$

where $\mathbf{X}_t$ follows the dynamics driven by $\mathbf{u}_t^{\theta,\phi} = (\pi_\theta, C_\phi)^\top$, and $\nabla_{\mathbf{x}}\mathcal{H} = (\partial\mathcal{H}/\partial x, \partial\mathcal{H}/\partial y)^\top$ is evaluated using the current policy and the corresponding adjoints. This adjoint vector $\boldsymbol{\lambda}_t^{\theta,\phi} = (\lambda_t^{X,\theta,\phi}, \lambda_t^{Y,\theta,\phi})^\top$ represents the sensitivity of the objective functional $J(\theta, \phi)$ to perturbations in the state $\mathbf{X}_t = (X_t, Y_t)^\top$ under the fixed policy (θ, ϕ), while the matrix process $\mathbf{Z}_t^{\theta,\phi}$ reflects how this sensitivity itself fluctuates due to the underlying stochastic shocks $\mathbf{W}_t$.

These policy-fixed adjoints are instrumental in computing the gradients $\nabla_\theta J$ and $\nabla_\phi J$, which guide the policy updates. Applying the chain rule (or adjoint method for stochastic processes), the gradients can be expressed via expectations involving the adjoints and policy derivatives:

$$\frac{\partial J}{\partial \theta} = \mathbb{E}\left[\int_0^T \left(\frac{\partial \mathcal{H}}{\partial \pi}\bigg|_{\mathbf{u}^{\theta,\phi},\boldsymbol{\lambda}^{\theta,\phi},\mathbf{Z}^{\theta,\phi}}\right) \frac{\partial \pi_\theta(t, X_t, Y_t)}{\partial \theta} dt\right], \tag{15}$$

$$\frac{\partial J}{\partial \phi} = \mathbb{E}\left[\int_0^T \left(\frac{\partial \mathcal{H}}{\partial C}\bigg|_{\mathbf{u}^{\theta,\phi},\boldsymbol{\lambda}^{\theta,\phi},\mathbf{Z}^{\theta,\phi}}\right) \frac{\partial C_\phi(t, X_t, Y_t)}{\partial \phi} dt\right], \tag{16}$$

where the relevant partial derivatives of the Hamiltonian, evaluated at the current policy (π_θ, C_ϕ) and corresponding adjoint processes $(\boldsymbol{\lambda}_t^{\theta,\phi}, \mathbf{Z}_t^{\theta,\phi})$, represent the deviation from the FOCs derived in Section 3.1. Specifically, the term for consumption is

$$\frac{\partial \mathcal{H}}{\partial C} = e^{-\rho t}U'(C_\phi) - \lambda_t^X, \tag{17}$$

which corresponds to the left-hand side of the optimal consumption condition (11). Similarly, the term for investment is

$$\frac{\partial \mathcal{H}}{\partial \pi} = \lambda_t^X(\mu(t, Y_t) - r)X_t + Z_{X,t}^X \sigma(t, Y_t)X_t, \tag{18}$$

corresponding to the left-hand side of the optimal investment condition (12). These terms (17)-(18) measure the extent to which the current policy locally deviates from satisfying the PMP FOCs.

Repeating gradient ascent steps using these computed gradients aims to drive $\partial\mathcal{H}/\partial\pi$ and $\partial\mathcal{H}/\partial C$ towards zero, thus iteratively improving the policy (π_θ, C_ϕ) towards optimality.

Crucially, our PG-DPO framework utilizes AD applied to the computational graph of the forward simulation (will be discussed in Section 4.2) to obtain the necessary sensitivities $\boldsymbol{\lambda}_k = \partial J/\partial \mathbf{X}_k$ for the discretized problem. This avoids the significant challenge of numerically solving the potentially high-dimensional BSDE (14) at each iteration, while still leveraging the gradient information prescribed by PMP. This distinguishes our approach from value-based methods like deep BSDE

[29, 10, 12], as we directly target policy improvement guided by adjoint sensitivities rather than approximating the value function first.

4. **Gradient-based algorithm for stationary point convergence.** In this section, we describe the practical implementation of our PG-DPO approach for the general portfolio optimization problem formulated in Section 2. We detail how the continuous-time system is discretized, how gradients are computed using BPTT, and how the policy-fixed adjoint processes, $\boldsymbol{\lambda}_t$ and matrix $\mathbf{Z}_t$, guide the policy networks $\pi_\theta(t, X_t, Y_t)$ and $C_\phi(t, X_t, Y_t)$ towards a Pontryagin-aligned solution. We also introduce an alignment penalty designed to steer the learned controls towards locally optimal controls derived from PMP, thereby potentially improving convergence and stability.

The structure of this section is as follows. We first discuss the single-path approach for estimating the necessary adjoint information for a given state (t, X_t, Y_t) (Section 4.1). We then detail the discrete-time gradient computation algorithm using BPTT (Section 4.2). Section 4.3 explains the formulation and incorporation of the adjoint-based alignment penalty. Subsequently, Section 4.4 discusses the extension to handle arbitrary initial conditions (t_0, X_0, Y_0) using an extended value function concept and outlines the two main algorithmic variants: PG-DPO and PG-DPO-Align. Overall, by integrating PMP principles within a flexible BPTT-based learning framework, our approach aims to find near-optimal policies for the continuous-time portfolio problem.

4.1. **Single-path approach for each state $(t, \mathbf{X}_t)$.** A central insight of our method is that a single forward simulation (or single path) starting from a given state $(t, \mathbf{X}_t)$ can produce unbiased estimates of the sensitivities required for gradient computation. Specifically, AD applied to the simulation path allows us to compute the components of the adjoint vector $\lambda_t^X = \partial J / \partial X_t$, which measures how changes in wealth X_t affect the total expected cost J. The matrix process $\mathbf{Z}_t$, representing the sensitivity of the adjoint vector $\boldsymbol{\lambda}_t$ to the Brownian motions $dW_{X,t}$ and $dW_{Y,t}$, also plays a role defined by the PMP framework. While AD implicitly calculates the necessary influence of these terms for the policy gradients (as shown in Section 4.2), explicitly estimating components of $\mathbf{Z}_t$ (like $Z_{X,t}^X$) might be needed for constructing the alignment penalty (Section 4.3).

Using just one forward simulation path from each sampled state node $(t_k, \mathbf{X}_k)$ offers several computational advantages. First, although the estimates derived from a single path are inherently noisy, averaging over many such paths (e.g., in a mini-batch) leads to estimates that converge to the true expected sensitivities [16, 3]. Second, the computational overhead per sample is minimal, as we avoid simulating large ensembles for the same starting node; each node simply triggers one forward simulation run. Third, this approach naturally supports online learning and adaptation: as the policy parameters (θ, ϕ) evolve, we continuously generate fresh simulation paths reflecting the updated policy and potentially changing market conditions.

However, the primary drawback of this approach is high variance. The adjoint sensitivity estimates derived from individual paths can fluctuate significantly, leading directly to noisy averaged gradient estimates $(\nabla_\theta J, \nabla_\phi J)$. Such noise can slow down convergence and often necessitate smaller learning rates. While averaging over larger mini-batches helps mitigate this issue, variance reduction strategies, such as

the alignment penalty discussed later (Section 4.3), can offer further stabilization. We now detail the discrete-time BPTT algorithm that integrates this single-path simulation methodology (Section 4.2).

4.2. **Discrete-time algorithm for gradient computation.** We now present a concrete procedure, in discrete time, for computing the necessary gradient information for the policy parameters (θ, ϕ) in the generalized portfolio problem (Section 2). This algorithm relies on BPTT applied to discretized state trajectories and utilizes the sensitivity information interpreted through the lens of the policy-fixed adjoint processes $(\boldsymbol{\lambda}_t, \mathbf{Z}_t)$.

(a) **Discretize dynamics and objective.** Partition the interval $[0, T]$ into N steps of size $\Delta t = T/N$. Define $t_k = k\Delta t$ for $k = 0, \ldots, N$. Approximate the state SDEs (3) and (2) using the Euler–Maruyama scheme. Let $\mathbf{X}_k = (X_k, Y_k)^\top$. Generate correlated Brownian increments $(\Delta W_{X,k}, \Delta W_{Y,k})^\top \sim \mathcal{N}(\mathbf{0}, \Delta t \cdot \Sigma_W)$, where $\Sigma_W = [[1, \rho_{XY}], [\rho_{XY}, 1]]$. The state update is:

$$X_{k+1} = X_k + [rX_k + \pi_k(\mu(t_k, Y_k) - r)X_k - C_k]\,\Delta t + \pi_k \sigma(t_k, Y_k) X_k \Delta W_{X,k},$$

$$Y_{k+1} = Y_k + b(t_k, Y_k)\Delta t + a(t_k, Y_k)\Delta W_{Y,k}, \tag{19}$$

where $\pi_k = \pi_\theta(t_k, X_k, Y_k)$ and $C_k = C_\phi(t_k, X_k, Y_k)$. While this Euler-Maruyama scheme is generally applied, ensuring non-negative wealth ($X_k > 0$) requires particular attention when specific utility functions, such as the CRRA are employed. For such cases, we implement the exponential Euler method to $\log X_t$. This approach is often combined with parameterizing consumption as a fraction of wealth, i.e., $C_k = c_k X_k$ where $c_k = c_\phi(t_k, X_k, Y_k)$. The continuous-time cost (4) is discretized as:

$$J(\theta, \phi) \approx \mathbb{E}\left[\sum_{k=0}^{N-1} e^{-\rho t_k}\, U(C_k)\,\Delta t + \kappa e^{-\rho T}\, U(X_N)\right].$$

(b) **Forward path simulation and sensitivity computation via BPTT.** First, generate a full simulation trajectory $\{\mathbf{X}_j\}_{j=0}^N$ starting from an initial node $(t_0, \mathbf{X}_0)$ and running until time T, using the dynamics in step (a). This simulation defines a computational graph mapping the parameters (θ, ϕ) and the initial state $\mathbf{X}_0$ to the final objective $J(\theta, \phi)$. Applying AD, specifically BPTT, to this graph yields the overall policy gradients $\nabla_\theta J$ and $\nabla_\phi J$. Crucially, the BPTT process also computes the sensitivity of the objective J with respect to the state at each intermediate time step k ($0 \le k \le N$), denoted $\partial J/\partial \mathbf{X}_k = (\partial J/\partial X_k, \partial J/\partial Y_k)^\top$. Consistent with the Pontryagin interpretation, we identify the key component $\lambda_k^X = \partial J/\partial X_k$ as the adjoint variable related to wealth at time t_k.

(c) **Compute higher-order derivatives (for PG-DPO-Align only).** For the PG-DPO-Align variant (will be discussed in Section 4.3), constructing the alignment penalty typically requires additional information beyond the standard gradients. This may involve computing higher-order derivatives (e.g., $\partial \lambda_k^X/\partial x$ and $\partial \lambda_k^X/\partial y$) using further applications of AD, which incurs extra computational cost. Standard PG-DPO does not require this step.

(d) **Update network parameters.** The policy gradients $\nabla_\theta J$ and $\nabla_\phi J$ computed via BPTT correspond to the theoretical expressions (15)-(16), which involve the Hamiltonian derivatives defined in Section 3.2 (cf. Eqs. (17) and

(18)). AD directly yields the numerical values for these gradients. Note that AD, by differentiating through the stochastic term in the wealth dynamics (3), implicitly incorporates the sensitivity corresponding to the $Z^X_{X,k}$ term in the theoretical gradient expression (18). Finally, the parameters (θ, ϕ) are updated using a stochastic optimizer (e.g., Adam, SGD) with the computed policy gradients $(\nabla_\theta \widetilde{J}, \nabla_\phi \widetilde{J})$. The expectation $\mathbb{E}[\cdots]$ inherent in the theoretical gradient expressions (cf. Eqs. (15)-(16)) is approximated empirically by averaging the gradient contributions calculated from a mini-batch of M independent simulation paths:

$$\mathbb{E}\Big[\cdots\Big] \approx \frac{1}{M}\sum_{i=1}^{M}\Big[\cdots\Big]_i.$$

This iterative process of gradient estimation and parameter update is repeated until the policy converges.

This algorithm effectively performs BPTT on the forward dynamics (3)-(2) discretized using the Euler-Maruyama scheme (step (a) above). Crucially, instead of explicitly discretizing and solving the full coupled forward-backward system implied by the continuous-time PMP (Section 3.1), our approach instead leverages AD. Applying AD directly to the discretized forward computation automatically yields the required policy gradients and sensitivities such as $\lambda^X_k = \partial J/\partial X_k$.

The resulting gradient estimates, averaged over a mini-batch, are then used within the stochastic approximation framework described in step (d) and analyzed in Appendix B to update the policy parameters towards a stationary point. Interpreting the AD-computed sensitivities through the PMP lens, as discussed, offers valuable insights beyond black-box gradient computation and facilitates techniques like the alignment penalty (Section 4.3).

Remark: The sensitivity computed by AD, $\lambda^X_k = \partial J/\partial X_k$ satisfies the discrete adjoint BSDE required by PMP. A detailed derivation and justification are provided in Appendix C.

4.3. **Adjoint-based regularization for pontryagin alignment.** As discussed in Section 4.2, our approach utilizes AD to compute sensitivities, including estimates of the adjoint component $\lambda^X_k = \partial J/\partial X_k$. To implement the alignment penalty, we may need additional information derived from the PMP necessary conditions, specifically the locally optimal controls based on the current state and estimated adjoint information.

From the PMP FOC (11), which states $U'(C^*_t) = e^{\rho t}\lambda^{X,*}_t$, the Pontryagin-derived optimal consumption C^{PMP} at state $(t_k, \mathbf{X}_k = (X_k, Y_k)^\top)$ can be expressed using the inverse of the marginal utility function, denoted by $(U')^{-1}$, given the estimate λ^X_k:

$$C^{\text{PMP}}(t_k, X_k, Y_k) = (U')^{-1}(e^{\rho t_k}\lambda^X_k). \tag{20}$$

The Pontryagin-derived optimal investment proportion, π^{PMP}, is given by the more general formula (13) derived in Section 3.1:

$$\pi^{\text{PMP}}(t_k, X_k, Y_k) = \frac{\mu_k - r}{\sigma_k^2}\left(\frac{-\lambda^X_k}{X_k(\partial\lambda^X_k/\partial x)}\right) - \frac{\rho_{XY}a_k(\partial\lambda^X_k/\partial y)}{\sigma_k X_k(\partial\lambda^X_k/\partial x)}, \tag{21}$$

where $\mu_k = \mu(t_k, Y_k)$, $\sigma_k = \sigma(t_k, Y_k)$, $a_k = a(t_k, Y_k)$, and the terms λ^X_k, $\partial\lambda^X_k/\partial x$, and $\partial\lambda^X_k/\partial y$ represent estimates obtained at state (t_k, X_k, Y_k). Computing the

partial derivatives $\partial\lambda_k^X/\partial x$ and $\partial\lambda_k^X/\partial y$ typically requires applying higher-order AD to the objective J with respect to the state components X_k and Y_k, as described in Section 4.2(d).

The neural network policies $\pi_\theta(t_k, X_k, Y_k)$ and $C_\phi(t_k, X_k, Y_k)$ might deviate from these PMP-derived target controls C^{PMP} and π^{PMP}. To encourage alignment, we introduce a penalty term:

$$\mathcal{L}_{\mathrm{align}}(\theta,\phi) = \beta_C \sum_k \left| C_\phi(t_k, X_k, Y_k) - C^{\mathrm{PMP}}(t_k, X_k, Y_k) \right| + \beta_\pi \sum_k \left| \pi_\theta(t_k, X_k, Y_k) - \pi^{\mathrm{PMP}}(t_k, X_k, Y_k) \right|, \tag{22}$$

where $\beta_C, \beta_\pi > 0$ are weighting coefficients, and the sum runs over sampled nodes $(t_k, \mathbf{X}_k)$ in the mini-batch. This penalty measures the discrepancy between the learned policy and the controls suggested by the PMP necessary conditions using the current adjoint estimates.

We then define the augmented objective function:

$$J_{\mathrm{align}}(\theta,\phi) = J(\theta,\phi) - \mathcal{L}_{\mathrm{align}}(\theta,\phi).$$

Optimizing J_{align} involves updating the parameters (θ,ϕ) using its gradient:

$$\nabla_{(\theta,\phi)} J_{\mathrm{align}}(\theta,\phi) = \nabla_{(\theta,\phi)} J(\theta,\phi) - \nabla_{(\theta,\phi)} \mathcal{L}_{\mathrm{align}}(\theta,\phi).$$

This combines the original utility maximization objective with a component that actively pushes the learned policies (C_ϕ, π_θ) towards the PMP-derived targets $(C^{\mathrm{PMP}}, \pi^{\mathrm{PMP}})$.

The weights β_C and β_π control the strength of this alignment pressure. Moderate values can encourage the policy to adopt characteristics consistent with PMP optimality without overly restricting the neural network's flexibility, potentially improving convergence speed and stability. Conversely, very large values might enforce strict adherence, which could hinder exploration or slow down learning if the adjoint estimates are noisy or inaccurate, especially early in training.

This regularization strategy is conceptually similar to other methods that incorporate domain knowledge or theoretical priors into machine learning models through penalties or soft constraints. Examples include PINNs using PDE residuals [23], PDE-constrained optimization techniques employing adjoint information [8, 11], and imitation learning methods in reinforcement learning that regularize against an expert policy [26]. Our aim here is analogous: to leverage the theoretical structure provided by PMP to guide the neural network training process effectively, while retaining the benefits of data-driven policy representation and optimization.

4.4. **Extended value function and algorithmic variants .**

4.4.1. *Extended value function and discretized rollouts .* In Section 4.2, we introduced a discretization scheme for the portfolio problem over a fixed interval $[0, T]$ starting from a specific initial state. Here, we extend that approach to accommodate any initial state $(t_0, \mathbf{X}_0)$ where $\mathbf{X}_0 = (X_0, Y_0)^\top$, by defining an objective based on an extended value function concept over a broader state-time domain.

Many continuous-time control problems require finding a policy (π_θ, C_ϕ) that is valid for any admissible initial condition $(t_0, \mathbf{X}_0)$. To train such a policy, we define an objective function that averages the expected utility over a distribution $\eta(\cdot)$ of

initial nodes $(t_0, \mathbf{X}_0)$ within the relevant domain $\mathcal{D} \subset [0,T] \times (0,\infty) \times \mathbb{R}$ (or the appropriate range for Y). Concretely, we aim to maximize:

$$\widetilde{J}(\theta,\phi) = \mathbb{E}_{(t_0,\mathbf{X}_0)\sim\eta}\left[J(t_0,\mathbf{X}_0;\pi_\theta,C_\phi)\right]$$

where

$$J(t_0,\mathbf{X}_0;\pi_\theta,C_\phi) = \mathbb{E}\left[\int_{t_0}^{T} e^{-\rho u}U(C_\phi(u,X_u,Y_u))\,du + \kappa e^{-\rho T}U(X_T)\middle|\mathbf{X}_{t_0}=\mathbf{X}_0\right].$$

Maximizing $\widetilde{J}(\theta,\phi)$ yields a policy (π_θ, C_ϕ) applicable across various starting times $t_0 \in [0,T]$ and initial states $\mathbf{X}_0$.

Sampling initial states $(t_0, \mathbf{X}_0)$ from a suitable distribution η (e.g., uniform over $\mathcal{D}$) prevents overfitting to a single starting scenario and ensures the learned policy generalizes across a broad region of the state-time domain. This sampling strategy is common in reinforcement learning and numerical methods for PDEs and control problems aiming for a single policy function valid over the entire domain (cf. [10]).

The practical implementation involves sampling a mini-batch of initial states, performing a discretized simulation (rollout) from each sampled state to the terminal time T, calculating the objective for each path, and then performing backpropagation to update (θ,ϕ), as detailed in Algorithm 1.

4.4.2. *Gradient-based algorithmic variants.* Based on the extended value function objective and the PMP-guided gradient computation, we propose two main algorithmic variants:

(a) **PG-DPO (No Alignment Penalty).**
The baseline version (Algorithm 1) maximizes $\widetilde{J}(\theta,\phi)$ using standard BPTT gradients without the alignment penalty. This approach is computationally simpler but might converge slower, exhibit more variance, or find suboptimal solutions compared to the guided approach of PG-DPO-Align.

(b) **PG-DPO-Align (With Alignment Penalty).**
This enhanced version (Algorithm 2) optimizes the augmented objective

$$\widetilde{J}_{\text{align}} \coloneqq \widetilde{J} - \mathbb{E}[\mathcal{L}_{\text{align}}],$$

incorporating the alignment penalty from Section 4.3. While calculating the PMP targets requires estimates of adjoint sensitivities and their derivatives at $(t_0, \mathbf{X}_0)$. (typically obtained via higher-order AD, incurring extra computational cost), this alignment term often yields more stable and faster convergence to theoretically sound policies.

Both PG-DPO and PG-DPO-Align naturally handle the random initial nodes $(t_0, \mathbf{X}_0)$ drawn from η, enabling the learned policy (π_θ, C_ϕ) to cover the intended operational domain. The strategy of computing the alignment penalty based only on the initial state information $(t_0, \mathbf{X}_0)$ for each sampled trajectory remains efficient. Since $\mathbf{X}_0$ is sampled from a distribution covering the relevant state space, performing BPTT with respect to $\mathbf{X}_0$ (including computing second derivatives if needed for the penalty) allows us to estimate the necessary initial adjoint sensitivities $(\lambda_0^X, \partial_x\lambda_0^X, \partial_y\lambda_0^X)$. This avoids storing or computing adjoint information along the entire path for the penalty term, saving memory and simplifying implementation while still providing a rich set of alignment targets across the domain.

Algorithm 1 PG-DPO

Inputs:
Policy networks $\pi_\theta(t, X, Y)$, $C_\phi(t, X, Y)$.
Step sizes $\{\alpha_j^\theta, \alpha_j^\phi\}$, total iterations K.
Domain sampler η for $(t_0^{(i)}, X_0^{(i)}, Y_0^{(i)})$ over $\mathcal{D} \subset [0, T] \times (0, \infty) \times \mathbb{R}$.
Fixed integer N (number of time steps per path).
Market and utility parameters (r, ρ, κ, γ, functions μ, σ, b, a, correlation ρ_{XY}).

for $j = 1$ **to** K **do**

(a) Sample mini-batch of size M: For each $i \in \{1, \dots, M\}$, draw initial $(t_0^{(i)}, X_0^{(i)}, Y_0^{(i)})$ from η.

(b) Local Single-Path Simulation for each i:

(i) *Define local time steps:* Set $\Delta t^{(i)} \leftarrow (T - t_0^{(i)})/N$, and $t_k^{(i)} = t_0^{(i)} + k\,\Delta t^{(i)}$ for $k = 0, \dots, N$, with $t_N^{(i)} = T$.

(ii) *Initialize state:* $\mathbf{X}_0^{(i)} \leftarrow (X_0^{(i)}, Y_0^{(i)})^\top$.

(iii) *Euler–Maruyama Simulation:* For $k = 0, \dots, N-1$:

(iv) Generate correlated noise $(\Delta W_{X,k}^{(i)}, \Delta W_{Y,k}^{(i)})^\top \sim \mathcal{N}(\mathbf{0}, \Delta t^{(i)} \cdot \Sigma_W)$.

(v) Get controls $\pi_k^{(i)} = \pi_\theta(t_k^{(i)}, X_k^{(i)}, Y_k^{(i)})$, $C_k^{(i)} = C_\phi(t_k^{(i)}, X_k^{(i)}, Y_k^{(i)})$.

(vi) Update state $\mathbf{X}_{k+1}^{(i)} = (X_{k+1}^{(i)}, Y_{k+1}^{(i)})^\top$ using:

$$X_{k+1}^{(i)} = X_k^{(i)} + \left[r X_k^{(i)} + \pi_k^{(i)} (\mu(t_k^{(i)}, Y_k^{(i)}) - r) X_k^{(i)} - C_k^{(i)} \right] \Delta t^{(i)} + \pi_k^{(i)} \sigma(t_k^{(i)}, Y_k^{(i)}) X_k^{(i)} \Delta W_{X,k}^{(i)},$$

$$Y_{k+1}^{(i)} = Y_k^{(i)} + b(t_k^{(i)}, Y_k^{(i)}) \Delta t^{(i)} + a(t_k^{(i)}, Y_k^{(i)}) \Delta W_{Y,k}^{(i)}.$$

(vii) *Calculate local cost:*

$$J^{(i)}(\theta, \phi) = \sum_{k=0}^{N-1} e^{-\rho t_k^{(i)}} U(C_k^{(i)}) \Delta t^{(i)} + \kappa\, e^{-\rho T}\, U(X_N^{(i)}).$$

(c) Compute Average Objective and Gradients via BPTT:

$$\widetilde{J}(\theta, \phi) = \frac{1}{M} \sum_{i=1}^{M} J^{(i)}(\theta, \phi).$$

$\nabla_\theta \widetilde{J}, \nabla_\phi \widetilde{J} \leftarrow$ Compute gradients of $\widetilde{J}$ w.r.t. θ, ϕ using BPTT.

(d) Parameter Update:

$$\theta \leftarrow \theta + \alpha_j^\theta \nabla_\theta \widetilde{J}, \quad \phi \leftarrow \phi + \alpha_j^\phi \nabla_\phi \widetilde{J}.$$

end for

return Final policy (π_θ, C_ϕ).

5. **Numerical results.** This section demonstrates our Pontryagin-guided direct policy optimization (PG-DPO) framework on two benchmark problems with known analytical solutions: the classical Merton problem (with consumption and investment) and a single-factor Kim & Omberg model [14] (investment only). The Merton case allows direct comparison, while the Kim & Omberg case tests applicability to

Algorithm 2 PG-DPO-Align

Inputs:
  All inputs from Algorithm 1.
  Alignment weights (β_C, β_π).
Requires:
  Ability to compute the following derivatives via AD:

$$\lambda_0^X = \partial J/\partial X_0, \quad \partial_x \lambda_0^X = \partial^2 J/\partial X_0^2, \quad \partial_y \lambda_0^X = \partial^2 J/(\partial Y_0 \partial X_0)$$

for $j = 1$ **to** K **do**
  (a) Sample mini-batch and Simulate trajectories:
  Perform steps (a) and (b) from Algorithm 1 to obtain trajectories and costs $J^{(i)}(\theta, \phi)$ for $i = 1, \ldots, M$.
  (b) Retrieve Adjoint Information (requires extra AD pass):
  For each sample i:
  (i) Compute initial adjoint component and its derivatives using AD:

$$\lambda_0^{X,(i)} = \frac{\partial J^{(i)}}{\partial X_0^{(i)}}, \quad \partial_x \lambda_0^{X,(i)} = \frac{\partial^2 J^{(i)}}{\partial (X_0^{(i)})^2}, \quad \partial_y \lambda_0^{X,(i)} = \frac{\partial^2 J^{(i)}}{\partial Y_0^{(i)} \partial X_0^{(i)}}.$$

  (ii) Calculate PMP target controls at $(t_0^{(i)}, \mathbf{X}_0^{(i)})$ using (20) and (21):
  (iii) $C_0^{\mathrm{PMP},(i)} \leftarrow (U')^{-1}(e^{\rho t_0^{(i)}} \lambda_0^{X,(i)})$.
  (iv) $\mu_0^{(i)} \leftarrow \mu(t_0^{(i)}, Y_0^{(i)})$, $\sigma_0^{(i)} \leftarrow \sigma(t_0^{(i)}, Y_0^{(i)})$, $a_0^{(i)} \leftarrow a(t_0^{(i)}, Y_0^{(i)})$.
  (v) $\pi_0^{\mathrm{PMP},(i)} \leftarrow \frac{\mu_0^{(i)} - r}{(\sigma_0^{(i)})^2} \left(\frac{-\lambda_0^{X,(i)}}{X_0^{(i)} (\partial_x \lambda_0^{X,(i)})} \right) - \frac{\rho_{XY} a_0^{(i)} (\partial_y \lambda_0^{X,(i)})}{\sigma_0^{(i)} X_0^{(i)} (\partial_x \lambda_0^{X,(i)})}$.
  (c) Compute Alignment Loss and Augmented Objective:
  For each sample i, get policy outputs $C_0^{(i)} = C_\phi(t_0^{(i)}, X_0^{(i)}, Y_0^{(i)})$ and $\pi_0^{(i)} = \pi_\theta(t_0^{(i)}, X_0^{(i)}, Y_0^{(i)})$. Calculate alignment loss for sample i:

$$\mathcal{L}_{\mathrm{align}}^{(i)} = \beta_C |C_0^{(i)} - C_0^{\mathrm{PMP},(i)}| + \beta_\pi |\pi_0^{(i)} - \pi_0^{\mathrm{PMP},(i)}|.$$

  Compute average augmented objective:

$$\widetilde{J}_{\mathrm{align}}(\theta, \phi) = \frac{1}{M} \sum_{i=1}^{M} \left[J^{(i)}(\theta, \phi) - \mathcal{L}_{\mathrm{align}}^{(i)} \right].$$

  (d) Compute Gradients and Update Parameters:
  $\nabla_\theta \widetilde{J}_{\mathrm{align}}, \nabla_\phi \widetilde{J}_{\mathrm{align}} \leftarrow$ Compute gradients of $\widetilde{J}_{\mathrm{align}}$ w.r.t. θ, ϕ using BPTT.

$$\theta \leftarrow \theta + \alpha_j^\theta \nabla_\theta \widetilde{J}_{\mathrm{align}}, \quad \phi \leftarrow \phi + \alpha_j^\phi \nabla_\phi \widetilde{J}_{\mathrm{align}}.$$

end for
return Final policy (π_θ, C_ϕ).

state-dependent dynamics. We also present value function approximations for the Merton problem, estimated via Monte Carlo simulations using the learned policy and fitted with a separate network (V_ψ), clarifying this two-stage approach.

5.1. **Merton problem with consumption and investment.** We first apply our method to the standard Merton portfolio problem [18] incorporating both consumption and investment decisions. This corresponds to a special case of the general

formulation presented in Section 2 with $d = 1$ risky asset, constant market coefficients ($\mu(t, Y_t) = \mu$, $\sigma(t, Y_t) = \sigma$), and no external stochastic factor (Y_t). The investor aims to maximize $J(\pi, C)$ in (4) using a CRRA utility $U(x) = x^{1-\gamma}/(1-\gamma)$ with $\kappa = 0.1$. This problem admits a closed-form solution, providing a benchmark. Although our framework accommodates time-varying coefficients, we use constant parameters for direct comparison.

We consider a one-year horizon $T = 1$ and restrict the wealth (state) domain to $[0.1, 2]$. The model parameters are $r = 0.07$, $\mu = 0.18$, $\sigma = 0.3$, $\gamma = 2$, and $\rho = 0.1$ (discount rate). Two neural networks, C_ϕ and π_θ, approximate the consumption rate and portfolio proportion, respectively. Both networks take the state (t, X_t) as input and consist of two hidden layers with 200 nodes each and Leaky-ReLU activation. Training utilizes mini-batches of 10,000 initial states (t_0, X_0) sampled uniformly from the domain $\mathcal{D} = [0, T] \times [0.1, 2]$. At each iteration, fresh trajectories are simulated using the Euler-Maruyama scheme (with log-Euler for wealth to ensure positivity) starting from these sampled initial states, mitigating overfitting.

We compare two variants of our approach:

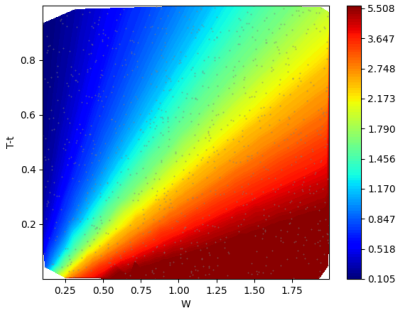
(a) **PG-DPO**: The baseline algorithm (Algorithm 1) maximizing $\widetilde{J}(\theta, \phi)$ without an explicit alignment penalty.
(b) **PG-DPO-Align**: The enhanced version (Algorithm 2) incorporating the adjoint-based alignment penalty $\mathcal{L}_{\text{align}}$ (cf. Section 4.3) with weights $(\beta_C, \beta_\pi) = (10^{-3}, 10^{-1})$, controlling the penalty strength, chosen after preliminary tuning. For the Merton problem, the PMP targets in $\mathcal{L}_{\text{align}}$ are given by

$$C_0^{\text{PMP}} = (e^{\rho t_0} \lambda_0^X)^{-1/\gamma}, \quad \pi_0^{\text{PMP}} = -\frac{\mu - r}{\sigma^2 X_0} \frac{\lambda_0^X}{\partial \lambda_0^X / \partial x}.$$

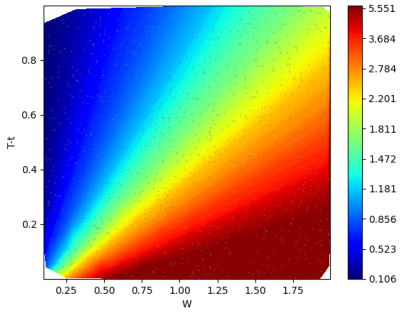
Both variants use the Adam optimizer with learning rates 1×10^{-5} (C_ϕ) and 1×10^{-3} (π_θ) for up to 100,000 iterations. Performance is evaluated using relative MSEs against the known Merton solution, empirical utility (averaged over 500 validation rollouts), and elapsed training time.

Table 1 summarizes the performance metrics. For the policy networks, PG-DPO-Align generally achieves lower relative MSEs compared to the baseline PG-DPO, particularly at higher iteration counts (10k and 100k), indicating faster or more accurate convergence towards the optimal controls. Both methods demonstrate good accuracy in approximating the value function itself and rapidly converge to high levels of empirical utility. Notably, significant accuracy (low MSEs and near-maximal utility) is achieved by just 10,000 iterations, requiring less than 10 minutes of computation for both methods on our hardware (8.92m for PG-DPO, 9.60m for PG-DPO-Align). The additional computational cost for the alignment penalty in PG-DPO-Align is modest (7-8% increase in total training time). These findings suggest that near-optimal policies can be learned efficiently, and the benefits of alignment (accuracy, stability) come at a small overhead.

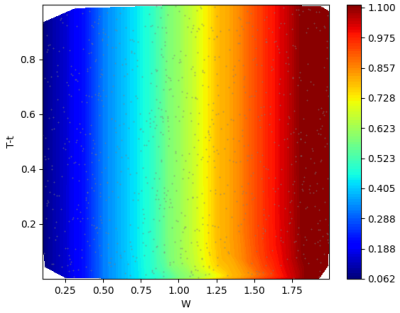
Figure 1 visually confirms the close match between the learned results (using PG-DPO-Align after 100k iterations) and the exact Merton solutions for consumption, investment, and the value function. This demonstrates the framework's ability to accurately capture optimal controls and the associated expected utility, despite the challenges of simultaneous optimization.



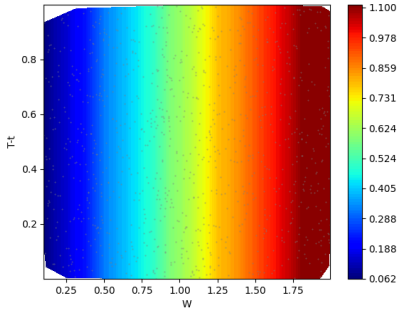
(A) Learned consumption policy



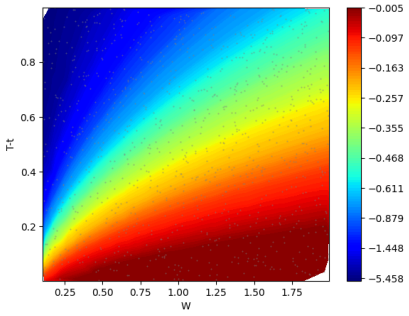
(B) Exact consumption formula



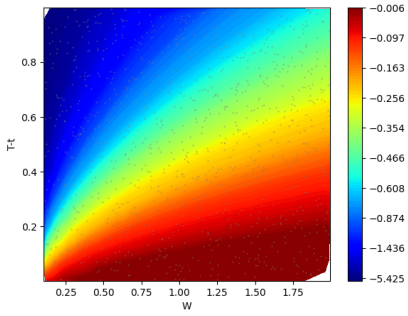
(C) Learned investment policy



(D) Exact investment formula



(E) Learned value function



(F) Exact value function

FIGURE 1. Neural vs. exact solutions for consumption policy, investment policy, and value function under the Merton model. The learned plots (left column) are from our PG-DPO-Align run at iteration 100,000; the exact plots (right column) show the corresponding closed-form Merton solutions.

TABLE 1. PG-DPO vs. PG-DPO-Align in the Merton problem. We compare relative MSEs (consumption/investment/value function), empirical utility, and Elapsed Time at various iteration milestones. Learning rates are 1×10^{-3} (π_θ) and 1×10^{-5} (C_ϕ). PG-DPO-Align uses alignment weights $(\beta_C, \beta_\pi) = (10^{-3}, 10^{-1})$.

Iterations		1,000	10,000	100,000
Rel. MSE (Consumption)	PG-DPO	3.14e+00	5.73e-01	9.65e-02
	PG-DPO-Align	3.00e+00	3.75e-01	3.46e-02
Rel. MSE (Investment)	PG-DPO	2.39e-02	2.13e-02	1.19e-02
	PG-DPO-Align	7.22e-02	1.57e-02	8.43e-03
Rel. MSE (Value Func.)	PG-DPO	4.33e-02	1.49e-04	1.21e-04
	PG-DPO-Align	3.37e-02	1.04e-04	2.18e-04
Empirical Utility	PG-DPO	6.5420e-01	6.4795e-01	6.4791e-01
	PG-DPO-Align	6.5434e-01	6.4793e-01	6.4791e-01
Elapsed Time	PG-DPO	1.33m	8.92m	53.45m
	PG-DPO-Align	1.43m	9.60m	57.55m

While our methods demonstrate computational efficiency (see Table 1), careful consideration is required when comparing runtime with existing literature. For example, Nguwi et al. [20] report runtimes of approximately 1 hour for Deep Branching and 3 hours for Deep BSDE [10] in solving the 1-dimensional Merton problem (cf. Table 5 in [20]), whereas our PG-DPO methods achieve high accuracy in under 10 minutes. However, direct comparison remains challenging due to differences in hardware environments, implementation details, chosen model parameters (e.g., variations in $\mu, \sigma, \gamma, \rho, T$), and the problem formulations themselves (e.g., Nguwi et al. learn the value $V(0.1, x)$ at a fixed initial time $t = 0.1$, while we learn policies across the entire (t, X) domain and subsequently approximate the value function $V(t, X)$ in a separate step). Despite these caveats, the significant gap in observed computational times indicates a potential advantage of our direct policy optimization framework. Our primary goal remains the efficient learning of optimal controls (π_θ, C_ϕ), with the approximation of the value function mainly used for validation.

5.2. **Single-factor kim & omberg model (investment only).** To evaluate our framework in a setting with stochastic investment opportunities, we consider a model inspired by Kim and Omberg [14]. In this model, the risky asset price S_t and a stochastic factor Y_t follow the SDEs:

$$\frac{dS_t}{S_t} = (r + \sigma\alpha Y_t)dt + \sigma dW_{X,t}, \tag{23}$$

$$dY_t = \kappa_Y(\theta_Y - Y_t)dt + \sigma_Y dW_{Y,t}, \tag{24}$$

where Y_t is an Ornstein-Uhlenbeck (OU) process representing the stochastic investment opportunities. Here, r is the risk-free rate, σ is the constant volatility of the

risky asset, α is the factor loading, and $\kappa_Y, \theta_Y, \sigma_Y$ are the mean-reversion speed, long-term mean, and volatility of the factor Y_t, respectively. The Brownian motions $W_{X,t}$ and $W_{Y,t}$ have an instantaneous correlation ρ_{XY}.

This setup corresponds to the general framework of Section 2 with $\mu(t, Y_t) = r + \sigma\alpha Y_t$, $\sigma(t, Y_t) = \sigma$, $b(t, Y_t) = \kappa_Y(\theta_Y - Y_t)$, and $a(t, Y_t) = \sigma_Y$. Similar to the Merton problem, we focus on maximizing the expected utility of terminal wealth, corresponding to the general objective (4) with $C_t \equiv 0$ and $\kappa = 1$, i.e., $J(\pi, C = 0) = \mathbb{E}[e^{-\rho T} U(X_T)]$ using the CRRA utility $U(x) = x^{1-\gamma}/(1-\gamma)$. This model, while representing a more complex, dynamic environment, admits a known closed-form solution for the optimal investment policy $\pi^*(t, Y_t)$, serving as our benchmark.

For the numerical tests presented here, we set the horizon $T = 3$ years and use a risk aversion coefficient $\gamma = 2$. The specific parameters used are $r = 0.05$, $\kappa_Y = 0.1$, $\theta_Y = 0.02$, $\sigma_Y = 1.0$, $\sigma = 0.3$, $\alpha = 1.0$, and $\rho_{XY} = 0.8$. The state domain for sampling initial conditions covers the time horizon $[0, T]$ and relevant ranges for wealth X_t and the factor Y_t. We sample X_t uniformly from $[0.1, 3]$ and Y_t uniformly from $[-0.48, 0.52]$ (corresponding to $\theta_Y \pm 0.5\sigma_Y$).

The investment policy $\pi_\theta(t, X_t, Y_t)$ is parameterized by a neural network. This network takes the state vector (t, X_t, Y_t) as input, incorporating time-to-maturity $T - t$, wealth, and the stochastic factor. It consists of three hidden layers, each containing 200 nodes and utilizing the Leaky-ReLU activation function. The final output layer produces the scalar investment proportion.

Similar to the Merton experiment, we compare PG-DPO (Algorithm 1) and PG-DPO-Align (Algorithm 2), both adapted here to maximize the expected utility of terminal wealth only ($C_t \equiv 0, \kappa = 1$). For PG-DPO-Align in this K&O setting, the alignment penalty $\mathcal{L}_{\text{align}}$ only targets the investment policy π_θ. The target π_0^{PMP}, calculated at the initial state (t_0, X_0, Y_0), uses the general form from Eq. (21) (or (29)):

$$\pi_0^{\text{PMP}} = -\frac{\mu_0 - r}{\sigma^2} \frac{\lambda_0^X}{X_0(\partial\lambda_0^X/\partial x)} - \frac{\rho_{XY}\sigma_Y(\partial\lambda_0^X/\partial y)}{\sigma X_0(\partial\lambda_0^X/\partial x)},$$

where $\mu_0 = r + \sigma\alpha Y_0$. Unlike the Merton case, the hedging term persists due to the factor Y_t. Calculating this target requires estimates of the initial adjoint state λ_0^X and its derivatives $\partial\lambda_0^X/\partial x = \partial^2 J/\partial X_0^2$ and $\partial\lambda_0^X/\partial y = \partial^2 J/(\partial Y_0 \partial X_0)$ via AD. The alignment weight is set to $\beta_\pi = 10^{-4}$ after tuning.

Key differences in training compared to the Merton experiment include: the Adam optimizer uses a lower learning rate of 1×10^{-7} for π_θ; mini-batches contain $M = 1000$ samples; each trajectory is simulated for $N = 20$ time steps using Euler-Maruyama (adapted for OU and log-wealth dynamics) with antithetic variates employed for variance reduction. We train for up to 50,000 iterations to keep the total runtime under approximately one hour. Performance is primarily measured by the relative MSE between the learned policy π_θ and the closed-form solution π^*, although empirical utility is also reported in Table 2.

The numerical results for the Kim & Omberg model are presented in Table 2. Both PG-DPO variants successfully learn effective policies, achieving similar high levels of empirical utility that stabilize as training progresses. Notably, PG-DPO-Align consistently yields a lower relative MSE for the investment policy π_θ compared to the baseline PG-DPO, indicating a closer match to the optimal solution π^*. This advantage in accuracy is particularly evident around the 10,000 iteration mark and is maintained throughout training. The improved accuracy from the alignment penalty comes at a modest increase in computation time (approximately 10% longer

TABLE 2. Performance of PG-DPO vs. PG-DPO-Align on the Kim & Omberg investment-only problem (one risky asset and one stochastic factor). Relative MSE for the investment policy π_θ and Empirical Utility, and Elapsed Time are compared across different iteration counts.

Iterations		1,000	10,000	50,000
Rel. MSE (Investment)	PG-DPO	3.59e-01	3.50e-01	1.84e-01
	PG-DPO-Align	3.58e-01	3.38e-01	1.09e-01
Empirical Utility	PG-DPO	1.0888e+00	1.0605e+00	9.3780e-01
	PG-DPO-Align	1.0875e+00	1.0477e+00	9.0350e-01
Elapsed Time	PG-DPO	2.25m	11.20m	57.98m
	PG-DPO-Align	2.75m	12.33m	59.37m

runtime for Align), which aligns with the expected overhead from calculating the penalty term involving hedging demands.

Overall, these results demonstrate that the PG-DPO framework, especially when enhanced with the Pontryagin-guided alignment penalty, can successfully learn near-optimal investment strategies in environments with stochastic investment opportunities driven by factors like Y_t, closely replicating the known theoretical solution structure without relying on external comparisons for this specific setup.

Extension to non-markovian dynamics with markovization

The preceding sections have demonstrated PG-DPO's efficacy in Markovian setups (e.g., the Kim & Omberg model). However, many empirically relevant phenomena, such as price momentum, are inherently path-dependent and thus non-Markovian in nature. For example, Liu (2022) recently showed that optimal dynamic momentum strategies explicitly depend on historical asset price paths, highlighting the practical relevance and importance of addressing path-dependent dynamics in continuous-time portfolio optimization. When a path-dependent signal admits a finite-dimensional Markovian embedding (so-called Markovization), PG-DPO-Align applies without structural changes: we only augment the state vector and include additional adjoint sensitivities in the alignment penalty. Below, we illustrate this with two common momentum filters, exponentially weighted moving-average (EWMA) and simple moving-average (SMA), both of which can be cast into finite-dimensional Markovian form.

5.3. **Exponentially weighted moving-average momentum.** Consider the augmented state $\widetilde{\mathbf{X}}_t = (X_t, M_t)$, where X_t is the wealth and M_t represents the EWMA of past asset returns. For a fixed decay rate $\beta > 0$, M_t is defined by

$$M_t = \beta \int_{-\infty}^{t} e^{-\beta(t-s)} \frac{dS_s}{S_s}, \qquad \text{which implies} \quad dM_t = -\beta M_t\, dt + \beta \frac{dS_t}{S_t}.$$

If the asset price S_t follows $dS_t/S_t = \mu_t dt + \sigma dW_t$ with constant volatility $\sigma > 0$, then M_t is an OU-type process with diffusion coefficient $\beta\sigma$, driven by the same

Brownian motion W_t. Consequently, the pair (X_t, M_t) forms a two-dimensional Markovian state vector.

Introducing the costate pair $\boldsymbol{\lambda}_t = (\lambda_t^X, \lambda_t^M)^\top$, the Pontryagin FOC mirror those in Appendix A. In particular, adapting equation (29) by identifying $Y_t \equiv M_t$, $\mu(t, Y_t) = f(M_t)$, $a(t, Y_t) = \beta\sigma$, $\sigma(t, Y_t) = \sigma$, and $\rho_{XY} = 1$ (since M_t and S_t share W_t), we obtain the optimal investment proportion:

$$\pi_t = \frac{f(M_t) - r}{\sigma^2} \left(\frac{-\lambda_t^X}{X_t \left(\partial \lambda_t^X / \partial x\right)} \right) - \frac{\beta \left(\partial \lambda_t^X / \partial m\right)}{X_t \left(\partial \lambda_t^X / \partial x\right)}. \tag{25}$$

This expression decomposes into a myopic demand (first term) and an intertemporal hedging demand against shifts in M_t (second term).

The policy network π_θ and C_ϕ now receive the augmented state (t, X_t, M_t) as input. AD during the BPTT step provides the sensitivities λ_t^X, $\partial \lambda_t^X / \partial x$, and $\partial \lambda_t^X / \partial m$. These are then substituted into (25) and the consumption FOC $U'(C_t^*) = e^{\rho t} \lambda_t^X$ to compute the Pontryagin targets π^{PMP} and C^{PMP} for the alignment loss $\mathcal{L}_{\mathrm{align}}$. Other than the expanded input dimension for the policy networks and the corresponding derivative calculations for the alignment targets, the core PG-DPO optimization loop and network architecture remain unchanged.

5.4. **Fixed-window simple moving-average momentum.** A K-period SMA of returns, while inherently path-dependent, can be represented as a Markovian system through a lag-stack embedding. Fix a time step $\Delta = T_{\mathrm{lag}}/K$, where T_{lag} is the look-back period. The lagged returns are defined by

$$dR_t^{(0)} = \frac{dS_t}{S_t}, \qquad dR_t^{(j)} = \frac{R_t^{(j-1)} - R_t^{(j)}}{\Delta}\, dt, \quad 1 \le j < K,$$

where $R_t^{(0)}$ denotes the most recent instantaneous return, so that $R_t^{(j)}$ approximates the return from roughly $j\Delta$ time units ago. The SMA factor is then

$$M_t^{\mathrm{SMA}} = \frac{1}{K} \sum_{j=0}^{K-1} R_t^{(j)}.$$

In this construction, only the lead component $R_t^{(0)}$ directly shares the Brownian motion W_t with the asset S_t. Hence, the augmented state is $\widetilde{\mathbf{X}}_t = \left(X_t,\, R_t^{(0)},\, R_t^{(1)},\, \ldots,\, R_t^{(K-1)}\right)$. Following a similar PMP application as in the EWMA case (Section 5.3), where the asset's expected return μ_t is now a function of the SMA factor, $f(M_t^{\mathrm{SMA}})$, the optimal investment proportion is:

$$\pi_t = \frac{f(M_t^{\mathrm{SMA}}) - r}{\sigma^2} \left(\frac{-\lambda_t^X}{X_t \left(\partial \lambda_t^X / \partial x\right)} \right) - \frac{\partial \lambda_t^X / \partial r^{(0)}}{X_t \left(\partial \lambda_t^X / \partial x\right)}. \tag{26}$$

The structure of (26) shows a single intertemporal hedging term, associated with $R_t^{(0)}$. This arises because $R_t^{(0)}$ is the only component among the $R_t^{(j)}$ whose dynamics are directly driven by dW_t, the remaining lags $R_t^{(j)}$ $(j \ge 1)$ influence only the drift of the system.

Remark 5.1 (Hedging Structure with a Single Noise Source)**.** In this Markovian embedding driven by a single underlying Brownian motion W_t for the asset price, the optimal policy features one hedging component. This is linked to the factor dynamics $(R_t^{(0)})$ that share this common noise source, irrespective of the augmented

state's overall dimension $(K + 2)$. Additional independent noise sources would typically introduce further hedging terms.

The PG-DPO-Align implementation is analogous to the EWMA case: the networks now receive the lag-stack state $(t, X_t, R_t^{(0)}, \ldots, R_t^{(K-1)})$, and the same AD step supplies the required derivatives for (26) and the consumption FOC.

The EWMA and SMA examples illustrate that PG-DPO can accommodate path-dependent momentum signals by casting them into finite-dimensional Markovian representations. A key observation is that the optimal policy in these extended settings retains the fundamental myopic-plus-hedging structure derived from the general PMP formulation in Appendix A. The specific form and number of hedging terms are determined by the way factors share common noise sources with the primary asset(s). Investigating scenarios with multiple independent Brownian drivers, which would naturally lead to richer hedging structures, remains an interesting direction for future research.

6. **Conclusion.**

In this paper, we introduced PG-DPO, a framework that synergizes PMP from classical control theory with modern neural network-based policy optimization. Our approach directly learns optimal policies for continuous-time portfolio problems by utilizing suboptimal adjoint processes, computed efficiently by leveraging AD, thereby offering an alternative to methods centered on value function approximation. The framework is designed to generalize effectively across the state-time domain by sampling initial states and employing mini-batch simulations during training.

We further proposed an enhanced variant, PG-DPO-Align, which incorporates an adjoint-based alignment penalty. This penalty leverages PMP's necessary conditions to guide the learning process, effectively regularizing the policy towards theoretically grounded PMP controls computed via runtime adjoint estimates. Our numerical experiments clearly validated the efficacy and robustness of the proposed framework. First, it successfully solved the classical Merton benchmark involving simultaneous optimization of consumption and investment, accurately recovering the known closed-form solution. Second, extending beyond constant-coefficient settings, the framework proved effective on the Kim & Omberg model, successfully learning the optimal investment strategy, including intertemporal hedging demands, in an environment with stochastic investment opportunities driven by an external factor (Y_t). The alignment penalty consistently enhanced training stability and final policy accuracy across these problems, with runtime increases typically under 10% relative to the unregularized PG-DPO method, highlighting an acceptable computational trade-off. Our experiments also suggest promising computational efficiency relative to existing value-based deep learning methods such as Deep BSDE and Deep Branching methods, though detailed benchmarking under identical conditions remains to be performed.

Moreover, we analytically established a clear methodology for extending PG-DPO beyond purely Markovian dynamics. By explicitly augmenting the state vector and appropriately adapting the PMP-based adjoint framework, PG-DPO naturally incorporates important path-dependent phenomena such as EWMA and SMA momentum factors, preserving the myopic-plus-hedging decomposition structure of optimal policies. This established capability highlights the adaptability and broad applicability of our PMP-guided approach.

While demonstrating effectiveness on these important benchmarks, this research also highlights areas for future exploration. Further theoretical work remains necessary to establish convergence guarantees for the discrete-time PG-DPO algorithm, particularly clarifying its consistency with continuous-time optimality conditions derived from PMP. Such analysis would provide formal backing for the empirical convergence and stability observed in our numerical results. Extending the PG-DPO methodology presents several compelling avenues. Particularly promising directions include: (i) scaling the framework to high-dimensional asset universes with multiple stochastic factors; (ii) incorporating realistic trading constraints such as short-selling bans or borrowing limits; and (iii) conducting comprehensive empirical validation of PG-DPO's performance in broader classes of non-Markovian settings, including long-memory momentum dynamics and multi-factor path-dependent structures. Although Section 5.2 already provides a foundation for such non-Markovian extensions through state augmentation (Markovization), future research will further investigate genuinely non-Markovian scenarios which may require alternative approximation techniques beyond direct Markovization.

In conclusion, by effectively integrating the adjoint perspective of PMP within contemporary deep learning paradigms, PG-DPO offers a powerful and potentially efficient direct policy optimization approach for complex continuous-time stochastic control problems. Ultimately, our work demonstrates that bridging classical control-theoretic insights with deep learning offers a flexible and promising approach for addressing sophisticated decision-making problems in finance and beyond.

Appendix A. **Derivation of optimal portfolio decomposition via PMP.** This appendix provides a derivation demonstrating how the optimal investment proportion π_t^* obtained from the PMP necessary conditions (Section 3.1) can be explicitly decomposed into myopic and intertemporal hedging demand components. This derivation utilizes the relationship between the adjoint diffusion process $\mathbf{Z}_t^*$ and the gradients of the costate vector $\boldsymbol{\lambda}_t^*$, obtained via Itô's Lemma.

We start from the FOC with respect to the investment proportion π_t, derived from maximizing the Hamiltonian (9) and evaluated at the optimal processes $(\mathbf{X}_t^*, \mathbf{u}_t^*, \boldsymbol{\lambda}_t^*, \mathbf{Z}_t^*)$:

$$\left.\frac{\partial \mathcal{H}}{\partial \pi_t}\right|_* = \lambda_t^{X,*}(\mu(t, Y_t^*) - r)X_t^* + Z_{X,t}^{X,*}\sigma(t, Y_t^*)X_t^* = 0. \qquad \text{(12 revisited)}$$

This equation relates the optimal adjoint component $Z_{X,t}^{X,*}$ (sensitivity of $d\lambda_t^{X,*}$ to the asset noise $dW_{X,t}$) to the optimal costate $\lambda_t^{X,*}$:

$$Z_{X,t}^{X,*} = -\lambda_t^{X,*}\frac{\mu(t, Y_t^*) - r}{\sigma(t, Y_t^*)}. \tag{27}$$

Separately, we can derive an expression for $Z_{X,t}^{X,*}$ by applying Itô's Lemma to the costate $\lambda_t^{X,*}$. Assuming sufficient smoothness and identifying λ_t^X with the partial derivative of the value function $V(t, X_t, Y_t)$ with respect to wealth, $\lambda_t^X = V_x(t, X_t, Y_t)$, Itô's lemma gives:

$$\begin{aligned} d\lambda_t^X = {} & (\dots)\, dt + V_{xx}dX_t + V_{xy}dY_t \\ & + \frac{1}{2}V_{xxx}(dX_t)^2 + \frac{1}{2}V_{xyy}(dY_t)^2 + V_{xxy}(dX_t)(dY_t). \end{aligned}$$

The diffusion part of $d\lambda_t^X$ (terms involving $dW_{X,t}$ and $dW_{Y,t}$) arises from the $V_{xx}dX_t$ and $V_{xy}dY_t$ terms. Substituting the SDEs (3) and (2), and decomposing $dW_{Y,t} = \rho_{XY}dW_{X,t} + \sqrt{1-\rho_{XY}^2}dW_{Z,t}$, where W_Z is independent of W_X.

This relationship, connecting the diffusion term of the adjoint process to the derivatives of the value function, is a standard result derived from Itô's lemma in the context of BSDEs and stochastic control theory (see e.g., [21, 30]):

$$\text{coeff of } dW_{X,t} \text{ in } d\lambda_t^X = V_{xx}(\pi_t \sigma X_t) + V_{xy}(a\rho_{XY}).$$

By definition, this coefficient is $Z_{X,t}^X$. Replacing the value function derivatives with costate derivatives ($V_{xx} = \partial_x \lambda_t^X$, $V_{xy} = \partial_y \lambda_t^X$) and evaluating at the optimum yields:

$$Z_{X,t}^{X,*} = \frac{\partial \lambda_t^{X,*}}{\partial x}(\pi_t^* \sigma(t, Y_t^*) X_t^*) + \frac{\partial \lambda_t^{X,*}}{\partial y}(a(t, Y_t^*)\rho_{XY}). \tag{28}$$

Now, equating the two expressions for $Z_{X,t}^{X,*}$ from (27) and (28):

$$-\lambda_t^{X,*}\frac{\mu(t, Y_t^*) - r}{\sigma(t, Y_t^*)} = \frac{\partial \lambda_t^{X,*}}{\partial x}(\pi_t^* \sigma X_t^*) + \frac{\partial \lambda_t^{X,*}}{\partial y}(a\rho_{XY}).$$

We solve for the optimal investment proportion π_t^*:

$$\frac{\partial \lambda_t^{X,*}}{\partial x}\pi_t^* \sigma X_t^* = -\lambda_t^{X,*}\frac{\mu - r}{\sigma} - \frac{\partial \lambda_t^{X,*}}{\partial y}a\rho_{XY}.$$

Assuming $\partial \lambda_t^{X,*}/\partial x \neq 0$ (i.e., $V_{xx} \neq 0$), we divide by $(\partial \lambda_t^{X,*}/\partial x)\sigma X_t^*$:

$$\pi_t^* = \frac{\mu(t, Y_t^*) - r}{\sigma(t, Y_t^*)^2}\left(\frac{-\lambda_t^{X,*}}{X_t^*(\partial \lambda_t^{X,*}/\partial x)}\right) - \frac{\rho_{XY}a(t, Y_t^*)(\partial \lambda_t^{X,*}/\partial y)}{\sigma(t, Y_t^*)X_t^*(\partial \lambda_t^{X,*}/\partial x)}. \tag{29}$$

This derivation rigorously shows the explicit decomposition of the optimal investment proportion π_t^* within the PMP framework. The first term represents the myopic demand, driven by the instantaneous risk premium adjusted for risk aversion (captured by the ratio involving λ^X and its derivative). The second term represents the intertemporal hedging demand, accounting for the desire to hedge against unfavorable movements in the stochastic factor Y_t, incorporating its volatility (a), its correlation with the asset (ρ_{XY}), and the cross-sensitivity of the marginal utility of wealth to the factor ($\partial \lambda^X/\partial y$). This decomposition aligns perfectly with the structure derived from the HJB approach (8), confirming the consistency between the two fundamental methodologies of optimal control.

Appendix B. **Stochastic approximation and convergence analysis.** In this Appendix, we provide a convergence result for our stochastic approximation scheme, building on classical Robbins–Monro theory. We first analyze the baseline objective and then extend the result to handle the augmented objective that includes the alignment penalty. The core results rely on standard assumptions, including Lipschitz continuity of the gradients, which holds rigorously for utility functions like CARA (Constant Absolute Risk Aversion). We discuss the implications for CRRA (Constant Relative Risk Aversion) utility, used in our main experiments, where global Lipschitz continuity does not hold.

B.1. **Convergence analysis for the baseline objective $J(\theta, \phi)$.** We begin with the baseline problem of maximizing J in (4), viewed as a function of the parameters $J(\theta, \phi) := J(\pi_\theta, C_\phi)$, where (θ, ϕ) parametrize the policy networks π_θ and C_ϕ. In practice, as discussed in Section 4.2, the continuous-time Merton objective is discretized by a time-step $\Delta t = T/N$ and approximated by a mini-batch or single-path sampling scheme, thus yielding gradient estimates $\hat{g}_k(\theta, \phi)$. Strictly speaking, when $\Delta t > 0$ is finite, a small residual discretization bias may remain. However, as $\Delta t \to 0$, this bias diminishes, and under a well-designed mini-batch sampling, $\hat{g}_k$ is regarded as an unbiased (or nearly unbiased) estimator of ∇J.
Key Assumptions (cf. [16, 3]).

(A1) (Lipschitz Gradients) The mapping $\nabla J(\theta, \phi)$ is globally Lipschitz on a compact domain $\Theta \times \Phi$. That is, there exists some constant $L_J > 0$ such that

$$\|\nabla J(\theta, \phi) - \nabla J(\theta', \phi')\| \le L_J \|(\theta, \phi) - (\theta', \phi')\|$$

for all (θ, ϕ) and (θ', ϕ') in $\Theta \times \Phi$.
Note: This assumption holds rigorously for problems involving utility functions with bounded derivatives, such as the CARA utility $U(x) = -e^{-ax}$. However, the CRRA utility function $U(x) = x^{1-\gamma}/(1-\gamma)$ used in our main experiments does not have a globally Lipschitz gradient, particularly near $x = 0$. The following convergence proof applies directly to the CARA case or other settings where (A1) is satisfied.

(A2) (Unbiased, Bounded-Variance Gradients) For each iteration k, the gradient estimate $\hat{g}_k(\theta, \phi)$ satisfies

$$\mathbb{E}\big[\hat{g}_k(\theta, \phi)\big] = \nabla J(\theta, \phi) + \delta_k, \quad \mathbb{E}\big[\|\hat{g}_k(\theta, \phi)\|^2\big] \le B,$$

where $\|\delta_k\| \to 0$ as $\Delta t \to 0$ (eliminating discretization bias). This vanishing bias relies on the convergence properties of the Euler-Maruyama scheme used for the FSDE discretization under standard regularity conditions for the SDE coefficients [22, 15]. In typical mini-batch SGD, $\hat{g}_k$ is unbiased at each iteration (assuming i.i.d. sampling from the underlying distribution), and the variance is uniformly bounded by $B > 0$.

(A3) (Robbins–Monro Step Sizes) The step sizes $\{\alpha_k\}$ satisfy

$$\sum_{k=0}^{\infty} \alpha_k = \infty, \qquad \sum_{k=0}^{\infty} \alpha_k^2 < \infty.$$

This ensures a standard Robbins–Monro (stochastic gradient) iteration.

Under these assumptions, our parameter update is

$$(\theta_{k+1}, \phi_{k+1}) = (\theta_k, \phi_k) + \alpha_k \hat{g}_k(\theta_k, \phi_k),$$

where the term δ_k (if any) shrinks as $\Delta t \to 0$.

Theorem B.1 (Baseline Robbins–Monro Convergence for Lipschitz Case)**.** *Suppose Assumptions **(A1)**–**(A3)** hold (e.g., for CARA utility), and that $\|\delta_k\| \to 0$ as $\Delta t \to 0$. Then, with probability one,*

$$(\theta_k, \phi_k) \xrightarrow[k\to\infty]{a.s.} (\theta^\dagger, \phi^\dagger),$$

where $\nabla J(\theta^\dagger, \phi^\dagger) = 0$. In other words, $(\theta^\dagger, \phi^\dagger)$ is a stationary point of J in the parameter space.

Proof. As $\Delta t \to 0$, each $\hat{g}_k(\theta, \phi)$ becomes (approximately) unbiased with bounded variance. Under the Lipschitz condition (A1) and the step size conditions (A3), the classical Robbins–Monro argument [16, 3] guarantees that the iterates (θ_k, ϕ_k) converge almost surely to the set of points where $\nabla J = 0$. (Assuming uniqueness of the stationary point within the convergence region, it converges to that point). □

This result ensures that, under the specified conditions (including Lipschitz gradients, valid for CARA utility), our parameter sequence converges to a stationary point in the parameter space (θ, ϕ).

Discussion on CRRA Utility: While the global Lipschitz condition (A1) does not hold for the CRRA utility function used in our main experiments, we empirically observe robust convergence to policies closely aligned with the known global optimum. This might be attributed to the algorithm operating within regions where the objective function exhibits locally Lipschitz properties, or the inherent robustness of stochastic gradient methods in deep learning contexts [7]. A convergence analysis for the CRRA case under potentially weaker assumptions (e.g., local Lipschitzness, specific properties of the neural network optimizer) remains an important direction for future research. The non-convex nature of the neural network objective surface [5] also implies convergence is generally to a local optimum or stationary point, although in the Merton problem with concave utility, this often corresponds to the global optimum in practice.

B.2. **Convergence analysis for the augmented objective $J_{\text{align}}(\theta, \phi)$.** We now extend the convergence analysis to the augmented objective J_{align} which includes the adjoint-based regularization term $\mathcal{L}_{\text{align}}$ (see Section 4.3). Recall

$$J_{\text{align}}(\theta, \phi) = J(\theta, \phi) - \mathcal{L}_{\text{align}}(\theta, \phi),$$

where $\mathcal{L}_{\text{align}}$ penalizes deviations from Pontryagin-derived controls $\left(\pi^{\text{PMP}}, C^{\text{PMP}}\right)$. The gradient is given by

$$\nabla_{(\theta,\phi)} J_{\text{align}}(\theta, \phi) = \nabla_{(\theta,\phi)} J(\theta, \phi) - \nabla_{(\theta,\phi)} \mathcal{L}_{\text{align}}(\theta, \phi). \tag{30}$$

Key Assumptions for the Augmented Objective. Let $\hat{g}_k(\theta, \phi)$ be the estimator for $\nabla J(\theta, \phi)$ as in (A2), and let $\hat{h}_k(\theta, \phi)$ be an (approximately) unbiased, bounded-variance estimator of $\nabla \mathcal{L}_{\text{align}}(\theta, \phi)$.

(B1) (Lipschitz Gradients for J_{align}) Suppose $\nabla \mathcal{L}_{\text{align}}$ is also globally Lipschitz on $\Theta \times \Phi$. If the policy networks and the functions defining the PMP targets $\pi^{\text{PMP}}, C^{\text{PMP}}$ (which depend on $\lambda_k, \partial_x \lambda_k$) result in a Lipschitz gradient for $\mathcal{L}_{\text{align}}$, and assuming (A1) holds, then ∇J_{align} is globally Lipschitz. Similar to (A1), this assumption needs careful verification depending on the utility function; it holds more directly for CARA than for CRRA.

(B2) (Approximately Unbiased, Bounded-Variance Gradients for J_{align}) We define the gradient estimate for J_{align} as

$$\hat{f}_k(\theta, \phi) = \hat{g}_k(\theta, \phi) - \hat{h}_k(\theta, \phi),$$

mirroring (30). Assuming $\hat{h}_k$ is also approximately unbiased with bounded variance, and $\|\delta_k\| \to 0$ for bias terms as $\Delta t \to 0$, then $\hat{f}_k$ is an approximately unbiased estimator of ∇J_{align} with uniformly bounded variance (by some constant $B' > 0$).

(B3) (Robbins–Monro Step Sizes) The step sizes $\{\alpha_k\}$ satisfy

$$\sum_{k=0}^{\infty} \alpha_k = \infty, \quad \sum_{k=0}^{\infty} \alpha_k^2 < \infty.$$

Under these assumptions, the parameter update is

$$(\theta_{k+1}, \phi_{k+1}) = (\theta_k, \phi_k) + \alpha_k \hat{f}_k(\theta_k, \phi_k).$$

Theorem B.2 (Stationarity of J_{align} for Lipschitz Case)**.** *Suppose Assumptions **(B1)**–**(B3)** hold. Then, with probability one,*

$$(\theta_k, \phi_k) \xrightarrow[k\to\infty]{a.s.} (\theta^\dagger, \phi^\dagger),$$

where $\nabla J_{align}(\theta^\dagger, \phi^\dagger) = 0$. *In other words,* $(\theta^\dagger, \phi^\dagger)$ *is a stationary point of the augmented objective* J_{align}.

Proof. If ∇J_{align} is Lipschitz (B1), the gradient estimates $\hat{f}_k$ are approximately unbiased with bounded variance (B2), and the step sizes satisfy Robbins–Monro conditions (B3), then the same argument as in Theorem B.1 applies, guaranteeing almost sure convergence to the set of stationary points where $\nabla J_{\text{align}} = 0$. □

Theorem B.2 establishes that, under appropriate Lipschitz conditions (more directly applicable to CARA utility), adding the alignment penalty $\mathcal{L}_{\text{align}}$ still leads to an algorithm that converges almost surely to a stationary point of the augmented objective J_{align}. As discussed for the baseline case, while CRRA utility does not satisfy the global Lipschitz condition, empirical results suggest stability and convergence, motivating further theoretical investigation for this setting as future work. In classical Merton settings, the unique global optimum of J also optimizes J_{align}, and the penalty term aids numerical stability in finding this optimum.

Appendix C. Discrete adjoint BSDE via AD sensitivities.

We define the following pathwise objective:

$$J_{\text{path}}^{(i)} = \sum_{l=0}^{N-1} e^{-\rho t_l} U\big(C_l^{(i)}\big)\, \Delta t + \kappa\, e^{-\rho T}\, U\big(X_N^{(i)}\big).$$

To clarify why the sensitivity $\lambda_k^X = \frac{\partial J_{\text{path}}}{\partial X_k}$ computed by AD satisfies the discrete adjoint BSDE structure required by PMP, we consider simulation paths over the time grid $0 = t_0 < t_1 < \cdots < t_N = T$ with step Δt.

For notational simplicity within this appendix, the following derivation focuses on the component λ_k^X (sensitivity with respect to wealth X_k) and denotes it simply by λ_k. Similarly, Z_k herein refers to the component $Z_{X,k}^X$ related to the noise driving X_k (typically $dW_{X,k}$ in the context of the main text).

We distinguish between pathwise adjoint variables, evaluated along a single Monte–Carlo trajectory (i), and their filtered counterparts obtained by conditioning on the information available at time t_k. Pathwise quantities carry a superscript (i), e.g. $\lambda_k^{(i)}$, whereas filtered quantities omit it, e.g. λ_k. These filtered variables are $\mathcal{F}_{t_k}$-adapted; moreover, the accompanying noise coefficient Z_k is required to be predictable, ensuring that the stochastic integral $\int Z_k\, dW$ is well defined.

1. Pathwise adjoint process via AD. For each trajectory the Euler–Maruyama step is

$$X_{k+1}^{(i)} = X_k^{(i)} + \big[rX_k^{(i)} + \pi_k^{(i)}(\mu - r)X_k^{(i)} - C_k^{(i)}\big]\Delta t + \pi_k^{(i)}\sigma X_k^{(i)}\,\Delta W_k^{(i)}.$$

AD of the pathwise objective $J_{\text{path}}^{(i)}$ gives the pathwise sensitivity

$$\lambda_k^{(i)} := \frac{\partial J_{\text{path}}^{(i)}}{\partial X_k^{(i)}}, \qquad \lambda_N^{(i)} = \kappa\, e^{-\rho T} U'\big(X_N^{(i)}\big).$$

A single backward chain-rule step yields

$$\lambda_k^{(i)} = \lambda_{k+1}^{(i)} + \lambda_{k+1}^{(i)}\big[r + \pi_k^{(i)}(\mu - r)\big]\,\Delta t + \lambda_{k+1}^{(i)}\,\pi_k^{(i)}\sigma\,\Delta W_k^{(i)}.$$

Setting

$$Z_k^{(i)} := -\lambda_{k+1}^{(i)}\,\pi_k^{(i)}\sigma$$

leads to the pathwise BSDE

$$\lambda_k^{(i)} = \lambda_{k+1}^{(i)} + \lambda_{k+1}^{(i)}\big[r + \pi_k^{(i)}(\mu - r)\big]\Delta t - Z_k^{(i)}\,\Delta W_k^{(i)}.$$

2. Adapted λ_k and predictable Z_k. Passing from pathwise quantities to their filtered counterparts, we set

$$\lambda_k \;:=\; \mathbb{E}\big[\lambda_k^{(i)} \mid \mathcal{F}_{t_k}\big], \qquad Z_k \;:=\; \frac{1}{\Delta t}\,\mathbb{E}\big[\lambda_{k+1}^{(i)}\Delta W_k^{(i)} \mid \mathcal{F}_{t_k}\big]. \tag{31}$$

Here, λ_k is $\mathcal{F}_{t_k}$-measurable, hence adapted; Z_k belongs to the predictable σ-algebra (a stronger requirement), so that the stochastic integral $\int Z_k\,dW_k$ is well defined.

We now proceed with the algebra. Let us start with the pathwise BSDE

$$\lambda_k^{(i)} - \lambda_{k+1}^{(i)} \;=\; \lambda_{k+1}^{(i)}[r + \pi_k^{(i)}(\mu - r)]\Delta t \;-\; Z_k^{(i)}\Delta W_k^{(i)}. \tag{32}$$

Taking the conditional expectation of the drift term in (32) gives

$$\begin{aligned}\mathbb{E}\big[\lambda_{k+1}^{(i)}\,[r + \pi_k^{(i)}(\mu - r)]\,\Delta t \;\big|\; \mathcal{F}_{t_k}\big] &= \mathbb{E}\big[\lambda_{k+1}^{(i)} \mid \mathcal{F}_{t_k}\big]\,\big(\,r + \pi_k(\mu - r)\,\big)\,\Delta t \\ &= \lambda_{k+1}\,\big(\,r + \pi_k(\mu - r)\,\big)\,\Delta t.\end{aligned}$$

For the stochastic term in (32), we use $Z_k^{(i)} = -\lambda_{k+1}^{(i)}\pi_k\sigma$ and the covariation definition $Z_k = \frac{1}{\Delta t}\mathbb{E}[\lambda_{k+1}^{(i)}\Delta W_k^{(i)} \mid \mathcal{F}_{t_k}]$:

$$\begin{aligned}\mathbb{E}\big[Z_k^{(i)}\,\Delta W_k^{(i)} \mid \mathcal{F}_{t_k}\big] &= \;-\,\pi_k\sigma\,\mathbb{E}\big[\lambda_{k+1}^{(i)}\,\Delta W_k^{(i)} \mid \mathcal{F}_{t_k}\big] \\ &= \;-\pi_k\,\sigma\,Z_k\,\Delta t.\end{aligned}$$

Substituting the two conditional–expectation results and grouping terms,

$$\lambda_k - \lambda_{k+1} = \lambda_{k+1}[r + \pi_k(\mu - r)]\,\Delta t + \pi_k\sigma\,Z_k\,\Delta t + O\big((\Delta t)^{3/2}\big). \tag{33}$$

Then, by martingale representation theorem, we may write (32) as

$$\lambda_k = \lambda_{k+1} + \big[\lambda_{k+1}(r + \pi_k(\mu - r)) + \pi_k\,\sigma\,Z_k\big]\Delta t - Z_k\,\Delta W_k + O(\Delta t),$$

and dropping the $O(\Delta t)$ term yields the discrete predictable BSDE

$$\lambda_k = \lambda_{k+1} + \partial_x\mathcal{H}(t_k, X_k, u_k, \lambda_{k+1}, Z_k)\,\Delta t - Z_k\,\Delta W_k,$$

where $\partial_x\mathcal{H} = \lambda_{k+1}[r + \pi_k(\mu - r)] + \pi_k\,\sigma\,Z_k$.

One can alternatively set

$$\widetilde{Z}_k := \mathbb{E}\big[Z_k^{(i)} \mid \mathcal{F}_{t_k}\big] = -\,\sigma\,\pi_k\,\lambda_{k+1}, \qquad Z_k^{(i)} := -\lambda_{k+1}^{(i)}\pi_k\sigma.$$

A short calculation shows

$$\widetilde{Z}_k = Z_k + O\big((\Delta t)^{1/2}\big),$$

so the two notions coincide as $\Delta t \to 0$. However, inserting $\widetilde{Z}_k$ in place of Z_k in the discrete BSDE leaves a residual drift of order $O((\Delta t)^{3/2})$, whereas the covariation form (31) eliminates this error and gives an exact discrete BSDE on the time grid.

Acknowledgments. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2025-00562904). This research was supported by Global-Learning & Academic research institution for Master's · PhD students, and Postdocs (LAMP) Program of the National Research Foundation of Korea (NRF) grant funded by the Ministry of Education (No. RS-2024-00442775). This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2024-00355646).

REFERENCES

[1] C. Beck, W. E and A. Jentzen, Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations, *Journal of Nonlinear Science*, **29** (2019), 1563-1619.
[2] S. Becker, P. Cheridito and A. Jentzen, Deep optimal stopping, *Journal of Machine Learning Research*, **20** (2019), 1-25.
[3] V. S. Borkar, *Stochastic Approximation: A Dynamical Systems Viewpoint*, Cambridge University Press, Cambridge; Hindustan Book Agency, New Delhi, 2008.
[4] H. Buehler, L. Gonon, J. Teichmann and B. Wood, Deep hedging, *Quantitative Finance*, **19** (2019), 1271-1291.
[5] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous and Y. LeCun, The loss surfaces of multilayer networks, in *Artificial Intelligence and Statistics*, PMLR, 2015, 192-204.
[6] M. Dai, Y. Dong, Y. Jia and X. Y. Zhou, Learning merton's strategies in an incomplete market: recursive entropy regularization and biased gaussian exploration, arXiv preprint, arXiv:2312.11797.
[7] I. Goodfellow, Deep learning, 2016.
[8] M. D. Gunzburger, *Perspectives in Flow Control and Optimization*, Adv. Des. Control, 5, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2003.
[9] J. Han and W. E, Deep learning approximation for stochastic control problems, arXiv preprint, arXiv:1611.07422.
[10] J. Han, A. Jentzen and W. E, Solving high-dimensional partial differential equations using deep learning, *Proceedings of the National Academy of Sciences*, **115** (2018), 8505-8510.
[11] M. Hinze, R. Pinnau, M. Ulbrich and S. Ulbrich, Optimization with PDE Constraints, Math. Model. Theory Appl., 23, Springer, New York, 2009.
[12] C. Huré, H. Pham and X. Warin, Deep backward schemes for high-dimensional nonlinear PDEs, *Mathematics of Computation*, **89** (2020), 1547-1579.
[13] I. Karatzas and S. E. Shreve, *Methods of Mathematical Finance*, Appl. Math. (N. Y.), 39, Springer-Verlag, New York, 1998.
[14] T. S. Kim and E. Omberg, Dynamic nonmyopic portfolio behavior, *The Review of Financial Studies*, **9** (1996), 141-161.
[15] H. J. Kushner and P. G. Dupuis, *Numerical Methods for Stochastic Control Problems in Continuous Time,* vol. 24 of Applications of Mathematics, Springer-Verlag, New York, 1992.
[16] H. J. Kushner and G. G. Yin, *Stochastic Approximation and Recursive Algorithms and Applications*, vol. 35 of Applications of Mathematics, Stoch. Model. Appl. Probab., Springer-Verlag, New York, 2003.
[17] J. Liu, Portfolio selection in stochastic environments, *The Review of Financial Studies*, **20** (2007), 1-39.
[18] R. C. Merton, Optimum consumption and portfolio rules in a continuous-time model, *Journal of Economic Theory*, **3** (1971), 373-413.
[19] C. Munk, *Financial Asset Pricing Theory*, Oxford University Press, Oxford, 2015.

[20] J. Y. Nguwi, G. Penent and N. Privault, A deep branching solver for fully nonlinear partial differential equations, *Journal of Computational Physics*, **499** (2024), Paper No. 112712, 14 pp.
[21] H. Pham, *Continuous-time Stochastic Control and Optimization with Financial Applications*, vol. 61 of Stochastic Modelling and Applied Probability, Springer-Verlag, Berlin, 2009.
[22] E. Platen and N. Bruti-Liberati, *Numerical Solution of Stochastic Differential Equations with Jumps in Finance*, Stoch. Model. Appl. Probab., 64, Springer-Verlag, Berlin, 2010.
[23] M. Raissi, P. Perdikaris and G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *Journal of Computational Physics*, **378** (2019), 686-707.
[24] A. M. Reppen, H. M. Soner and V. Tissot-Daguette, Deep stochastic optimization in finance, *Digital Finance*, **5** (2023), 91-111.
[25] A. M. Reppen and H. M. Soner, Deep empirical risk minimization in finance: Looking into the future, *Mathematical Finance*, **33** (2023), 116-145.
[26] S. Ross, G. Gordon and D. Bagnell, A reduction of imitation learning and structured prediction to no-regret online learning, in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2011, 627-635.
[27] D. Valladão, T. Silva and M. Poggi, Time-consistent risk-constrained dynamic portfolio optimization with transactional costs and time-dependent returns, *Annals of Operations Research*, **282** (2019), 379-405.
[28] J. A. Wachter, Portfolio and consumption decisions under mean-reverting returns: An exact solution for complete markets, *Journal of Financial and Quantitative Analysis*, **37** (2002), 63-91.
[29] E. Weinan, A proposal on machine learning via dynamical systems, *Communications in Mathematics and Statistics*, **5** (2017), 1-11.
[30] J. Yong and X. Y. Zhou, *Stochastic Controls: Hamiltonian Systems and HJB Equations*, vol. 43 of Stochastic Modelling and Applied Probability, Springer-Verlag, New York, 1999.
[31] W. Zhang and C. Zhou, Deep learning algorithm to solve portfolio management with proportional transaction cost, in *2019 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*, IEEE, 2019, 1-10.

Received February 2025; revised June 2025; early access July 2025.