MULTI-TRAJECTORY PHYSICS-INFORMED NEURAL NETWORKS FOR HJB EQUATIONS WITH HARD TERMINAL CONSTRAINTS: OPTIMAL EXECUTION AND HIGH-DIMENSIONAL LQR

Anthime Valin
University College London
anthime.valin@gmail.com

ABSTRACT

Hard terminal constraints are central to Hamilton-Jacobi-Bellman (HJB) formulations of optimal control, yet standard physics-informed neural networks (PINNs) trained with PDE residuals and soft terminal penalties often violate these constraints under rollout, yielding unstable controls near terminal. We propose Multi-Trajectory PINNs (MT-PINNs), which augment PINN training with a differentiable rollout-based loss that propagates terminal constraint penalties backward through time via backpropagation-through-time (BPTT). MT-PINNs directly couple PDE satisfaction with realized trajectory feasibility. We evaluate MT-PINNs on two representative HJB problems with hard terminal constraints. On a Gatheral-Schied optimal execution benchmark with hard-zero terminal inventory, MT-PINNs match closed-form solutions more closely along optimal paths and substantially tighten terminal-inventory enforcement relative to baseline PINNs. We further demonstrate the method on high-dimensional linear-quadratic regulation (LQR) HJBs, where MT-PINNs produce stable controls and strong terminal constraint satisfaction in dimensions where grid solvers are impractical.

1 INTRODUCTION

Many problems in finance, engineering, and economics can be formulated as optimal control problems, where a control input is chosen to minimize a cost functional subject to system dynamics and admissible constraints (Brayton et al., 2014; Mease, 1989) . The solution is characterized by the value function, which typically satisfies a Hamilton-Jacobi-Bellman (HJB) equation. HJB equations provide a unified framework for both computing the optimal value and recovering the optimal feedback control, but closed-form solutions are only available in a limited number of settings.

For HJB equations whose solution cannot be analytically derived, traditionally, numerical methods, such as finite differencing, are employed. These numerical schemes often require forming a mesh grid over the state and time variables, on which the PDE is discretized and solved. The computational effort of these mesh-based techniques scale exponentially, known as the curse of dimensionality, making them infeasible for higher dimensional problems (Beck et al., 2020). Neural networks, which use a mesh-free approach, have thus been proposed to solve PDEs, including HJB equations Darbon et al. (2020). Although neural networks are not as prone to the curse of dimensionality, they require labeled solution data, which must be generated by solving the equation itself, and they suffer from unstable training and convergence to poor local minima (Krishnapriyan et al., 2021). Motivated by this, physics-informed neural networks (PINNs) have been proposed, which embed the PDE residuals and boundary conditions directly into the loss, eliminating the need for labeled data and enforcing the governing physical constraints of the problem during training (Raissi et al., 2019).

Some optimal control problems, and thus HJB equations, require zero-terminal state. One notable example of this is in optimal execution/liquidation problems, where financial security inventory is required to reach zero at the end of a trading horizon. These constraints create a singularity near terminal (i.e., $\tau = T - t \to 0$), where the value function exhibits non-smooth structure for non-zero state. PINNs, trained on PDE residuals and soft boundary penalties, frequently under-enforce this constraint (Wang et al., 2022), often yielding non-zero state in simulated rollouts, and unstable control near terminal.

This work proposes Multi-trajectory PINNs (MT-PINNs) as a means to improve the accuracy of the value function, optimal controls, and terminal enforcement for HJB equations that require a hard zero-terminal state. MT-PINNs directly penalize terminal state via a rollout-based trajectory loss, using backpropagation-through-time (BPTT) to propagate the terminal penalty through simulated trajectories under the current network from a sample set of initial conditions and horizons (where applicable). This reduces terminal state violations and reduces error along optimal control paths. MT-PINNs are applied to two applications:

(i) Application A: Gatheral-Schied single-asset optimal execution model (Gatheral & Schied, 2011) with a risk-aversion curriculum.

(ii) Application B: High-dimensional hard terminal constraint LQR for scalability testing.

Our main contributions are:

- A trajectory-aware PINN formulation that enforces hard terminal constraints by propagating terminal penalties through simulated dynamics via BPTT.
- Synthetic and real-market validation on an optimal execution problem, demonstrating tighter terminal inventory enforcement, improved path-wise accuracy, and stable controls.
- A high-dimensional hard-terminal LQR study illustrating robustness and scalability beyond low-dimensional PDE settings.

The remainder of the paper is organized as follows. Section §2 reviews the class of problems related to hard terminal constraint optimal control. Section §3 presents the MT-PINN framework. Sections §4 and §5 report results for optimal execution and hard-terminal LQR, respectively, and Section §6 concludes.

2 Problem Class

We consider optimal control problems whose value functions satisfy deterministic Hamilton-Jacobi-Bellman (HJB) equations.

We consider controlled dynamics

$$\dot{x}(t) = f(x(t), u(t), t), \qquad x(0) = x_0, \tag{1}$$

where $x(t) \in \mathbb{R}^d$ denotes the state and $u(t) \in \mathcal{U}$ the control. The objective is to minimize the expected cost functional

$$J(t, x; u) = \mathbb{E}\left[\int_t^T \ell(x(s), u(s), s)\ \mathrm{d}s + g(x(T))\right], \tag{2}$$

with running cost ℓ and terminal cost g.

The associated value function

$$V(t, x) = \inf_u J(t, x; u)$$

satisfies a Hamilton-Jacobi-Bellman equation of the form

$$\partial_t V(t, x) + \min_u \Big\{\ell(x, u, t) + \nabla_x V(t, x)^\top f(x, u, t)\Big\} = 0, \tag{3}$$

with additional second-order diffusion terms when the HJB is derived from stochastic state dynamics.

We focus on problems with hard terminal constraints, encoded through the terminal cost

$$g(x) = \begin{cases} 0, & x \in \mathcal{X}_T, \\ +\infty, & \text{otherwise,} \end{cases} \tag{4}$$

which enforce the feasibility condition $x(T) \in \mathcal{X}_T$ under the optimal policy. Such constraints arise naturally in liquidation and regulation problems and typically induce steep or singular behavior of the value function near maturity.

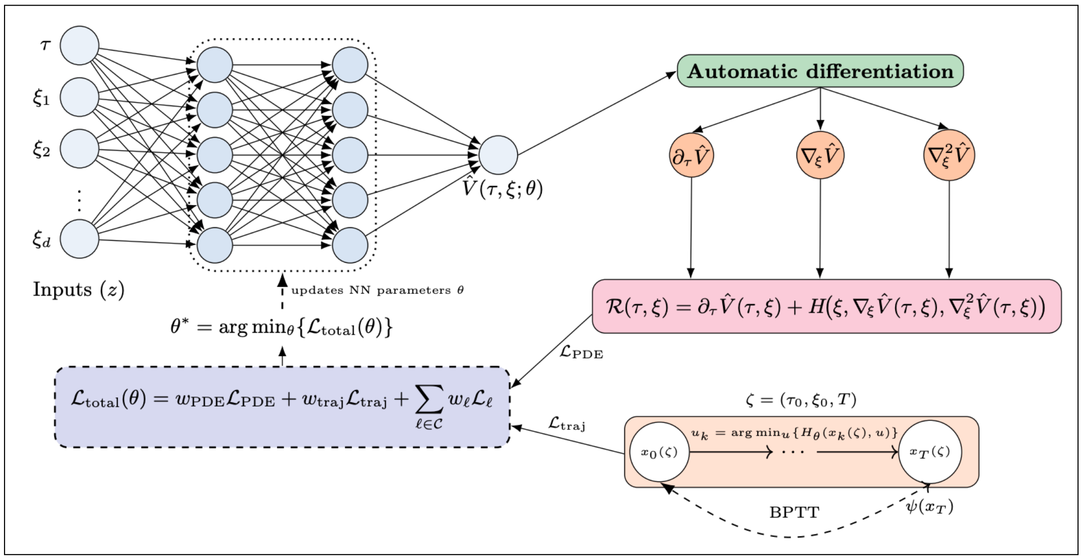

τ
ξ_1
ξ_2
ξ_d
Inputs (z)
$\hat{V}(\tau, \xi; \theta)$
Automatic differentiation
$\partial_\tau \hat{V}$
$\nabla_\xi \hat{V}$
$\nabla_\xi^2 \hat{V}$
$\mathcal{R}(\tau, \xi) = \partial_\tau \hat{V}(\tau, \xi) + H(\xi, \nabla_\xi \hat{V}(\tau, \xi), \nabla_\xi^2 \hat{V}(\tau, \xi))$
updates NN parameters θ
$\theta^* = \arg\min_\theta \{\mathcal{L}_{\text{total}}(\theta)\}$
$\mathcal{L}_{\text{PDE}}$
$\mathcal{L}_{\text{total}}(\theta) = w_{\text{PDE}} \mathcal{L}_{\text{PDE}} + w_{\text{traj}} \mathcal{L}_{\text{traj}} + \sum_{\ell \in \mathcal{C}} w_\ell \mathcal{L}_\ell$
$\mathcal{L}_{\text{traj}}$
$\zeta = (\tau_0, \xi_0, T)$
$u_k = \arg\min_u \{H_\theta(x_k(\zeta), u)\}$
$x_0(\zeta)$
$x_T(\zeta)$
BPTT
$\psi(x_T)$

Figure 1: Core MT-PINN framework diagram. A neural network $\hat{V}(\tau, \xi; \theta)$ outputs an approximation of the value function. Automatic differentiation then provides the derivatives for the PDE. Using this, the PDE residual $\mathcal{R}(\tau, \xi)$ and the controls u_k are calculated. The PDE residual form the PDE loss function $\mathcal{L}_{\text{PDE}}$ and the controls form the trajectory rollout penalty $\mathcal{L}_{\text{traj}}$ via BPTT. Both contribute to the total loss $\mathcal{L}_{\text{total}}(\theta)$, which updates the neural network parameters θ via gradient descent.

3 Method: Multi-Trajectory PINNs

We now describe the Multi-Trajectory Physics-Informed Neural Network (MT-PINN) framework. An overview of the architecture and loss construction is shown in Figure 1.

3.1 Value-function approximation

We approximate the value function by a neural network

$$\hat{V}(\tau, \xi; \theta),$$

where $\tau = T - t$ denotes time-to-maturity and $\xi \in \mathbb{R}^d$ collects the relevant state variables (e.g. inventory and price in optimal execution). The network parameters θ are optimized during training. All required derivatives of $\hat{V}$ with respect to τ and ξ are computed via automatic differentiation.

3.2 HJB residual and PDE loss

Substituting $\hat{V}(\tau, \xi; \theta)$ into the Hamilton-Jacobi-Bellman equation yields the PDE residual

$$\mathcal{R}(\tau, \xi) = \partial_\tau \hat{V}(\tau, \xi) + \mathcal{H}\Big(\xi, \nabla_\xi \hat{V}(\tau, \xi), \nabla_\xi^2 \hat{V}(\tau, \xi)\Big), \tag{5}$$

where $\mathcal{H}$ denotes the Hamiltonian associated with the control problem. For first-order HJBs, the second-derivative term is absent.

The PDE residual loss is defined as

$$\mathcal{L}_{\text{PDE}}(\theta) = \mathbb{E}_{(\tau,\xi)\sim\mathcal{D}_{\text{PDE}}}\big[\mathcal{R}(\tau, \xi)^2\big], \tag{6}$$

where $\mathcal{D}_{\text{PDE}}$ is a sampling distribution over the space-time domain.

3.3 Control recovery and trajectory rollout

The optimal feedback control is recovered analytically from the Hamiltonian minimization,

$$u_k = \arg\min_u \mathcal{H}_\theta(x_k(\zeta), u), \tag{7}$$

where $\zeta = (\tau_0, \xi_0, T)$ denotes a sampled trajectory specification and $x_k(\zeta)$ the discretized state at step k.

Given u_k, trajectories are rolled out forward using a differentiable time discretization,

$$x_{k+1}(\zeta) = x_k(\zeta) + f\big(x_k(\zeta), u_k, t_k\big)\,\Delta t, \qquad k = 0, \dots, N-1. \tag{8}$$

3.4 Trajectory loss for hard terminal constraints

To enforce hard terminal constraints, we penalize violations at the terminal state $x_T(\zeta)$ using a robust penalty function ψ:

$$\mathcal{L}_{\text{traj}}(\theta) = \mathbb{E}_{\zeta \sim \mathcal{D}_{\text{traj}}}\big[\psi(x_T(\zeta))\big]. \tag{9}$$

Gradients of $\mathcal{L}_{\text{traj}}$ are propagated through the full rollout via backpropagation-through-time (BPTT), allowing terminal constraint violations to influence the value-function approximation at earlier times.

3.5 Composite objective

The total training objective is

$$\mathcal{L}_{\text{total}}(\theta) = w_{\text{PDE}}\,\mathcal{L}_{\text{PDE}} + w_{\text{traj}}\,\mathcal{L}_{\text{traj}} + \sum_{\ell \in \mathcal{C}} w_\ell\,\mathcal{L}_\ell, \tag{10}$$

where $\mathcal{C}$ denotes additional problem-specific constraints (e.g. symmetry conditions or internal boundaries) and w represents loss weights adjusted using DWA-style adaptive weighting (Appendix C.2). The parameters θ are optimized via gradient descent.

4 Application I: Optimal Execution with Hard-Zero Terminal Inventory

Model & notation. We study the single-asset optimal execution model of Gatheral and Schied (Gatheral & Schied, 2011), a variant of Almgren-Chriss with a hard terminal liquidation requirement. The unaffected mid-price follows geometric Brownian motion $dS_t = \sigma S_t\,dW_t$ with volatility $\sigma > 0$. Market impact consists of linear permanent impact with strength $\kappa > 0$ and standard temporary impact. We work over a horizon $T > 0$ with time-to-maturity $\tau = T - t$ and initial inventory $X_0 > 0$.

Inventory evolves under the sell trading rate v_t via

$$\dot{X}_t = -v_t, \qquad X_T = 0,$$

where the terminal condition enforces full liquidation. The value function is denoted $\Gamma(\tau, X)$ in the risk-neutral case ($\lambda = 0$) and $\Gamma(\tau, X, S)$ in the risk-averse case ($\lambda > 0$). It represents the minimal expected execution objective from the current state to maturity, accounting for market impact and, when applicable, inventory-price risk aversion λ.

Reduced HJB. The HJB equation differs across regimes. In the risk-neutral case ($\lambda = 0$), the problem is invariant in S, while for $\lambda > 0$ the price becomes a state variable. Minimizing the Hamiltonian with respect to v and substituting the optimal feedback $v^* = \frac{1}{2}\,\partial_X \Gamma$ yields the reduced HJB

$$\frac{\partial \Gamma}{\partial \tau} = \begin{cases} \kappa^2 X^2 - \dfrac{1}{4}\Big(\dfrac{\partial \Gamma}{\partial X}\Big)^2, & \lambda = 0, \\ \dfrac{1}{2}\sigma^2 S^2 \dfrac{\partial^2 \Gamma}{\partial S^2} + \kappa^2 X^2 + \lambda S X - \dfrac{1}{4}\Big(\dfrac{\partial \Gamma}{\partial X}\Big)^2, & \lambda > 0. \end{cases} \tag{11}$$

The hard liquidation constraint is expressed as the singular terminal condition

$$\Gamma(\tau \to 0, X, S) = \begin{cases} 0, & X = 0, \\ +\infty, & X \neq 0. \end{cases}$$

At $\lambda = 0$, the state dimension is one (τ, X); for $\lambda > 0$, it is two (τ, X, S).

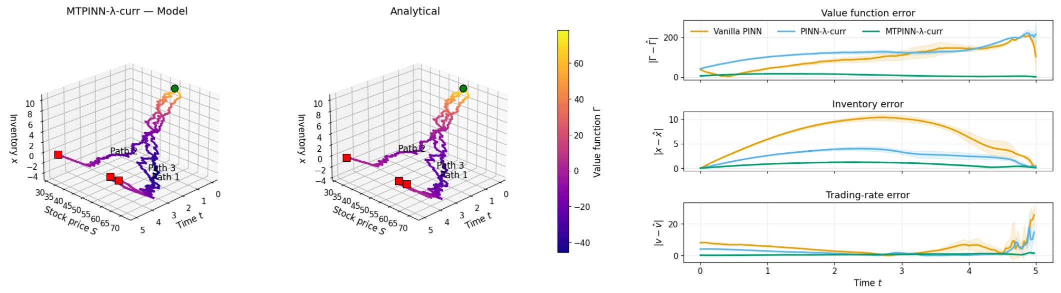


Figure 2: *Left:* MT-PINN vs analytical optimal paths for $\lambda = 0.10$ on three price paths, colored by the value function Γ with a shared colormap. *Right:* three stacked panels show mean $\pm$ one standard deviation across the same paths of absolute errors along the optimal path: (top) $|\Gamma - \hat{\Gamma}|$, (middle) $|x - \hat{x}|$, (bottom) $|v - \hat{v}|$.

MT-PINN parameterization. We approximate the value function with a smooth MLP $\hat{\Gamma}(\tau, X, S; \theta)$, dropping S from the input when $\lambda = 0$. All derivatives in (11) are computed via automatic differentiation. The feedback control used for rollouts is

$$v_\theta^*(\tau, X, S) = \tfrac{1}{2}\,\partial_X \hat{\Gamma}(\tau, X, S).$$

Multi-trajectory terminal-inventory loss. To explicitly enforce the hard constraint $X_T = 0$, we add a rollout-based loss that penalizes terminal inventory across batches of initial states and horizons. Let $\{(X_0^{(p)}, S_0^{(p)})\}_{p=1}^{P} \subset \mathcal{X}_0 \times \mathcal{S}_0$ and horizons $\{T_j\}_{j=1}^{J} \subset (0, T]$. For each pair (p, j), the inventory dynamics are rolled out under the network-implied control using forward Euler discretization with N_{dt} steps:

$$X_{k+1} = X_k - v_\theta^*(\tau_k, X_k, S_k)\Delta t, \qquad \Delta t = \tfrac{T_j}{N_{\text{dt}}}.$$

Denoting the terminal inventory by $X_{T_j}^{(p)}$, we define

$$\mathcal{L}_{\text{traj}} = \frac{1}{PJ}\sum_{p=1}^{P}\sum_{j=1}^{J} \psi\left(X_{T_j}^{(p)}\right), \qquad \psi(x) = \begin{cases} |x|, & |x| \le 1, \\ x^2, & |x| > 1. \end{cases}$$

Gradients are propagated through the rollout via backpropagation-through-time (BPTT); details are provided in Appendix B.2.

Composite loss. Training minimizes the composite objective

$$\mathcal{L}_{\text{total}}(\theta; \lambda) = w_{\text{PDE}}\mathcal{L}_{\text{PDE}} + w_{\text{traj}}\mathcal{L}_{\text{traj}} + w_{\text{IC}}\mathcal{L}_{\text{IC}} + w_{\text{sym}}\mathcal{L}_{\text{sym}} + \mathbf{1}_{\{\lambda>0\}}\, w_0\mathcal{L}_{0\text{-term}},$$

where $\mathcal{L}_{\text{PDE}}$ is the squared residual of (11), $\mathcal{L}_{\text{IC}}$ enforces the inventory-axis condition, $\mathcal{L}_{\text{sym}}$ enforces symmetry, and $\mathcal{L}_{0\text{-term}}$ sets $\Gamma(0, 0, S) = 0$ when $\lambda > 0$.

λ-curriculum. Following Krishnapriyan et al. (2021), we employ a learning curriculum on the risk-aversion parameter $0 = \lambda_0 < \cdots < \lambda_K = \lambda^*$, warm-starting each stage from the previous solution. Training begins at $\lambda = 0$, where the value function is price-invariant, and transitions to $\lambda > 0$. This learning curriculum stabilizes optimization and reduces PDE residuals.

4.1 Synthetic Benchmark

Baselines. We compare MT-PINN against two residual-based PINN baselines that omit the multi-trajectory term and instead enforce a quadratic terminal penalty: (i) a vanilla PINN trained directly at the target λ, and (ii) a vanilla PINN trained with the same λ-curriculum.

Setup. We use horizon $T = 5$, inventory domain $X \in [-10, 10]$, price domain $S \in [10, 100]$, volatility $\sigma = 0.1$, and permanent impact $\kappa = 0.1$. Each model is trained with 30,000 PDE collocation points. For $\lambda > 0$, all methods are trained for 55,000 epochs. The target risk aversion is $\lambda^* \in \{0, 0.05, 0.1\}$, with five curriculum stages. The multi-trajectory loss samples $P = 820$ initial states and rolls out to multiple horizons using 200 Euler steps. Additional details are provided in Appendix D.

Table 1: Absolute terminal-inventory enforcement on the synthetic benchmark for 200 simulated price paths and $\lambda = 0.10$ (for varying λ see Table 4 in Appendix F). We report $p_\varepsilon = \Pr(|X_T| \leq \varepsilon)$ with $\varepsilon = 0.05$.

	$\lvert X_T \rvert$ statistics		
Method	Mean $\pm$ Std	95th pct	p_ε
Vanilla PINN	0.777 ± 0.444	1.407	0.055
PINN-λ-curr	0.164 ± 0.161	0.527	0.205
MT-PINN-λ-curr	**0.073** $\pm$ **0.092**	**0.241**	**0.600**

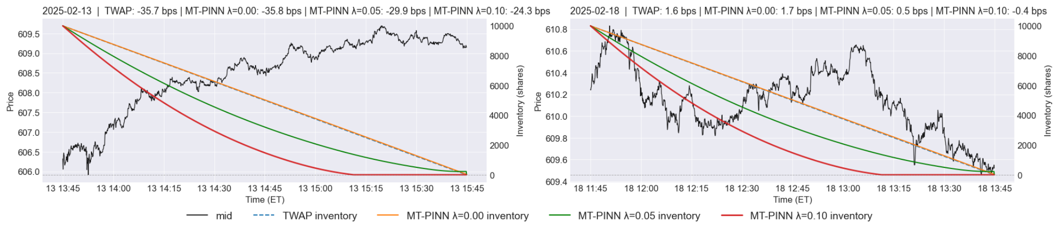


Figure 3: Execution trajectories and costs for MT-PINN with varying $\lambda \in [0, 0.05, 0.1]$ versus Time-Weighted Average Price (TWAP), over two sampled trading windows of the SPY. *Left:* rising SPY window; *Right:* falling SPY window.

Key results. MT-PINN achieves the lowest error along optimal trajectories across value function, inventory, and trading rate metrics, closely tracking the analytical solution without the late-maturity instability exhibited by baseline PINNs (Figure 2). This improvement translates into substantially tighter enforcement of the hard-zero terminal constraint, as shown in Table 1. Additional residual maps and results across λ values are reported in Appendix F.

4.2 Real-Market SPY Backtest

Setup. We apply the model to intraday execution on SPY. Each trading day is split into three 2-hour windows sampled at 5-second intervals, across seven trading days (Feb 10–19, 2025). The first and last 15 minutes of each day are excluded to avoid elevated volatility. Inventory is normalized by the initial position, and time is expressed in trading-day units ($T = 0.308$). Daily realized volatility is estimated from data ($\sigma = 0.0038$), and permanent impact is set to $\kappa = 0.2$. MT-PINN with $\lambda \in \{0, 0.05, 0.1\}$ is compared against TWAP. Further details are provided in Appendix E.

Key results. MT-PINN with $\lambda = 0$ closely matches TWAP, both in exposure (0.336 vs. 0.334) and cost (-6.37 vs. -6.35 bps), confirming the risk-neutral case (see Table 2). Increasing risk aversion traces a clean risk-cost frontier: mean exposure drops while average cost rises moderately (likely due to the general uptrend of the SPY). Figure 3 shows $\lambda > 0$ front-loads the execution and outperforms TWAP in down-moves, whereas MT-PINN (with $\lambda = 0$)/TWAP remains competitive in rising windows. Across all windows, MT-PINN policies remain smooth and strongly satisfy the zero terminal inventory constraint, aligning with the synthetic benchmark (see Appendix F for more results).

5 Application II: High-Dimensional LQR with Hard Terminal Constraint

We next evaluate MT-PINNs on a high-dimensional Linear-Quadratic Regulator (LQR) problem with a hard terminal constraint.

Problem formulation. We consider a deterministic LQR in dimension d, with linear dynamics

$$\dot{x}(t) = Ax(t) + Bu(t), \qquad x(0) = x_0 \in \mathbb{R}^d, \tag{12}$$

Table 2: Execution exposure and cost comparison of MT -PINN (varying λ) vs. TWAP. Results averaged over $n = 21$ windows. Costs in basis points (1 bps = 0.01%).

Model	**Mean Exposure**	**Exposure Std.**	**Mean Cost (bps)**	**Cost Std. (bps)**
TWAP	0.334	0.0000	-6.35	12.56
MT-PINN $\lambda = 0.00$	0.336	0.0000	-6.37	12.58
MT-PINN $\lambda = 0.05$	0.231	0.0004	-5.01	11.02
MT-PINN $\lambda = 0.10$	0.164	0.0002	-3.69	9.67

Table 3: Terminal-state inventory enforcement for the 20D LQR example across the 20 states. Reported are the mean and standard deviation of $|x(T)|$, the L_2 norm $||x(T)||_2$, and the maximum absolute component with its index i.

Method	Mean $\|x(T)\|$	Std $\|x(T)\|$	$\\|x(T)\\|_2$	Max $\|x_i(T)\|$ (at x_i)
PINN	0.172	0.110	0.9132	0.4014 (x_3)
MT-PINN	0.0625	0.0447	0.3438	0.1367 (x_1)

and quadratic running cost

$$\ell(x, u) = x^\top Q x + u^\top R u, \tag{13}$$

where $Q \succeq 0$ and $R \succ 0$. The control horizon is $t \in [0, T]$, and we impose a hard terminal constraint

$$x(T) = 0. \tag{14}$$

While the unconstrained LQR admits a smooth quadratic value function characterized by a Riccati ODE, the hard terminal condition induces singular behavior of the value function as $t \to T$, analogous to the liquidation constraint in optimal execution.

Reference solution and HJB structure. The constrained LQR admits a closed-form solution via a time-dependent Riccati equation whose coefficients diverge near maturity in order to enforce $x(T) = 0$. We use this solution as a reference for evaluating value function accuracy, control accuracy, and terminal-state enforcement. Details of the Riccati formulation andMT-PINN instantiation are provided in Appendix G.

MT-PINN instantiation. We approximate the value function by a neural network $\hat{V}_\theta(t, x)$. The optimal feedback control is recovered analytically from the value gradient as

$$u^*_\theta(t, x) = -\tfrac{1}{2} R^{-1} B^\top \nabla_x \hat{V}_\theta(t, x), \tag{15}$$

ensuring consistency between the HJB residual, the control policy, and the rollout dynamics. As in Application I, MT-PINN augments the PDE residual loss with a multi-trajectory rollout loss that penalizes violations of the terminal constraint. Trajectories are simulated forward under the network-implied control using Euler discretization, and gradients are propagated through time using backpropagation-through-time (BPTT).

Experimental setup. We evaluate the method on a $d = 20$-dimensional LQR problem. Initial states are sampled uniformly from a bounded hypercube, and all models are trained on identical PDE collocation sets and network architectures. MT-PINN is compared against a vanilla PINN baseline that omits the trajectory loss and instead enforces the terminal condition through a quadratic terminal value penalty. All matrix specifications, sampling ranges, and training hyperparameters are reported in Appendix G.4.

Results. We first examine terminal-state enforcement across simulated rollouts. Table 3 reports summary statistics of the terminal state error, including component-wise absolute errors and the ℓ_2-norm $\|x(T)\|_2$. Despite achieving low PDE residuals, the vanilla PINN exhibits substantial dispersion in terminal states, with many trajectories failing to reach the origin. In contrast, MT-PINN achieves a sharp concentration of terminal states around zero, reducing both the mean and tail of the

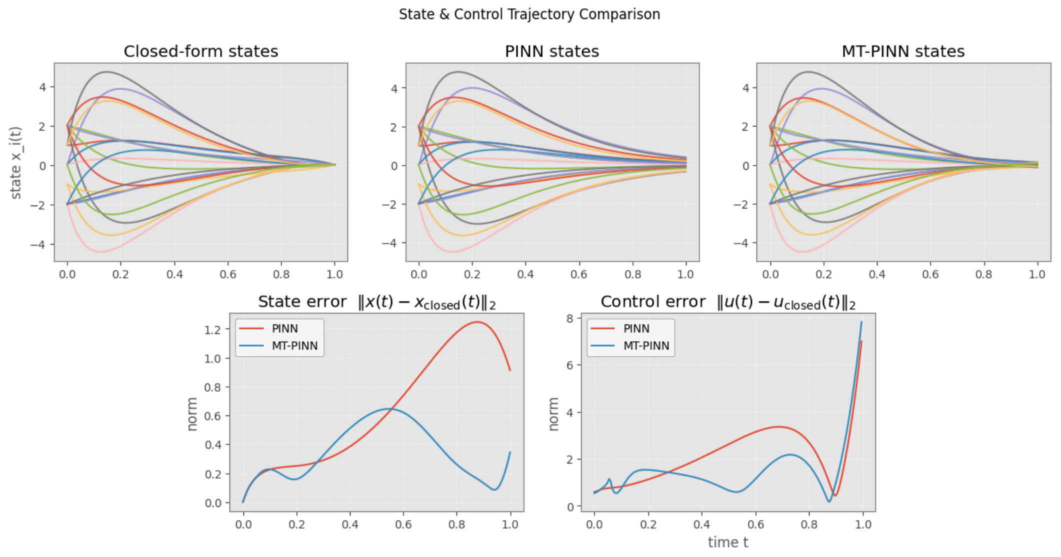


Figure 4: Inventory trajectories $x^*(t)$ and control policy $u^*(t)$ comparison between closed-form solution, vanilla PINN, and MT-PINN

terminal error distribution by an order of magnitude. This demonstrates that enforcing the terminal constraint through a soft value penalty is insufficient in high dimensions.

Representative state trajectories under the learned controls are shown in Figure 4. While vanilla PINN trajectories remain smooth early in the horizon, they fail to collapse near maturity and exhibit systematic drift. MT-PINN trajectories, by contrast, track the Riccati solution throughout the horizon and sharply enforce the terminal constraint as $t \to T$.

Discussion. Taken together, these results show that MT-PINNs outperform residual-based PINNs in a 20-dimensional LQR with a hard terminal constraint, despite identical PDE residual training. Terminal feasibility emerges as a global dynamical property that must be enforced through trajectories rather than boundary penalties. This experiment confirms that MT-PINNs scale beyond low-dimensional financial models and remain effective even in high-dimensional control problems.

6 Conclusion

We introduced Multi-Trajectory Physics-Informed Neural Networks (MT-PINNs) as a framework for solving Hamilton-Jacobi-Bellman (HJB) equations with hard zero-terminal constraints. The central idea is to augment standard residual-based PINN training with differentiable trajectory rollouts and backpropagation-through-time, explicitly propagating terminal feasibility through the system dynamics rather than enforcing it only through boundary penalties. This addresses a common failure mode of PINNs in HJB problems with singular terminal structure, where low PDE residuals do not necessarily imply feasible or stable control policies.

Empirically, MT-PINNs consistently improved terminal constraint enforcement and reduced path-wise errors in value functions, state trajectories, and controls relative to vanilla PINNs. In synthetic optimal execution benchmarks, MT-PINNs closely tracked analytical solutions and substantially tightened zero-terminal inventory enforcement, particularly near maturity. When combined with a learning curriculum on the risk-aversion parameter, training became more stable in regimes where the HJB transitions from smooth to singular. In real-market SPY backtests, MT-PINNs recovered realistic risk-cost trade-offs, matching TWAP in the risk-neutral limit and producing risk-aware, front-loaded execution strategies under positive risk aversion.

Beyond financial execution models, MT-PINNs were shown to scale to a high-dimensional LQR problem with a hard terminal constraint, where residual-only PINNs failed to enforce feasibility despite achieving low PDE residuals. In this setting, MT-PINNs achieved substantially tighter terminal-state concentration and more stable rollouts, confirming that hard terminal constraints are global dynamical properties that must be enforced through trajectories rather than local boundary conditions. Together, these results suggest that MT-PINNs provide a mesh-free and scalable alternative to grid-based numerical solvers for constrained optimal control problems.

Limitations. MT-PINNs incur additional computational overhead due to multi-trajectory rollouts and backpropagation-through-time, increasing both runtime and memory requirements relative to standard PINNs. Moreover, stable performance currently depends on careful loss reweighting and heuristic choices for rollout horizons and initial-state sampling. In the financial experiments, market assumptions are simplified (e.g., absence of transaction costs and explicit microstructure), which limits direct real-world deployment.

Future work. Several directions could further improve the framework. Adaptive selection of rollout horizons and sampling distributions may reduce computational cost while focusing training on regions where terminal enforcement is most challenging. Extending MT-PINNs to multi-asset or portfolio-level control problems would test scalability and practical relevance. Finally, evaluating MT-PINNs in richer market environments or control settings, and incorporating more principled loss-balancing or adaptive sampling schemes, may further improve robustness and convergence.

Overall, this work demonstrates that trajectory-aware physics-informed learning is a promising direction for solving HJB equations with hard constraints, and highlights the importance of coupling value-based PDE learning with dynamical supervision in high-dimensional optimal control.

REFERENCES

Christian Beck, Fabian Hornung, Martin Hutzenthaler, Arnulf Jentzen, and Thomas Kruse. Overcoming the curse of dimensionality in the numerical approximation of allen–cahn partial differential equations via truncated full-history recursive multilevel picard approximations. *Journal of Numerical Mathematics*, 28(4):197–222, 2020.

Flint Brayton, Thomas Laubach, and David L. Reifschneider. Optimal-control monetary policy in the frb/us model. FEDS Notes 2014-11-21-1, Board of Governors of the Federal Reserve System, 2014. URL `https://www.federalreserve.gov/econresdata/notes/feds-notes/2014/optimal-control-monetary-policy-in-frbus-20141121.html`.

Jérôme Darbon, Gabriel P Langlois, and Tingwei Meng. Overcoming the curse of dimensionality for some hamilton–jacobi partial differential equations via neural network architectures. *Research in the Mathematical Sciences*, 7(3):20, 2020.

Jim Gatheral and Alexander Schied. Optimal trade execution under geometric brownian motion in the almgren and chriss framework. *International Journal of Theoretical and Applied Finance*, 14 (03):353–368, 2011.

A. Can Inci and Deniz Ozenbas. Intraday volatility and the implementation of a closing call auction at borsa istanbul. *Emerging Markets Review*, 33:79–89, 2017. ISSN 1566-0141. doi: https://doi.org/10.1016/j.ememar.2017.09.002. URL `https://www.sciencedirect.com/science/article/pii/S1566014117303552`.

Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in neural information processing systems*, 34:26548–26560, 2021.

Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1871–1880, 2019.

Kenneth D. Mease. Trajectory optimization and guidance for an airbreathing-assisted orbital ascent. Technical report, NASA, 1989. URL `https://ntrs.nasa.gov/api/citations/19900004053/downloads/19900004053.pdf`.

Maziar Raissi, Paris Perdikaris, and George 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*, 378:686–707, 2019.

Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality is all you need for training physics-informed neural networks. *arXiv preprint arXiv:2203.07404*, 2022.

A MATHEMATICAL DETAILS & PROOFS

We seek the optimal strategy to liquidate an initial inventory of $X > 0$ shares over some fixed time horizon $[0, T]$. The unaffected asset price follows a driftless geometric Brownian motion (GBM):

$$dS_t = \sigma S_t dW_t, \quad S_0 > 0$$

Gatheral and Schied in (Gatheral & Schied, 2011) assume that the number of shares is represented by an absolutely continuous adapted process X_t that evolves as:

$$X_t = X - \int_0^t v_s ds,$$

with boundary conditions $X_0 = X$ and $X_T = 0$ and where $v_t = \dot{X}_t \in \mathbb{R}$ is the trading rate at time t. Furthermore, transactions occur at

$$\tilde{S}_t = S_t + \eta v_t + \gamma(X_t - X_0),$$

where S_t is the unaffected stock price process, ηv_t models temporary impact, and $\gamma(X_t - X_0)$ models permanent impact. Following (Gatheral & Schied, 2011), the temporary impact coefficient is scaled to one and the effect of permanent impact is represented by a quadratic inventory penalty with parameter κ.

The control problem minimizes the expected cumulative cost

$$J = \mathbb{E}\left[\int_0^T (v_t^2 + \kappa^2 X_t^2 + \lambda X_t S_t) dt\right],$$

where $\kappa > 0$ weights inventory-holding, $\lambda \geq 0$ denotes the risk aversion parameter (VaR-based penalty on inventory exposure), and v_t^2 encodes the temporary impact costs.

We formulate two risk-aversion regimes in the HJB equation: when $\lambda = 0$ and when $\lambda > 0$.

Proposition 1 (Risk-Neutral HJB). *For zero risk penalty (i.e., $\lambda = 0$), the resulting stochastic control problem, whose value function $\Gamma(\tau, X)$ with $\tau := T - t$ satisfies the nonlinear HJB equation under optimal control*

$$\frac{\partial \Gamma}{\partial \tau} = \kappa^2 X^2 - \frac{1}{4}\left(\frac{\partial \Gamma}{\partial X}\right)^2 \tag{16}$$

with

$$\Gamma(\tau, 0) = 0, \ \ \forall \tau \in [0, T] \tag{17}$$

subject to the terminal condition

$$\Gamma(0, X) = \begin{cases} 0, & \textit{if } X = 0 \\ +\infty, & \textit{otherwise.} \end{cases} \tag{18}$$

Proof. The cost functional for zero risk penalty is

$$J(\tau, X) = \mathbb{E}\left[\int_0^\tau (v_t^2 + \kappa^2 X_t^2) dt\right],$$

and thereby no longer dependent on the price S_t. The value function can then be written as the infimum of the cost functional

$$\Gamma(\tau, X) = \inf_{v \in \mathbb{R}} \mathbb{E}\left[\int_0^\tau ((v_t^2 + \kappa^2 X_t^2) dt)\right],$$

Using the Principle of Optimality, we can represent the value function as:

$$\Gamma(\tau, X) = \inf_{v_{[t,t+\Delta t]}} \int_t^{t+\Delta t} (v_s^2 + \kappa^2 X_s^2) ds + \Gamma(\tau - \Delta t, X(t + \Delta t)) \tag{19}$$

For small Δt:

$$\int_t^{t+\Delta t} (v_s^2 + \kappa^2 X_s^2) ds = (v_t^2 + \kappa^2 X_t^2)\Delta t + \mathcal{O}(\Delta t^2), \text{ and}$$

$$\begin{aligned} X(t + \Delta t) &= X + \dot{X}\Delta t + \mathcal{O}(\Delta t^2) \\ &= X - v_t \Delta t + \mathcal{O}(\Delta t^2) \end{aligned}$$

Using Taylor expansion, the value function can be expressed by:

$$\Gamma(\tau - \Delta t, X - v_t \Delta t) = \Gamma(\tau, X) - \frac{\partial \Gamma}{\partial \tau}\Delta t - v\frac{\partial \Gamma}{\partial X}\Delta t + \mathcal{O}(\Delta t^2)$$

Inserting these back into 19 and subtracting $\Gamma(\tau, X)$ from both sides:

$$0 = \inf_{v \in \mathbb{R}} \left[v^2 + \kappa^2 X^2 - \frac{\partial \Gamma}{\partial \tau} - v\frac{\partial \Gamma}{\partial X} \right] \Delta t + \mathcal{O}(\Delta t^2)$$

Dividing by Δt and letting $\Delta t \to 0$:

$$\frac{\partial \Gamma}{\partial \tau} = \kappa^2 X^2 + \inf_{v \in \mathbb{R}} \left[v^2 - v\frac{\partial \Gamma}{\partial X} \right]$$

The optimal control is calculated by minimizing the Hamiltonian $H(\frac{\partial \Gamma}{\partial X}, v) := v^2 - v\frac{\partial \Gamma}{\partial X}$ by setting its derivative with respect to v to zero. This yields the optimal trading rate with respect to the value function:

$$v^* = \frac{1}{2}\frac{\partial \Gamma}{\partial X} \tag{20}$$

This further reduces the HJB to obtain (16)

$$\frac{\partial \Gamma}{\partial \tau} = \kappa^2 X^2 - \frac{1}{4}\left(\frac{\partial \Gamma}{\partial X}\right)^2$$

Note that the cost functional is minimized when implementing the ”do nothing” control (i.e., $X_t = 0$), which results in the condition 17:

$$\Gamma(\tau, 0) = 0, \quad \forall \tau \in [0, T]$$

□

Proposition 2 (Risk-Averse HJB). *For positive risk penalty (i.e., $\lambda > 0$), the resulting stochastic control problem, whose value function $\Gamma(\tau, X, S)$ with $\tau := T - t$ satisfies the nonlinear HJB equation under optimal control*

$$\frac{\partial \Gamma}{\partial \tau} = \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 \Gamma}{\partial S^2} + \kappa^2 X^2 + \lambda S X - \frac{1}{4}\left(\frac{\partial \Gamma}{\partial X}\right)^2, \tag{21}$$

with

$$\Gamma(\tau, 0, S) \leq 0 \quad \forall \tau \in [0, T], \ \forall S > 0, \tag{22}$$

subject to terminal condition

$$\Gamma(0, X, S) = \begin{cases} 0, & \textit{if } X = 0 \\ +\infty, & \textit{otherwise} \end{cases} \tag{23}$$

Proof. Gatheral and Schied in (Gatheral & Schied, 2011) derive the HJB equation in equation (3.18) as

$$\frac{\partial \Gamma}{\partial \tau} = \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 \Gamma}{\partial S^2} + \kappa^2 X^2 + \lambda S X + \inf_{v \in \mathbb{R}} \left(v^2 - v \frac{\partial \Gamma}{\partial X} \right), \tag{24}$$

Reducing the HJB using 20, we obtain 21. To obtain 22, suppose a deterministic $y \in H_0^1(0,\tau)$ with non-zero integral and such that $y(0) = y(\tau) = 0$, e.g., $y(t) = \sin(\frac{\pi t}{\tau})$. For $\epsilon > 0$ set:

$$X_t^{(\epsilon)} := \epsilon y(t), \quad v_t^{(\epsilon)} := \dot{X}_t^{(\epsilon)} = \epsilon \dot{y}(t)$$

As $X_0^{(\epsilon)} = X_\tau^{(\epsilon)} = 0$, $X^{(\epsilon)} \in \mathbb{R}$. Its cost functional is

$$J(\tau, X, S) = \mathbb{E}\left[\int_0^\tau \left(\left(v_t^{(\epsilon)}\right)^2 + \kappa^2 \left(X_t^{(\epsilon)}\right)^2 + \lambda X_t^{(\epsilon)} S_t \right) dt \right]$$

As $X^{(\epsilon)}$ is deterministic and $\mathbb{E}[S_t] = S_0$ for driftless GBM:

$$\mathbb{E}\left[\int_0^\tau \lambda X_t^{(\epsilon)} S_t dt \right] = \lambda S_0 \epsilon \int_0^\tau y(t) dt,$$

such that

$$J(\tau, X, S) = \underbrace{\epsilon^2 \int_0^\tau \dot{y}(t)^2 + \kappa^2 y(t)^2 dt}_{:=A\epsilon^2} + \underbrace{\lambda S_0 \epsilon \int_0^\tau y(t) dt}_{:=B\epsilon}$$

From Proposition 1, we know $A\epsilon^2 \geq 0$. Choose sign of y so that $B \leq 0$. Then for sufficiently small $\epsilon \in [0, -B/2A]$:

$$J(\tau, X, S) = A\epsilon^2 + B\epsilon \leq 0, \;\therefore\; \Gamma(\tau, 0, S) \leq 0$$

□

Gatheral and Schied in (Gatheral & Schied, 2011) showed that the closed-form optimal strategy is:

$$X_t^* = \sinh\left((T-t)\kappa\right) \left[\frac{X}{\sinh(T\kappa)} - \frac{\lambda}{2\kappa} \int_0^t \frac{S_s}{1 + \cosh((T-s)\kappa)} ds \right]$$

with trading rate $v_t^* = -\dot{X}_t^*$, such that:

$$v_t^* = X_t^* \kappa \coth(\kappa(T-t)) + \frac{\lambda S_t}{2\kappa} \tanh\left(\frac{\kappa(T-t)}{2} \right)$$

In addition, the closed-form value function is given by:

$$\Gamma^*(\tau, X, S) = \kappa X^2 \coth(\tau\kappa) + \frac{\lambda X S}{\kappa} \tanh\left(\frac{\tau\kappa}{2}\right) - \frac{\lambda^2 S^2 e^{\sigma^2 \tau}}{4\kappa^2} \int_0^\tau \left[\tanh\left(\frac{u\kappa}{2}\right) \right]^2 e^{-\sigma^2 u} du$$

Corollary 1 (Zero Risk Penalty Value Function Symmetry Property). *For the value function defined in Proposition 1 the following symmetry holds true:*

$$\Gamma(\tau, X) = \Gamma(\tau, -X) \tag{25}$$

Proof.

$$\begin{aligned}\Gamma(\tau,-X) &= \inf_{v\in\mathbb{R}} \mathbb{E}\left[\int_0^\tau v_t^2 + \kappa^2(-X_t)^2 dt\right]\\ &= \inf_{v\in\mathbb{R}} \mathbb{E}\left[\int_0^\tau v_t^2 + \kappa^2 X_t^2 dt\right]\\ &= \Gamma(\tau,X)\end{aligned}$$

□

Corollary 2 (Positive Risk Penalty Value Function Symmetry Property)**.** *For the value function defined in Proposition 2 the following symmetry holds true:*

$$\Gamma(\tau,X,S) = \Gamma(\tau,-X,-S) \tag{26}$$

Proof.

$$\begin{aligned}\Gamma(\tau,-X,-S) &= \inf_{v\in\mathbb{R}} \mathbb{E}\left[\int_0^\tau v_t^2 + \kappa^2(-X_t)^2 + \lambda(-X_t)(-S_t) dt\right]\\ &= \inf_{v\in\mathbb{R}} \mathbb{E}\left[\int_0^\tau v_t^2 + \kappa^2 X_t^2 + \lambda X_t S_t dt\right]\\ &= \Gamma(\tau,X,S)\end{aligned}$$

□

B Core MT-PINN framework

B.1 Generic Architecture

The objective is to approximate the scalar value function $\Gamma(\tau,\xi)$ with a smooth neural network $\hat{\Gamma}(\tau,\xi;\theta)$ that supports accurate derivatives for HJB residuals. The PINNs used in this report, including MT-PINN, were developed in JAX/FLAX, using a single-head Multilayer Perceptron (MLP) with inputs $z = (\tau,\xi) \in \mathbb{R}^{1+d}$, with $\tau \in [0,T]$ (time-to-maturity) and $\xi \in \mathbb{R}^d$ (state), and with tanh smooth activation. Normalization layers and stochastic regularizers are avoided to preserve stable derivatives. Automatic differentiation was used to obtain $\partial_\tau\hat{\Gamma}$, $\nabla_\xi\hat{\Gamma}$, and second-order derivatives of the Hessian, where required, for computing the HJB residuals. Although the optimal control is not specifically parameterized, it is recovered analytically via these derivatives. MT-PINNs use the formulated optimal control derived from the HJB equation to then unroll the system dynamics along sampled trajectories and define a trajectory-based loss (viz., terminal zero-state and running-cost penalties) computed via backpropagation through time (BPTT), thereby coupling the HJB residuals with realized control performance. The intention of this approach is that the trajectory-based loss not only naturally enforces the terminal zero-state condition, but also ensures that the value function along the optimal path is accurate.

B.2 MT-PINN training mechanics (BPTT) & complexity

At each training step, trajectories, via forward Euler discretization, are simulated. At each time step the network is queried to produce the control. After reaching the terminal time, the terminal trajectory loss is computed and then Backpropagation Through Time (BPTT) is used to propagate sensitivities backward through all time steps to obtain the gradient with respect to the network parameters of that trajectory loss. Then, an Adam optimizer performs one parameter update using the sum of gradients from all loss terms.

For a single trajectory, consider the loss:

$$\mathcal{L}(\theta) = \sum_{k=0}^{N-1} l_k(x_k) + l_N(x_N),$$

where l_k are all the running penalties and l_N is the terminal penalty. In many applications, a terminal-only penalty is used (i.e., $l_k \equiv 0$ for $k < N$). Nonetheless, the derivation below covers both cases.

Let θ denote all trainable network parameters and $x_k \in \mathbb{R}^n$ denote the state at step $k = 0, .., N$. Let z_k be the per-step exogenous inputs. A differentiable step map $f_\theta : \mathbb{R}^n \times \mathcal{U} \to \mathbb{R}^n$ advances the state by one time step:

$$x_{k+1} = f_\theta(x_k, z_k),$$

For an explicit forward Euler discretization of continuous dynamics $\dot{x} = g_\theta(x, z)$, we iterate the step map to obtain $x_1, ..., x_N$ and evaluate $\mathcal{L}(\theta)$ as follows:

$$x_{k+1} = x_k + g_\theta(x_k, z_k)\Delta t,$$

assuming that f_θ is differentiable in both x and θ.

Then, define the sensitivity/adjoint:

$$\lambda_k := \frac{\partial \mathcal{L}}{\partial x_k} \in \mathbb{R}^n,$$

and define the local Jacobians

$$A_k := \frac{\partial f_\theta}{\partial x_k}(x_k, z_k) \in \mathbb{R}^{n\times n}, \quad B_k := \frac{\partial f_\theta}{\partial \theta}(x_k, z_k) \in \mathbb{R}^{n\times|\theta|}$$

BPTT then follows as:

$$\begin{aligned}
\text{(initialization)} \quad \lambda_N &= \frac{\partial l_N}{\partial x_N}(x_N), \\
\text{(backward recursion)} \quad \lambda_k &= \frac{\partial l_k}{\partial x_k}(x_k) + A_k^\intercal \lambda_{k+1}, \quad k = N-1, ..., 0, \\
\text{(parameter gradient)} \quad \nabla_\theta \mathcal{L} &= \sum_{k=0}^{N-1} \left(B_k^\intercal \lambda_{k+1} + \frac{\partial l_k}{\partial \theta} \right) + \frac{\partial l_N}{\partial \theta}
\end{aligned}$$

In the case of terminal-only penalties, more specifically when $l_k \equiv 0,$ for $k < N \Rightarrow \partial l_k / \partial x_k = 0$, this can be simplified to:

$$\begin{aligned}
\text{(backward recursion)} \quad \lambda_k &= A_k^\intercal \lambda_{k+1}, \quad k = N-1, ..., 0, \\
\text{(parameter gradient)} \quad \nabla_\theta \mathcal{L} &= \sum_{k=0}^{N-1} \left(B_k^\intercal \lambda_{k+1} + \frac{\partial l_k}{\partial \theta} \right),
\end{aligned}$$

Practically, modern frameworks (e.g., JAX) implement this via reverse-mode autodiff, by applying vector-Jacobian products on the step function at each step to obtain updated sensitivity and parameter gradients.

Complexity. Let N_{dt} be the Euler steps per simulation, B be the number of trajectories in a batch (i.e., number of initial states $\times$ horizons sampled), and C_{f_θ} represents the cost of evaluating the network f_θ.

Time per epoch: $\mathcal{O}(N_{\text{dt}} \times B \times C_{f_\theta})$.

Space per epoch: $\mathcal{O}(N_{\text{dt}} \times B \times A)$, where A denotes the per-step activation memory per network evaluation saved for backpropagation.

Note that these complexities are equivalent to $\mathcal{O}(J \times N_{\text{dt}} \times P)$ used in the main body under $B = P \times J$. Furthermore, this new penalty is also amenable to parallelism as simulated trajectories are independent and vectorized.

Compute resources Experiments were run on 1x v6e-1 TPU and constructed using JAX/FLAX. Typical runtime ranged from 1 minute to 4 minutes.

B.3 Generic Loss Template

Let $\mathcal{C}$ denote the set of constraint/boundary conditions that are application-specific. The total loss for MT-PINNs can be then expressed as:

$$\mathcal{L}_{\text{total}} = w_{\text{PDE}}\mathcal{L}_{\text{PDE}} + w_{\text{traj}}\mathcal{L}_{\text{traj}} + \sum_{\ell \in \mathcal{C}} w_\ell \mathcal{L}_\ell,$$

$$\mathcal{L}_{\text{PDE}} = \mathbb{E}_{z\sim\mathcal{D}_{\text{PDE}}}\left[\mathcal{R}_{\text{HJB}}(z;\theta)^2\right], \quad \mathcal{L}_{\text{traj}} = \mathbb{E}_{\zeta\sim\mathcal{D}_{\text{traj}}}\left[\psi(x_T(\zeta;\theta))\right], \quad \psi(x_T) = \begin{cases} |x_T|, & |x_T| \le 1, \\ x_T^2, & |x_T| > 1. \end{cases}$$

where

- $\mathcal{D}_{\text{PDE}}$: is the sampling distribution of PDE collocation points over z.
- $\mathcal{D}_{\text{traj}}$: is its sampling distribution.
- $\mathcal{R}_{\text{HJB}}(z;\theta)$: is the HJB/PDE residual at z using the network.
- ζ: is a trajectory sample tuple.
- $x_T(\zeta;\theta)$: is the state at terminal time from rolling out dynamics under the control implied by the network starting at ζ.
- $\psi(x_T)$: is the terminal penalty, defined as a piecewise function to enforce zero state/inventory at maturity.
- $w_{\text{PDE}}, w_{\text{traj}}, \{w_\ell\}$ are non-negative.

See Figure 1 for an illustration of the core MT-PINN framework.

C Full MT-PINN Losses & Weighting (DWA)

C.1 Full Loss (by regime)

The total loss function for MT-PINN is expressed as a piecewise function depending on whether the risk-aversion is zero or positive:

$$\mathcal{L}_{\text{total}} = \begin{cases} w_{\text{PDE}}\mathcal{L}^{(0)}_{\text{PDE}} + w_{\text{traj}}\mathcal{L}^{(0)}_{\text{traj}} + w_{\text{IC}}\mathcal{L}^{(0)}_{\text{IC}} + w_{\text{sym}}\mathcal{L}^{(0)}_{\text{sym}}, & \lambda = 0, \\ w_{\text{PDE}}\mathcal{L}^{(\lambda)}_{\text{PDE}} + w_{\text{traj}}\mathcal{L}^{(\lambda)}_{\text{traj}} + w_{\text{IC}}\mathcal{L}^{(\lambda)}_{\text{IC}} + w_{\text{sym}}\mathcal{L}^{(\lambda)}_{\text{sym}} + w_{\text{0-term}}\mathcal{L}_{\text{0-term}}, & \lambda > 0. \end{cases}$$

PDE loss Let Γ_θ denote the value function from the network, and its partial derivatives are $\Gamma_\tau = \frac{\partial \Gamma_\theta}{\partial \tau}$, $\Gamma_X = \frac{\partial \Gamma_\theta}{\partial X}$, $\Gamma_S = \frac{\partial \Gamma_\theta}{\partial S}$, and $\Gamma_{SS} = \frac{\partial^2 \Gamma_\theta}{\partial S^2}$. The PDE residual uses the HJB form according to the regime:

$$\text{For } \lambda = 0: \ \mathcal{R}^{(0)}_{\text{HJB}} = \Gamma_\tau - \kappa^2 X^2 + \frac{1}{4}\Gamma_X^2, \quad \text{(Proposition 1)}$$

$$\text{For } \lambda > 0: \ \mathcal{R}^{(\lambda)}_{\text{HJB}} = \Gamma_\tau - \frac{1}{2}\sigma^2 S^2 \Gamma_{SS} - \kappa^2 X^2 - \lambda S X + \frac{1}{4}\Gamma_X^2. \quad \text{(Proposition 2)}$$

$\mathcal{L}^{(\cdot)}_{\text{PDE}}$ is then defined as:

$$\mathcal{L}^{(0)}_{\text{PDE}} = \frac{1}{N^{(0)}_{\text{PDE}}} \sum_{i=1}^{N^{(0)}_{\text{PDE}}} \left[\mathcal{R}^{(0)}_{\text{HJB}}(\tau_i, X_i)\right]^2, \quad (\tau_i, X_i) \sim \mathcal{D}^{(0)}_{\text{PDE}},$$

$$\mathcal{L}^{(\lambda)}_{\text{PDE}} = \frac{1}{N^{(\lambda)}_{\text{PDE}}} \sum_{i=1}^{N^{(\lambda)}_{\text{PDE}}} \left[\mathcal{R}^{(\lambda)}_{\text{HJB}}(\tau_i, X_i, S_i)\right]^2, \quad (\tau_i, X_i, S_i) \sim \mathcal{D}^{(\lambda)}_{\text{PDE}},$$

Inventory-zero loss (internal condition at $X = 0$. The loss of the internal condition at $X = 0$, defined by (17) for the risk-neutral regime and (22) for the risk-averse regime, is as follows:

$$\mathcal{L}_{IC}^{(0)} \approx \frac{1}{N_{IC}^{(0)}} \sum_{i=1}^{N_{IC}^{(0)}} \Gamma_\theta(\tau_i, 0)^2, \;\; \tau_i \sim \mathcal{D}_{IC}^{(0)}$$

$$\mathcal{L}_{IC}^{(\lambda)} \approx \frac{1}{N_{IC}^{(\lambda)}} \sum_{i=1}^{N_{IC}^{(\lambda)}} \left[\max\{\Gamma_\theta(\tau_i, 0, S_i), 0\}\right]^2, \;\; \tau_i, S_i \sim \mathcal{D}_{IC}^{(\lambda)}.$$

Symmetry loss. The symmetric condition loss encodes Corollary 1 for the risk-neutral regime and Corollary 2 for the risk-averse regime as:

$$\mathcal{L}_{\text{sym}}^{(0)} \approx \frac{1}{N_{\text{sym}}^{(0)}} \sum_{i=1}^{N_{\text{sym}}^{(0)}} \left[\Gamma_\theta(\tau_i, X_i) - \Gamma_\theta(\tau_i, -X_i)\right]^2, \;\; \tau_i, X_i \sim \mathcal{D}_{\text{PDE}}^{(0)},$$

$$\mathcal{L}_{\text{sym}}^{(\lambda)} \approx \frac{1}{N_{\text{sym}}^{(\lambda)}} \sum_{i=1}^{N_{\text{sym}}^{(\lambda)}} \left[\Gamma_\theta(\tau_i, X_i, S_i) - \Gamma_\theta(\tau_i, -X_i, -S_i)\right]^2, \;\; \tau_i, X_i, S_i \sim \mathcal{D}_{\text{PDE}}^{(\lambda)}.$$

Terminal zero-inventory loss. Due to the internal condition imposed on the risk-averse regime of (22), a separate loss term was created to ensure the network learns that $\Gamma(0, 0, S) = 0$ defined as:

$$\mathcal{L}_{\text{0-term}} \approx \frac{1}{N_{\text{term}}} \sum_{i=1}^{N_{\text{term}}} \Gamma(\tau = 0, X = 0, S_i)^2, \quad S_i \sim \mathcal{D}_{\text{term}}.$$

Note that the risk-neutral regime already explicitly imposes this constraint on all (τ, S) for $X = 0$.

Sampler definitions. $\mathcal{D}$ denotes the uniform sampling distribution for each collocation domain. Experiment-specific grids and counts (i.e., $N_{(\cdot)}$, horizon sets, and initial state grids) are provided in the respective experiment appendices.

C.2 DWA-style weight rebalancing

We adapt task weights with a dynamic weight average (DWA)-style scheme (Liu et al., 2019). Instead of using DWA's raw ratio $L_\mathcal{T}(t-1)/L_\mathcal{T}(t-2)$ and softmax, we use a smoothed exponential moving average (EMA) of each loss and update weights from its relative scale. Concretely, we make the following lightweight stabilizations:

- EMA smoothing of losses, relative to initial scaling
- Geometric mean centering across task
- Gentle inverse power-law update instead of softmax and clip weights to $[w_{\text{min}}, w_{\text{max}}]$
- Mean-one normalization of weights across active tasks each update
- Optional freezing for tasks that become tiny over several updates
- Scheduled updates (i.e., updating weights every Δ epochs)

Hyperparameters are listed within the respective experiment appendices.

D Synthetic benchmark: complete experiment specification

The MT-PINN used for this experiment is outlined in Appendix C.1. As a replacement for the trajectory loss, which aims to naturally fulfill the terminal condition stated in equations (18) and (23), both PINN with λ-curriculum and vanilla PINN have a loss term that sets an arbitrarily large penalty for leftover inventory to fulfill the terminal condition. This loss term is formulated as:

$$\mathcal{L}_{\text{term}}^{(0)} = \frac{1}{N_{\text{term}}^{(0)}} \sum_{i=1}^{N} \left(\Gamma_\theta(0, X_i) - cX_i^2\right)^2, \;\; X_i \sim \mathcal{D}_{\text{term}}^{(0)},$$

$$\mathcal{L}_{\text{term}}^{(\lambda)} = \frac{1}{N_{\text{term}}^{(\lambda)}} \sum_{i=1}^{N} \left(\Gamma_\theta(0, X_i, S_i) - cX_i^2\right)^2, \;\; X_i, S_i \sim \mathcal{D}_{\text{term}}^{(\lambda)},$$

where $c > 0$ is the penalty strength. Otherwise the loss terms across the PINN variant models are the same:

$$\mathcal{L}_{\text{total}} = \begin{cases} w_{\text{PDE}}\mathcal{L}_{\text{PDE}}^{(0)} + w_{\text{IC}}\mathcal{L}_{\text{IC}}^{(0)} + w_{\text{sym}}\mathcal{L}_{\text{sym}}^{(0)} + w_{\text{term}}\mathcal{L}_{\text{term}}^{(0)}, & \lambda = 0, \\ w_{\text{PDE}}\mathcal{L}_{\text{PDE}}^{(\lambda)} + w_{\text{IC}}\mathcal{L}_{\text{IC}}^{(\lambda)} + w_{\text{sym}}\mathcal{L}_{\text{sym}}^{(\lambda)} + w_{\text{term}}\mathcal{L}_{\text{term}}^{(\lambda)}, & \lambda > 0. \end{cases}$$

For the synthetic benchmark experiment, three values of risk-aversion were evaluated: $\lambda^* \in \{0, 0.05, 0.1\}$. For $\lambda = 0$ all methods are trained for 30k epochs. For $\lambda > 0$, vanilla PINN was trained for 55k epochs, while the curriculum-based models used a two-phase schedule totaling 55k epochs (30k at $\lambda = 0$, then 5 curriculum stages of 5k epochs each up to the target λ^*).

Common setting (unless stated otherwise). Time horizon $T = 5.0$, state ranges $X \in [-10, 10], S \in [10, 100]$ (the S range only used when $\lambda > 0$). The PDE collocation points were $N_{PDE} = 30,000$ and internal conditions points were $N_{IC} = 5000$ (see Figure 5). Model parameters: $\kappa = 0.1$ (risk-adjusted inventory penalty), $\sigma = 0.1$ (price volatility). Optimization uses AdamW with learning rate 5×10^{-4}. Loss weights are adapted with DWA-style scheme as briefly described in Appendix C.2 with weights updated every $\Delta = 1000$ epochs, weight clipping $[0.1, 2.0]$, EMA smoothing factor $\beta = 0.95$, update strength $\alpha = 0.3$, and freeze tolerance of 10^{-4}. Initial weights were set to $[w_{\text{PDE}} = 1.0, w_{\text{traj}} = 1.0, w_{\text{IC}} = 0.1, w_{\text{sym}} = 0.5, w_{\text{0-term}} = 0.5, w_{\text{term}} = 1.0]$.

Vanilla PINN (single-stage). MLP width of (500, 500, 500). For $\lambda = 0$: 30k epochs. For $\lambda \in \{0.05, 0.10\}$: 55k epochs with the same sampler sizes and network input dimensions were set by the regime. Terminal points were $N_{\text{term}} = 5000$.

PINN + λ-Curriculum (warm-start 1D state to 2D state). Phase A (1D, $\lambda = 0$): MLP width of (500, 500, 500), two input dimensions, and run for 30k epochs. Phase B (2D, $\lambda > 0$): warm start the trained 1D parameters to 2D then train 5 curriculum stages with $\alpha \in (0.25, 0.5, 0.75, 0.9, 1.0)$. Each stage runs 5k epochs, with $\lambda_\alpha = \alpha \cdot \lambda^*$. Terminal points were $N_{\text{term}} = 5000$.

MT-PINN + λ-Curriculum (warm-start 1D state to 2D state). Phase A (1D, $\lambda = 0$): MLP width of (32, 32, 32), two input dimensions, and run for 30k epochs. Phase B (2D, $\lambda > 0$): warm start the trained 1D parameters to 2D then train 5 curriculum stages with $\alpha \in (0.25, 0.5, 0.75, 0.9, 1.0)$. Each stage runs 5k epochs, with $\lambda_\alpha = \alpha \cdot \lambda^*$. Multi-trajectory batch samples $P = 820$ ($n_X = 41$ and where applicable $n_S = 20$) and roll out horizons of $\{T/50, T/10, T/5, 2T/5, 3T/5, 4T/5, T\}$ with $N_{\text{dt}} = 200$ Euler steps. Terminal points were $N_{\text{term}} = 2000$.

E Real-market SPY backtest: data, preprocessing, metrics, & experiment specification

E.1 Data & preprocessing

Intraday market data for SPY was obtained via Databento, sourced from the Nasdaq TotalView-ITCH feed. Data was used under Databento's academic/historical data provisions, subject to Nasdaq TotalView-ITCH license/ToS. We cannot redistribute the raw data.

The trading day is split into three 2-hour fixed intraday windows on 5-second intervals, and repeated across seven days (Feb 10 - Feb 19, 2025, excluding weekends). The windows are:

- Window 1 (W1): 09:45 - 11:45 ET
- Window 2 (W2): 11:45 - 13:45 ET

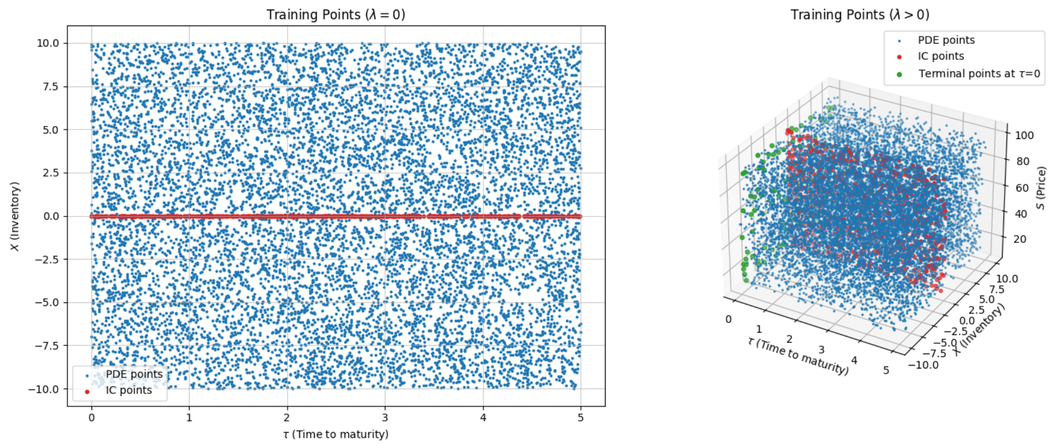


Figure 5: Sampling/collocation points

- Window 3 (W3): 13:45 - 15:45 ET

This intraday segmentation of SPY mid-price is illustrated in Figure 6. Note that these windows exclude the first 15 minutes of market open and the last 15 minutes of market close. This was done to avoid the noisy and elevated volatilities typically observed at those times (Inci & Ozenbas, 2017).

E.2 Metrics

The main criteria for comparing these models were their exposure and cost trade-off. Exposure measures how much inventory the strategy was carrying on average for a given window. This was calculated using

$$\text{Exposure}(W_j) = \frac{1}{N} \sum_{k=0}^{N-1} \left(\chi_k^{(j)} \right)^2,$$

where W_j is the j^{th} trading window, N is the number of discretized equal steps, χ_k is the normalized inventory at step k. Cost is measured as the implementation shortfall relative to liquidating the entire initial position X_0 at the initial mid-price S_0. In practice, this measures the cost associated with using the execution model for liquidation versus dumping the entire position immediately at the current mid-price. Cost was measured in basis points (1 bps = 0.01%) and computed using

$$\text{Cost}(W_j) = \frac{S_0^{(j)} X_0 - \sum_{k=0}^{N-1} q_k^{(j)} S_k^{(j)}}{S_0^{(j)} X_0} \times 10^4 \text{ bps},$$

where q_k denotes the trade at step k. The exposure and cost across all the trading windows are then aggregated to report the mean and standard deviation of the exposure and cost using NumPy built-in functions (see Table 2 for results).

E.3 Experiment specification

Part of applying these models into real-market data meant adjusting the model parameters so that they accurately match the data.

Time Normalization. Time horizon was expressed by trading day units, where elapsed time is normalized by the effective length of a trading day on the U.S. equity markets (6.5 hours, from 9:30 a.m. to 4:00 p.m. EST). Specifically, the two-hour backtest horizon corresponded to

$$T_{\text{days}} = \frac{2}{6.5} \approx 0.308,$$

which represents roughly 31% of the trading day.

Inventory Normalization. Inventory was normalized by the initial position, so that the inventory lies in the interval $[-1, 1]$, and represents the fraction of the initial inventory:

$$\chi_t = \frac{x_t}{X_0},$$

where $\chi_t = 1$ corresponds to holding the full initial inventory, $\chi_t = 0$ to fully liquidated inventory, and $\chi_t = -1$ corresponds to over-liquidation (shorting) of equal size of the initial inventory.

Stock Price Range. Over the full 21 time windows, the stock price range was: \$599.60 $-$ \$613.20. Therefore, the S range used for MT-PINN was $590 - 620$, leaving a small buffer of roughly 1% each side.

Volatility Estimation. To estimate volatility (σ) of the assumed unaffected price process S_t, the realized volatility for each intraday execution window was computed and then averaged. Given a set of stock mid-prices $\{S_{t_i}\}_{i=1}^{N}$ in a given window of length ΔT_{W}, the log-returns can be calculated as

$$r_i = \ln\left(\frac{S_{t_i}}{S_{t_{i-1}}}\right), \quad i = 1, ..., N.$$

Hence the realized volatility for a given window is

$$\hat{\sigma}_{\text{W}}^2 = \frac{1}{N-1} \sum_{i=1}^{N} (r_i - \bar{r})^2,$$

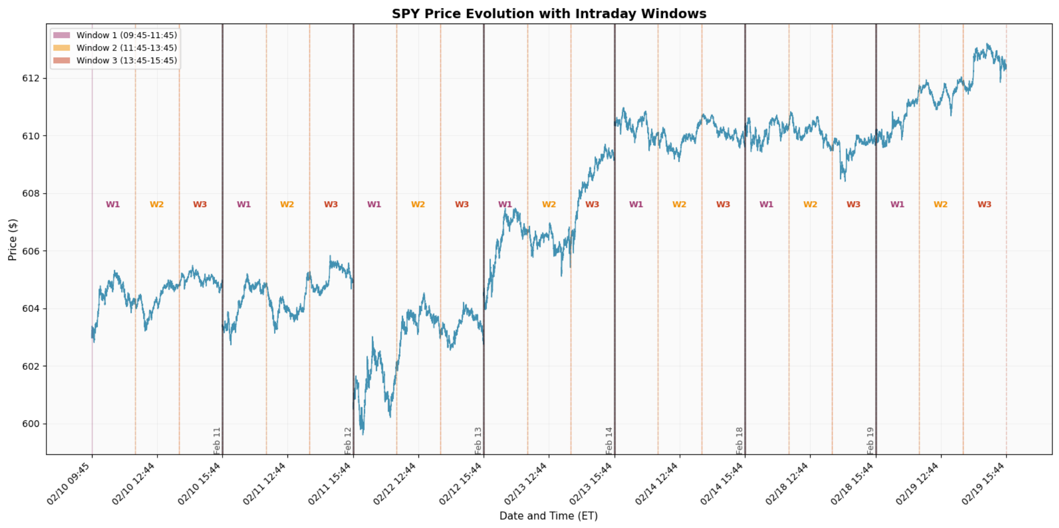


Figure 6: Intraday segmentation of SPY mid-price series into fixed windows. Each trading day is partitioned into three non-overlapping 2-hour intervals: Window 1 (W1) from 09:45 to 11:45 ET, Window 2 (W2) from 11:45 to 13:45 ET, and Window 3 (W3) from 13:45 to 15:45 ET. Vertical dashed lines mark the boundaries between windows, while solid black vertical lines indicate the start of each new trading day.

where $\bar{r}$ denotes the sample mean of returns. For window length ΔT_W, realized volatility was rescaled into trading-day units as

$$\hat{\sigma}_{\text{W, daily}} = \frac{\hat{\sigma}_{\text{W}}}{\sqrt{\Delta T_{\text{W}}}}.$$

Finally, averaging across all execution windows yields the representative daily volatility parameter for the simulation:

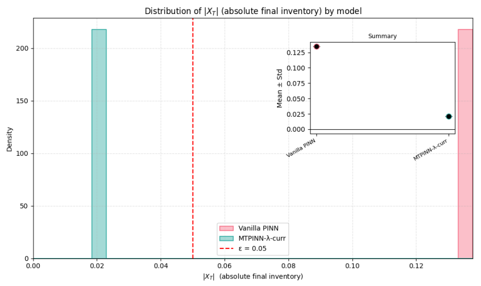
$$\sigma = \frac{1}{J} \sum_{j=1}^{J} \hat{\sigma}^{(j)}_{\text{W, daily}},$$

where J represents the number of execution windows. From the SPY market data, the calculated volatility was:

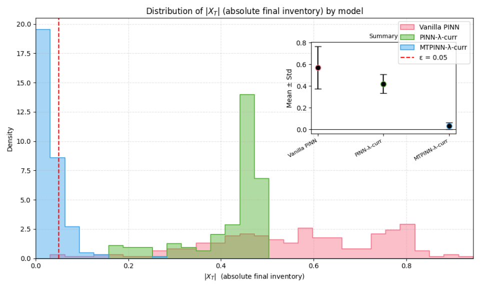
$$\sigma \approx 0.0038 \quad (\approx 6\% \text{ annualized}).$$

MT-PINN (Phase A: 1D, Phase B: 2D). MT-PINN with width (32,32,32), scalar output, and tanh activations. Time horizon $T \approx 0.308$, with horizon grid $T_j \in \{T/20, T/10, T/4, T/2, 3T/4, T\}$, $Ndt_{\text{traj}} = 50$ Euler steps and $n_X = 200$ inventory grid points. Inventory range set to $[-1, 1]$ and $\kappa = 0.2$. This experiment uses the same hyperparameters for the weights as outlined in Appendix D, with the exception to weighting clipping $[0.1, 5.0]$ and initial trajectory loss weighting $w_{\text{traj}} = 5.0$.

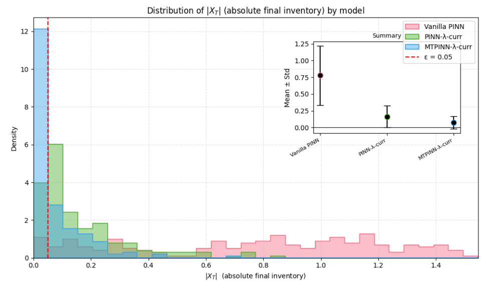
Phase A: uses (τ, X) inputs, 20k training epochs with 30k collocation points, and risk-neutral regime $\lambda = 0$. Phase B: uses (τ, X, S) with price $S \in [590, 620]$, discretized with $n_S = 41$ for multi-trajectory loss, and trained on 20k collocation points for 5k epochs. Volatility set to $\sigma = 0.0038$. Risk aversion parameter introduced through curriculum stages $\alpha \in \{0.25, 0.5, 0.75, 1.0\}$ for $\lambda^* = [0.05, 0.1]$, with $\lambda_\alpha = \alpha \cdot \lambda^*$. Sample sizes: $N_{IC} = 2000, N_{\text{0-term}} = 200$.



(a) $\lambda = 0$

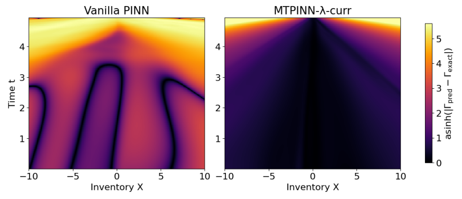


(b) $\lambda = 0.05$

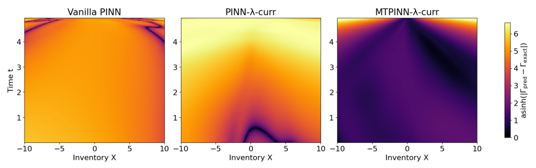


(c) $\lambda = 0.10$

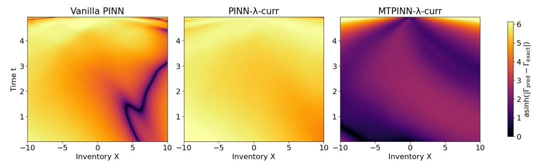
Figure 7: Distribution of $|X_T|$ (absolute final inventory) by model for each risk aversion λ. Each panel shows the histogram across rollouts for the three models (Vanilla PINN, PINN-λ-curr, MT-PINN-λ-curr; PINN-λ-curr omitted for $\lambda = 0$) with the tolerance line at $\epsilon = 0.05$ and the inset reporting mean $\pm$std.



(a) $\lambda = 0$



(b) $\lambda = 0.05$



(c) $\lambda = 0.10$

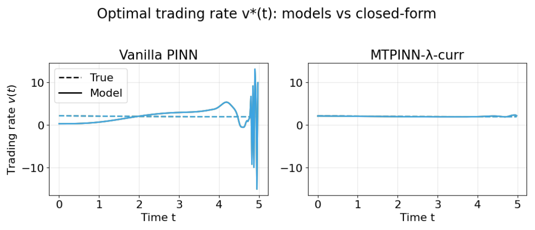
Figure 8: Heatmaps of domain-wide error over (t, X) for each risk aversion λ with fixed $S = 55$. Each panel compares methods as in the per-λ heatmap export.

F Further Synthetic Benchmark & SPY Real-Market Backtest Results

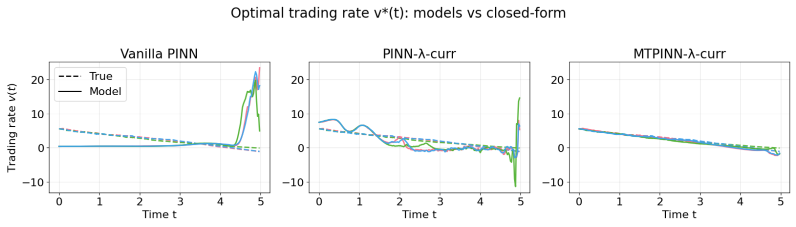
F.1 Synthetic Benchmark Results

For $\lambda = 0$ only vanilla PINN and MT-PINN-λ-curriculum are shown as PINN-λ-curriculum collapses to vanilla PINN and is omitted.

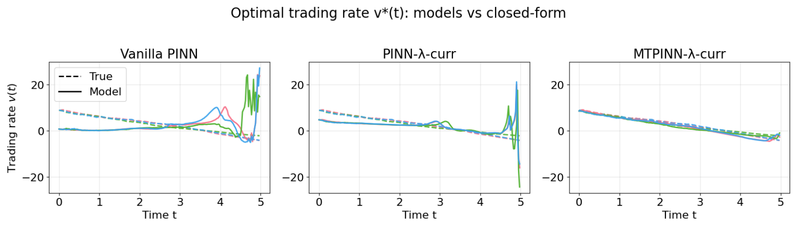
Figure 7 shows the histogram of the absolute terminal inventory $|X_T|$ across 200 stock price simulations with an ϵ tolerance market and mean±std inset of the terminal inventory. It is evident that the MT-PINN-λ-curriculum concentrates heavily near zero and attains the highest pass-rate



(a) $\lambda = 0$



(b) $\lambda = 0.05$



(c) $\lambda = 0.10$

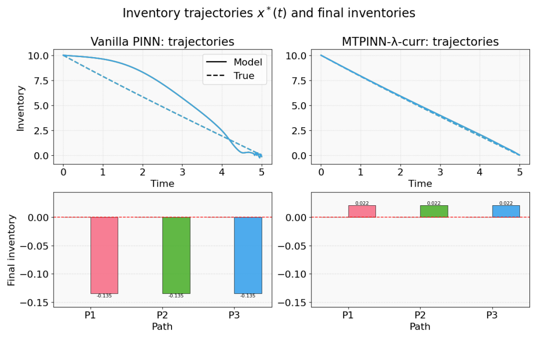
Figure 9: Optimal trading rate $v^*(t)$: model vs. closed form for each λ.

Table 4: Absolute terminal-inventory enforcement on the synthetic benchmark for 200 simulated price paths for varying λ. It is reported $p_\varepsilon = \Pr(|X_T| \le \varepsilon)$ with $\varepsilon = 0.05$.

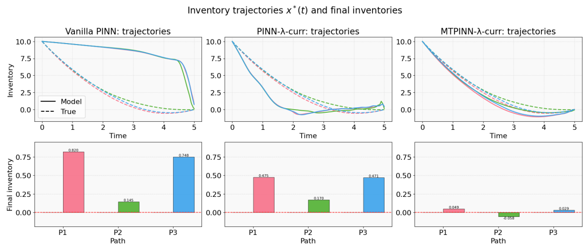
	$\lambda = 0$			$\lambda = 0.05$			$\lambda = 0.10$		
Method	Mean±Std	95th pct	p_ε	Mean±Std	95th pct	p_ε	Mean±Std	95th pct	p_ε
Vanilla PINN	0.135±0.000	0.135	0.000	0.570±0.196	0.826	0.000	0.777±0.444	1.407	0.055
PINN-λ-curr	-	-	-	0.420±0.085	0.481	0.000	0.164±0.161	0.527	0.205
MT-PINN-λ-curr	**0.022±0.000**	**0.022**	**1.000**	**0.031±0.030**	**0.072**	**0.810**	**0.073±0.092**	**0.241**	**0.600**

Table 5: Error statistics for Vanilla PINN, PINN-λ-curr, and MT-PINN-λ-curr across $\lambda \in \{0, 0.05, 0.10\}$ (with PINN-λ-curr omitted at $\lambda = 0$) and fixed stock price $S = 55$. Mean absolute error, max absolute error, mean relative error, and root-mean squared error are reported in both original space and arcsinh-transformed space.

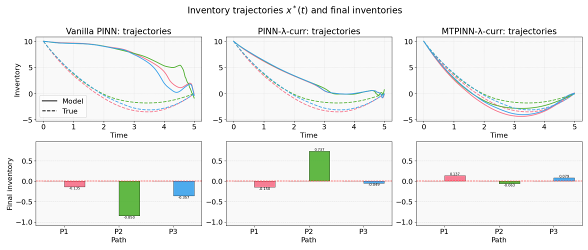
λ	Method	Original Space				Arcsinh Space			
		MAE	MaxAE	MRE	RMSE	MAE	MaxAE	MRE	RMSE
0	Vanilla PINN	22.75	**363.18**	7.43	**45.45**	0.80	4.96	3.66	1.07
	PINN-λ-curr	-	-	-	-	-	-	-	-
	MT-PINN-λ-curr	**16.44**	1773.53	**0.24**	116.00	**0.12**	**2.18**	**0.18**	**0.25**
0.05	Vanilla	78.98	573.98	33.69	**88.12**	3.97	10.03	2.80	5.29
	PINN-λ-curr	145.77	**413.17**	62.52	192.12	3.49	9.30	2.66	4.51
	MT-PINN-λ-curr	**17.73**	1756.74	**0.77**	115.68	**0.28**	**2.10**	**0.51**	**0.50**
0.10	Vanilla PINN	81.22	475.45	33.92	**99.78**	4.33	9.71	2.27	5.61
	PINN-λ-curr	140.12	**270.59**	44.04	147.14	5.23	11.02	2.60	6.57
	MT-PINN-λ-curr	**19.65**	1742.84	**1.09**	115.39	**0.37**	**3.84**	**0.56**	**0.73**



(a) $\lambda = 0$

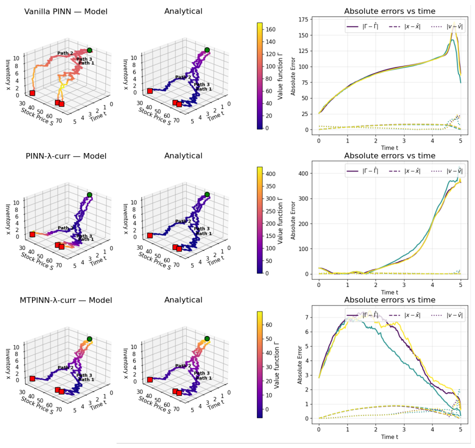


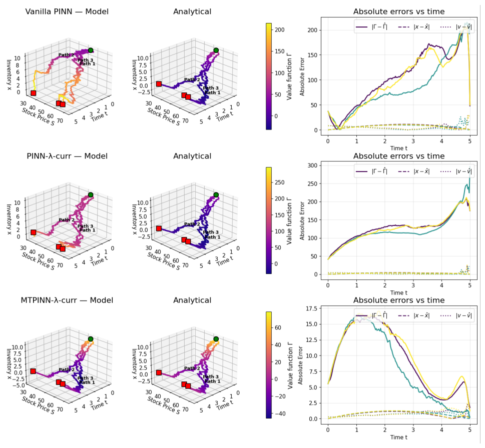
(b) $\lambda = 0.05$



(c) $\lambda = 0.10$

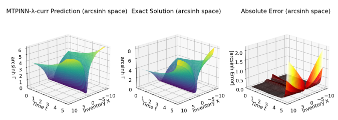
Figure 10: Inventory trajectories $x^*(t)$ and final inventories across paths for each λ.



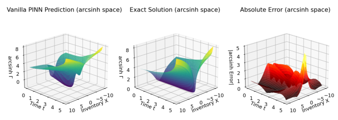


(a) $\lambda = 0.05$ (b) $\lambda = 0.10$

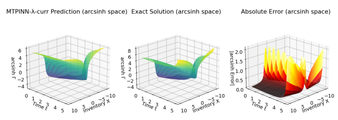
Figure 11: Pathwise rollouts: model vs. analytic trajectories and absolute error over time for each $\lambda > 0$.



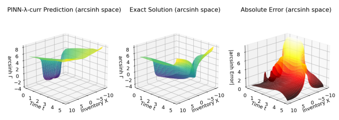
(a) MT-PINN-λ-curr, $\lambda = 0$



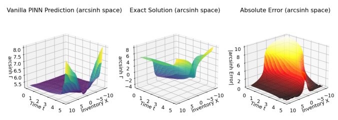
(b) Vanilla PINN, $\lambda = 0$



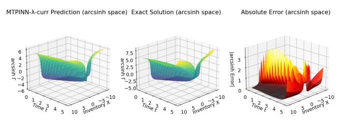
(c) MT-PINN-λ-curr, $\lambda = 0.05$



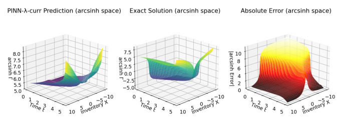
(d) PINN-λ-curr, $\lambda = 0.05$



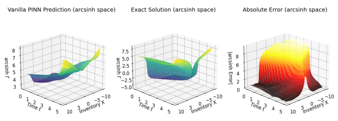
(e) Vanilla PINN, $\lambda = 0.05$



(f) MT-PINN-λ-curr, $\lambda = 0.10$



(g) PINN-λ-curr, $\lambda = 0.10$



(h) Vanilla PINN, $\lambda = 0.10$

Figure 12: Arcsinh-space value surfaces (each panel is organized as: Prediction / Exact / Absolute Error). Rows vary λ, columns vary the method.

$p_\epsilon = Pr(|X_T| \leq \epsilon)$ for every λ (see Table 4). Baselines show larger violations for zero terminal inventory, with the PINN-λ-curriculum exhibiting improvements over the vanilla PINN for larger λ.

Figure 9 illustrates the implied trading rate $v^*(t) = \frac{1}{2}\partial_X \Gamma_\theta$ versus the analytical solution derived by Gatheral and Schied for all three paths. MT-PINN-λ-curriculum follows closer to the closed-form across the horizon for all λ, while the baselines show higher variance.

Figure 10 reveals the inventory trajectories $x^*(t)$ under the learned trading rate for all three paths. Although MT-PINN-λ-curriculum's accuracy in matching the closed-form solution reduces as λ increases, it consistently produces near zero-inventory at terminal and nevertheless outperforms the baseline PINNs. The PINN-λ-curriculum improves on the vanilla PINN for higher λ, somewhat grasping the concept of reduced risk exposure, unlike the vanilla PINN which increases its risk exposure.

Figure 11 displays the models versus analytical inventory trajectories for three paths, showing the value function along the path and the absolute error over time for the value function, inventory trajectory, and trading rate. Note that the plots only show for $\lambda > 0$, because, as previously stated, for $\lambda = 0$, the optimal control and trajectories are invariant to the stock price S. MT-PINN-λ-curriculum shows a reduction in nearer terminal errors, reduced path-variance, and orders of magnitude smaller error than the baselines.

Figure 8 exhibits the domain-wide inverse hyperbolic sine function absolute value function error $(\operatorname{asinh}(|\Gamma_{\text{pred}} - \Gamma_{\text{exact}}|))$ over (t, X) for fixed mid-range stock price $S = 55$. MT-PINN-λ-curriculum has reduced error across the domain with higher error near terminal, as can be seen in Table 5. Table 5 shows that across the arcsinh space, MT-PINN-λ-curriculum performs the best across all metrics. In the original space, MT-PINN-λ-curriculum performs best in mean absolute error and mean-relative error, while PINN-λ-curriculum consistently performed best in root-mean-squared error and, for the most part, maximum absolute error. The rationale for why MT-PINN-λ-curriculum performs better in the arcsinh space than it does in the original space is because the errors in the outliers, mostly found near maturity, aren't penalized as much in the arcsinh space. This demonstrates that the baseline PINNs capture the singularity in the value function with less error, however, overall, MT-PINN-λ-curriculum better captures the value function across the entire domain.

Figure 12 provides a 3D view of Figure 8. It is evident that, domain-wide, the baseline PINNs have less error near terminal than the MT-PINN-λ-curriculum due to their terminal condition loss; however, as has been outlined, this doesn't result in accurate control policy.

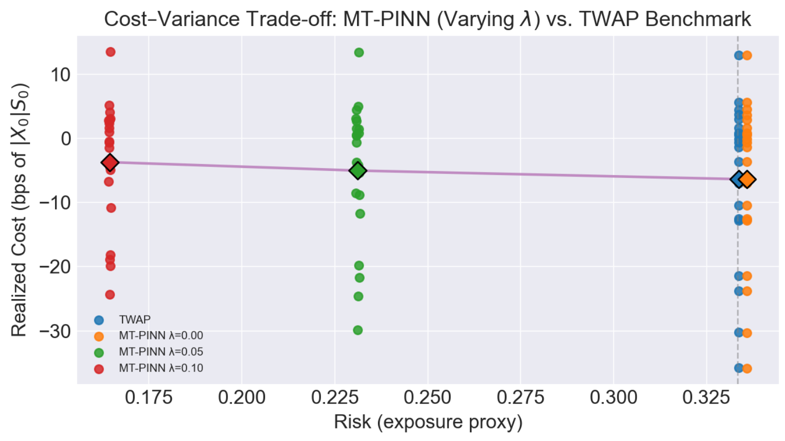


Figure 13: Cost-variance/exposure trade-off of MT-PINN for varying $\lambda \in [0, 0.05, 0.1]$ and TWAP.

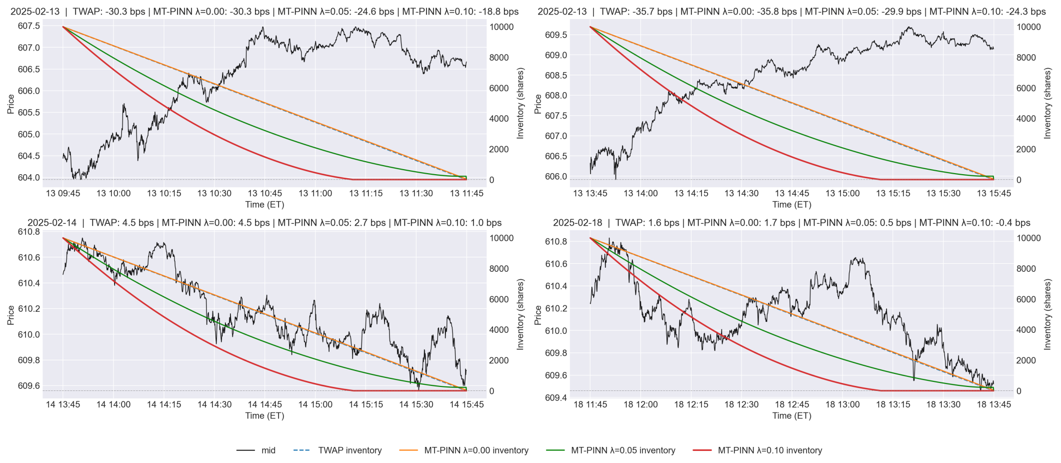
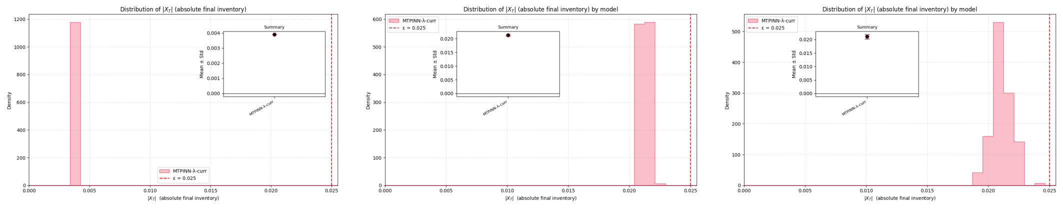


Figure 14: Execution trajectories and costs for MT-PINN with varying $\lambda \in [0, 0.05, 0.1]$ versus TWAP over four sampled trading windows of the SPY ($[W_{10}, W_{12}, W_{15}, W_{17}]$). The top two subplots show rising SPY windows, while the bottom two subplots show falling SPY windows.

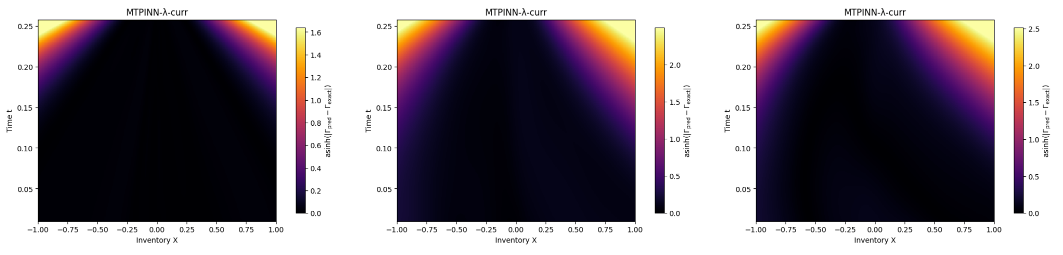
F.2 SPY Real-Market Backtest Results

Figure 13 and Table 2 both show the cost-variance/exposure trade-offs. It is clear that the MT-PINN is behaving well for varying λ in the sense that for higher λ, the model is experiencing less exposure and for $\lambda = 0$ the model behaves very similarly to TWAP, both as expected. For this given SPY



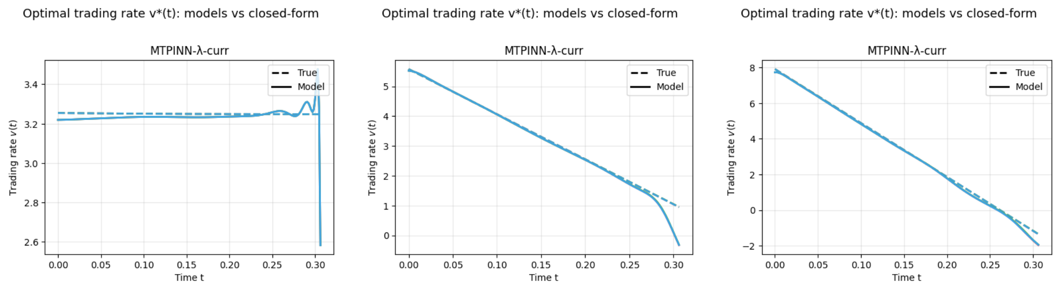
(a) $\lambda = 0$ (b) $\lambda = 0.05$ (c) $\lambda = 0.10$

Figure 15: Distribution of $|X_T|$ (absolute final inventory) for each risk aversion λ. Each panel shows the histogram across rollouts for the MT-PINN model with the tolerance line at $\epsilon = 0.025$ and the inset reporting mean $\pm$std.



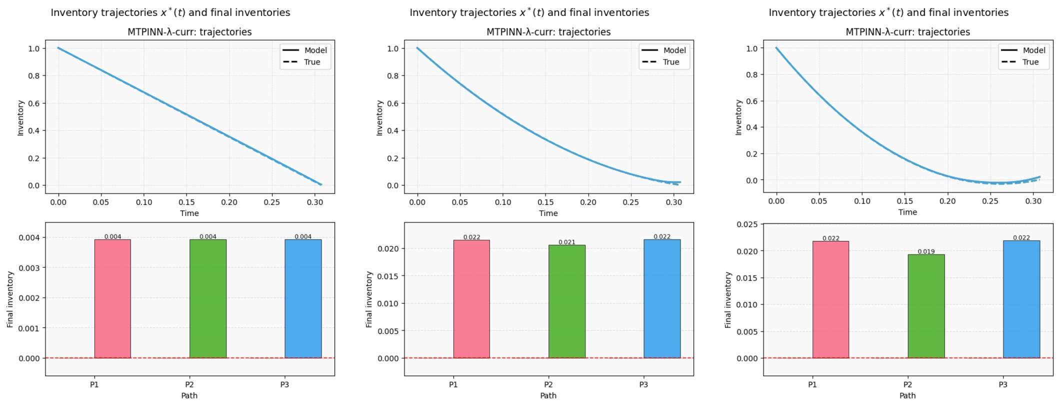
(a) $\lambda = 0$ (b) $\lambda = 0.05$ (c) $\lambda = 0.10$

Figure 16: Heatmaps of domain-wide error over (t, X) for each risk aversion λ with fixed $S = 605$.



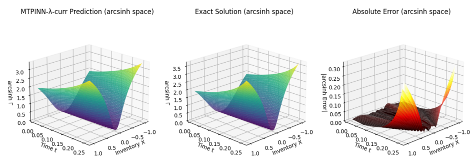
(a) $\lambda = 0$ (b) $\lambda = 0.05$ (c) $\lambda = 0.10$

Figure 17: Optimal trading rate $v^*(t)$: model vs. closed form for each λ.

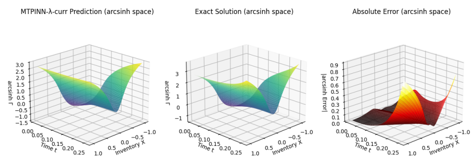


(a) $\lambda = 0$ (b) $\lambda = 0.05$ (c) $\lambda = 0.10$

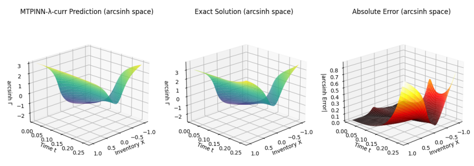
Figure 18: Inventory trajectories $x^*(t)$ and final inventories across paths for each λ.



(a) $\lambda = 0$



(b) $\lambda = 0.05$



(c) $\lambda = 0.10$

Figure 19: Arcsinh-space value surfaces for each λ.

market data, TWAP/risk-neutral MT-PINN experience lower costs, likely as a result of rising SPY market prices across the sampled time windows. This is evidenced in Figure 14, where the execution trajectories and costs for the MT-PINN with different λ and TWAP across four chosen trading windows. The top two time window plots show rising SPY prices and report lowest costs associated with the TWAP algorithm. In contrast, the bottom two time window plots show falling SPY prices and detail lowest costs associated with the highest risk-averse MT-PINN. It is important to mention that this backtest assumed no shorting/over-liquidation and ensures that once zero-inventory is achieved, no subsequent trades are made.

Figure 15 further exemplifies MT-PINN's consistency with reaching near-zero terminal inventory, especially with lower λ values. Figure 16 exhibits the domain-wide inverse hyperbolic sine function absolute value function error ($\operatorname{arcsinh}(|\Gamma_{\text{pred}} - \Gamma_{\text{exact}}|)$) over (t, X) for fixed stock price $S = 605$, showing low error across most of the domain with the exception to high terminal inventory. This is similarly illustrated in the arcsinh space value surfaces in Figure 19. Nonetheless, Figure 17 and 18 illustrates the MT-PINNs ability in closely matching the trading rate and inventory trajectory closed form solution across varying λ. Finally, Figure 20 displays the risk-averse MT-PINNs vs analytical inventory trajectories for three paths, showing the value function along the path and the absolute error over time for the value function, inventory trajectory, and trading rate.

G Application II (LQR): Mathematical Details and Extended Results

This appendix provides the full mathematical formulation, MT-PINN instantiation, and extended experimental results for the high-dimensional Linear–Quadratic Regulator (LQR) problem with a hard terminal constraint studied in Section 5.

G.1 Problem Setup and HJB Formulation

We consider the deterministic LQR

$$\dot{x}(t) = Ax(t) + Bu(t), \qquad x(t) \in \mathbb{R}^d, \tag{27}$$

with quadratic running cost

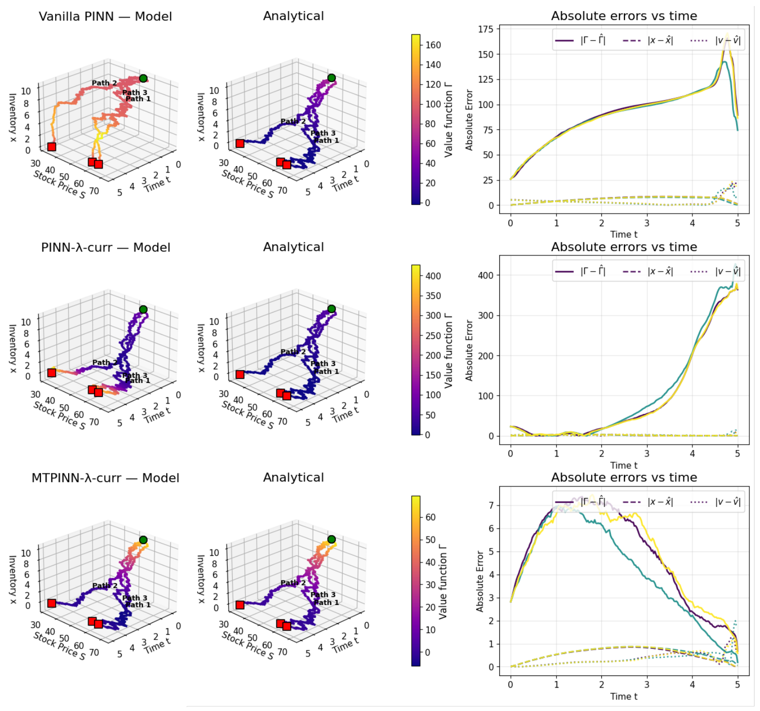
$$\ell(x, u) = x^\top Qx + 2x^\top Su + u^\top Ru, \tag{28}$$

where $Q \succeq 0$ and $R \succ 0$. The horizon is $t \in [0, T]$ and we impose the hard terminal constraint

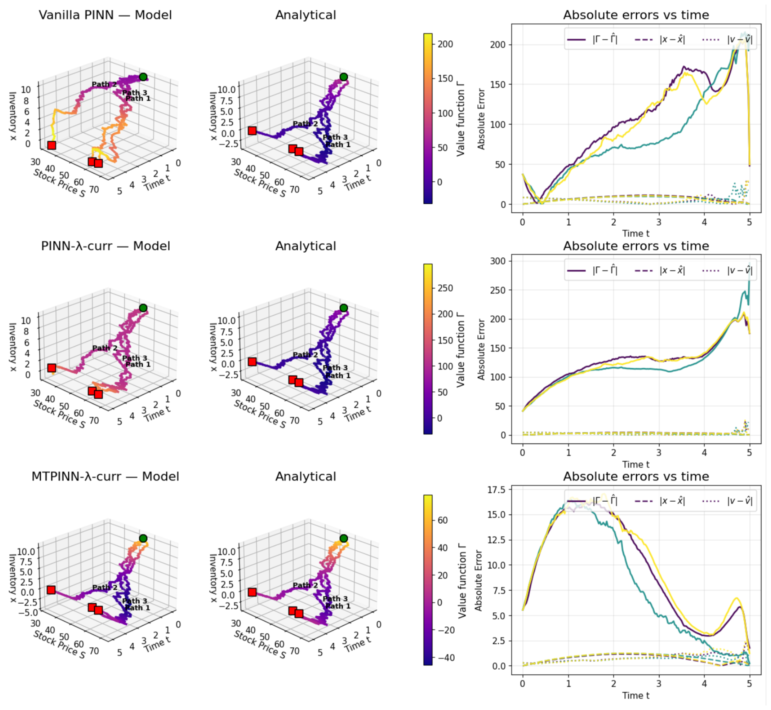
$$x(T) = 0. \tag{29}$$

The associated value function

$$V(t, x) = \inf_u \int_t^T \ell(x(s), u(s))\, ds$$



(a) $\lambda = 0.05$



(b) $\lambda = 0.10$

Figure 20: Pathwise rollouts: model vs. analytic trajectories and absolute error over time for each $\lambda > 0$.

satisfies the Hamilton–Jacobi–Bellman equation

$$\partial_t V + \inf_u \left[x^\top Q x + 2x^\top S u + u^\top R u + \nabla_x V^\top (Ax + Bu) \right] = 0, \tag{30}$$

with the singular terminal condition

$$V(T, x) = \begin{cases} 0, & x = 0, \\ +\infty, & x \neq 0. \end{cases}$$

Minimizing the Hamiltonian yields the optimal feedback

$$u^*(t, x) = -R^{-1} \left(S^\top x + \tfrac{1}{2} B^\top \nabla_x V(t, x) \right), \tag{31}$$

which leads to the reduced HJB

$$\partial_t V + x^\top Q x + \nabla_x V^\top A x - \left(S^\top x + \tfrac{1}{2} B^\top \nabla_x V \right)^\top R^{-1} \left(S^\top x + \tfrac{1}{2} B^\top \nabla_x V \right) = 0. \tag{32}$$

G.2 Closed-Form Riccati Solution

For linear–quadratic structure, the value function admits a quadratic ansatz

$$V(t, x) = x^\top P(t) x,$$

where $P(t)$ is symmetric and diverges as $t \to T$ in order to enforce $x(T) = 0$. Substitution into (32) yields a Riccati differential equation with a singular terminal condition.

Following **?**, the solution can be written in closed form as

$$P(t) = A(t) X(t)^{-1},$$

where $A(t)$ and $X(t)$ are constructed from two continuous-time algebraic Riccati equations (one forward-time and one backward-time). The resulting optimal control is

$$u^*(t, x) = -R^{-1} \left(B^\top A(t) X(t)^{-1} + S^\top \right) x. \tag{33}$$

This solution serves as a ground truth for evaluating value accuracy, control accuracy, and terminal-state enforcement.

G.3 MT-PINN Instantiation for LQR

The MT-PINN approximates the value function $V(t, x)$ with a neural network $\hat{V}_\theta(t, x)$. The feedback control is recovered analytically as

$$u^*_\theta(t, x) = -R^{-1} \left(S^\top x + \tfrac{1}{2} B^\top \nabla_x \hat{V}_\theta(t, x) \right), \tag{34}$$

ensuring consistency between the PDE residual, control policy, and rollout dynamics.

The total loss is

$$\mathcal{L}_{\text{total}} = w_{\text{PDE}} \mathcal{L}_{\text{PDE}} + w_{\text{traj}} \mathcal{L}_{\text{traj}} + w_{\text{0-term}} \mathcal{L}_{\text{0-term}},$$

where:

- $\mathcal{L}_{\text{PDE}}$ penalizes the residual of (32),
- $\mathcal{L}_{\text{traj}}$ penalizes $\|x(T)\|_2$ over rollouts under u^*_θ,
- $\mathcal{L}_{\text{0-term}}$ enforces $\hat{V}_\theta(T, 0) = 0$.

Rollouts are performed using forward Euler discretization, and gradients are propagated via backpropagation-through-time (Appendix B).

G.4 Experimental Configuration

We evaluate a $d = 20$ dimensional LQR. Unless stated otherwise:

- $T = 1.0$, state domain $x \in [-2, 2]^{20}$,
- $N_{\text{PDE}} = 30{,}000$ collocation points,
- $N_{\text{term}} = 1{,}000$ terminal samples,
- MLP with `tanh` activations.

The vanilla PINN baseline replaces $\mathcal{L}_{\text{traj}}$ with a quadratic terminal-value penalty. MT-PINN uses a smaller network but includes trajectory rollouts from batches of perturbed initial states.