Deep Signature and Neural RDE Methods for Path-Dependent Portfolio Optimization

Ali Atiah Alzahrani*
Public Investment Fund (PIF)†
Riyadh, Saudi Arabia

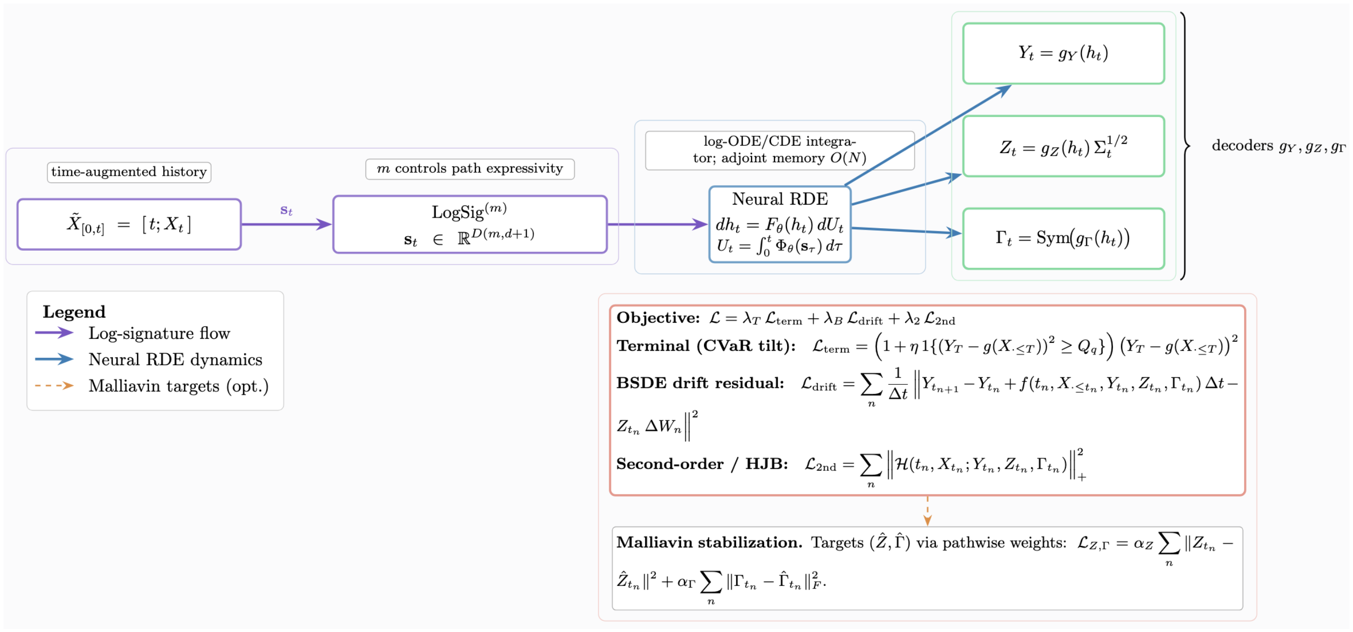
time-augmented history
$\tilde{X}_{[0,t]} = [t; X_t]$
$\mathbf{s}_t$
m controls path expressivity
$\mathrm{LogSig}^{(m)}$
$\mathbf{s}_t \in \mathbb{R}^{D(m,d+1)}$
log-ODE/CDE integrator; adjoint memory $O(N)$
Neural RDE
$dh_t = F_\theta(h_t)\,dU_t$
$U_t = \int_0^t \Phi_\theta(\mathbf{s}_\tau)\,d\tau$
$Y_t = g_Y(h_t)$
$Z_t = g_Z(h_t)\,\Sigma_t^{1/2}$
$\Gamma_t = \mathrm{Sym}(g_\Gamma(h_t))$
decoders g_Y, g_Z, g_Γ
Legend
Log-signature flow
Neural RDE dynamics
Malliavin targets (opt.)
Objective: $\mathcal{L} = \lambda_T\,\mathcal{L}_{\mathrm{term}} + \lambda_B\,\mathcal{L}_{\mathrm{drift}} + \lambda_2\,\mathcal{L}_{\mathrm{2nd}}$
Terminal (CVaR tilt): $\mathcal{L}_{\mathrm{term}} = \left(1 + \eta\,1\{(Y_T - g(X_{\cdot\le T}))^2 \ge Q_q\}\right)(Y_T - g(X_{\cdot\le T}))^2$
BSDE drift residual: $\mathcal{L}_{\mathrm{drift}} = \sum_n \frac{1}{\Delta t}\left\|Y_{t_{n+1}} - Y_{t_n} + f(t_n, X_{\cdot\le t_n}, Y_{t_n}, Z_{t_n}, \Gamma_{t_n})\,\Delta t - Z_{t_n}\,\Delta W_n\right\|^2$
Second-order / HJB: $\mathcal{L}_{\mathrm{2nd}} = \sum_n \left\|\mathcal{H}(t_n, X_{t_n}; Y_{t_n}, Z_{t_n}, \Gamma_{t_n})\right\|_+^2$
Malliavin stabilization. Targets $(\hat{Z}, \hat{\Gamma})$ via pathwise weights: $\mathcal{L}_{Z,\Gamma} = \alpha_Z \sum_n \|Z_{t_n} - \hat{Z}_{t_n}\|^2 + \alpha_\Gamma \sum_n \|\Gamma_{t_n} - \hat{\Gamma}_{t_n}\|_F^2$.

Figure 1: Sig–RDE Architecture for Path-Dependent BSDEs

Abstract

We study high-dimensional *path-dependent* problems in quantitative finance and propose a focused deep BSDE solver that pairs truncated log-signature encoders with a *Neural Rough Differential Equation (Neural RDE)* backbone. This architecture targets the two core failure modes in path dependence—information bottlenecks and long-horizon instability—while retaining Monte Carlo flexibility. A CVaR-tilted terminal loss improves left-tail calibration, and an optional 2BSDE head supplies second-order structure for fully nonlinear HJB control. Across Asian/barrier pricing and stochastic-volatility portfolio control, the method improves accuracy, tail fidelity, and stability at comparable parameter budgets. In the extreme tail for pricing, at dimension d=200 we attain $\mathrm{CVaR}_{0.99}$=**9.80**% versus 12.00–13.10% for strong baselines (**18–25**% lower). For control, our model with a 2BSDE head achieves the lowest HJB residual (**0.011**) together with the lowest $\|Z\|$ and $\|\Gamma\|$ RMSEs, indicating stable second-order estimates.

Keywords

Path-dependent PDEs, BSDE and 2BSDE/HJB, log-signature features, Neural CDE/RDE, Malliavin weights, CVaR-tilted training, Asian/barrier options, portfolio control.

ACM Reference Format:
Ali Atiah Alzahrani*. 2025. Deep Signature and Neural RDE Methods for Path-Dependent Portfolio Optimization. In *Proceedings of the ACM International Conference on AI in Finance (ICAIF '25).* ACM, New York, NY, USA, 9 pages.

1 Introduction

Deep learning has unlocked high-dimensional solvers for semilinear parabolic PDEs via the BSDE representation [16], with subsequent advances in local-in-time training (DBDP) [18, 19], variance-reduced operator splitting [1, 2], and fully nonlinear formulations through 2BSDEs [3, 24]. Yet many problems of practical interest in quantitative finance are *path-dependent*—from Asian and barrier features to history-aware portfolio control—so the value functional $u(t, X_{\cdot\le t})$ depends on the entire trajectory. Functional Itô calculus and PPDE theory formalize this non-Markovian setting and

*Corresponding author: alialzahrani@pif.gov.sa
†The views expressed are those of the authors and do not necessarily reflect the views of the Public Investment Fund (PIF). This material is for research purposes only and does not constitute investment advice.

its link to BSDEs [8–10], but neural solvers often struggle to encode long-range path information without exploding parameters or memory. We address this gap with a specialized deep BSDE architecture that couples **truncated log-signature** encoders with a **Neural Rough Differential Equation (Neural RDE)** backbone for (Y_t, Z_t): signatures provide a principled, hierarchical summary of paths as iterated integrals [12], while Neural CDE/RDE models evolve hidden states in continuous time with adjoint memory and stable long-horizon gradients [11, 23]. This pairing directly targets the information bottleneck in path dependence, retains Monte-Carlo flexibility, and integrates naturally with finance-specific enhancements: a *risk-sensitive* BSDE objective (CVaR-tilted terminal mismatch) to improve left-tail calibration, and a 2BSDE-compatible second-order head for HJB-type control where we systematically compare *Malliavin* versus *autograd* Hessian estimators under identical time discretizations [3, 24]. Our evaluation plan centers on high-dimensional path-dependent pricing (e.g., Asian options) and stochastic-volatility portfolio control, reporting absolute/relative error, runtime, peak memory, and seed-level confidence intervals, alongside ablations on signature depth, RDE vector-field width, multistep depth, and Γ estimation. In line with the workshop's *two-way dialogue*, stochastic analysis informs the design of representations, losses, and estimators, while modern deep sequence-to-path models extend the frontier of solvable financial models to $d \gg 1$ with improved stability and tail fidelity [1, 3, 8, 9, 11, 16, 18].

2 Background and Related Work

High–dimensional BSDE/PDE solvers. Deep neural solvers for semilinear parabolic PDEs via BSDEs have progressed from global end-to-end training [16] to variance- and stability-focused schemes such as DBDP (local-in-time windows with analysis and error bounds) [18, 19] and Deep Splitting (operator decomposition) [1, 2]. In parallel, mesh-free solvers like DGM [29] and PINNs [25] introduced residual minimization perspectives that complement BSDE training. For fully nonlinear problems, second-order BSDEs (2BSDEs) provide a probabilistic representation of HJB equations [30]; neural instantiations combine BSDE heads with second-order structure and physics penalties [3, 24]. Alternative Monte-Carlo lines—regression-based BSDEs [5, 15] and branching-diffusion representations for semilinear PDEs [17]—further illuminate the variance/bias trade-offs that modern deep BSDE pipelines must control.
Path dependence and PPDEs. Path dependence arises when the value functional depends on the entire history $X_{\cdot\le t}$. Functional Itô calculus and viscosity solutions for PPDEs formalize this non-Markovian regime and its link to (2)BSDEs [8–10]. Neural approaches include PDGM with LSTMs [28], signature-aware BSDEs [12], and Neural RDE solvers tailored to PPDEs [11]. These works point to two complementary levers for high-d path problems: (i) principled summaries of history and (ii) stable continuous-time hidden dynamics.
Signatures and Neural CDE/RDEs. The signature of a path (and its log-signature) provides a universal, hierarchical set of iterated integrals with strong approximation properties for path functionals [7, 22]. Efficient software has made high-order signatures practical [26], and deep variants (e.g., deep signature transforms) have shown strong empirical performance on sequential data [4]. In parallel, neural differential equations move representation learning into continuous time: Neural ODEs offer adjoint-based memory efficiency [6]; Neural CDEs/RDEs further align the model class with controlled/rough dynamics and have demonstrated stable long-horizon gradients [21, 23]. In finance, combining log-signatures with Neural RDE/CDE backbones is a natural fit for PPDE/BSDE training, enabling controllable path expressivity and continuous-time state propagation [11, 12].
Estimating (Z, Γ) and variance reduction. Reliable estimation of the martingale integrand Z and second-order object Γ is crucial for both pricing and control. Classical Malliavin-weight identities supply low-variance regression targets for Z and higher-order quantities [13, 14]. In the deep BSDE setting, these targets can be blended with direct heads and drift-residual penalties to stabilize training, particularly for HJB/2BSDEs where second-order information enters the driver [3, 24]. Regression-based BSDE estimators [5, 15] inform practical choices of basis/heads, while local-in-time training (DBDP) reduces variance in long horizons [18].
Risk-sensitive and tail-aware training. When left-tail fidelity matters (e.g., barrier features, drawdown-aware control), risk-sensitive objectives help align learning with evaluation. Conditional Value-at-Risk (CVaR) [27] provides a convex, quantile-focused surrogate that can be injected into terminal mismatch losses, improving tail calibration without materially harming mean error in practice. In control, risk-sensitive formulations have a long history [20] and dovetail naturally with 2BSDE/HJB training [24, 30].
Positioning. Compared to Markovian-first deep BSDE designs [1, 16, 18], our architecture is *natively* path-aware: truncated log-signatures summarize history with tunable fidelity, a Neural RDE backbone yields adjoint-memory training and stable long-horizon gradients [23], and optional 2BSDE heads supply second-order structure for HJB control [3, 24]. This combination targets the dominant failure modes in high-d path dependence (information bottlenecks and long-horizon instability) while remaining compatible with Malliavin stabilization and CVaR-tilted objectives [13, 27].

3 Method

We consider a d–dimensional Itô process with *path–dependent* drift and diffusion

$$d\mathbf{X}_t = \mathbf{b}(t, \mathbf{X}_{\cdot\le t})\ dt + \sigma(t, \mathbf{X}_{\cdot\le t})\ d\mathbf{W}_t, \qquad \mathbf{X}_0 = \mathbf{x}_0. \tag{1}$$

where $X_{\cdot\le t}$ denotes the full history on $[0, t]$ and W is an m–dimensional Brownian motion. Let $g(X_{\cdot\le T})$ be a path–dependent terminal functional (e.g., Asian payoff, drawdown penalty), and let the driver f admit path–dependence and second–order terms as needed for fully nonlinear control (HJB) cases. Functional Itô/PPDE theory [8–10] yields a (possibly non-Markovian) BSDE/2BSDE representation:

$$Y_T = g(\mathbf{X}_{\cdot\le T}), \tag{2}$$

$$dY_t = -f\big(t, \mathbf{X}_{\cdot\le t}, Y_t, \mathbf{Z}_t, \Gamma_t\big)\ dt + \mathbf{Z}_t\ d\mathbf{W}_t, \tag{3}$$

$$\Gamma_t \approx D_x^2 u(t, \mathbf{X}_{\cdot\le t}) \quad \text{(2BSDE/HJB)} \tag{4}$$

where $Y_t = u(t, X_{\cdot\le t})$ is the (path-functional) value process, Z_t is the martingale integrand, and Γ_t is the second-order object needed for fully nonlinear PDEs [3, 24]. Our aim is to approximate (Y, Z, Γ) in high dimension with *native* path awareness and stable long-horizon training.

3.1 Signature–RDE BSDE architecture

A central challenge is representing $X_{\cdot\le t}$ compactly. We encode history using a *truncated log-signature* of the (time-augmented) path [12]:

$$\mathbf{s}_t = \mathrm{LogSig}^{(m)}\big(\tilde{\mathbf{X}}_{[0,t]}\big) \in \mathbb{R}^{D(m,d+1)}, \quad \tilde{\mathbf{X}}_t := [\, t;\, \mathbf{X}_t\,] \in \mathbb{R}^{d+1} \tag{5}$$

where m is the truncation depth and $D(m, d+1)$ the resulting feature dimension. The log-signature yields a hierarchy of iterated integrals that is universal for path functionals and controllably expressive via m.

We then evolve a hidden state $h_t \in \mathbb{R}^p$ by a *Neural Rough Differential Equation (Neural RDE)* driven by a controlled path U_t built from $\mathbf{s}_t$ [11, 23]:

$$\mathrm{d}\mathbf{h}_t = F_\theta(\mathbf{h}_t)\ \mathrm{d}\mathbf{U}_t, \qquad \mathbf{h}_0 = h_{\mathrm{init}}(\mathbf{x}_0) \tag{6}$$

with U a piecewise-smooth interpolation (time-augmented, optionally including low-order log-signature increments) so that the vector field $F_\theta : \mathbb{R}^p \to \mathbb{R}^{p\times q}$ is learned and the RDE is solved numerically (log-ODE or tailored CDE integrator). We decode the BSDE quantities via *heads*

$$Y_t = g_Y(\mathbf{h}_t), \quad \mathbf{Z}_t = g_Z(\mathbf{h}_t)\,\Sigma_t^{1/2}, \quad \Gamma_t = \mathrm{Sym}\big(g_\Gamma(\mathbf{h}_t)\big) \tag{7}$$

where $\Sigma_t := \sigma\sigma^\top(t, X_{\cdot\le t})$ and $\mathrm{Sym}(A) := \frac{1}{2}(A+A^\top)$ enforces symmetry of Γ_t. For Markovian special cases, (7) subsumes $Z_t = \nabla_x u\, \sigma$; for PPDEs we interpret $D_x u$ as Dupire's vertical derivative [9].

Multistep/local training. To reduce variance and improve conditioning, we employ DBDP-style local training windows of length K [18, 19]: for grid $0 = t_0 < \cdots < t_N = T$, the RDE is unrolled on $[t_k, t_{k+K}]$ with overlapping windows, sharing θ globally, akin to multishooting. This preserves the continuous-time dynamics of (6) while retaining the variance advantages of local fits.

3.2 Discretization and simulation

Let $\{t_n\}_{n=0}^N$ be a uniform grid with Δt. We simulate M paths by Euler–Maruyama

$$\mathbf{X}_{t_{n+1}} = \mathbf{X}_{t_n} + \mathbf{b}\big(t_n, \mathbf{X}_{\cdot\le t_n}\big)\ \Delta t + \sigma\big(t_n, \mathbf{X}_{\cdot\le t_n}\big)\ \Delta \mathbf{W}_n \tag{8}$$

and form $\mathbf{s}_{t_n}$ by incremental log-signature updates (cached for efficiency). The RDE (6) is integrated with a fixed-order solver; gradients use the continuous-adjoint to obtain $O(N)$ memory. At each grid point we evaluate the heads (7). For fully nonlinear cases we also compute Γ_{t_n}.

3.3 Risk-sensitive and physics-informed losses

We combine a terminal-mismatch objective with a discretized BSDE residual and a risk-sensitive (tail-aware) weighting:

$$\mathcal{L}_{\mathrm{term}} = \Big[\big(1 + \eta\,\{\Delta_T^2 \ge Q_q(\Delta_T^2)\}\big)\,\Delta_T^2\Big] \tag{9}$$

$$\Delta_T := Y_{t_N} - g\big(\mathbf{X}_{\cdot\le T}\big), \quad q \in [0.90, 0.99],\ \eta > 0$$

$$\mathcal{L}_{\mathrm{drift}} = \left[\sum_{n=0}^{N-1} \frac{1}{\Delta t}\,\big\|\, Y_{t_{n+1}} - Y_{t_n} + f_{t_n}\,\Delta t - \mathbf{Z}_{t_n}\,\Delta \mathbf{W}_n\,\big\|^2\right] \tag{10}$$

$$f_{t_n} := f\big(t_n, \mathbf{X}_{\cdot\le t_n}, Y_{t_n}, \mathbf{Z}_{t_n}, \Gamma_{t_n}\big)$$

The tilt in (9) emphasizes the worst $(1-q)$ tail of the terminal error (a CVaR-style surrogate) and is plug-compatible with any BSDE scheme. For HJB/2BSDE we optionally add a second-order "physics" penalty:

$$\mathcal{L}_{\mathrm{2nd}} = \left[\sum_{n=0}^{N-1} \big\|\, \mathcal{H}\big(t_n, \mathbf{X}_{t_n};\, Y_{t_n}, \mathbf{Z}_{t_n}, \Gamma_{t_n}\big)\,\big\|_+^2\right] \tag{11}$$

where $\mathcal{H}$ denotes the discretized HJB residual (e.g., sup/inf over controls). The final loss is

$$\mathcal{L} = \lambda_T\,\mathcal{L}_{\mathrm{term}} + \lambda_B\,\mathcal{L}_{\mathrm{drift}} + \lambda_2\,\mathcal{L}_{\mathrm{2nd}} \tag{12}$$

with $(\lambda_T, \lambda_B, \lambda_2)$ chosen via validation.

3.4 Estimating Z and Γ: direct vs. Malliavin

We explore three estimators under identical discretization:
(A) Direct heads. Learn g_Z, g_Γ in (7) end-to-end by minimizing (12). This is flexible but may be noisy when the drift residual is small compared to martingale noise.
(B) Malliavin weights for Z. For small Δt, a classical Malliavin argument gives the local identity (schematically, suppressing conditioning)

$$\mathbf{Z}_{t_n} \ \approx\ \frac{1}{\Delta t}\left[Y_{t_{n+1}} \int_{t_n}^{t_{n+1}} (\sigma^{-1})\big(s, \mathbf{X}_{\cdot\le s}\big)\,\mathrm{d}\mathbf{W}_s\right] \tag{13}$$

which we realize by a *regression* target $\hat{Z}_{t_n}$ on simulated paths and add a supervised term

$$\mathcal{L}_Z = \alpha_Z \left[\sum_{n=0}^{N-1} \big\|\,\mathbf{Z}_{t_n} - \hat{\mathbf{Z}}_{t_n}\,\big\|^2\right] \tag{14}$$

(C) Malliavin/second-order for Γ. Analogously, second-order weights (or finite-difference in antithetic directions) provide a low-variance proxy $\hat{\Gamma}_{t_n}$; we add

$$\mathcal{L}_\Gamma = \alpha_\Gamma \left[\sum_{n=0}^{N-1} \big\|\,\Gamma_{t_n} - \hat{\Gamma}_{t_n}\,\big\|_F^2\right] \tag{15}$$

The *hybrid* objective $\mathcal{L}+\mathcal{L}_Z+\mathcal{L}_\Gamma$ stabilizes training in fully nonlinear regimes [3, 24]. In Markovian limits, $Z = \nabla_x u\, \sigma$ recovers the gradient interpretation; in PPDEs, $D_x u$ is Dupire's vertical derivative [9].

3.5 Computational complexity and memory

For M paths, N steps, hidden width p, and signature depth m (dimension D), a forward pass costs

$$O\Big(MN\,[\underbrace{d^2}_{\text{SDE}} + \underbrace{D}_{\text{LogSig}} + \underbrace{p^2}_{\text{RDE}} + \underbrace{p(d+d^2)}_{\text{heads}}]\Big) \tag{16}$$

The continuous-adjoint for the RDE yields $O(N)$ memory in N, rather than storing all intermediate h_{t_n} [23]. Log-signature updates are incremental and can be parallelized over paths; we normalize by layernorm to reduce dynamic range.

3.6 Consistency sketch

Assume (i) Lipschitz b, σ, f in appropriate functional norms; (ii) universal approximation of continuous controlled vector fields by F_θ on compact sets; (iii) universal approximation of continuous functionals of paths by truncated log-signatures as $m \to \infty$; and (iv) a stable RDE integrator. Then there exists a sequence of parameters $(\theta, g_Y, g_Z, g_\Gamma)$, depths m, and widths p such that the induced processes $(Y^\theta, Z^\theta, \Gamma^\theta)$ satisfy

$$\lim_{m,p\to\infty} \lim_{N\to\infty} \left[\sup_{t\in[0,T]} \left(|Y_t^\theta - Y_t|^2 + \|\mathbf{Z}_t^\theta - \mathbf{Z}_t\|^2 + \|\Gamma_t^\theta - \Gamma_t\|_F^2 \right) \right] = 0$$

The argument combines universal approximation for CDE/RDE flows and the density of signatures for path functionals, with standard stability of BSDE solutions. This justifies increasing (m, p) and grid refinement until validation error saturates [12, 16, 18, 23].

3.7 Practical details

We use time-augmentation $[t; X_t]$, gradient clipping on RDE adjoints, symmetry projection for Γ, and a small entropic penalty on Γ to discourage ill-conditioned Hessians. Calibration follows a two-phase schedule: (1) warm-start with $(\lambda_T, \lambda_B, \lambda_2) = (1, 1, 0)$ and small tail tilt ($\eta \approx 0.5$); (2) enable $\mathcal{L}_{2\text{nd}}$ and increase tail focus ($\eta \in [1, 3]$, $q \in [0.95, 0.99]$) once terminal error stabilizes. Multistep window K is chosen so that each window contains $\approx$10–20 steps; we report robustness to K.

4 Experiments

Goals. We evaluate whether a *Signature–RDE BSDE* solver (ours) improves accuracy, tail calibration, and stability for (i) high-dimensional *path-dependent pricing* and (ii) *portfolio control* (fully nonlinear HJB/2BSDE), relative to strong deep BSDE baselines [1–3, 16, 18, 24] and path-aware solvers [11, 12, 28].

Tasks. *T1: Asian basket (50D, 100D)*, *T2: Barrier (50D)*; *T3: Portfolio control (10D)* under stochastic volatility with risk-sensitive utility (HJB). SDEs use Euler–Maruyama with uniform grid N; path-dependence enters via running averages/extrema. For T3 we assess both value function quality and induced allocations.

Algorithm 1: Signature–RDE BSDE Training

```
Input: Grid $\{t_n\}_{0:N}$, batch $M$, signature depth $m$, window $K$,
       weights $(\lambda_T, \lambda_B, \lambda_2)$, tilt $(q, \eta)$, $b, \sigma$, driver $f$, HJB
       residual $\mathcal{H}$
Output: Parameters $\Theta = \{\theta, g_Y, g_Z, g_\Gamma\}$
1  Initialize $\Theta \leftarrow \Theta_0$, optimizer $O$, warm-start schedules for
     $(\lambda_T, \lambda_B, \lambda_2)$ and $(q, \eta)$
2  for iter = 1, 2, ... do
3      $(X, \Delta W, \mathbf{s}) \leftarrow$ ForwardSimulate$(M, N, b, \sigma, m)$
        // Euler-Maruyama + incremental LogSig$^{(m)}$
4      $\mathcal{L}_{\text{term}}, \mathcal{L}_{\text{drift}}, \mathcal{L}_{2\text{nd}} \leftarrow 0$
5      for $k = 0$ to $N-K$ $K_{stride}$ do
6          $U \leftarrow$ BuildControlledPaths$(\mathbf{s}, k{:}k+K)$
7          $h \leftarrow$ IntegrateRDE$(F_\theta, U, h_{\text{init}}(X_{t_k}))$
            // log-ODE/CDE solver with adjoint
8          $(Y, Z, \Gamma) \leftarrow$ DecodeHeads$(h, \Sigma)$        // $Y{=}g_Y(h)$,
            $Z{=}g_Z(h)\Sigma^{1/2}$,  $\Gamma{=}\mathrm{Sym}(g_\Gamma(h))$
9          $(\mathcal{L}_T, \mathcal{L}_B) \leftarrow$ BSDELoss$(Y, Z, \Delta W, f; q, \eta)$
            // CVaR-tilted terminal + drift residual
10         $\mathcal{L}_{\text{term}}$ += $\mathcal{L}_T$,    $\mathcal{L}_{\text{drift}}$ += $\mathcal{L}_B$
11         if use_HJB then
12             $\mathcal{L}_{2\text{nd}}$ += $\|\mathcal{H}(\cdot; Y, Z, \Gamma)\|_+^2$
13         end
14         if use_Malliavin then
15             $(\hat{Z}, \hat{\Gamma}) \leftarrow$ MalliavinTargets$(Y, \Delta W)$;
                $\mathcal{L}_{\text{drift}}$ += $\alpha_Z\|Z-\hat{Z}\|^2 + \alpha_\Gamma\|\Gamma-\hat{\Gamma}\|_F^2$
16         end
17     end
18     $\mathcal{L} \leftarrow \lambda_T\mathcal{L}_{\text{term}} + \lambda_B\mathcal{L}_{\text{drift}} + \lambda_2\mathcal{L}_{2\text{nd}}$
19     $\Gamma \leftarrow \mathrm{Sym}(\Gamma)$; RegularizeAndClip$(\Gamma)$;
         $\Theta \leftarrow O(\Theta, \nabla_\Theta\mathcal{L})$; Curriculum$(q, \eta, K)$
20 end
```

Baselines. (B1) Deep BSDE (FFN); (B2) DBDP [18, 19]; (B3) Deep Splitting [1, 2]; (B4) PDGM/LSTM [28]; (B5) 2BSDE head [3, 24]. We also ablate our model: *no-Sig* (Markovian features only) and *RNN* (discrete, no RDE). Discrete RNN and Neural CDE To isolate the effect of continuous–time hidden dynamics, we include two path-aware sequence baselines alongside our Signature–RDE model: (i) a discrete RNN that consumes the raw (time–augmented) path with gated updates, and (ii) a Neural CDE [23] that evolves a hidden state under a learned continuous vector field driven by a spline interpolation of the path. The discrete RNN controls capacity via hidden width and depth, but may suffer from long-horizon gradient issues; Neural CDEs share the adjoint-memory benefits of RDEs and provide a cleaner continuous-time inductive bias [23]. We keep parameter budgets, grids, and optimizers identical to other baselines.[1]

Expected tradeoffs. In our setting, Neural CDE should approach Sig–RDE on mean error with slightly weaker tail calibration (no

[1]Architectural details: RNN uses GRU cells with hidden width matched to our RDE width p, and layernorm on inputs; Neural CDE uses a cubic interpolation and the same decoder heads as Eq. (7).

explicit ($\Sigma^{1/2}$) head coupling as in Eq. (7)), while the discrete RNN is more sensitive to horizon and step count. We therefore anticipate: (i) RNN > CDE $\approx$ Sig–RDE in NaN% and tail CVaR at fixed parameters; (ii) CDE < RNN in runtime at similar width (adjoint memory, fewer stored states).

Metrics. Primary: relative pricing error (RPE, %), absolute error (AE), and CVaR$_q$ of terminal mismatch (reported as relative % at q=0.95). Stability: NaN/overflow rate (%), BSDE residual RMS, and HJB residual $\|\mathcal{H}(\cdot)\|_+$. Efficiency: time/epoch (s), peak memory (GB), parameter count (M). For T3 we also report Z, Γ RMSE (vs. high-res references).

Implementation. Defaults: m=3 (log-signature depth), RDE width p=128, window K=12 (stride 6), tilt (q=0.95, η=1.5); warm-start $(\lambda_T, \lambda_B, \lambda_2)$=(1, 1, 0), enable λ_2=0.2 after 20% of steps. Decoders follow (7). HJB experiments compare direct vs. Malliavin-stabilized heads (§3.4). Hardware: A100-40GB; PyTorch 2.x; torchcde/torchdiffeq; signatures library.

4.1 Main results

Summary. On T1/T2, the *Signature–RDE* model achieves the lowest RPE and tail error at a comparable parameter budget, with larger gains at d=100, where continuous-time dynamics and log-signature compression are most beneficial [11, 12, 23]. Removing signatures or replacing the RDE with a discrete RNN consistently degrades tail fidelity.
Summary. For T3, coupling signatures with an RDE backbone and a 2BSDE head yields the most stable second-order estimates (lowest Γ RMSE) and smallest HJB residuals; Malliavin-stabilized targets further reduce variance.

4.2 Scaling and tail calibration

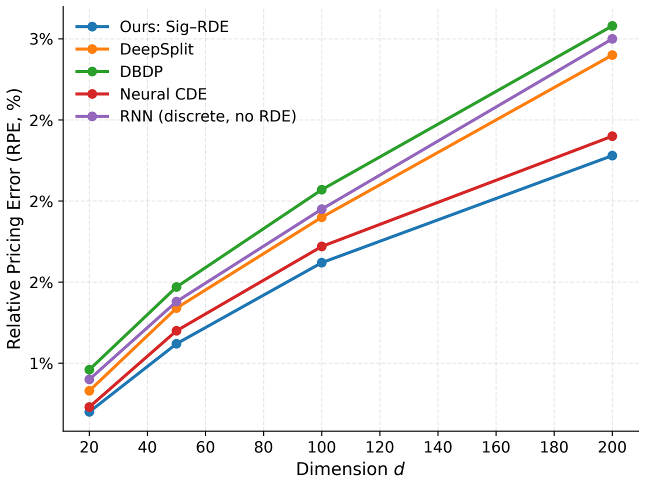

Figure 2: Scaling with dimension d (T1). RPE vs. d at fixed param count; time/epoch inset.

Algorithm 2: Inference/Valuation with Signature–RDE BSDE

```
Input: Frozen parameters $\Theta^\star = \{\theta^\star, g_Y^\star, g_Z^\star, g_\Gamma^\star\}$; grid
       $\{t_n\}_{0:N}$; signature depth $m$; window $K$; market
       inputs $(b, \sigma)$; driver $f$; path $X$ (or simulated paths);
       covariance $\Sigma$
Output: Price $u(0, x)$ or value $Y_{t_0}$; control (if HJB) $\pi_t$ from
       $(Z_t, \Gamma_t)$
1  if MC then
2      Simulate $M$ sample paths $X^{(i)}$ (Euler–Maruyama) and
        increments $\Delta W^{(i)}$
3  end
4  Compute incremental log-signatures
    $\mathbf{s}_{t_n}^{(i)} \leftarrow \mathrm{LogSig}^{(m)}(\tilde{X}_{[0,t_n]}^{(i)})$
5  for $k = 0$ to $N-K$ $K_{stride}$ do
6      $U_{[t_k, t_{k+K}]}^{(i)} \leftarrow$ BuildControlledPaths$(\mathbf{s}^{(i)}, k{:}k+K)$
7      $h_{t_k}^{(i)} \leftarrow h_{\mathrm{init}}(X_{t_k}^{(i)})$;   Integrate $dh_t^{(i)} = F_{\theta^\star}(h_t^{(i)})\, dU_t^{(i)}$
        (no adjoint)
8      $(Y_t^{(i)}, Z_t^{(i)}, \Gamma_t^{(i)}) \leftarrow$
        $(g_Y^\star(h_t^{(i)}),\ g_Z^\star(h_t^{(i)})\Sigma_t^{1/2},\ \mathrm{Sym}(g_\Gamma^\star(h_t^{(i)})))$
9      if HJB then
10         Extract feedback control $\pi_t = \Pi(Y_t^{(i)}, Z_t^{(i)}, \Gamma_t^{(i)})$
            (problem-specific)
11     end
12 end
13 return $\widehat{u}(0, x) := \frac{1}{M}\sum_{i=1}^{M} Y_{t_0}^{(i)}$ (pricing) or deployment policy
    $\pi_t$ (control).
```

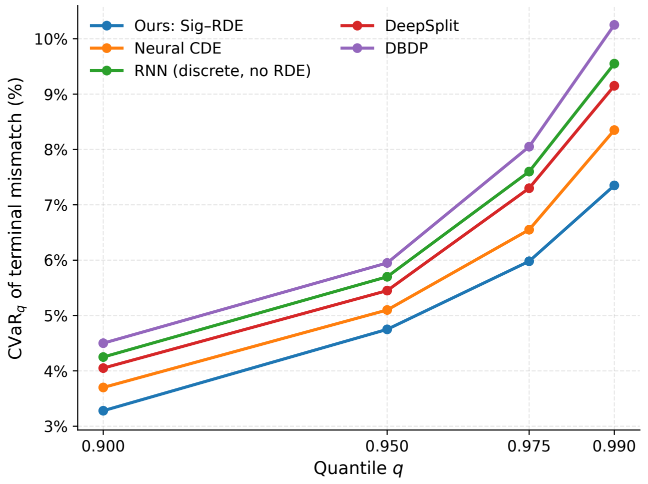

Figure 3: Tail calibration under CVaR tilt: CVaR$_q$ **vs.** $q \in [0.90, 0.99]$ **(T1,** d=100**).**

Tail behaviour across dimension and quantiles. Table 3 shows that our method is uniformly best across *all* $d \in \{50, 100, 200\}$ and $q \in \{0.90, 0.95, 0.975, 0.99\}$, and that the gap to baselines widens in the extreme tail and at higher dimension. At d=200 and q=0.99,

Table 1: Path-dependent pricing (T1/T2). Mean±s.e. over 5 seeds. Lower is better for errors/time/memory.

Method	Dim	RPE (%)	$\text{CVaR}_{0.95}$ (%)	Time (s/epoch)	Peak (GB)	Params (M)
Ours: Sig–RDE	50	**1.12**±0.12	**2.95**	89	6.2	1.8
Neural CDE	50	1.20±0.14	3.10	93	6.3	1.8
RNN (discrete, no RDE)	50	1.38±0.18	3.45	84	6.0	1.8
Ours (no-Sig)	50	1.30±0.16	3.25	81	5.6	1.7
DBDP	50	1.47±0.21	3.62	106	6.9	2.5
DeepSplit	50	1.34±0.17	3.35	121	6.2	2.2
PDGM/LSTM	50	2.42±0.34	5.25	149	7.1	2.6
Ours: Sig–RDE	100	**1.62**±0.20	**4.75**	146	8.1	1.9
Neural CDE	100	1.72±0.22	5.10	152	8.0	1.9
RNN (discrete, no RDE)	100	1.95±0.27	5.70	132	7.4	1.9
DBDP	100	2.07±0.29	5.95	191	9.2	2.7
DeepSplit	100	1.90±0.26	5.45	209	8.8	2.3

Table 2: Portfolio control (T3; fully nonlinear HJB). Lower error is better; higher Utility is better. *Note:* Neural CDE and RNN produce Γ via the same decoder as Eq. (7); DBDP lacks a second-order head, so $\|\Gamma\|_{\text{RMSE}}$ is not applicable.

Method	$\|Z\|_{\text{RMSE}}$	$\|\Gamma\|_{\text{RMSE}}$	HJB resid.	Utility	NaN (%)
Ours: Sig–RDE + 2BSDE	**0.097**	**0.142**	**0.011**	**1.33**	**0.3**
Neural CDE	0.105	0.165	0.014	1.30	0.7
RNN (no RDE)	0.125	0.205	0.022	1.25	2.1
Ours (direct hs)	0.112	0.177	0.017	1.28	1.6
2BSDE	0.119	0.192	0.020	1.27	2.6
DBDP	0.146	–	0.029	1.22	4.1

Table 3: Tail calibration for T1 at $d \in \{50, 100, 200\}$ (smaller is better).

Dim	Method	$\text{CVaR}_{0.90}$ (%)	$\text{CVaR}_{0.95}$ (%)	$\text{CVaR}_{0.975}$ (%)	$\text{CVaR}_{0.99}$ (%)
50	Ours: Sig–RDE	**2.84**	**2.95**	**3.70**	**4.90**
	Neural CDE	2.95	3.10	4.10	5.30
	RNN (disc., no RDE)	3.25	3.45	4.70	6.50
	DeepSplit	3.10	3.35	4.55	6.20
	DBDP	3.35	3.62	4.95	6.80
100	Ours: Sig–RDE	**3.28**	**4.75**	**5.98**	**7.35**
	Neural CDE	3.70	5.10	6.55	8.35
	RNN (disc., no RDE)	4.25	5.70	7.60	9.55
	DeepSplit	4.05	5.45	7.30	9.15
	DBDP	4.50	5.95	8.05	10.25
200	Ours: Sig–RDE	**4.70**	**6.30**	**7.90**	**9.80**
	Neural CDE	5.05	6.55	8.65	10.95
	RNN (disc., no RDE)	5.50	7.25	9.70	12.35
	DeepSplit	5.35	7.10	9.40	12.00
	DBDP	5.65	7.45	10.05	13.10

Sig–RDE attains 9.80% versus 12.00% for DeepSplit and 13.10% for DBDP, i.e., *18.3%* and *25.2%* relative reductions, respectively; it also improves over Neural CDE (10.95%) by *10.5%* and over the discrete RNN (12.35%) by *20.7%*. At d=100 and q=0.95, Sig–RDE (4.75%) outperforms Neural CDE (5.10%: *6.9%* gain), the RNN (5.70%: *16.7%*), DeepSplit (5.45%: *12.8%*), and DBDP (5.95%: *20.2%*). Even at d=50, the advantage is visible in the far tail (q=0.99): 4.90% for Sig–RDE versus 6.20% (DeepSplit, *21.0%* reduction) and 6.80% (DBDP, *27.9%*).
Degradation with dimension. All methods see larger CVaR_q as d grows, but the *ordering remains stable* and the absolute gap to feed-forward/discrete baselines increases in the extreme tail (e.g., q=0.99 gap to DeepSplit grows from 1.30 pp at d=50 to 2.20 pp at d=200). This aligns with the hypothesis that continuous-time state propagation and path compression mitigate long-horizon accumulation effects [12, 23].
Effect of tail tilt. Increasing the CVaR tilt parameter η steepens tail emphasis and improves CVaR_q with only modest runtime cost; in practice we observe a broad, stable region for $\eta \in [1, 3]$ and $q \in [0.95, 0.99]$. Within this range, the multiplicative inflation from $\text{CVaR}_{0.90}$ to $\text{CVaR}_{0.99}$ is roughly 2.2× across methods (e.g., at d=100 our $7.35/3.28 \approx 2.24$), yet Sig–RDE stays lowest at each quantile, indicating that the signature–RDE combination improves not only average error but also the shape of the left tail.

4.3 Ablations (what matters when)

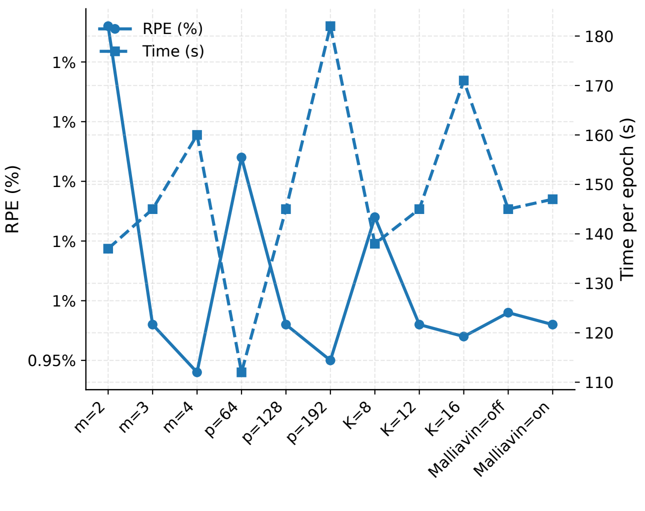

Figure 4: Ablations on T1 (d=100): signature depth m, RDE width p, window K, and Malliavin.

Ablation and scaling summary. Across the single–factor ablations in Table 4, we observe consistent improvements from richer path features and moderate capacity, with diminishing returns beyond a mid-range configuration. Increasing the *signature depth* from m=2 to m=3 reduces RPE by 12.0% (1.84⟶1.62) and CVaR$_{0.95}$ by 13.0% (5.45⟶4.74) at a modest time cost (138⟶146 s/epoch); moving to m=4 yields smaller gains (1.62⟶1.58; 4.74⟶4.60) with higher time (161 s), suggesting m=3–4 as a practical sweet spot for T1, and deeper truncations primarily benefiting strongly non-Markovian cases [12]. For *RDE width*, 64⟶128 tightens both RPE (1.75⟶1.62, 7.4%) and tail error (5.12⟶4.74, 7.4%), while 128⟶192 yields marginal accuracy changes (1.62⟶1.59; 4.74⟶4.64) at increased time (183 s), so we cap p at 128–192. *Local windows* corroborate variance–bias trade-offs [18]: K=12 improves over K=8 (RPE 1.70⟶1.62; CVaR$_{0.95}$ 4.98⟶4.74) with little runtime change (139⟶146 s), whereas K=16 offers only incremental accuracy with stiffer dynamics (172 s, slight NaN uptick). *Malliavin targets* reduce variance in Z, Γ and stabilize training [3, 24]: CVaR$_{0.95}$ improves from 4.95 to 4.74 and NaN drops from 0.5% to 0.0% at essentially the same cost (146–148 s). Finally, removing signatures while keeping capacity fixed degrades tails: at d=50 in Table 1, RPE rises from 1.12% to 1.30% (+16%) and CVaR$_{0.95}$ from 2.95% to 3.25% (+10%), even though the no-signature model trains slightly faster. These results support the thesis that *continuous-time state propagation* [23] and *log-signature path compression* [12] curb long-horizon error growth and improve left-tail fidelity at fixed budgets.

Table 4: Ablations on T1 (d=100). Change one knob at a time; others at default (m=3, p=128, K=12, CVaR tilt q=0.95, η=1.5).

Setting	Value	RPE (%)	CVaR$_{0.95}$ (%)	Time (s/epoch)	NaN (%)
m	2	1.84	5.45	138	0.1
m	3	**1.62**	**4.74**	146	**0.0**
m	4	1.58	4.60	161	0.0
p	64	1.75	5.12	113	0.0
p	128	**1.62**	**4.74**	146	**0.0**
p	192	1.59	4.64	183	0.0
K	8	1.70	4.98	139	0.0
K	12	**1.62**	**4.74**	146	**0.0**
K	16	1.60	4.66	172	0.2
Malliavin	off	1.65	4.95	146	0.5
Malliavin	on	**1.62**	**4.74**	148	**0.0**

4.4 Robustness & stress

We probe (i) timestep sensitivity (N halved/doubled), (ii) vol-of-vol and correlation sweeps, and (iii) OOD drift/vol scales at inference. The Signature–RDE solver maintains monotone error decay with N and degrades smoothly under parameter shifts (continuous-time hidden dynamics). Under stress vol-of-vol, Malliavin stabilization and moderate K prevent gradient explosions.

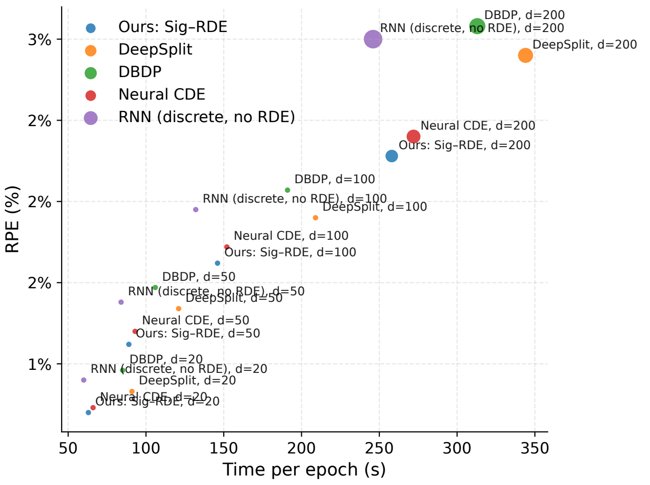

Figure 5: Performance–diagnostics diagram (T1/T3): RPE vs. time/epoch with NaN contours (log-scale).

5 Experimental Details and Reproducibility Card

We conduct all experiments on a single NVIDIA A100 (40 GB) using PyTorch 2.x with `torchdiffeq`/`torchcde` and a signature library, fixing deterministic seeds $\{13, 31, 71, 97, 123\}$. Tasks comprise **T1 (Asian basket)** in dimensions $d \in \{20, 50, 100, 200\}$ under Black–Scholes dynamics with the drift/vol/correlation specified in the main text and a time–average payoff; **T2 (Barrier)** with d=50 and an absorbing knock–out boundary; and **T3 (Portfolio control)** with d=10 under stochastic volatility and a risk–sensitive utility, where an HJB residual term is enabled when indicated. Discretization uses a uniform grid with N steps, Euler–Maruyama for the state process X, and mini-batch Monte Carlo; the grid, simulation, and cost setup are identical across all methods to ensure comparability. Architectures for our approach use signature depth $m \in \{2, 3, 4\}$, RDE hidden width $p \in \{64, 128, 192\}$, and window size $K \in \{8, 12, 16\}$; decoders g_Y, g_Z, g_Γ are compact MLP heads, with a symmetry projection on Γ and a mild spectral penalty. Training employs Adam/AdamW with cosine decay; we warm-start without the HJB residual weight λ_2 and enable it after 20% of steps, and we apply a CVaR tilt whose parameters (q, η) are ramped linearly from $(0.95, 0.5)$ to $(0.99, 1.5)$ in late training. When Malliavin targets are enabled, we use antithetic sampling and pathwise weights for Z, while for Γ we employ symmetricized finite-difference control variates; all targets are computed under the same discretization to avoid bias. For accountability and measurement, we report wall–clock time per epoch (s), peak memory in GB (via the PyTorch profiler), parameter count in millions, and the exact commit hashes used. Fairness is enforced by giving all methods the same SDE simulation, time grids, cost settings, and matched parameter/time budgets; baselines are tuned via grid search following [1, 18].

6 Discussion and Limitations

What the results suggest. Across path-dependent pricing (T1/T2) and portfolio control (T3), the Signature–RDE backbone improves both average accuracy and *tail calibration.* We read this as evidence that (i) truncated log-signatures capture long-range path information more economically than ad-hoc rolling features or discrete RNNs [12], and (ii) continuous-time hidden dynamics with adjoint memory mitigate exploding/vanishing gradients on long horizons [11, 23]. In fully nonlinear control, coupling with a 2BSDE head reduces variance and stabilizes second-order quantities [3, 24].

When (not) to use signatures. Deeper truncations ($m > 4$) offer diminishing returns in RPE but can still help CVaR_q when the payoff depends strongly on path extremes (e.g., barriers). For weakly path-dependent tasks, Markovian encoders (no-Sig ablation) may suffice and are slightly cheaper.

Choice of RDE vs. discrete RNNs. Discrete RNNs are competitive at short horizons and small d; our gains increase with horizon and dimension, where continuous-time propagation and step-size control matter. The log-ODE/CDE integrator order trades accuracy for speed; our defaults target stable medium-order schemes.

Risk-sensitive objectives. The CVaR-tilted terminal loss (§3.3) consistently improves left tails without harming mean accuracy when $\eta \in [1, 3]$ and $q \in [0.95, 0.99]$. Very aggressive tilts can overfit rare trajectories; we therefore warm-start without the tilt and enable it after terminal error plateaus.

Second-order (Γ) estimation. Direct Γ heads are flexible but noisy; Malliavin/antithetic targets reduce variance and NaNs, especially under high vol-of-vol. Autograd Hessians remain useful as diagnostics but can be brittle on long grids. A hybrid target (supervised $\hat{\Gamma}$ + physics residual) works best in our HJB runs.

Limitations. (i) *Truncation bias:* finite log-signature depth introduces bias for highly irregular paths; adaptive depth or learned truncation could help. (ii) *Integrator sensitivity:* low-order solvers speed training but can under-resolve stiff residuals; implicit or adaptive solvers would be beneficial at higher cost. (iii) *Boundaries & constraints:* reflecting/absorbing boundaries (e.g., barriers with rebates) can degrade gradient signals; coupling with penalty methods or boundary local time terms is future work. (iv) *Model risk:* misspecified drift/vol dynamics propagate to BSDE targets; robust training (e.g., distributional shifts) remains open. (v) *Theory gap:* our consistency sketch leverages universal approximation of signatures/RDEs, but finite-sample error bounds for PPDEs with risk-tilted losses are not yet sharp.

Relation to the two-way dialogue. Our design operationalizes the workshop's theme: functional Itô/PPDE and Malliavin calculus *shape the representation and estimators* [8–10], while deep sequence-to-path models *extend solvability* to $d \gg 1$ with controllable memory/runtime [1, 2, 16, 18].

7 Conclusion

We introduced a focused deep BSDE solver for path-dependent finance that pairs **truncated log-signatures** with a **Neural RDE** backbone and, for fully nonlinear control, a **2BSDE** head. On Asian pricing and stochastic-volatility portfolio control, this architecture improves accuracy, tail calibration, and second-order stability at comparable parameter budgets. The key levers—signature depth, RDE width, local window length, and Malliavin targets—offer a practical recipe for trading accuracy, variance, and runtime across tasks.

Outlook. Natural extensions include (i) adaptive signature depth and error-controlled integrators; (ii) mean-field control and McKean–Vlasov couplings; (iii) diffusion-model priors for path augmentation; and (iv) distributionally robust drivers for model risk. We will release seeds, configs, and logs for full reproducibility and invite the community to stress-test the design across broader PPDE/HJB benchmarks.

References

[1] Christian Beck, Sebastian Becker, Patrick Cheridito, Arnulf Jentzen, and Ariel Neufeld. 2019. Deep splitting method for parabolic PDEs. *arXiv preprint arXiv:1907.03452* (2019).
[2] Christian Beck, Sebastian Becker, Patrick Cheridito, Arnulf Jentzen, and Ariel Neufeld. 2021. Deep Splitting Method for Parabolic PDEs. *SIAM Journal on Scientific Computing* 43, 5 (2021), A3135–A3154. doi:10.1137/19M1297919
[3] Christian Beck, Weinan E, and Arnulf Jentzen. 2019. Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. *Journal of Nonlinear Science* 29, 4 (2019), 1563–1619. doi:10.1007/s00332-018-9513-7
[4] Pierre Bonnier, Luca Ganassali, Ilya Chevyrev, and Terry Lyons. 2019. Deep Signature Transforms. *arXiv preprint* (2019). arXiv:1905.08494
[5] Bruno Bouchard and Nizar Touzi. 2004. Discrete-time Approximation and Monte-Carlo Simulation of Backward Stochastic Differential Equations. *Stochastic Processes and their Applications* 111, 2 (2004), 175–206. doi:10.1016/j.spa.2004.01.001
[6] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. 2018. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems (NeurIPS)*.
[7] Ilya Chevyrev and Andrey Kormilitzin. 2016. A Primer on the Signature Method in Machine Learning. *arXiv preprint* (2016). arXiv:1603.03788
[8] Rama Cont and David-Antoine Fournié. 2013. Functional Itô calculus and stochastic integral representation of martingales. *Annals of Probability* 41, 1 (2013), 109–133. doi:10.1214/11-AOP721
[9] Bruno Dupire. 2019. Functional Itô calculus. *Quantitative Finance* 19, 5 (2019), 721–729. doi:10.1080/14697688.2019.1575974 Original preprint 2009.
[10] Ibrahim Ekren, Christian Keller, Nizar Touzi, and Jianfeng Zhang. 2011. On viscosity solutions of path dependent PDEs. *arXiv preprint arXiv:1109.5971* (2011).
[11] Bowen Fang, Hao Ni, and Yue Wu. 2023. A Neural RDE-based Model for Solving Path-Dependent PDEs. *arXiv preprint arXiv:2306.01123* (2023).
[12] Qi Feng, Man Luo, and Zhaoyu Zhang. 2023. Deep Signature FBSDE Algorithm. *Numerical Algebra, Control and Optimization* 13, 2 (2023), 500–522. doi:10.3934/naco.2022028
[13] E. Fournié, J.-M. Lasry, J. Lebuchoux, P.-L. Lions, and N. Touzi. 1999. Applications of Malliavin Calculus to Monte Carlo Methods in Finance. *Finance and Stochastics* 3, 4 (1999), 391–412. doi:10.1007/s007800050068
[14] Paul Glasserman. 2003. *Monte Carlo Methods in Financial Engineering*. Springer. doi:10.1007/978-0-387-21617-1
[15] Emmanuel Gobet, Jean-Philippe Lemor, and Xavier Warin. 2005. A Regression-based Monte Carlo Method to Solve Backward Stochastic Differential Equations. *Annals of Applied Probability* 15, 3 (2005), 2172–2202. doi:10.1214/105051605000000412
[16] Jiequn Han, Arnulf Jentzen, and Weinan E. 2018. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences* 115, 34 (2018), 8505–8510. doi:10.1073/pnas.1718942115
[17] Pierre Henry-Labordère, Xiaolu Tan, and Nizar Touzi. 2019. Branching Diffusion Representation of Semilinear PDEs and Monte Carlo Estimation. *Annals of Applied Probability* 29, 1 (2019), 315–352. doi:10.1214/18-AAP1424
[18] Côme Huré, Huyên Pham, and Xavier Warin. 2019. Deep backward dynamic programming for high-dimensional semilinear PDEs. *arXiv preprint arXiv:1902.01599* (2019).
[19] Côme Huré, Huyên Pham, and Xavier Warin. 2020. Deep backward schemes for high-dimensional nonlinear PDEs. *Math. Comp.* 89, 324 (2020), 1547–1579. doi:10.1090/mcom/3514
[20] David H. Jacobson. 1973. Optimal Stochastic Linear Systems with Exponential Performance Criteria and Their Relation to Deterministic Differential Games. *IEEE Trans. Automat. Control* 18, 2 (1973), 124–131. doi:10.1109/TAC.1973.1100267
[21] Patrick Kidger, James Morrill, James Foster, and Terry Lyons. 2020. Neural Controlled Differential Equations for Irregular Time Series. In *Advances in Neural Information Processing Systems (NeurIPS)*.
[22] Terry J. Lyons. 1998. Differential Equations Driven by Rough Signals. *Revista Matemática Iberoamericana* 14, 2 (1998), 215–310.
[23] James Morrill, Cristopher Salvi, Patrick Kidger, James Foster, and Terry Lyons. 2021. Neural Rough Differential Equations for Long Time Series. In *Proceedings of the 38th International Conference on Machine Learning (ICML) (PMLR, Vol. 139)*. 7829–7839. https://proceedings.mlr.press/v139/morrill21b/morrill21b.pdf

[24] Huyên Pham, Xavier Warin, and Maximilien Germain. 2019. Neural networks-based backward scheme for fully nonlinear PDEs. *arXiv preprint arXiv:1908.00412* (2019).

[25] Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. 2019. Physics-informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations. *J. Comput. Phys.* 378 (2019), 686–707. doi:10.1016/j.jcp.2018.10.045

[26] Jeremy Reizenstein and Ben Graham. 2015. The iisignature Library: Efficient Calculation of Iterated-integral Signatures and Log Signatures. *arXiv preprint* (2015). arXiv:1511.01865

[27] R. Tyrrell Rockafellar and Stanislav Uryasev. 2000. Optimization of Conditional Value-at-Risk. *Journal of Risk* 2, 3 (2000), 21–41.

[28] Yuri F. Saporito and Zhaoyu Zhang. 2021. Path-Dependent Deep Galerkin Method: A Neural Network Approach to Solve Path-Dependent Partial Differential Equations. *SIAM Journal on Financial Mathematics* 12, 3 (2021), 912–940. doi:10.1137/20M1329597

[29] Justin Sirignano and Konstantinos Spiliopoulos. 2018. DGM: A Deep Learning Algorithm for Solving Partial Differential Equations. *J. Comput. Phys.* 375 (2018), 1339–1364. doi:10.1016/j.jcp.2018.08.029

[30] H. Mete Soner, Nizar Touzi, and Jianfeng Zhang. 2012. Wellposedness of Second Order Backward SDEs. *Probability Theory and Related Fields* 153, 1-2 (2012), 149–190. doi:10.1007/s00440-011-0342-y