INEUS: Iterative Neural Solver for High-Dimensional PIDEs

Jean-Loup Dupret[1] **Davide Gallon**[1,2] **Patrick Cheridito**[1]

[1]Department of Mathematics and RiskLab, ETH Zurich, Switzerland

[2]Applied Mathematics: Institute for Analysis and Numerics, University of Münster, Germany

{jdupret,patrickc}@ethz.ch, {davide.gallon}@uni-muenster.de

Abstract

In this paper, we introduce INEUS, a meshfree iterative neural solver for partial integro-differential equations (PIDEs). The method replaces the explicit evaluation of nonlocal jump integrals with single-jump sampling and reformulates PIDE solving as a sequence of recursive regression problems. Like Physics-Informed Neural Networks (PINNs), INEUS learns global solutions over the entire space-time domain, yet it offers a more efficient treatment of nonlocal terms and avoids the computationally expensive differentiation of full PIDE residuals. These features make INEUS particularly well suited for high-dimensional PDEs and PIDEs. Supported by a contraction-based convergence proof for linear PIDEs, our numerical experiments show that INEUS delivers accurate and scalable solutions for various high-dimensional linear and nonlinear examples.

1 Introduction

Partial differential equations (PDEs) and their extensions, partial integro-differential equations (PIDEs), are important modeling tools in the natural sciences, engineering, economics and finance [1; 24; 29]. While traditional numerical methods such as finite differences [14; 31] and finite elements [45; 39] are effective for low-dimensional problems, they quickly become computationally intractable as the dimension increases. This limitation, widely known as the curse of dimensionality, arises because mesh-based approaches require spatial discretization, causing computational costs to scale exponentially with the state dimension.

Deep learning for PDEs. In recent years, neural networks have emerged as effective tools for mitigating the curse of dimensionality in the numerical solution of high-dimensional PDEs [10; 11]. One prominent approach is based on Physics-Informed Neural Networks (PINNs) [41; 42; 44; 9; 27; 46; 37; 48; 16], which minimize PDE residuals using gradient-based optimization and automatic differentiation. A key strength of PINNs is their flexibility: they can be applied to a wide range of challenging PDEs and provide *global* approximations over the entire space-time domain, making them particularly suitable for real-time applications. A second strand of the literature leverages the Feynman–Kac representation or backward stochastic differential equations (BSDEs, also known as nonlinear Feynman–Kac representations) to approximate parabolic PDEs on a finite time grid using neural networks [25; 26; 12; 7; 6; 40]. Because they rely on time-discretized stochastic dynamics, these methods introduce approximation errors that may accumulate over time steps and iterations, while their accuracy depends strongly on how well the sample paths cover the space-time domain. Beyond these two paradigms, alternative deep learning frameworks include the deep mixed residual method (MIM) [35], weak adversarial networks (WAN) [51], the deep Ritz method [18], and the deep Nitsche method [32].

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Deep learning for PIDEs. Despite these advances, the deep learning literature on PIDEs remains comparatively sparse. Existing PINN-based extensions [36; 43; 28; 5] struggle with nonlocal terms because they require the computationally expensive numerical approximation of the integral term at every sampled space-time point. On the other hand, Feynman–Kac and BSDE-based approaches are able to handle nonlocal terms efficiently by incorporating jumps into the dynamics of the underlying auxiliary process [19; 3; 22; 38; 20; 47; 34; 49]. However, in contrast to PINNs which are trained directly on the original PIDE, Feynman–Kac and BSDE-based methods instead solve a time-discretized approximation of the PIDE and are trained only on data points along a given time grid. Even when arbitrary restarts are used to improve domain coverage, BSDE methods still rely on simulated time-discretized trajectories. As a result, approximation errors tend to accumulate over successive time steps, their accuracy depends on how well simulated paths cover the spatial domain and a finer time grid increases computational cost. Recently, Georgoulis et al. [21] also propose a time-discretized approach for PIDEs, but combine a variational neural loss with an implicit-explicit minimizing movement time-stepping scheme.

Contributions. To bridge this gap, we introduce INEUS, a meshfree iterative neural solver for nonlinear PIDEs, inspired by [13], with the following key features:

- It combines the global approximation capabilities of PINNs with the efficient treatment of nonlocal terms provided by Feynman–Kac and BSDE-based methods.
- By using an expectation-free recursive target based on single-point jump sampling, it avoids explicit numerical integration of nonlocal terms and remains efficient in high dimensions.
- Its recursive formulation reduces the differentiation burden of residual-based PINNs by avoiding backpropagation through the full integro-differential operator and the explicit evaluation of gradients, Hessians, and integral terms altogether.

We further provide a contraction-based convergence proof for linear PIDEs, showing that repeated minimization of the INEUS loss function approximates both the standard Feynman–Kac operator and a relaxed contractive variant that enhances algorithmic stability via Polyak averaging. We illustrate the accuracy and scalability of INEUS on several linear and nonlinear numerical examples, and compare its performance with PINNs and deep BSDE methods for PDEs and PIDEs. Proofs of the theoretical results and additional numerical experiments are given in the Appendix.

2 INEUS

We describe INEUS for a representative class of parabolic PIDEs, noting that it extends directly to more general PIDEs. Specifically, we consider equations of the form

$$\partial_t u(t,x) + \mathcal{F}\big(t,x,u(t,x),\nabla_x u(t,x),\nabla_x^2 u(t,x)\big) + \mathcal{I}(t,x,u) = 0, \quad (t,x) \in [0,T) \times D, \tag{1}$$

$$u(T,x) = \varphi(x), \quad x \in D, \tag{2}$$

$$u(t,x) = g(t,x), \quad (t,x) \in [0,T) \times D^c, \tag{3}$$

for a finite time horizon $T > 0$ and an open subset[1] $D \subseteq \mathbb{R}^d$. We write $D_T := [0,T) \times D$ and $\bar{D}_T = [0,T] \times \bar{D}$. The function $\mathcal{F}\colon D_T \times \mathbb{R} \times \mathbb{R}^d \times \mathbb{R}^{d\times d} \to \mathbb{R}$ specifies the local differential operator, while $\mathcal{I}$ is a nonlocal operator given by

$$\mathcal{I}(t,x,u) = \lambda(t,x) \int_E \ell(u(t,x+\gamma(t,x,e)) - u(t,x))\, \nu(de),$$

for a probability measure ν on $E := \mathbb{R}^l \setminus \{0\}$, together with suitable functions $\lambda : D_T \to \mathbb{R}_+$, $\gamma : D_T \times E \to \mathbb{R}^d$ and $\ell\colon \mathbb{R} \to \mathbb{R}$. Using random variables $E_1, E_2, \ldots \overset{\text{i.i.d.}}{\sim} \nu$, one can write

$$\mathcal{I}(t,x,u) = \lambda(t,x)\, \mathbb{E}_\nu \left[\ell(u(t,x+\gamma(t,x,E_1)) - u(t,x))\right]. \tag{4}$$

Let us denote by $\mathcal{U}$ the set of all functions in $C^{1,2}(D_T) \cap C(\bar{D}_T)$ such that the jump expectation in (4) is finite for all $(t,x) \in D_T$ and by $\mathcal{A}$ the operator on $\mathcal{U}$ given by the left-hand side of (1):

$$\mathcal{A}[u](t,x) := \partial_t u(t,x) + \mathcal{F}(t,x,u(t,x),\nabla_x u(t,x),\nabla_x^2 u(t,x)) + \mathcal{I}(t,x,u)\,. \tag{5}$$

[1] In the special case $D = \mathbb{R}^d$, the complement D^c is empty, so condition (3) becomes vacuous.

For a given scaling parameter $\xi \in \mathbb{R} \setminus \{0\}$, we introduce the corresponding expectation-free operator $\mathcal{G}_\xi : D_T \times E \times \mathcal{U} \to \mathbb{R}$ by[2]

$$\mathcal{G}_\xi(t,x,e,u) := u(t,x) + \xi\Big[\partial_t u(t,x) + \\ \mathcal{F}(t,x,u(t,x),\nabla_x u(t,x),\nabla_x^2 u(t,x)) + \lambda(t,x)\,\ell(u(t,x+\gamma(t,x,e)) - u(t,x))\Big].$$

In contrast to $\mathcal{A}$, the map $\mathcal{G}_\xi$ evaluates the jump term at a single realization $e \in E$, thereby avoiding the computation of the jump expectations $\mathcal{I}$ at every sampled space-time point (t,x). However, averaging over $E_1 \sim \nu$ recovers the PIDE operator:

$$\mathbb{E}_\nu[\mathcal{G}_\xi(t,x,E_1,u)] = u(t,x) + \xi\mathcal{A}[u](t,x).$$

The following proposition then shows that, when u solves the PIDE, the INEUS target $\mathcal{G}_\xi$ yields an exact population regression that characterizes u.

Proposition 2.1. *Let $u \in \mathcal{U}$ be a solution of the PIDE (1)–(3) and let Y be a D_T-valued random variable independent of E_1 such that $\mathbb{E}[\mathcal{G}_\xi(Y,E_1,u)^2] < \infty$. Then $u(Y) = v^*(Y)$ almost surely for the Borel measurable function $v^* : D_T \to \mathbb{R}$ minimizing the mean squared error*

$$\mathbb{E}\Big[\big(v(Y) - \mathcal{G}_\xi(Y,E_1,u)\big)^2\Big]$$

over all Borel measurable functions $v : D_T \to \mathbb{R}$.

Motivated by Proposition 2.1, we approximate the PIDE solution u with a neural network $u_\theta : [0,T] \times \mathbb{R}^d \to \mathbb{R}$ and iteratively update its parameters by solving the regression

$$\theta^{(k+1)} \in \arg\min_\theta \mathbb{E}\Big[\big(u_\theta(Y) - \mathcal{G}_\xi(Y,E_1,u_{\theta^{(k)}})\big)^2\Big], \tag{6}$$

for a D_T-valued random variable Y with distribution μ and independent of E_1. We use hard-constrained PINNs (hPINNs) [33], which enforce the terminal and boundary conditions (2)–(3) directly by specifying the network architecture of u_θ as

$$u_\theta(t,x) = A(t,x) + B(t,x)v_\theta(t,x)\,, \tag{7}$$

for a function $A(t,x)$ encoding the terminal and boundary conditions, a smooth distance function $B(t,x)$ vanishing for $t = T$ and $x \in D^c$, and another neural network $v_\theta : D_T \to \mathbb{R}$ trained only in the interior D_T. In practice, we sample independent realizations $(t_m,x_m,e_m)_{m=1}^M$ of $(Y,E_1) \sim \mu \otimes \nu$ on $D_T \times E$ and solve

$$\theta^{(k+1)} \in \arg\min_\theta \frac{1}{M}\sum_{m=1}^M \big(u_\theta(t_m,x_m) - \mathcal{G}_\xi(t_m,x_m,e_m,u_{\theta^{(k)}})\big)^2.$$

This allows us to mitigate the curse of dimensionality by avoiding any space-time discretization of the PIDE (1)–(3), resulting in the meshfree INEUS Algorithm 1. This recursive formulation offers three main computational advantages over standard PINN training.

Cheaper gradients. Standard PINNs minimize an empirical objective of the form $(1/M)\sum_{m=1}^M A[u_\theta]^2(t_m,x_m)$ using a gradient method with respect to θ. Since $\mathcal{A}$ already contains second-order spatial derivatives, this produces third-order mixed derivatives during backpropagation, which are both expensive to compute and numerically sensitive. In our formulation, the target $\mathcal{G}_\xi(Y,E_1,u_{\theta^{(k)}})$ is evaluated at the frozen weights $\theta^{(k)}$ and treated as a constant during the backward pass. Differentiating (6) with respect to θ therefore requires only $\nabla_\theta u_\theta$; see Algorithm 1 and the explicit gradient update (33). Notably, this advantage persists even without nonlocal terms, so also for standard PDEs; see Figures 8–9 below.

No numerical integration of the jump term. The expectation-free target $\mathcal{G}_\xi$ evaluates the jump integrand at a single draw $E_1 \sim \nu$, rather than numerically approximating the full integral term $\mathcal{I}$ at each sampled point (t,x). In Appendix C.1, we examine which choice of the scaling parameter ξ optimally mitigates the jump-induced variance and discuss the trade-off between accuracy and computational cost when averaging multiple jump samples in $\mathcal{G}_\xi$. Comparing INEUS with PINNs using the same number of jump samples per space-time point also helps distinguish the benefits of the regression-based INEUS formulation from those of its single-jump sampling.

[2] with the convention that whenever $x + \gamma(t,x,e) \in D^c$, we set $u(t,x+\gamma(t,x,e)) = g(t,x+\gamma(t,x,e))$.

Preservation of population minimizers under jump sampling. Appendix C.1 shows that approximating the nonlocal term by Monte Carlo jump sampling introduces an additional jump variance term into the PINN population loss, which can shift its minimizer. This is not the case for INEUS since this additional jump variance term is independent of the function being optimized and evaluated at the frozen target $\mathcal{G}_\xi$.

The next lemma provides a useful characterization of (6) in terms of its underlying operator.

Lemma 2.2. *Let μ be a probability measure on D_T with full support and $u_{\theta^{(k)}} \in \mathcal{U}$ be given at the k-th iteration step such that $\mathbb{E}\left[\mathcal{G}_\xi(Y, E_1, u_{\theta^{(k)}})^2\right] < \infty$. Let $u^* : \bar{D}_T \to \mathbb{R}$ be a Borel measurable function that minimizes the objective*

$$\mathbb{E}\Big[\big(v(Y) - \mathcal{G}_\xi(Y, E_1, u_{\theta^{(k)}})\big)^2\Big], \tag{8}$$

over all Borel functions $v : D_T \to \mathbb{R}$, where $(Y, E_1) \sim \mu \otimes \nu$. Then, u^* satisfies the update rule

$$u^*(t,x) = \overline{\mathcal{T}}_\xi\, u_{\theta^{(k)}}(t,x) := u_{\theta^{(k)}}(t,x) + \xi \mathcal{A}[u_{\theta^{(k)}}](t,x)\,, \quad \mu\text{-almost everywhere on } D_T. \tag{9}$$

Moreover, if $u^* \in \mathcal{U}$ is realized by a hPINN of the form (7) and is a fixed point of the operator $\overline{\mathcal{T}}_\xi$, then u^* is a classical solution of the PIDE problem (1)–(3).

Lemma 2.2 shows that the update rule (6) based on an hPINN u_θ is realized by the operator $\overline{\mathcal{T}}_\xi$, whose fixed points are solutions of the PIDE (1)–(3). However, $\overline{\mathcal{T}}_\xi$ is, in general, not a contraction. Proposition 3.2 below shows that a suitable modification of the operator $\overline{\mathcal{T}}_\xi$ defines a contraction mapping on the space of bounded continuous functions and that this modified operator can be recovered through repeated minimization of (6).

3 Convergence analysis

The convergence analysis exploits the Feynman–Kac representation and the contractivity of the associated semigroup, which are naturally available for linear PIDEs. We therefore restrict to this setting, noting that the algorithm itself applies unchanged to the nonlinear problems of Section 4. Appendix C.3 also presents an extension of the linear theory to solution functions of polynomial growth and discusses nonlinear settings. We hence consider linear PIDEs in $D = \mathbb{R}^d$ of the form[3]

$$\partial_t u(t,x) + \mathcal{L}[u](t,x) + \mathcal{I}[u](t,x) - c(t,x)u(t,x) + f(t,x) = 0, \qquad (t,x) \in D_T, \tag{10}$$

$$u(T,x) = \varphi(x), \qquad x \in D. \tag{11}$$

Here, we denote by $\mathcal{C}$ the space of bounded, uniformly continuous real-valued functions on $\bar{D}_T$. We assume that $f, c, \varphi \in \mathcal{C}$ and that there exists a constant $c_0 > 0$ such that

$$c(t,x) \geq c_0 > 0, \qquad (t,x) \in \bar{D}_T\,. \tag{12}$$

The non-local operator $\mathcal{I}$ in (4) is such that $\ell(z) = z$ and the differential operator $\mathcal{L}$ is of the form

$$\mathcal{L}[u](t,x) = b^\top(t,x)\nabla_x u(t,x) + \frac{1}{2}\mathrm{Tr}\left[\sigma\sigma^\top(t,x)\nabla_x^2 u(t,x)\right],$$

with coefficients $b : D_T \to \mathbb{R}^d$, $\sigma : D_T \to \mathbb{R}^{d\times q}$, $\gamma : D_T \times E \to \mathbb{R}^d$, $\lambda : D_T \to \mathbb{R}_+$ ensuring that (14) has a unique solution exhibiting uniform stochastic continuity and that the associated Feynman–Kac operator $\mathcal{T}_h : \mathcal{C} \to \mathcal{C}$ introduced below is well defined for every $h \geq 0$.

Theorem 3.1 (Feynman–Kac)**.** *Let $u \in \mathcal{U} \cap \mathcal{C}$ be a classical solution of the PIDE (10)–(11), then for every $(t,x) \in \bar{D}_T$,*

$$u(t,x) = \mathbb{E}_x\left[\int_t^T e^{-\int_t^s c(r, X_r^{t,x})dr} f(s, X_s^{t,x})ds + e^{-\int_t^T c(r, X_r^{t,x})dr}\varphi(X_T^{t,x})\right], \tag{13}$$

for a stochastic process $(X_s^{t,x})_{t\leq s\leq T}$ satisfying

$$dX_s^{t,x} = b(s, X_s^{t,x})ds + \sigma(s, X_s^{t,x})dW_s + \int_E \gamma(s, X_{s-}^{t,x}, e)\, dN(ds, de), \quad X_t^{t,x} = x\,, \tag{14}$$

where W is a q-dimensional Brownian motion and N is an integer-valued random measure on $[0,T] \times E$ with predictable compensator $\lambda(s, X_{s-}^{t,x})\nu(de)ds$.

[3] Boundary conditions (3) and general open domains $D \subseteq \mathbb{R}^d$ can both be treated easily by introducing the stopping time $\tau := T \wedge \inf\{s \geq t : X_s \notin D\}$ in the Feynman–Kac formula, see Th. 4.2 in [50]. We here omit this extension for readability.

3.1 The Feynman–Kac semigroup and its generator

The Feynman–Kac representation (13) naturally defines a two-parameter family of affine operators. To obtain a one-parameter semigroup structure, we fix an initial point $y = (t, x) \in \bar{D}_T$ and introduce the space-time process $Y_s^y := (t+s, X_{t+s}^{t,x})$ for $0 \le s \le T-t$. For $h \ge 0$, we set $\tau_h(t) := h \wedge (T-t)$ and introduce the function spaces

$$\mathcal{C}_\varphi := \{\psi \in \mathcal{C} : \psi(T, x) = \varphi(x) \text{ for all } x \in D\},$$
$$\mathcal{C}_0 := \{\psi \in \mathcal{C} : \psi(T, x) = 0 \text{ for all } x \in D\}.$$

The set $\mathcal{C}_\varphi$ is a closed affine subspace of $\mathcal{C}$ and is therefore complete under the supremum norm $\|\cdot\|_\infty$, whereas $\mathcal{C}_0$ is a Banach space under $\|\cdot\|_\infty$. For $\psi \in \mathcal{C}$, define the Feynman–Kac operator

$$\mathcal{T}_h\psi(y) := \mathbb{E}_y\left[\int_0^{\tau_h(t)} e^{-\int_0^s c(Y_r^y)dr} f(Y_s^y)ds + e^{-\int_0^{\tau_h(t)} c(Y_r^y)dr}\psi(Y_{\tau_h(t)}^y)\right], \tag{15}$$

where $\mathbb{E}_y$ denotes expectation conditional on $Y_0^y = y$. The affine operator $\mathcal{T}_h$ admits the decomposition

$$\mathcal{T}_h\psi = \mathcal{R}_h + \mathcal{S}_h\psi, \tag{16}$$

where $\mathcal{R}_h$ accounts for the source term f,

$$\mathcal{R}_h(y) := \mathbb{E}_y\left[\int_0^{\tau_h(t)} e^{-\int_0^s c(Y_r^y)dr} f(Y_s^y)ds\right],$$

and $\mathcal{S}_h$ is the linear Feynman–Kac operator

$$\mathcal{S}_h\psi(y) := \mathbb{E}_y\left[e^{-\int_0^{\tau_h(t)} c(Y_r^y)dr}\psi(Y_{\tau_h(t)}^y)\right], \quad \psi \in \mathcal{C}. \tag{17}$$

Proposition 3.2. *Let $(\mathcal{S}_h)_{h\ge0}$ be the family of linear operators defined in (17). Then, the following statements hold.*

(i) *The family $(\mathcal{S}_h)_{h\ge0}$ is a strongly continuous, monotone semigroup on $\mathcal{C}$.*

(ii) *Under Assumption (12), for every $h > 0$, the operator $\mathcal{S}_h$ maps $\mathcal{C}_0$ into itself and is a strict contraction on this space:*

$$\|\mathcal{S}_h\psi\|_\infty \le e^{-c_0 h}\|\psi\|_\infty, \qquad \psi \in \mathcal{C}_0.$$

Moreover, $\mathcal{T}_h$ maps $\mathcal{C}_\varphi$ into itself and is a strict contraction on this space, with

$$\|\mathcal{T}_h\psi_1 - \mathcal{T}_h\psi_2\|_\infty \le e^{-c_0 h}\|\psi_1 - \psi_2\|_\infty, \qquad \psi_1, \psi_2 \in \mathcal{C}_\varphi.$$

(iii) *Let $\mathcal{A}_0$ denote the infinitesimal generator of $(\mathcal{S}_h)_{h\ge0}$, with domain*

$$D(\mathcal{A}_0) := \left\{\psi \in \mathcal{C} : \exists\, \Phi \in \mathcal{C},\ \lim_{h\downarrow0}\left\|\frac{\mathcal{S}_h\psi - \psi}{h} - \Phi\right\|_\infty = 0\right\}.$$

For every $\psi \in D(\mathcal{A}_0) \cap \mathcal{U}$ such that the function $\partial_t\psi + \mathcal{L}[\psi] + \mathcal{I}[\psi] - c\psi$ defined on D_T admits a bounded, uniformly continuous extension to $\bar{D}_T$, then $\mathcal{A}_0\psi$ coincides with this extension:

$$\mathcal{A}_0\psi(t, x) = \partial_t\psi(t, x) + \mathcal{L}[\psi](t, x) + \mathcal{I}[\psi](t, x) - c(t, x)\psi(t, x), \qquad (t, x) \in D_T.$$

In particular, if $u \in D(\mathcal{A}_0) \cap \mathcal{U}$ is a classical solution of the PIDE (10), then we obtain from (5),

$$\mathcal{A}[u](t, x) = \mathcal{A}_0 u(t, x) + f(t, x) = 0, \qquad (t, x) \in D_T.$$

3.2 A contractive simulation-free operator

Recall that update rule (9) is realized by the operator $\overline{\mathcal{T}}_\xi$ acting on $\mathcal{U}$, where

$$\overline{\mathcal{T}}_\xi\,\psi = \psi + \xi\mathcal{A}[\psi].$$

On the other hand, one can see from (16) and the pointwise relation $\mathcal{R}_h(t, x)/h \to f(t, x)$ that

$$\overline{\mathcal{T}}_\xi\,\psi(t, x) = \psi(t, x) + \xi\lim_{h\downarrow0}\frac{\mathcal{T}_h\psi(t, x) - \psi(t, x)}{h}, \qquad (t, x) \in D_T,$$

for all $\psi \in \mathcal{D}(\mathcal{A}_0) \cap \mathcal{U}$. By Lemma 2.2, the exact solution u is a fixed point of $\overline{\mathcal{T}}_\xi$. However, in general, $\overline{\mathcal{T}}_\xi$ is not contractive. We therefore introduce a *relaxed operator* based on the Feynman–Kac affine family $(\mathcal{T}_h)_{h\ge0}$.

Proposition 3.3. *For $h > 0$ and $\alpha > 0$, define $\mathcal{T}_{h,\alpha} : \mathcal{C}_\varphi \to \mathcal{C}_\varphi$ by*

$$\mathcal{T}_{h,\alpha}\psi := \psi + \frac{\alpha}{h}\left(\mathcal{T}_h\psi - \psi\right) = \Big(1 - \frac{\alpha}{h}\Big)\psi + \frac{\alpha}{h}\mathcal{T}_h\psi\,. \tag{18}$$

(i) Assume that (12) holds. If $0 < \alpha < 2h/(1 + e^{-c_0 h})$, then $\mathcal{T}_{h,\alpha}$ is a strict contraction on $\mathcal{C}_\varphi$, with Lipschitz constant

$$\Big|1 - \frac{\alpha}{h}\Big| + \frac{\alpha}{h}e^{-c_0 h} \in (0, 1)\,.$$

(ii) The solution u of the PIDE (10)–(11) is the unique fixed point of $\mathcal{T}_{h,\alpha}$ in $\mathcal{C}_\varphi$.

First, $\mathcal{T}_{h,\alpha}$ can be viewed as a finite-time-step approximation of $\overline{\mathcal{T}}_\xi$: when $\alpha = \xi$, it replaces the infinitesimal increment $\mathcal{A}[\psi]$ by $(\mathcal{T}_h\psi - \psi)/h$. More generally, the *relaxation* parameter α in (18) need not coincide with the *scaling* parameter ξ used in $\overline{\mathcal{T}}_\xi$. This additional degree of freedom will play an important role in Theorem 3.5, where it provides greater flexibility in controlling the stability of the numerical scheme. Second, although $\mathcal{T}_{h,\alpha}$ is contractive for $0 < \alpha < 2h/(1 + e^{-c_0 h})$, a direct implementation of this operator would still require the evaluation of the Feynman–Kac operator $\mathcal{T}_h$, which in turn involves simulating the dynamics of Y^y and approximating the conditional expectation by Monte Carlo averages over many sample paths. Such a procedure tends to be computationally expensive, introduces time-discretization errors and yields only a local approximation for fixed initial conditions y. We avoid these difficulties by using the following semigroup expansion.

Proposition 3.4. *Fix $n \geq 1$ and let $\psi \in \mathcal{U} \cap \mathcal{C}$. Assume the residual*

$$\mathcal{A}\psi = \partial_t\psi + \mathcal{L}[\psi] + \mathcal{I}[\psi] - c\psi + f\,,$$

satisfies $\mathcal{A}\psi \in D(\mathcal{A}_0^n) \cap \mathcal{C}_0$. Then, for every $h \geq 0$, there exist remainder terms $R_{n+1}(h) \in \mathcal{C}$ such that the Feynman–Kac operator (15) admits the expansion

$$\mathcal{T}_h\psi = \psi + \sum_{j=1}^{n} \frac{h^j}{j!}\mathcal{A}_0^{j-1}(\mathcal{A}\psi) + R_{n+1}(h)\,, \tag{19}$$

where

$$\|R_{n+1}(h)\|_\infty \leq \frac{h^{n+1}}{(n+1)!}\big\|\mathcal{A}_0^n(\mathcal{A}\psi)\big\|_\infty = \mathcal{O}(h^{n+1}) \qquad \text{as } h \downarrow 0.$$

Moreover, for $h > 0$ and $\psi \in \mathcal{C}_\varphi$, the relaxed operator (18) satisfies

$$\mathcal{T}_{h,\alpha}\psi = \psi + \alpha\left[\sum_{j=1}^{n} \frac{h^{j-1}}{j!}\,\mathcal{A}_0^{j-1}(\mathcal{A}\psi) + \widetilde{R}_n(h)\right], \tag{20}$$

where $\widetilde{R}_n(h) := R_{n+1}(h)/h$ satisfies

$$\|\widetilde{R}_n(h)\|_\infty \leq \frac{h^n}{(n+1)!}\big\|\mathcal{A}_0^n(\mathcal{A}\psi)\big\|_\infty = \mathcal{O}(h^n) \qquad \text{as } h \downarrow 0.$$

3.3 An efficient training algorithm

Importantly, since $\mathcal{A}_0$ is a second-order partial integro-differential operator, evaluating $\mathcal{A}_0^j$ requires computing partial derivatives of order up to $2j$, which quickly becomes computationally prohibitive. The following theorem shows that repeated exact minimization of the INEUS objective (6) generates an approximation of both the truncated expansions (19) and (20) of the Feynman–Kac operator $\mathcal{T}_h$ and its relaxed operator $\mathcal{T}_{h,\alpha}$, while avoiding the explicit computation of the costly operators $\mathcal{A}_0^j$.

Theorem 3.5. *Fix an epoch k and an integer $n \geq 1$. Let $h > 0$, set $\xi = h/n$, and choose $0 < \alpha < 2h/(1 + e^{-c_0 h})$. Assume that*

$$u_0 := u_{\theta^{(k)}} \in \mathcal{U} \cap \mathcal{C}_\varphi, \qquad \mathcal{A}u_0 \in D(\mathcal{A}_0^n) \cap \mathcal{C}_0. \tag{21}$$

Define the n-step inner recursion, for $i = 0, \ldots, n-1$, by

$$u_{i+1} := u_{\theta_{i+1}}, \qquad \theta_{i+1} \in \arg\min_\theta \mathbb{E}\Big[\big(u_\theta(Y) - \mathcal{G}_\xi(Y, E_1, u_i)\big)^2\Big], \tag{22}$$

where $(Y, E_1) \sim \mu \otimes \nu$ and μ is a probability measure with full support on D_T. At every inner step i, assume that $\mathbb{E}\left[\mathcal{G}_\xi(Y, E_1, u_i)^2\right] < \infty$ and that the corresponding minimum over all square-integrable measurable functions of Y is attained by an admissible hPINN $u_{i+1} \in \mathcal{U} \cap \mathcal{C}_\varphi$. Define the next outer iterate by the Polyak average

$$u^{(k+1)} := \left(1 - \frac{\alpha}{h}\right) u_{\theta^{(k)}} + \frac{\alpha}{h} u_n. \tag{23}$$

Then, on $\bar{D}_T$,

$$u^{(k+1)} = u_{\theta^{(k)}} + \alpha \sum_{j=1}^{n} \frac{h^{j-1}}{j!} \mathcal{A}_0^{j-1}\left(\mathcal{A} u_{\theta^{(k)}}\right) + \frac{\alpha}{h} E_n(h), \tag{24}$$

where the remainder $E_n(h) \in \mathcal{C}$ is given by (42) and satisfies

$$\|E_n(h)\|_\infty \leq \frac{h^2}{2n} \sum_{j=2}^{n} \frac{h^{j-2}}{(j-2)!} \|\mathcal{A}_0^{j-1}(\mathcal{A} u_0)\|_\infty \,. \tag{25}$$

In particular, the relaxed operator $\mathcal{T}_{h,\alpha}$ satisfies as $h \downarrow 0$,

$$\|u^{(k+1)} - \mathcal{T}_{h,\alpha} u_{\theta^{(k)}}\|_\infty \leq \frac{\alpha}{h} \|E_n(h)\|_\infty + \alpha \|\widetilde{R}_n(h)\|_\infty = \mathcal{O}\left(\frac{\alpha h}{n}\right) + \mathcal{O}(\alpha h^{\,n}) \,. \tag{26}$$

Moreover, for $\alpha = h$, one has $u^{(k+1)} = u_n$ and $\mathcal{T}_{h,\alpha} = \mathcal{T}_h$. Hence, as $h \downarrow 0$,

$$\left\|u^{(k+1)} - \mathcal{T}_h u_{\theta^{(k)}}\right\|_\infty \leq \|E_n(h)\|_\infty + \|R_{n+1}(h)\|_\infty = \mathcal{O}\left(\frac{h^2}{n}\right) + \mathcal{O}(h^{n+1}). \tag{27}$$

In practice, each inner minimization (22) is approximated by several steps of mini-batch gradient descent [15; 2], yielding the efficient and robust INEUS training procedure described in Algorithm 1. Importantly, considering the relaxed operator $\mathcal{T}_{h,\alpha}$ instead of the Feynman–Kac operator $\mathcal{T}_h$ provides an additional tunable parameter α, which increases the flexibility and stability of the scheme. In particular, since $\alpha = h$ gives back the original Feynman–Kac operator, $\mathcal{T}_{h,\alpha}$ contains $\mathcal{T}_h$ as a special case. Since INEUS is an iterative scheme, it is known to be less stable for nonlinear function approximation than residual-based methods such as standard PINNs, see [4; 13]. But for $0 < \alpha < h$, the relaxed operator $\mathcal{T}_{h,\alpha}$ then helps mitigate this training instability by damping changes in the network output through the Polyak averaging step (23), as illustrated in Figure 7 of Appendix D.

Moreover, choosing the scaling parameter $\xi = h/n$ mitigates the jump-induced variance of the single-jump target $\mathcal{G}_\xi$, as quantified in Proposition C.1. Also, the operator $\mathcal{G}_\xi$ requires the evaluation of the second-order differential operator $\mathcal{L}[u_\theta]$, which in principle involves computing $\partial_t u_\theta$, $\nabla_x u_\theta$, and $\nabla_x^2 u_\theta$. However, Proposition C.2 reduces the cost of evaluating $\mathcal{G}_\xi$ by replacing this explicit computation of the gradients and Hessians of u_θ with a single directional second derivative. Finally, the hPINN parameterization u_θ can be modified to impose the higher-order terminal compatibility assumption (21). However, we did not observe a performance gain from this modification in our experiments and therefore retained the original and more expressive hPINN parameterization (7).

4 Numerical results

The proposed INEUS method for nonlinear PIDEs is summarized in Algorithm 1 below. In all examples, we use the DGM architecture of [44] as it has been shown to improve PINN performance empirically. Further details of the network design and hyperparameters are given in Appendix A. Additional experiments for both PDEs and PIDEs are reported in Appendix D.

Linear PIDE with quadratic terminal condition. We first assess INEUS on a benchmark problem with closed-form solution. We consider the linear PIDE in d dimensions on $D_T = [0, T) \times \mathbb{R}^d$,

$$\partial_t u(t,x) + b x^\top \nabla_x u(t,x) + \frac{1}{2} \mathrm{Tr}[\Sigma\Sigma^\top \nabla_x^2 u(t,x)] - c u(t,x) + \lambda \int_{\mathbb{R}^d} \big(u(t, x+e) - u(t,x)\big) \varphi_{\Sigma_J}(e) de = 0 \,, \tag{28}$$

subject to the terminal condition $u(T,x) = \|x\|^2$, where $\Sigma \in \mathbb{R}^{d \times q}$ and φ_{Σ_J} is the $\mathcal{N}(0, \Sigma_J)$-density with $\Sigma_J \in \mathbb{R}^{d \times d}$. The corresponding solution is explicitly given by

$$u(t,x) = e^{(2b-c)(T-t)} \|x\|^2 + e^{-c(T-t)} \big(\mathrm{Tr}[\Sigma\Sigma^\top] + \lambda \mathrm{Tr}[\Sigma_J]\big) \frac{e^{2b(T-t)} - 1}{2b} \,. \tag{29}$$

Figure 1 displays the INEUS approximation for this problem in dimension $d = 100$. In this high-dimensional setting with jumps, standard PINNs become computationally intractable, whereas INEUS still provides an accurate approximation of the solution.

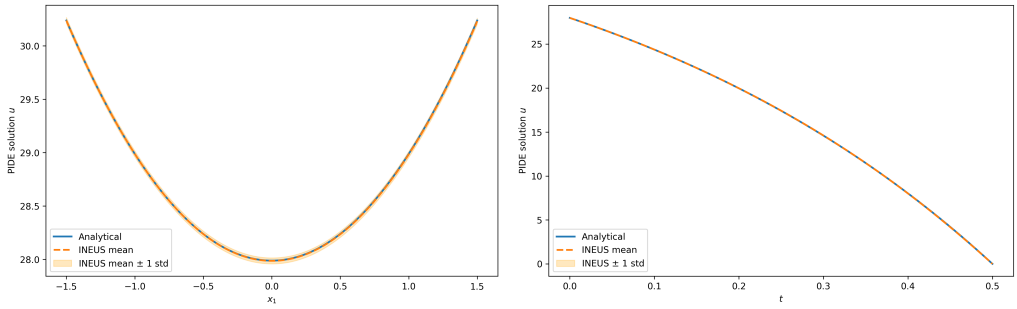

Figure 1: PIDE solution $u(0, x)$ for $x = (x_1, 0, \ldots, 0)$ with $x_1 \in [-1.5, 1.5]$ (left) and $u(t, x)$ for $t \in [0, 0.5]$ and $x = \mathbf{0}_{100}$ (right) for a 100-dimensional linear PIDE of the form (28). Orange dotted lines: numerical results of INEUS with ± 1 standard deviation given by orange shaded area. Blue lines: analytical solution (29).

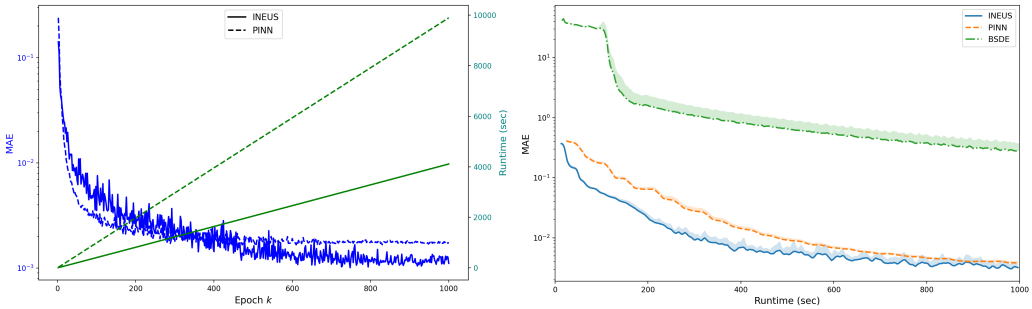

Figure 2: Numerical solutions of a 10-dimensional linear PIDE of the form (28). Left panel: comparison of MAE (blue) and runtime in seconds (green) of INEUS (solid lines) and PINN (dashed lines) as functions of the epoch k. Right panel: comparison of INEUS, PINN and deep BSDE methods as functions of runtime (seconds).

Figure 2 compares INEUS with competing methods for $d = 10$, where PINNs are now computationally feasible. The left plot reports the runtime and mean absolute error (MAE) of $u_{\theta^{(k)}}$ as functions of the number of epochs k, where

$$\mathrm{MAE} = \frac{1}{\mathcal{M}} \sum_{i=1}^{\mathcal{M}} |u_{\theta^{(k)}}(t_i, x_i) - u(t_i, x_i)|$$

on a test set of $\mathcal{M} = 2000$ uniformly sampled points from $[0, 0.5] \times [-1.5, 1.5]^d$. Overall, INEUS exhibits faster convergence in terms of epochs and a substantially lower computational cost. This gain comes from avoiding both the numerical evaluation of third-order derivatives and the numerical integration of the jump term. However, we see that the residual-based structure of PINN makes it slightly more stable than INEUS [4]. The right panel shows that INEUS also outperforms the deep BSDE method in terms of MAE as a function of runtime. To obtain a meaningful comparison over the whole space-time domain, we ran the jump-adapted deep BSDE [3] starting from many randomly sampled initial points $(t_0, x_0) \in D_T$. However, because that approach depends on simulated discretized stochastic processes, its performance deteriorates when used to approximate a global solution, both due to discretization errors that accumulate and to its limited ability to cover the full domain. These observations are further supported by the following nonlinear examples and by the additional experiments on higher-dimensional PIDEs and PDEs in Appendix A.3 and D.

Nonlinear Hamilton–Jacobi–Bellman PIDE. Let us consider the nonlinear PIDE on $[0, T) \times \mathbb{R}^d$,

$$\partial_t u(t, x) + \mathrm{Tr}[\nabla_x^2 u(t, x)] - \eta \|\nabla_x u(t, x)\|^2 - \frac{\lambda}{\eta} \int_{\mathbb{R}^d} \left(e^{-\eta(u(t,x+e)-u(t,x))} - 1 \right) \nu(de) + f(t, x) = 0, \quad (30)$$

with terminal condition $u(T, x) = \varphi(x)$. By the Feynman–Kac formula (13), the solution is given by

$$u(t, x) = -\frac{1}{\eta} \log \left(\mathbb{E} \left[\exp \left(- \int_t^T \eta f(s, X_s) ds \right) e^{-\eta \varphi(X_T)} \,\middle|\, X_t = x \right] \right). \quad (31)$$

where $dX_s = \sqrt{2}\, dW_s + \int_{\mathbb{R}^d} eN(ds, de), t \le s \le T,\ X_t = x$, and N is a Poisson measure with intensity $\lambda\nu(de)ds$, $\lambda \ge 0$. This probabilistic representation provides a Monte Carlo (MC) estimator for $u(t, x)$, thereby serving as a reference solution for assessing the INEUS accuracy on the nonlinear PIDE (30). Here we study a 100-dimensional case with $\varphi(x) = ||x||^2$ and Gaussian jumps $\nu = \mathcal{N}(\mu_J, \Sigma_J)$.

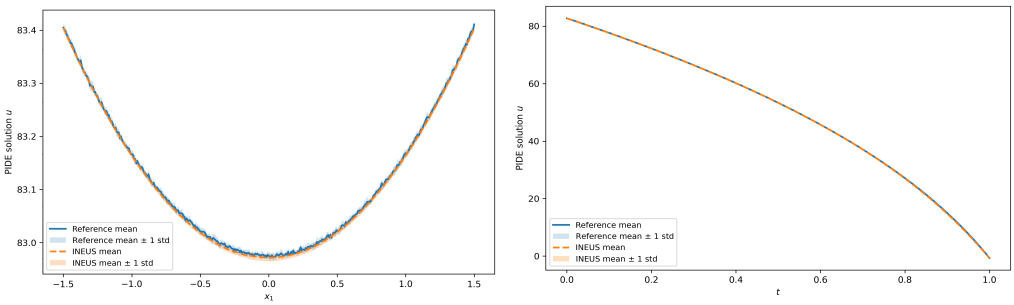

Figure 3: PIDE solution $u(0, x)$ for $x = (x_1, 0, \ldots, 0)$ with $x_1 \in [-1.5, 1.5]$ (left) and $u(t, x)$ for $t \in [0, 1]$ and $x = \mathbf{0}_{100}$ (right) for a 100-dimensional nonlinear PIDE of the form (30). Orange dotted lines: numerical results of INEUS with ± 1 standard deviation given by orange shaded area. Blue lines: reference MC solution based on (31).

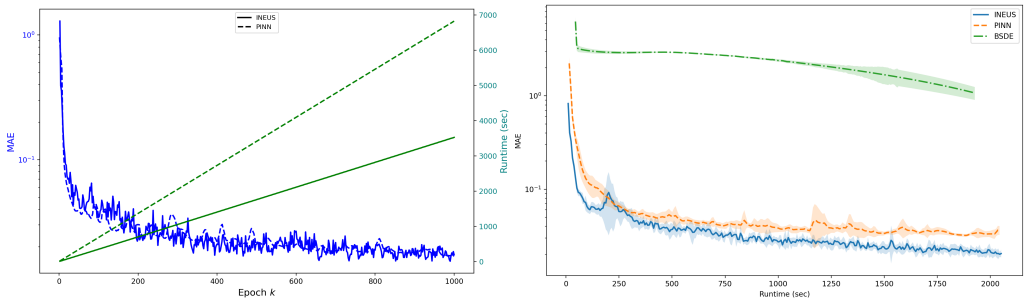

Figure 4: Numerical solutions of a 10-dimensional nonlinear PIDE of the form (30). Left panel: comparison of MAE (blue) and runtime in seconds (green) of INEUS (solid lines) and PINN (dashed lines) as functions of the epoch k. Right panel: comparison of INEUS, PINN and deep BSDE methods as functions of runtime (seconds).

Nonlinear Black–Scholes PIDE. In our last example, we add jumps to the d-dimensional nonlinear Black–Scholes PDE with default risk from [26], yielding the following PIDE on $D_T = [0, T) \times \mathbb{R}^d_+$:

$$\partial_t u + \beta\, x \cdot \nabla_x u + \frac{1}{2}\mathrm{Tr}\big[\sigma^2\, \mathrm{diag}(x)^2 \nabla_x^2 u\big] - \big[(1-\delta)Q(u) + R\big]u + \lambda \mathbb{E}_\nu\big[u\big(t,\, x \odot e^{\mathbf{E}}\big) - u(t, x)\big] = 0, \tag{32}$$

with terminal condition $u(T, x) = \min_{i=1,\ldots,d} x_i$ and $Q(u)$ the piecewise-linear default intensity (48). Here, $\sigma, \beta, \delta, R, \lambda \in \mathbb{R}_+$, and $\odot$ denotes component-wise multiplication. The jump vector $\mathbf{E} = (E_1, \ldots, E_d)$ has distribution ν and is specified by $E_i = \mu_J + \sigma_J\big(\sqrt{\rho_J}\, Z_0 + \sqrt{1-\rho_J}\, Z_i\big)$, where $Z_0, Z_1, \ldots, Z_d$ are independent standard normal random variables. The upper row of Figure 5 considers the diffusion-only case ($\lambda = 0$), for which a reference value $u(0, x_0) = 57.300$ is available from [26]. The lower row of Figure 5 reports the corresponding experiment with jumps, for which the reference value $u(0, x_0) = 55.810$ is computed via the forward Picard scheme [8], based on the Feynman–Kac representation with 4×10^5 antithetic paths and 30 Picard iterations. It can be seen that, as training progresses, INEUS substantially reduces the variability and standard deviation of the prediction, while converging toward the correct solution. Additional experiments for both the nonlinear Black–Scholes PIDE (32) and a linear Black–Scholes variant (47) (with reference solution) are provided in Appendix D.

5 Conclusions, limitations and future work

We have proposed INEUS, a recursive learning framework for high-dimensional nonlinear PIDEs with jumps. INEUS combines hard-constrained PINNs with an expectation-free target, thereby avoiding

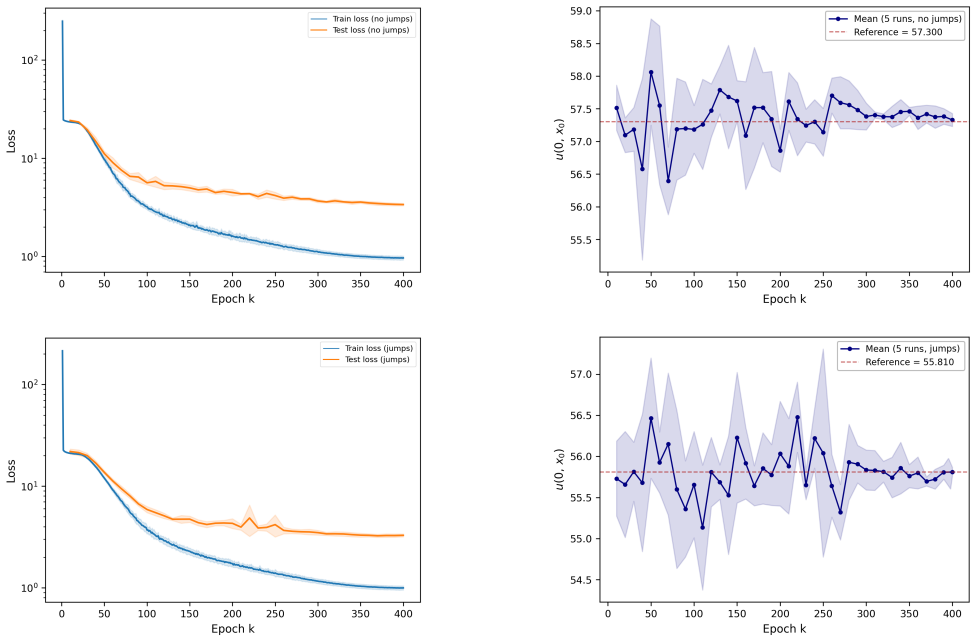

Figure 5: INEUS for the 100-dimensional nonlinear Black–Scholes equation (32), without jumps (upper row) and with jumps (lower row). The left panels show the training and test losses over 400 epochs, while the right panels show the prediction trajectory of $u_\theta(0, x_0)$ with $x_0 = (100, \dots, 100)$. The no-jump estimate is compared with the reference value 57.300 of the deep-BSDE method, while the jump estimate is compared with the reference value 55.810 of the Picard iteration.

both explicit jump integration and the expensive higher-order derivatives required by standard residual-based PINN training. This yields a meshfree global solver that remains effective in high dimension and is naturally adapted to jump processes. For linear PIDEs, we established a contraction-based theoretical foundation by linking the recursive updates to both the Feynman–Kac semigroup and a relaxed version. This shows that INEUS approximates a class of fixed-point operators with provable convergence properties, while the relaxed formulation provides an additional mechanism to improve training stability. Our numerical results show that INEUS achieves accurate global approximations at a lower computational cost than standard residual-based PINNs and deep BSDE approaches for high-dimensional PDEs and PIDEs.

Several directions remain open. Further theoretical work is needed to extend the convergence analysis to nonlinear PIDEs and infinite-activity Lévy operators. These extensions would require developing a substantially different mathematical machinery from the one used in the paper to handle nonlinear evolution and to control the sampling, approximation and optimization errors arising in the practical algorithm. On the algorithmic side, adaptive relaxation strategies and improved sampling schemes may further enhance stability and efficiency.

Acknowledgments and disclosures

The authors declare that they have no conflicts of interest and thank the reviewers for their helpful comments. The work was supported by the Swiss National Science Foundation under Grant No. 10003723. The code accompanying this paper is available at https://github.com/jdupret97/INEUS.

References

[1] F. Abergel and R. Tachet. A nonlinear partial integro-differential equation from mathematical finance. *Discrete and Continuous Dynamical Systems. Series A*, 27(3):907–917, 2010.

[2] Z. Allen-Zhu, Y. Li, and Z. Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019.

[3] K. Andersson, A. Gnoatto, M. Patacca, and A. Picarelli. A deep solver for BSDEs with jumps. *SIAM Journal on Financial Mathematics*, 16(3):875–911, 2025.

[4] L. Baird. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pages 30–37. Elsevier, 1995.

[5] S. Bansal, P. Boro, and N. Srinivasan. Application of physics informed neural networks to partial integro-differential equations in financial modeling and decision making. *Applied Soft Computing*, 186:114208, 2026.

[6] C. Beck, S. Becker, P. Cheridito, A. Jentzen, and A. Neufeld. Deep splitting method for parabolic PDEs. *SIAM Journal on Scientific Computing*, 43(5):A3135–A3154, 2021.

[7] C. Beck, S. Becker, P. Grohs, N. Jaafari, and A. Jentzen. Solving the Kolmogorov PDE by means of deep learning. *Journal of Scientific Computing*, 88(3):73, 2021.

[8] C. Bender and R. Denk. A forward scheme for backward SDEs. *Stochastic Processes and Their Applications*, 117(12):1793–1812, 2007.

[9] J. Berg and K. Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28–41, 2018.

[10] J. Blechschmidt and O. G. Ernst. Three ways to solve partial differential equations with neural networks—A review. *GAMM-Mitteilungen*, 44(2):e202100006, 2021.

[11] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis. Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37:1727–1738, 2021.

[12] Q. Chan-Wai-Nam, J. Mikael, and X. Warin. Machine learning for semi linear PDEs. *Journal of Scientific Computing*, 79(3):1667–1712, 2019.

[13] P. Cheridito, J.-L. Dupret, and D. Hainaut. Deep learning for continuous-time stochastic control with jumps. *arXiv preprint arXiv:2505.15602*, 2025.

[14] R. Cont and E. Voltchkova. A finite difference scheme for option pricing in jump diffusion and exponential Lévy models. *SIAM Journal on Numerical Analysis*, 43(4):1596–1626, 2005.

[15] S. Du, J. Lee, H. Li, L. Wang, and X. Zhai. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pages 1675–1685. PMLR, 2019.

[16] J.-L. Dupret and D. Hainaut. Deep learning for high-dimensional continuous-time stochastic optimal control without explicit solution. *Operations Research*, 2026.

[17] R. Durrett. *Probability: Theory and Examples*. Cambridge University Press, fifth edition, 2019.

[18] W. E and B. Yu. The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6:1–12, 2018.

[19] R. Frey and V. Köck. Deep neural network algorithms for parabolic PIDEs and applications in insurance and finance. *Computation*, 10(11):201, 2022.

[20] R. Frey and V. Köck. Convergence analysis of the deep splitting scheme: The case of partial integro-differential equations and the associated forward backward SDEs with jumps. *SIAM Journal on Scientific Computing*, 47(1):A527–A552, 2025.

[21] E. H. Georgoulis, A. Papapantoleon, and C. Smaragdakis. A deep implicit-explicit minimizing movement method for partial integro-differential equations, with application to option pricing in jump-diffusion models. *Communications in Nonlinear Science and Numerical Simulation*, page 109709, 2026.

[22] A. Gnoatto, K. Oberpriller, and A. Picarelli. Convergence of a Deep BSDE solver with jumps. *arXiv preprint arXiv:2501.09727*, 2025.

[23] E. Gobet, J.-P. Lemor, and X. Warin. A regression-based Monte Carlo method to solve backward stochastic differential equations. *The Annals of Applied Probability*, 15(3):2172–2202, 2005.

[24] A. Goswami, J. Patel, and P. Shevgaonkar. A system of non-local parabolic PDE and application to option pricing. *Stochastic Analysis and Applications*, 34(5):893–905, 2016.

[25] J. Han, A. Jentzen, et al. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):349–380, 2017.

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

[27] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[28] M. Kartik and N. T. Shah. Physics-informed neural networks for option pricing and hedging in illiquid jump markets. In *Proceedings of the 2025 3rd International Conference on Machine Learning and Pattern Recognition*, pages 88–96, 2025.

[29] N. I. Kavallaris and T. Suzuki. *Non-local Partial Differential Equations for Engineering and Biology*, volume 31 of *Mathematics for Industry*. Springer, 2018.

[30] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*. arXiv preprint arXiv:1412.6980, 2015.

[31] Y. Kwon and Y. Lee. A second-order finite difference method for option pricing under jump-diffusion models. *SIAM Journal on Numerical Analysis*, 49(6):2598–2617, 2011.

[32] Y. Liao and P. Ming. Deep Nitsche method: Deep Ritz method with essential boundary conditions. *Communications in Computational Physics*, 29:1365–1384, 2021.

[33] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. G. Johnson. Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing*, 43(6):B1105–B1132, 2021.

[34] L. Lu, H. Guo, X. Yang, and Y. Zhu. Temporal difference learning for high-dimensional PIDEs with jumps. *SIAM Journal on Scientific Computing*, 46(4):C349–C368, 2024.

[35] L. Lyu, Z. Zhang, M. Chen, and J. Chen. MIM: A deep mixed residual method for solving high-order partial differential equations. *Journal of Computational Physics*, 452:110930, 2022.

[36] S. Mishra and R. Molinaro. Physics informed neural networks for simulating radiative transfer. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 270:107705, 2021.

[37] S. Mowlavi and S. Nabi. Optimal control of PDEs using physics-informed neural networks. *Journal of Computational Physics*, 473:111731, 2023.

[38] A. Neufeld, P. Schmocker, and S. Wu. Full error analysis of the random deep splitting method for nonlinear parabolic PDEs and PIDEs. *arXiv preprint arXiv:2405.05192*, 2024.

[39] A. Quarteroni and A. Valli. *Numerical Approximation of Partial Differential Equations*. Springer Berlin Heidelberg, 1994.

[40] M. Raissi. Forward-backward stochastic neural networks: Deep learning of high-dimensional partial differential equations. In *Peter Carr Gedenkschrift: Research Advances in Mathematical Finance*, pages 637–655. 2024.

[41] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics informed deep learning (Part 1): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[42] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[43] E. Saleh, S. Ghaffari, T. Bretl, L. Olson, and M. West. Learning from integral losses in physics informed neural networks. *arXiv preprint arXiv:2305.17387*, 2023.

[44] J. Sirignano and K. Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

[45] P. Šolín. *Partial Differential Equations and the Finite Element Method*. John Wiley & Sons, 2005.

[46] S. Wang, X. Yu, and P. Perdikaris. When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

[47] W. Wang, J. Wang, J. Li, F. Gao, and Y. Fu. Deep learning numerical methods for high-dimensional fully nonlinear PIDEs and coupled FBSDEs with jumps. *arXiv preprint arXiv:2301.12895*, 2023.

[48] C. Wu, M. Zhu, Q. Tan, Y. Kartha, and L. Lu. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403:115671, 2023.

[49] Z. Ye and W. Wang. FBSJNN: a theoretically interpretable and efficiently deep learning method for solving partial integro-differential equations. *arXiv preprint arXiv:2412.11010*, 2024.

[50] J. Yong and X. Y. Zhou. *Stochastic Controls: Hamiltonian Systems and HJB Equations*, volume 43. Springer Science & Business Media, 1999.

[51] Y. Zang, G. Bao, X. Ye, and H. Zhou. Weak adversarial networks for high-dimensional partial differential equations. *Journal of Computational Physics*, 411:109409, 2020.

A Algorithm and network architecture

A.1 INEUS algorithm

We now describe the practical implementation of the full scheme in Theorem 3.5, which approximates the relaxed contracting operator $\mathcal{T}_{h,\alpha}$ (or $\mathcal{T}_h$ when $\alpha = h$). The algorithm is organized into outer epochs indexed by $k = 1, 2, \ldots$, inner recursion steps indexed by $i = 0, \ldots, n-1$, and gradient updates indexed by $j = 0, \ldots, N-1$. To simplify notation, we suppress the dependence of the weights on the inner-step index i and write $\theta_j^{(k)}$ instead of the more explicit $\theta_{i,j}^{(k)}$.

Algorithm 1 Iterative Neural Solver for PIDEs (INEUS)

Require: learning rate $\eta > 0$, relaxation parameter $\alpha > 0$, scale $h > 0$, number of inner steps $n \geq 1$, number of gradient steps N, batch size M.

Initialize admissible weights $\theta^{(0)}$ for the hPINN u_θ satisfying boundary conditions (2)–(3).
Set $\xi \leftarrow h/n$ and $k \leftarrow 0$.
repeat
  Set $\theta_0^{(k)} \leftarrow \theta^{(k)}$. ▷ Initialize the frozen target network
  for $i = 0$ to $n-1$ **do**
    for $j = 0$ to $N-1$ **do**
      Sample $\{(y_m, e_m)\}_{m=1}^M$ i.i.d. from $\mu \otimes \nu$ on $D_T \times E$.
      Form the empirical loss

$$\widehat{\mathscr{L}}^{(k)}(\theta\,;\xi) := \frac{1}{M}\sum_{m=1}^{M}\Big(u_\theta(y_m) - \mathcal{G}_\xi(y_m, e_m, u_{\theta_0^{(k)}})\Big)^2.$$

      Update

$$\theta_{j+1}^{(k)} \leftarrow \theta_j^{(k)} - \eta\nabla_\theta\widehat{\mathscr{L}}^{(k)}(\theta_j^{(k)}\,;\xi),$$

      or equivalently,

$$\theta_{j+1}^{(k)} \leftarrow \theta_j^{(k)} - \frac{2\eta}{M}\sum_{m=1}^{M}\Big(u_{\theta_j^{(k)}}(y_m) - \mathcal{G}_\xi(y_m, e_m, u_{\theta_0^{(k)}})\Big)\nabla_\theta u_{\theta_j^{(k)}}(y_m). \tag{33}$$

    end for
    Set $\theta_0^{(k)} \leftarrow \theta_N^{(k)}$. ▷ Freeze target for next inner step
  end for
  Apply the Polyak update

$$u^{(k+1)} \leftarrow \Big(1 - \frac{\alpha}{h}\Big)\,u_{\theta^{(k)}} + \frac{\alpha}{h}\,u_{\theta_N^{(k)}}.$$

  Choose parameters $\theta^{(k+1)}$ such that $u_{\theta^{(k+1)}} \approx u^{(k+1)}$, by regression onto the hPINN class, see Equation (35).
  $k \leftarrow k+1$.
until some convergence criterion is satisfied.
return $u_{\theta^{(k)}}$ and set $k_* \leftarrow k$.

We emphasize that after the Polyak update, the relaxed function $u^{(k+1)}$ is generally not itself represented exactly by the chosen hPINN parametrization. We therefore determine new parameters $\theta^{(k+1)}$ by projecting this updated target back onto the hPINN class, that is by choosing $u_{\theta^{(k+1)}}$ to approximate $u^{(k+1)}$ in a least-squares sense over sampled points (t, x). Since performing this regression at every epoch can be computationally costly, we instead carry it out only once every block of K epochs. Unrolling the Polyak recursion over this block, starting from epoch k, yields the target

$$u^{(k+K)} = \Big(1 - \frac{\alpha}{h}\Big)^K u_{\theta^{(k)}} + \sum_{j=1}^{K}\frac{\alpha}{h}\Big(1 - \frac{\alpha}{h}\Big)^{K-j} u_{\theta_N^{(k+j-1)}}. \tag{34}$$

Thus, at the end of the K-block, the new hPINN $u_{\theta^{(k+K)}}$ is obtained by regressing this weighted combination $u^{(k+K)}$ of the block-start network and the intermediate candidate networks back onto

the admissible hPINN class, that is,

$$\min_{\theta} \mathbb{E}_{(t,x)\sim\mu}\Big[u_\theta(t,x) - \Big((1-\alpha/h)^K u_{\theta^{(k)}}(t,x) + \sum_{j=1}^{K} \frac{\alpha}{h}(1-\alpha/h)^{K-j} u_{\theta_N^{(k+j-1)}}(t,x)\Big)\Big]^2, \quad (35)$$

where μ is the same space-time sampling distribution used for training (see A.2 below). For $K = 1$, this reduces exactly to Algorithm 1. The student network has the same hPINN architecture as the main model and is initialized with the last candidate network in the block. In the numerical experiments of Section 4, we use $K = 10$. The distillation is performed for 200 Adam steps with learning rate 10^{-3}, after which the distilled network replaces the current model and training resumes.

A.2 Hyperparameters

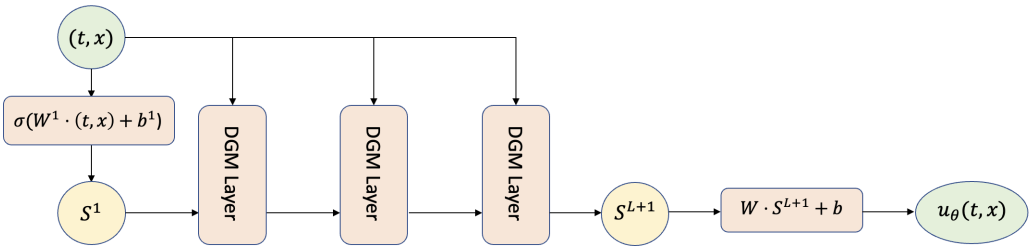

Figure 6: DGM architecture of the neural network u_θ with $L = 3$ (*i.e.* 4 hidden layers).

DGM network. Each DGM layer of the network u_θ in Figure 6 is of the following form

$$\begin{aligned}
S^1 &= \sigma\left(W^1 \cdot (t,x) + b^1\right), \\
Z^\ell &= \sigma\left(U^{z,\ell} \cdot (t,x) + W^{z,\ell} \cdot S^\ell + b^{z,\ell}\right), \quad \ell = 1,\dots,L, \\
G^\ell &= \sigma\left(U^{g,\ell} \cdot (t,x) + W^{g,\ell} \cdot S^\ell + b^{g,\ell}\right), \quad \ell = 1,\dots,L, \\
R^\ell &= \sigma\left(U^{r,\ell} \cdot (t,x) + W^{r,\ell} \cdot S^\ell + b^{r,\ell}\right), \quad \ell = 1,\dots,L, \\
H^\ell &= \sigma\left(U^{h,\ell} \cdot (t,x) + W^{h,\ell} \cdot \left(S^\ell \odot R^\ell\right) + b^{h,\ell}\right), \quad \ell = 1,\dots,L, \\
S^{\ell+1} &= \left(1 - G^\ell\right) \odot H^\ell + Z^\ell \odot S^\ell, \quad \ell = 1,\dots,L, \\
u_\theta(t,x) &= A(t,x) + B(t,x)\left(W \cdot S^{L+1} + b\right),
\end{aligned}$$

where the number of hidden layers is $L + 1$, $\cdot$ denotes matrix multiplication, $\odot$ element-wise multiplication, $A : [0,T] \times \mathbb{R}^d \to \mathbb{R}$ encodes the exact terminal and boundary conditions and $B : [0,T] \times \mathbb{R}^d \to \mathbb{R}$ is a smooth distance function vanishing for $t = T$ and $x \in D^c$, see (7). The DGM parameters of the value network are

$$\theta = \Big\{W^1, b^1, \left(U^{z,\ell}, W^{z,\ell}, b^{z,\ell}\right)_{\ell=1}^{L}, \left(U^{g,\ell}, W^{g,\ell}, b^{g,\ell}\right)_{\ell=1}^{L}, \left(U^{r,\ell}, W^{r,\ell}, b^{r,\ell}\right)_{\ell=1}^{L}, \left(U^{h,\ell}, W^{h,\ell}, b^{h,\ell}\right)_{\ell=1}^{L}, W, b\Big\}.$$

The number of units in each layer is n_{hid} and $\sigma : \mathbb{R}^{n_{\text{hid}}} \to \mathbb{R}^{n_{\text{hid}}}$ is a twice-differentiable element-wise nonlinearity. Throughout the numerical examples of the paper, we use $L = 3$ with $n_{\text{hid}} = 50$ neurons in each of the DGM layers, and we take $\tanh$ for the activation function σ.

Training setup. In Algorithm 1, INEUS is implemented with batch size $M = 2048$, number of inner steps $n = 20$, number of gradient steps $N = 32$, and a maximum number of epochs $k_* = 1000$ as the stopping criterion[4]. The network parameters are updated using Adam [30] with a constant learning rate $\eta = 0.0005$. Table 1 examines the sensitivity of INEUS to the learning rate η, the number of gradient steps N and the batch size M for the linear PIDE 28 in $d = 50$ dimensions. For

[4]We use a fixed epoch budget to facilitate comparison and reproducibility, but any other terminal criterion could be used equivalently, for instance based on the variation of the loss function $\widehat{\mathscr{L}}^{(k)}$ or in successive network outputs $u_{\theta^{(k)}}$.

each configuration, we report the mean and standard deviation of the mean absolute error (MAE) over 10 runs, together with the computation time. Overall, we observe that INEUS is fairly robust with respect to these hyperparameters, while emphasizing that their effects are strongly interdependent, making it difficult to isolate the influence of a single parameter.

Table 1: Sensitivity to the learning rate (top left), number of gradient steps (top right), and number of samples (bottom).

Learning rate η	Mean MAE	Std. dev. MAE	Time (sec.)
0.00005	0.0246	0.0025	6 364
0.0001	0.0123	0.0019	6 394
0.0005	0.0045	0.0006	6 570
0.001	0.0091	0.0021	6 633

Gradient steps N	Mean MAE	Std. dev. MAE	Time (sec.)
8	0.0116	0.0041	2 646
16	0.0076	0.0011	3 977
32	0.0045	0.0006	6 570
64	0.0053	0.0012	11 768

Batch size M	Mean MAE	Std. dev. MAE	Time (sec.)
256	0.0677	0.01143	6 139
1 024	0.0081	0.0014	6 402
2 048	0.0045	0.0006	6 570
4 096	0.0039	0.0004	6 986
8 192	0.0038	0.0002	7 338

Choice of α and h. The relaxation parameter α and scaling parameter h should be chosen within the contraction range $0 < \alpha < 2h/(1 + e^{-c_0 h})$. For fixed h, the contraction modulus $|1 - \frac{\alpha}{h}| + \frac{\alpha}{h} e^{-c_0 h}$ is minimized at $\alpha = h$ and equal to $e^{-c_0 h}$. In practice, α should therefore be chosen close to, but not necessarily equal to, h: $\alpha = h$ gives the strongest contraction, while $\alpha < h$ provides a more damped update that can improve numerical stability when recursive training introduces variability. The choice of h involves a separate trade-off: larger h reduces $e^{-c_0 h}$ and strengthens the contraction, but Theorem 3.5 shows that the update approximation error grows as $\mathcal{O}(\alpha h/n) + \mathcal{O}(\alpha h^n)$. Hence, h should balance stronger contraction against the accuracy of the recursive approximation. That is why, in most experiments, we set $\alpha = 0.4$ and $h = 0.5$, which provides a good trade-off between convergence speed, numerical stability and accuracy of the iterative updates in INEUS, although problem-specific fine-tuning may be beneficial. Alternatively, one may also consider an epoch-dependent relaxation factor $\alpha_k > 0$ in (23) to adapt the size of the outer update throughout training. Choosing $\alpha_k < h$ indeed dampens the update and improves stability by avoiding overly large changes in the networks $\|u_{\theta^{(k+1)}} - u_{\theta^{(k)}}\|$, while choosing $\alpha_k > h$ may accelerate convergence in flatter regions of the optimization landscape. These choices are justified by the following sensitivity analysis on the linear PIDE (28) in 50 dimensions, although their effects are again strongly interdependent.

Table 2: Sensitivity of INEUS to the relaxation parameter (top left), scale parameter (top right), and number of inner steps (bottom).

Relaxation parameter α	Mean MAE	Std. dev. MAE	Time (sec.)
0.05	0.0029	0.0007	6 408
0.1	0.0034	0.0008	6 623
0.25	0.0029	0.0019	6 599
0.4	0.0045	0.0006	6 570
0.5	0.0061	0.0018	6 299

Scale parameter h	Mean MAE	Std. dev. MAE	Time (sec.)
0.05	0.0081	0.0017	6 325
0.1	0.0059	0.0003	6 559
0.25	0.0056	0.0010	6 566
0.5	0.0045	0.0006	6 570
1	0.0066	0.0042	6 551

Inner steps n	Mean MAE	Std. dev. MAE	Time (sec.)
2	0.1408	0.0633	1 366
5	0.0149	0.0021	2 063
10	0.0131	0.0085	3 223
20	0.0045	0.0006	6 570
50	0.0081	0.0024	11 829

Sampling measure. Another crucial component is the choice of the sampling measure μ, which plays an important role in the quality of the approximation. We therefore use the residual-adaptive distribution (RAD) introduced by Wu et al. [48], which has been shown to improve the performance of PINN-style methods. We refer to Wu et al. [48] for an ablation study comparing RAD with standard uniform and Gaussian sampling strategies. Only for the Black–Scholes PIDEs (32)–(47), we use path-based sampling, which generates collocation points along geometric Brownian motion trajectories, to accelerate convergence.

A.3 Baselines

In the main text, we compared INEUS with a standard PINN/DGM baseline [44] and a classical deep BSDE method with jumps [3]. For all methods, the reported MAE is computed on the same fixed test set of $\mathcal{M} = 2000$ independently sampled space-time points. The PINN baseline uses the same network architecture, sampling domain, optimizer, and training hyperparameters as INEUS. The only difference is that it is trained by minimizing the standard squared PIDE residual directly and the nonlocal jump operator $\mathcal{I}[u]$ is approximated by Monte Carlo using 64 jump samples. One reported PINN or INEUS epoch then corresponds to $nN = 640$ optimizer updates. For the deep BSDE baseline, we use the classical local formulation for jump-diffusions. Since this method is intrinsically local and parameterizes a single initial value $u(t_0, x_0)$, it does not directly produce the global value function required for comparison with INEUS. We therefore sample multiple initial training points $(t_0, x_0) \sim \mu$, train independent local BSDE models at those points, and then fit a supervised global surrogate $u_\theta(t, x)$ to the resulting local solution labels. The global test MAE is computed by evaluating this surrogate on the fixed test set. Each local BSDE model is trained for 10,000 optimizer steps with batch size 1024, Adam learning rate 3×10^{-4}, gradient clipping norm 1.0, 25 time steps, and 64 Monte Carlo samples for the jump expectation. All methods are implemented in TensorFlow/Keras and run with GPU acceleration on an NVIDIA RTX 4090.

To avoid repeatedly solving the above independent deep BSDE problem for each initial condition (t_0, x_0) and subsequently fitting a global surrogate, we also consider a global BSDE-based alternative, namely the FBSNN method [40] and its extension to PIDEs, FBSJNN [49]. These methods directly parameterize the PIDE solution through $Y_t^{t,x} = u_\theta(t, x)$, thereby allowing the initial pair (t_0, x_0) to be sampled during training and a single network to approximate the solution over the full space-time domain. The first table below compares INEUS with this global FBSJNN extension of the deep BSDE method for the 50-dimensional linear PIDE (28). To ensure a fair comparison, accuracy is reported as a function of runtime (in seconds). Performance is evaluated using the three following metrics: (i) the global MAE, computed on a fixed space–time test set of $2{,}000$ observations; (ii) the trajectory MAE, computed over the states visited along 256 simulated trajectories on a fixed time grid; and (iii) the absolute local error at the initial condition $(t_0, x_0) = (0, \mathbf{0}_{50})$. Across these metrics and runtimes, INEUS consistently and substantially outperforms FBSJNN. In particular, the FBSJNN method does not appear to converge when learning the solution over the full space-time domain.

Table 3: Comparison of INEUS and the global FBSJNN baseline on the 50-dimensional linear PIDE (28) at matched wall-clock runtimes.

Method	Global MAE	Trajectory MAE	Local error at $(0, \mathbf{0}_{50})$
INEUS (100 sec.)	0.031	0.067	0.068
FBSJNN (100 sec.)	0.081	0.083	0.377
INEUS (500 sec.)	0.007	0.021	0.004
FBSJNN (500 sec.)	0.116	0.077	0.139
INEUS (1000 sec.)	0.005	0.016	0.001
FBSJNN (1000 sec.)	0.096	0.071	0.016

We finally provide local comparisons with temporal difference learning (TDL) [34] and deep splitting extensions [19] for PIDEs, using the publicly available implementations released by the respective authors. Extending these methods to global solvers over the full space-time domain would require substantial modifications to their existing implementations and we therefore restrict the comparison to the local error at the initial condition $(t_0, x_0) = (0, \mathbf{0}_{50})$. Nevertheless, the results show that INEUS

achieves competitive accuracy at the initial point $(0, x_0)$, despite simultaneously learning the solution over the full space-time domain.

Table 4: Local comparison of INEUS with TDL (left) and deep splitting (right) on the 50-dimensional linear PIDE (28). Errors are reported at the initial condition $(0, \mathbf{0}_{50})$ for matched wall-clock runtimes.

Method	Local error at $(0, \mathbf{0}_{50})$
INEUS (100 sec.)	0.068
Local TDL (100 sec.)	0.054
INEUS (500 sec.)	0.004
Local TDL (500 sec.)	0.014
INEUS (1000 sec.)	0.0015
Local TDL (1000 sec.)	0.011

Method	Local error at $(0, \mathbf{0}_{50})$
INEUS (100 sec.)	0.068
Local deep splitting (100 sec.)	0.042
INEUS (500 sec.)	0.004
Local deep splitting (500 sec.)	0.003
INEUS (1000 sec.)	0.0015
Local deep splitting (1000 sec.)	0.0023

To conclude, these comparisons highlight a structural advantage of INEUS for global approximation. Like PINNs, INEUS is explicitly designed to approximate the solution over the entire space-time domain. Indeed,

- INEUS and PINNs are trained directly on the original PIDE and aim to provide solutions for all space-time points (t, x) in the domain.
- Time-discretized BSDE-based methods instead solve a discretized approximation of the equation and are trained on data points only along a given time grid. Even when arbitrary restarts are used to improve domain coverage, BSDE-based methods still rely on simulated time-discretized trajectories. As a result, approximation errors tend to propagate/accumulate over successive time steps and their accuracy depends on how well the simulated paths cover the spatial domain.

B Proof of theoretical results

B.1 Proof of Proposition 2.1

Proof. Since Y is independent of E_1, one obtains from the definition of $\mathcal{G}_\xi$ that

$$\begin{aligned}\mathbb{E}[\mathcal{G}_\xi(Y, E_1, u) \mid Y = (t,x)] &= u(t,x) + \xi \Big[\partial_t u(t,x) + \mathcal{F}(t,x,u(t,x), \nabla_x u(t,x), \nabla_x^2 u(t,x)) \\ &\qquad + \lambda(t,x)\, \mathbb{E}_\nu \left[\ell(u(t, x+\gamma(t,x,E_1)) - u(t,x))\right]\Big] \\ &= u(t,x) + \xi\, \mathcal{A}[u](t,x) \qquad (36) \\ &= u(t,x),\end{aligned}$$

where the last equality follows from Equation (5). On the other hand, it is well known that

$$\mathbb{E}[\mathcal{G}_\xi(Y, E_1, u) \mid Y] = v^*(Y) \qquad (37)$$

for the Borel measurable function $v^* \colon D_T \to \mathbb{R}$ minimizing the mean squared error

$$v \mapsto \mathbb{E}\Big[\big(v(Y) - \mathcal{G}_\xi(Y, E_1, u)\big)^2\Big],$$

over all Borel measurable functions $v : D_T \to \mathbb{R}$, see e.g. Theorem 4.1.15 of Durrett [17]. This concludes the proof. □

B.2 Proof of Lemma 2.2

Proof. Since u^* is an exact minimizer of (8), Equations (36) and (37) in B.1 imply that, for μ-a.e. $(t,x) \in D_T$,

$$u^*(t,x) = \mathbb{E}[\mathcal{G}_\xi((t,x), E_1, u_{\theta^{(k)}})|(t,x)] = u_{\theta^{(k)}}(t,x) + \xi \mathcal{A}[u_{\theta^{(k)}}](t,x).$$

Hence the update rule (9) holds μ-a.e. on D_T. Now let u^* be a fixed point of $\overline{\mathcal{T}}_\xi$. By definition of (9), μ-a.e. on D_T,

$$u^* = \overline{\mathcal{T}}_\xi\, u^* \quad \Longleftrightarrow \quad u^* = u^* + \xi \mathcal{A}[u^*],$$

so that

$$\xi \mathcal{A}[u^*] = 0 \qquad \mu\text{-a.e. on } D_T.$$

Since $\xi \neq 0$, it follows that

$$\mathcal{A}[u^*] = 0 \qquad \mu\text{-a.e. on } D_T.$$

Because μ has full support and $(t,x) \mapsto \mathcal{A}[u^*](t,x)$ is continuous, we conclude that

$$\mathcal{A}[u^*](t,x) = 0 \qquad \text{for all } (t,x) \in D_T.$$

Finally, if u^* is represented by the hPINN architecture (7), then the terminal and boundary conditions are satisfied by construction, namely

$$u^*(T,x) = \varphi(x), \qquad x \in D,$$

and

$$u^*(t,x) = g(t,x), \qquad (t,x) \in [0,T) \times D^c.$$

Therefore, u^* satisfies (1) together with (2)–(3), and is thus a classical solution of the PIDE problem (1)–(3). □

B.3 Proof of Proposition 3.2

Proof. We establish (i), (ii), and (iii) in sequence.

Proof of (i). The operator $\mathcal{S}_h$ is linear by linearity of expectation. Since $c \geq c_0 > 0$,

$$\|\mathcal{S}_h \psi\|_\infty \leq \|\psi\|_\infty, \qquad \psi \in \mathcal{C}.$$

In particular, $\mathcal{S}_h$ is bounded. The identity $\mathcal{S}_0 = \mathbb{I}$ is clear from (17). For $h_1, h_2 \geq 0$, the truncated time increments satisfy

$$\tau_{h_1+h_2}(t) = \tau_{h_1}(t) + \tau_{h_2}\big(t + \tau_{h_1}(t)\big).$$

Indeed, both sides are equal to $(h_1+h_2) \wedge (T-t)$. Using this identity, the Markov property of Y^y, and the tower property of conditional expectation, we obtain

$$\begin{aligned}
\mathcal{S}_{h_1+h_2}\psi(y) &= \mathbb{E}_y\left[\exp\left(-\int_0^{\tau_{h_1+h_2}(t)} c(Y_r^y)\,dr\right)\psi\big(Y^y_{\tau_{h_1+h_2}(t)}\big)\right] \\
&= \mathbb{E}_y\left[\exp\left(-\int_0^{\tau_{h_1}(t)} c(Y_r^y)\,dr\right)\cdot \mathbb{E}\left[\exp\left(-\int_{\tau_{h_1}(t)}^{\tau_{h_1+h_2}(t)} c(Y_r^y)\,dr\right)\psi\big(Y^y_{\tau_{h_1+h_2}(t)}\big)\,\middle|\,\mathcal{F}_{\tau_{h_1}(t)}\right]\right] \\
&= \mathbb{E}_y\left[\exp\left(-\int_0^{\tau_{h_1}(t)} c(Y_r^y)\,dr\right)\mathcal{S}_{h_2}\psi\big(Y^y_{\tau_{h_1}(t)}\big)\right] \\
&= \mathcal{S}_{h_1}\big(\mathcal{S}_{h_2}\psi\big)(y).
\end{aligned}$$

Hence, we have the semigroup property

$$\mathcal{S}_{h_1+h_2} = \mathcal{S}_{h_1} \circ \mathcal{S}_{h_2}.$$

Monotonicity follows from the positivity of the discount factor. Indeed, if $\psi_1 \leq \psi_2$ pointwise, then

$$\exp\left(-\int_0^{\tau_h(t)} c(Y_r^y)\,dr\right)\psi_1\big(Y^y_{\tau_h(t)}\big) \leq \exp\left(-\int_0^{\tau_h(t)} c(Y_r^y)\,dr\right)\psi_2\big(Y^y_{\tau_h(t)}\big),$$

and therefore

$$\mathcal{S}_h\psi_1 \leq \mathcal{S}_h\psi_2.$$

To verify strong continuity, let $\psi \in \mathcal{C}$. For $y = (t,x) \in \bar{D}_T$, we have

$$|\mathcal{S}_h\psi(y) - \psi(y)| \leq \mathbb{E}_y\left[\left|\exp\left(-\int_0^{\tau_h(t)} c(Y_r^y)\,dr\right) - 1\right|\left|\psi\big(Y^y_{\tau_h(t)}\big)\right|\right] + \mathbb{E}_y\left[\left|\psi\big(Y^y_{\tau_h(t)}\big) - \psi(y)\right|\right].$$

Since c is bounded and $\tau_h(t) \leq h$,

$$\left|\exp\left(-\int_0^{\tau_h(t)} c(Y_r^y)\,dr\right) - 1\right| \leq \|c\|_\infty h.$$

Consequently, the first term converges to zero uniformly in y. The second term converges uniformly to zero by the uniform stochastic continuity of the space-time process Y^y and the uniform continuity of ψ. Hence,

$$\lim_{h\downarrow 0} \|\mathcal{S}_h\psi - \psi\|_\infty = 0.$$

Thus, $(\mathcal{S}_h)_{h\geq 0}$ is a strongly continuous, monotone semigroup on $\mathcal{C}$, which shows (i).

Proof of (ii)**.** For every $\psi \in \mathcal{C}$, we have

$$\mathcal{S}_h\psi(T,x) = \psi(T,x),$$

because $\tau_h(T) = 0$. In particular, $\mathcal{S}_h$ maps $\mathcal{C}_0$ into $\mathcal{C}_0$. Moreover, $\mathcal{R}_h(T,x) = 0$, and hence $\mathcal{T}_h$ maps $\mathcal{C}_\varphi$ into $\mathcal{C}_\varphi$.

Let $\psi \in \mathcal{C}_0$ and fix $y = (t,x) \in \bar{D}_T$. We distinguish two cases.

If $h \leq T - t$, then $\tau_h(t) = h$, and Assumption (12) yields

$$|\mathcal{S}_h\psi(y)| \leq \mathbb{E}_y\left[\exp\left(-\int_0^h c(Y_r^y)\,dr\right)|\psi(Y_h^y)|\right] \leq e^{-c_0 h}\|\psi\|_\infty.$$

If $h > T - t$, then $\tau_h(t) = T - t$ and $Y^y_{\tau_h(t)} = \big(T, X_T^{t,x}\big)$. Since $\psi \in \mathcal{C}_0$, $\psi\big(T, X_T^{t,x}\big) = 0$, and therefore

$$\mathcal{S}_h\psi(y) = 0.$$

Combining the two cases and taking the supremum over $y \in \bar{D}_T$ gives

$$\|\mathcal{S}_h\psi\|_\infty \leq e^{-c_0 h}\|\psi\|_\infty.$$

Since $e^{-c_0 h} < 1$ for every $h > 0$, the restriction of $\mathcal{S}_h$ to $\mathcal{C}_0$ is a strict contraction, showing (ii).

Now let $\psi_1, \psi_2 \in \mathcal{C}_\varphi$. Since they satisfy the same terminal condition,

$$\psi_1 - \psi_2 \in \mathcal{C}_0.$$

Moreover, the affine term $\mathcal{R}_h$ cancels, so that

$$\mathcal{T}_h\psi_1 - \mathcal{T}_h\psi_2 = \mathcal{S}_h(\psi_1 - \psi_2).$$

It follows that

$$\|\mathcal{T}_h\psi_1 - \mathcal{T}_h\psi_2\|_\infty \leq e^{-c_0 h}\|\psi_1 - \psi_2\|_\infty.$$

Proof of (iii). First, let $\psi \in D(\mathcal{A}_0)$. Since $\mathcal{S}_h\psi(T,x) = \psi(T,x)$ for every $h \geq 0$, the definition of the generator gives

$$\mathcal{A}_0\psi(T,x) = \lim_{h\downarrow 0}\frac{\mathcal{S}_h\psi(T,x) - \psi(T,x)}{h} = 0, \qquad x \in D.$$

Now let $\psi \in D(\mathcal{A}_0) \cap \mathcal{U}$ satisfy the additional regularity condition in the statement, and denote by $\Phi \in \mathcal{C}$ the continuous extension of

$$\partial_t\psi + \mathcal{L}[\psi] + \mathcal{I}[\psi] - c\psi$$

from D_T to $\bar{D}_T$.

Fix $y = (t,x) \in D_T$ and choose $0 < h < T - t$. For $0 \leq s \leq h$, define

$$U_s := e^{-\int_0^s c(Y_r^y)\,dr}, \qquad Z_s := U_s\psi(Y_s^y).$$

Itô's formula for jump-diffusions, together with the product rule and $dU_s = -c(Y_s^y)U_s\,ds$, yields

$$Z_s = \psi(y) + \int_0^s U_r\Phi(Y_r^y)\,dr + N_s, \qquad 0 \leq s \leq h, \tag{38}$$

where N is a local martingale with $N_0 = 0$.

Because $0 < U_s \leq 1$ and ψ, Φ are bounded, (38) implies

$$|N_s| \leq 2\|\psi\|_\infty + h\|\Phi\|_\infty, \qquad 0 \leq s \leq h.$$

Thus, N is a bounded local martingale on $[0, h]$. By localization and dominated convergence, it is a true martingale and $\mathbb{E}_y[N_h] = 0$. Taking expectations in (38) gives

$$\mathcal{S}_h\psi(y) - \psi(y) = \int_0^h \mathbb{E}_y\left[e^{-\int_0^s c(Y_r^y)\,dr}\Phi(Y_s^y)\right] ds = \int_0^h \mathcal{S}_s\Phi(y)\,ds,$$

where the last equality uses $s \le h < T - t$.

Since $\Phi \in \mathcal{C}$, strong continuity from (i) implies

$$\left|\frac{\mathcal{S}_h\psi(y) - \psi(y)}{h} - \Phi(y)\right| \le \frac{1}{h}\int_0^h \|\mathcal{S}_s\Phi - \Phi\|_\infty\, ds \longrightarrow 0$$

as $h \downarrow 0$. On the other hand, $\psi \in D(\mathcal{A}_0)$ ensures that the difference quotient converges in $\mathcal{C}$, and hence pointwise, to $\mathcal{A}_0\psi$. Uniqueness of the pointwise limit therefore gives

$$\mathcal{A}_0\psi(t,x) = \partial_t\psi(t,x) + \mathcal{L}[\psi](t,x) + \mathcal{I}[\psi](t,x) - c(t,x)\psi(t,x), \qquad (t,x) \in D_T.$$

Finally, both $\mathcal{A}_0\psi$ and Φ belong to $\mathcal{C}$ and agree on D_T. By continuity, they also agree at the terminal boundary. In particular, the extension necessarily satisfies $\Phi(T, \cdot) = 0$. which proves (iii). □

B.4 Proof of Proposition 3.3

Proof. We first verify that $\mathcal{T}_{h,\alpha}$ is well defined on $\mathcal{C}_\varphi$. By construction, $\mathcal{T}_h$ preserves $\mathcal{C}_\varphi$. In particular, for every $\psi \in \mathcal{C}_\varphi$,

$$\mathcal{T}_h\psi(T,x) = \psi(T,x) = \varphi(x), \qquad x \in D,$$

because $\tau_h(T) = 0$. Therefore,

$$\mathcal{T}_{h,\alpha}\psi(T,x) = \left(1 - \frac{\alpha}{h}\right)\psi(T,x) + \frac{\alpha}{h}\mathcal{T}_h\psi(T,x) = \left(1 - \frac{\alpha}{h}\right)\varphi(x) + \frac{\alpha}{h}\varphi(x) = \varphi(x).$$

Hence, $\mathcal{T}_{h,\alpha}\psi \in \mathcal{C}_\varphi$.

Proof of (i). Let $\psi_1, \psi_2 \in \mathcal{C}_\varphi$. Since both functions satisfy the same terminal condition,

$$\psi_1 - \psi_2 \in \mathcal{C}_0.$$

By Proposition 3.2 (ii),

$$\|\mathcal{T}_h\psi_1 - \mathcal{T}_h\psi_2\|_\infty \le e^{-c_0 h}\,\|\psi_1 - \psi_2\|_\infty.$$

Therefore, using (18),

$$\begin{aligned}
\mathcal{T}_{h,\alpha}\psi_1 - \mathcal{T}_{h,\alpha}\psi_2 &= \left(\psi_1 + \frac{\alpha}{h}\left(\mathcal{T}_h\psi_1 - \psi_1\right)\right) - \left(\psi_2 + \frac{\alpha}{h}\left(\mathcal{T}_h\psi_2 - \psi_2\right)\right) \\
&= (\psi_1 - \psi_2) + \frac{\alpha}{h}\left(\mathcal{T}_h\psi_1 - \mathcal{T}_h\psi_2 - (\psi_1 - \psi_2)\right) \\
&= (1 - \frac{\alpha}{h})(\psi_1 - \psi_2) + \frac{\alpha}{h}\left(\mathcal{T}_h\psi_1 - \mathcal{T}_h\psi_2\right).
\end{aligned}$$

Taking the supremum norm gives

$$\begin{aligned}
\|\mathcal{T}_{h,\alpha}\psi_1 - \mathcal{T}_{h,\alpha}\psi_2\|_\infty &\le \left|1 - \frac{\alpha}{h}\right|\|\psi_1 - \psi_2\|_\infty + \frac{\alpha}{h}\|\mathcal{T}_h\psi_1 - \mathcal{T}_h\psi_2\|_\infty \\
&\le \left(\left|1 - \frac{\alpha}{h}\right| + \frac{\alpha}{h}e^{-c_0 h}\right)\|\psi_1 - \psi_2\|_\infty.
\end{aligned}$$

Thus, $\mathcal{T}_{h,\alpha}$ is Lipschitz with constant

$$L_h := \left|1 - \frac{\alpha}{h}\right| + \frac{\alpha}{h}e^{-c_0 h}.$$

It remains to verify that $L_h \in (0,1)$. We distinguish two cases. If $0 < \alpha \le h$, then

$$L_h = 1 - \frac{\alpha}{h} + \frac{\alpha}{h}e^{-c_0 h} = 1 - \frac{\alpha}{h}(1 - e^{-c_0 h}).$$

Since $0 < e^{-c_0 h} < 1$ and $0 < \alpha/h \le 1$, it follows that $0 < L_h < 1$. If $\alpha > h$, then

$$L_h = \frac{\alpha}{h} - 1 + \frac{\alpha}{h}e^{-c_0 h} = \frac{\alpha}{h}(1 + e^{-c_0 h}) - 1.$$

In this case, $L_h > 0$, while

$$L_h < 1$$

is equivalent to

$$\frac{\alpha}{h}(1 + e^{-c_0 h}) < 2, \quad \text{or} \quad \alpha < \frac{2h}{1 + e^{-c_0 h}}.$$

Consequently,

$$0 < \alpha < \frac{2h}{1 + e^{-c_0 h}} \quad \text{implies} \quad L_h \in (0, 1),$$

and $\mathcal{T}_{h,\alpha}$ is a strict contraction on $\mathcal{C}_\varphi$. This proves (i).

Proof of (ii). By the Feynman–Kac Theorem 3.1, together with the Markov property and the tower property of conditional expectation, the solution $u \in \mathcal{U} \cap \mathcal{C}$ of (10)–(11) satisfies

$$u = \mathcal{T}_h u.$$

Substituting this identity into (18) gives

$$\mathcal{T}_{h,\alpha} u = u + \frac{\alpha}{h}(\mathcal{T}_h u - u) = u,$$

thus u is a fixed point of $\mathcal{T}_{h,\alpha}$. To prove uniqueness, let $\psi \in \mathcal{C}_\varphi$ be any fixed point of $\mathcal{T}_{h,\alpha}$. Then

$$\psi = \psi + \frac{\alpha}{h}\left(\mathcal{T}_h \psi - \psi\right).$$

Since $\alpha/h > 0$, this implies

$$\mathcal{T}_h \psi = \psi.$$

Therefore, the fixed points of $\mathcal{T}_{h,\alpha}$ coincide with those of $\mathcal{T}_h$. By Proposition 3.2 (ii), the operator $\mathcal{T}_h$ is a strict contraction on the complete metric space $\mathcal{C}_\varphi$. It therefore has at most one fixed point by Banach's fixed-point theorem. Since u is such a fixed point, we conclude that

$$\psi = u.$$

Hence, u is the unique fixed point of $\mathcal{T}_{h,\alpha}$ in $\mathcal{C}_\varphi$. □

B.5 Proof of Proposition 3.4

Proof. Fix $h > 0$. We first derive an integral representation of $\mathcal{T}_h \psi - \psi$. For every $y = (t, x) \in \bar{D}_T$, the discounted Itô–Dynkin formula gives

$$\mathcal{S}_h \psi(y) - \psi(y) = \mathbb{E}_y \left[\int_0^{\tau_h(t)} e^{-\int_0^s c(Y_r^y)\, dr} \big(\mathcal{A}\psi - f\big)(Y_s^y)\, ds \right].$$

Using $\mathcal{T}_h \psi = \mathcal{S}_h \psi + \mathcal{R}_h$, we therefore obtain

$$\mathcal{T}_h \psi(y) - \psi(y) = \mathbb{E}_y \left[\int_0^{\tau_h(t)} e^{-\int_0^s c(Y_r^y)\, dr} (\mathcal{A}\psi)(Y_s^y)\, ds \right].$$

Since $\mathcal{A}\psi \in \mathcal{C}_0$, for every $y = (t, x) \in \bar{D}_T$ and $s \geq T - t$, we have

$$\mathcal{S}_s(\mathcal{A}\psi)(y) = \mathbb{E}_y \left[e^{-\int_0^{T-t} c(Y_r^y)\, dr} (\mathcal{A}\psi)(T, X_T^{t,x}) \right] = 0.$$

Therefore,

$$\begin{aligned} \mathcal{T}_h \psi(y) - \psi(y) &= \int_0^{\tau_h(t)} \mathbb{E}_y \left[e^{-\int_0^s c(Y_r^y)\, dr} (\mathcal{A}\psi)(Y_s^y) \right] ds \\ &= \int_0^{\tau_h(t)} \mathcal{S}_s(\mathcal{A}\psi)(y)\, ds = \int_0^h \mathcal{S}_s(\mathcal{A}\psi)(y)\, ds, \end{aligned}$$

where Fubini's theorem applies because $\mathcal{A}\psi$ is bounded and the discount factor is at most one. Hence,

$$\mathcal{T}_h \psi - \psi = \int_0^h \mathcal{S}_s(\mathcal{A}\psi)\, ds \qquad \text{on } \bar{D}_T.$$

Since $\mathcal{A}\psi \in D(\mathcal{A}_0^n)$, the standard Taylor expansion for the strongly continuous semigroup $(\mathcal{S}_s)_{s\geq 0}$ gives

$$\mathcal{S}_s(\mathcal{A}\psi) = \sum_{j=0}^{n-1} \frac{s^j}{j!}\, \mathcal{A}_0^j(\mathcal{A}\psi) + \frac{1}{(n-1)!}\int_0^s (s-r)^{n-1}\mathcal{S}_r\mathcal{A}_0^n(\mathcal{A}\psi)\, dr.$$

Integrating from 0 to h gives

$$\begin{aligned}\mathcal{T}_h\psi &= \psi + \sum_{j=0}^{n-1}\frac{1}{j!}\left(\int_0^h s^j\, ds\right)\mathcal{A}_0^j(\mathcal{A}\psi) + \frac{1}{(n-1)!}\int_0^h\int_0^s (s-r)^{n-1}\mathcal{S}_r\mathcal{A}_0^n(\mathcal{A}\psi)\, dr\, ds \\ &= \psi + \sum_{j=1}^{n}\frac{h^j}{j!}\,\mathcal{A}_0^{j-1}(\mathcal{A}\psi) + R_{n+1}(h),\end{aligned}$$

where, by Fubini's theorem,

$$\begin{aligned}R_{n+1}(h) &:= \frac{1}{(n-1)!}\int_0^h\int_0^s (s-r)^{n-1}\mathcal{S}_r\mathcal{A}_0^n(\mathcal{A}\psi)\, dr\, ds \\ &= \frac{1}{n!}\int_0^h (h-r)^n\mathcal{S}_r\mathcal{A}_0^n(\mathcal{A}\psi)\, dr.\end{aligned}$$

By strong continuity, the integrands are continuous with values in $\mathcal{C}$, which justifies the Bochner integrals and the preceding use of Fubini's theorem. In particular,

$$R_{n+1}(h) \in \mathcal{C}.$$

Moreover, $\mathcal{S}_r$ is nonexpansive on $\mathcal{C}$:

$$\|\mathcal{S}_r\Phi\|_\infty \leq \|\Phi\|_\infty, \qquad \Phi \in \mathcal{C}.$$

Consequently,

$$\begin{aligned}\|R_{n+1}(h)\|_\infty &\leq \frac{1}{n!}\int_0^h (h-r)^n \left\|\mathcal{S}_r\mathcal{A}_0^n(\mathcal{A}\psi)\right\|_\infty dr \\ &\leq \frac{\left\|\mathcal{A}_0^n(\mathcal{A}\psi)\right\|_\infty}{n!}\int_0^h (h-r)^n\, dr \\ &= \frac{h^{n+1}}{(n+1)!}\left\|\mathcal{A}_0^n(\mathcal{A}\psi)\right\|_\infty.\end{aligned}$$

Thus,

$$\|R_{n+1}(h)\|_\infty = \mathcal{O}(h^{n+1}) \qquad \text{as } h \downarrow 0,$$

which proves (19) and its remainder estimate.

Dividing the preceding expansion by h gives

$$\frac{\mathcal{T}_h\psi - \psi}{h} = \sum_{j=1}^{n}\frac{h^{j-1}}{j!}\,\mathcal{A}_0^{j-1}(\mathcal{A}\psi) + \widetilde{R}_n(h),$$

where

$$\widetilde{R}_n(h) := \frac{R_{n+1}(h)}{h} \in \mathcal{C}.$$

The preceding remainder estimate yields

$$\|\widetilde{R}_n(h)\|_\infty \leq \frac{h^n}{(n+1)!}\left\|\mathcal{A}_0^n(\mathcal{A}\psi)\right\|_\infty = \mathcal{O}(h^n) \qquad \text{as } h \downarrow 0.$$

Finally, for $\psi \in \mathcal{C}_\varphi$, using the definition (18), we obtain for every $\alpha > 0$,

$$\begin{aligned}\mathcal{T}_{h,\alpha}\psi &= \psi + \frac{\alpha}{h}\,(\mathcal{T}_h\psi - \psi) \\ &= \psi + \alpha\left[\sum_{j=1}^{n}\frac{h^{j-1}}{j!}\,\mathcal{A}_0^{j-1}(\mathcal{A}\psi) + \widetilde{R}_n(h)\right].\end{aligned}$$

This proves (20). □

We also note that, for a classical solution $u \in \mathcal{U} \cap \mathcal{C}_\varphi$ of (10)–(11), the function $\mathcal{A}u$ vanishes on D_T and admits the zero extension to $\bar{D}_T$. Consequently,

$$\mathcal{A}u \in D(\mathcal{A}_0^n) \cap \mathcal{C}_0, \qquad \mathcal{A}_0^j(\mathcal{A}u) = 0, \qquad j = 0, 1, \dots, n.$$

Hence, all the explicit correction terms in (20), as well as the integral remainder $R_{n+1}(h)$, vanish when $\psi = u$. This gives the exact fixed-point identities

$$\mathcal{T}_h u = u \qquad \text{and} \qquad \mathcal{T}_{h,\alpha} u = u.$$

Under the condition

$$0 < \alpha < \frac{2h}{1 + e^{-c_0 h}},$$

this fixed point is unique by Proposition 3.3.

B.6 Proof of Theorem 3.5

Proof. Set $u_0 := u_{\theta^{(k)}}$. We first justify the inner recursion and the continuous extensions of the functions $\mathcal{A}u_i$.

By Lemma 2.2 and the square-integrability assumption, for each $i = 0, \dots, n-1$, the regression loss

$$\mathscr{L}^{(k,i)}(v; \xi) := \mathbb{E}\left[\Big(v(Y) - \mathcal{G}_\xi(Y, E_1, u_i)\Big)^2\right]$$

is minimized over all square-integrable measurable functions of Y by the conditional expectation $\mathbb{E}[\mathcal{G}_\xi(Y, E_1, u_i) \mid Y]$. By the exact representability assumption, the hPINN minimizer therefore satisfies

$$u_{i+1}(y) = \mathbb{E}[\mathcal{G}_\xi(y, E_1, u_i)] = u_i(y) + \xi \mathcal{A}u_i(y), \qquad \mu\text{-a.e. on } D_T. \tag{39}$$

Under the coefficient assumptions of Theorem 3.1, the differential and jump expressions defining $\mathcal{A}u_i$ are continuous on D_T. The full support of μ therefore implies that this identity holds on all of D_T. Consequently, $\mathcal{A}u_i$ admits the continuous extension

$$\mathcal{A}u_i = \frac{u_{i+1} - u_i}{\xi} \qquad \text{on } \bar{D}_T.$$

Since $u_i, u_{i+1} \in \mathcal{U} \cap \mathcal{C}_\varphi$ and the jump operator is linear, this also gives

$$\mathcal{A}u_i \in \mathcal{U} \cap \mathcal{C}_0, \qquad i = 0, \dots, n-1.$$

Hence,

$$u_{i+1} = u_i + \xi \mathcal{A}u_i \qquad \text{on } \bar{D}_T, \qquad i = 0, \dots, n-1.$$

We next prove by induction that

$$\mathcal{A}u_i \in D(\mathcal{A}_0^{n-i}) \cap \mathcal{C}_0, \qquad i = 0, \dots, n-1.$$

The claim holds for $i = 0$ by assumption. Suppose that it holds for some $i = 0, \dots, n-2$. By the linearity of the differential and jump terms, (39) gives

$$\partial_t(\mathcal{A}u_i) + \mathcal{L}[\mathcal{A}u_i] + \mathcal{I}[\mathcal{A}u_i] - c\,\mathcal{A}u_i = \frac{\mathcal{A}u_{i+1} - \mathcal{A}u_i}{\xi} \qquad \text{on } D_T.$$

The right-hand side belongs to $\mathcal{C}$ by the preceding extension argument. Since $\mathcal{A}u_i \in D(\mathcal{A}_0) \cap \mathcal{U}$, the generator identification in Proposition 3.2 (iii) therefore yields

$$\mathcal{A}u_{i+1} = \mathcal{A}u_i + \xi \mathcal{A}_0(\mathcal{A}u_i) = (I + \xi \mathcal{A}_0)(\mathcal{A}u_i) \qquad \text{on } \bar{D}_T.$$

The right-hand side belongs to $D(\mathcal{A}_0^{n-1-i})$, and $\mathcal{A}u_{i+1} \in \mathcal{C}_0$ was already established. This completes the induction and justifies the operator powers used below. In particular,

$$\mathcal{A}u_i = (I + \xi \mathcal{A}_0)^i(\mathcal{A}u_0), \qquad i = 0, \dots, n-1.$$

Iterating (39) n times from $u_0 = u_{\theta^{(k)}}$ yields

$$u_n = u_0 + \xi \sum_{m=0}^{n-1} (I + \xi \mathcal{A}_0)^m (\mathcal{A}u_0). \tag{40}$$

By the hockey-stick identity,

$$\xi \sum_{m=0}^{n-1} (I + \xi \mathcal{A}_0)^m (\mathcal{A} u_0) = \sum_{j=1}^{n} \binom{n}{j} \xi^j \mathcal{A}_0^{j-1} (\mathcal{A} u_0).$$

Substituting into (40) gives

$$u_n = u_0 + \sum_{j=1}^{n} \binom{n}{j} \xi^j \mathcal{A}_0^{j-1} (\mathcal{A} u_0).$$

Define

$$p_{n,j} := \prod_{m=0}^{j-1} \left(1 - \frac{m}{n}\right) \in [0,1], \qquad j = 1, \dots, n.$$

Since $\xi = h/n$, we have

$$\binom{n}{j} \xi^j = \binom{n}{j} \left(\frac{h}{n}\right)^j = \frac{h^j}{j!} p_{n,j}.$$

Consequently, we can rewrite u_n as

$$\begin{aligned} u_n &= u_0 + \sum_{j=1}^{n} \frac{h^j}{j!} \mathcal{A}_0^{j-1} (\mathcal{A} u_0) + \sum_{j=1}^{n} \frac{h^j}{j!} (p_{n,j} - 1) \mathcal{A}_0^{j-1} (\mathcal{A} u_0) \\ &= u_0 + \sum_{j=1}^{n} \frac{h^j}{j!} \mathcal{A}_0^{j-1} (\mathcal{A} u_0) + E_n(h), \end{aligned} \tag{41}$$

where

$$E_n(h) := \sum_{j=1}^{n} \frac{h^j}{j!} (p_{n,j} - 1)\, \mathcal{A}_0^{j-1} (\mathcal{A} u_0)\,. \tag{42}$$

The domain assumption on $\mathcal{A} u_0$ implies that every term in this finite sum belongs to $\mathcal{C}$, so $E_n(h) \in \mathcal{C}$.

It remains to estimate $E_n(h)$. Since $p_{n,1} = 1$, the term $j = 1$ vanishes. Moreover, for any finite family $(x_m)_m \subset [0,1]$,

$$1 - \prod_m (1 - x_m) \le \sum_m x_m.$$

Applying this with $x_m = m/n$ gives, for $j = 1, \dots, n$,

$$1 - p_{n,j} = 1 - \prod_{m=0}^{j-1} \left(1 - \frac{m}{n}\right) \le \sum_{m=0}^{j-1} \frac{m}{n} = \frac{j(j-1)}{2n}.$$

Therefore,

$$\begin{aligned} \|E_n(h)\|_\infty &\le \sum_{j=1}^{n} \frac{h^j}{j!} (1 - p_{n,j}) \left\| \mathcal{A}_0^{j-1} (\mathcal{A} u_0) \right\|_\infty \\ &\le \frac{1}{2n} \sum_{j=2}^{n} \frac{h^j}{j!} j(j-1) \left\| \mathcal{A}_0^{j-1} (\mathcal{A} u_0) \right\|_\infty \\ &= \frac{h^2}{2n} \sum_{j=2}^{n} \frac{h^{j-2}}{(j-2)!} \left\| \mathcal{A}_0^{j-1} (\mathcal{A} u_0) \right\|_\infty. \end{aligned}$$

This proves (25). When $n = 1$, the sum is empty and $E_1(h) = 0$. Since n is fixed and all the norms in the sum are finite,

$$\|E_n(h)\|_\infty = \mathcal{O}\left(\frac{h^2}{n}\right), \qquad h \downarrow 0.$$

We now insert (41) into the relaxed update (23). Since $u_0 = u_{\theta^{(k)}}$, we obtain

$$\begin{aligned} u^{(k+1)} &= \left(1 - \frac{\alpha}{h}\right) u_0 + \frac{\alpha}{h} u_n \\ &= \left(1 - \frac{\alpha}{h}\right) u_0 + \frac{\alpha}{h} \left[u_0 + \sum_{j=1}^{n} \frac{h^j}{j!} \mathcal{A}_0^{j-1}(\mathcal{A}u_0) + E_n(h) \right] \\ &= u_0 + \alpha \sum_{j=1}^{n} \frac{h^{j-1}}{j!} \mathcal{A}_0^{j-1}(\mathcal{A}u_0) + \frac{\alpha}{h} E_n(h), \end{aligned}$$

which is exactly (24) on $\bar{D}_T$.

Finally, the expansion (20) from Proposition 3.4, evaluated at $\psi = u_0$, gives

$$\mathcal{T}_{h,\alpha} u_0 = u_0 + \alpha \sum_{j=1}^{n} \frac{h^{j-1}}{j!} \mathcal{A}_0^{j-1}(\mathcal{A}u_0) + \alpha \widetilde{R}_n(h),$$

where

$$\widetilde{R}_n(h) = \frac{R_{n+1}(h)}{h}, \qquad \|\widetilde{R}_n(h)\|_\infty = \mathcal{O}(h^n).$$

Comparing with (24) yields

$$\begin{aligned} \left\| u^{(k+1)} - \mathcal{T}_{h,\alpha} u_{\theta^{(k)}} \right\|_\infty &\le \frac{\alpha}{h} \|E_n(h)\|_\infty + \alpha \|\widetilde{R}_n(h)\|_\infty \\ &= \mathcal{O}\left(\frac{\alpha h}{n}\right) + \mathcal{O}(\alpha h^n), \end{aligned}$$

as $h \downarrow 0$ with n fixed, which proves (26).

If $\alpha = h$, then (23) reduces to

$$u^{(k+1)} = u_n,$$

and $\mathcal{T}_{h,\alpha} = \mathcal{T}_h$. In this case, comparing (41) with the expansion (19) from Proposition 3.4, namely

$$\mathcal{T}_h u_0 = u_0 + \sum_{j=1}^{n} \frac{h^j}{j!} \mathcal{A}_0^{j-1}(\mathcal{A}u_0) + R_{n+1}(h),$$

gives

$$\begin{aligned} \left\| u^{(k+1)} - \mathcal{T}_h u_{\theta^{(k)}} \right\|_\infty &\le \|E_n(h)\|_\infty + \|R_{n+1}(h)\|_\infty \\ &= \mathcal{O}\left(\frac{h^2}{n}\right) + \mathcal{O}(h^{n+1}), \end{aligned}$$

as $h \downarrow 0$ with n fixed. This proves (27) and completes the proof. □

C Additional theoretical results

C.1 Jump-induced variance and population minimizers

We quantify the jump-induced variance of the INEUS target and examine how Monte Carlo noise affects the population objectives of INEUS and residual-based PINNs. For INEUS, replacing the jump expectation by a sample average preserves the population minimizers of each regression problem, while for a standard PINN loss, squaring the sampled residual introduces an additional variance term that alters the population minimizers.

Proposition C.1 (Variance of the single-jump INEUS target)**.** *Fix $u \in \mathcal{U}$, $y = (t, x) \in D_T$ and $m \ge 1$. Assume that*

$$\ell(u(t, x + \gamma(t, x, E)) - u(t, x))$$

has finite variance under $E \sim \nu$. Let

$$\widehat{I}_m[u](y) := \lambda(y) \frac{1}{m} \sum_{j=1}^{m} \ell(u(t, x + \gamma(t, x, E_j)) - u(t, x)), \qquad E_1, \dots, E_m \overset{\text{i.i.d.}}{\sim} \nu,$$

be the m-sample Monte Carlo estimator of the nonlocal term $I[u](y)$, and let

$$\mathcal{G}_\xi(y, E_1, u) = u(y) + \xi \left[\partial_t u(y) + \mathcal{F}(y, u, \nabla_x u, \nabla_x^2 u) + \lambda(y)\ell(u(t, x + \gamma(t, x, E_1)) - u(t, x))\right] \tag{43}$$

be the single-jump INEUS target. Then, conditionally on y,

$$\mathrm{Var}_\nu(\mathcal{G}_\xi(y, E_1, u)) = \xi^2 m \ \mathrm{Var}_\nu\Big(\widehat{I}_m[u](y)\Big).$$

Proof. For fixed y, the only randomness in both estimators comes from the jump samples. Since $E_1, \ldots, E_m$ are independent and identically distributed,

$$\begin{aligned}\mathrm{Var}_\nu\Big(\widehat{I}_m[u](y)\Big) &= \lambda^2(y)\,\mathrm{Var}_\nu\left(\frac{1}{m}\sum_{j=1}^m \ell(u(t, x + \gamma(t, x, E_j)) - u(t, x))\right) \\ &= \frac{\lambda^2(y)}{m}\,\mathrm{Var}_\nu(\ell(u(t, x + \gamma(t, x, E)) - u(t, x)))\,.\end{aligned}$$

Moreover, all other terms in $\mathcal{G}_\xi(y, E_1, u)$ are deterministic for fixed y, and hence

$$\mathrm{Var}_\nu(\mathcal{G}_\xi(y, E_1, u)) = \xi^2\lambda^2(y)\,\mathrm{Var}_\nu(\ell(u(t, x + \gamma(t, x, E)) - u(t, x)))\,.$$

The result follows immediately. □

In particular, for $\xi = h/n = 0.5/20$, we obtain

$$\xi = 0.025, \qquad \xi^2 = 6.25 \times 10^{-4}.$$

Hence, when $m = 64$, $\xi^2 m = 0.04 < 1$, and therefore

$$\mathrm{Var}_\nu\big(\mathcal{G}_\xi(y, E_1, u)\big) < \mathrm{Var}_\nu\big(\widehat{I}_m[u](y)\big).$$

Thus, despite relying on a single jump sample, the INEUS target has substantially lower jump-induced variance than the 64-point Monte Carlo estimator of the nonlocal term appearing in a standard PINN residual, thanks to the small scaling factor ξ. This comparison alone, however, does not imply greater statistical efficiency since the scaling parameter ξ reduces both the jump-induced variance and the size of the corresponding INEUS update.

A more fundamental distinction concerns the effect of jump sampling on the respective population objective. Fix $u \in \mathcal{U}$ and let $Y \sim \mu$ be independent of $\mathbf{E} = (E_1, \ldots, E_m) \sim \nu^{\otimes m}$. Define the m-sample INEUS target by

$$\mathcal{G}_{\xi,m}(y, \mathbf{E}, u) := \frac{1}{m}\sum_{j=1}^m \mathcal{G}_\xi(y, E_j, u). \tag{44}$$

This amounts to replacing the single-jump term in (43) by $\widehat{I}_m[u](y)$ and recovers the original single-jump target when $m = 1$. Lemma 2.2 directly implies that its conditional mean is again

$$\mathbb{E}[\mathcal{G}_{\xi,m}(Y, \mathbf{E}, u) \mid Y] = u(Y) + \xi\mathcal{A}[u](Y).$$

Consequently, the conditional bias-variance decomposition yields the following INEUS population objective over $v \in L^2(\mu)$,

$$\begin{aligned}&\mathbb{E}\Big[\big(v(Y) - \mathcal{G}_{\xi,m}(Y, \mathbf{E}, u)\big)^2\Big] \\ &= \mathbb{E}_\mu\Big[\big(v(Y) - u(Y) - \xi\mathcal{A}[u](Y)\big)^2\Big] + \frac{\xi^2}{m}\mathbb{E}_\mu\big[\lambda^2(Y)\,\mathrm{Var}_\nu\big(\ell(u(Y + \gamma(Y, E)) - u(Y))\big)\big]\,,\end{aligned} \tag{45}$$

where $\mathbb{E}_\mu = \mathbb{E}_{Y\sim\mu}$ and $\mathrm{Var}_\nu = \mathrm{Var}_{E\sim\nu}$. In particular, since u is fixed, the second term is independent of v and therefore, for every $m \geq 1$, does not affect the population minimizer of the INEUS regression problem. By contrast, for $v \in \mathcal{U}$, consider the m-sample PINN residual

$$\widehat{\mathcal{A}}_m[v](y) := \partial_t v(y) + \mathcal{F}(y, v, \nabla_x v, \nabla_x^2 v) + \widehat{I}_m[v](y).$$

Since $\mathbb{E}_\nu\big[\widehat{\mathcal{A}}_m[v](y)\big] = \mathcal{A}[v](y)$, the corresponding PINN population loss over $v \in \mathcal{U}$ satisfies

$$\mathbb{E}\Big[\widehat{\mathcal{A}}_m[v](Y)^2\Big] = \mathbb{E}_\mu\big[\mathcal{A}[v](Y)^2\big] + \frac{1}{m}\mathbb{E}_\mu\big[\lambda^2(Y)\,\mathrm{Var}_\nu(\ell(v(Y+\gamma(Y,E)) - v(Y)))\big]\,. \tag{46}$$

Hence, unlike in the INEUS frozen-target regression problem, the additional Monte Carlo variance term generally depends on the function v being optimized. Consequently, although $\widehat{\mathcal{A}}_m[v]$ is an unbiased estimator of $\mathcal{A}[v]$, its square is generally an upward-biased estimator of $\mathcal{A}[v]^2$. The resulting additional variance term therefore modifies the finite-m population objective and can alter the PINN population minimizer v^*.

Regression formulation vs. jump sampling. To assess the contribution of the regression-based formulation, we compare INEUS with a residual-based PINN using the same number m of jump samples per space-time point y. For INEUS, we replace the single-jump term in (43) by $\widehat{I}_m[u](y)$, giving the m-sample target $\mathcal{G}_{\xi,m}$ defined in (44); the case $m = 1$ recovers the original INEUS target. Using the same number of jump samples m in INEUS and PINN targets indeed allows us to control for the accuracy of the Monte Carlo approximation of the jump integral and therefore to isolate the effect of the regression-based INEUS formulation from that of the residual-based PINN. The new table below reports results for the linear PIDE (28) in $d = 50$ dimensions.

The results show that the regression-based INEUS formulation is particularly advantageous when the nonlocal term in $\mathcal{G}_{\xi,m}$ is approximated using only a small number of jump samples ($m = 1, 8$). This suggests that the INEUS regression target is substantially more robust than the PINN residual to the Monte Carlo noise induced by a coarse approximation of the jump integral. This behavior is consistent with (45) and (46): jump sampling leaves the population minimizers of each INEUS regression problem unchanged, whereas it introduces a function-dependent variance penalty into the PINN objective. For larger m, the PINN accuracy gradually improves and approaches that of INEUS, but at the expense of a much higher computational cost. Overall, INEUS remains both more accurate and considerably faster across all values of m. The table also shows that using between 1 and 16 jump samples provides a good compromise for INEUS between Monte Carlo variance reduction, accuracy and runtime.

Table 5: Comparison of INEUS and PINN for the linear PIDE (28) in dimension $d = 50$, using the same number of jump samples m per space–time point y. We report the mean and standard deviation of the MAE, together with the runtime in seconds.

	INEUS			PINN		
Jump samples m	Mean MAE	Std. dev.	Time (s)	Mean MAE	Std. dev.	Time (s)
1	0.0046	0.0009	5,867	0.0323	0.0029	9,067
8	0.0036	0.0008	6,053	0.0065	0.0013	9,693
16	0.0040	0.0006	6,107	0.0048	0.0009	10,211
32	0.0034	0.0004	6,451	0.0047	0.0012	11,184
64	0.0033	0.0003	7,529	0.0035	0.0007	13,739

C.2 Scalable implementation via directional differentiation

The next result from [13] further reduces the cost of evaluating the differential part of the operator $\mathcal{G}_\xi$ by replacing the explicit computation of gradients and Hessians of u_θ with a single directional second derivative. Indeed, the operator $\mathcal{G}_\xi$ requires the evaluation of the second-order differential operator $\mathcal{L}[u_\theta]$, which in principle involves $\partial_t u_\theta$, $\nabla_x u_\theta$, and $\nabla_x^2 u_\theta$ at each sample point (t, x).

Proposition C.2. *Let $u \in \mathcal{U}$ and $(t, x) \in D_T$ be given. Define the function $\Psi : \mathbb{R} \to \mathbb{R}$ by*

$$\Psi(l) := \sum_{i=1}^{q} u\left(t + \frac{l^2}{2q}, x + \frac{l}{\sqrt{2}}\sigma_i(t,x) + \frac{l^2}{2q}b(t,x)\right),$$

where $\sigma_i(t, x)$ is the ith column of the $d \times q$ matrix $\sigma(t, x)$. Then,

$$\Psi''(0) = \partial_t u(t,x) + b^\top(t,x)\,\nabla_x u(t,x) + \frac{1}{2}\mathrm{Tr}\left[\sigma\sigma^\top(t,x)\nabla_x^2 u(t,x)\right]$$

Proof. See Proposition 3.1 of [13]. □

Proposition C.2 makes it possible to replace the explicit computation of gradients and Hessians of u by evaluating the univariate function $\Psi''(0)$, the cost of which, using automatic differentiation, is a small multiple of $q \cdot cost(u)$ and hence avoids the $\mathcal{O}(d^2)$ scaling associated with explicitly forming the full Hessian $\nabla_x^2 u$.

The next lemma shows how the same idea can be exploited in the nonlinear PIDE (30) through the logarithmic transformation $v = e^{-\eta u}$.

Lemma C.3. *Assume that* $u(t,x) = -\log(v(t,x))/\eta$. *Then*

$$\partial_t u(t,x) + \Delta u(t,x) - \eta \|\nabla_x u(t,x)\|^2 = -\frac{1}{\eta}\,\frac{\partial_t v(t,x) + \Delta v(t,x)}{v(t,x)},$$

where $\Delta = \operatorname{Tr} \nabla_x^2$. *Moreover, if*

$$\Psi_v(l) := \sum_{i=1}^{d} v\left(t + \frac{l^2}{2d}, x + l e_i\right) = \sum_{i=1}^{d} \exp\left(-\eta\, u\left(t + \frac{l^2}{2d}, x + l e_i\right)\right),$$

then

$$\Psi_v''(0) = \partial_t v(t,x) + \Delta v(t,x),$$

and therefore

$$\partial_t u(t,x) + \Delta u(t,x) - \eta \|\nabla_x u(t,x)\|^2 = -\frac{1}{\eta}\,\frac{\Psi_v''(0)}{v(t,x)} = -\frac{e^{\eta u(t,x)}}{\eta}\,\Psi_v''(0).$$

Proof. Since $v = e^{-\eta u}$, we have

$$\partial_t v = -\eta v\, \partial_t u, \qquad \nabla_x v = -\eta v\, \nabla_x u,$$

and hence

$$\Delta v = -\eta v\, \Delta u + \eta^2 v\, \|\nabla_x u\|^2.$$

Dividing the identity

$$\partial_t v + \Delta v = -\eta v \left(\partial_t u + \Delta u - \eta \|\nabla_x u\|^2\right)$$

by $-\eta v$ yields

$$\partial_t u + \Delta u - \eta \|\nabla_x u\|^2 = -\frac{1}{\eta}\,\frac{\partial_t v + \Delta v}{v}.$$

Applying Proposition C.2 to v with $b \equiv 0$ and $\sigma = \sqrt{2}\, I_d$ gives

$$\Psi_v''(0) = \partial_t v + \Delta v.$$

Substituting this into the previous identity yields

$$\partial_t u + \Delta u - \eta \|\nabla_x u\|^2 = -\frac{1}{\eta}\,\frac{\Psi_v''(0)}{v} = -\frac{e^{\eta u}}{\eta}\,\Psi_v''(0),$$

which proves the claim. □

C.3 Setting generalizations

Polynomial-growth extension of the linear theory. The bounded-data assumption in Section 3 can be relaxed by replacing the ordinary supremum norm with a polynomially weighted supremum norm. For $p > 0$, define

$$V_p(x) := (1 + \|x\|^2)^{p/2}$$

and the spatial polynomial-growth space

$$\mathcal{C}^p := \left\{ \psi \in C(\mathbb{R}^d) : \|\psi\|_p := \sup_{x \in \mathbb{R}^d} \frac{|\psi(x)|}{V_p(x)} < \infty \right\}.$$

For $\beta > 0$, define the corresponding space–time space

$$\bar{\mathcal{C}}^{\beta,p} := \left\{ \psi \in C(\bar{D}_T) : \|\psi\|_{\beta,p} := \sup_{(t,x)\in\bar{D}_T} \frac{|\psi(t,x)|}{e^{\beta(T-t)}V_p(x)} < \infty \right\}.$$

Since

$$1 \le e^{\beta(T-t)} \le e^{\beta T},$$

this norm is equivalent, on the finite time interval, to the usual polynomially weighted supremum norm. We assume that

$$\varphi \in \mathcal{C}^p, \qquad f \in \bar{\mathcal{C}}^{\beta,p}.$$

We also assume the following property:

$$\mathcal{S}_h \bar{\mathcal{C}}^{\beta,p} \subseteq \bar{\mathcal{C}}^{\beta,p}, \qquad 0 \le h \le T-t.$$

More explicitly, this means that, whenever ψ is continuous and has growth controlled by V_p, the function

$$(t,x) \longmapsto \mathcal{S}_h\psi(t,x)$$

is also continuous and has growth controlled by V_p. The moment estimate below provides the growth bound, whereas preservation of continuity can follow e.g. from standard Lipschitz/growth condition on the jump diffusion coefficients or locally uniform moment bound of order $q > p$. Let

$$\mathcal{K}_t := \mathcal{L}_t + \mathcal{I}_t$$

denote the diffusion–jump generator without the killing term. We assume that $\mathcal{K}_t V_p$ is well defined and for some $\kappa_p \ge 0$,

$$\mathcal{K}_t V_p(x) \le \kappa_p V_p(x).$$

By the standard Lyapunov estimate, obtained from the localized Dynkin formula and Gronwall's inequality,

$$\mathbb{E}_{t,x}\left[V_p(X_{t+s})\right] \le e^{\kappa_p s} V_p(x), \qquad 0 \le s \le T-t.$$

Choose $\beta > 0$ such that

$$\delta_p := c_0 + \beta - \kappa_p > 0.$$

We now define

$$\mathcal{C}^{\beta,p}_{\varphi} := \{\psi \in \bar{\mathcal{C}}^{\beta,p} : \psi(T,\cdot) = \varphi\}.$$

Let $\tau_h(t) := \min\{h, T-t\}$, as in Section 3. For $\psi_1, \psi_2 \in \mathcal{C}^{\beta,p}_{\varphi}$, consider first $t + h < T$. Then

$$\begin{aligned} |(\mathcal{S}_h\psi_1 - \mathcal{S}_h\psi_2)(t,x)| &\le e^{-c_0 h}\|\psi_1 - \psi_2\|_{\beta,p}\, e^{\beta(T-t-h)}\, \mathbb{E}_{t,x}[V_p(X_{t+h})] \\ &\le e^{-\delta_p h}\, e^{\beta(T-t)} V_p(x)\|\psi_1 - \psi_2\|_{\beta,p}. \end{aligned}$$

Dividing by $e^{\beta(T-t)}V_p(x)$ and taking the supremum yields

$$\|\mathcal{S}_h\psi_1 - \mathcal{S}_h\psi_2\|_{\beta,p} \le e^{-\delta_p h}\, \|\psi_1 - \psi_2\|_{\beta,p}.$$

If $t + h \ge T$, then $\tau_h(t) = T - t$ and

$$(\mathcal{S}_h\psi_1 - \mathcal{S}_h\psi_2)(t,x) = 0,$$

because $\psi_1(T,\cdot) = \psi_2(T,\cdot) = \varphi$. The preceding bound therefore holds throughout $\bar{D}_T$. The affine source term also has controlled polynomial growth. Using $c \ge c_0$, the weighted bound on f, and $\mathbb{E}_{t,x}[V_p(X_{t+s})] \le e^{\kappa_p s}V_p(x)$, we obtain

$$\begin{aligned} |\mathcal{R}_h(t,x)| &\le \int_0^{\tau_h(t)} e^{-c_0 s}\mathbb{E}_{t,x}\left[|f(t+s, X_{t+s})|\right] ds \\ &\le \|f\|_{\beta,p}\, e^{\beta(T-t)} V_p(x) \int_0^h e^{-(c_0+\beta-\kappa_p)s}\, ds. \end{aligned}$$

Dividing by $e^{\beta(T-t)}V_p(x)$, taking the supremum, and evaluating the integral gives

$$\|\mathcal{R}_h\|_{\beta,p} \le \frac{1 - e^{-\delta_p h}}{\delta_p}\|f\|_{\beta,p}.$$

Thus, $\mathcal{T}_h = \mathcal{R}_h + \mathcal{S}_h$ is an affine self-map of $C_\varphi^{\beta,p}$, and its linear part has contraction modulus $e^{-\delta_p h}$. The relaxed operator $\mathcal{T}_{h,\alpha}$ therefore remains a strict contraction whenever

$$0 < \alpha < \frac{2h}{1 + e^{-\delta_p h}}.$$

The preceding results admit the following weighted formulation. Define

$$D^{\beta,p}(\mathcal{A}_0) := \left\{ \psi \in \bar{\mathcal{C}}^{\beta,p} : \lim_{h \downarrow 0} \frac{\mathcal{S}_h \psi - \psi}{h} \text{ exists in } \bar{\mathcal{C}}^{\beta,p} \right\}.$$

In the generator characterization of Proposition 3.2, take $\mathcal{C} = \bar{\mathcal{C}}^{\beta,p}$ and require

$$\psi \in D^{\beta,p}(\mathcal{A}_0) \cap \mathcal{U}.$$

For $n \geq 1$, the expansions and remainder estimates in Proposition 3.4–Theorem 3.5 retain their form, with the assumptions interpreted as

$$\psi \in \mathcal{U} \cap \bar{\mathcal{C}}^{\beta,p}, \qquad \mathcal{A}\psi \in D^{\beta,p}(\mathcal{A}_0^n) \cap \mathcal{C}_0^{\beta,p}$$

in Proposition 3.4, and

$$u_0 \in \mathcal{U} \cap \mathcal{C}_\varphi^{\beta,p}, \qquad \mathcal{A}u_0 \in D^{\beta,p}(\mathcal{A}_0^n) \cap \mathcal{C}_0^{\beta,p}$$

in Theorem 3.5, where $\mathcal{C}_0^{\beta,p} := \{\psi \in \bar{\mathcal{C}}^{\beta,p} : \psi(T, \cdot) = 0\}$. All supremum-norm estimates are then understood in $\|\cdot\|_{\beta,p}$, and the contraction and relaxation estimates use $e^{-\delta_p h}$ in place of $e^{-c_0 h}$. The minimization objective in Theorem 3.5 is unchanged. In particular, taking $p = 2$ accommodates the quadratic terminal payoff of the linear experiment.

Use for nonlinear PIDEs. The convergence analysis in Section 3 is deliberately restricted to linear PIDEs. Extending our convergence result to nonlinear PIDEs requires additional nontrivial assumptions. Our linear analysis relies crucially on the fact that $(\mathcal{S}_h)_{h \geq 0}$ is a linear Feynman–Kac semigroup with linear infinitesimal generator $\mathcal{A}_0$. This allows us to expand $\mathcal{S}_h$ in powers of $\mathcal{A}_0$ in Proposition 3.4 and to show that the repeated INEUS minimizations $(I + \xi \mathcal{A}_0)^n$ reproduce this expansion in Theorem 3.5. Conversely, for a nonlinear operator $\mathcal{A}$ this composition cannot be expanded using the binomial theorem. Even if the nonlinear PIDE generates a well-posed solution operator $(\mathcal{P}_h)_{h \geq 0}$, its formal higher-order expansion generally contains terms such as

$$D\mathcal{A}(\psi)\big[\mathcal{A}(\psi)\big]$$

rather than powers $\mathcal{A}^j \psi$. Stability of the exact nonlinear solution operator therefore does not imply stability or contractivity of the explicit INEUS update $I + \xi \mathcal{A}$. Nevertheless, the two nonlinear examples considered in Section 4 possess problem-specific stability structures.

For the HJB PIDE (30), the transformation $v = e^{-\eta u}$ converts the exact nonlinear equation into a linear PIDE, yielding the representation in Lemma C.3. This gives an order-preserving and stable exact solution operator, although INEUS is trained directly on u, and the transformation does not commute with the explicit INEUS recursion.

For the nonlinear Black–Scholes PIDE, let $\mathcal{P}_h \psi$ denote the exact solution obtained by evolving the continuation value $\psi(t + h, \cdot)$ backward from time $t + h$ to time t. The reaction function

$$g(y) = -\big[(1 - \delta)Q(y) + R\big]y$$

is globally Lipschitz because Q is continuous and piecewise affine and is constant outside $[v^h, v^l]$. Consequently, under the standard moment and well-posedness assumptions

$$\|\mathcal{P}_h \psi_1 - \mathcal{P}_h \psi_2\|_{\beta,p} \leq e^{-(\beta - \kappa_p - L_g)h} \|\psi_1 - \psi_2\|_{\beta,p}$$

where L_g is the Lipschitz constant. Thus the exact evolution is contractive when $\beta > \kappa_p + L_g$. Its terminal payoff $\min_i x_i$ has linear growth and is covered by $p = 1$. These structures help explain the observed numerical stability, but they do not establish convergence of the explicit INEUS recursion.

D Additional numerical experiments

Linear PIDE with quadratic terminal condition. We use as parameters in the linear PIDE experiments: $T = 0.5$, $b = 1$, $c = 2$, $\Sigma = 0.28\,\mathbf{1}_{d\times q}$, $\lambda = 0.25$, $\Sigma_J = 0.4\,I_{d\times d}$, and a test set uniformly sampled from $[0, 0.5] \times [-1.5, 1.5]^d$. Table 6 reports the performance of INEUS for higher-dimensional linear PIDEs of the form (28). The results are obtained after $k_* = 1000$ training epochs and averaged over 10 independent runs of Algorithm 1. As the state dimension d increases from 10 to 150 (with fixed $q = 10$), the mean MAE remains small, while the computational time grows only moderately. These results provide empirical evidence of the computational scalability of INEUS. This is consistent with Proposition C.2 which allows the differential operator to be evaluated at a cost proportional to $q\,\mathrm{cost}(u_\theta)$, without explicitly forming the full spatial Hessian.

Table 6: Performance of INEUS for linear PIDEs of the form (28) as a function of the state dimension d, with fixed $q = 10$. Results are computed after $k_* = 1000$ training epochs and reported as means and standard deviations over 10 independent runs of Algorithm 1.

d	Mean MAE	Std. Dev. MAE	Mean Loss $\widehat{\mathscr{L}}^{(k_*)}$	Std. Dev. Loss $\widehat{\mathscr{L}}^{(k_*)}$	Time (sec)
10	0.00043	3.7×10^{-5}	0.10523	0.00334	4120
25	0.00106	1.4×10^{-4}	0.17669	0.00217	4225
50	0.00291	5.54×10^{-4}	0.22776	0.00315	4303
100	0.01529	6.63×10^{-3}	0.34239	0.01347	4687
150	0.04091	1.67×10^{-2}	0.56882	0.08724	5028

The following figure then confirms that the relaxed operator $\mathcal{T}_{h,\alpha}$ in (18) helps improve the stability properties of our recursive approach based on single-jump point sampling, compared with the standard Feynman–Kac operator $\mathcal{T}_h$ (i.e., $\alpha = h$). We also observe that this relaxed operator yields faster convergence in terms of epochs, even if performing the regression step on the target $u^{(k+K)}$ in (34) once every block of $K = 2$ epochs results in a higher overall computational cost. To improve stability without increasing runtime excessively, we then set $K = 10$ in the experiments of Section 4.

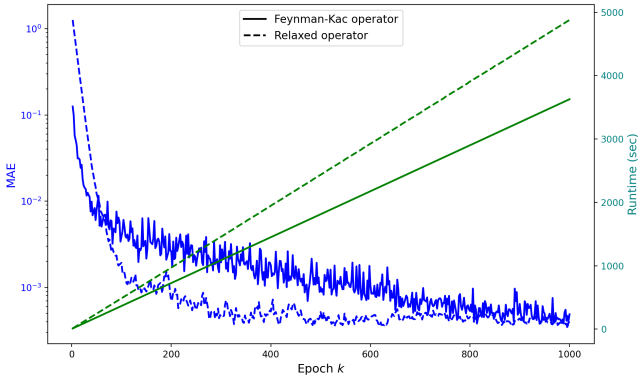

Figure 7: Comparison of INEUS using the Feynman–Kac operator $\mathcal{T}_h$ (solid line) and the relaxed operator $\mathcal{T}_{h,\alpha}$ (dotted line, $K = 2$) in terms of MAE (blue) and runtime in seconds (green), as functions of the number of epochs k for the linear PIDE (28).

Finally, we consider the linear PDE (28) without jumps ($\lambda = 0$). As shown in Figure 8, INEUS remains more efficient even in this purely local setting. The gain in accuracy is consistent with the contractive structure of the INEUS fixed-point update in the linear case, while the reduction in computational cost comes from the recursive formulation: the training step avoids backpropagation through the full PDE residual and therefore does not require the higher-order derivatives that arise in residual-based PINNs.

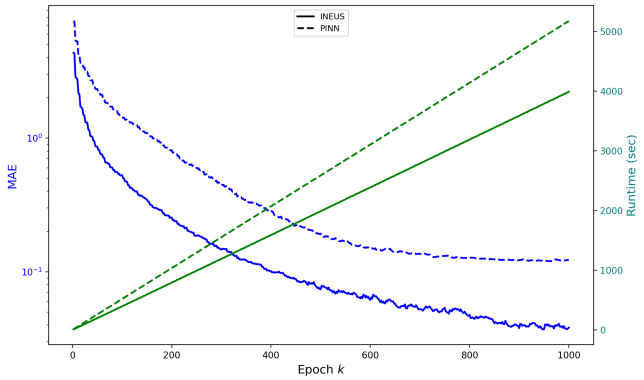

Figure 8: Comparison of MAE (blue) and runtime in seconds (green) of INEUS (solid lines) and PINNs (dashed lines) for a 100-dimensional linear PDE (28) without jumps ($\lambda = 0$).

Nonlinear Hamilton–Jacobi-Bellman PIDE. The parameters for the PIDE (30) are: $T = 1$, $\eta = 1$, $f(\cdot) \equiv 2$, $\lambda = 0.5$, $\mu_J = \mathbf{0}_d$, $\Sigma_J = 0.2 I_d$, and a test set uniformly sampled from $[0, 1] \times [-1.5, 1.5]^d$. Table 7 further confirms the scalability of INEUS as the dimension d increases, while Figure 9 illustrates its computational advantage over PINNs even for nonlinear PDEs (30) without nonlocal term ($\lambda = 0$). The reference solution is evaluated semi-analytically from the Feynman–Kac representation (31) by conditioning on the Poisson jump count. For the quadratic terminal condition and Gaussian jump sizes considered here, the resulting conditional Gaussian expectation admits a closed-form expression. We evaluate the remaining Poisson series up to 40 jumps using a numerically stable log-sum-exp formulation.

Table 7: Performance of INEUS for the nonlinear PIDE (30) as a function of the state dimension d. Results are computed after $k_* = 1000$ epochs and reported as means and standard deviations over 10 independent runs of Algorithm 1.

d	Mean MAE	Std. Dev. MAE	Mean Loss $\widehat{\mathscr{L}}^{(k_*)}$	Std. Dev. Loss $\widehat{\mathscr{L}}^{(k_*)}$	Time (sec)
5	0.0049	0.0007	0.0513	0.0029	3278
10	0.0158	0.0015	0.1299	0.0068	3289
25	0.0599	0.0185	0.3040	0.0853	4737
50	0.1284	0.0201	0.7561	0.1243	5772
100	0.2526	0.0277	0.8640	0.1053	10,422

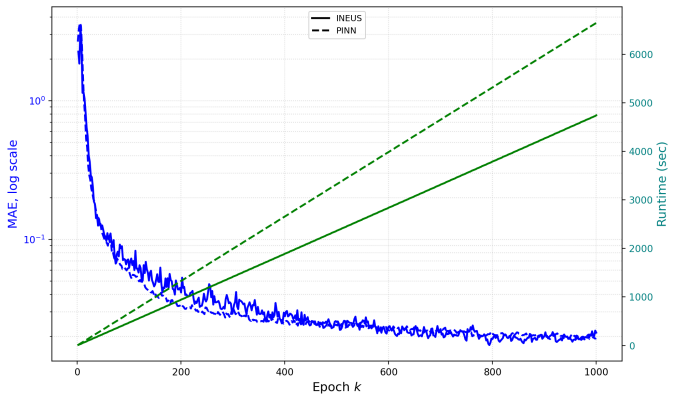

Figure 9: Comparison of MAE (blue) and runtime in seconds (green) of INEUS (solid lines) and PINNs (dashed lines) for a 100-dimensional nonlinear PDE (30) without jumps ($\lambda = 0$).

Linear Black–Scholes PIDE. We consider a financial derivative on a basket of d underlying stocks, whose Black–Scholes pricing PIDE is given on $D_T = [0, T) \times \mathbb{R}^d_+$ by

$$\partial_t u(t,x) + (r - \kappa \odot \lambda) \odot x \cdot \nabla_x u(t,x) + \frac{1}{2}\mathrm{Tr}[\mathrm{diag}(x)\sigma\Sigma\sigma^\top \mathrm{diag}(x)\nabla_x^2 u(t,x)] - ru(t,x) + \sum_{i=1}^{d} \lambda_i \mathbb{E}_\nu \left[u(t, x \odot e^{E_i e_i}) - u(t,x)\right] = 0\,, \tag{47}$$

with terminal condition $u(T,x) = (\sum_{i=1}^d w_i x_i - K)_+$, for $w_i \in [0,1]$ and $\sum_{i=1}^d w_i = 1$. We consider here i.i.d. normal random variables $E_i \sim \mathcal{N}(\mu_J^i, (\sigma_J^i)^2)$ with $\mu_J^i \in \mathbb{R}$, $\sigma_J^i \in \mathbb{R}_+$, and denote $\kappa = (\kappa_1, \dots, \kappa_d)$ where $\kappa_i := \mathbb{E}[e^{E_i} - 1]$, $\sigma = \mathrm{diag}(\sigma_1, \dots, \sigma_d)$, $\odot$ the component-wise multiplication, and e_i the i-th unit vector. The parameters in this setting are: $T = 1$, $r = 0.05$, $K = 30$, $\Sigma = 0.1\,\mathbf{1}_{d\times d} + 0.9\,I_d$ with $w_i = 1/d$, $\sigma_i = 0.15$, $\lambda_i = 0.5$, $\mu_J^i = 0.2$, $\sigma_J^i = 0.3$, $\kappa_i = \exp(\mu_J^i + \frac{1}{2}(\sigma_J^i)^2) - 1 = 0.278$ for $i = 1, \dots, d$, and a test set uniformly sampled from $[0,1] \times [0,100]^d$. The reference solution in Figures 10–12 and Table 8 is then approximated by Monte Carlo simulation based on the Feynman–Kac representation (13).

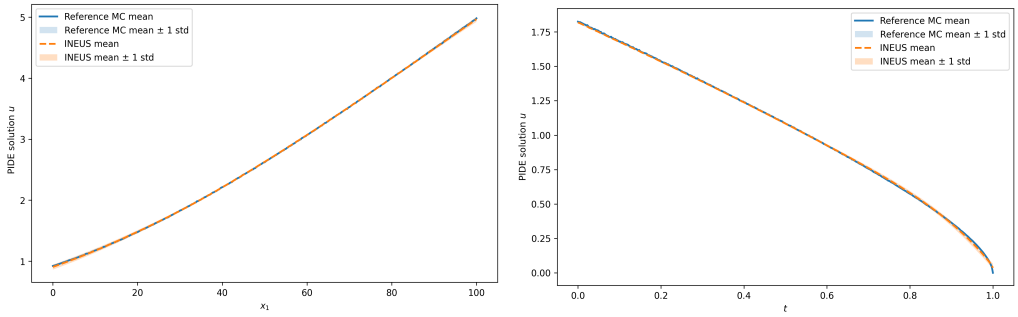

Figure 10: PIDE solution $u(0,x)$ for $x = (x_1, K, \dots, K)$ with $x_1 \in [0,100]$ (left) and $u(t,x)$ for $t \in [0,1]$ and $x = K\mathbf{1}_{20}$ (right) for a 20-dimensional linear Black–Scholes PIDE (47). Orange dotted lines: numerical results of INEUS with ± 1 standard deviation given by orange shaded area. Blue lines: reference MC solution obtained from the Feynman–Kac formula (13) with ± 1 standard deviation given by blue shaded area.

Table 8: Performance of INEUS for linear Black–Scholes PIDEs of the form (47) as a function of the state dimension d. Results are computed after $k_* = 1000$ training epochs and reported as means and standard deviations over 10 independent runs of Algorithm 1.

d	Mean MAE	Std. Dev. MAE	Mean Loss $\widehat{\mathscr{L}}^{(k_*)}$	Std. Dev. Loss $\widehat{\mathscr{L}}^{(k_*)}$	Time (sec)
5	0.0202	0.0031	0.3045	0.0167	3259
10	0.0224	0.0045	0.2749	0.0355	4049
25	0.0246	0.0036	0.1933	0.0200	8248
50	0.0257	0.0054	0.2305	0.0879	10,984
100	0.0294	0.0064	0.2277	0.0950	15,799

Table 8 again confirms the scalability of INEUS across dimensions. As the state dimension d increases from 5 to 100 (with the Brownian dimension increasing accordingly as $q = d$ in (47)), the mean MAE remains small, while the runtime increase remains moderate. This is consistent with the directional-differentiation trick C.2, whose differential-operator cost scales with q rather than with the $\mathcal{O}(d^2)$ complexity of the full Hessian.

Figure 11 then again compares INEUS performance with PINN and deep BSDE methods for the 10-dimensional linear Black–Scholes PIDE (where PINNs remain feasible). The left panel shows that INEUS reaches lower MAE values within significantly less runtime than the PINN approach, reflecting the efficiency gained by avoiding direct numerical integration of the jump term and the computation of higher-order derivatives. The right panel further shows that, when errors are compared as functions of runtime, INEUS attains the smallest MAE on the test set of randomly sampled space–time points. This suggests that INEUS provides a more accurate global approximation of the PIDE solution than the deep BSDE method with jumps, which is typically designed to approximate the solution along simulated trajectories with fixed initial point (t_0, x_0). However, despite the use of Polyak averaging with $K = 10$ in (34), the INEUS error curve displays larger fluctuations, indicating that its training dynamics remain less stable than those of the competing methods. Figure 12 shows

the corresponding results for a Black–Scholes PDE (47) without jumps ($\lambda = 0$).

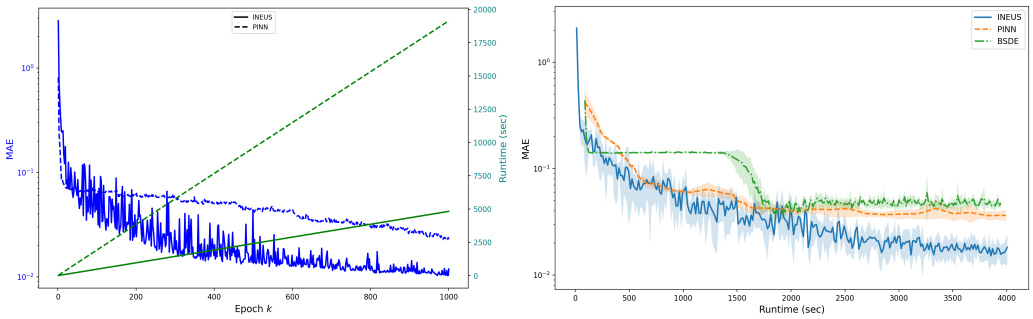

Figure 11: Left panel: comparison of MAE (blue) and runtime in seconds (green) of INEUS (solid lines) and PINN (dashed lines) as functions of the epoch k. Right panel: comparison of INEUS, PINN, and deep BSDE with jumps as functions of runtime. Both panels correspond to a 10-dimensional linear Black–Scholes PIDE (47).

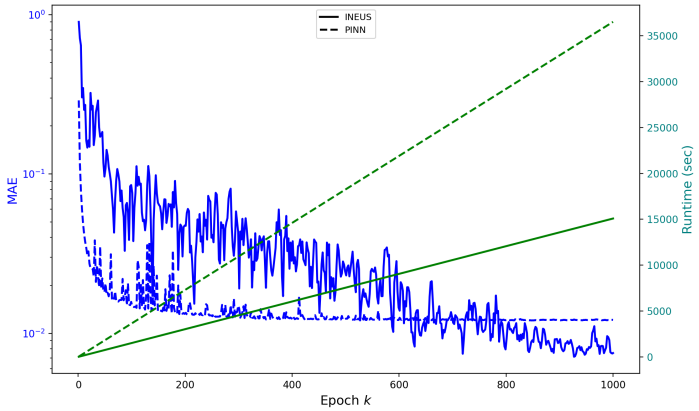

Figure 12: Comparison of MAE (blue) and runtime in seconds (green) of INEUS (solid lines) and PINNs (dashed lines) for a 100-dimensional linear Black–Scholes PDE (47) without jumps ($\lambda = 0$).

Nonlinear Black–Scholes PIDE with default risk. The piecewise-linear default intensity Q in the nonlinear Black–Scholes PIDE (32) is given by

$$Q(u) = \mathbb{1}_{(-\infty, v^h)}(u)\gamma^h + \mathbb{1}_{[v^l, \infty)}(u)\gamma^l + \mathbb{1}_{[v^h, v^l)}(u)\left[\frac{(\gamma^h - \gamma^l)}{(v^h - v^l)}\left(u - v^h\right) + \gamma^h\right], \tag{48}$$

with $v^h = 50, v^l = 70, \gamma^h = 0.2$, $\gamma^l = 0.02$. The parameters in (32) are set to $d = 100$, $T = 1$, $\beta = 0.02$, $\sigma = 0.20$, recovery rate $\delta = 2/3$, discount rate $R = 0.02$, intensity $\lambda = 0.1$, log-jump mean $\mu_J = -0.2$, log-jump volatility $\sigma_J = 0.15$, correlation $\rho_J = 0.5$, and initial condition $x_0 = 100 \cdot \mathbf{1}_{100}$. In the no-jump case, the reference value $u(0, x_0) = 57.300$ is obtained using the deep BSDE method of [26]. This method is designed to approximate the solution at a prescribed initial point and is thus known to give a reliable pointwise benchmark at $(0, x_0)$, but not a global reference solution over the full space-time domain (see Introduction and Appendix A.3). Similarly, in the jump case, the reference value $u(0, x_0) = 55.81$ is computed using the forward Picard scheme [8], based on the Feynman–Kac representation with 4×10^5 antithetic paths and 30 Picard iterations[5]. In Figure 5, we then assess convergence to the benchmark value $u(0, x_0)$, rather than reporting a global MAE since obtaining the reference solutions for the full test set on the whole domain would be computationally very intensive.

[5]This value was also verified using the backward regression method of Gobet et al. [23].

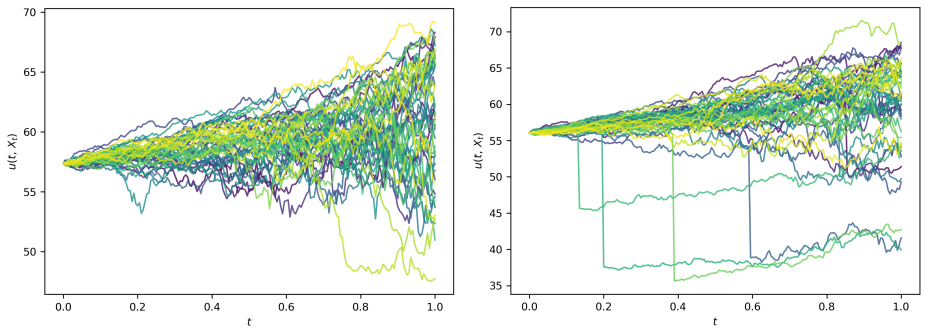

Figure 13: Learned solution $u_\theta(t, X_t)$ of the Black–Scholes nonlinear equation (32) without jumps (left, $\lambda = 0$) and with jumps (right, $\lambda = 0.1$). In both panels, the solution is evaluated along 50 independent (jump-)diffusion sample paths starting from $x_0 = 100 \cdot \mathbf{1}_{100}$. In the no-jump setting (left), trajectories start from $u_\theta(0, x_0) \approx 57.3$ and spread as the diffusion variance grows, consistent with the expected behavior of the min-payoff pricing function under default risk. In the jump setting (right), trajectories start from $u_\theta(0, x_0) \approx 55.81$ and exhibit additional discontinuities induced by the jump component ($\lambda = 0.1$), leading to increased variability.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Every claim in the abstract and introduction is either theoretically (Section 3) or numerically (Section 4) supported in the paper. Similarly, the *Conclusion, limitations, and future work* Section discusses the limitations and possible extensions of the current work.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The main limitation of our approach is that the convergence analysis is restricted to the linear case, where the contractive properties of the Feynman–Kac operator are available. From an algorithmic perspective, adaptive relaxation strategies and improved sampling schemes may further enhance stability and efficiency. We have emphasized these points in a separate part of the conclusion.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors

should reflect on how these assumptions might be violated in practice and what the implications would be.

- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: In the novel Proposition 2.1, Lemma 2.2, Proposition 3.2, Proposition 3.3, Proposition 3.4, Theorem 3.5, Proposition C.1 and Proposition C.2, all assumptions required to establish the results are clearly stated (or corresponding references provided). Similarly for the well-known Theorem 3.1, an exact reference is given. All the proofs are then available in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 3.3 describes the implementation of the proposed methodology and we provide the detailed pseudocode of Algorithm 1 in Appendix A. Further details on the architecture, hyperparameters, training scheme and baselines are given in Appendix A, enabling reproduction of the numerical results. The code is available at `https://github.com/davidegallon/INEUS`.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code is available at https://github.com/davidegallon/INEUS.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Please consult Section 3.3 and supplementary material in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All relevant figures are presented with ± 1 standard deviation over independent runs. This is stated every time needed and the computation of these confidence intervals is explained in Section 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Please consult the supplementary material in Appendix A.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This paper presents work whose goal is to advance the field of Machine Learning by proposing a new deep-learning approach for solving high-dimensional partial integro-differential equations. It has therefore no direct societal impact.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

Justification: The paper poses no such risks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [N/A]

Justification: We do not use any existing assets.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The paper does not release new assets.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve any crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve any crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: This research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.