A fully nonlinear Feynman-Kac formula with derivatives of arbitrary orders

Jiang Yu Nguwi* Guillaume Penent† Nicolas Privault‡

Division of Mathematical Sciences

School of Physical and Mathematical Sciences

Nanyang Technological University

21 Nanyang Link, Singapore 637371

January 27, 2023

Abstract

We present an algorithm for the numerical solution of nonlinear parabolic partial differential equations. This algorithm extends the classical Feynman-Kac formula to fully nonlinear partial differential equations, by using random trees that carry information on nonlinearities on their branches. It applies to functional, non-polynomial nonlinearities that are not treated by standard branching arguments, and deals with derivative terms of arbitrary orders. A Monte Carlo numerical implementation is provided.

Keywords: Fully nonlinear PDEs, quasilinear PDEs, semilinear PDEs, parabolic PDEs, gradient nonlinearities, branching processes, Monte-Carlo method.

Mathematics Subject Classification (2020): 35G20, 35K55, 35K58, 35B65, 60J85, 60G51, 65C05.

1 Introduction

The objective of this paper is to provide probabilistic representations for the solutions of fully nonlinear parabolic partial differential equations involving higher order derivatives, of the form

$$\begin{cases} \partial_t u(t,x) + \dfrac{1}{2}\partial_x^2 u(t,x) + f\big(u(t,x), \partial_x u(t,x), \ldots, \partial_x^n u(t,x)\big) = 0, \\ u(T,x) = \phi(x), \qquad (t,x) \in [0,T] \times \mathbb{R}, \end{cases} \tag{1.1}$$

*nguw0003@e.ntu.edu.sg

†pene0001@e.ntu.edu.sg

‡nprivault@ntu.edu.sg

$n \geqslant 0$, where ∂_x^2 is the standard Laplacian on $\mathbb{R}$ and $f(x, y, z_1, \ldots, z_m)$ is a smooth functional nonlinearity involving derivatives of arbitrary orders. Probabilistic representations for the solutions of first order nonlinear partial differential equations (PDEs) of the form

$$\begin{cases} \partial_t u(t,x) + \dfrac{1}{2}\partial_x^2 u(t,x) + f\big(t, x, u(t,x), \partial_x u(t,x)\big) = 0 \\ \\ u(T,x) = \phi(x), \qquad x \in \mathbb{R}^d, \end{cases}$$

can be obtained using backward stochastic differential equations (BSDEs) [Pen91, PP92], by representing $u(t,x)$ as $u(t,x) = Y_t^{t,x}$, $(t,x) \in [0,T] \times \mathbb{R}$, where $(Y_s^{t,x})_{t \leqslant s \leqslant T}$ is the solution of the backward stochastic differential equation

$$\begin{cases} dY_s^{t,x} = -f\big(s, X_s^{t,x}, Y_s^{t,x}, Z_s^{t,x}\big)ds + Z_s^{t,x} dX_s^{t,x}, \qquad 0 \leqslant t \leqslant s \leqslant T, \\ \\ Y_T^{t,x} = \phi(X_T^{t,x}), \end{cases}$$

with random anticipating terminal condition, where $(X_s^{t,x})_{t \leqslant s \leqslant T}$ is a standard Brownian motion started at $x \in \mathbb{R}$ at time $t \in [0,T]$. This method also extends to second order fully nonlinear PDEs of the form

$$\begin{cases} \partial_t u(t,x) + f\big(t, x, u(t,x), \partial_x u(t,x), \partial_x^2 u(t,x)\big) = 0 \\ u(T,x) = \phi(x), \qquad x \in \mathbb{R}^d, \end{cases}$$

using second order backward stochastic differential equations [CSTV07], [STZ12].

From a computational point of view, neural networks methods for fully nonlinear PDEs have been introduced in [HJE18] and in [SS18] using BSDEs and the Galerkin method, respectively. See also [BEJ19, BBC+19], and [HPW20, PWG21, LLP22], for related deep learning-based numerical algorithms.

On the other hand, stochastic diffusion branching mechanisms for the representation of solutions of partial differential equations have been introduced in [Sko64], and extended to branching Markov processes in [INW69]. Branching diffusions have also been applied to give a probabilistic representation of the solutions of the Kolmogorov-Petrovskii-Piskunov (KPP) equation in [McK75], and to more general PDEs with polynomial nonlinearities in [HL12, HLTT14], see also [CLM08] for existence of solutions of parabolic PDEs with power series nonlinearities.

In [HLOT+19], this branching argument has been applied to polynomial gradient nonlinearities using branching trees. In this approach, branches corresponding to gradient terms

are identified by marks and associated random weights which are used in Malliavin integration by parts, see also [HLT21] for an application to semilinear and higher-order hyperbolic PDEs. In [FTW11], see also [Tan13], [GZZ15], [KZZ15], [HLZ20], a finite difference scheme combined with Monte Carlo estimation has been introduced for fully nonlinear PDEs with gradients of order up to 2 using integration by parts.

Numerical solutions of semilinear PDEs have also been obtained by the multilevel Picard method [EHJK19, HJKN20, EHJK21, HJK22], with numerical experiments provided in [BBH+20], see also [NW22] for a treatment of nonlocal PDEs. However, this approach is currently restricted to first order gradient nonlinearities.

Extending those techniques to nonlinearities in higher order derivatives involves several technical difficulties. In the case of branching diffusion approaches, this involves a lack of integrability of the Malliavin-type weights used in repeated integration by parts argument, see page 199 of [HLOT+19]. This problem was also noted when dealing with pseudo-differential operators of the form $-\eta(\partial_x^2)$ for the treatment of nonlocal PDEs [PP22a].

In this paper, our method to deal with fully nonlinear PDEs of the form (1.1) relies on a marked branching process called a coding tree, represented by a random tree whose branches bear operators, called *codes*, instead of function values. A general multiplicative functional whose expected value provides a probabilistic representation of the PDE solution is then associated to the coding tree. To ensure the validity of the probabilistic representation we derive sufficient conditions on f and ϕ that ensure the finiteness of expected values.

Other probabilistic methods that can deal with higher order derivatives usually involve pseudo-processes created as limits of discrete random walks, see [BDM19]. However, the method developed in this paper is different, as instead of creating a specific process whose generator is behaving as a higher order derivative, we use codes that carry information on the branches along the tree. Once the tree leaves are reached, we make use of the code on the known terminal condition ϕ of the solution u.

The idea of carrying information on nonlinearities along trees is not new, and has been developed in the case of ordinary differential equations (ODEs) in [But63], see also, e.g., [But10], Chapters 4-6 of [DB02], [MMMKV17]. Butcher trees have found applications ranging from geometric numerical integration to stochastic differential equations, see for instance [HLW06] and references therein, and [Gub10], [BHZ19], [Fos21], for the use of decorated

trees for stochastic partial differential equations, and for their connections with the Butcher-Connes-Kreimer Hopf algebra [CK99].

In the approach of [But63], the general idea is to write a Taylor expansion for the solution of a differential equation, and to represent every term using a specific tree structure. In this case, numerical evaluation of the solution requires to truncate the series by selecting certain trees. On the other hand, the stochastic branching method does not rely on truncations and can be used to estimate an infinite series as an expected value over almost surely finite random trees. This approach has been applied in [PP22b] to the numerical estimation of ODE solutions by the Monte Carlo method without the use of diffusion processes. On the other hand, PDEs can be treated by this method by attaching a random Brownian evolution to each tree branch.

In this paper, we provide probabilistic representations for the solutions of a class of fully nonlinear parabolic PDEs of the form (1.1) with functional nonlinearity $f(z_0, \ldots, z_n)$ in the solution u and its derivatives $\partial_x^k u(t,x)$, $k = 1, \ldots, n$. In the sequel, we denote by

$$\varphi(t,x) := \frac{\mathrm{e}^{-x^2/(2t)}}{\sqrt{2\pi t}}, \qquad x \in \mathbb{R},$$

the standard Gaussian kernel with variance $t > 0$. We denote by $\mathcal{C}^k(\mathbb{R}^m)$ the set of k-times differentiable functions with continuous derivatives of orders up to $k \in \{0, \ldots, \infty\}$ on $\mathbb{R}^m$, and for any $h \in \mathcal{C}^k(\mathbb{R}^m)$ and $(\lambda_1, \ldots, \lambda_n) \in \{0, \ldots, k\}^m$ we use the notation

$$\partial_{z_1}^{\lambda_1} \cdots \partial_{z_m}^{\lambda_m} h(z_1, \ldots, z_m) := \frac{\partial^{\lambda_1}}{\partial z_1^{\lambda_1}} \cdots \frac{\partial^{\lambda_m}}{\partial z_m^{\lambda_m}} h(z_1, \ldots, z_m), \quad m \geqslant 1.$$

Similarly, we denote by $\mathcal{C}^{1,k}([0,T] \times \mathbb{R})$ functions $u(t,x)$ which are differentiable in time $t \in [0,T]$ and k times differentiable in $x \in \mathbb{R}$ with continuous partial derivatives, $1 \leqslant k \leqslant \infty$.

Assumption (A). *Assume that*

i) $f \in \mathcal{C}^\infty(\mathbb{R}^{n+1})$ *and* $\phi \in \mathcal{C}^\infty(\mathbb{R})$,

ii) the PDE (1.1) *admits a unique solution* $u \in \mathcal{C}^{1,\infty}([0,T] \times \mathbb{R})$, *written in integral or Duhamel formulation as*

$$\begin{aligned} u(t,x) &= \int_{-\infty}^{\infty} \varphi(T-t, y-x)\phi(y)dy \\ &\quad + \int_t^T \int_{-\infty}^{\infty} \varphi(s-t, y-x) f\left(u(s,y), \partial_y u(s,y), \ldots, \partial_y^n u(s,y)\right) dyds, \end{aligned} \tag{1.2}$$

iii) $\phi^{(k)}(u) \in \cap_{p=1}^{n+1} L^p(\mathbb{R}, \varphi(\eta,x)dx)$, $k \geqslant 0$, *and* $\partial_{z_0}^{\lambda_0} \cdots \partial_{z_n}^{\lambda_n} f(u, \partial_x u, \ldots, \partial_x^n u) \in \cap_{p=1}^{n+1} L^p([0,T] \times \mathbb{R}, \varphi(\eta,x)dxds)$, $(\lambda_0, \ldots, \lambda_n) \in \mathbb{N}^{n+1}$, *for all* $\eta > 0$.

We refer to e.g. Theorem 1.1 in [Kry83] for sufficient conditions for existence and uniqueness of smooth solutions to such fully nonlinear PDEs in the second order case.

Starting from (1.2), we will construct a random coding tree $\mathcal{T}_{t,x,c}$ rooted at (t,x) which is a random branching process driven by a standard Brownian motion $(W_t)_{t \in \mathbb{R}_+}$, with branches bearing operators called codes and indexed by a set $\mathfrak{C}$, such that the first branch of this tree bears the code $c = \mathrm{Id}$.

Next, we will construct a universal multiplicative functional $\mathcal{H}_\phi$ of $\mathcal{T}_{t,x,c}$, such that the expectation $u_c(t,x) := \mathbb{E}[\mathcal{H}_\phi(\mathcal{T}_{t,x,c})]$ solves the system of equations

$$\begin{cases} \partial_t u_c(t,x) + \dfrac{1}{2}\partial_x^2 u_c(t,x) + \displaystyle\sum_{Z \in \mathcal{M}(c)} \prod_{z \in Z} u_z(t,x) = 0, \quad c \in \mathfrak{C}, \\ u_c(T,x) = c(u)(T,x), \qquad (t,x) \in [0,T] \times \mathbb{R}, \end{cases} \tag{1.3}$$

where $\mathcal{M}$ is a mapping called the *mechanism*, which sends any code $c \in \mathfrak{C}$ to a family of code tuples $Z \in \mathcal{M}(c)$ which are associated to the new branches created in the random coding tree $\mathcal{T}_{t,x,c}$.

In Theorem 4.2, supposing in addition to Assumption (A) that the solution of the system (1.3) is unique, and given $T > 0$ such that the functional $\mathcal{H}(\mathcal{T}_{t,x,c})$ is integrable for all $(t,x) \in [0,T] \times \mathbb{R}$, we derive a probabilistic representation of the form

$$u(t,x) := \mathbb{E}\big[\mathcal{H}_\phi(\mathcal{T}_{t,x,\mathrm{Id}})\big], \quad (t,x) \in [0,T] \times \mathbb{R},$$

for the solution $u(t,x)$ of (1.1). Sufficient conditions on f, ϕ for the boundedness of the functional $\mathcal{H}(\mathcal{T}_{t,x,c})$ are derived Proposition 4.3 under additional conditions on the probability density function ρ of interbranching times in the random tree $\mathcal{T}_{t,x,c}$, over a sufficiently small time interval $[0,T]$.

In Section 5 we present a Monte Carlo implementation of our algorithm for the numerical solutions of fully nonlinear PDEs on a sufficiently small time interval. Numerical applications are presented to semilinear, quasilinear and fully nonlinear PDEs. This includes in particular functional nonlinearities which are not covered by standard branching methods that are designed for polynomial nonlinearities. We also deal with examples involving higher order

derivatives that may not be treated by Malliavin-type integration by parts arguments due to integrability issues, see page 199 of [HLOT+19], and are also not covered by multilevel Picard methods [BBH+20], or BSDE methods, see e.g. [HJE17, HJE18], which are limited to first and second order gradients, respectively.

Although our results are only valid in small time, the numerical experiments performed in Section 5 for the Allen-Cahn equation (5.1)-(5.2) and for the HJB equation (5.9), see Tables 2, 3, 4, 5 and Figures 3 and 5-*b*), show that the performance of our coding tree method compares favorably to those of the BSDE [HJE17, HJE18], branching diffusion [HLTT14], and MLP [EHJK19] methods. In addition, some of our fully nonlinear examples, see Examples 3-a) and 3-b), are currently out of reach by other methods.

This paper is organized as follows. Sections 2 and 3 present the constructions of codes, mechanisms, and random coding trees. In Section 4 we state our main result Theorem 4.2 which gives the probabilistic representation of the solution and its partial derivatives and give a sufficient condition that ensures the integrability needed for the probabilistic representation of Theorem 4.2 to hold. In Section 5, we present numerical simulations that illustrate the method on specific examples.

The appendix contains a Mathematica implementation of the algorithm of Theorem 4.2 in dimension one. The Python codes designed for other numerical experiments are available at `https://github.com/nguwijy/coding_trees`.

Preliminaries

For simplicity of exposition, Sections 2-4 are presented in the one-dimensional case of PDEs of a single space variable $x \in \mathbb{R}$, while the codes used in Section 5 are implemented in the d-dimensional setting. In the sequel we will use the following version of the multivariate Faà di Bruno formula, which follows from Theorem 2.1 in [CS96].

Proposition 1.1 *Let $n \geqslant 0$ and $k \geqslant 1$. Given $g \in \mathcal{C}^k(\mathbb{R}^{n+1})$ function of $(z_0, \dots, z_n)$ and $v \in \mathcal{C}^n([0,T] \times \mathbb{R})$ function of (t,x), we have*

$$\partial_x^k\big(g\big(v(t,x), \dots, \partial_x^n v(t,x)\big)\big)$$

$$= k! \sum_{\substack{1\leqslant\lambda_0+\cdots+\lambda_n\leqslant k\\ 1\leqslant s\leqslant k}} \partial_{z_0}^{\lambda_0}\cdots\partial_{z_n}^{\lambda_n} g\big(v(t,x),\ldots,\partial_x^n v(t,x)\big) \sum_{\substack{1\leqslant|\mathbf{k}_1|,\ldots,|\mathbf{k}_s|,\ 1\leqslant l_1<\cdots<l_s\\ k_1^i+\cdots+k_s^i=\lambda_i,\ 0\leqslant i\leqslant n\\ |\mathbf{k}_1|l_1+\cdots+|\mathbf{k}_s|l_s=k}} \prod_{\substack{1\leqslant j\leqslant s\\ 0\leqslant q\leqslant n}} \frac{\big(\partial_x^{q+l_j} v(t,x)\big)^{k_j^q}}{k_j^q!(l_j!)^{k_j^q}}, \tag{1.4}$$

with $\mathbf{k}_j := (k_j^0,\ldots,k_j^n)$, $|\mathbf{k}_j| := k_j^0+\cdots+k_j^n$ *and* $\mathbf{k}_j! := k_j^0!\cdots k_j^n!$, $j=1,\ldots,k$.

We will also need the Duhamel formula, which shows that the solution $v(t,x)$ of an equation of the form

$$\begin{cases} \partial_t v(t,x) + \dfrac{1}{2}\partial_x^2 v(t,x) + g(t,x) = 0, \\ v(T,x) = \phi(x), \quad (t,x)\in[0,T]\times\mathbb{R}, \end{cases}$$

can be represented in integral form as

$$v(t,x) = \int_{-\infty}^{\infty} \varphi(T-t,y-x)\phi(y)dy + \int_t^T\int_{-\infty}^{\infty} \varphi(s-t,y-x)g(s,y)dyds, \tag{1.5}$$

$(t,x)\in[0,T]\times\mathbb{R}$.

2 Codes and mechanism

In this section we start by constructing the set of codes $\mathfrak{C}$ based on the Duhamel formula (1.5), and by iterations of the Duhamel formula we deduce the mechanism $\mathcal{M}$, which will results in the construction of a random coding tree $\mathcal{T}_{t,x,c}$ on which every particle evolves according to a Brownian motion with generator is $\partial_x^2/2$.

In order to derive a probabilistic representation for the solution of (1.2), we will derive an integral formulation for $f\big(u(t,x),\partial_x u(t,x),\ldots,\partial_x^n u(t,x)\big)$ and iterate this process. In the sequel, given $h\in\mathcal{C}^\infty(\mathbb{R}^{n+1})$ we let h^* denote the mapping

$$\begin{aligned} h^* &: \mathcal{C}^{0,\infty}([0,T]\times\mathbb{R}) \longrightarrow \mathcal{C}^{0,\infty}([0,T]\times\mathbb{R}) \\ \psi = \{(t,x)\mapsto\psi(t,x)\} &\longmapsto h^*(\psi) := \big\{(t,x)\mapsto h\big(\psi(t,x),\partial_x\psi(t,x),\ldots,\partial_x^n\psi(t,x)\big)\big\}, \end{aligned} \tag{2.1}$$

where $\mathbb{R}^{[0,T]\times\mathbb{R}}$ represents the set of functions from $[0,T]\times\mathbb{R}$ to $\mathbb{R}$, and for $k\geqslant 1$ we identify ∂_x^k to the operator defined as

$$\begin{aligned} \partial_x^k &: \mathcal{C}^{0,k}([0,T]\times\mathbb{R}) \longrightarrow \mathcal{C}^{0,0}([0,T]\times\mathbb{R}) \\ \psi = \{(t,x)\mapsto\psi(t,x)\} &\longmapsto \partial_x^k(\psi) := \big\{(t,x)\mapsto\partial_x^k\psi(t,x)\big\}. \end{aligned}$$

Letting $v(t,x) := g\big(u(t,x),\partial_x u(t,x),\dots,\partial_x^n u(t,x)\big)$ where $g\in\mathcal{C}^\infty(\mathbb{R}^{n+1})$, by the Faà di Bruno formula (1.4) we have

$$\begin{aligned}
&\partial_t v(t,x)+\frac{1}{2}\partial_x^2 v(t,x)=\sum_{k=0}^n \partial_{z_k} g\big(u(t,x),\dots,\partial_x^n u(t,x)\big)\partial_x^k\left(\partial_t u(t,x)+\frac{1}{2}\partial_x^2 u(t,x)\right)\\
&\quad+\frac{1}{2}\sum_{k=0}^n\sum_{l=0}^n \partial_{z_l}\partial_{z_k} g\big(u(t,x),\dots,\partial_x^n u(t,x)\big)\partial_x^{k+1}u(t,x)\partial_x^{l+1}u(t,x)\\
&=-\sum_{k=0}^n \partial_{z_k} g\big(u(t,x),\dots,\partial_x^n u(t,x)\big)\partial_x^k f\big(u(t,x),\dots,\partial_x^n u(t,x)\big)\\
&\quad+\frac{1}{2}\sum_{k=0}^n\sum_{l=0}^n \partial_{z_l}\partial_{z_k} g\big(u(t,x),\dots,\partial_x^n u(t,x)\big)\partial_x^{k+1}u(t,x)\partial_x^{l+1}u(t,x)\\
&=-\partial_{z_0} g\big(u(t,x),\dots,\partial_x^n u(t,x)\big) f\big(u(t,x),\dots,\partial_x^n u(t,x)\big)\\
&\quad-\sum_{k=1}^n k!\partial_{z_k} g\big(u(t,x),\dots,\partial_x^n u(t,x)\big)\sum_{\substack{1\leqslant|\lambda|\leqslant k\\ 1\leqslant s\leqslant k}}\big(\partial_{z_0}^{\lambda_0}\cdots\partial_{z_n}^{\lambda_n} f\big)^*\sum_{\substack{1\leqslant|\mathbf{k}_1|,\dots,|\mathbf{k}_s|,\ 1\leqslant l_1<\cdots<l_s\\ k_1^i+\cdots+k_s^i=\lambda_i,\ 0\leqslant i\leqslant n\\ |\mathbf{k}_1|l_1+\cdots+|\mathbf{k}_s|l_s=k}}\prod_{\substack{1\leqslant j\leqslant s\\ 0\leqslant q\leqslant n}}\frac{\big(\partial_x^{q+l_j}u(t,x)\big)^{k_j^q}}{k_j^q!(l_j!)^{k_j^q}}\\
&\quad+\frac{1}{2}\sum_{j=0}^n\sum_{l=0}^n \partial_{z_l}\partial_{z_j} g\big(u(t,x),\dots,\partial_x^n u(t,x)\big)\partial_x^{j+1}u(t,x)\partial_x^{l+1}u(t,x),
\end{aligned}\tag{2.2}$$

where $|\lambda|:=\lambda_0+\cdots+\lambda_n$, $\lambda\in\mathbb{N}^{n+1}$. The PDE (2.2) can be rewritten in integral form by the Duhamel formula (1.5) as

$$\begin{aligned}
&g^*(u)(t,x)=g\big(u(t,x),\partial_x u(t,x),\dots,\partial_x^n u(t,x)\big)\\
&=\int_{-\infty}^{\infty}\varphi(T-t,y-x)g\big(\phi(y),\partial_y\phi(y),\dots,\partial_y^n\phi(y)\big)dy\\
&\quad+\int_t^T\int_{-\infty}^{\infty}\varphi(s-t,y-x)\Bigg(\partial_{z_0} g\big(u(s,y),\dots,\partial_y^n u(s,y)\big) f\big(u(s,y),\dots,\partial_y^n u(s,y)\big)\\
&\qquad+\sum_{k=1}^n k!\partial_{z_k} g\big(u(s,y),\dots,\partial_y^n u(s,y)\big)\sum_{\substack{1\leqslant|\lambda|\leqslant k\\ 1\leqslant s\leqslant k}}\big(\partial_{z_0}^{\lambda_0}\cdots\partial_{z_n}^{\lambda_n} f\big)^*\sum_{\substack{1\leqslant|\mathbf{k}_1|,\dots,|\mathbf{k}_s|,\ 1\leqslant l_1<\cdots<l_s\\ k_1^i+\cdots+k_s^i=\lambda_i,\ 0\leqslant i\leqslant n\\ |\mathbf{k}_1|l_1+\cdots+|\mathbf{k}_s|l_s=k}}\prod_{\substack{1\leqslant j\leqslant s\\ 0\leqslant q\leqslant n}}\frac{\big(\partial_y^{q+l_j}u(s,y)\big)^{k_j^q}}{k_j^q!(l_j!)^{k_j^q}}\\
&\qquad-\frac{1}{2}\sum_{j=0}^n\sum_{l=0}^n \partial_{z_l}\partial_{z_j} g\big(u(s,y),\dots,\partial_y^n u(s,y)\big)\partial_y^{j+1}u(s,y)\partial_y^{l+1}u(s,y)\Bigg)dyds.
\end{aligned}\tag{2.3}$$

In order to formalize and extend the above iteration we introduce the following definition, which relies on (2.1).

Definition 2.1 *We let $\mathfrak{C}$ denote the set of operators from $\mathcal{C}^{0,\infty}([0,T]\times\mathbb{R})$ to $\mathcal{C}^{0,\infty}([0,T]\times\mathbb{R})$,*

called codes, and defined as

$$\mathfrak{C} := \left\{ \mathrm{Id}, \quad \left(a\partial_{z_0}^{\lambda_0} \cdots \partial_{z_n}^{\lambda_n} f\right)^*, \ \partial_x^k \ : \ \lambda \in \mathbb{N}^{n+1}, \ a \in \mathbb{R} \setminus \{0\}, \ k \geqslant 1 \right\},$$

where Id *denotes the identity on* $\mathcal{C}^{0,\infty}([0,T] \times \mathbb{R})$.

The role of the parameter $a \in \mathbb{R} \setminus \{0\}$ appearing in the definition of $\mathfrak{C}$ is to account for possible real coefficients appearing in front of partial derivatives in the mapping $\mathcal{M}$, called the *mechanism*, defined on $\mathfrak{C}$ according to (1.4) and (2.3), by matching a *code* $c \in \mathfrak{C}$ to a set $\mathcal{M}(c)$ of code tuples.

Definition 2.2 *The mechanism* $\mathcal{M}$ *is defined on* $\mathfrak{C}$ *by letting* $\mathcal{M}(\mathrm{Id}) := \{f^*\}$, *and*

$$\begin{aligned} \mathcal{M}(g^*) := \left\{(f^*, (\partial_{z_0} g)^*)\right\} & \\ \bigcup \bigcup_{\substack{1 \leqslant \lambda_0 + \cdots + \lambda_n \leqslant k \\ 1 \leqslant |\mathbf{k}_1|, \ldots, |\mathbf{k}_s|, 1 \leqslant l_1 < \cdots < l_s \\ k_1^i + \cdots + k_s^i = \lambda_i, \ 0 \leqslant i \leqslant n \\ |\mathbf{k}_1| l_1 + \cdots + |\mathbf{k}_s| l_s = k, \ 1 \leqslant s \leqslant k \leqslant n}} & \left\{ \left((\partial_{z_k} g)^*, k! \left(\partial_{z_0}^{\lambda_0} \cdots \partial_{z_n}^{\lambda_n} f\right)^*, \left(\frac{\partial_x^{q+l_j}}{k_j^q! (l_j!)^{k_j^q}} \right)_{\substack{r=1,\ldots,k_j^q \\ j=1,\ldots,s \\ q=0,\ldots n}} \right) \right\} \\ \bigcup \bigcup_{j,l=0,\ldots,n} & \left\{ \left(-\frac{1}{2} (\partial_{z_l} \partial_{z_j} g)^*, \partial_x^{j+1}, \partial_x^{l+1} \right) \right\}, \qquad g^* \in \mathfrak{C}, \end{aligned} \tag{2.4}$$

and

$$\mathcal{M}\left(\partial_x^k\right) := \bigcup_{\substack{1 \leqslant \lambda_0 + \cdots + \lambda_n \leqslant k, \ s = 1, \ldots, k \\ 1 \leqslant |\mathbf{k}_1|, \ldots, |\mathbf{k}_s|, \ 1 \leqslant l_1 < \cdots < l_s \\ k_1^i + \cdots + k_s^i = \lambda_i, \ 0 \leqslant i \leqslant n \\ |\mathbf{k}_1| l_1 + \cdots + |\mathbf{k}_s| l_s = k}} \left\{ \left(k! \left(\partial_{z_0}^{\lambda_0} \cdots \partial_{z_n}^{\lambda_n} f\right)^*, \left(\frac{\partial_x^{q+l_j}}{k_j^q! (l_j!)^{k_j^q}} \right)_{\substack{r=1,\ldots,k_j^q \\ j=1,\ldots,s \\ q=0,\ldots n}} \right) \right\}, \quad k \geqslant 1. \tag{2.5}$$

Example - semilinear PDEs

As an example, let $n = 0$ and consider a semilinear PDE of the form

$$\begin{cases} \partial_t u(t,x) + \dfrac{1}{2} \partial_x^2 u(t,x) + f(u(t,x)) = 0 \\ u(T,x) = \phi(x), \qquad (t,x) \in [0,T] \times \mathbb{R}. \end{cases} \tag{2.6}$$

Letting $v(t,x) := g(u(t,x))$, Equation (2.2) reads

$$\begin{aligned} \partial_t v(t,x) + \frac{1}{2} \partial_x^2 v(t,x) &= \left(\partial_t u(t,x) + \frac{1}{2} \partial_x^2 u(t,x) \right) g'(u(t,x)) + \frac{1}{2} (\partial_x u(t,x))^2 g''(u(t,x)) \\ &= -f(u(t,x)) g'(u(t,x)) + \frac{1}{2} (\partial_x u(t,x))^2 g''(u(t,x)). \end{aligned}$$

Therefore, by Duhamel's formula (1.5), $v(t,x)$ satisfies the integral equation

$$v(t,x) = \int_{-\infty}^{\infty} \varphi(T-t, y-x) g(\phi(y)) dy + \int_t^T \int_{-\infty}^{\infty} \varphi(s-t, y-x) \left(f(u(s,y)) g'(u(s,y)) - \frac{1}{2} (\partial_y u(s,y))^2 g''(u(s,y)) \right) dy ds$$

as in (2.3), and the set of $\mathfrak{C}$ codes is given by

$$\mathfrak{C} := \big\{ \mathrm{Id},\ \partial_x,\ af^{(k)},\ a \in \mathbb{R} \setminus \{0\},\ k \in \mathbb{N} \big\}.$$

In this case, the mechanism $\mathcal{M}$ is given by

$$\mathcal{M}(\mathrm{Id}) := \{f^*\},\quad \mathcal{M}(g^*) := \left\{ \big(f^*, (g')^*\big); \left(\partial_x, \partial_x, -\frac{1}{2}(g'')^*\right) \right\},\quad \mathcal{M}(\partial_x) := \big\{ \big((f')^*, \partial_x\big) \big\}, \tag{2.7}$$

for $g \in \mathcal{C}^\infty(\mathbb{R}^{n+1})$ of the form $g = af^{(k)}$, $a \in \mathbb{R} \setminus \{0\}$, $k \geqslant 0$. In this case, every code tuple in $\mathcal{M}(c)$ has at most 3 elements for any $c \in \mathfrak{C}$, and the time complexity of the algorithm can be estimated from the mean depth of the random tree $\mathcal{T}_{0,x,c}$, which grows exponentially as a function of $T > 0$, see e.g. § 4 of [PP22b].

Example - first order gradient nonlinearity

As a second example, let $n = 1$ and consider the nonlinear PDE

$$\begin{cases} \partial_t u(t,x) + \frac{1}{2}\partial_x^2 u(t,x) + f(u(t,x), \partial_x u(t,x)) = 0 \\ u(T,x) = \phi(x). \end{cases}$$

Letting $v(t,x) := g(u(t,x), \partial_x u(t,x))$, Equation (2.2) reads

$$\begin{aligned} \partial_t v(t,x) + \frac{1}{2}\partial_x^2 v(t,x) &= \partial_{z_0} g(u(t,x), \partial_x u(t,x)) \left(\partial_t u(t,x) + \frac{1}{2}\partial_x^2 u(t,x) \right) \\ &\quad + \partial_{z_1} g(u(t,x), \partial_x u(t,x)) \left(\partial_x \partial_t u(t,x) + \frac{1}{2}\partial_x^3 u(t,x) \right) \\ &\quad + \frac{1}{2}\partial_{z_0}^2 g(u(t,x), \partial_x u(t,x)) (\partial_x u(t,x))^2 + \frac{1}{2}\partial_{z_1}^2 g(u(t,x), \partial_x u(t,x)) (\partial_x^2 u(t,x))^2 \\ &\quad + \partial_{z_0}\partial_{z_1} f(u(t,x), \partial_x u(t,x)) \partial_x u(t,x) \partial_x^2 u(t,x) \\ &= -\partial_{z_0} g(u(t,x), \partial_x u(t,x)) f(u(t,x), \partial_x u(t,x)) \\ &\quad - \partial_{z_1} g(u(t,x), \partial_x u(t,x)) \big(\partial_{z_0} f(u(t,x), \partial_x u(t,x)) \partial_x u(t,x) + \partial_{z_1} f(u(t,x), \partial_x u(t,x)) \partial_x^2 u(t,x) \big) \\ &\quad + \frac{1}{2}\partial_{z_0}^2 g(u(t,x), \partial_x u(t,x)) (\partial_x u(t,x))^2 + \frac{1}{2}\partial_{z_1}^2 g(u(t,x), \partial_x u(t,x)) (\partial_x^2 u(t,x))^2 \end{aligned}$$

$$+\partial_{z_0}\partial_{z_1}g(u(t,x),\partial_x u(t,x))\partial_x u(t,x)\partial_x^2 u(t,x),$$

and by Duhamel's formula (1.5), $v(t,x)$ satisfies the integral equation

$$\begin{aligned}
v(t,x) &= \int_{-\infty}^{\infty}\varphi(T-t,y-x)g(u(t,y),\partial_y u(t,y))dy\\
&+\int_t^T\int_{-\infty}^{\infty}\varphi(s-t,y-x)\Big(\partial_{z_0}g(u(t,y),\partial_y u(t,y))f(u(t,y),\partial_y u(t,y))\\
&+\partial_{z_1}g(u(t,y),\partial_y u(t,y))\big(\partial_{z_0}f(u(t,y),\partial_y u(t,y))\partial_y u(t,y)+\partial_{z_1}f(u(t,y),\partial_y u(t,y))\partial_y^2 u(t,y)\big)\\
&-\frac{1}{2}\partial_{z_0}^2 g(u(t,y),\partial_y u(t,y))(\partial_y u(t,y))^2-\frac{1}{2}\partial_{z_1}^2 g(u(t,y),\partial_y u(t,y))\big(\partial_y^2 u(t,y)\big)^2\\
&-\partial_{z_0}\partial_{z_1}g(u(t,y),\partial_y u(t,y))\partial_y u(t,y)\partial_y^2 u(t,y)\Big)dyds
\end{aligned}\tag{2.8}$$

as in (2.3). In this case, the set of codes is given by

$$\mathfrak{C}:=\big\{\mathrm{Id},\ \partial_x^k,\ (a\partial_{z_0}^l\partial_{z_1}^m f)^*\ :\ a\in\mathbb{R}\setminus\{0\},\ k\geqslant 1,\ l,m\geqslant 0\big\},$$

and by (2.8), the mechanism $\mathcal{M}$ satisfies $\mathcal{M}(\mathrm{Id}):=\{f^*\}$,

$$\begin{aligned}
\mathcal{M}(g^*):=&\big\{\big(f^*,(\partial_{z_0}g)^*\big);\big((\partial_{z_1}g)^*,(\partial_{z_0}f)^*,\partial_x\big);\big((\partial_{z_1}g)^*,(\partial_{z_1}f)^*,\partial_x^2\big);\\
&\Big(-\frac{1}{2}(\partial_{z_0}^2 g)^*,\partial_x,\partial_x\Big);\Big(-\frac{1}{2}(\partial_{z_1}^2 g)^*,\partial_x,\partial_x\Big);\Big(-\frac{1}{2}(\partial_{z_0}\partial_{z_1}g)^*,\partial_x,\partial_x^2\Big);\Big(-\frac{1}{2}(\partial_{z_0}\partial_{z_1}g)^*,\partial_x,\partial_x^2\Big)\Big\}
\end{aligned}$$

for $g\in\mathcal{C}^\infty(\mathbb{R}^{n+1})$ of the form $g=a\partial_{z_0}^k\partial_{z_1}^l f$, $a\in\mathbb{R}\setminus\{0\}$, $k,l\geqslant 0$, and

$$\mathcal{M}(\partial_x):=\big\{\big((\partial_{z_0}f)^*,\partial_x\big);\big((\partial_{z_1}f)^*,\partial_x^2\big)\big\}.$$

This makes it possible to find the image of ∂_x^k as well, for example we have

$$\begin{aligned}
\mathcal{M}(\partial_x^2)=\big\{&\big((\partial_{z_0}f)^*,\partial_x^2\big);\big((\partial_{z_0}^2 f)^*,\partial_x,\partial_x\big);\big((\partial_{z_0}\partial_{z_1}f)^*,\partial_x^2,\partial_x\big);\big((\partial_{z_1}f)^*,\partial_x^3\big);\big((\partial_{z_0}\partial_{z_1}f)^*,\partial_x,\partial_x^2\big);\\
&\big((\partial_{z_1}^2 f)^*,\partial_x^3,\partial_x^2\big)\big\}.
\end{aligned}$$

We close this section with the following key lemma which shows that $c(u)$ satisfies a system of equations indexed by $c\in\mathfrak{C}$, and present its application to a semilinear example.

Lemma 2.3 *For any code $c\in\mathfrak{C}$ we have*

$$c(u)(t,x)=\int_{-\infty}^{\infty}\varphi(T-t,y-x)c(u)(T,y)dy+\sum_{Z\in\mathcal{M}(c)}\int_t^T\int_{-\infty}^{\infty}\varphi(s-t,y-x)\prod_{z\in Z}z(u)(s,y)dyds,\tag{2.9}$$

$(t,x)\in[0,T]\times\mathbb{R}$.

Proof. When $c = \mathrm{Id}$ we have

$$
\begin{aligned}
&\mathrm{Id}(u)(t,x) = u(t,x) \\
&= \int_{-\infty}^{\infty} \varphi(T-t, y-x)\phi(y)dy + \int_t^T \int_{-\infty}^{\infty} \varphi(s-t, y-x) f\left(u(s,y), \partial_y u(s,y), \dots, \partial_y^n u(s,y)\right) dyds \\
&= \int_{-\infty}^{\infty} \varphi(T-t, y-x)u(T,y)dy + \int_t^T \int_{-\infty}^{\infty} \varphi(s-t, y-x) f^*(u)(s,y)dyds \\
&= \int_{-\infty}^{\infty} \varphi(T-t, y-x)\mathrm{Id}(u)(T,y)dy + \int_t^T \int_{-\infty}^{\infty} \varphi(s-t, y-x) f^*(u)(s,y)dyds,
\end{aligned}
$$

hence (2.9) holds since $\mathcal{M}(\mathrm{Id}) = \{f^*\}$. When $c \neq \mathrm{Id}$ in written as $c = g^* \in \mathfrak{C}$, the equation (2.3) satisfied by $g^*(y)(t)$ reads

$$
\begin{aligned}
g^*(u)(t,x) &= \int_{-\infty}^{\infty} \varphi(T-t, y-x) g^*(u)(T,y)dy \\
&\quad + \int_t^T \int_{-\infty}^{\infty} \varphi(s-t, y-x) \Bigg((\partial_{z_0} g)^*(u)(s,y) f^*(u)(s,y) \\
&\qquad + \sum_{k=1}^n k! (\partial_{z_k} g)^*(u)(s,y) \sum_{\substack{1 \leqslant |\lambda| \leqslant k \\ 1 \leqslant s \leqslant k}} \left(\partial_{z_0}^{\lambda_0} \cdots \partial_{z_n}^{\lambda_n} f\right)^*(s,y) \sum_{\substack{1 \leqslant |\mathbf{k}_1|, \dots, |\mathbf{k}_s|, \ 1 \leqslant l_1 < \cdots < l_s \\ k_1^i + \cdots + k_s^i = \lambda_i, \ 0 \leqslant i \leqslant n \\ |\mathbf{k}_1| l_1 + \cdots + |\mathbf{k}_s| l_s = k}} \prod_{\substack{1 \leqslant j \leqslant s \\ 0 \leqslant q \leqslant n}} \frac{\left(\partial_x^{q+l_j}(u)(s,y)\right)^{k_j^q}}{k_j^q! l_j!^{k_j^q}} \\
&\qquad - \frac{1}{2} \sum_{j=0}^n \sum_{l=0}^n (\partial_{z_l} \partial_{z_j} g)^*(u)(s,y) \partial_x^{j+1}(u)(s,y) \partial_x^{l+1}(u)(s,y) \Bigg) dyds.
\end{aligned}
$$

Finally, when $c = \partial_x^k$, $k \geqslant 1$, the Faà di Bruno formula shows that

$$
\begin{aligned}
&\partial_x^k(u)(t,x) = \int_{-\infty}^{\infty} \varphi(T-t, y-x) \partial_x^k(u)(T,y)dy \\
&+ k! \int_t^T \int_{-\infty}^{\infty} \varphi(s-t, y-x) \sum_{\substack{1 \leqslant |\lambda| \leqslant k \\ 1 \leqslant s \leqslant k}} \left(\partial_{z_0}^{\lambda_0} \cdots \partial_{z_n}^{\lambda_n} f\right)^*(s,y) \sum_{\substack{1 \leqslant |\mathbf{k}_1|, \dots, |\mathbf{k}_s|, \ 1 \leqslant l_1 < \cdots < l_s \\ k_1^i + \cdots + k_s^i = \lambda_i, \ 0 \leqslant i \leqslant n \\ |\mathbf{k}_1| l_1 + \cdots + |\mathbf{k}_s| l_s = k}} \prod_{\substack{1 \leqslant j \leqslant s \\ 0 \leqslant q \leqslant n}} \frac{\left(\partial_x^{q+l_j}(u)(s,y)\right)^{k_j^q}}{k_j^q! l_j!^{k_j^q}} dyds,
\end{aligned}
$$

and (2.9) follows from the definition (2.4) of $\mathcal{M}$. The exchange between summation over $Z \in \mathcal{M}(c)$ and integrals is justified by Assumption (A)-(iii). □

Example - semilinear PDEs

In the case of a semilinear PDE of the form (2.6) with $n = 0$, by the Duhamel formulation for the PDE satisfied by the functions $u(t,x)$, $\partial_x u(t,x)$, $af^{(k)}(u(t,x))$, the system of equations

(2.9) reads

$$\begin{cases} u(t,x) = \displaystyle\int_{-\infty}^{\infty} \varphi(T-t,y-x)\phi(y)dy + \int_t^T \int_{-\infty}^{\infty} \varphi(s-t,y-x) f(u(s,y))dyds \\ af^{(k)}(u(t,x)) = \displaystyle\int_{-\infty}^{\infty} \varphi(T-t,y-x) af^{(k)}(\phi(y))dy \\ \quad + \displaystyle\int_t^T \int_{-\infty}^{\infty} \varphi(s-t,y-x) \left(af(u(s,y)) f^{(k+1)}(u(s,y)) - \frac{a}{2}(\partial_y u(s,y))^2 f^{(k+2)}(u(s,y)) \right) dyds \\ \partial_x u(t,x) = \displaystyle\int_{-\infty}^{\infty} \varphi(T-t,y-x)\partial_x\phi(y)dy + \int_t^T \int_{-\infty}^{\infty} \varphi(s-t,y-x) f'(u(s,y))\partial_y u(s,y)dyds, \end{cases}$$

$a \in \mathbb{R} \setminus \{0\}$, $k \in \mathbb{N}$, where the last equation can also be obtained by applying Duhamel's formula (1.2) to

$$\partial_t \partial_x u(t,x) + \frac{1}{2}\partial_x^2 \partial_x u(t,x) + f'(u(t,x))\partial_x u(t,x) = 0.$$

3 Random coding trees

This section introduces the random coding trees used for the probabilistic representation of PDE solutions. Let $\rho : \mathbb{R}^+ \to (0,\infty)$ be a probability density function on $\mathbb{R}_+$. For each $c \in \mathfrak{C}$ we let I_c be a random variable taking values uniformly in $\mathcal{M}(c)$ and note $q_c(b) := \mathbb{P}(I_c = b)$ where $b \in \mathcal{M}(c)$. In addition, we consider

- an i.i.d. family $(\tau^{i,j})_{i,j\geqslant 1}$ of random variables with probability density function ρ on $\mathbb{R}_+$,
- for each $c \in \mathfrak{C}$, an independent family $(I_c^{i,j})_{i,j\geqslant 1}$ of i.i.d. discrete random variables on the finite set $\mathcal{M}(c)$, with distribution

$$\mathbb{P}\big(I_c^{i,j} = b\big) = q_c(b) > 0, \qquad b \in \mathcal{M}(c),$$

- an independent family $(W^{i,j})_{i,j\geqslant 1}$ of Brownian motions.

In addition, the sequences $(\tau^{i,j})_{i,j\geqslant 1}$, $(I_c^{i,j})_{c\in\mathfrak{C},i,j\geqslant 1}$ and $(W^{i,j})_{i,j\geqslant 1}$ are assumed to be mutually independent.

We consider a coding branching process starting from a particle $x \in \mathbb{R}$ at time $t \in [0,T]$ with label $\bar{1} = (1)$, which evolves according to the process $X_{s,x}^{\bar{1}} = x + W_{s-t}^{1,1}$, $s \in [t, t+\tau^{1,1}]$

and bears a code $c \in \mathfrak{C}$. If $\tau^{1,1} < T - t$, the process branches at time $t + \tau^{1,1}$ into new independent copies of $(W_t)_{t\in\mathbb{R}_+}$, each of them started at $X_{t+\tau^{1,1}}$ at time $t + \tau^{1,1}$. Based on the value of $\left|I_c^{1,1}\right| \in \mathbb{N}$, a family of $\left|I_c^{1,1}\right|$ new branches are created. If $I_c^{1,1} = (c_1, \ldots, c_l)$ the i-th new branch will bear the code c_i, $i = 1, \ldots, l$.

Every new particle then follows independently another copy of the same branching process as the initial particle, and every branch stops when it reaches the horizon time T. Particles at generation $n \geqslant 1$ are assigned a label of the form $\bar{k} = (1, k_2, \ldots, k_n) \in \mathbb{N}^n$, and their parent is labeled $\bar{k}- := (1, k_2, \ldots, k_{n-1})$. The particle labeled $\bar{k}$ is born at time $T_{\bar{k}-}$ and its lifetime $\tau^{n,\pi_n(\bar{k})}$ is the element of index $\pi_n(\bar{k})$ in the i.i.d. sequence $(\tau^{n,j})_{j\geqslant 1}$, defining an injection

$$\pi_n : \mathbb{N}^n \to \mathbb{N}, \qquad n \geqslant 1.$$

The random evolution of particle $\bar{k}$ is given by

$$X_{s,x}^{\bar{k}} := X_{T_{\bar{k}-},x}^{\bar{k}-} + W_{s-T_{\bar{k}-}}^{n,\pi_n(\bar{k})}, \qquad s \in [T_{\bar{k}-}, T_{\bar{k}}],$$

where $T_{\bar{k}} := T_{\bar{k}-} + \tau^{n,\pi_n(\bar{k})}$.

If $T_{\bar{k}} := T_{\bar{k}-} + \tau^{n,\pi_n(\bar{k})} < T$, we draw a sample $I_c^{n,\pi_n(\bar{k})} = (c_1, \ldots, c_l)$ uniformly in $\mathcal{M}(c)$, and the particle $\bar{k}$ branches into $\left|I_c^{n,\pi_n(\bar{k})}\right|$ offsprings at generation $(n+1)$, which are labeled by $\bar{k} = (1, \ldots, k_n, i)$, $i = 1, \ldots, \left|I_c^{n,\pi_n(\bar{k})}\right|$. The particle with label ending with an integer i will carry the code c_i, and the code of particle $\bar{k}$ is denoted by $c_{\bar{k}} \in \mathfrak{C}$. The labels are only used to distinguish the particles in the branching process.

The set of particles dying before time T is denoted by $\mathcal{K}^\circ$, whereas those dying after T form a set denoted by $\mathcal{K}^\partial$.

Definition 3.1 *When started at time $t \in [0, T]$ from a position $x \in \mathbb{R}$ and a code $c \in \mathfrak{C}$ on its first branch, the above construction yields a marked branching process called a random coding tree, and denoted by $\mathcal{T}_{t,x,c}$.*

We note that the random branching tree $\mathcal{T}_{t,x,c}$ is non-explosive in finite time since the number of branching times is a.s. finite, as the sequence $(\tau^{i,j})_{i,j\geqslant 1}$ is i.i.d.. The random tree $\mathcal{T}_{t,x,\mathrm{Id}}$ will be used for the stochastic representation of the solution $u(t, x)$ of the PDE (1.1), while the trees $\mathcal{T}_{t,x,c}$ will be used for the stochastic representation of $c(u)(t, x)$. The next table summarizes the notation introduced so far.

Object	Notation
Initial time	t
Initial position	x
Tree rooted at (t,x) with initial code c	$\mathcal{T}_{t,x,c}$
Particle (or label) of generation $n \geqslant 1$	$k=(1,k_2,\ldots,k_n)$
First branching time	$T_{\bar{1}}$
Lifespan of a particle	$\tau_{\bar{k}} = T_{\bar{k}} - T_{\bar{k}-}$
Birth time of a particle k	$T_{\bar{k}-}$
Death time of a particle k	$T_{\bar{k}}$
Position at birth	$X^k_{T_{\bar{k}-},x}$
Position at death	$X^k_{T_{\bar{k}},x}$
Code of a particle $\overline{k}$	$c_{\bar{k}}$

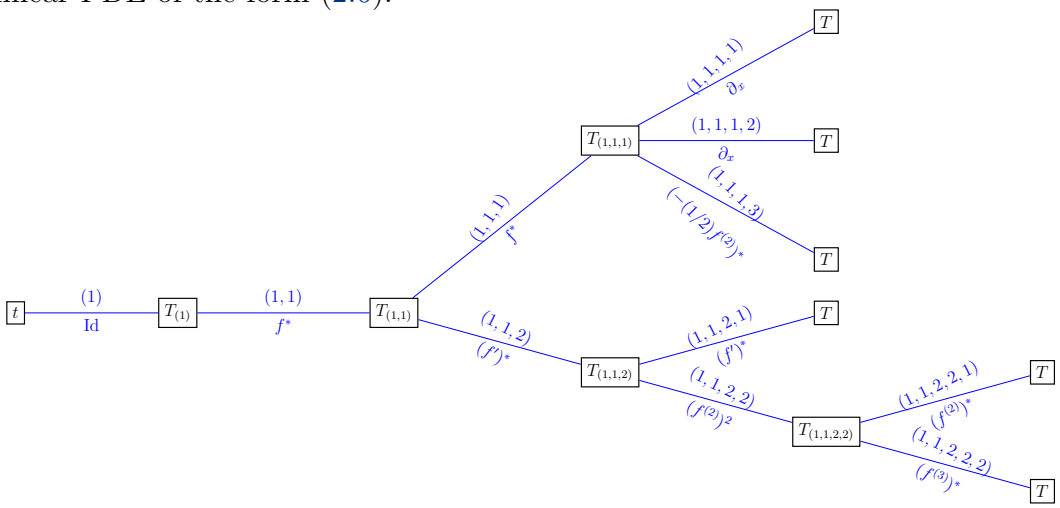
Table 1: Summary of branching tree notation.

Example - semilinear PDEs

In the case of a semilinear PDE of the form (2.6) with $n=0$, the distributions q_c, $c \in \mathfrak{C}$, on the mechanism (2.7) is given by $q_{\mathrm{Id}}(f^*)=1$, $q_{\partial_x}(((f')^*,\partial_x))=1$, and

$$\begin{cases} q_{af^{(k)}}\big((f^*,(af^{(k+1)})^*)\big) = \mathbb{P}\big(I_{af^{(k)}} = (f^*,(af^{(k+1)})^*)\big) = \dfrac{1}{2}, \\ q_{af^{(k)}}\left(\left(\partial_x,\partial_x,-\dfrac{1}{2}(af^{(k+2)})^*\right)\right) = \mathbb{P}\left(I_{af^{(k)}} = \left(\partial_x,\partial_x,-\dfrac{1}{2}(af^{(k+2)})^*\right)\right) = \dfrac{1}{2}, \quad k \geqslant 0. \end{cases}$$

The next illustration represents a sample of the random tree $\mathcal{T}_{t,x,\mathrm{Id}}$ started from $c=\mathrm{Id}$ for a semilinear PDE of the form (2.6).



4 Probabilistic representation of PDE solutions

In this section, we derive a probabilistic representation formula for the solution of fully nonlinear PDEs of the form (1.1), using a multiplicative functional $\mathcal{H}(\mathcal{T}_{t,x,c})$ of the random tree $\mathcal{T}_{t,x,c}$. For this, we will link the codes introduced in the previous section to the Duhamel formulation of the PDE (1.1) by deriving a system of equations satisfied by $\mathbb{E}[\mathcal{H}(\mathcal{T}_{t,x,c})]$, $c \in \mathfrak{C}$. We let

$$\overline{F}(t) := \int_t^\infty \rho(u)du = \mathbb{P}(\tau > t), \qquad t \in \mathbb{R}_+,$$

denote the tail distribution function of $\tau^{i,j}$, $i, j \geqslant 1$.

Definition 4.1 *We define the functional $\mathcal{H}(\mathcal{T}_{t,x,c})$ of the random coding tree $\mathcal{T}_{t,x,c}$ started at time $t \in [0,T]$, location $x \in \mathbb{R}$ and code $c \in \mathfrak{C}$ as*

$$\mathcal{H}(\mathcal{T}_{t,x,c}) := \prod_{\bar{k} \in \mathcal{K}^\circ} \frac{1}{q_{c_{\bar{k}}}(I_{c_{\bar{k}}})\rho(\tau_{\bar{k}})} \prod_{\bar{k} \in \mathcal{K}^\partial} \frac{c_{\bar{k}}(u)\big(T, X^{\bar{k}}_{T,x}\big)}{\overline{F}(T - T_{\bar{k}-})}.$$

Note that for $c \in \mathfrak{C}$ of the form $c = \partial_x^k$ we have

$$c(u)(T,x) = \partial_x^k \phi(x),$$

and for $c \in \mathfrak{C}$ of the form $c = g^*$ with $g = a\partial_{z_0}^{\lambda_0} \cdots \partial_{z_n}^{\lambda_n} f$, we have

$$c(u)(T,x) = g\big(\phi(T,x), \partial_x\phi(T,x), \ldots, \partial_x^n \phi(T,x)\big), \quad (\lambda_0, \ldots, \lambda_n) \in \mathbb{N}^{n+1}.$$

The next result gives the probabilistic representation of solutions of (1.1) as an expectation over random coding trees.

Theorem 4.2 *Under Assumption (A), let $T > 0$ such that*

$$\mathbb{E}\big[\big|\mathcal{H}(\mathcal{T}_{t,x,c})\big|\big] < \infty, \quad (t,x) \in [0,T] \times \mathbb{R}, \ c \in \mathfrak{C}.$$

Then, for any $c \in \mathfrak{C}$ the function

$$u_c(t,x) := \mathbb{E}\big[\mathcal{H}(\mathcal{T}_{t,x,c})\big], \qquad (t,x) \in [0,T] \times \mathbb{R},$$

is a solution of the system

$$u_c(t,x) = \int_{-\infty}^\infty \varphi(T-t, y-x)c(u)(T,y)dy + \sum_{Z \in \mathcal{M}(c)} \int_t^T \int_{-\infty}^\infty \varphi(s-t, y-x) \prod_{z \in Z} u_z(s,y)dyds, \tag{4.1}$$

with $u_c(T,x) = c(u)(T,x)$, $(t,x) \in [0,T] \times \mathbb{R}$. *Moreover, if the solution* $(u_c)_{c\in\mathfrak{C}}$ *of* (4.1) *is unique, then we have*

$$c(u)(t,x) = u_c(t,x) = \mathbb{E}\big[\mathcal{H}(\mathcal{T}_{t,x,c})\big], \qquad (t,x) \in [0,T] \times \mathbb{R}.$$

In particular, taking $c = \mathrm{Id}$*, we have the probabilistic representation*

$$u(t,x) = \mathbb{E}\big[\mathcal{H}(\mathcal{T}_{t,x,\mathrm{Id}})\big], \qquad (t,x) \in [0,T] \times \mathbb{R}. \tag{4.2}$$

Proof. For $c \in \mathfrak{C}$, we let

$$u_c(t,x) := \mathbb{E}\big[\mathcal{H}(\mathcal{T}_{t,x,c})\big], \qquad (t,x) \in [0,T] \times \mathbb{R}.$$

By conditioning on the first branching time $T_{\bar{1}}$, the first particle bearing the code Id branches at time $T_{\bar{1}}$ into a new particle bearing the code f^* as $\mathcal{M}(\mathrm{Id}) = \{f^*\}$, hence

$$\begin{aligned}
u_{\mathrm{Id}}(t,x) &= \mathbb{E}\big[\mathcal{H}(\mathcal{T}_{t,x,\mathrm{Id}})\mathbb{1}_{\{T_{\bar{1}}>T\}} + \mathcal{H}(\mathcal{T}_{t,x,\mathrm{Id}})\mathbb{1}_{\{T_{\bar{1}}\leqslant T\}}\big] \\
&= \mathbb{E}\left[\frac{\phi\big(X^{\bar{1}}_{T,x}\big)}{\overline{F}(T-t)}\mathbb{1}_{\{T_{\bar{1}}>T\}}\right] + \mathbb{E}\left[\frac{u_{f^*}\big(T_{\bar{1}}, X^{\bar{1}}_{T_{\bar{1}},x}\big)}{\rho(T_{\bar{1}}-t)}\mathbb{1}_{\{T_{\bar{1}}\leqslant T\}}\right] \\
&= \frac{\mathbb{P}(T_{\bar{1}}>T)}{\overline{F}(T-t)}\int_{-\infty}^{\infty}\varphi(T-t,y-x)\phi(y)dy + \int_t^T\int_{-\infty}^{\infty}\varphi(s-t,y-x)u_{f^*}(s,y)dyds \\
&= \int_{-\infty}^{\infty}\varphi(T-t,y-x)\phi(y)dy + \int_t^T\int_{-\infty}^{\infty}\varphi(s-t,y-x)u_{f^*}(s,y)dyds, \quad (t,x) \in [0,T]\times\mathbb{R}.
\end{aligned}$$

Similarly, starting from any code $c \in \mathfrak{C}$ different from Id, we draw a sample of I_c uniformly in $\mathcal{M}(c)$. As each code in the tuple I_c yields a new branch at time $T_{\bar{1}}$, we obtain

$$\begin{aligned}
u_c(t,x) &= \mathbb{E}\big[\mathcal{H}(\mathcal{T}_{t,x,c})\mathbb{1}_{\{T_{\bar{1}}>T\}} + \mathcal{H}(\mathcal{T}_{t,x,c})\mathbb{1}_{\{T_{\bar{1}}\leqslant T\}}\big] \\
&= \mathbb{E}\left[\frac{c(u)\big(T, X^{\bar{1}}_{T,x}\big)}{\overline{F}(T-t)}\mathbb{1}_{\{T_{\bar{1}}>T\}} + \mathbb{1}_{\{T_{\bar{1}}\leqslant T\}}\sum_{Z\in\mathcal{M}(c)}\mathbb{1}_{\{I_c=Z\}}\frac{\prod_{z\in Z}u_z\big(T_{\bar{1}}, X^{\bar{1}}_{T_{\bar{1}},x}\big)}{q_c(Z)\rho(T_{\bar{1}}-t)}\right] \\
&= \int_{-\infty}^{\infty}\varphi(T-t,y-x)c(u)(T,y)dy + \sum_{Z\in\mathcal{M}(c)}\int_t^T\int_{-\infty}^{\infty}\varphi(s-t,y-x)\prod_{z\in Z}u_z(s,y)dyds,
\end{aligned}$$

which yields the system of equations (4.1). We conclude by noting that from Lemma 2.3, the family of functions $(c(u))_{c\in\mathfrak{C}}$ is the solution of the system (4.1), hence we have $(c(u))_{c\in\mathfrak{C}} = (u_c)_{c\in\mathfrak{C}}$, and

$$\mathbb{E}\big[\mathcal{H}(\mathcal{T}_{t,x,c})\big] = u_c(t,x) = c(u)(t,x), \qquad (t,x) \in [0,T]\times\mathbb{R}, \quad c \in \mathfrak{C}.$$

□

In the case of a semilinear PDE of the form (2.6) with $n = 0$, the system (4.1) reads

$$\begin{cases} \partial_t u_{\mathrm{Id}}(t,x) + \dfrac{1}{2}\partial_x^2 u_{\mathrm{Id}}(t,x) + u_{f^*}(t,x) = 0 \\ \partial_t u_{(f^{(k)})^*}(t,x) + \dfrac{1}{2}\partial_x^2 u_{(f^{(k+2)})^*}(t,x) + u_{(f^{(k)})^*}(t,x) u_{(f^{(k+1)})^*}(t,x) - \dfrac{1}{2}(u_{\partial_x}(t,x))^2 u_{(f^{(k+2)})^*}(t,x) = 0, \\ \partial_t u_{\partial_x}(t,x) + \dfrac{1}{2}\partial_x^2 u_{\partial_x}(t,x) + u_{\partial_x}(t,x) u_{(f')^*}(t,x) = 0 \\ u_c(T,x) = c(u)(T,x), \quad c \in \mathfrak{C}, \end{cases} \tag{4.3}$$

$k \geqslant 0$. For example, in case $n = 0$ and $f(z_0) = \mathrm{e}^{z_0}$, the system (4.3) simplifies to

$$\begin{cases} \partial_t u_{\mathrm{Id}}(t,x) + \dfrac{1}{2}\partial_x^2 u_{\mathrm{Id}}(t,x) + u_{f^*}(t,x) = 0 \\ \partial_t u_{f^*}(t,x) + \dfrac{1}{2}\partial_x^2 u_{f^*}(t,x) + (u_{f^*}(t,x))^2 - \dfrac{1}{2}(u_{\partial_x}(t,x))^2 u_{f^*}(t,x) = 0, \\ \partial_t u_{\partial_x}(t,x) + \dfrac{1}{2}\partial_x^2 u_{\partial_x}(t,x) + u_{\partial_x}(t,x) u_{f^*}(t,x) = 0 \\ u_c(T,x) = c(u)(T,x), \quad c \in \mathfrak{C}. \end{cases}$$

The next result provides sufficient conditions for the uniform boundedness of the random functional $\mathcal{H}(\mathcal{T}_{t,x,c})$, therefore ensuring the integrability needed for the validity of the probabilistic representation (4.2) in Theorem 4.2.

Proposition 4.3 *Under Assumption (A), suppose in addition that*

$$\|\partial_x^k \phi\|_{L^\infty(\mathbb{R})} \leqslant K, \ \left\|\partial_{z_0}^{\lambda_0} \cdots \partial_{z_n}^{\lambda_n} f\big(\phi, \partial_x\phi, \ldots, \partial_x^n \phi\big)\right\|_{L^\infty(\mathbb{R})} \leqslant K, \quad k \geqslant 0, \ \lambda_0, \ldots, \lambda_n \geqslant 0,$$

for some $K \in (0,1)$, and that the probability density function $\rho(t)$ is decreasing and satisfies the conditions

$$\rho(T) \geqslant \frac{1}{\min_{c\in\mathfrak{C}} q_c(I_c)} \quad \text{and} \quad K \leqslant \overline{F}(T)$$

for some $T > 0$. Then we have $|\mathcal{H}(\mathcal{T}_{t,x,c})| \leqslant 1$, a.s., $(t,x,c) \in [0,T] \times \mathbb{R} \times \mathfrak{C}$.

Proof. Since $\partial_x^k(u)(T,x)$ and $g^*(u)(T,x)$ respectively denote the functions $\partial_x^k\phi(x)$ and $g\big(\phi(T,x), \partial_x\phi(T,x), \ldots, \partial_x^n\phi(T,x)\big)$, we have the bound

$$\begin{aligned} |\mathcal{H}(\mathcal{T}_{t,x,c})| &\leqslant \prod_{\overline{k}\in\mathcal{K}^\circ} \frac{1}{q_{c_{\overline{k}}}(I_{c_{\overline{k}}})\rho(\tau_{\overline{k}})} \prod_{\overline{k}\in\mathcal{K}^\partial} \frac{K}{\overline{F}(T - T_{\overline{k}-})} \\ &\leqslant \prod_{\overline{k}\in\mathcal{K}^\circ} \frac{1}{\rho(T)\min_{c\in\mathfrak{C}} q_c(I_c)} \prod_{\overline{k}\in\mathcal{K}^\partial} \frac{K}{\overline{F}(T)} \\ &\leqslant 1, \quad a.s.. \end{aligned}$$

□

5 Numerical examples

In this section we provide numerical confirmations of the validity of our algorithm on examples of nonlinear and fully nonlinear PDEs, by benchmarking its output to closed-form solutions. Numerical computations are done in Mathematica using the codes provided in appendix, which apply to the general fully nonlinear case with higher order derivatives, with ρ the standard exponential probability density. Semilinear examples are treated using a simplified code that does not use gradient nonlinearities.

For the Allen-Cahn and Hamilton-Jacobi-Bellman equations, the performance of our coding tree algorithm compares favorably to the deep BSDE method of [HJE17, HJE18] and to the branching diffusion method of [HLTT14] in terms of stability and explosion time. Similar conclusions regarding the multilevel Picard method [EHJK19] are derived in the subsequent examples which use of functional nonlinearities that cannot be treated by the branching diffusion method. Finally, we consider fully nonlinear examples involving higher order gradient terms, to which the above mentioned methods do not apply.

Semilinear examples

Example 1-a). Consider the Allen-Cahn (or Ginzburg-Landau) equation

$$\partial_t u(t,x) + \Delta_x u(t,x) + u(t,x) - u^3(t,x) = 0, \quad (t,x) \in [0,T] \times \mathbb{R}^d, \tag{5.1}$$

as in [HJE18] or § 4.2 of [HJE17], with terminal condition $\phi(x) = 1/(2 + 2\|x\|^2/5)$, where $\|x\| := \sqrt{x_1^2 + \cdots + x_d^2}$, $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$. In Table 2 we compare our results to the ones obtained in Table 1 of [HJE17] for the estimation of $u(0,x)$ at $x = (0, \ldots, 0)$ for $T = 0.3$ with 4000 iterations in dimension $d = 100$. We note that such results can be recovered by the multilevel Picard method, see also § 3 of [BBH+20].

	[HJE17]	Coding trees
Mean	0.0528	0.052754
Standard deviation	0.0002	0.000364
Mean of rel. L^1 error	0.0030	0.005916
SD of rel. L^1 error	0.0022	0.003661

Table 2: BSDE [HJE17] *vs* coding tree method for (5.1) with $T = 0.3$ and $d = 100$.

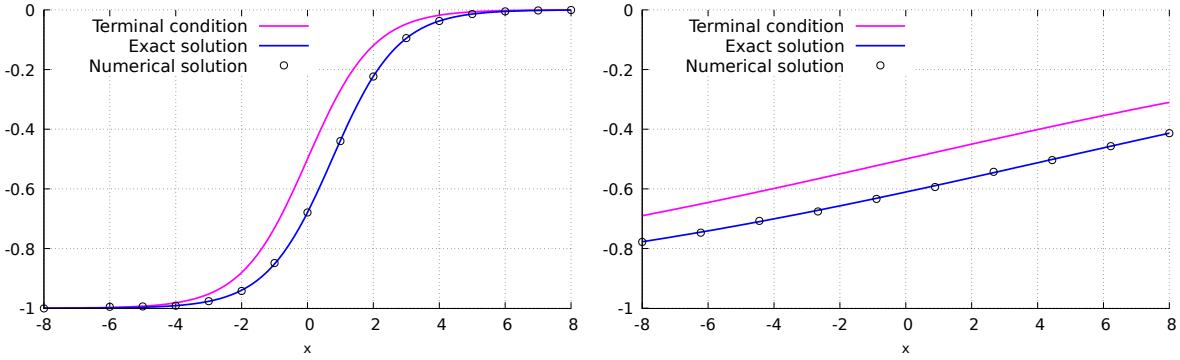
Next, we consider the Allen-Cahn equation

$$\partial_t u(t,x) + \frac{1}{2}\Delta_x u(t,x) + u(t,x) - u^3(t,x) = 0, \quad (t,x) \in [0,T] \times \mathbb{R}^d, \tag{5.2}$$

which admits the traveling wave solution

$$u(t,x) = -\frac{1}{2} - \frac{1}{2}\tanh\left(\frac{3}{4}(T-t) - \sum_{i=1}^{d}\frac{x_i}{2\sqrt{d}}\right), \quad (t,x) \in [0,T]\times\mathbb{R}^d. \tag{5.3}$$

In Figure 1 we compare the closed-form solution (5.3) of (5.2) to its estimation by the coding tree method.



(a) Dimension $d = 1$ with $T = 0.5$.

(b) Dimension $d = 100$ with $T = 0.3$.

Figure 1: Numerical solution $u(0,x)$ of (5.2) with $100,000$ Monte Carlo samples.

In Table 3 we compare the stability of the coding tree method to that of the branching diffusion of [HLTT14] according to Table 8 in [EHJK19] in dimension $d = 1$ with $T = 1$ and the explicit solution

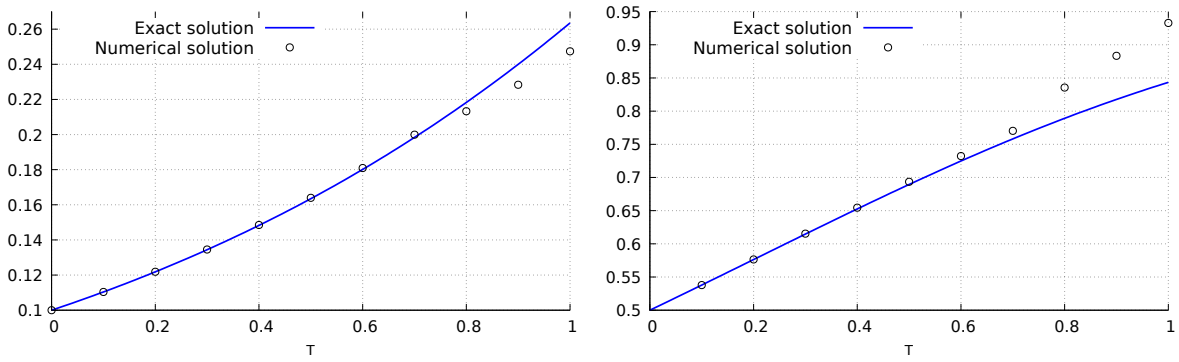
$$u(t,x) = \frac{1}{\sqrt{1-(1-(\phi(0))^{-2})\mathrm{e}^{-2(T-t)}}}, \qquad t \in [0,T]. \tag{5.4}$$

It turns out that our coding tree algorithm remains stable for higher values of $\phi(0)$ in this example.

$\phi(0)$	Exact value $u(0,0)$	Coding trees	[HLTT14]
0.1	0.263540	0.247403	0.271007
0.2	0.485183	0.472720	0.499103
0.3	0.649791	0.723543	0.848879
0.4	0.764605	0.866281	3.495457
0.5	0.843347	0.932852	21.68436
0.6	0.897811	0.968213	136.6667
0.7	0.936233	1.005440	7321.326
0.8	0.963981	0.950816	
0.9	0.984496	0.944715	
1.0	1.0	1.000164	
1.1	1.011955	1.182766	
1.2	1.021340	1.576551	
1.5	1.039856	5.182978	
2.0	1.054973	30.006351	

Table 3: Branching diffusion [HLTT14] *vs* coding trees for the Allen-Cahn equation (5.2).

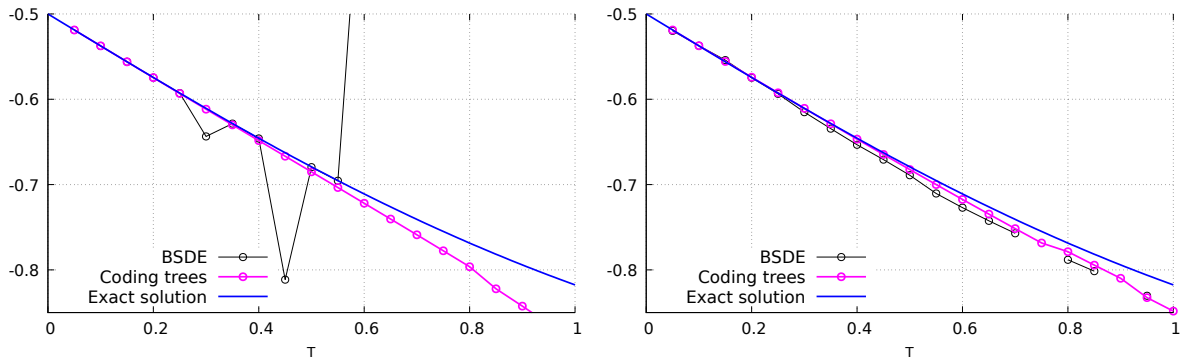
The stability over time of the coding tree algorithm applied to the Allen-Cahn equation (5.2) with solution (5.4) is illustrated in Figure 2.



(a) $\phi(0) = 0.1$. (b) $\phi(0) = 0.5$.

Figure 2: Numerical solution $u(0,0)$ of (5.2) by the coding tree method.

In Figure 3, the stability over time of the coding tree algorithm is compared to that of the BSDE method [HJE17] for the Allen-Cahn equation (5.2) with solution (5.3), using the BSDE solver available at `https://github.com/frankhan91/DeepBSDE`.



(a) Dimension $d = 1$.

(b) Dimension $d = 100$.

Figure 3: BSDE method [HJE17] *vs* coding trees for the Allen-Cahn equation (5.2).

The data of Figure 3 with $d = 1$ and $d = 100$ is presented in Table 4, where it turns out that our coding tree algorithm remains stable for higher values of T and that the BSDE method is less stable in low dimension in this example. We also note that this BSDE solver requires the input of an initial guess interval y_init_range for the algorithm to run, which is not the case in branching type methods.

		$d = 1$		$d = 100$	
T	Exact value	Coding trees	BSDE	Coding trees	BSDE
0.10	-0.537430	-0.537451	-0.537470	-0.537318	-0.537169
0.20	-0.574443	-0.574662	-0.574581	-0.574111	-0.574704
0.30	-0.610639	-0.611628	-0.643579	-0.610616	-0.615149
0.40	-0.645656	-0.648401	-0.645983	-0.646737	-0.653324
0.50	-0.679179	-0.685096	-0.679532	-0.682355	-0.688992
0.60	-0.710950	-0.721866	-0.270275	-0.717349	-0.727000
0.70	-0.740775	-0.758880	NaN	-0.751602	-0.757082
0.80	-0.768525	-0.796317	NaN	-0.778700	-0.788100
0.90	-0.794130	-0.842476	NaN	-0.809848	NaN
1.00	-0.817574	-0.884167	NaN	-0.848294	NaN
1.20	-0.858149	-0.945912	NaN	-0.900085	-0.530944
1.40	-0.890903	-1.023799	NaN	-0.965941	-0.473223
1.60	-0.916827	-1.104303	NaN	-1.021344	NaN
1.80	-0.937027	-1.155304	NaN	-1.089572	NaN
2.00	-0.952574	-1.193363	NaN	-1.127749	NaN

Table 4: BSDE method [HJE17] *vs* coding trees for the Allen-Cahn equation (5.2).

Example 1-*b*). As the Allen-Cahn Example 1-*a*) only involves polynomial nonlinearities, it can be treated by the branching diffusion method, see [HL12, HLOT+19]. On the other hand, the following example, which makes use of a functional nonlinearity, cannot be

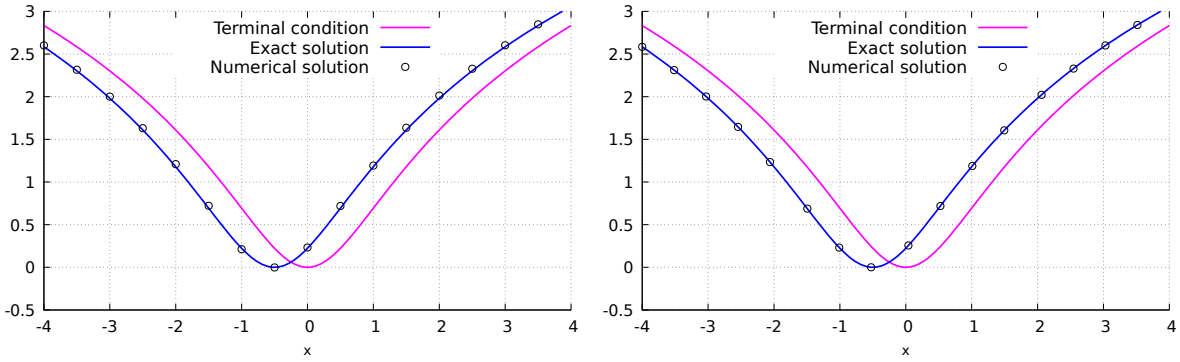
treated by such a method. Consider the equation

$$\partial_t u(t,x) + \frac{\alpha}{d}\sum_{i=1}^{d} \partial_{x_i} u(t,x) + \frac{1}{2}\Delta_x u(t,x) + \mathrm{e}^{-u(t,x)}(1 - 2\mathrm{e}^{-u(t,x)})d = 0, \tag{5.5}$$

which admits the traveling wave solution

$$u(t,x) = \log\left(1 + \left(\alpha(T-t) + \sum_{i=1}^{d} x_i\right)^2\right), \qquad (t,x) \in [0,T] \times \mathbb{R}^d. \tag{5.6}$$

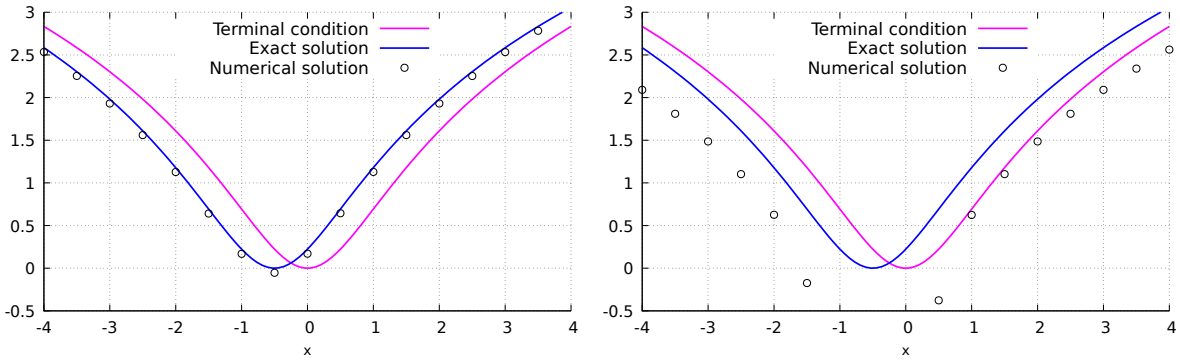
In Figure 4 we take $T = 0.05$, $\alpha = 10$ and $100,000$ Monte Carlo samples.



(a) Dimension $d = 1$. (b) Dimension $d = 10$.

Figure 4: Numerical solution $u(0,x)$ of (5.5).

In Figure 5 we take $T = 0.05$, $\alpha = 10$ and use the multilevel Picard code available at `https://github.com/seb-becker/mlp` with 8 iterations. We note that the performance of the multilevel Picard method is dimension-dependent in this example.



(a) Dimension $d = 1$. (b) Dimension $d = 10$.

Figure 5: Numerical solution $u(0,x)$ of (5.5) by the multilevel Picard method.

Quasilinear examples

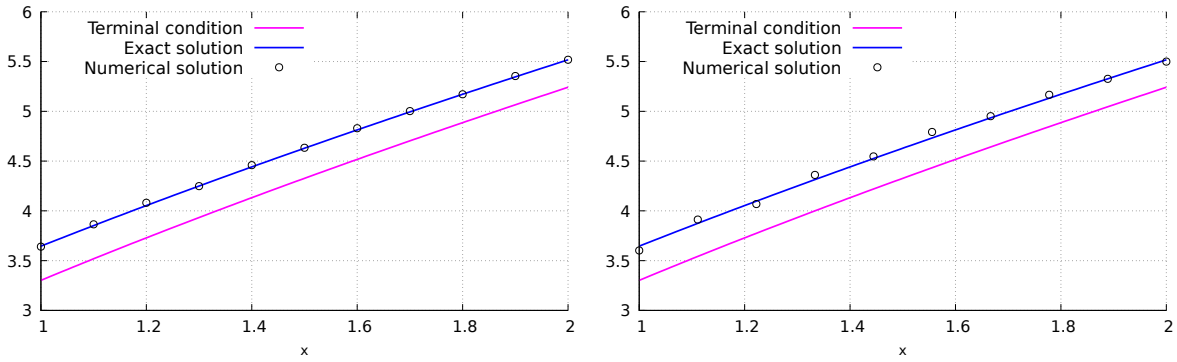
Example 2-*a*). Consider the Dym equation

$$\partial_t u(t,x) + \frac{1}{d} u^3(t,x) \sum_{i=1}^{d} \partial_{x_i}^3 u(t,x) = 0, \tag{5.7}$$

which admits the traveling wave solution

$$u(t,x) = \left(3\alpha \left(4\alpha^2 (T-t) + \sum_{i=1}^{d} x_i \right) \right)^{2/3}, \qquad (t,x) \in [0,T] \times \mathbb{R}^d.$$

with $\alpha > 0$. In Figure 6 we take $T = 0.01$, $\alpha = 2$ and $100,000$ Monte Carlo samples.



(a) Dimension $d = 1$. (b) Dimension $d = 5$.

Figure 6: Numerical solution $u(0,x)$ of (5.7).

As this example and the next one involve derivatives of order greater than one, they may not be treated by the approach of [HLOT+19] due Malliavin weight integrability issues, see page 199 therein.

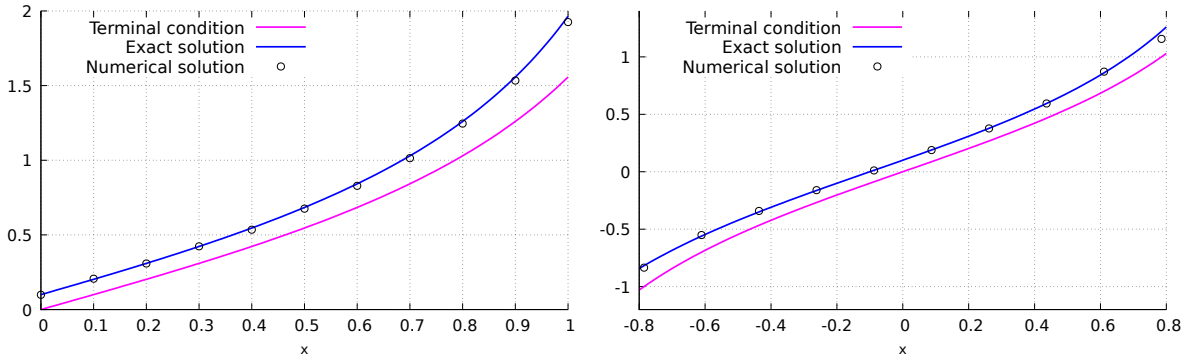
Example 2-*b*). For a quasilinear example using non-polynomial nonlinearities, consider the equation

$$\partial_t u(t,x) + \frac{\alpha}{d} \sum_{i=1}^{d} \partial_{x_i} u(t,x) + \frac{\Delta_x u(t,x)}{1 + u^2(t,x)} - 2u(t,x) = 0, \tag{5.8}$$

which admits the traveling wave solution

$$u(t,x) = \tan\left(\alpha(T-t) + \sum_{i=1}^{d} x_i \right), \qquad (t,x) \in [0,T] \times \mathbb{R}^d,$$

for $\alpha \in \mathbb{R}$. In Figure 7 we take $T = 0.01$, $\alpha = 10$ and one million Monte Carlo samples.



(a) Dimension $d = 1$.

(b) Dimension $d = 5$.

Figure 7: Numerical solution $u(0, x)$ of (5.8).

Quadratic gradient nonlinearity

Consider the Hamilton-Jacobi-Bellman (HJB) equation

$$\partial_t u(t,x) + \Delta_x u(t,x) = \sum_{i=1}^{d} (\partial_{x_i} u(t,x))^2, \tag{5.9}$$

as in [HJE18] or § 4.3 of [HJE17], with terminal condition $\phi(x) = \log((1 + \|x\|^2)/2)$, $x \in \mathbb{R}^d$. In Table 5 we compare our results to the ones obtained for the estimation of $u(0, x)$ at $x = (0, \ldots, 0)$ for $T = 1$ with 2000 iterations in Table 2 of [HJE17], in dimension $d = 100$.

	[HJE17]	Coding trees
Mean	4.5977	4.580340
Standard deviation	0.0019	0.001869
Mean of rel. L^1 error	0.0017	0.002126
SD of rel. L^1 error	0.0004	0.000407

Table 5: Comparison results with the BSDE method [HJE17] for the HJB equation (5.9).

Fully nonlinear examples

In this section, we consider fully nonlinear examples involving higher order gradient terms.

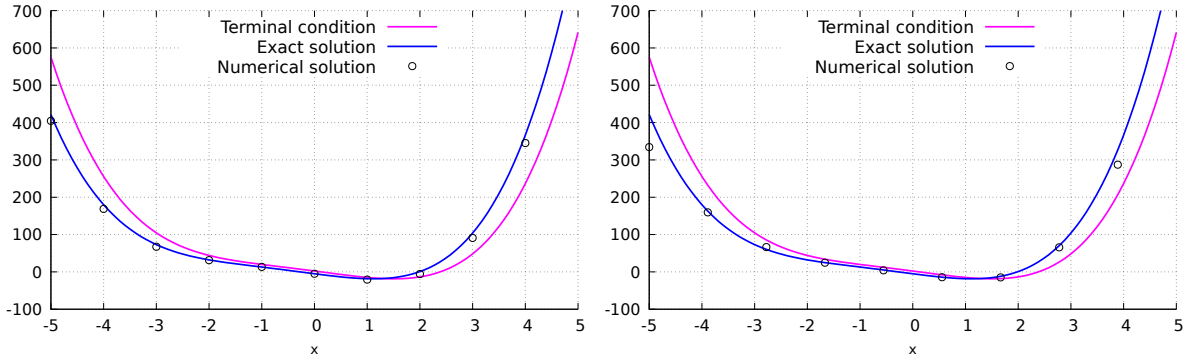
Example 3-*a*). For a fully nonlinear example involving a fourth derivative, consider the equation

$$\partial_t u(t,x) + \frac{\alpha}{d} \sum_{i=1}^{d} \partial_{x_i} u(t,x) + u(t,x) - \left(\frac{\Delta_x u(t,x)}{12d} \right)^2 + \frac{1}{d} \sum_{i=1}^{d} \cos\left(\frac{\pi \partial_{x_i}^4 u(t,x)}{4!} \right) = 0, \tag{5.10}$$

with terminal condition $\phi(x) := x^4 + x^3 + bx^2 + cx + d$ where $b = -36/47$, $c = 24b$, $d = 4b^2$, $\alpha = 10$, and solution

$$u(t,x) = \varphi\left(\alpha(T-t) + \sum_{i=1}^{d} x_i\right), \qquad (t,x) \in [0,T] \times \mathbb{R}^d.$$

In Figure 8 we take $T = 0.04$ and $100,000$ Monte Carlo samples. In dimension $d = 1$, the graph below is obtained by letting f[y_] :=-y[[3]]/2 + α y[[2]] + y[[1]] - y[[3]]^2/144 + Cos[Pi*y[[5]]/24]; phi[x_] := x^4 + x^3 + bx^2 + cx + d, and by running Sol[f, 0, T, x, phi, 100000, 4] in Mathematica for $x \in [-5, 5]$.



(a) Dimension $d = 1$. (b) Dimension $d = 5$.

Figure 8: Numerical solution $u(0,x)$ of (5.10).

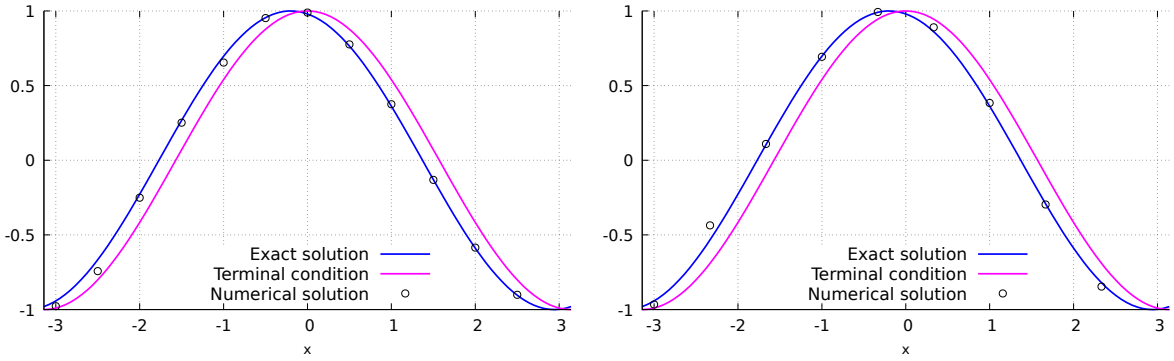
Example 3-*b*). For another fully nonlinear example, consider the equation

$$\partial_t u(t,x) + \frac{\alpha}{d}\sum_{i=1}^{d} \partial_{x_i} u(t,x) + \log\left(\frac{1}{d}\sum_{i=1}^{d}\left(\partial^2_{x_i} u(t,x)\right)^2 + \left(\partial^3_{x_i} u(t,x)\right)^2\right) = 0, \qquad (5.11)$$

with terminal condition $\phi(x) = \cos\left(\sum_{i=1}^{d} x_i\right)$, $x \in \mathbb{R}^d$, and solution

$$u(t,x) = \cos\left(\alpha(T-t) + \sum_{i=1}^{d} x_i\right), \qquad (t,x) \in [0,T] \times \mathbb{R}^d,$$

where $\alpha = 5$. In Figure 9 we take $T = 0.02$ and $100,000$ Monte Carlo samples. In dimension $d = 1$, the graph below is obtained by letting f[y_] := α*y[[2]] - y[[3]]/2 + Log[y[[3]]^2 + y[[4]]^2]; phi[x_] := Cos[x], and by running Sol[f, 0, T, x, phi, 100000, 3] in Mathematica for $x \in [-\pi, \pi]$.



(a) Dimension $d = 1$.

(b) Dimension $d = 5$.

Figure 9: Numerical solution $u(0, x)$ of (5.11).

A Computer codes

The following codes implement the algorithm of Theorem 4.2 in Mathematica using an exponential distribution $\rho(t) = \mathrm{e}^{-t}$, $t \geqslant 0$. In the above examples the values of fdb[n,k] have been precomputed by *memoization* for k up to 7 in order to speed up the solution algorithm, where n denotes the highest order of derivative ∂_x^n in (1.1). The next code implements the mechanism $c \mapsto \mathcal{M}(c)$ in the procedure "codetofunction" via the combinatorics of the Faà di Bruno formula written the function "fdb".

```
Needs["Combinatorica`"]
kuple[lambda__, s_, l__, k_] := (Module[{m, D, E}, m = Length[lambda]; If[m == 1,
     Return[Compositions[lambda[[1]], s]]]; E = {}; Do[Do[If[m == 2, D = Append[{k1}, k2], D = Append[k1,
     k2]]; E = Append[E, D], {k1, kuple[Drop[lambda, -1], s, l, k]}], {k2, Compositions[lambda[[m]],
     s]}]; Return[E];])
fdb[n_, k_] := fdb[n, k] = (L = {}; Do[Do[Do[ Do[Do[If[n == 1, kv = {ku}, kv = ku];
       kw = Sum[kv[[q]], {q, 1, n}]; If[ Signature[l] != 0 && ! MemberQ[kw, 0] && Total[ kw*l] == k , L =
            Append[L, {k!/Product[l[[p]]!^kw[[p]]*Product[kv[[q]][[p]]!, {q, 1, n}], {p, 1, s}], lambda,
            kv, l, s}]], {ku, kuple[lambda, s, l, k]}], {l, Select[Tuples[Range[k], s], OrderedQ ]}], {s,
            1, k}], {lambda, Compositions[r, n]}], {r, 1, k}]; Return[L])
codetofunction[f_, c__, y0_, phi_] := (Module[{x, y, s1}, If [c == {Id}, Return [phi[y0]]]; If [Length[c]
     == 1 && c[[1]] < 0, Return [D[phi[y], {y, -c[[1]]}] /. {y -> y0}]]; y = Array[s1, Length[c]]; z =
     D[f[y], Sequence @@ Transpose[{y, c}]]; Do[z = z /. {s1[k + 1] -> D[phi[x], {x, k}] /. {x -> y0}},
     {k, 0, Length[y] - 1}]; Return[z]])
```

Faà di Bruno combinatorics and mechanism.

Numerical solution estimates are then computed using the following program, in which the code ∂_x^k is represented by $\{-k\}$, $k \geqslant 1$, and the code $\left(\partial_{z_0}^{\lambda_0} \cdots \partial_{z_n}^{\lambda_n} f\right)^*$ is represented by $\{\lambda_0, \ldots, \lambda_n\} \in \mathbb{N}^{n+1}$.

```
 G[x_, tau_] := RandomVariate[NormalDistribution[x, Sqrt[tau]]];
 MCS[f_, t_, tf_, x_, c__, h_, phi_, n_] := (Module[{A, B, U, tau, L, j, l, l1, k1, g, ct},
  If[t == tf, Return[phi[x]]]; tau = RandomVariate[ExponentialDistribution[1]];
```

```
   If[t + tau >= tf, Return [h* codetofunction[f, c, G[x, tf - t], phi]/Exp[-(tf - t)]]];GS=G[x,tau];
        If[c == {Id}, Return[MCS[f, t + tau, tf, GS, ConstantArray[0, n + 1], h/Exp[-tau], phi, n]]];
   U = RandomVariate[UniformDistribution[1]][[1]];
   If[Length[c] == 1, L = fdb[n + 1, -c[[1]]]; g = Ceiling[U*Length[L]]; A = L[[g]][[1]]* MCS[f, t + tau,
        tf, GS, L[[g]][[2]], Length[L]*h/Exp[-tau], phi, n];
    Do[Do[Do[ A = A*MCS[f, t + tau, tf, GS, {-q - L[[g]][[4]][[j]]}, 1, phi, n], {i1, 1,
         L[[g]][[3]][[q]][[j]]}], {j, 1, L[[g]][[5]]}], {q, 1, n}]; Return[A]];
   l1 = 1 + Sum[Length[fdb[n + 1, k2]], {k2, 1, n}] + (n + 1)^2;
   If[U <= 1/l1, A = MCS[f, t + tau, tf, GS, ConstantArray[0, n + 1], 1, phi, n];
    Return[MCS[f, t + tau, tf, GS, c + UnitVector[n + 1, 1], l1*A*h/Exp[-tau], phi, n]]]; If[U <= (1 + (n
         + 1)^2)/l1, j = Floor[U*l1*n/(1 + (n + 1)^2)];
    l = Ceiling[n*(U - j*(1 + (n + 1)^2)/l1)]; A = MCS[f, t + tau, tf, GS, {-j - 1}, 1, phi, n];
    B = MCS[f, t + tau, tf, GS, {-l - 1}, 1, phi, n];
    Return[MCS[f, t + tau, tf, GS, c + UnitVector[n + 1, l + 1] + UnitVector[n + 1, j + 1],
         -l1*A*B*h/Exp[-tau]/2, phi, n]]]; g = Ceiling[U*l1] - 1 - (n + 1)^2;
   k1 = 1; While[k1 <= n, L = fdb[n + 1, k1]; If[g <= Length[L], Break[]]; g = g - Length[L]; k1++];
   A = L[[g]][[1]]* MCS[f, t + tau, tf, GS, L[[g]][[2]], 1, phi, n];
   Do[Do[Do[ A = A*MCS[f, t + tau, tf, GS, {-q - L[[g]][[4]][[j]]}, 1, phi, n], {i1, 1,
        L[[g]][[3]][[q]][[j]]}], {j, 1, L[[g]][[5]]}], {q, 1, n}];
   Return[MCS[f, t + tau, tf, GS, c + UnitVector[n + 1, k1 + 1], A*l1*h/Exp[-tau], phi, n]]]
Sol[f_, t_, tf_, x_, phi_, n2_, n_] := (temp = 0; For[i = 1, i <= n2, i++,
   If[Mod[i, 100000] == 0, Print[i, " sol=", temp/i]]; temp += MCS[f, t, tf, x, {Id}, 1, phi, n];];
        Return[temp/n2])
```

PDE Solution code.

References

[BBC+19] C. Beck, S. Becker, P. Cheridito, A. Jentzen, and A. Neufeld. Deep splitting method for parabolic PDEs. Preprint arXiv:1907.03452, 2019.

[BBH+20] S. Becker, R. Braunwarth, M. Hutzenthaler, A. Jentzen, and Ph. von Wurstemberger. Numerical simulations for full history recursive multilevel Picard approximations for systems of high-dimensional partial differential equations. Preprint arXiv:2005.10206v2, 2020.

[BDM19] S. Bonaccorsi, M. D'Ovidio, and S. Mazzucchi. Probabilistic representation formula for the solution of fractional high-order heat-type equations. *J. Evol. Equ.*, 19:523–558, 2019.

[BEJ19] C. Beck, W. E, and A. Jentzen. Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. *J. Nonlinear Sci.*, 29(4):1563–1619, 2019.

[BHZ19] Y. Bruned, M. Hairer, and L. Zambotti. Algebraic renormalisation of regularity structures. *Invent. Math.*, 215:1039–1156, 2019.

[But63] J.C. Butcher. Coefficients for the study of Runge-Kutta integration processes. *J. Austral. Math. Soc.*, 3:185–201, 1963.

[But10] J.C. Butcher. Trees and numerical methods for ordinary differential equations. *Numerical Algorithms*, 53:153–170, 2010.

[CK99] A. Connes and D. Kreimer. Lessons from quantum field theory: Hopf algebras and spacetime geometries. *Letters in Mathematical Physics*, 48:85–96, 1999.

[CLM08] S. Chakraborty and J.A. López-Mimbela. Nonexplosion of a class of semilinear equations via branching particle representations. *Advances in Appl. Probability*, 40:250–272, 2008.

[CS96] G.M. Constantine and T.H. Savits. A multivariate Faa di Bruno formula with applications. *Trans. Amer. Math. Soc.*, 348(2):503–520, 1996.

[CSTV07] P. Cheridito, H.M. Soner, N. Touzi, and N. Victoir. Second-order backward stochastic differential equations and fully nonlinear parabolic PDEs. *Comm. Pure Appl. Math.*, 60(7):1081–1110, 2007.

[DB02] P. Deuflhard and F. Bornemann. *Scientific Computing with Ordinary Differential Equations*, volume 42 of *Texts in Applied Mathematics*. Springer-Verlag, New York, 2002.

[EHJK19] W. E, M. Hutzenthaler, A. Jentzen, and T. Kruse. On multilevel Picard numerical approximations for high-dimensional nonlinear parabolic partial differential equations and high-dimensional nonlinear backward stochastic differential equations. *Journal of Scientific Computing*, 79:1534–1571, 2019.

[EHJK21] W. E, M. Hutzenthaler, A. Jentzen, and T. Kruse. Multilevel Picard iterations for solving smooth semilinear parabolic heat equations. *Partial Differential Equations and Applications*, 2, 2021.

[Fos21] L. Fossy. Algebraic structures on typed decorated rooted trees. *SIGMA*, 17:1–28, 2021.

[FTW11] A. Fahim, N. Touzi, and X. Warin. A probabilistic numerical method for fully nonlinear parabolic PDEs. *Ann. Appl. Probab.*, 21(4):1322–1364, 2011.

[Gub10] M. Gubinelli. Ramification of rough paths. *J. Differential Equations*, 248(4):693–721, 2010.

[GZZ15] W. Guo, J. Zhang, and J. Zhuo. A monotone scheme for high-dimensional fully nonlinear PDEs. *Ann. Appl. Probab.*, 25(3):1540–1580, 2015.

[HJE17] J. Han, A. Jentzen, and W. E. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. Preprint arXiv:1706.04702, 39 pages, 2017.

[HJE18] 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.

[HJK22] M. Hutzenthaler, A. Jentzen, and T. Kruse. Overcoming the curse of dimensionality in the numerical approximation of parabolic partial differential equations with gradient-dependent nonlinearities. *Found. Comput. Math.*, 22:905–966, 2022.

[HJKN20] M. Hutzenthaler, A. Jentzen, T. Kruse, and T.A. Nguyen. Multilevel Picard approximations for high-dimensional semilinear second-order PDEs with Lipschitz nonlinearities. Preprint arXiv:2009.02484v4, 2020.

[HL12] P. Henry-Labordère. Counterparty risk valuation: a marked branching diffusion approach. Preprint arXiv:1203.2369, 2012.

[HLOT+19] P. Henry-Labordère, N. Oudjane, X. Tan, N. Touzi, and X. Warin. Branching diffusion representation of semilinear PDEs and Monte Carlo approximation. *Ann. Inst. H. Poincaré Probab. Statist.*, 55(1):184–210, 2019.

[HLT21] P. Henry-Labordère and N. Touzi. Branching diffusion representation for nonlinear Cauchy problems and Monte Carlo approximation. *Ann. Appl. Probab.*, 31(5):2350–2375, 2021.

[HLTT14] P. Henry-Labordère, X. Tan, and N. Touzi. A numerical algorithm for a class of BSDEs via the branching process. *Stochastic Processes and their Applications*, 124(2):1112–1140, 2014.

[HLW06] E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration*, volume 31 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 2006.

[HLZ20] S. Huang, G. Liang, and T. Zariphopoulou. An approximation scheme for semilinear parabolic PDEs with convex and coercive Hamiltonians. *SIAM J. Control Optim.*, 58(1):165–191, 2020.

[HPW20] C. Huré, H. Pham, and X. Warin. Deep backward schemes for high-dimensional nonlinear PDEs. *Mathematics of Computation*, 2020.

[INW69] N. Ikeda, M. Nagasawa, and S. Watanabe. Branching Markov processes I, II, III. *J. Math. Kyoto Univ.*, 8-9:233–278, 365–410, 95–160, 1968-1969.

[Kry83] N.V. Krylov. Boundedly nonhomogeneous elliptic and parabolic equations. *Math. USSR, Izv.*, 20:459–492, 1983.

[KZZ15] T. Kong, W. Zhao, and T. Zhou. Probabilistic high order numerical schemes for fully nonlinear parabolic PDEs. *Commun. Comput. Phys.*, 18(5):1482–1503, 2015.

[LLP22] W. Lefebvre, G. Loeper, and H. Pham. Differential learning methods for solving fully nonlinear PDEs. Preprint arXiv:2205.09815, 2022.

[McK75] H.P. McKean. Application of Brownian motion to the equation of Kolmogorov-Petrovskii-Piskunov. *Comm. Pure Appl. Math.*, 28(3):323–331, 1975.

[MMMKV17] R.I. McLachlan, K. Modin, H. Munthe-Kaas, and O. Verdier. Butcher series: a story of rooted trees and numerical methods for evolution equations. *Asia Pac. Math. Newsl.*, 7(1):1–11, 2017.

[NW22] A. Neufeld and S. Wu. Multilevel Picard approximation algorithm for semilinear partial integro-differential equations and its complexity analysis. Preprint arXiv:2205.09639, 2022.

[Pen91] S. Peng. Probabilistic interpretation for systems of quasilinear parabolic partial differential equations. *Stochastics Stochastics Rep.*, 37(1-2):61–74, 1991.

[PP92] É. Pardoux and S. Peng. Backward stochastic differential equations and quasilinear parabolic partial differential equations. In *Stochastic partial differential equations and their applications (Charlotte, NC, 1991)*, volume 176 of *Lecture Notes in Control and Inform. Sci.*, pages 200–217. Springer, Berlin, 1992.

[PP22a] G. Penent and N. Privault. Existence and probabilistic representation of the solutions of semilinear parabolic PDEs with fractional Laplacians. *Stochastics and Partial Differential Equations: Analysis and Computations*, 10:446–474, 2022.

[PP22b] G. Penent and N. Privault. Numerical evaluation of ODE solutions by Monte Carlo enumeration of Butcher series. *BIT Numerical Mathematics*, 62:1921–1944, 2022.

[PWG21] H. Pham, X. Warin, and M. Germain. Neural networks-based backward scheme for fully nonlinear PDEs. *Partial Differ. Equ. Appl.*, 2(1):Paper No. 16, 24, 2021.

[Sko64] A.V. Skorokhod. Branching diffusion processes. *Teor. Verojatnost. i. Primenen.*, 9:492–497, 1964.

[SS18] J. Sirignano and K. Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

[STZ12] H.M. Soner, N. Touzi, and J. Zhang. Wellposedness of second order backward SDEs. *Probab. Theory Related Fields*, 153(1-2):149–190, 2012.

[Tan13] X. Tan. A splitting method for fully nonlinear degenerate parabolic PDEs. *Electron. J. Probab.*, 18:no. 15, 24, 2013.