Higher-order deep solver of non-linear PDEs implied by a non-linear discrete Clark–Ocone formula

Jiro Akahori¹, Yui Furuichi¹ and Kaori Okuma¹

¹Department of Mathematical Sciences, Ritsumeikan University, 1-1-1 Nojihigashi, Kusatsu, Shiga 525-8577, Japan

*Corresponding author: kaori.okuma@gmail.com

Received October 01, 2021, Accepted December 18, 2021

Abstract
In the present paper, we introduce a variant of the numerical scheme called the deep solver of PDE. Our scheme is based on the non-linear version of the discrete-time Clark–Ocone formula, which describes the convergent expansion of the error terms. Our new scheme incorporates the higher-order error terms, which we conjecture to stabilize the stochastic gradient descent procedure, and also the irregularities in the driver and the terminal function of the associated forward-backward stochastic differential equation.

Keywords deep solver of PDE, non-linear Clark–Ocone formula, stochastic gradient descent, forward-backward stochastic differential equation

Research Activity Group Mathematical Finance

1. Introduction

Recently, a new type of numerical scheme relying on a deep learning algorithm for solving nonlinear parabolic partial differential equation (PDE) in high dimension has been proposed and paid a lot of attention. The PDE is of the following type:

\[
\partial_t u + Lu + f((\cdot, \cdot), u, \sigma^T \nabla_x u) = 0, \quad \text{on } [0, T) \times \mathbb{R}^d,
\]

\[
u(T, \cdot) = g, \quad \text{on } \mathbb{R}^d
\]

with a non-linearity in the solution \(u\) itself and its gradient via the function \(f(t, x, y, z)\) defined on \([0, T) \times \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d\), a terminal condition \(g\), and a second-order differential operator \(L\) defined by

\[
L u := \frac{1}{2} \text{Tr}(\sigma \sigma^T \nabla_x^2 u) + \mu \cdot \nabla_x u,
\]

where \(\mu\) is a function defined on \([0, T) \times \mathbb{R}^d\) with values in \(\mathbb{R}^d\), \(\sigma\) is a function defined on \([0, T) \times \mathbb{R}^d\) with values in \(M^d\) the set of \(d \times d\) matrices.

Recall that \(L\) is the infinitesimal generator of the semigroup associated with the diffusion process given as a solution to the following stochastic differential equation:

\[
X_t = x_0 + \int_0^t \mu(s, X_s) \, ds + \int_0^t \sigma(s, X_s) \, dW_s, \quad 0 \leq t \leq T,
\]

where \(W\) is a \(d\)-dimensional Wiener process on some probability space \((\Omega, \mathcal{F}, \mathbb{P})\) equipped with a filtration \(\mathbb{F} = (\mathcal{F}_t)_{0 \leq t \leq T}\) satisfying the usual conditions.

A probabilistic numerical scheme for solving the PDE (1) is proposed in [1], which relies on the so-called nonlinear Feynman–Kac formula describing the relation between the solution \(u\) of (1) and the pair of the solution \(X\) of the forward stochastic differential equation (SDE) (2) and the solution to the following backward stochastic differential equation (BSDE): for \(0 \leq t \leq T\),

\[
Y_t = g(X_T) + \int_t^T f(s, X_s, Y_s, Z_s) \, ds - \int_t^T Z_s \cdot dW_s.
\]

The formula characterizes the solution \(u\) of (1) as

\[
Y_t = u(t, X_t), \quad 0 \leq t \leq T,
\]

and provided that \(u\) is smooth, we have

\[
Z_t = \sigma(t, X_t) \nabla_x u(t, X_t), \quad 0 \leq t \leq T.
\]

More importantly, the method proposed by W. E, J. Han and A. Jentzen [1], which will be summarized in Section 3.1, heavily incorporates a deep learning algorithm, that is, the neural network approximation with stochastic gradient descent (SGD), and shows some desirable dimension-free natures. There have been already many variants appearing in the literature, and this type of algorithm is now referred to as “deep solver” (of PDE).

In the present paper, we also propose a variant of the deep solver algorithm, based on a non-linear version of the discrete-time Clark–Ocone formula. The Clark–Ocone formula states that the integrand of the stochastic integral in the martingale representation of a Wiener functional can be expressed in terms of the conditional expectation of its Malliavin derivatives (see (4) below). Among many applications of the formula, the recent papers [2] and [3] rely on the formula to reduce the variance of the expectation \(E[X]\), and then derive efficient algorithms for Monte-Carlo simulation of \(E[X]\). We note that their methods can be regarded as a linear but efficient —higher-order— version of the above-mentioned deep solver, as the forward-backward stochastic differential equation (3) can be regarded as a non-linear version of the Clark–Ocone formula (see Section 2.2).

We will introduce, in Section 2.2, a non-linear ver-
sion of the discrete-time Clark–Ocone formula, which is a variant of the formula proposed by the first and the third authors together with T. Amaba in [4]. Then, a new deep solver type algorithm for solving the PDE (1), which can be regarded as a non-linear version of the algorithms in [2] or [3], will be proposed in Section 3.2. The algorithm reduces the variance, and therefore is more efficient in terms of the SGD where a variant of the law of large number and the central limit theorem control the efficiency. Some simple numerical examples are given in Section 3.3.

2. Non-linear discrete Clark–Ocone formula

2.1 The Clark–Ocone formula and its discrete-time analogue

Let \( W \) be the Wiener process and \( X \) be a Wiener functional in \( \mathbb{D}^{1,2} \), a square integrable “differentiable” function of \( W \). Then, for \( T > 0 \), it holds that

\[
X = E[X] + \int_0^T E[D_sX|\mathcal{F}_s] dW_s, \quad (4)
\]

where \( D_s \) means the Malliavin derivative (evaluated at \( s \)). The formula plays a very important role in Malliavin calculus. For example, a simplified proof of the logarithmic Sobolev inequality is given by the formula. In the context of mathematical finance, the formula gives an alternative description of the hedging portfolio in terms of Malliavin derivatives. See e.g. [5] for details.

A discrete-time version of Clark–Ocone formula was introduced in [4], in view of applications to stochastic numerical analysis. Here we recall the formula. The dimension \( d \) is set to be one, following the convention in [4]. Let \( \Delta W_k = W_{k\Delta t} - W_{(k-1)\Delta t} \) for \( k \in \mathbb{N} \), where \( \Delta t := T/N \) (\( N \in \mathbb{N} \)) is a fixed constant. Then, for a fixed \( n \), the random variable \( (\Delta W_1, \ldots, \Delta W_n) \) is distributed as \( N(0, \Delta tI) \). Let \( \mathcal{G}_k, k = 1, \ldots, N \), be the sigma-algebra generated by \( (\Delta W_1, \ldots, \Delta W_k) \). Note that \( (\mathcal{G}_k)_{k=0}^N \) is a filtration, and

\[
L^2(\mathcal{G}_N, P) \approx L^2(\mathbb{R}^N, \mu^N),
\]

where

\[
\mu^N(dx) = \frac{1}{(2\pi\Delta t)^{N/2}} e^{-\frac{|x|^2}{2\Delta t}} dx.
\]

**Theorem 1** (A discrete version of Clark–Ocone formula [4]) For \( X \in L^2(\mathcal{G}_N, P) \approx L^2(\mathbb{R}^N, \mu^N) \), we have the following \( L^2 \)-convergent series expansion:

\[
X - E[X] = \sum_{m=1}^N \sum_{l=1}^N \frac{(\Delta t)^{m/2}}{\sqrt{m!}} E[\partial_l^{m} X|\mathcal{G}_{l-1}] H_m \left( \frac{\Delta W_1}{\sqrt{\Delta t}} \right),
\]

where \( H_m \) is the \( m \)-th Hermite polynomial for \( m \in \mathbb{Z}_+ \):

\[
H_m(x) = \frac{(-1)^m}{\sqrt{m!}} e^{\frac{x^2}{2}} \frac{d^m}{dx^m} e^{-\frac{x^2}{2}} \quad (m \in \mathbb{Z}_+).
\]

*Here the differentiations are understood in the distribution sense.*

The following are established in [4]:

- When \( X^N \) and \( X \) are smooth, the following central limit theorem for \( n \)-th order approximation holds:

\[
\begin{align*}
\text{Err}_N(0) & \sim (\Delta t)^{-1/2} \text{Err}_N(1) \\
& \quad \vdots \\
& \sim (\Delta t)^{-n/2} \text{Err}_N(n),
\end{align*}
\]

\[
\begin{pmatrix}
\int_0^T E[D_sX|\mathcal{G}_s] dW_s \\
\frac{1}{\sqrt{2}} \int_0^T E[D_s^2X|\mathcal{G}_s] dB_s^1 \\
\vdots \\
\frac{1}{\sqrt{(n+1)!}} \int_0^T E[D_s^{n+1}X|\mathcal{G}_s] dB_s^n
\end{pmatrix}
\]

in law when \( |\Delta t| \to 0 \), where

\[
\text{Err}_N(k) := \sum_{m=1}^N \sum_{l=1}^N \frac{(\Delta t)^{m}}{\sqrt{m!}} E[\partial_l^{m} X|\mathcal{G}_{l-1}] H_m \left( \frac{\Delta W_1}{\sqrt{\Delta t}} \right)
\]

and \( (B_1, \ldots, B^n) = (B_1^1, \ldots, B_1^n)_{0 \leq t \leq T} \) \( n \)-th Brownian motion is independent of \( W = (W_t)_{0 \leq t \leq T} \).

- If \( X \) is only fractionally differentiable for \( s \in (0, 1) \), then the mean squared error of the first order approximation is \( O(N^{-s/2}) \).

Furthermore, in [6], it is shown that the mean squared error of \( n \)-th approximation is \( O(N^{-n/2}) \), if \( X \) is only fractionally differentiable for \( s \in (0, 1) \). In the context of mathematical finance, the result indicates that gamma hedge and higher-order hedge are useful for an option that is difficult to be hedged as discontinuity of the corresponding pay-off function, for instance, binary option.

2.2 Non-linear discrete Clark–Ocone formula

The representation \( Z_t = D_tY_t \), where \( Z_t \) and \( Y_t \) are the solution of (3), is shown in [7]. Substituting the representation to (3), we obtain

\[
Y_t = g(X_T) + \int_0^T f(s, X_s, Y_s, E[D_sX|\mathcal{F}_s]) ds \\
- \int_0^T E[D_sY|\mathcal{F}_s] dW_s,
\]

where \( X_T \) and \( Y_T \) are given by Section 1. If \( f \equiv 0 \), the above is reduced to (4) and thus (5) can be viewed as non-linear Clark–Ocone formula.

Let us consider a discrete-time analog of the equation (3) as follows. Here we work on any \( d \geq 1 \).

**Definition 2** For a given adapted process \( X^N \) and \( g_T \in L^2(\mathcal{G}_N, P) \), if there exist adapted process \( Y^N \) and \( Z^N \) such that

\[
Y_j^N = g_T + \sum_{j=t}^{N-1} f_j(X_j^N, Y_j^N, Z_j^N) \Delta t \\
+ \sum_{j=t}^{N} Z_j^N \cdot \Delta W_j + M_t,
\]
where $M$ is a martingale orthogonal to $\sum_{j=1}^{N} Z_j \cdot \Delta W_j$, then we call the pair of adapted process $Y^N$ and the $d$-dimensional predictable process $Z^N$ called a solution to the discrete BSDE (6).

**Theorem 3** (Non-linear discrete Clark–Ocone formula) Let $y \mapsto y - f(t,x,y,z)\Delta t : \mathbb{R} \to \mathbb{R}$ be a surjective. Then, there is a solution to the discrete BSDE (6), and the solution is given by the following recursive equation: Let $Y_N = g_T$ and $Z_N = E[\theta_N g_T]$, and for $j = N - 1, N - 2, \ldots , 1$,

$$Y_{j-1} - f((j-1)\Delta t, x_{j-1}, y_{j-1}, E[\partial_y Y_{j}]|\cal{G}_{j-1}) \Delta t = E[Y_{j}|\cal{G}_{j-1}],$$

and

$$Z_{j-1} = E[\partial_y Y_{j}|\cal{G}_{j-1}],$$

where we have the following non-linear Clark–Ocone formula: for $j = N, N - 1, \ldots, 1$,

$$Y_{j-1} - Y_j - f((j-1)\Delta t, x_{j-1}, y_{j-1}, E[\partial_y Y_{j}]|\cal{G}_{j-1}) \Delta t = \sum_{m=1}^{N} \sum_{k_1+\ldots+k_m=m} \frac{(\Delta t)^{m/2}}{\sqrt{m!}} E[\theta^{k_1}_{\Delta W_{j_1}} \cdots \theta^{k_m}_{\Delta W_{j_m}} Y_{j} | \cal{G}_{j-1}]$$

$$\times \prod_{i=1}^{d} H_{k_i} \left( \frac{\Delta W_j}{\sqrt{\Delta t}} \right).$$

Moreover, if $y \mapsto f(t, x, y, z)\Delta t - y : \mathbb{R} \to \mathbb{R}$ be a bijection, the solution is unique.

**Proof** Here we only prove the case of $d = 1$. General cases follow only by a slight modification.

We can expand $g_T$ partially with respect to $\Delta W_N$ as

$$g_T = E[\theta_N|\cal{G}_{N-1}]$$

$$+ \sum_{m=1}^{\infty} \frac{(\Delta t)^{m/2}}{\sqrt{m!}} E[\partial^{m}_{\Delta W_{N}} Y_{N} | \cal{G}_{N-1}]$$

$$H_m \left( \frac{\Delta W_N}{\sqrt{\Delta t}} \right)$$

$$\iff Y_{N} = E[g_T | \cal{G}_{N-1}] + Z_{N} \cdot \Delta W_N$$

$$+ \sum_{m=2}^{\infty} \frac{(\Delta t)^{m/2}}{\sqrt{m!}} E[\partial^{m}_{\Delta W_{N}} Y_{N} | \cal{G}_{N-1}]$$

$$H_m \left( \frac{\Delta W_N}{\sqrt{\Delta t}} \right).$$

Since $y \mapsto f(t, x, y, z) - y$ is surjective, we can find $Y_{N-1}$ such that (7) holds for $j = N - 1$, and we can set

$$M_N - M_{N-1} = \sum_{m=2}^{\infty} \frac{(\Delta t)^{m/2}}{\sqrt{m!}} E[\partial^{m}_{\Delta W_{N}} Y_{N} | \cal{G}_{N-1}] H_m \left( \frac{\Delta W_N}{\sqrt{\Delta t}} \right).$$

Next, we can expand $y_{N-1}$ with respect to $\Delta W_{N-1}$, and by the same argument we can find $Y_{N-2}$ and $M_{N-2} - M_{N-3}$.

This procedure can be repeated until we find $Y_0^N$. The constructed martingale

$$M_N^N = M_0^N + \sum_{i=1}^{N} \sum_{m=2}^{\infty} E[\partial^m Y_i|\cal{G}_{i-1}] H_m \left( \frac{\Delta W_i}{\sqrt{\Delta t}} \right)$$

for $j = 1, \ldots, N$ is clearly orthogonal to

$$\sum_{i=1}^{j} Z_i^N \Delta W_i = \sum_{i=1}^{j} E[\partial_y Y_i|\cal{G}_{i-1}] \Delta W_i$$

for $j = 1, \ldots, N$. The uniqueness of $y$ follows from the property that $y \mapsto y - f(t, x, y, z)\Delta t$ is a bijection.

(QED)

3. Higher order deep solver

3.1 Deep solvers of quasi-linear PDE

The method proposed by W. E, J. Han and A. Jentzen [1] is summarized as follows. (i) The forward process $X$ in equation (2), when it is not directly simulatable, is numerically approximated by an Euler scheme on a time grid: $\pi = \{ t_0 = 0 < t_1 < \cdots < t_N = T \}$,

$$X_{t_{i+1}} = X_{t_i} + \mu(t_i, X_{t_i}) \Delta t_i + \sigma(t_i, X_{t_i}) \cdot \Delta W_{t_i},$$

$$i = 0, \ldots, N - 1, \quad X_0 = x_0.\quad (9)$$

Here, $\Delta t_i := t_{i+1} - t_i$, $\Delta W_{t_i} := W_{t_{i+1}} - W_{t_i}$. (ii) The approximation $u(t_i, X_{t_i})$ of the solution $u$ of the partial differential equation (1) is given by

$$u(t_{i+1}, X_{t_{i+1}}) \approx F(t_i, X_{t_i}, u(t_i, X_{t_i}), \sigma^T(t_i, X_{t_i}) \nabla_x u(t_i, X_{t_i}), \Delta t_i, \Delta W_{t_i})$$

with

$$F(t, x, y, z, h, \Delta) := y - f(t, x, y, z)h + z^T \Delta.\quad (10)$$

(iii) The function $x \in \mathbb{R}^d \mapsto \sigma^T(t_i, x) \nabla_x u(t_i, x)$ for each $i$, as well as $x \mapsto u(t_0, x)$ is approximated by a multi-layer neural network parameterized by $\theta_i: x \in \mathbb{R}^d \mapsto Z_i(x; \theta_i)$ ($i = 1, \ldots, N - 1$). Then, an estimator $\hat{U}_i$ of $u(t_i, X_i)$ for each $i$ can be obtained, forward-recursively in $i$, as

$$\hat{U}_{i+1} = F(t_i, X_{t_i}, \hat{U}_i, Z_i(t_i; \theta_i), \Delta t_i, \Delta W_{t_i}),$$

$$i = 1, \ldots, N - 1.\quad (iv)$$ The parameter $\theta$ is obtained via SGD as the unique minimizer of

$$\theta \mapsto E[g(X_{t_N}) - \hat{U}_N(\theta)]^2.$$

Here we also recall the algorithm proposed by C. Hure, H. Pham, and X. Warin [8], on which our new algorithm based.

- Initialize from an estimation $\hat{U}_N$ of $u(t_N, \cdot)$ with $\hat{U}_N = g$.

- For $i = N - 1, \ldots, 0$, given $\hat{U}_{i+1}$, use a deep neural network $\hat{U}_i(\cdot; \theta)$ and compute (by SGD) the minimizer of the expected quadratic loss function:

$$\hat{L}_i(\theta) := E[\hat{U}_{i+1}(X_{t_{i+1}}) - F(t_i, X_{t_i}, \hat{U}_i(X_{t_i}; \theta), \sigma^T(t_i, X_{t_i}) \nabla_x \hat{U}_i(X_{t_i}; \theta), \Delta t_i, \Delta W_{t_i})]^2,$$

$$\hat{\theta} \in \text{argmin}_{\theta \in \mathbb{R}^N} \hat{L}_i(\theta).$$
where \( \nabla_t U_t(X_t; \theta) \) is either

- a neural network function (DBDP1 in [8]), or
- a numerical differentiation of \( \nabla_t U_t \) (DBDP2 in [8]).

Then, update: \( \hat{U}_t = U_t(\cdot; \theta^*_t) \).

3.2 Higher-order deep solver implied by the non-linear discrete Clark–Ocone formula

Our new algorithm replaces \( F \) in (10) with

\[
F_M(t, x, y, z_1, \ldots, z_M, h, \Delta_1, \ldots, \Delta_M)
:= y - f(t, x, y, z_1)h + \sum_{m=1}^{M} h_{\frac{\Delta_m}{2}} z_m \cdot \Delta_m,
\]

where \( z_m \) and \( \Delta_m \in \mathbb{R}^{d_m} \), \( m = 1, \ldots, M \), and

\[
U(t_{i+1}, X_{t_{i+1}}) = F_M(t_i, X_i, U(t_i, X_i; \theta), \sigma^T(t_i, X_i)^m \nabla_x \partial_{\theta} U(t_i, X_i; \theta)_m)_{m=1}^M, \Delta t_i, \left( H_m \left( \Delta_W \sqrt{\Delta t} \times \frac{1}{\sqrt{m!}} \right) \right)_{m=1}^M.
\]

Here we set, for \( (x^1, \ldots, x^d) \in \mathbb{R}^d \) and \( (i_1, \ldots, i_m) \in \{1, \ldots, d\}^m \),

\[
(H_m(x^1, \ldots, x^d))_{i_1, \ldots, i_m} = \prod_{k=1}^{d} H_{i_k} (x^k).
\]

As in the algorithms of [8] explained above, in each step we approximate \( U(t_i, X_i; \theta) \) by a neural network function and \( \nabla_x \partial_{\theta} U \) by either neural network functions or numerical approximation of derivatives, and find minimizer \( \theta \) by SGD.

The algorithm is implied by the \( M \)-th truncation of the formula (8) in Theorem 3, together with the following observations.

**Proposition 4** Assume (9). Then we have the following. (i) For \( j = 0, 1, \ldots, N-1 \), there exist functions \( u_j^N \) such that \( Y_j = u_j^N(X_j) \). (ii) For any smooth function, say, \( g \),

\[
E[\partial_{\Delta t} W_j g(X_j) | F_{j-1}] = \sum_{i=1}^{d} \sigma_i x_{j-1,i} E[\partial_{x_i} g(X_j) | F_{j-1}]
\]

for any \( i \) and \( j \), and (iii)

\[
E[\partial_{x_i} g(X_j) | F_{j-1}] - \partial_{x_i} E[g(X_j) | F_{j-1}] \sim o(\Delta t).
\]

**Proof** The assertion (i) follows from the Markov property of \( X \). The assertions (ii) and (iii) are obvious from the expression (9).

\( \text{(QED)} \)

3.3 Numerical Experiments

We exemplify a numerical experiment, where the parameters are set as \( \sigma = 1, \mu = 0, d = 1, T = 1, x_0 = 1, \)

\[
N = 240,
\]

\[
f(t, x, y, z) = \cos(x) e^{-\frac{t}{T}} - \frac{1}{2} (\sin(x) \cos(x) e^{-t})^2 + \frac{1}{2} (yz)^2,
\]

\( g(x) = \cos(x) \), for which, the explicit analytic solution is \( u(t, x) = e^{(T-t)/2} \cos(x) \) (see 5.1.1 of [8]). The results are indicated in Tables 1 and 2. In all experiments, we used the numerical differentiation in *pytorch* of Python for \( \hat{U} \). Averages and standard deviations are taken out of 7 independent runs. The results imply that the efficiency is improved by using higher order terms.

4. Concluding remark

In the present paper, we proposed a higher-order order version of the deep solver algorithm of [8], which can also be regarded as a non-linear version of the algorithms in [2] and [3], based on the discrete-time Clark–Ocone formula proposed in [4]. The efficiency is based on a variance reduction in SGD, which has not yet been theoretically well studied. The results in [4] and [6] may well imply that the convergence against the number of steps becomes faster, and the irregularity in the driver as well as the terminal function might be handled by our higher-order scheme. These issues are left for the full paper we are preparing.

**References**

[1] W. E, J. Han and A. Jentzen, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations, Commun. Math. Stat. 5 (2017), 349–380.

[2] D. Belomestny, S. Häfele, T. Nagapetyan and M. Urusov, Variance reduction for discretized diffusions via regression, J. Math. Anal. Appl. 458 (2018), 393–418.

[3] M. Sabate-Vidales, D. Siska and L. Szpruch, Unbiased deep solvers for parametric PDEs, arXiv:1810.05094 [q-fin.CP].

[4] J. Akahori, T. Amaba and K. Okuma, A discrete-time Clark–Ocone formula and its application to an error analysis, J. Theor. Probab. 30 (2017), 932–960.

[5] D. Nualart, The Malliavin Calculus and Related Topics, 2nd edn., Springer-Verlag, Berlin, 2006.

[6] T. Nishimura, K. Yasutomi and T. Yuasa, Higher order error estimate of the discrete-time Clark–Ocone formula, arXiv:2101.07087 [math.PR].

[7] N. El Karoui, S. Peng and M. C. Quenez, Backward stochastic differential equations in finance, Math. Finance 7 (1997), 1–71.

[8] C. Huré, H. Pham and X. Warin, Deep backward schemes for high-dimensional nonlinear PDEs, Math. Comp. 89 (2020), 1547–1579.