CQF Project: NAIS-Net Framework for Solving High Dimensional FBSDEs in the Pricing of European Options

Lang Tao, Wang

August 2024

Contents

1 Introduction

In the pricing of derivatives, the resolution of partial differential equations (PDEs) plays a crucial role. While certain PDEs permit analytical solutions, the majority necessitate the use of numerical approximation methods. Classical numerical techniques, such as finite difference methods and finite element methods, encounter substantial challenges, especially when dealing with high-dimensional PDEs-a difficulty commonly referred to as the "curse of dimensionality", as the dimensionality grows, the complexity of the algorithms grows exponentially. In recent years, the rapid advancement of machine learning has broadened its applications across various fields. Notably, the literature(see [2],[3],[5],[15],[16],[17],[18]) has documented the successful application of deep neural networks (DNNs), demonstrating their potential in solving high-dimensional PDEs.

In this report, building on established methods, we first reformulate the PDE-solving process into a system of forward-backward stochastic differential equations (FBSDEs). Based on this transformation, we design a network architecture to achieve numerical approximation. To enhance training stability, we employ the NAIS-Net architecture as recommended in the [2]. The model is implemented using Keras and TensorFlow and validated against the one-dimensional Black-Scholes equation as well as a case study from the [2]. For extended applications, we utilize the rough Bergomi model(rBM) for pricing put options and simulate variance using a **hybrid scheme** proposed in previous studies([8],[9]).

The rest of the report is structured as follows: In Section 2, we review the systems of Forward-Backward Stochastic Differential Equations (FBSDEs) and Backward Stochastic Partial Differential Equation(BSPDE) in the rough Bergomi model. In Section 3, we describe stable architectures for deep learning(NAIS-Net). Meanwhile, we described the links PDE and deep learning. In Section 4, we report preliminary numerical results and extended NAIS-Net framework to the option pricing problem using the rough Bergomi model.

2 Background knowledge

In this section, we first introduce some basics of forward-backward stochastic differential equations (FBSDEs). Both instruments are used to perform our numerical tests in Section 4.

2.1 Forward-backward stochastic differential equation

We consider the following system of stochastic differential equations,

$$\begin{aligned} dX_t &= \mu\left(t, X_t, Y_t, Z_t\right) dt + \sigma\left(t, X_t, Y_t\right) dW_t, \qquad X_0 = \xi \\ dY_t &= \psi\left(t, X_t, Y_t, Z_t\right) dt + Z_t^T \sigma\left(t, X_t, Y_t\right) dW_t, \qquad Y_T = g\left(X_T\right) \end{aligned} \tag{2.1}$$

Where $X_0 = \xi \in R_d$ is the initial condition for forward SDE, and $Y_T = g\left[X_T\right]$ is the terminal condition of the backward SDE. W_t is a vector valued Brownian motion(standard d-dimensional Brownian motion).

Consider the following non-linear PDE,

$$u_t = f\left(t, x, u, Du, D^2 u\right) \tag{2.2}$$

where Du, D^2u represent the gradient and Hessian of u respectively. when function f is given by,

$$f\left(t, x, y, z, \gamma\right) = \psi\left(t, x, y, z\right) - \mu\left(t, x, y, z\right)^T z - \frac{1}{2} Tr\left[\sigma\left(t, x, y\right) \sigma\left(t, x, y\right)^T \gamma\right] \tag{2.3}$$

and the terminal condition of the PDE is given by $u\left(T, x\right) = g\left(x\right)$then it is known,

$$\begin{aligned} Y_t &= u\left(t, X_t\right) \\ Z_t &= \nabla u\left(t, X_t\right) \end{aligned} \tag{2.4}$$

Therefore the solution to the PDE can be obtained by solving (2.1). Note that the described above is for the case when the PDE is linear in D^2u (the PDE is call semi-linear in this case). Similar results can be obtained for the case where f is nonlinear in D^2u.

Now, we focus on a forward-backed stochastic differential equation(FBSDE) of the form,

$$\begin{aligned} dX_t &= \mu\left(t, X_t\right) dt + \sigma\left(t, X_t\right) dW_t, \\ X_0 &= \xi, \\ -dY_t &= f\left(t, X_t, Y_t, Z_t\right) dt - Z_t dW_t, \\ Y_T &= g\left(X_T\right). \end{aligned} \tag{2.5}$$

Suppose X_t is the stock value and it follows,

$$dX_t = \left(r - D\right) X_t dt + \sigma X_t dW_t. \tag{2.6}$$

For simplicity, we assume r is the constant discount rate, D is the constant dividend, and σ is the constant volatility. We only use subscript when it is necessary. Proceeding in the same fashion as in the derivation of Black-Scholes PDE, we construct a Delta-Hedging portfolio $\Pi = Y - \Delta X$, so that the value of the portfolio is deterministic.

$$d\Pi = dY - \Delta dX - D\Delta X dt = \left(\frac{\partial Y}{\partial t} + \frac{1}{2}\sigma^2 X^2 \frac{\partial^2 Y}{\partial X^2}\right) dt + \frac{\partial Y}{\partial X} dX - \Delta dX - D\Delta X dt$$

The term $D\Delta X dt$ arises since the stock pays dividend which decreases the value of the portfolio by the amount of the dividend. If we select $\Delta = \frac{\partial Y}{\partial X}$, from Itô's Lemma, we then have

$$d\Pi = \left(\frac{\partial Y}{\partial t} + \frac{1}{2}\sigma^2 X^2 \frac{\partial^2 Y}{\partial X^2}\right) dt - D\Delta X dt.$$

$$d\Pi = r\Pi dt = r\left(Y - \Delta X\right) dt.$$

This leads to the following Black Scholes PDE,

$$\frac{\partial Y}{\partial t} + \frac{1}{2}\sigma^2 X^2 \frac{\partial^2 Y}{\partial X^2} + (r - D) X \frac{\partial Y}{\partial X} - rY = 0. \tag{2.7}$$

To derive backward-SDE(BSDE), apply Itô's Lemma to Y_t,

$$\begin{aligned} dY &= \left(\frac{\partial Y}{\partial t} + \frac{1}{2}\sigma^2 X^2 \frac{\partial^2 Y}{\partial X^2}\right) dt + \frac{\partial Y}{\partial X} dX \\ &= \left(rY - (r - D) X \frac{\partial Y}{\partial X}\right) dt + \frac{\partial Y}{\partial X}\left((r - D) X dt + \sigma X dW\right) \\ &= rY dt + \sigma X \frac{\partial Y}{\partial X} dW, \end{aligned}$$

or

$$-dY = -rY dt - \sigma X \frac{\partial Y}{\partial X} dW, \tag{2.8}$$

that is $f = -rY$, and $Z = \sigma X \frac{\partial Y}{\partial X}$ in Eq (2.5).

The above statement can be easily extended to a high-dimensional derivative pricing $\left(Y = Y\left(X^1, X^2, \cdots, X^d\right)\right)$, and we have (neglecting subscript t)

$$\begin{aligned} dX^i &= \mu^i\left(t, X^i\right) dt + \sigma^i\left(t, X^i\right) dW^i \\ X_0^i &= \xi^i \\ -dY &= -rY dt - \sum \sigma^i X^i \frac{\partial Y}{\partial X^i} dW^i \qquad (2.9) \\ Y_T &= g\left(X_T^1, X_T^2, \cdots, X_T^d\right) \\ cov\left(dW^i, dW^j\right) &= \rho^{ij} dt \quad \left|\rho^{ij}\right| < 1. \end{aligned}$$

2.2 Backward SPDE

We solve the option pricing problems under rough volatility with backward stochastic partial differential equations(BSPDEs) as mentioned in the [1]. In this section, we introduce some basic notation.

Let $\left(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\in[0,T]}, \mathbb{P}\right)$ be a complete filtered probability space with the filtration $(\mathcal{F}_t)_{t\in[0,T]}$ being the augmented filtration generated by two independent Winner processes W and B. We denoted by $\left(\mathcal{F}_t^W\right)_{t\in[0,T]}$ the augmented filtration generated by the Winner process W. The predictable σ-algebras on $\Omega \times [0,T]$ corresponding to $\left(\mathcal{F}_t^W\right)_{t\in[0,T]}$ and $(\mathcal{F}_t)_{t\in[0,T]}$ denoted by $\mathcal{F}^W$ and $\mathcal{F}$, respectively.

We consider a general stochastic volatility model given under a risk neutral probability measures as

$$\begin{aligned} dS_t &= rS_t dt + S_t\sqrt{V_t}\left(\rho dW_t + \sqrt{1-\rho^2}dB_t\right) \\ S_0 &= s_0 \end{aligned} \tag{2.10}$$

where $\rho \in [-1,1]$ denote the correlation coefficient and the constant r the interest rate. We assume that the stochastic variance process V (V is adapted to the filtration generated by Brownian motion W) is integrable, i.e,

$$E = \left[\int_0^T V_s ds\right] < \infty, \qquad T > 0.$$

For each $(t,s) \in [0,T] \times \mathbb{R}^+$, denote the security price process by $S_\tau^{t,s}$, for $\tau \in [t,T]$ which satisfies the stochastic differential equation (SDE) in (2.10) but with initial time t and initial state s (price at time t). The fair price of a European option with payoff H, as the smallest initial wealth required to finance an admissible (super-replicating) wealth process, is given by

$$P_t(s) \;:=\; E\left[e^{-r(T-t)}H\left(S_T^{t,s}\right)|\mathcal{F}_t\right] \tag{2.11}$$

Taking $X_t = -rt + logS_t$, we may reformulate the above pricing problem,

$$u_t(s) \;:=\; E\left[e^{-r(T-t)}H\left(e^{X_T^{t,x}+rT}\right)|\mathcal{F}_t\right], \quad (t,x) \in [0,T] \times \mathbb{R} \tag{2.12}$$

subject to

$$\begin{aligned} dX_t &= \sqrt{V_s}\left(\rho dW_s + \sqrt{1-\rho^2}dB_s\right) - \frac{V_s}{2}ds, \quad 0 \le t \le s \le T; \\ X_t^{t,x} &= x \end{aligned} \tag{2.13}$$

Obviously, we have the relation $u_t(s) = P_t(e^{x+rt})$. The non-Markovianity of the pair (S,V) (or (X,V)) makes it impossible to characterize the value function $u_t(x)$ with a conventional (deterministic) partial differential equation (PDE). The value function $u_t(x)$ is a random field which together with another random field $\psi_t(s)$ satisfies the following backward stochastic partial differential equation (BSPDE):

$$\begin{aligned} -du_t(x) &= \left[\frac{V_t}{2}D^2u_t(x) + \rho\sqrt{V_t}D\psi_t(x) - \frac{V_t}{2}Du_t(x) - ru_t(x)\right]dt - \psi_t(x)dW_s \\ u_x^T &= H\left(e^{x+rT}\right) \end{aligned} \tag{2.14}$$

A weak solution theory is established for the well-posedness of general nonlinear BSPDEs and associated stochastic Feynman-Kac formula, particularly applicable to 2.14. For the following nonlinear BSPDE:

$$\begin{aligned} -du_t(x) = &\left[\frac{V_t}{2}D^2u_t(x) + \rho\sqrt{V_t}D\psi_t(x) - \frac{V_t}{2}Du_t(x)\right. \\ &\left.+F_t\left(e^x, u_t(x), \sqrt{(1-\rho^2)V_t}Du_t(x), \psi_t(x) + \rho\sqrt{V_t}Du_t(x)\right)\right]dt \\ &- \psi_t(x)dW_s, \qquad (t,x) \in [0,T]\times\mathbb{R} \\ u_x^T = &\, G\left(e^x\right), \qquad x \in \mathbb{R} \end{aligned} \tag{2.15}$$

Noteworthily, BSPDE(2.14)turns out to be a particular case when $F_t(x,y,z,\tilde{z}) \equiv -ry$ and $G(e^x) = H(e^{x+rT})$. We will address the representation relationship between and BSPDE(2.15) and associated FBSDE. Corresponding to BSPDE(2.14), there follows the BSDE:

$$\begin{aligned} -dY_s^{t,x} &= F_s\left(e^{X_s^{t,x}}, Y_s^{t,x}, Z_s^{t,x}, \tilde{Z}_s^{t,x}\right)ds \\ &\quad - Z_s^{\tilde{t},x}dW_s - Z_s^{t,x}dB_s, \quad 0 \le t \le s \le T \\ Y_T^{t,x} &= G\left(X_T^{t,x}\right) \end{aligned} \tag{2.16}$$

where the triple $(Y_s^{t,x}, Z_s^{t,x}, \tilde{Z}_s^{t,x})$ is defined as the solution to BSDE(2.16). The stochastic Feynman-Kac formula is the probabilistic representation of weak solution to BSPDE(2.15) via the solution associated BSDE(2.16) coupled with the forward SDE(2.13), accordingly.

Let (u,ψ) be a weak solution of BSPDE(2.15), then (u,ψ) admits a version(denoted by itself) satisfying,

$$\begin{aligned} u_\tau(X_\tau^{t,x}) &= Y_\tau^{t,x} \\ Z_\tau^{t,x} &= \sqrt{(1-\rho^2)V_\tau}Du_\tau(X_\tau^{t,x}) \\ \tilde{Z}_\tau^{t,x} &= \psi_\tau(X_\tau^{t,x}) + \rho\sqrt{V_\tau}Du_\tau(X_\tau^{t,x}) \end{aligned}$$

Based on the above framework and relationship between SPDE, we construct the deep learning-based numerical methods.

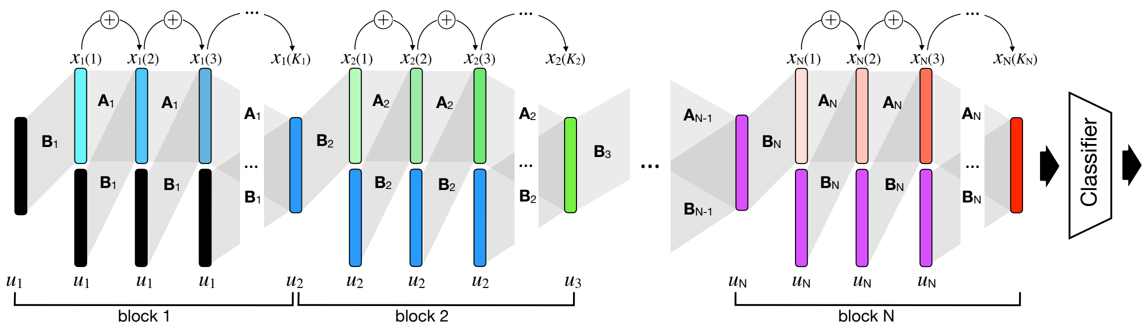


Figure 3.1: **NAIS-Net architecture**. Each block represents a time-invariant iterative process as the first layer in the i-th block, $x_i(1)$, is unrolled into a pattern-dependent number, K_i, of processing stages, using weight matrices $\mathbf{A}_i$ and $\mathbf{B}_i$. The skip connections from the input, u_i, to all layers in block i make the process non-autonomous. Blocks can be chained together (each block modeling a different latent space) by passing final latent representation, $x_i(K_i)$, of block i as the input to block $i+1$.

3 Deep Learning Method for FBSDE

In the section above we described how the partial differential equation is related to the forward-backward stochastic differential equation. In this section we discuss the numerical approximation using a neural network.

3.1 NAIS-Net Architecture

We use the non-autonomous network architecture proposed in [2] called NAIS-Net(as figure 3.1). The reason we use the NAIS-Net model is that the resulting network is globally asymptotically stable for every initial condition. This is a crucial property for the PDE problem described in Section 2. NAIS-Net, a non-autonomous input-output stable neural network, fully connected layer is defined by

$$x(k+1) = x(k) + h\sigma(Ax(k) + Bu + C) \tag{3.1}$$

In this setting, A, B and C are trainable parameters. Parameter A refers to the traditional weight matrix and C refers to the bias. The activation σ is vector of (element-wise) instances of an activation function, in this report, we only consider the Sine activation function. The extra term Bu is made of a matrix B and the input of the network u. Involving the input u makes the system non-autonomous, and the output of the system input-dependent. Moreover, the weight matrix is constrained to be symmetric definite negative:

$$A = -R^T R - \epsilon I \tag{3.2}$$

Algorithm 1 Fully Connected Reprojection

```
Input: $R \in \mathbb{R}^{\tilde{n}\times n}$, $\tilde{n} \leq n$, $\delta = 1 - 2\epsilon$, $\epsilon \in (0, 0.5)$.

if $\|R^T R\|_F > \delta$ then
    $\tilde{R} \leftarrow \sqrt{\delta}\frac{R}{\sqrt{\|R^T R\|_F}}$
else
    $\tilde{R} \leftarrow R$
end if
Output: $\tilde{R}$
```

Figure 3.2: Proposed algorithms for enforcing stability.

where ϵ is a hyper-parameter that ensures the eigenvalues are strictly negative, $0 < \epsilon$ ($\epsilon = 0.01$ in our case). An additional constraint is proposed on the Frobenius norm $\|R^T R\|_F$ with the algorithm 1. The projection used in this setting forces the weights of the neural network to stay within the set of feasible solutions and makes it more robust.

3.2 Solving FBSDEs using Deep Learning

The learning of the solution will be based on the sample paths of the FBSDEs, which are linked to the PDE solution in (2.4). Paths of the FBSDEs will be produced by a time discretization algorithm with samples of the Brownian motion W_t.

Let $0 = t_0 < \cdots < t_N = T$ be a uniform partition of $[0, T]$. On each interval $[t_n, t_{n+1}]$, define time and Brownian motion increments as

$$\Delta t_n = t_{n+1} - t_n, \quad \Delta W_n = W_{t_{n+1}} - W_{t_n}. \tag{3.3}$$

Denoting X_{t_n}, Y_{t_n} and Z_{t_n} by X_n, Y_n and Z_n, respectively, and applying the Euler–Maruyama scheme to the FBSDEs (2.1), respectively, we have

$$X_{n+1} \approx X_n + \mu(t_n, X_n, Y_n, Z_n)\Delta t_n + \sigma(t_n, X_n, Y_n)\Delta W_n, \tag{3.4}$$

$$Y_{n+1} \approx Y_n + \varphi(t_n, X_n, Y_n, Z_n)\Delta t_n + Z_n^T\sigma(t_n, X_n, Y_n)\Delta W_n. \tag{3.5}$$

Due to the relationship with the parabolic PDE, the solution to the parabolic PDE provides an alternative representation for Y_{n+1} and Z_{n+1},

$$Y_{n+1} = u(t_{n+1}, X_{n+1}), \tag{3.6}$$

$$Z_{n+1} = \nabla u(t_{n+1}, X_{n+1}). \tag{3.7}$$

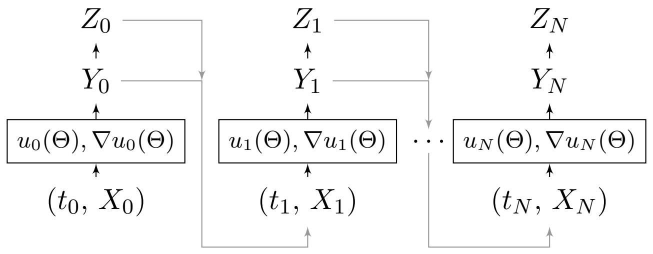

Z_0
Y_0
$u_0(\Theta), \nabla u_0(\Theta)$
(t_0, X_0)
Z_1
Y_1
$u_1(\Theta), \nabla u_1(\Theta)$
(t_1, X_1)
$\cdots$
Z_N
Y_N
$u_N(\Theta), \nabla u_N(\Theta)$
(t_N, X_N)

Figure 3.3: Approximation of a system of FBSDE using the same parametric approximation for every time step.

In this report, fully connected networks of L hidden layers will be used, which are given in the following form,

$$f_\theta(\mathbf{x}) = \mathbf{W}\sigma\circ(\cdots(\mathbf{x}^{[1]}+\mathbf{A}^{[1]}\sigma\circ(\mathbf{x}^{[0]}+\mathbf{A}^{[0]}(\mathbf{x}^{[0]})+\mathbf{B}^{[0]}u+\mathbf{C}^{[0]})+\mathbf{B}^{[1]}u+\mathbf{C}^{[1]})\cdots)+\mathbf{b} \quad (3.8)$$

where $A^{[1]},\cdots,A^{[L]},B^{[1]},\cdots,B^{[L]}$ and $C^{[1]},\cdots,C^{[L]}$ are the weight matrices and bias unknowns,respectively, W, b is parameters for output layer, denoted collectively by θ, to be optimized via the training, $\sigma(x)$ is the activation function and $\circ$ is the application of the activation function σ applied to a vector quantity component-wisely.

As mentioned in [2],The FBSNNs trains a network $u_\theta(t, x)$ that directly approximates the solution to the PDE (2.3) in some region in the (t, x) space. The network has a fixed size of number of hidden layers and neurons per layer. The main idea is explained in Figure 3.3. The algorithm can be organized as follows.

1. The initial value $X_0 = x_0$ is given. Evaluate Y_0 and Z_0 using the network

$$Y_0 = u_\theta(t_0, X_0), \quad Z_0 = \nabla u_\theta(t_0, X_0). \quad (3.9)$$

   The gradient above is calculated by an automatic differentiation.

2. On each time interval $[t_n, t_{n+1}]$, use the Euler–Maruyama scheme (3.4) to calculate X_{n+1}, and use the network for Y_{n+1} and Z_{n+1}, i.e.

$$\begin{aligned} X_{n+1} &= X_n + \mu(t_n, X_n, Y_n, Z_n)\Delta t_n + \sigma(t_n, X_n, Y_n)\Delta W_n, \\ Y_{n+1} &= u_\theta(t_{n+1}, X_{n+1}), \\ Z_{n+1} &= \nabla u_\theta(t_{n+1}, X_{n+1}). \end{aligned} \quad (3.10)$$

   On the other hand, calculate a reference value $Y^\star_{n+1}$ using the Euler–Maruyama scheme (3.5)

$$Y^\star_{n+1} = Y_n + \varphi(t_n, X_n, Y_n, Z_n)\Delta t_n + Z_n^T\sigma(t_n, X_n, Y_n)\Delta W_n. \quad (3.11)$$

3. The loss function is taken as a Monte Carlo approximation of

$$\mathbb{E}\left[\sum_{n=1}^{N}\|Y_n - Y_n^{\star}\|^2 + \|Y_N - g(X_N)\|^2 + \|Z_N - \nabla g(X_N)\|^2\right]. \tag{3.12}$$

As noted in [5], While the discrete stochastic process $\{Y_n\}$ can be expected to approach a continuous stochastic process as defined in the backward SDE (2.1), the question whether the discrete sequence of random variables $\{Y_n^*\}$ will converge to the same stochastic process is not clear. As a result, the rate and extent for the difference between $\{Y_n\}$ and $\{Y_n^*\}$, the loss function, approaching to zero is not certain. Thus, in the rough Bergomi model, we modify the loss function,

$$L[u_\theta; x_0] = \frac{1}{M}\left[\sum_{\omega}\frac{1}{N}\sum_{n=1}^{N}\|Y_n - Y_n^{\star}\|^2 + \beta_1\|Y_N - g(X_N)\|^2 + \beta_2\|Z_N - \nabla g(X_N)\|^2\right] \tag{3.13}$$

where M serves as the batch size of the training and ω denotes any instance of sampling of the discretized Brownian motion $W_n, 0 \le n \le N-1$, and β_1, β_2 are the penalty parameters for the terminal conditions. The averaging factor $1/N$ is introduced for consistency consideration as the reduction of the loss function as N increases, when applied to the exact solution, is expected.

4 Numerical Results

4.1 European Option for 1-dimensional Black-Scholes equation

We will pick an 1-dimensional Black-Scholes equation since for this PDE we have analytical solutions and the results can be easily visualized. We consider the 1-dimensional Black-Scholes model with constant drift and short rate of 0.05 and constant volatility of 0.4. We either start the underlier at a spot of 1.0 or vary it uniformly between 0.5 and 1.5. We consider a long call at K=100%($strike = K * X0$). Calls have maturity of a year,1.0. We discretize time with 48 time steps.

We use mini-batches of size 100. Our network have 6 layers of sizes 2(have an additional t input), 110 *4, and 1 for one initial value; 51, 4*256, and 50 for varied initial value. We used Sine as an activation function for all hided layers except the output layer, for which we used identity. Learning rate is 1e-3 and we use Adam with Tensor Flow standard parameters. We run 20000 mini-batches for one initial value and 15000 mini-batches for varied initial values.

Figure 4.1 and 4.2 shows that the loss functional decays quite quickly as a function of number mini-batches run and how price at fixed and variant initial values converge up

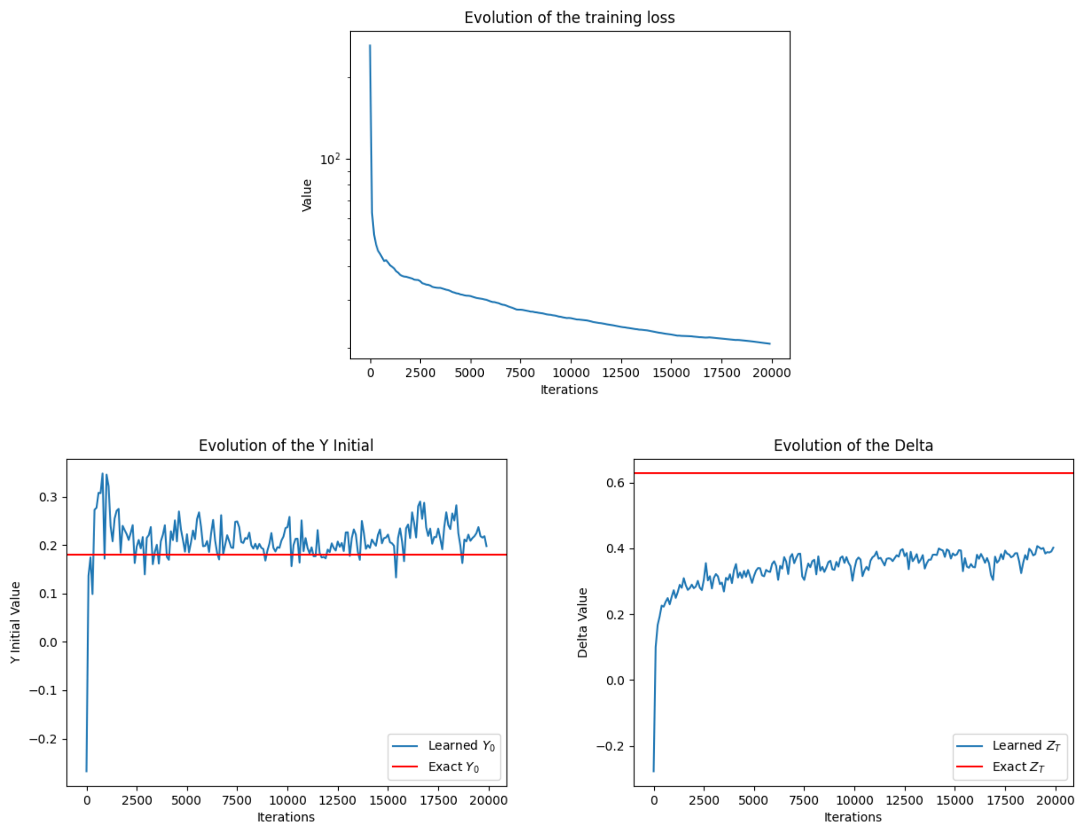


Figure 4.1: Fixed initial values. Upper: loss function over mini-batch number. Lower-left: convergence of price. Lower-right: convergence of delta.

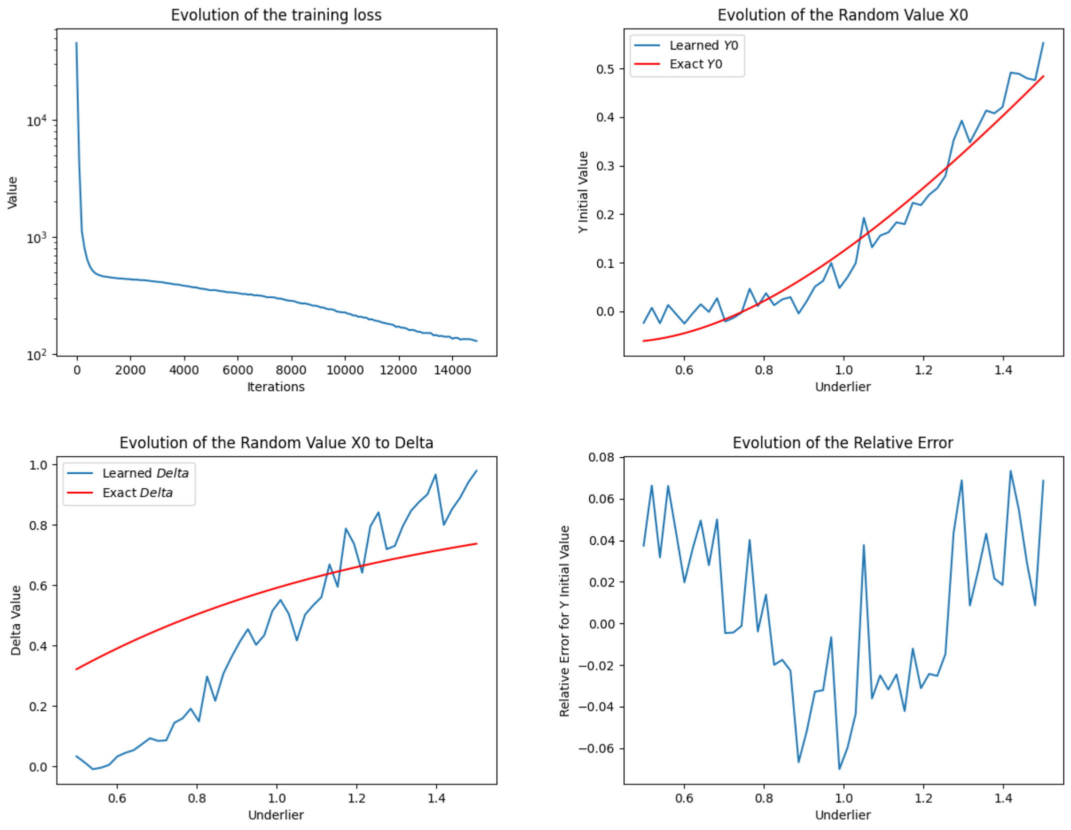


Figure 4.2: Random initial values. Upper-left: Loss function over mini-batch number. Upper-right: comparison of initial Y network and analytical solution. Lower-left: comparison of terminal delta and analytical delta. Lower-right: Relative error between Y0 and analytical solution.

to good accuracy, but the convergence of Delta is not ideal. For the variant with initial values, The precision is degraded, and the results are oscillatory. The indicates the network training might dominate the error compared to time discretization error. To accuracy, we can study further([5]has mentioned the use of the Richardson extrapolation method.). Code can be found in the "Code" files: Fixed initial values, *BSmodel_1_dimension.html*; Random initial values, *BSmodel_multi_initial_dnn.html*.

Table 4.1: comparison of initial Y network and analytical solution

Dimensions	Exact solution	Estimated value	Relative error	Standard deviation(estimated)
d =1	0.1802	0.2075	0.0273	5.9604e-08

4.2 100-dimensional Black-Scholes-Barenblatt equation

Consider the following 100-dimensional Black-Scholes-Barenblatt(BSB) equation from [2], assessed the viability of model with Keras and Tensor Flow. For $t \in [0, T]$ and $x \in \mathbb{R}^d$, the function $u(t, x)$ satisfies

$$\begin{aligned} u_t &= -\frac{1}{2}\mathrm{Tr}\left[\sigma^2 diag(X^2) D^2 u\right] + r\left(u - (Du)^T x\right) \\ u(T, x) &= g(x) \end{aligned} \tag{4.1}$$

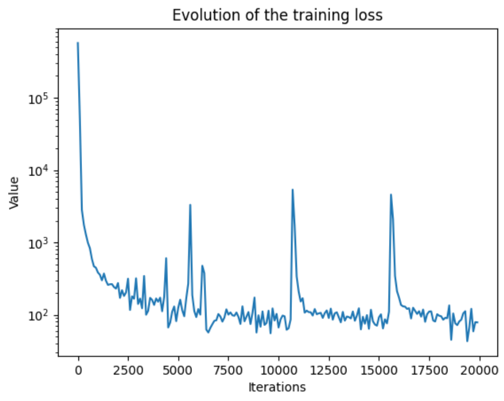
The above PDE is linked to the FBSDEs

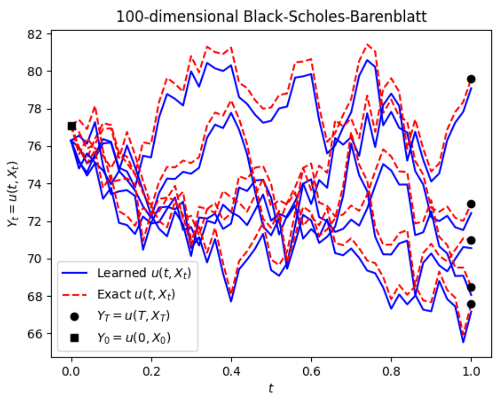
$$\begin{aligned} dX_t &= \sigma diag(X_t) dW_t \\ X_0 &= \xi \\ dY_t &= r\left(Y_t - Z_t^T\right) dt + \sigma Z_t^T diag(X_t) dW_t \\ Y_T &= g(X_T) \end{aligned} \tag{4.2}$$

where $T = 1, \sigma = 0.4, r = 0.05, \xi = (1, 0.5, 1, 0.5, \cdots, 1, 0.5) \in \mathbb{R}^{100}$, and $g(x) = \|x\|^2$. The PDE 4.1 admits the explicit solution

$$u(t, x) = e^{(r+\sigma^2)(T-t)} \|x\|^2, \tag{4.3}$$

We use a 6-layer fully connected NAIS-Net for $u_\theta(t, x)$ with 4 hidden layers, each having 256 neurons. The activation is the Sine function as suggested by [2].We train the network with Adam optimizer with learning rate 1e-3, each for 20000 steps. The batch size is $M = 100$. The time steps are $N = 50$. The training results are shown in Figure 4.3. Every 100 iterations of the optimizer takes about 10.4 seconds on a GPU P100(kaggle platform) and total times takes about 2036.13 seconds. To improve the speed of the algorithm,





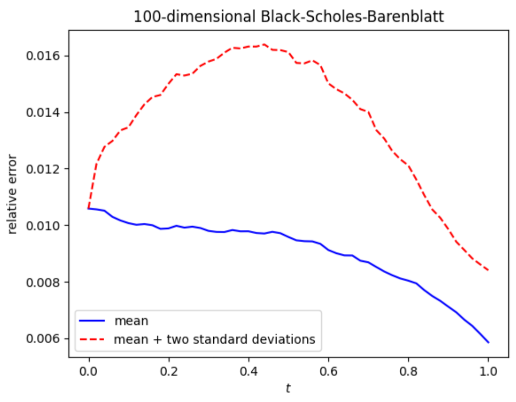


Figure 4.3: Results for Black-Scholes-Barenblatt equation in 100 dimensions. Upper: Loss function over mini-batch number. Lower-left: The error in Y_t for five paths. Lower-right: The average error for all paths.

we should investigate more advanced numerical discretization techniques for the system of FBSDEs as suggested by [2].Code can be found in the "Code" files: *100-dimensional Black-Scholes-Barenblatt equation.html.*

4.3 European Option Pricing for the Rough Bergomi Model

In the rough Bergomi model(see[7]) , the stochastic variant is given as

$$V_t = \xi_t \mathcal{E}(\eta \widehat{W}_t) \tag{4.4}$$

where ξ_t denotes the *forward variance curve* (a quantity which can be computed from the implied volatility surface). We assume that the forward variance curve ξ_t is flat for all $t \in [0,T] : \xi_t = \xi_0 > 0$. $\mathcal{E}$ denotes the *Wick exponential*, i.e., $\mathcal{E}(Z) := \exp\left(Z - \frac{1}{2}\operatorname{var} Z\right)$ for a zero-mean normal random variable Z, and $\eta \geq 0$. Where $\widehat{W}$ is a Volterra process with scaling property $\mathbf{Var}\left[\widehat{W}_t\right] = t^{2H}$. So far $\widehat{W}$ behave just like fBM.

$$\widehat{W}_t := \int_0^t \mathcal{K}(t-s)dW_s, \quad \mathcal{K}(r) := \sqrt{2H} r^{H-1/2}, \quad r > 0. \tag{4.5}$$

where $\mathcal{K}$ is the kernel function. Thus, the spot variance V_t is given by

$$V_t = \xi_0 \exp(\eta\sqrt{2\alpha+1}\int_0^t (t-s)^\alpha dW_s - \frac{\eta^2}{2} t^{2\alpha+1}), \quad \alpha = H - \frac{1}{2} \tag{4.6}$$

Hybrid simulation scheme: To simulate the Volterra-type integral $\tilde{X} = \sqrt{2\alpha+1}\int_0^t (t-s)^\alpha dB_s$, we apply the hybrid scheme proposed in [8], [9], which approximate the kernel function of the Brownian semi-stationary process by a Winner integral of the power function at $t = s$ and Riemann sum elsewhere. We consider a particular class of non-stationary processes, namely, truncated Brownian semi-stationary (tBss) processes,

$$\tilde{X}_t = \int_0^t g(t-s)\sigma_s dW_s \qquad t \geq 0 \tag{4.7}$$

where the kernel function $g(t)$, the volatility process σ_s, and the driving Brownian motion W_s are defined as in the definition of Bss processes. $\tilde{X}_t$ cam also be seen as the truncated stochastic integral at 0 of the Bss processes $\tilde{X}_t$.

According to [8], [9], the observation $\tilde{X}^N_{t_j}, j = 0, 1, \cdots, N$ can be computed via($k = 1 \quad case$).

$$\tilde{X}^N_{t_j} = L_g(\Delta t)\sigma^N_{j-1} W^N_{j-1,1} + \sum_{k=1}^{j} g(b^*_k \Delta t)\sigma^N_{j-k}\overline{W}^N_{j-k} \tag{4.8}$$

where N is the total number of time-steps, $\Delta t = T/N$ is the time-step size, and $t_0 = 0 \leq \cdots \leq t_j = j\Delta t \leq \cdots \leq t_N = T$ is a time grid on the internal $[0, T]$,

$$b_k^* = \left(\frac{k^{\alpha+1} - (k-1)^{\alpha+1}}{\alpha + 1}\right)^{\frac{1}{\alpha}} \tag{4.9}$$

As suggested by [8], we use

$$\begin{aligned} L_g &\equiv 1 \\ g(x) &\equiv x^{H-\frac{1}{2}} \\ \sigma(x) &\equiv \sqrt{2\alpha + 1} \end{aligned}$$

Then,

$$\begin{aligned} W_{j-1,1}^N &= \int_{t_{j-1}}^{t_j} (t_j - s)^\alpha dW_s \approx \left(\frac{\Delta t}{2}\right)^\alpha (W_{t_j} - W_{t_{j-1}}) \\ \overline{W}_j^N &= \int_{t_j}^{t_{j+1}} dW_s = W_{t_{j+1}} - W_{t_j} \\ \sigma_j^N &= \sigma_{t_j} \end{aligned}$$

The corresponding matrix representation takes the form of

$$\begin{pmatrix} \tilde{X}_{t_1} \\ \tilde{X}_{t_2} \\ \vdots \\ \tilde{X}_{t_N} \end{pmatrix} = \begin{pmatrix} \overline{W}_{0,1} & 0 & \cdots & \cdots & 0 \\ \overline{W}_{1,1} & \overline{W}_0 & \ddots & \ddots & 0 \\ \overline{W}_{2,1} & \overline{W}_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \overline{W}_{N-1,1} & \overline{W}_{N-2} & \cdots & \overline{W}_1 & \overline{W}_0 \end{pmatrix} \begin{pmatrix} \sigma_{t_1} \\ g(b_2^*\Delta t)\sigma_{t_2} \\ g(b_3^*\Delta t)\sigma_{t_3} \\ \vdots \\ g(b_N^*\Delta t)\sigma_{t_N} \end{pmatrix},$$

In the rBergomi model, σ_{t_j} is a constant for $i = 1, 2 \cdots, N$ defined as $\eta\sqrt{2\alpha + 1}$. Multiplying a lower triangular Toeplitz matrix can be regard as a discrete convolution which can be evaluated efficiently by fast Fourier transform. The variance simulations are shown below Figure 4.4.

According to section 2.2, the rough Bergomi model is linked to the following FBSDE,

$$\begin{aligned} -dY_s &= F_s(e^{X_s}, Y_s, Z_s, \tilde{Z}_s)ds - \tilde{Z}_s dW_s - Z_s dB_s, \quad 0 \leq s \leq T \\ Y_T &= G(e^{X_T}) \\ dX_s &= \sqrt{V_s}\left(\rho dW_s + \sqrt{1-\rho^2}dB_s\right) - \frac{V_s}{2}ds, \quad 0 \leq s \leq T \\ X_0 &= x \\ V_s &= \xi_s \mathcal{E}(\eta\widehat{W}_s), \quad s \in [0, T] \end{aligned} \tag{4.10}$$

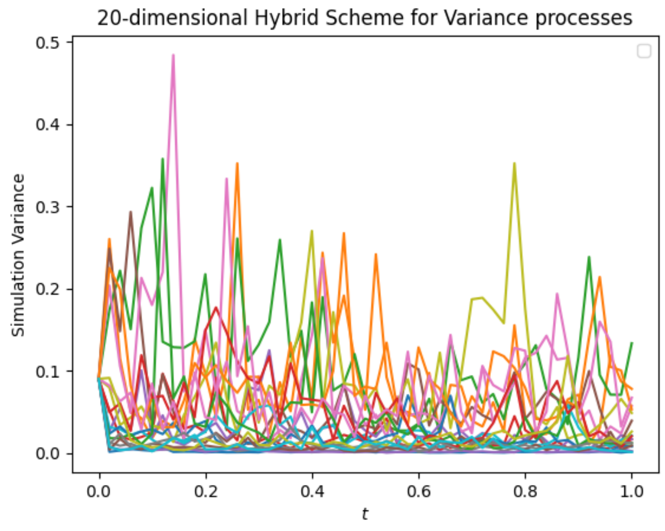


Figure 4.4: Hybrid simulation scheme for the variance processes(V_t) in 20 dimensions – one trajectory.

and has

$$\begin{aligned} u_\tau(X_\tau) &= Y_\tau \\ Z_\tau &= \sqrt{(1-\rho^2)V_\tau} Du_\tau(X_\tau) \\ \tilde{Z}_\tau &= \psi_\tau(X_\tau) + \rho\sqrt{V_\tau} Du_\tau(X_\tau) \end{aligned}$$

where the pair (u, ψ) is the unique weak solution to BSPDE (2.15), for $0 \le \tau \le T$.

4.3.1 The rough Bergomi Model in 1 dimension

We compute the numerical approximation to European put option for one dimension. The choice of parameters are:

$$H = 0.07, \eta = 1.9, \rho = -0.9, r = 0.05, T = 1.0, X_0 = ln(100), \xi(t) = \xi(0) \equiv 0.09$$

and

$$F_s(e^{X_s}, Y_s, Z_s, \tilde{Z}_s) = -rY, \quad G(e^x) = (K - e^{x+rT})^+$$

Then the NAIS-Net framework is used for the numerical approximation(see Figure 4.5). We take N=50 in the Euler Scheme, M=100 in mini-batch, and 6 layers, set a single

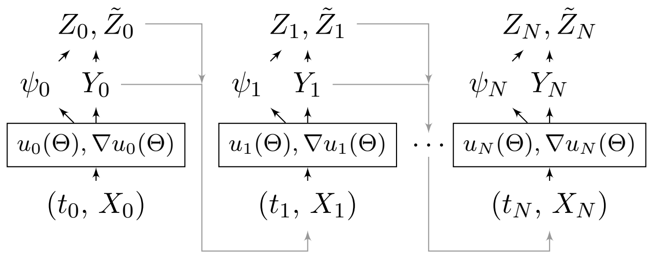


Figure 4.5: Approximation of a system of rBM in 1-dimension.

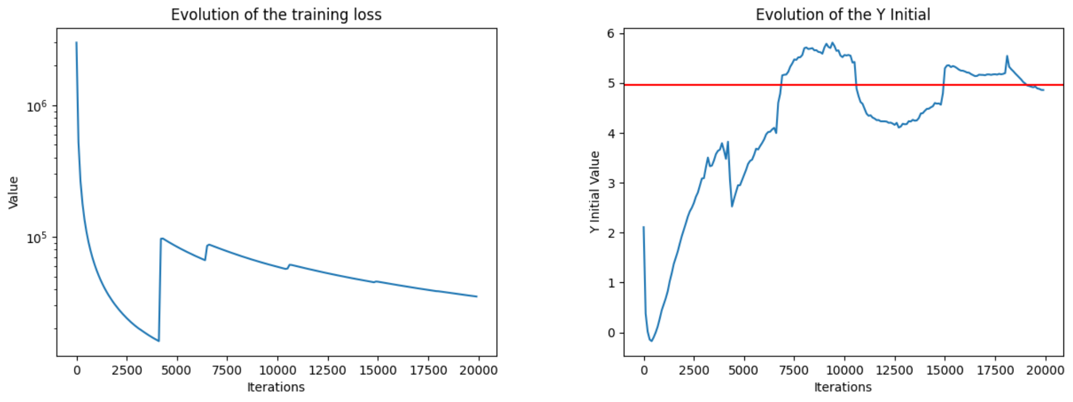
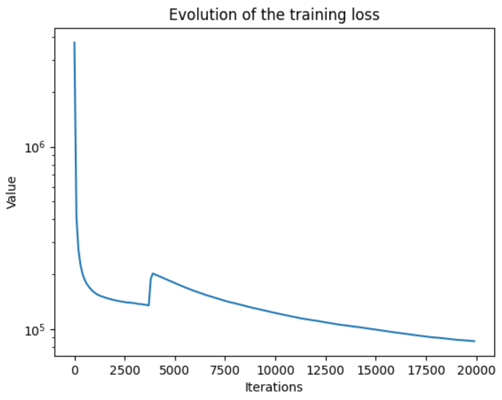


Figure 4.6: Results for the rough Bergomi model in 1 dimensions – having penalty term. Left: Loss function over mini-batch number. Right: convergence of price in Y_0.

hidden layer whose number of neurons is equal to 256. We adopt the Sine function for the activation function and the optimization algorithm is Adam. In the following table 4.2, 4.3, the reference value is defined as 4.9550(refer to [1]). The alternative reference value is calculated by Monte Carlo using 200000 trajectories. Setting the additional penalty term(see 3.13, $\beta_1 = \beta_2 = 0.02$) to loss function improve the accuracy. Figure 4.6 and 4.7 show the trains results. Code can be found in the "Code" files: The Loss function has penalty term, *nais-rbm-single-asset-penalty.html*; The Loss function has no penalty term, *nais-rbm-single-asset-no-penalty.html*.

Table 4.2: Price of European put option at t = 0, loss function has penalty term

Strike price	reference value	MC reference value	Estimated value	Standard deviation(estimated)	Relative error	MC Relative error
K = 90	4.9550	4.62747	4.9617	1.43051e-06	0.00670092	0.336957



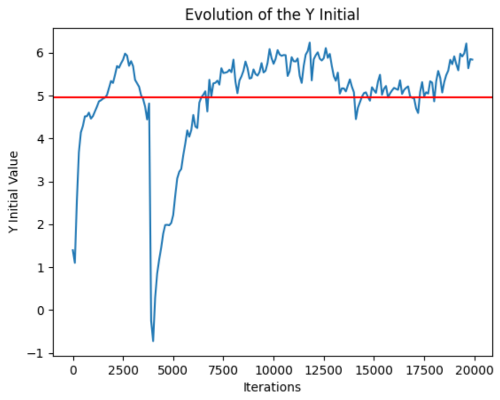


Figure 4.7: Results for the rough Bergomi model in 1 dimensions – no penalty term. Left: Loss function over mini-batch number. Right: convergence of price in Y_0.

Table 4.3: Price of European put option at t = 0, no penalty term

Strike price	reference value	Estimated value	Standard deviation(estimated)	Relative error
K = 90	4.9550	5.7160	2.8610e-06	0.7610

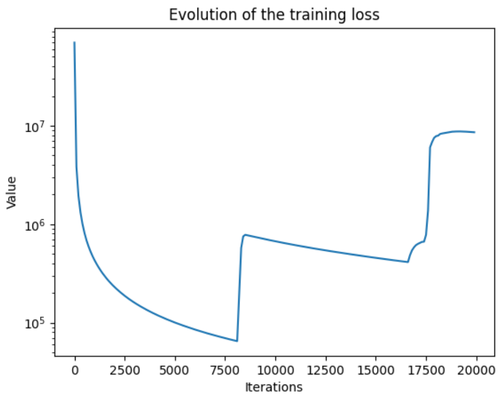
4.3.2 The rough Bergomi Model in 20 dimensions

We compute the numerical approximation to European put option for 20 dimensions, we assume that every assets are independent and the Brownian motions are all uncorrelated i.e. $cov(dW^i dW^j), cov(dB^i dB^j)$ for $i \neq j$. The choice of parameters are the same as above case, the rough Bergomi Model in one dimension except for $X_0 = (ln(100.0), ln(50.0), \cdots, ln(100.0), ln(50.0)) \in \mathbb{R}^{20}$. For the PDE solutions $\psi_t(X_t)$, we attempted four distinct approaches:

1. **Method 1**: In a NAIS-Net framework with a single output layer, the number of neurons determines the output. The total size of the output layer is [1 + Dimensions], where "Dimensions" neurons represents $\psi_t(X_t)$ and "1" neuron represents Y_t. Figure 4.8 shows the training results. Code files: *nais-rbm-20-dimension-methd1-penalty.html.*

2. **Method 2**: The NAIS-Net framework employs two independent output layers. See figure 4.9. Code files: *nais-rbm-20-dimension-methd2-penalty.html.*

3. **Method 3**: An additional linear layer is introduced to specifically represent $\psi_t(X_t)$. This layer is independent of the block that represents Y_t. Training results: Figure 4.10; Code files: *nais-rbm-20-dimension-methd3-penalty.html.*



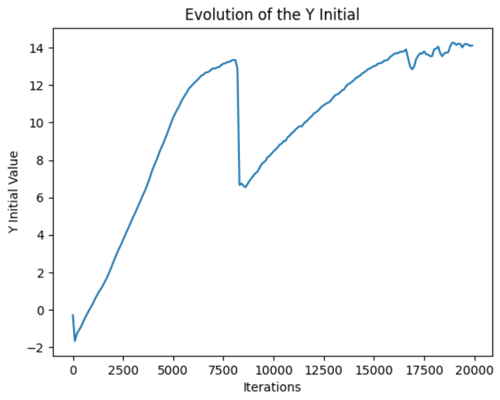


Figure 4.8: Results for the **Method 1** in 20 dimensions – having penalty term. Left: Loss function over mini-batch number. Right: convergence of price in Y_0.

4. **Method 4**: Based on above method 2, the loss function can be modified ($\beta_1 = \beta_2 = \beta_3 = 0.02$), training results: Figure 4.11; code files: *nais-rbm-20-dimension-methd4-penalty.html*.

$$L[u_\theta; x_0] = \frac{1}{M}\left[\sum_{\omega}\frac{1}{N}\sum_{n=1}^{N}\|Y_n - Y_n^\star\|^2 + \beta_1\|Y_N - g(X_N)\|^2 \right.$$
$$\left. + \beta_2\|Du_N - Dg(X_N)\|^2 + \beta_3\left\|\tilde{Z}_N - (\psi_N(X_N) + \rho\sqrt{V_N}Dg(X_N))\right\|^2\right] \tag{4.11}$$

Table 4.4: Price of European put option at t = 0.

Distinct Approaches	reference value	Estimated value(mean)	Standard deviation(estimated)	Relative error
Method 1-penalty	13.9343	14.2044	1.6212e-05	0.2701
Method 2-penalty	13.9343	14.6078	1.3351e-05	0.6734
Method 3-penalty	13.9343	10.5977	1.1444e-05	-3.3366
Method 4-penalty	13.9343	14.3464	1.5258e-05	0.4120

In the table 4.4, reference value is calculated by Monte Carlo using 10000 trajectories. Comparing the aforementioned methods, all four exist significant errors. While Method 1 intuitively yields the smallest error, it also exhibits the greatest fluctuation, as indicated by its largest standard deviation. In contrast, Method 4 generally performs better, as it provides both accuracy and numerical stability. Method 3 proves to be the least reliable, as it relies on a single linear layer to simulate the ψ_t results, which fails to ensure numerical

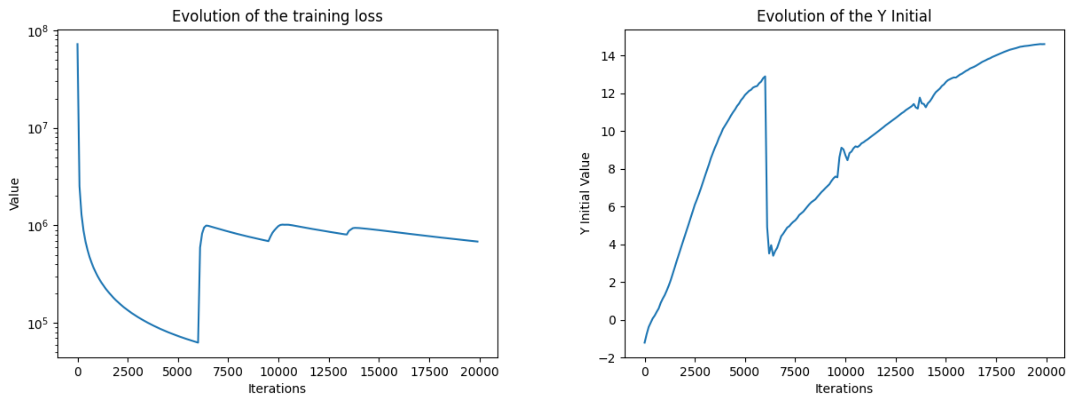


Figure 4.9: Results for the **Method 2** in 20 dimensions – having penalty term. Left: Loss function over mini-batch number. Right: convergence of price in Y_0.

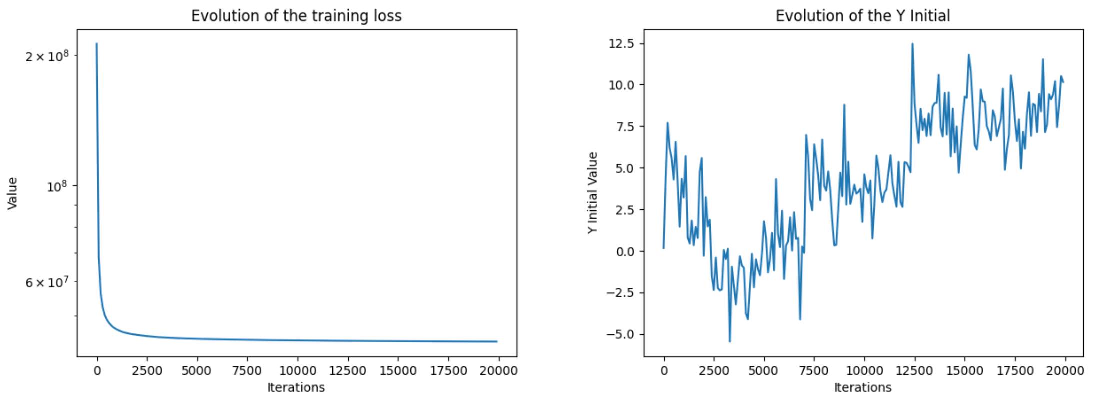


Figure 4.10: Results for the **Method 3** in 20 dimensions – having penalty term. Left: Loss function over mini-batch number. Right: convergence of price in Y_0.

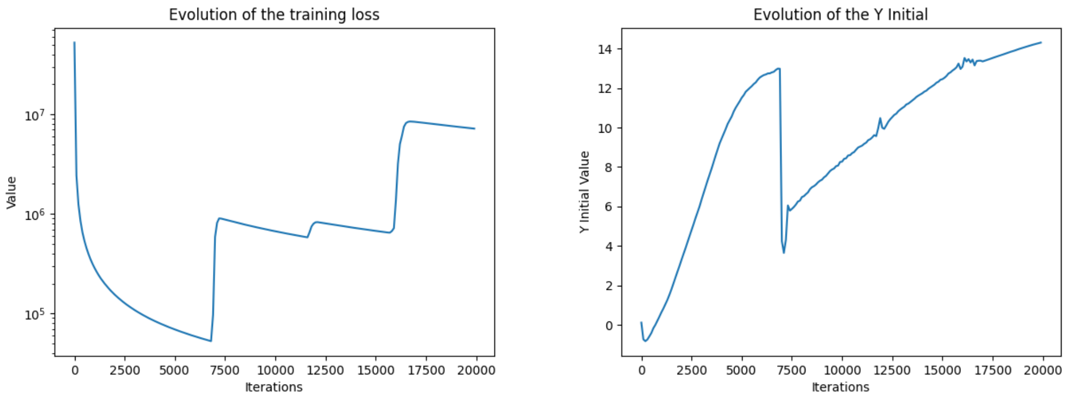


Figure 4.11: Results for the **Method 4** in 20 dimensions – having penalty term. Left: Loss function over mini-batch number. Right: convergence of price in Y_0.

stability, making the output vulnerable to minor perturbations. When examining the loss function on a logarithmic scale, Methods 1, 2, and 4 display upward discontinuities, likely influenced by inherent noise that adversely affects training outcomes, with the ψ_t solution potentially being a major contributing factor. Comparing Table 4.2, the numerical values obtained using the Monte Carlo method by ourselves are consistently biased low, suggesting that despite the errors observed in Method 4, the model's reliability remains intact, making it suitable for extended application. Based on these findings, further investigation into the configuration of the ψ_t solution is warranted.

5 Conclusions

We presented fundamental concepts of Forward-Backward Stochastic Differential Equations (FBSDEs) and constructed a network architecture based on methods from the [2], applying it to the numerical approximation of high-dimensional PDEs (Partial Differential Equations) to address high-dimensional option pricing problems. Through project practice, it has been shown that deep learning-based numerical methods can effectively solve the numerical simulation of high-dimensional PDEs, especially under nonlinear conditions. Moreover, the NAIS-Net architecture can obtain the solution for $Y_t = u(t, X_t)$ at a later time $t > 0$, reducing the need for additional training. However, the training process for deep learning models necessitates substantial data and resource consumption, significantly impacting time efficiency. Notably, the NAIS-Net architecture exhibited lower time efficiency compared to other network architectures (see [2]). Throughout the training process, ensuring numerical stability was critical to avoid significant deviations caused by minor perturbations, and the

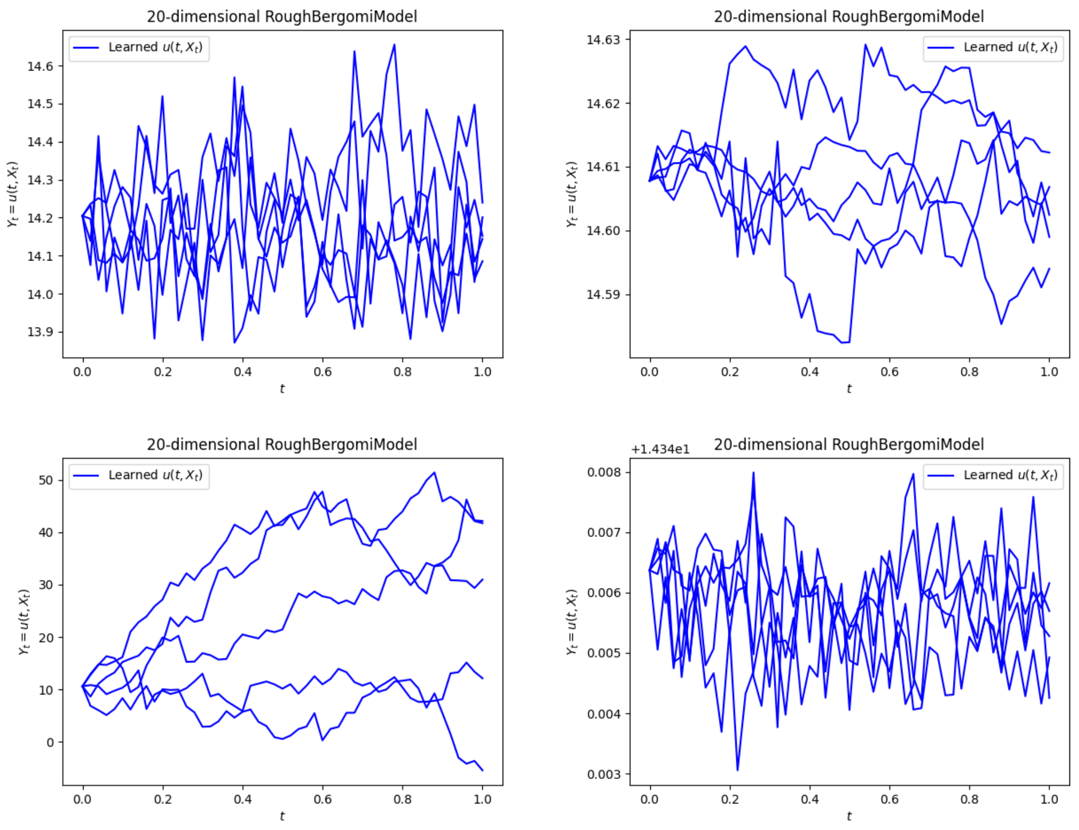


Figure 4.12: The solutions for Y_t in 20-dimensional rough Bergomi model (rBM) across five different paths, as obtained from the four approaches. Upper-left: Method 1; Upper-right: Method 2; Lower-left: Method 3; Lower-right: Method 4.

NAIS-Net architecture provided a degree of robustness in this regard.

In the validation of the rough Bergomi model (rBM), the hybrid scheme method was used to simulate stochastic volatility, which further increased the model's complexity, thereby reducing the efficiency of model training. Improving the efficiency of model training is a subject for further research. The stochastic volatility was simulated numerically, and deep learning methods provided in the [10] can be further adopted to expand the model's applicability. Overall, the application of deep learning-based numerical methods has demonstrated potential in mitigating the challenges associated with solving multi-dimensional option pricing problems.

References

[1] Yao, Y. (2021). Deep Learning-based Numerical Methods for Stochastic Partial Differential Equations and Applications (Master's thesis, University of Calgary, Calgary, Canada). Retrieved from `https://prism.ucalgary.ca`.

[2] Güler, B., Laignelet, A., Parpas, P. (2019). Towards Robust and Stable Deep Learning Algorithms for Forward Backward Stochastic Differential Equations. ArXiv, abs/1910.11623.

[3] Raissi, M. (2018). Forward-Backward Stochastic Neural Networks: Deep Learning of High-dimensional Partial Differential Equations. ArXiv, abs/1804.07010.

[4] Ciccone, M., Gallieri, M., Masci, J., Osendorfer, C., Gomez, F.J. (2018). NAIS-Net: Stable Deep Networks from Non-Autonomous Differential Equations. Neural Information Processing Systems.

[5] Wenzhong Zhang and Wei Cai. 2022. FBSDE based neural network algorithms for high-dimensional quasilinear parabolic PDEs. J. Comput. Phys. 470, C (Dec 2022). `https://doi.org/10.1016/j.jcp.2022.111557`.

[6] Hientzsch, B. (2019). Introduction to Solving Quant Finance Problems with Time-Stepped FBSDE and Deep Learning. DecisionSciRN: Other Investment Decision-Making (Sub-Topic).

[7] Christian Bayer, Peter Friz, and Jim Gatheral. Pricing under rough volatility. Quantitative Finance, 16(6):887–904, 2016.

[8] Zhu Q, Loeper G, Chen W, Langrené N. Markovian Approximation of the Rough Bergomi Model for Monte Carlo Option Pricing. Mathematics. 2021; 9(5):528. `https://doi.org/10.3390/math9050528`.

[9] Bennedsen, M., Lunde, A. & Pakkanen, M.S. Hybrid scheme for Brownian semistationary processes. Finance Stoch 21, 931–965 (2017). `https://doi.org/10.1007/s00780-017-0335-5`.

[10] Changqing Teng and Guanglian Li, 2024. Neural option pricing for rough Bergomi model, Papers 2402.02714, arXiv.org. `https://ideas.repec.org/p/arx/papers/2402.02714.html`.

[11] Jacquier, A., Martini, C., & Muguruza, A. (2017). On VIX futures in the rough Bergomi model. Quantitative Finance, 18, 45 - 61.

[12] Guyon, J., & Henry-Labordere, P. (2013). Nonlinear Option Pricing (1st ed.). Chapman and Hall/CRC. `https://doi.org/10.1201/b16332`.

[13] Ma, Jin & Yong, Jiongmin. (2007). Forward-Backward Stochastic Differential Equations and Their Applications. `10.1007/bfb0092524`.

[14] Yong, J., & Zhou, X.Y. (1999). Stochastic Controls: Hamiltonian Systems and HJB Equations.

[15] Han, J., Jentzen, A., & E, W. (2017). Solving high-dimensional partial differential equations using deep learning. Proceedings of the National Academy of Sciences, 115, 8505 - 8510.

[16] Yu Y, Ganesan N, Hientzsch B. Backward Deep BSDE Methods and Applications to Nonlinear Problems. Risks. 2023; 11(3):61. `https://doi.org/10.3390/risks11030061`.

[17] Sun, H., & Bao, F. (2024). Solving high dimensional FBSDE with deep signature techniques with application to nonlinear options pricing.

[18] Liang, J., Xu, Z., & Li, P. (2021). Deep learning-based least squares forward-backward stochastic differential equation solver for high-dimensional derivative pricing. Quantitative Finance, 21(8), 1309–1323. `https://doi.org/10.1080/14697688.2021.1881149`.

[19] Zimeng,Luo, Fractional Brownian Motion and Application to Option Pricing. CQF Electives, Advanced Volatility modeling.

[20] CQF Course M5 L8 - Practical Machine Learning Case Studies for Finance.

[21] Toeplitz and Circulant Matrices: A review.`https://ee.stanford.edu/~gray/toeplitz.pdf`