

Research on UQ Problem Based on Feynman–Kac Formula and Neural Network

Yihang Zhang [†], Jiaxi Wang [†], Jie Zhao ^{*,†}

Faculty of Science, China University of Mining & Technology (Beijing), Beijing, China, 100083

* Corresponding Author Email: 18273525265@163.com

[†] These authors contributed equally.

Abstract. A high-dimensional adaptation method of neural network regression to obtain the expected mean solution of the uncertainty quantification problem based on Monte Carlo simulation and the Feynman-Kac formula is presented. In the numerical tests, by the Feynman-Kac formula, the numerical solution on each given point is obtained by rectangle method. Polynomial and neural network regression of different dimensions lead to obtain the expected mean solution of the UQ problem. This thesis illustrates that neural network are more conducive to high-dimensional problems. This thesis analyzes the effects of time step, simulation frequency, generations, hidden layers, and learning rate on the convergence by drawing convergence graphs.

Keywords: UQ problem, Feynman–Kac formula, polynomial regression, neural network, Brownian motion.

1. Introduction

Uncertainty quantification problem (UQ problem) refers to the problem where there are uncertain factors in the decision-making process that require quantitative analysis. Uncertainty refers to the inability to determine the specific outcome of the development or the attributes of parameters. In practical decision-making, uncertainty generally exists in various fields, such as finance, hydrodynamics, healthcare, etc.

The development of probability theory in the early 20th century laid the foundation for UQ problem, providing mathematical tools to describe and reason about uncertainty. Methods used in UQ problem include probability theory, statistics, fuzzy mathematics and information theory. These methods help to analyze the sources, properties and impacts of uncertainty, providing decision support and management tools.

With the deepening of research on UQ problem, calculation methods of random model have been applied, including:

(1) Monte Carlo method: Monte Carlo (MC) method is a numerical solution method based on random sampling. We use the probability distribution or to generate samples randomly. From this, uncertain problems such as stochastic PDEs are transformed into certain problems such as deterministic PDEs. We obtain statistical information by solving the deterministic problems. Since the mean convergence rate of the MC method is $\frac{1}{\sqrt{N}}$ (N is the numbers of the samples) a large number of samples are usually needed.

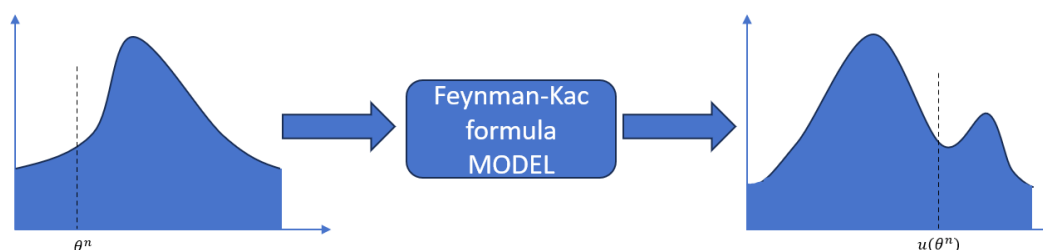


Figure 1. Randomly generated sample calculation model

(2) Polynomial Chaos method [1]: Uses orthogonal polynomials to approximate the random distribution of solutions and represents uncertainty quantification results through series expansion. It is accurate and computationally efficient for solving equations or systems with random parameters.

(3) HARS method: Based on response surface modeling, it establishes a functional relationship between input parameters and output response to describe the impact of parameter uncertainty. It quickly generates and analyzes a large number of samples based on the established response surface model.

(4) Sampling based methods: Generate samples by sampling from the probability distribution of input parameters for solving and statistical analysis. Common methods include Latin Hypercube Sampling and Monte Carlo Sampling. It is easy to implement but may require a large number of samples for accurate uncertainty quantification in high-dimensional problems.

(5) Response Surface Methods: Use a mathematical model to fit the relationship between input parameters and output response for uncertainty quantification approximation. A common method is polynomial regression.

Selecting appropriate uncertainty quantification methods based on the characteristics and requirements of specific problems can improve the computational efficiency and accuracy of UQ problems through different methods and techniques.

However, in high-dimensional problems, the traditional PDEs method for solving UQ problems may encounter some challenges. In traditional PDEs based methods such as finite element method and finite difference method, the dimension of the problem will lead to an exponential increase in computational complexity. Therefore, directly applying these methods to high-dimensional problems may become very difficult or even infeasible. In order to overcome the challenges of high-dimensional problems, this thesis turns to the Feynman–Kac formula and neural network regression method. Thus, the UQ problem has transformed into a probabilistic statistical problem, which helps to solve high-dimensional problems. Secondly, for the N sampled samples, the numerical solution of the UQ problem is first calculated by using the Feynman–Kac formula, and then the expected mean of the UQ problem solution, which is the mean absolute error, is obtained through polynomial regression. However, the results show that as the dimension increases, the average absolute error obtained by polynomial regression model becomes larger and does not achieve the desired effect. Therefore, this thesis turns to use neural network to replace polynomial regression method and the neural network reaches relatively ideal errors in high-dimensional situations. It effectively solves the UQ problem. Finally, the thesis conducts a convergence analysis on a 5-dimensional neural network model, analyzing the impact of five indicators including time step, number of simulations, generation, hidden layer, and learning rate on the convergence of the model. By drawing convergence graphs, it turns out that shorter time steps, more simulations, suitable generations, hidden layers, and learning rate have a positive impact on the convergence of the model.

2. Feynman–Kac formula

2.1. Feynman–Kac formula of the Laplace equation problem

The Feynman–Kac formula is a renowned tool that enables the random representation of pointwise solutions to many partial differential equations, such as diffusion equations or transport equations [2].

This article mainly considers a commonly used mathematical model Laplace equation in the UQ problem, which is also a special type of stochastic elliptic problem. When the Dirichlet boundary value problem in a domain $D(\subset R^n)$ with an amply smooth boundary ∂D :

$$-\frac{1}{2}\Delta u = f \quad (1)$$

And boundary conditions

$$u = g \quad (2)$$

On ∂D , we have $\forall x \in D$:

$$u(x) = E_x[g(X_{\tau_D}) + \int_0^{\tau_D} f(X_s)ds]. \quad (3)$$

In the above equation, $X_t(t \geq 0)$ represents a stochastic process and τ_D is the post-set time of the stochastic process from the domain D . Additionally, the error comes from the use of N simulated standard Monte Carlo errors, i.e. $\frac{\sigma}{\sqrt{N}}$, where

$$\sigma = \sqrt{\text{Var}(g(X_{\tau_D}^x) + \int_0^{\tau_D} f(X_s^x)ds)}. \quad (4)$$

In the case of Laplace equation, the simulation process is Brownian motion, and use the Euler scheme [9] to calculate $\int_0^{\tau_D} f(X_s^x)ds$.

2.2. The simulation of the Brownian motion

Starting from $x \in D(\subset R^n)$, simulate the Euler format of Brownian motion with parameter Δt that shows

$$B_0 = x, B_{k+1} = B_k + Y_k \sqrt{\Delta t}. \quad (5)$$

In the above equation, Y_k represents an n -dimensional vector, where each component follows a standard normal distribution.

If the motion stops between steps k and $k+1$, that is to say, $B_k \in D$ and $B_{k+1} \in D^c$. Then we use projections based on B_k and B_{k+1} to approximate B_{τ_D} , thus approximating $g(B_{\tau_D})$.

The standard approximation of the rectangular method for $\int_0^{\tau_D} f(X_s^x)ds$ is given by

$$A_1 = \Delta t \sum_{i=1}^k E(f(B_{i\Delta t})). \quad (6)$$

We use $A_2 = k\Delta t f(B_{R\Delta t})$ where R is a discrete uniform random variable on the set $[1, 2, \dots, k]$ to replace the approximation and we apparently have

$$E(A_2) = \Delta t \sum_{i=1}^k E(f(B_{i\Delta t})). \quad (7)$$

3. Model construction

3.1. Construction of the seventh order polynomial regression model

In this part, we introduce data sampling method. Use the Feynman–Kac formula to solve UQ problem after sampling. In the expected estimation of the solution of the UQ problem, the function estimation is obtained by using the seventh order polynomial regression.

For the convenience of narration, we give the notation (8)

$$x = (x_1, x_2, \dots, x_n). \quad (8)$$

The full expectation in the solution of the stochastic PDEs at a fixed point x may simply be solved by considering a probabilistic representation

$$v(x) = E[\theta^x], x \in D \subset R^n. \quad (9)$$

In (9), for each fixed x , θ^x is a real-valued random variable on a probability space. In this paper, we consider

$$v(x) = E[u(x)], \quad x \in D \subset R^n, \quad (10)$$

Where $u(x)$ is the solution of the random PDEs that we are concerned about. By (3), the above relation holds for

$$\theta^x = g(X_{\tau_D}^x) + \int_0^{\tau_D} f(X_s^x) ds. \quad (11)$$

We estimate $v(x)$ by Monte Carlo simulation, i.e. in notation

$$\begin{aligned} &\theta_1^{\alpha_1}, \dots, \theta_M^{\alpha_1}, \\ &\theta_{M+1}^{\alpha_2}, \dots, \theta_{2M}^{\alpha_2}, \\ &\dots \\ &\theta_{(N-1)M+1}^{\alpha_N}, \dots, \theta_{NM}^{\alpha_N}. \end{aligned} \quad (12)$$

In (12), $(\alpha^n, n=1,2,\dots)$ is a series of i.i.d. random variables satisfying, α^n holding a probability measure on $(D, B(D))$ are copies from the same random variable on D . Below, we introduce a concise recap of the averages for the convenience of narration

$$\bar{\theta}_n^x = \bar{\theta}_{n,M}^x = \frac{1}{M} \sum_{m=1}^M \theta_{(n-1)M+m}^x, \quad x \in D. \quad (13)$$

Below, we carry out polynomial regression by least squares method. Take a system of functions $\psi_k : D \rightarrow R, k=1,2,\dots$, and consider for an MC sample (12) and a fixed number K the solution of the regression problem

$$\hat{\gamma} = \arg \min_{\gamma \in R^K} \frac{1}{N} \sum_{n=1}^N (\bar{\theta}_n^{\alpha_n} - \sum_{k=1}^K \gamma_k \psi_k(\alpha_n))^2. \quad (14)$$

The solution of (14) is given by

$$\hat{\gamma} = (\Lambda^T \Lambda)^{-1} \Lambda^T Y, \quad (15)$$

Where $\Lambda \in R^{N \times K}$ with entries given by

$$\Lambda_{n,k} = \psi_k(\alpha_n), n=1,\dots,N, k=1,\dots,K \quad (16)$$

And

$$Y = [\bar{\theta}_{1,M}^{\alpha_1}, \dots, \bar{\theta}_{N,M}^{\alpha_N}]^T. \quad (17)$$

It is estimated that

$$\hat{v}(x) = \sum_{k=1}^K \hat{\gamma}_k \psi_k(x), \quad (18)$$

Which depends on basis functions $(\psi_k, k=1,\dots,K)$ and the specific number of M, N, K .

The following are steps of the construction [4]:

(1) In n -dimensional space, generate N sample point $(x_1, x_2, \dots, x_n)$ and $\forall x_i (i=1,2,\dots,n)$, x_i follows a uniform distribution of $[0, 1]$. Since we are considering the UQ problems, the simulation of random variables $z = (z_1, \dots, z_n)$ are given in step (2).

(2) At each sample point, simulate the implementation value z of the random variable space $z = (z_1, z_2, \dots, z_{n+1})$. z_1 follows a uniform distribution of $[-1, 1]$ while $z_2, z_3, \dots, z_{n+1}$ follow a uniform distribution of $[0, 1]$. In this procedure we then obtain N (x, z) combinations.

(3) For all the combination (x, z) , since each z has already been simulated and determined (has a numerical value), the original UQ problem is transformed into the deterministic PDEs problem.

Below we firstly give the polynomial regression error estimation and the reason why we let $M = 1$.

According to the literature [5], we have

$$\frac{1}{N}[\Lambda^T \Lambda]_{kl} = \frac{1}{N} \sum_{n=1}^N \psi_k(\alpha_n) \psi_l(\alpha_n) \xrightarrow{N \rightarrow \infty} \int_D \psi_k(x) \psi_l(x) \mu(dx) = G_{kl}. \quad (19)$$

Let $\bar{\lambda}_{\min}^K > 0$ be the smallest eigenvalue of G .

Suppose that the set of basis functions is such that the variances are bounded by $\tau > 0$ for all k if $K \rightarrow \infty$ and that

$$\bar{\lambda}_{\min}^K \geq \bar{\lambda}_{\min} > 0, K \rightarrow \infty. \quad (20)$$

We then have the error estimate:

$$\|\varepsilon\|^2 \leq \frac{\tau \cdot K}{\bar{\lambda}_{\min} N} \left(1 + \frac{2}{\sqrt{M}} + \frac{1}{M}\right), \quad (21)$$

Where ε denote the regression error. Hence, we come to a conclusion:

In the case of the same calculation, the smaller the number of times of simulation at each point, the smaller the error.

Therefore, in this paper, we let $M = 1$. In the program, we thus simulate the path (the Brownian motion trajectory) only once and we use the Feynman–Kac formula to estimate the deterministic PDEs. In this procedure, we obtain numerical solution $u(x)$. For N x and corresponding estimated solutions $u(x)$, use Polynomial Features in Python to perform 7th degree polynomial transformation on input features. Additionally, perform polynomial transformation on $x_1, x_2, \dots, x_n$ to obtain $x_{\text{transformed}}$.

In the process of polynomial regression, the increase of dimension has a certain rule. For example, if there are only two features x_1 and x_2 in the original feature matrix, and after conversion, we get:

$$\begin{aligned} x_1, x_2 &\Rightarrow 1, \\ &x_1, x_2, \\ &x_1^2, x_2^2, x_1 x_2, \\ &x_1^3, x_2^3, x_1^2 x_2, x_1 x_2^2, \\ &x_1^4, x_2^4, x_1^3 x_2, x_1^2 x_2^2, x_1 x_2^3, \\ &\dots \\ &x_1^7, x_2^7, x_1^6 x_2, x_1^5 x_2^2, \dots, x_1 x_2^6. \end{aligned} \quad (22)$$

Similarly, if there are n features, the polynomial will produce all terms from the number of 1 to the number of n .

In terms of data, we create a data box df in form of a dictionary, which contains data with column names $x_1, x_2, \dots, x_n$ and u .

(4) Generate a new uniformly distributed random variable $x_{11}, x_{21}, \dots, x_{n1}$, which will be used as input data for the model. Use the polynomial model to predict the new input data $x_{11}, x_{21}, \dots, x_{n1}$ and

obtain the prediction result u_1 . In terms of data, we also create another data box df_1 in form of a dictionary, which contains data with column names $x_{11}, x_{21}, \dots, x_{n1}$ and u_1 .

(5) Through a cycle, calculate the absolute value of each predicted value (each element in u_1) and take the average to obtain the average absolute error of each point. However, in this process, we discover that as the dimension increases, the average absolute error obtained by polynomial regression model becomes larger and doesn't achieve the desired effect. This article considers using neural networks to solve this problem.

3.2. Construction of neural network model

Multi-layer perceptron (MLP) is a common neural network model [6] that gives a function $f(\cdot): R^m \rightarrow R^o$ through training on a data set, where m is the input dimension and o is the output dimension. If a set of features $X = X_1, X_2, \dots, X_m$ and a target y is given, a non-linear function approximator is learned for regression.

A hidden layer of MLP with scalar output is showed in Figure 2. The input layer is composed of a set of neurons $\{X_i | X_1, X_2, \dots, X_m\}$ which represent the input features. Every neuron in the hidden layer converts the value of the previous layer through weighted linear summation $W_1X_1 + W_2X_2 + \dots + W_mX_m$.

The output layer gets the values from the last hidden layer and converts them into output values.

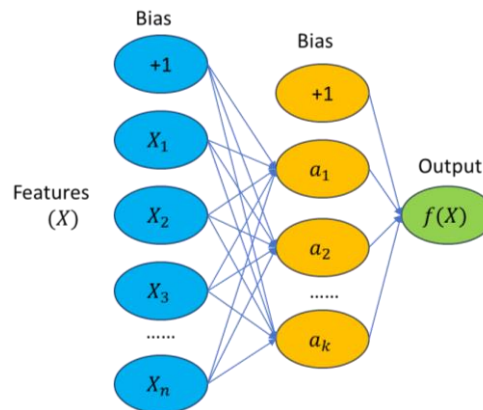


Figure 2. One hidden layer of MLP

The construction of neural network model is similar to that of polynomial regression model. We just need to change (4) in 3.1 to the specific content of the neural network and the following is the change.

Due to the multi-layer perceptron (MLP) regression model being a forward artificial neural network with one or more hidden layers, we selected a hidden layer containing 10 hidden neurons and trained it using Adam optimizer and the RELU activation function (23).

$$f_{RELU}(x) = \max(0, x). \quad (23)$$

By maximizing the number of iterations and setting the random seed to N , we ensured the stability and reproducibility of the model.

We discover that as the dimension increases, the average absolute error obtained by neural network model keeps small and achieve the desired effect, indicating that the neural network model fits well.

4. Numerical experiment

4.1. Model problem

Through utilizing the Feynman–Kac formula introduced in the second part, we will find that by using different g s, we can obtain different u s and f s through the Feynman–Kac formula on ∂D .

In the case where the domain D is n -dimensional, in the model construction section, we have already explained how we can obtain $x_1, x_2, \dots, x_n$. Then we select a special g of the Laplace equation for numerical simulation, and it is written in the equation below.

$$g(x_1, x_2, \dots, x_n) = z_1 e^{\frac{1}{5} \sum_{i=1}^n z_{i+1} x_i} \quad (24)$$

In the above equation, z_1 follows a uniform distribution of $[-1, 1]$, while $z_2, z_3, \dots, z_{n+1}$ follow a uniform distribution of $[0, 1]$.

According to boundary conditions, we can calculate $u(x)$ which is showed as follows:

$$u(x_1, x_2, \dots, x_n) = g(x_1, x_2, \dots, x_n) = z_1 e^{\frac{1}{5} \sum_{i=1}^n z_{i+1} x_i} \quad (25)$$

Additionally, f can be calculated:

$$f(x_1, x_2, \dots, x_n) = -\frac{1}{2} \Delta u = -\frac{1}{50} z_1 \cdot \sum_{i=2}^{n+1} z_i^2 \cdot e^{\frac{1}{5} \sum_{i=1}^n z_{i+1} x_i} \quad (26)$$

For a fixed (X, Z) , where $X = (x_1, x_2, \dots, x_n)$, $Z = (z_1, z_2, \dots, z_{n+1})$, let $X_t (t \geq 0)$ be the following stochastic differential equation:

$$dX_t = dB_t, X_0 = X. \quad (27)$$

Next, we will substitute the obtained $u(x)$, f and g into Feynman–Kac formula

$$u(x_1, x_2, \dots, x_n) = E_{x_1, x_2, \dots, x_n} [g(X_{\tau_D}, Z) + \int_0^{\tau_D} f(X_s, Z) ds] \quad (28)$$

For simulation. We perform polynomial regression and neural network model on the simulated numerical solution u , and analyze the data results in detail in section 4.2.

4.2. Display of calculation results

In the model construction section, we construct the seventh order polynomial regression model and the neural network model. Through these two models, We can solve the UQ problem and get the average absolute error respectively. Below, we show the average absolute error calculated by two methods for models with different dimensions.

Table 1. Model parameters of polynomial regression

parameters	2-dimensional	5-dimensional	10-dimensional
random coefficients number	3	6	11
sampling points' number	10000	10000	10000
initial time	0	0	0
time step	0.001	0.001	0.001
polynomial degree	7	7	7
intercept	0	0	0

We generate 10000 points through a uniform distribution from 0 to 1 in each numerical test. The intercept means the independent terms in the linear model (the linear model is a one-time intercept).

The initial time is the initial time point of the simulated Brownian motion. The time step parameter gives the trajectory point of the simulated Brownian motion and the length of each segment in the integral interval of the rectangular method.

Table 2. Model parameters of neural network

parameters	2-dimensional	5-dimensional	10-dimensional
random coefficients number	3	6	11
sampling points' number	10000	10000	10000
initial time	0	0	0
time step	0.001	0.001	0.001
hidden layers' number	2	1	1
neurons' number	(100,100)	10	10
solver	Adam	Adam	Adam
maximum-iterations	not set	1000	1000

For each hidden layer, calculate the activation value of each neuron. The activation value is obtained by weighted sum of the output on the previous layer, the weights connected to the neuron and non-linear transformation through the activation function. This calculation process is performed in each neuron in each hidden layer. Adam is a gradient based optimization method used in the process of neural network optimization. The maximum-iterations determine the maximum number of times the sample is trained. Due to the calculation amount of 2-dimensional problems is small, we have not set the maximum number of iterations.

Table 3. Average absolute error

	polynomial regression	neural network
2-dimensional	0.0293	0.0068
5-dimensional	0.1644	0.0263
10-dimensional	0.5346	0.0316

Through the data in Table 3, we found that the absolute error of the neural network algorithm is much smaller than that of the polynomial regression fitting algorithm, especially when the dimension increases. And the use of polynomial regression methods will produce relatively large absolute errors, while the average absolute error of the neural network algorithm is still very small.

Based on the above results, we will draw pictures to further explore differences in fitting performance between two algorithms. For each dimension, we use $\frac{1}{n} \sum_{i=1}^n x_i$ as the abscissa and predicted and true solutions in polynomial regression and neural network algorithms as the ordinate to draw plane diagrams for different $\frac{1}{n} \sum_{i=1}^n x_i$ s.

In order to make the conclusion more convincing, we selected x_i uniformly and unevenly, respectively. Firstly, we uniformly select x_i , which means taking a point every 0.01 on the [0, 1] interval as the value of x_i and draw Figure3.

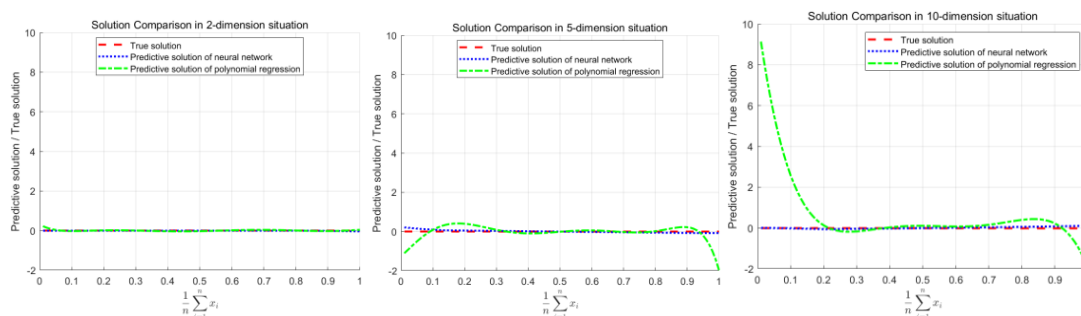


Figure 3. Solution Comparison in different dimensions (x_i selected uniformly)

Additionally, we select x_i unevenly. We use a column of arithmetic sequences to select $x_1, x_2, \dots, x_n$, where the tolerance of the arithmetic sequence is

$$d = \frac{1}{450(n-1)}, \quad (29)$$

And the value of $x_1, x_2, \dots, x_n$ is shown as follows:

$$x_1 = \frac{2}{225}t, x_2 = (\frac{2}{225} + d)t, \dots, x_n = \frac{1}{90}t, \quad (30)$$

Where t is a positive integer from 1 to 90.

In (30), the mean of $x_1, x_2, \dots, x_n$ is $0.01t$, so we can make $\frac{1}{n} \sum_{i=1}^n x_i$ from 0 to 0.9 and draw

Figure 4.

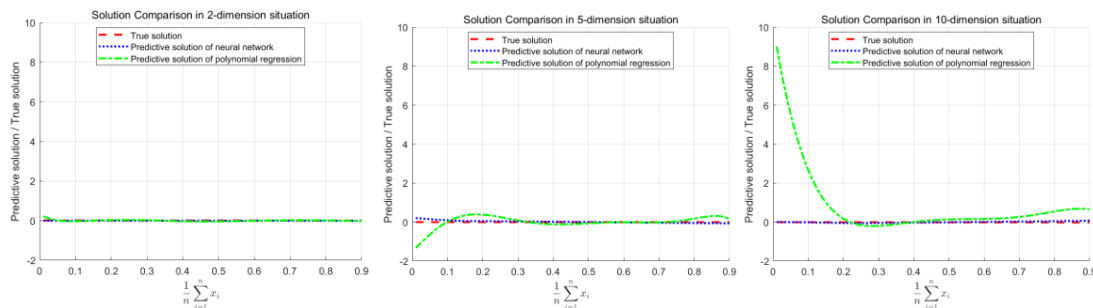


Figure 4. Solution Comparison in different dimensions (x_i selected unevenly)

In the two-dimensional case, the deviation between the predicted and true solutions of polynomial regression algorithm and neural network algorithm is not significant, indicating that both algorithms fit well in low dimensional situations. But as the dimensions continue to rise, by the 5th dimension, the predicted solution of polynomial regression has already deviated from the true solution, and by the 10th dimension, the predicted solution of polynomial regression has far deviated from the true solution, leading to the curse of dimensionality. In contrast, The estimated solution of the neural network algorithm is almost identical to the actual solution in all dimensions, which is sufficient to prove the feasibility and accuracy of the neural network algorithm.

4.3. Convergence analysis

In this section, we draw convergence graphs and perform convergence analysis on the neural network model we have established. In the error analysis of convergence graphs, errors can come from multiple aspects, and the following are some possible sources of error, that is, time step, number of simulations, generations, the number of hidden layers and learning rate. And generations [7], the number of hidden layers [8], and learning rate [9] are from neural networks itself.

Next, we will use the 5-dimensional neural network model to perform convergence analysis on the above five indicators.

Firstly, regarding the time step size, we select 12 different time steps and calculated 12 different average absolute errors through our established 5- dimensional neural network model, which are shown in Figure 5. The time step defines the time interval between each step in the simulation or optimization process. It is proved in Figure 5 that a smaller time step may increase the accuracy of the simulation and lead to a smoother convergence curve. A larger time step may result in the model being unable to accurately capture system dynamics in certain situations.

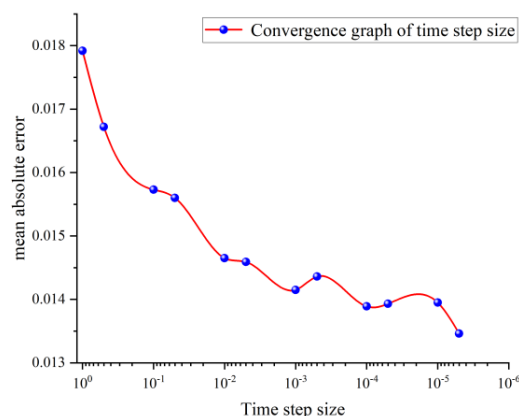


Figure 5. Convergence graph of time step size

Secondly, concerning the number of simulations, we select 12 different numbers of simulations and calculated 12 different average absolute errors through our established 5-dimensional neural network model, which are shown in Figure 6. The number of simulations represents the total number of times the model has been run or the optimization algorithm has been executed. It is proved in Figure 6 that more simulation times can usually provide more reliable statistical results, leading to a more stable convergence curve and reducing errors caused by randomness.

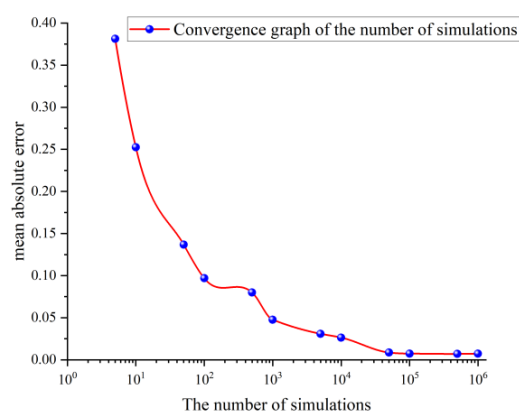


Figure 6. Convergence graph of the number of simulations

Finally, with regard to three indicators in neural networks, we select 7 different generations and calculated 7 different average absolute errors through our established 5-dimensional neural network model. Additionally, we select 5 different hidden layers and calculated 5 different average absolute errors through our established 5-dimensional neural network model. In the matter of the selection of learning rates, we select 6 different learning rates and calculated 6 different average absolute errors through our established 5-dimensional neural network model. Figure 7 shows convergence graphs of three indicators in neural networks. We discover that when selecting the appropriate generations, hidden layers and learning rate, the neural network model will have good goodness-of-fit.

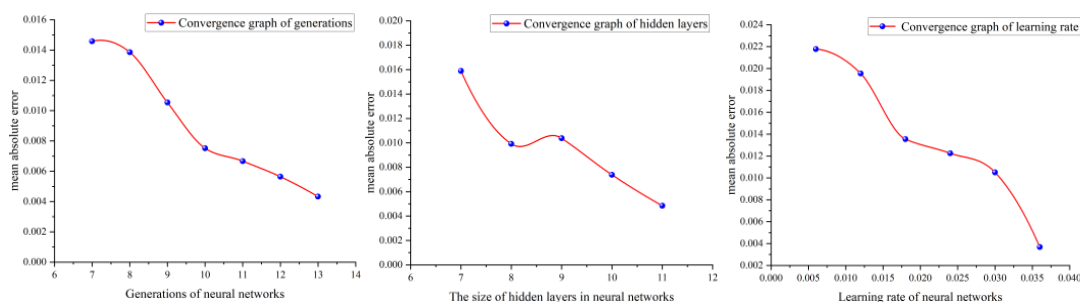


Figure 7. Convergence graphs of three indicators in neural networks

5. Conclusion

For the UQ problem, this thesis transforms it into a probability and statistics problem using the Feynman–Kac formula and neural network regression, which is more helpful in solving high-dimensional problems compared to ordinary PDEs solutions. In the process of numerical experiments, this thesis first uses N samples with uniform distribution sampling, and then calculate the numerical solution u of the UQ problem using the Feynman–Kac formula. Then, this thesis obtains the expected mean of the UQ problem solution through polynomial regression and neural network methods, respectively. Based on data comparison, it is found that the neural network model fits well, especially in high-dimensional situations where the polynomial regression model may encounter dimension curse problems while the neural network model does not. Finally, a convergence analysis is conducted on a 5-dimensional neural network model, analyzing the impact of five indicators - time step, number of simulations, generations, the number of hidden layers, and learning rate - on the convergence of the model. By drawing convergence graphs, it turns out that shorter time steps, more simulations, suitable generations, hidden layers, and learning rate have a positive impact on the convergence of the model. As far as the traditional method of solving the integration of the Feynman–Kac formula with the help of Monte Carlo simulation method [2], the calculation amount of this thesis is also relatively small. Their method is often calculated many times at one point, and the number of global calculations of this research method is relatively small. Secondly, for the simulation of the path of stochastic differential equations, there are also simulations under the polar coordinate system, and some other acceleration methods of Monte Carlo simulation, which may achieve better results when combined with this method. Finally, it is also worth exploring the use of deep learning algorithms to solve the UQ problem.

References

- [1] Kuzmin V A, Korbut N T. Adaptation of the Polynomial Chaos Method for the Uncertainty Analysis of Critical Experiments [J]. *Physics of Atomic Nuclei*, 2018, 81 (10): 1404-1407.
- [2] Sylvain Maire, and Etienne Tanré. Some new simulations schemes for the evaluation of Feynman–Kac representations. *Monte Carlo Methods and Applications* 14.1 (2008): 29-51.
- [3] Étoré Pierre, Martinez Miguel. A transformed stochastic Euler scheme for multidimensional transmission PDE [J]. *Journal of Computational and Applied Mathematics*, 2021, 394.
- [4] Tang Tao, Zhou Tao. High precision numerical methods and theories for uncertainty quantification dedicated to Professor Lin Qun's 80th birthday [J]. *Chinese Science: Mathematics*, 2015, 45 (07): 891-928.
- [5] Felix A, Christian B, Martin E, et al. SDE Based Regression for Linear Random PDEs [J]. *SIAM Journal on Scientific Computing*, 2017, 39 (3): A1168-A1200.
- [6] Yassin K, Danilo M, Ervin S. A review of Hidden Markov models and Recurrent Neural Networks for event detection and localization in biomedical signals [J]. *Information Fusion*, 2021, 6952-72.
- [7] Phuoc T N, Anh T N, Van-Hong T P, et al. A comprehensive analysis for multi-objective distributed generations and capacitor banks placement in radial distribution networks using hybrid neural network algorithm [J]. *Knowledge-Based Systems*, 2021, 231.
- [8] Ganesh M K, Cheston T, Camilo L, et al. A Nonlinear Hidden Layer Enables Actor-Critic Agents to Learn Multiple Paired Association Navigation [J]. *Cerebral cortex (New York, N.Y.: 1991)*, 2022, 32 (18).
- [9] Yuanyuan L, Qianqian Z, Won S Y. Gaussian process regression-based learning rate optimization in convolutional neural networks for medical images classification [J]. *Expert Systems With Applications*, 2021, 184.