Numerical Solution Method for Stochastic Partial Differential Equations Based on the Feynman-Kac Formula and Random Forest Regression

Zichang Liu *

School of Mathematics, Jilin University, Changchun, China, 130012

* Corresponding Author Email: liuzc1022@mails.jlu.edu.cn

Abstract. The Feynman-Kac formula establishes a crucial connection between stochastic processes and partial differential equations (PDEs). By expressing the solution of a PDE as the expected value of a stochastic variable, this formula provides an innovative approach to numerically solving PDEs and can be extended to compute the expected solutions of stochastic partial differential equations (SPDEs). However, in practical applications, the numerical stability of this formula is often affected by the range of stochastic variable values. When the stochastic variable takes excessively large or small values, computational errors may increase sharply, potentially rendering the formula ineffective. To address this issue, this paper proposes an adaptive algorithm that dynamically adjusts computational parameters, enhancing the applicability of the Feynman-Kac formula to a broader class of problems. Specifically, the algorithm adaptively optimizes the sampling strategy based on the range of stochastic variable values, ensuring computational stability and convergence. Furthermore, this paper extends the adaptive algorithm to the solution of SPDEs and integrates it with the random forest method to achieve data-driven fitting and approximation of the expected solution.

Keywords: Feynman-Kac Formula, Numerical Solution of Stochastic Differential Equations, Random Forest, Adaptive Step-Size Algorithm

1. Introduction

The Feynman-Kac formula is a powerful tool for solving partial differential equations (PDEs). It enables a pointwise stochastic representation when dealing with various PDEs such as diffusion equations and transport equations [1]. Taking the Dirichlet boundary value problem as an example, suppose the problem is defined within a certain domain $D \subset R^n$, where the domain boundary ∂D is sufficiently smooth and satisfies specific conditions:

$$-\Delta u = f \tag{1}$$

The boundary condition is set as:

$$u = g, x \in \partial D \tag{2}$$

In this case, the solution to the Dirichlet boundary value problem at any point within the domain can be expressed as [2]:

$$u(z) = E_z\left[g(X_\tau) + \int_0^\tau f(X_s) ds \right] \tag{3}$$

Where X_t the solution to the stochastic differential equation (SDE) is associated with the Laplace operator, and τ_D represents the exit time of this process from the physical domain D of the problem.

In practical computations, this paper employs the Monte Carlo method to obtain the solution [3-4]. This approach simulates many random paths using computational methods, calculates the corresponding values for each path, and finally takes the average to approximate the true solution.

When simulating a single path, discretization of the stochastic process is necessary for implementation on a computer. The simulation begins at the target solution location, and the next position is determined based on the previous position and the stochastic differential equation corresponding to the target partial differential equation. This process continues until the point reaches the boundary, which is determined when the distance to the boundary falls below a predefined small threshold (e.g., if the threshold distance is set to ε, the point is considered to have reached the boundary when its distance to the boundary is less than ε).

For each segment of the path, the integral is computed using the right rectangle rule. If each small segment of the path starts at time t_i and ends at time t_{i+1}, the right rectangle rule approximates the integral as follows:

$$\sum_i f\left(X_{t_{i+1}}\right)\left(t_{i+1}-t_i\right) \tag{4}$$

By summing the integral values computed for each segment along the path, this paper obtains the approximate value of the integral. The function value g is taken at the final point of the path.

Following the above algorithmic procedure, the expectation solution of the stochastic partial differential equation can be obtained.

To illustrate this approach, this paper considers a specific stochastic partial differential equation (SPDE) as an example [5]:

$$-\nabla\cdot\left(k\left(X,\omega\right)\cdot\nabla u\left(X,\omega\right)\right)=f\left(X,\omega\right),X\in D \tag{5}$$

Where the equation is defined within the physical domain $D\subset R^n$ and satisfies the boundary condition $u=g,X\in\partial D$. This equation is formulated in a complete probability space $(\Omega,\mathcal{F},P)$, where Ω is the sample space, $\mathcal{F}$ is the associated sigma-algebra, and P is the probability measure. The diffusion coefficient and the source term are given stochastic functions, leading to a stochastic solution. In most cases, directly obtaining an exact solution for this stochastic function is highly challenging. Therefore, this paper focus on solving its expected solution.

Using the Monte Carlo method [6], this paper samples values for the stochastic variables according to their respective distributions. This transforms the stochastic partial differential equation into an ordinary partial differential equation. This paper then solves the transformed PDE using the Dirichlet boundary problem approach discussed earlier. By repeatedly sampling the stochastic variables and solving the corresponding deterministic PDEs, this paper obtains multiple solutions. The expectation solution of the SPDE is then approximated by averaging these solutions.

For the given problem, the evolution formula of the associated stochastic process is:

$$dX_t=\nabla k\left(X,\omega\right)\cdot dt+\sqrt{2k\left(X,\omega\right)}dW_t \tag{6}$$

The corresponding discretized iterative formula is given by:

$$X_{t+1}=X_t+\nabla k\left(X,\omega\right)\cdot\Delta t+\sqrt{2k\left(X,\omega\right)\Delta t}\cdot N \tag{7}$$

Where Δt represents the time step, and N is an n-dimensional column vector, with each component independently following a standard normal distribution. When applying this iterative formula, it is important to note that it contains stochastic variables. If the stochastic variable takes excessively small values, the spatial step size becomes too small, significantly increasing computational cost. Conversely, if the stochastic variable takes excessively large values, the spatial step size becomes too large, leading to poor discretization of the trajectory, making the iterative formula no longer applicable.

To address this issue, this paper proposes an adaptive algorithm [7-8]. The core idea of this algorithm is to check the step size during the discretization of the motion trajectory, discard unreasonable step sizes, and adjust the time step accordingly to make it reasonable. This effectively mitigates the previously mentioned issues.

Furthermore, using machine learning techniques [9-11], once the numerical solutions at selected points in the domain are obtained, a random forest model is employed to fit the results. Each dimension of the solution acts as a feature in the model (i.e., an nnn-dimensional physical domain corresponds to nnn features). Experimental results demonstrate that when solving high-dimensional problems, the random forest model significantly reduces computation time and improves the accuracy of the solution compared to traditional polynomial regression, proving to be a highly effective method.

2. Research on Adaptive Step Size Strategy

During the implementation of this algorithm, the magnitude of the random variable significantly affects the discretization of the path. If the random variable takes excessively large values, the step size will become too large, which substantially increases computational errors and leads to greater deviation from the true value. Conversely, if the random variable takes excessively small values, although it theoretically improves the accuracy of path discretization, it significantly increases the computational time required by the algorithm, drastically reducing efficiency. Both extreme cases hinder the effectiveness of the original algorithm and may even render it unusable.

To address this issue, this paper proposes an adaptive step-size method [9]. The core idea of this method is to fully consider the size (measure) of the solution domain when simulating the movement path of points and establish a standard step size s accordingly. After obtaining the step size from path discretization, it is compared with the standard step size s. If the step size does not fall within a certain neighborhood centered at s (e.g., a neighborhood $(s-\delta, s+\delta)$, where δ is an adjustable parameter), the relevant parameters are adjusted. Specifically, the parameter magnitude is modified, and a new adjusted value $\Delta t'$ replaces the original Δt . Based on the updated $\Delta t'$, a new step size is generated. The verification process is then repeated—checking whether the new step size falls within the designated neighborhood $(s-\delta, s+\delta)$. This loop continues until the step size meets the specified conditions. The values of s and δ are adjustable parameters, and their proper selection plays a crucial role in ensuring the accuracy of the results.

Here is a numerical experiment example. Specifically, this paper aims to solve the following partial differential equation:

$$\begin{cases} -\nabla\left(k(A,x,y)\cdot\nabla u(A,x,y)\right) = f(A,x,y), (x,y)\in D \\ u(A,x,y) = g(A,x,y), (x,y)\in \partial D \end{cases} \tag{8}$$

Where:

$$\begin{aligned} &k(A,x,y) = A\cdot(x+y) \\ &g(A,x,y) = A\cdot e^{xy} \\ &f(A,x,y) = -A^2\cdot(x+y)\cdot\left(x^2+y^2+1\right)\cdot e^{xy} \\ &D = [0,1]\times[0,1] \end{aligned} \tag{9}$$

The exact solution to this equation is:

$$u(A,x,y) = A\cdot e^{xy} \tag{10}$$

The algorithm process is as follows: First, the parameters s , δ in the above approach are determined. Then, select the point to be solved, use this point as the starting point, simulate the path of the point's movement, and during the discretization of the path, if the step size for the next step is too large or too small, i.e., not within the interval $(s-\delta, s+\delta)$, Δt will be replaced by $\Delta t' = \left|\frac{s}{\delta}\right|\cdot\Delta t$, then the step size will also be replaced, and the process is repeated until the step size is appropriate.

Next, this paper evaluates the performance difference between the original algorithm and the improved algorithm by comparing them with the exact solution. To fully reflect the randomness of the random variable, a fixed random variable A is set, taking values sequentially as 100, 1, and 0.01.

Table 1 presents the numerical results of the original algorithm at the test point (0.5, 0.7), while Table 2 records the numerical results obtained using the adaptive sampling method.

Table 1. The numerical solution error results of the original algorithm.

Comparison	Numerical solution	Real solution	Time(s)
A=10000	-1.1126*1e255	14190.67	0.0056
A=1	1.4192	1.4191	0.5
A=0.001	0.001422	0.001419	56.29

Table 2. Numerical solution error results of the adaptive step-size algorithm.

Comparison	Numerical solution	Real solution	Time(s)
A=10000	14121.58	14190.67	0.0029
A=1	1.4190	1.4191	0.27
A=0.001	0.001412	0.001419	0.27

The comparison results clearly show that the improved algorithm effectively addresses the issues present in the original algorithm. When the value of the random variable is too large, the step size generated by the formula becomes excessively large, making it impossible to discretize the original stochastic process curve accurately. After improving the time step using the adaptive algorithm, the generated step size becomes more appropriate. The improved algorithm significantly ensures computational accuracy, avoiding large errors caused by overly large step sizes. When the value of the random variable is too small, the step size generated by the formula becomes too small, causing the discretization of the curve to be excessively fine, which greatly increases computation time. The improved algorithm significantly reduces the time required for implementation, lowers the computational burden, and enhances computational efficiency. This demonstrates that the adaptive sampling method has a clear advantage in optimizing algorithm performance and can better meet the demands of practical computations.

3. Numerical solution of PDEs based on adaptive step-size strategy and random forests.

Stochastic partial differential equations (SPDEs) have broad and significant applications in various fields, such as financial mathematics and biology, making the exploration of efficient and accurate solution methods highly important. In line with the approach described at the beginning of this article, the first step is to obtain numerical results at certain points within the domain, and then use these results for fitting, which will allow for the approximation of the true solution.

In low-dimensional scenarios, polynomial regression is commonly used as a solution method. However, when the dimensionality of the problem increases, polynomial regression reveals several shortcomings. On one hand, overfitting is a frequent issue, leading to good performance on training data but poor generalization ability on new data. On the other hand, as the dimensionality increases, the number of polynomial basis functions increases dramatically, resulting in a significant increase in computational complexity. Additionally, this method struggles to handle complex nonlinear relationships and becomes less suitable for solving high-dimensional nonlinear problems, gradually losing its practical applicability.

Considering this, this paper proposes a new solution strategy that combines an adaptive algorithm with a random forest algorithm. Random forests are an ensemble learning method and an extension of decision tree algorithms. They improve model accuracy and robustness by combining multiple decision trees. The random forest trains several decision trees, and then integrates their predictions to enhance model performance. The basic structure of a decision tree is a tree-like structure, where each node represents a feature decision, the connections between nodes represent feature values, and the leaf nodes indicate class labels or predicted values. The process of building a decision tree involves

selecting the optimal splitting point based on the data features, recursively splitting until certain stopping conditions (such as tree depth or the number of samples in the leaf nodes) are met.

Specifically, this paper first uses the adaptive algorithm to compute the results at certain points within the domain (for example, by selecting 100,000 points). These results are then organized into a dataset. Next, the random forest algorithm is used to train the dataset, constructing the corresponding regression model. Finally, the output of this regression model is taken as an approximation of the true solution of the stochastic partial differential equation. This combination of algorithms leverages the strengths of both methods and is expected to provide a more effective approach for solving high-dimensional stochastic partial differential equations.

To demonstrate the feasibility of this method, this paper presents the following numerical experiment:

$$\begin{cases} -\nabla\left(k(X,\omega)\cdot\nabla u(X,\omega)\right) = f(X,\omega), X \in D \\ u(X,\omega) = g(X,\omega), X \in \partial D \end{cases} \tag{11}$$

Were

$$\begin{aligned} & g(X,\omega) = \sum_{i=1}^{5} \sin(A_i x_i) \\ & k(X,\omega) = B\cdot\left(\sum_{i=1}^{5} x_i^2\right) + C \\ & f(X,\omega) = \left[B\cdot\left(\sum_{i=1}^{5} x_i^2\right) + C\right]\left(\sum_{i=1}^{5} A_i^2 \sin(A_i x_i)\right) - 2B\left[\sum_{i=1}^{5} A_i x_i \cos(A_i x_i)\right] \\ & D = [0,\pi]^5 \\ & A_i \sim U(0,1), i = 1,2,\ldots 5 \\ & B \sim U(0,100) \\ & C \sim Exp(1) \end{aligned} \tag{12}$$

And the true solution of this equation is:

$$u(X,\omega) = \sum_{i=1}^{5} \sin(A_i x_i) \tag{13}$$

Thus, the expected solution of this equation is:

$$E(u) = \sum_{i=1}^{5} \frac{1-\cos x_i}{x_i} \tag{14}$$

Now, this paper applys the above algorithm to solve this problem. First, this paper randomly selects 500,000 points within the physical domain D and use the adaptive algorithm for each point. For each selected point, this paper simulates 10 paths of the random variable to obtain an approximate expected solution. Next, this paper employs polynomial regression and random forest methods to construct regression functions or train predictive models. Then, this paper randomly selects another 500,000 points, compare their true solutions with the numerical solutions obtained from the regression functions or trained models, and finally compute the average relative error to evaluate the accuracy of the method.

The experimental results show that when the number of decision trees in the random forest is set to 100, the tree depth is 25, and the maximum number of features is 5, the running time is approximately 186 seconds with an average relative error of 1.92%, which gives good results. In contrast, when polynomial regression is used with a polynomial degree of 5 and 792 basis functions, the running time is approximately 47 seconds, but the average relative error reaches 6.4%. This demonstrates that for high-dimensional problems, polynomial regression struggles to achieve a good fit, whereas random forests clearly show superior performance in this regard.

Then, let's compare the performance of these two methods in different dimensions. The model equation this paper uses is as follows:

$$\begin{cases} -\nabla\left(k(X,\omega)\cdot\nabla u(X,\omega)\right)=f(X,\omega), X\in D \\ u(X,\omega)=g(X,\omega), X\in\partial D \end{cases} \tag{15}$$

Were

$$\begin{aligned} &g(X,\omega)=\sum_{i=1}^{k}\sin\left(A_i x_i\right) \\ &k(X,\omega)=B\cdot\left(\sum_{i=1}^{k}x_i^2\right)+C \\ &f(X,\omega)=\left[B\cdot\left(\sum_{i=1}^{k}x_i^2\right)+C\right]\left(\sum_{i=1}^{k}A_i^2\sin\left(A_i x_i\right)\right)-2B\left[\sum_{i=1}^{k}A_i x_i\cos\left(A_i x_i\right)\right] \\ &D=[0,\pi]^k \\ &A_i\sim U(0,1), i=1,2,\ldots k \\ &B\sim U(0,100) \\ &C\sim Exp(1) \end{aligned} \tag{16}$$

The true solution of this equation is:

$$u(X,\omega)=\sum_{i=1}^{k}\sin\left(A_i x_i\right) \tag{17}$$

Thus, the expected solution of this equation is:

$$E(u)=\sum_{i=1}^{k}\frac{1-\cos x_i}{x_i} \tag{18}$$

To compare the performance of polynomial regression and random forest algorithms in different dimensions, this paper sets k=1, 3, 5, and 7 in turn and compare the results of these two methods, as shown in the following table:

Table 3. The experimental results of different regression methods applied to SPDEs in different dimensions.

Dimension	Polynomial regression Model	Random Forest Regression Model
1	Average relative error: 0.096% time: 0.36s	Average relative error: 22% time: 1.19s
3	Average relative error: 1.1%time: 14.5s	Average relative error: 2.6% time: 78.4s
5	Average relative error: 6.4% time: 20.1s	Average relative error: 1.92% time: 120s
7	Average relative error: 14.2% time: 47s	Average relative error: 1.74% time: 186s

In high-dimensional cases, although the random forest algorithm requires more computational time, its accuracy can still be effectively guaranteed. This is because random forests can handle the complexity of high-dimensional data by integrating multiple decision trees and randomly selecting features, allowing them to effectively capture nonlinear relationships and avoid overfitting, thus providing stable and high-precision results. In contrast, polynomial regression often fails to guarantee accuracy in high-dimensional cases because it is susceptible to the curse of dimensionality, and when fitting complex nonlinear relationships, it is prone to overfitting or unable to find a valid solution. Therefore, random forests exhibit stronger robustness and adaptability in high-dimensional problems.

Next, this paper takes equations (11)-(14) as examples, focusing on the random forest model, and conducts a study on four key parameters (Here this paper selects 100,000 points): the number of decision trees, the depth of the trees, the maximum number of features, and the number of samples. Following the previous verification method, this paper analyzes the relationship between the number of decision trees, tree depth, maximum number of features, and the number of samples in the random forest model with the average relative error.

First, this paper observes how the average relative error changes with the number of decision trees. By plotting the number of decision trees on the horizontal axis and the average relative error on the vertical axis, we can clearly see that as the number of decision trees increases, the average relative

error decreases. However, when the number of decision trees reaches around 100, the change in the average relative error becomes insignificant. To avoid overfitting and to balance computation time, after comprehensive consideration, setting the number of decision trees around 100 seems to be most appropriate.

Next, this paper analyzes how the average relative error changes with tree depth. Similarly, by plotting tree depth on the horizontal axis and the average relative error on the vertical axis, we can observe that when the tree depth is relatively high (greater than around 15), the average relative error no longer changes significantly with the tree depth. However, when the tree depth is below this value, the average relative error decrease drastically with increasing depth, reaching its minimum when the depth is about 15. Based on this, this paper divides the nodes equally around a depth of 15, ultimately determining the tree depth for this algorithm to be 15.

Next, this paper looks at how the average relative error changes with the maximum number of features. By plotting the maximum number of features on the horizontal axis and the average relative error on the vertical axis, we can see that when this parameter falls between 0.5 and 0.9, or when it is greater than 5, the average relative error is relatively smaller. To fully utilize all features, and after considering various factors, this paper selects 5 as the maximum number of features.

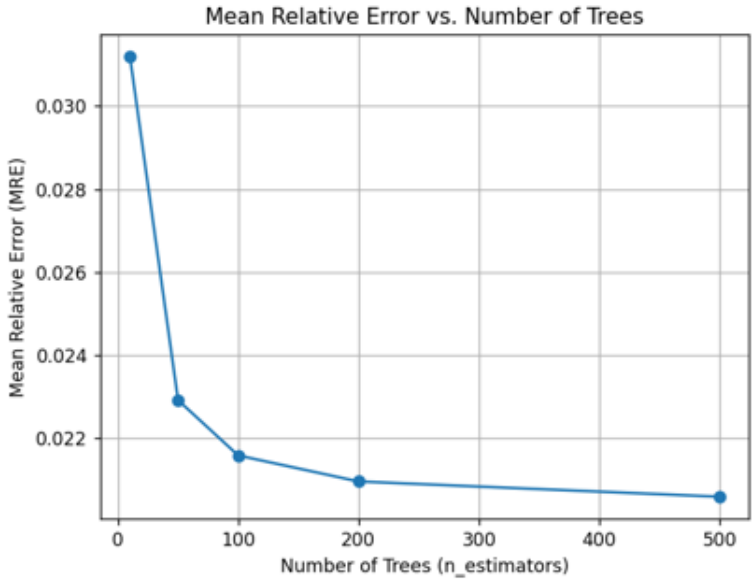

Figure 1. Variation of the mean relative error of the numerical solution with the number of trees.

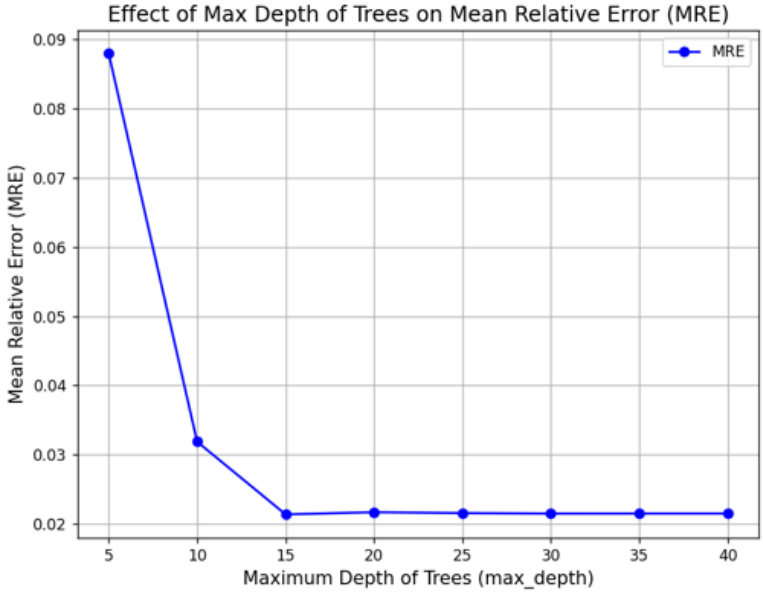

Figure 2. Variation of the mean relative error of the numerical solution with the depth of the trees.

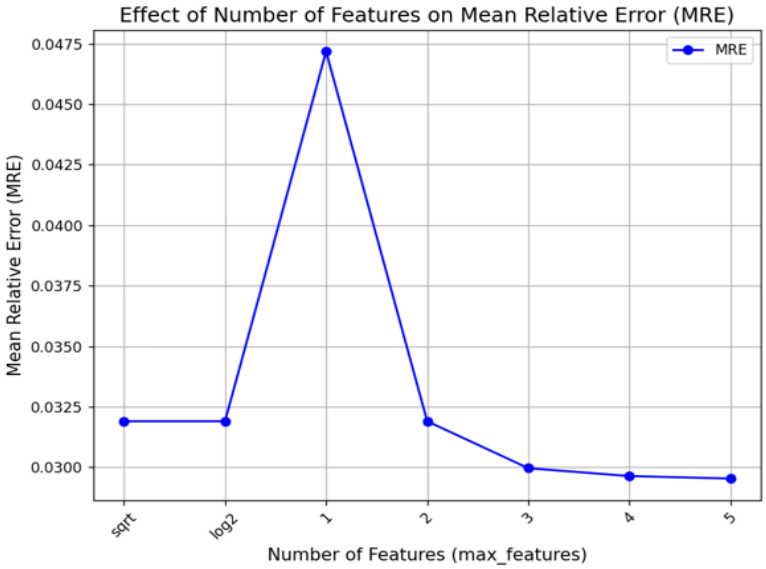

Figure 3. Variation of the mean relative error of the numerical solution with the depth of the maximum number of features

Finally, the variation of the mean relative error with the number of samples is studied. It can be observed that as the sample size increases, the mean relative error gradually decreases. However, the increase in sample size leads to an increase in computational cost. After considering both the computational cost and the error requirements, when the number of samples reaches 100,000, the obtained results meet the needs. Therefore, 100,000 is chosen as the sample size parameter for this algorithm.

Table.4. Numerical solution's average relative error with varying sample size

Samples	Average relative error
2000	0.0376
10000	0.0302
50000	0.0269
100000	0.0216
200000	0.0204
500000	0.0192

From the perspective of decision trees, as the number of decision trees increases, the algorithm will theoretically converge to a stable solution. As we observed when analyzing the change of average relative error with the number of decision trees, when the number of trees reaches about 100, the error variation becomes less significant, which initially suggests the convergence of the algorithm to a certain extent. However, we still need to conduct more rigorous derivations and proofs from a mathematical theoretical standpoint. By using probability theories such as the law of large numbers and the central limit theorem, we can analyze whether, as the number of decision trees increases infinitely, the algorithm's output converges to the true solution with probability.

The depth of the tree also plays a crucial role in convergence. Shallow tree depths may fail to adequately learn the complex patterns in the data, leading to slow convergence or even failure to reach a good solution. On the other hand, excessive tree depth may lead to overfitting, where the algorithm performs well on the training data but its generalization ability on new data deteriorates, thereby affecting the accuracy of convergence to the true solution. By analyzing the convergence speed and accuracy under different tree depths, this paper further optimizes the selection of tree depth to ensure that the algorithm converges quickly and accurately.

The number of samples is also closely related to convergence. An increase in sample size provides more information for the algorithm, helping it more accurately learn the distribution characteristics of the data, thus promoting convergence. However, an increase in sample size also leads to a

significant increase in computational load. Therefore, we need to strike a balance between convergence speed and computational cost. By combining theoretical analysis and numerical experiments, this paper can study the quantitative relationship between sample size, convergence speed, and convergence accuracy, providing a more solid theoretical foundation for the reasonable selection of sample size.

4. Conclusions

This paper focuses on the problem of solving the expected solution of random partial differential equations (SPDEs) and proposes an innovative improved adaptive algorithm. This algorithm, based on the Feynman-Kac formula, ensures the accuracy of the solution while cleverly addressing the many challenges posed by the uncertainty of random variables. Building upon this, the adaptive algorithm is further integrated with the random forest method, effectively solving the expected solution of SPDEs in high-dimensional settings, significantly outperforming traditional methods in terms of result precision.

As a powerful mathematical tool, SPDEs have broad and important applications in various fields such as meteorology, finance, and biology. However, in high-dimensional cases, traditional numerical methods often face the curse of dimensionality and computational complexity, making it difficult to solve efficiently and accurately. The adaptive algorithm proposed in this paper demonstrates exceptional adaptability and superiority, overcoming the high-dimensional challenges while ensuring computational precision. It is particularly adept at handling complex high-dimensional stochastic processes, such as dynamic simulations of meteorological changes, volatility forecasting in financial markets, and the evolution analysis of biological systems.

When the adaptive algorithm is combined with the random forest algorithm, the complementary strengths of both methods significantly enhance computational efficiency and the accuracy of the solution. This innovative combination opens new avenues for the numerical solution of high-dimensional SPDEs, providing a more efficient and precise tool that is expected to advance research and development in related fields in practical applications.

References

[1] Hawkins K P , Pakniyat A , Tsiotras P .Value function estimators for Feynman–Kac forward–b ackward SDEs in stochastic optimal control[J].Automatica, 2023, 158(000):-1.DOI:10.1016/j.autom atica.2023.111281.

[2] Choi B S , Choi M Y .On Some Results of the Nonuniqueness of Solutions Obtained by the Feynman–Kac Formula[J].Mathematics (2227-7390), 2024, 12(1).DOI:10.3390/math12010129.

[3] LAN Chengming, Xu Zhenqian, Ma Junming, et al. Comparison of Markov Chain Monte Carlo Sampling Algorithms in Subset Simulation [J]. Journal of Civil Engineering, 2022, 55(10):14. DOI: 10.15951/j.tmgcxb.22030200.

[4] Ma Guozhen, Wang Yunjia, Wang Zhumei, Du Wentong. Electric Vehicle Disorderly Charging Load Calculation Model Based on Monte Carlo Random Simulation Algorithm [J]. Journal of Computational Technology and Automation, 2024.

[5] Yang Xu, Tang Mengtong. Existence and Uniqueness of Strong Solutions for Stochastic Partial Differential Equations with Gradients [J]. Journal of Applied Probability and Statistics, 2022, 38(6):931-940.

[6] He Yugong. Convergence and Variance-Finiteness Conditions of Monte Carlo Methods for Solving Elliptic Partial Differential Equations [J]. Journal of Tsinghua University (Natural Science Edition), 1964(02):109-128. DOI: CNKI: SUN: QHXB.0.1964-02-006.

[7] Li Yan. Study on Implicit Runge-Kutta Algorithm for Richards Equation [D]. Hunan University, 2022.

[8] Sang Lin, Zhang Cheng. Adaptive Interpolation Wavelet Method for Numerical Solutions of Partial Differential Equations [J]. Journal of Biomathematics, 2001, 016(002):145-150.

[9] Chen Jingfei. Machine Learning Methods for Solving Partial Differential Equations: From Network Structures to Optimization Algorithms [D]. Soochow University, 2023.

[10] Zhu Quanhui. Using Machine Learning Methods to Solve Fractional Order Partial Differential Equations and Discontinuous Problems [D]. Harbin Institute of Technology, 2020.

[11] Peng Pai, Feng Xinlong. Solving Steady-State Partial Differential Equations Using Deep Learning Methods Based on Two-Level Grids [J]. Journal of Xinjiang University: Natural Science Edition (Chinese and English), 2022, 39(4):9.

[12] Fang Kuangnan, Wu Jianbin, Zhu Jianping, et al. A Review of Random Forest Methods [J]. Statistics and Information Forum, 2011, 26(3):7. DOI: 10.3969/j.issn.1007-3116.2011.03.006.