Solving Fokker–Planck equations using deep KD-tree with a small amount of data

Hao Zhang · Yong Xu · Qi Liu · Xiaolong Wang · Yongge Li

Abstract The Fokker–Planck (FP) equation can deterministically describe the evolution of the probability density function, which plays an extremely significant role in the fields of stochastic dynamics. Unfortunately, the limited samples that arise from the consideration of engineering practice are inevitable, which restricts the solving of the FP equation. Accordingly, in the present study, a super-DL-FP framework is established to solve the steady-state FP equation with a small amount of data, through combining the deep KD-tree and the DL-FP approach proposed in [Chaos 30, 013133 (2020)]. It should be emphasized that the normalization condition is of great importance and has to be considered in solving the steady-state FP equation. An appropriate integral estimation for the normalization condition under non-uniform meshing can effectively improve the precision of the solution, but it is still a challenging problem, especially for the case of small data. Thus, the so-called deep KD-tree method is innovatively proposed to estimate the normalized integral with a small random dataset. The main target is to obtain the appropriate discrete integral points and corresponding integral volumes by executing multiple KD-tree segmentation based on random data on the integral region. Several numerical experiments and comparisons are implemented to illustrate the superior performance of the super-DL-FP method. The obtained results indicate that the proposed algorithm can accomplish higher accuracy in the sense of lower cost than the well-known algorithms like center difference scheme, Chebyshev spectrum algorithm, and normalized flow approach.

Keywords Fokker–Planck equation · Deep KD-tree · Deep learning · Super-DL-FP

1 Introduction

The Fokker–Planck (FP) equation can describe the evolution for the distribution of fluctuating macroscopic variables in stochastic dynamics [19]. The application of the FP equation is not restricted to microscopic systems such as thermal equilibrium [6], bioscience [16,31], and chemistry [3], but also macroscopic systems such as economics [8] and engineering [25]. By solving the FP equation, one obtains distribution
functions from which any averages of macroscopic variables are obtained by integration. Although the FP equation is essentially a second-order parabolic partial differential equation, it is challenging to solve it theoretically [19]. The present research on the numerical solution to the FP equation is limited by meshing. In particular, for the steady-state FP equation with natural boundary, all the discrete points need to be involved in computation as the emergence of normalization condition. The number of calculation points increases exponentially with the increase in dimension, which leads to a huge amount of calculations. Besides, in engineering practice, no enough experimental points are available to meet the computing requirements in general. Therefore, this paper mainly focuses on how to reduce the need for calculation points.

The discretization methods for solving the FP equations are strongly depended on meshing, resulting in an exponential increase in computation with dimension, for instance, finite element [9,17], finite difference [13,20], path integral [5,26]. In particular, it must make a trade-off between precision and hardware storage capacity when dealing with high-dimensional cases. Meanwhile, recalculation or interpolation has to be done in solving the values on the off-grid points as only the points on the grid can be obtained. The Monte Carlo simulation [22,24] is also a common approach for solving FP equations, which is a simple and efficient tool for solving one- and two-dimensional probability density function (PDF). However, this approach needs to consume massive samples and is not feasible with the dimension increases.

Recently, to solve the grid dependency problem, the neural networks (NNs) are proposed to solve differential equations, even the steady-state FP equation with natural boundary [4,10,15,18,21,23,27–30]. However, the huge amount of calculation data is still required. The NN can calculate the derivative operator analytically, which means that the derivative of one point does not depend on other points. Hence, the random points can be adopted as calculation points. Meanwhile, the higher dimensional distribution function is sparse, which means that a small number of points can be used to estimate the integral of PDF. Thus, as long as the number of basic training points is satisfied, the solution can be completed by using NN. However, it also brings up a key question which is how to calculate the integral for these finite stochastic experimental data points with appropriate geometric volume. To solve this problem, we propose a deep KD-tree algorithm constructed by multi-layer KD-tree, which can adaptively and efficiently obtain the corresponding integral volume for random training points, and the separated boundary points for the boundary condition. Through combining with the DL-FP framework [27] proposed in our previous work, the super-DL-FP algorithm is proposed to make it possible to solve the FP equation with a small amount of data.

The structure of this paper is as follows. Sec. II gives a detailed description of the algorithms including KD-tree, deep KD-tree and the super-DL-FP approach. In Sec. III, three examples consisting of 2D, 4D and 6D systems are shown to demonstrate the super-DL-FP can be used successfully. At the same time, we compare the proposed algorithm with other approaches including difference schemes and normalized flow algorithm to illustrate the superiority of the super-DL-FP. In the last section, discussions and conclusions are presented to close this paper.
2 Methodology

In this section, three sub-sections are shown to explain the proposed algorithm. The FP equation and the necessary conditions for its solution will be described in detail in Sect. 2.1. In Sect. 2.2, the deep KD-tree algorithm constructed by multi-layer KD-tree will be elaborated at length and the algorithm flow will be given. We will introduce how to combine deep KD-tree and DL-FP algorithm to build super-DL-FP scheme in Sect.2.3, and give a detailed algorithm flow framework.

2.1 The Langevin equation and the Fokker–Planck equation

For \( N \) variables \( x = x_1, x_2, ..., x_N \) the general Langevin equations have the form (\( i = 1, 2, ..., N \))

\[
\dot{x}_i = h_i(x, t) + g_{ij} \Gamma_j(t),
\]

where \( \Gamma_j(t) \) are Gaussian random variables with zero mean and with correlation functions proportional to the \( \delta \) function, i.e., \( \langle \Gamma_i(t) \rangle = 0 \) and \( \langle \Gamma_i(t) \Gamma_j(t') \rangle = 2 \delta_{ij} \delta(t - t') \), where \( \langle \cdot \rangle \) represents the expectation.

The corresponding FP equation for the probability density \( p(x, t) \) is

\[
\frac{\partial p(x, t)}{\partial t} = \mathcal{L}_{FP} p(x, t),
\]

\[
\mathcal{L}_{FP} = -\frac{\partial}{\partial x} D^{(1)}(x, t) + \frac{\partial^2}{\partial x^2} D^{(2)}(x, t),
\]

where \( \mathcal{L}_{FP} \) is the FP operator. Under the definition of Itô integral, \( D^{(1)}(x, t) = h(x, t) \) represents the drift coefficient, and \( D^{(2)}(x, t) = g^2(x, t) \) is the diffusion coefficient.

For a stationary process the probability current must be zero, then the stationary FP equation is

\[
\mathcal{L}_{FP} p(x, t) = 0.
\]

When considering the natural boundary conditions \( (x_{\text{min}} = -\infty, x_{\text{max}} = \infty) \), the probability \( p(x) \) vanishes at \( x = \pm \infty \) and then guarantees that the normalization is preserved

\[
\int_{x_{\text{min}}}^{x_{\text{max}}} p(x) dx = 1.
\]

In summary, four conditions should be considered when solving the stationary FP equation. (i) Satisfying the format of FP equation. (ii) The probability at the boundary is zero. (iii) Satisfying the normalization condition. (iv) The \( p(x) \) is nonnegative. It is just a regularization constraint that can be satisfied as long as the assumed solution is always positive for (iv). As for the other conditions, it is easy to fall into trivial solutions if we only focus on (i) and (ii). Therefore, normalization condition (iii) should never be ignored. When discretized numerical method is used to solve the steady-state FP equation, normalization conditions as a limiting factor make it necessary to utilize all discrete data, which inevitably makes the amount of the computation points rise exponentially with the increase in dimension. Here, we mainly consider using a small amount of random data instead of the whole to solve the FP equation with NN. But along with this comes a problem, that is how to calculate the normalized value for these random points. At this point, the key problem is to match the appropriate integral volume for these random points. Therefore, the deep KD-tree is proposed to solve this problem which will be introduced at length in Sect. 2.2.

2.2 Deep KD-tree algorithm

When calculating integrals numerically, a very simple approach is to discretize the integral which can be expressed as

\[
\int_{x_{\text{min}}}^{x_{\text{max}}} p(x) dx \approx \sum_{i=1}^{N_{\text{all}}} p(x_i) v_i,
\]

where \( x_i \) is the selected integral point, \( v_i \) is the corresponding integral volume, and \( N_{\text{all}} \) represents the number of the selected integral points. The general way to select points is to take points at equal intervals, or some points obtained by polynomial approximation integration, such as the Gauss–Legendre points. At this time, it is easy to get \( v_i \) which can be seen as the volume or the weight of \( p(x_i) \). Nevertheless, it is imperative to select an appropriate segmentation approach to match the corresponding integral volume for these points if the integral points are random experimental points. A very naive idea is whether the KD-tree algorithm [2] could be directly used to divide the integral volume for these random points, inspired by the work [7] which applies the KD-tree to split the volume of the Markov Chain Monte Carlo (MCMC) points and accelerate the Vegas integral method. Three questions should be considered. What is the KD-tree method and how to apply it? Whether the training data and boundary data
required by DL-FP algorithm can be obtained from the KD-tree approach? Whether the volume divided by the KD-tree can be utilized directly? The deep KD-tree method proposed by us will be provided in the process of responding to the three questions.

What is the KD-tree method and how to apply it? The working principle of KD-tree is as follows. The median of the dimension with the largest variance is selected as the parent node, and then the remaining points are divided to both sides as the child nodes based on the selected dimension by making a straight line perpendicular to the hypercube. Then, the dividing method is repeated for the points on both sides until all points are divided. This leads to the problem that the points being divided are basically on the boundary of the partition volume. When the number of points is large enough, one can estimate the integral by taking the points as integral points and multiplying them by the corresponding volume, but when the number of points is small, this calculation becomes problematic. For example, if a high value is chosen where the fluctuation of the function is relatively large, the calculated integral will be incorrect. A conservative approach would be to pick the midpoint of the partition volume as the integral point, which works like the trapezoidal integral formula. Therefore, after KD-tree segmentation, the midpoint of the divided volume should be used as the position of the integral point.

Whether the training data and boundary data required by DL-FP algorithm can be obtained from the KD-tree approach? The training set and boundary set are required when training the FP equation using the DL-FP algorithm. The random points generated by experiments can roughly reflect the distribution function. The probability of points close to geometric boundaries is very small, so they can be used as boundary points for training. When KD-tree is used for partition, the segmented hypercube corresponding to the boundary points at geometric positions must have a dimension that is the boundary of the region to be calculated. Hence, boundary points can be extracted from the divided points. It should be noted that, according to the answer to the first question, the midpoint of the partitioned hypercube is actually used as the integral point, so the points on the boundary will expand out, which is more suitable as the boundary point.

Then, the algorithm flow of KD-tree is given based on the above two questions which is shown in Algorithm 1. Algorithm 1 is mainly used to obtain volume segmentation of random points and the corresponding geometric midpoints. Different from the classical KD-tree [2] approach, we modify the output form of data without caring about the structure of the generated tree.

### Algorithm 1: KD-tree algorithm

**Input:** $k$ dimensional spatial data set $T = \{x_1, x_2,..., x_N\}$, where $x_i = [x_i^{(1)}, x_i^{(2)}, ..., x_i^{(k)}]_j, i = 1, 2,..., N$ the boundary set of $k$ dimensional space $B_{min} = [b_{min}^{(1)}, b_{min}^{(2)}, ..., b_{min}^{(k)}]$ and $B_{max} = [b_{max}^{(1)}, b_{max}^{(2)}, ..., b_{max}^{(k)}].$

**Output:** KD-tree set, and the corresponding volume set.

1. **Construct root node:** choose the dimension $x^{(j)}$ with the largest variance as the segmentation dimension, taking the median $x^{(j)}_{\text{median}}$ of $x^{(j)}$ coordinates of all instances in $T$ as the segmentation point, the segmentation is achieved by a hyperplane that passes through the segmentation point and is perpendicular to the coordinate axis $x^{(j)}$.
2. **Generate the left and right child nodes:** the data in $T$ that is less than the median $x^{(j)}_{\text{median}}$ in the $x^{(j)}$ dimension are passed to the left subtree set and the rest to the right subtree set.
3. **Repeat:** repeat the steps of 1 and 2 for the two subtree sets obtained by step 2, until no instances exist in the subregion.
4. **return:** The volume set corresponding to KD-tree and the midpoint of the subregion corresponding to KD-tree constitute a new KD-tree.

Whether the volume divided by the KD-tree can be utilized directly? To better answer the question, two-dimensional random experimental data from limit cycle oscillation is used for illustration, as shown in Fig. 1. When KD-tree is used to segment the integral hypercube, the volume of the points near the integral boundary will be relatively large, which can be seen from the region covered by the orange points in Fig. 1b. When the DL-FP framework is used for training with all the points, a possible situation is that the NN will obtain an unpromising solution where all the probability masses are assigned to the boundary points while the probabilities of the important integral points, i.e., the green points in Fig. 1b, are zero. Thus, the volume segmentation obtained by using KD-tree cannot be directly used to estimate the integral. The boundary points in Fig. 1b are obtained from the expansion in Fig. 1a. But the region covered by the boundary points cannot be deleted directly, because there are integrals available in the region covered by the boundary points. In particular, the integral points occupy a smaller integral region with the increase in dimension. The process is similar to peeling the skin off a fruit, leaving behind the compact structure inside. Therefore, we need to expand

---

© Springer
The KD-tree method is used to segment the data in region $[-2.5, 2.5]^2$, and all the points are the midpoint of each small volume after segmentation. Merge data, the points are a combination of the original points in (a) and the boundary points in (b). After combining the data, KD-tree is used to segment the region again to obtain the integral points and boundary points where the green ones are the integral points and the orange ones represent the boundary points.

The points shown in Fig. 1c are formed by combining the orange points in Fig. 1b and the original data in Fig. 1a. Then, the points in Fig. 1c are segmented by Algorithm 1, and the segmentation in Fig. 1d can be obtained. By comparing Fig. 1d with Fig. 1b, it can be found that the effective integral region covered by the green points in the middle has increased. This expansion can be done multiple times as the dimensions increase. The specific process is as follows. The boundary points are separated by KD-tree once, then combined with the original data, and then iterating the KD-tree algorithm for the combined data.
from last iteration. The termination condition of iteration is to set the data ratio coefficient \( \gamma \). The amount of training data can be controlled by adjusting \( \gamma \) according to the actual situation. The amount of training data can affect the training speed of the NN, but this is really a trade-off between accuracy and speed. We name this method deep KD-tree which is shown in Algorithm 2. This process is equivalent to stretching the peeled fruit skin and then covering the original fruit. When KD-tree is performed again, more integral points with compact integral volume structure can be obtained. In other words, this process can also be regarded as matching suitable geometric boundaries and corresponding integral volume segmentation for random points.

**Algorithm 2** Deep KD-tree algorithm

**Input:** \( k \) dimensional spatial data set \( T = \{x_1, x_2, ..., x_N\} \), where \( x_i = [x_i^{(1)}, x_i^{(2)}, ..., x_i^{(k)}]^T \), \( i = 1, 2, ..., N \) the boundary set of \( k \) dimensional space \( B_{\text{min}} = [b_{\text{min}}^{(1)}, b_{\text{min}}^{(2)}, ..., b_{\text{min}}^{(k)}] \) and \( B_{\text{max}} = [b_{\text{max}}^{(1)}, b_{\text{max}}^{(2)}, ..., b_{\text{max}}^{(k)}] \), data ratio coefficient \( \gamma \)

**Output:** Integral data set and the corresponding volume set, non-integral data set.

1: let \( T_{\text{train}} = T \).
2: while \( N_{\text{non-int}} \leq \gamma \cdot N \) do
3: Apply algorithm 1 for set \( T_{\text{train}} \) to obtain the KD-tree set \( T' \) and the boundary value set for every corresponding volume.
4: Select the boundary values in set \( T' \) that exist in \( B_{\text{min}} \) and \( B_{\text{max}} \) as the non-integral data set \( T_{\text{non-int}} \) with the number \( N_{\text{non-int}} \), the rest construct the integral data set \( T_{\text{int}} \) and the corresponding volume set \( V_{\text{int}} \) with the number \( N_{\text{int}} \).
5: \( T_{\text{train}} = T \cup T_{\text{non-int}} \).
6: return Integral data set \( T_{\text{int}} \) and the corresponding volume set \( V_{\text{int}} \), non-integral data set \( T_{\text{non-int}} \).

**Remark 1** As a general rule, the entire integral space should be covered when computing integrals, but the deep KD-tree algorithm ignores the integral space at the boundary when dealing with integrals. In fact, the volume segmentation based on random points is not uniform, which leads to a small volume of segmentation at dense points and a large volume of sparse points near the boundary. Consequently, it is easy to fall into the local minimum value when training the NN. A possible scenario is that the normalization is satisfied locally on the boundary and zero at other points. In order to avoid this problem, it is better to directly omit the boundary part, and the integral contribution of the boundary part is very small, so it does not matter much. Meanwhile, although the deep KD-tree method cannot make the approximate integral result one, the analytic result obtained by the NN can be used for analytical processing. In order to make the final result standardized, the Vegas integral technology [14] can be used to complete the normalization.

### 2.3 Super-DL-FP algorithm

With the integral point data set, corresponding integral volume set and non-integral data set provided by deep KD-tree, it is easy to complete the training with the aid of DL-FP framework, then the super-DL-FP algorithm is proposed. Similar to the DL-FP algorithm, the most basic and common multi-layer perceptron (MLP) neural networks is still adopted as the NN. The NN is used to construct an experimental solution, and supervise the training from three perspectives. The trial solution should satisfy the form of the equation. The trial solution should meet the normalization condition, as well as fulfill the boundary condition. It should be noted that the PDF is nonnegative, it only needs to add a softPlus function to the last layer of the NN to ensure that the output is positive. However, different from the DL-FP algorithm, the form of the loss function is adjusted

\[
\mathcal{L} = \sum_{i=1}^{N_{\text{int}}+N_{\text{B}}} \left| \mathcal{L}_{\text{FP}} \left( \hat{p} \left( X_i^D | B | \theta_j \right) \right) \right|^2 + a_2 \cdot \sum_{i}^{N_{\text{int}}} \Delta v_i^D \cdot \hat{p} \left( X_i^D | \theta_j \right) - 1 \right|^2 + \sum_{i=1}^{N_{\text{B}}} \hat{p} \left( X_i^B | \theta_j \right) \right|^2,
\]

where \( \hat{p} \left( X_i | \theta_j \right) \) represents the trial solution constructed by NN with the parameters \( \theta_j \), and \( a_2 \) is the penalty factor. The integral point training set \( T_{\text{D}} = \{ X_i^D \}_{i=1}^{N_{\text{int}}} \) with the corresponding volume of the integral set \( V = \{ v_i^D \}_{i=1}^{N_{\text{int}}} \) is the integral data set in Algorithm 2. The boundary training set \( B_B = \{ X_i^B \}_{i=1}^{N_{\text{B}}} \) is the subset of non-integral data set \( T_{\text{non-int}} \) in Algorithm 2. If the number of \( T_{\text{non-int}} \) is small, the whole set \( T_{\text{non-int}} \) can be used as the boundary training set. Otherwise, a part of \( T_{\text{non-int}} \) extracted by uniform distribution can be adopted as the boundary training set in a certain proportion.

The supervision of equation and boundary is no longer in the form of MSE in \( \mathcal{L} \), but in the form of addition. The reason for this is to increase the
supervision of every value in the equation and boundary points when the number of training points is comparatively small, so as to ensure the correctness of every training point. Besides, different from the punishment used in DL-FP algorithm, the penalty factor is only added to the normalized condition. The deep KD-tree method is an approximate integral, so its supervision characteristics need to be weakened in the normalized part, and its appearance is more concerned with preventing the final training from falling into the trivial solution. At the same time, we also want to make the operation of penalty factor easier during training. The algorithm flow of super-DL-FP is shown in Algorithm 3.

### Algorithm 3 Super-DL-FP algorithm

**Input:** The integral point training set \( T_D = \{X^D_i\}_{i=1}^{N_D} \), the boundary training set \( B = \{X^B_i\}_{i=1}^{N_B} \), the volume of the integral set \( V = \{v^I_i\}_{i=1}^{N_I} \), the size of deep neural network (DNN).

**Output:** The Super-DL-FP solution.

1. Initialize the weights and biases of the neural network based on the DNN size.
2. for iteration \( j = 1, 2, ..., M \) do
3. Construct neural network only for training set and get the output \( \hat{p}(X|\theta_j) \).
4. Take the first and second derivatives of the output \( \hat{p}_X(X|\theta_j), \hat{p}_{XX}(X|\theta_j) \).
5. Construct the loss function.
6. Update the neural network by minimizing the loss function \( \mathcal{L} \).
7. Update the network: \( \theta_{j+1} = \theta_j + \delta \theta_j \).
8. return results

**Remark 2** For the training data, the Langevin equation and FP equation exist at the same time in most cases, and the steady-state random points generated by Eq. (1) can be used as experimental points. The method of Moment Networks [12] can be used to obtain approximate FP equation of neural network when only experimental data exists and FP equation is missing.

### 3 Numerical results

In this section, three numerical evidences are provided to support our conjectures and to demonstrate that our approach can be used successfully. In the first example, we will illustrate the advantage of super-DL-FP scheme in terms of computing points compared with other classical algorithms. Furthermore, we will explain how deep KD-tree works in practice and displays the correctness of the results through the second and the third examples.

#### 3.1 Example 1

Consider the Van der Pol system excited by Gaussian white noise as

\[
\begin{align*}
\dot{x} + \eta(-1 + x^2 + \dot{x}^2)\dot{x} + x &= \sigma \Gamma, \\
\dot{y} &= -\eta(-1 + x^2 + y^2)y - x + \sigma \Gamma.
\end{align*}
\]

Then, the corresponding stationary FP equation is

\[
\begin{align*}
-\frac{\partial}{\partial x} [yp(x, y)] - \frac{\partial}{\partial y} \left[ (-\eta(-1 + x^2 + y^2)y - x) p(x, y) \right] \\
+ \frac{\sigma^2}{2} \frac{\partial^2}{\partial y^2} p(x, y) &= 0,
\end{align*}
\]

and the exact steady-state solution is

\[
p_s(x, y) = C \cdot \exp \left[ -\frac{\eta}{\sigma^2} \left( x^2 + y^2 \right) - \frac{1}{2} \left( x^2 + y^2 \right)^2 \right],
\]

where \( C \) is the normalization constant. This system can be regarded as a two-dimensional system. The PDF solution will take on the shape of a crater with the parameters \( \eta = 1 \) and \( \sigma^2 = 0.3 \). The training data should be prepared first when applying the super-DL-FP. The solution of Eq. (9) obtained by Stochastic Runge–Kutta (SRK) II [11] method is taken as the training points instead of real experimental points in this example, and 100, 200 and 1000 position points are taken, respectively. For these three cases, the deep KD-tree method is used to obtain experimental points where the data ratio coefficient \( \gamma = 1 \). After obtaining the set of integral values, the corresponding set of volumes, and the non-integral data set, all the data is fed into the super-DL-FP framework. The penalty factor of normalized supervision is \( a_2 = 0.01 \). The structure of NN is constructed with three hidden layers and the number of nodes in each hidden layer is 20. The analytic results of the NN model can be obtained after training.
10,000 times, then the analytic model is normalized again by using Vegas integral algorithm. The reason for doing this is that NN has the advantage of being able to express the results analytically. At the same time, deep KD-tree is an approximate integral, which cannot guarantee the normalized result after training. In order to ensure that the final output PDF is standardized, the Vegas are used to normalize it. The results after training are shown in Fig. 2, and the results show that the accuracy of calculation is 84.21% under only 100 experimental points, 95.24% under 200 experimental points, and 99.30% under 1000 experimental points.

To illustrate the necessity of deep KD-tree method, the comparison between the data sets generated by deep KD-tree and KD-tree directly is analyzed, and the results are shown in Fig. 3. The integral contribution rate $R_{\text{int}}$ defined as

$$R_{\text{int}} = \frac{\sum_{i=1}^{N_{\text{int}}} p(x_i) \cdot v'_i}{1} \cdot 100\%,$$

Fig. 2 a and e The analytic solution for system (8) with $\eta = 1.0$, $\sigma^2 = 0.3$. b The super-DL-FP solution with 100 experimental data. c The super-DL-FP solution with 200 experimental data. d The super-DL-FP solution with 1000 experimental data. f–h The absolute error between analytic solution and super-DL-FP solution.

Fig. 3 Comparison of $R_{\text{int}}$ and $R_{\text{acc}}$ using KD-tree and deep KD-tree algorithm under different data amounts for system (9)
where \( p(\cdot) \) is the analytical PDF of Eq. (10), \( N_{\text{int}} \) represents the number of integral points and \( v'_i \) is the corresponding integral volume. To better analyze the effects of these two algorithms, we define the accuracy rate \( R_{\text{acc}} \) for quantitative analysis which reads

\[
R_{\text{acc}} = 1 - \frac{\| \hat{p} - p \|_2}{\| p \|_2},
\]

where \( \hat{p} \) is the approximate solution, i.e., the super-DL-FP solution, and \( p \) is the exact solution, and \( \| \cdot \|_2 \) denotes the \( L_2 \) norm. The closer \( R_{\text{acc}} \) is to 1, the higher the accuracy.

Two important conclusions can be drawn from the results. The first one is that the deep KD-tree method can greatly improve the integral contribution rate, as the deep KD-tree method can allow more experimental points to enter the integral point set and match appropriate integral volume for them. The second is that the accuracy of integration is positively correlated with the accuracy of training, which further illustrates the importance of ensuring the normalization conditions in solving FP equations. Normalization is an important factor in determining whether the PDF shape can be formed during training.

To further illustrate the value of this work, the results of super-DL-FP were compared with the central difference (CD), Chebyshev spectrum (CP), and Sylvester flow (SF) method [1]. Here, 200 calculation points are selected for different algorithms, as shown in Fig. 4, it can be obviously found that the result of the super-DL-FP scheme is more accurate and smooth. It should be emphasized here that the super-DL-FP obtains the solution in analytical form, which can be used for further analytical calculation, while the Chebyshev spectrum gains value at the point of calculation, which is difficult to be used for further analytical processing. Center difference and Chebyshev spectrum methods cannot guarantee that the solution is always positive, and there will be negative cases at the end of the PDF. Furthermore, the results of the comparison of the maximum absolute error when selecting 200 and 1000 calculation points are shown in Fig. 5. It can be found that at 200 points, the super-DL-FP algorithm has an error of a power of \( 10^{-2} \), while the other three schemes have an error of a power of \( 10^{-1} \). When the number of
calculation points is 1000, the super-DL-FP approach and Chebyshev spectrum method can reach the power order of $10^{-3}$, which is far ahead of the central difference scheme and flow approach.

3.2 Example 2

Consider a 4D system

$$
\begin{align*}
\dot{x}_1 &= x_3, \\
\dot{x}_2 &= x_4, \\
\dot{x}_3 &= -ax_3 - \frac{1}{M} \frac{\partial V}{\partial x_1} + \sigma_1 \Gamma, \\
\dot{x}_4 &= -bx_4 - \frac{1}{I} \frac{\partial V}{\partial x_2} + \sigma_2 \Gamma,
\end{align*}
$$

(14)

where $\Gamma$ represents a standard Gaussian white noise with the noise intensity $\sigma_1$ and $\sigma_2$, and $V = V(x_1, x_2) = k_1 x_1^2 + k_2 x_2^2 + (\lambda_1 x_1^4 + \lambda_2 x_2^4 + \mu x_1^2 x_2^2)$. The corresponding FP equation is

$$
\begin{align*}
- \frac{\partial (x_3 p)}{\partial x_1} - \frac{\partial (x_4 p)}{\partial x_2} - \frac{\partial \left( (-ax_3 - \frac{1}{M} \frac{\partial V}{\partial x_1} p) \right)}{\partial x_3} \\
- \frac{\partial \left( (-bx_4 - \frac{1}{I} \frac{\partial V}{\partial x_2} p) \right)}{\partial x_4} + \frac{\sigma_1^2}{2} \frac{\partial^2 p}{\partial x_3^2} + \frac{\sigma_2^2}{2} \frac{\partial^2 p}{\partial x_4^2} = 0.
\end{align*}
$$

(15)

The exact steady-state solution exists when $\sigma_1^2 M/a = \sigma_2^2 I/b = 2T$ and can be expressed as

$$
p_3(x_1, x_2, x_3, x_4) = C \cdot \exp \left[ -\frac{1}{2T} V(x_1, x_2) - \frac{a}{\sigma_1^2} x_3^2 - \frac{b}{\sigma_2^2} x_4^2 \right],
$$

(16)

where $C$ is the normalization constant. For system 15, the calculation interval is selected as $x_i \in [-5, 5], i = 1, 2, 3, 4$. The SRKII is still used to generate random points for Eq. (14) to simulate experimental data, and deep KD-tree algorithm is used to generate different training sets for these random points. In order to study the deep KD-tree, we force the iteration to loop 30 times instead of setting the data ratio coefficient. The results of the utilization rate which is the number of integral data divided by the number of original data, and integral contribution rate of integral points have been analyzed and calculated as shown in Fig. 6. It can be found that the deep KD-tree algorithm will reach the point where the utilization rate of integral points is 1 and converges after a finite number of iterations. Besides, there is still a positive correlation between the utilization rate of points and the contribution rate of points, which also confirms the conclusion of Example 1. As can be seen from the results in Fig. 6a, when the number of experimental points is 100,000, the integral contribution rate is basically close to 1, and the convergence is very stable.

![Fig. 6](image)

**Fig. 6 a** The integral contribution rate of different data sets increases with the depth of deep KD-tree for system (14). **b** The utilization rate of different data sets increases with the depth of deep KD-tree for system (14). The parameters are $a = 0.5, b = 1, k_1 = -0.5, \varepsilon = 0.5, \lambda_1 = 0.25, \lambda_2 = 0.125, \mu = 0.375, \sigma_1^2 = 2, \sigma_2^2 = 4, M = I = 1$.
However, we still hope to select a smaller number to complete the test. Here, 20,000 experimental points are selected, and the coefficient of data ratio coefficient is set as 1. At this time, 19,818 integral points and 8,559 non-integral points are obtained after 5 iterations, and the integral value of these points is 0.8. The training points are input into the super-DL-FP algorithm to obtain the analytical result with $a_2 = 0.01$. Similarly, the normalized value is calculated by using the Vegas integral. The normalized PDF is obtained and the results are shown in Fig. 7. Figure 7 shows the two-dimensional PDF, and it can be found that the error of the results is in the power of $10^{-4}$. In this example, we mainly want to emphasize the advantages of our method for the use of small amount of data. The data usage and error of the calculation results in this example are difficult to achieve by using the difference schemes.

To better analyze the setting rules of parameter $\gamma$ in deep KD-tree algorithm, we tested the relationship between $\gamma$ and $R_{\text{acc}}$ under 10,000 and 20,000 experimental points, respectively, the results are shown in Fig. 8. The accuracy of the results with standardization adopting the Vegas algorithm and the results without standardization are calculated. It can be seen from Fig. 8 that data ratio coefficient $\gamma$ is positively correlated with accuracy, which verifies the conclusion obtained in Example 1. Besides, the increase in orig-

![Fig. 7](image1.png)

**Fig. 7** (a) Analytical $p(x_1, x_2)$ for system (14). (b) The predicted $p(x_1, x_2)$ with super-DL-FP algorithm. (c) The absolute error between the analytical solution and the super-DL-FP solution. The parameters are $a = 0.5, b = 1, k_1 = -0.5, \epsilon = 0.5, \lambda_1 = 0.25, \lambda_2 = 0.125, \mu = 0.375, \sigma_1^2 = 2, \sigma_2^2 = 4, M = I = 1$

![Fig. 8](image2.png)

**Fig. 8** The relationship between $\gamma$ and $R_{\text{acc}}$ at 10,000 and 20,000 experimental points for system (14)
nal experimental data points has a positive effect on improving the accuracy of the results. It should be noted that the $R_{\text{acc}}$ will be negative when the KD-tree is only done once which can be seen in Fig. 8a. The reason for this phenomenon is that the predicted result is many times the real result. However, it also shows the necessity of adopting the deep KD-tree algorithm. As $\gamma$ approaches 1, the accuracy is also maximized. There will be less training data in solving FP equations when $\gamma$ is small, which leads to faster training speed when using neural network. Therefore, adjusting $\gamma$ is a trade-off problem in accuracy and training time. In the actual operation of the deep KD-tree, we suggest setting $\gamma$ close to 1 when the amount of data is not very large, which can achieve relatively good results.

3.3 Example 3

In this example, a 6D systems is considered which is

$$
\begin{align*}
\dot{x}_1 &= y_1, \\
\dot{y}_1 &= -y_1 - \frac{\partial V}{\partial x_1} + \sigma_1 \Gamma, \\
\dot{x}_2 &= y_2, \\
\dot{y}_2 &= -y_2 - \frac{\partial V}{\partial x_2} + \sigma_2 \Gamma, \\
\dot{x}_3 &= y_3, \\
\dot{y}_3 &= -y_3 - \frac{\partial V}{\partial x_3} + \sigma_3 \Gamma,
\end{align*}
$$

(17)

where $\Gamma$ represents a standard Gaussian white noise with the noise intensity $\sigma_1$, $\sigma_2$, $\sigma_3$, and $V = V(x_1, x_2, x_3) = \frac{1}{2}(x_1^2 + x_2^2 + x_3^2) + \frac{1}{2}(x_1 x_2 + x_1 x_3)$, and the corresponding stationary FP equation is

$$
\begin{align*}
\frac{\partial}{\partial x_1} \left( -y_1 - \frac{\partial V}{\partial x_1} \right) &= 0, \\
\frac{\partial}{\partial x_2} \left( -y_2 - \frac{\partial V}{\partial x_2} \right) &= 0, \\
\frac{\partial}{\partial x_3} \left( -y_3 - \frac{\partial V}{\partial x_3} \right) &= 0.
\end{align*}
$$

Similarly, when $\sigma_1^2 = \sigma_2^2 = \sigma_3^2 = 4T$, Eq. (18) owns the exact stationary solution

$$
\begin{align*}
p_s(x_1, y_1, x_2, y_2, x_3, y_3) &= C \cdot \exp \left[ -\frac{1}{2T} V(x_1, x_2, x_3) - \frac{1}{\sigma_1^2} y_1^2 \\
&\quad - \frac{1}{\sigma_2^2} y_2^2 - \frac{1}{\sigma_3^2} y_3^2 \right].
\end{align*}
$$

(19)

where $C$ is the normalization constant. When solving Eq. (18), the first step is still to obtain the training points which can be done by applying the SRKII method for Eq. (17). In order to further analyze and verify the performance of deep KD-tree, the data ratio coefficient is still not set. Instead, the iteration cycle is forced for 30 times. The results are shown in Fig. 9, which proves once again that the utilization rate of points is close to 100% and converges after reaching a certain number of cycles. Therefore, it can be found from the analysis of Example 3 and Example 2 that in the actual use of deep KD-tree, it is reasonable to set a data ratio coefficient. At the same time, the results in Fig. 9 again show

Fig. 9  a The integral contribution rate of different data sets increases with the depth of deep KD-tree for system (17). b The utilization rate of different data sets increases with the depth of deep KD-tree for system (17). The parameters are $\sigma_1^2 = \sigma_2^2 = \sigma_3^2 = 4$
that the utilization rate of integral points is positively correlated with the contribution rate of integral points. When the utilization rate of integral points converges, the contribution rate of integral points also converges. The contribution rate of integral points will increase with the increase of the number of experimental points.

Here, we selected 200,000 basic experimental data sets. 175,000 integral data points and 500,000 non-integral points can be generated after 5 times of deep KD-tree. At this point, it can be found that the number of non-integral points is too large. In this case, a specific gravity can be set and a certain number of non-integral points can be randomly selected according to uniform distribution as the final boundary set to be put into super-DL-FP. In this example, 20,000 boundary points are selected. The results in Fig. 10 show the comparison between the conditional PDF calculated using the super-DL-FP and the analytical results. It can be found that the error of the results is relatively small. In particular, the data usage of our approach is very competitive compared with other methods. It still needs to be emphasized that this algorithm can acquire the analytical results, which can be obtained at any point and can also be used for subsequent analytical processing.

4 Discussions and conclusions

This paper is mainly concerned about how to solve the FP equation with a small amount of data instead of the whole discrete points. A deep KD-tree method is proposed to estimate the normalization condition in FP equation with random points. The basic idea is to divide the mesh based on random points by an iterative KD-tree algorithm to get the integral volume partition of random points matching. The super-DL-FP algorithm which can solve the FP equations with a small amount of data is proposed through combining the deep KD-tree with the DL-FP framework and its correctness is proved by three numerical examples. We found that: (1) The deep KD-tree method can improve the accuracy of normalized integral estimation, and thus improve the accuracy of training results. (2) The deep KD-tree method can converge after a certain number of iterations. (3) Compared with the traditional difference method and normalized flow approach, the super-DL-FP algorithm has a high precision in solving FP equation with a small amount of data. (4) The super-DL-FP framework obtains an analytical solution, which can provide a good foundation for subsequent analytical analysis. Compared with the other NN-based methods for solving differential equation problems, this paper focuses more on how to fully utilize and process data without any increase in any prior condition. We believe that the approach presented in this paper can provide a data processing paradigm for other NN approaches to differential equation problems.

The main contribution of this paper is the deep KD-tree algorithm, which can be used to estimate normalized integrals with a small amount of data instead of the total amount. We believe that this method can also provide an idea for other integral–differential equations using the NN method. However, the deep KD-tree approach is only illustrated from the perspective of visualization, without theoretical proof, which is a place that needs to be further improved. Here, we only

Fig. 10  a Analytical conditional PDF $p(x_1, x_2|y_1 = x_3 = y_2 = y_3 = 0.6)$ for system (17). b the predicted $p(x_1, x_2|y_1 = x_3 = y_2 = y_3 = 0.6)$ super-DL-FP solution. c The absolute error between the analytical solution and the super-DL-FP solution. The parameters are $\sigma_1^2 = \sigma_2^2 = \sigma_3^2 = 4$
focus on FP equations with natural boundaries, and give the corresponding deep KD-tree algorithm, the applicability of deep KD-tree algorithm to FP equations with absorption and reflection boundaries is also the work that needs to be considered in the future. At the same time, we have not provided a mathematical expression for the data ratio coefficient. Therefore, the actual use of deep KD-tree requires experiments to determine the specific number of cycles. Based on our experience, it is better to set the data ratio coefficient \( \gamma \) close to 1. In terms of the scheme of integral estimation, this paper adopts the more conservative midpoint integral, there may be more appropriate integrals, such as quantile integrals, which also require further study.

Acknowledgements This work was partly supported by the NSF of China (No. 12072264), the Key International (Regional) Joint Research Program of the NSF of China (No. 12120101002), the Research Funds for Interdisciplinary Subject of NPU, and the Shaanxi Provincial Key R&D Program (Grant Nos. 2020KW-013 and 2019TD-010).

Data availability Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Berg, R.v.d., Hasenclever, L., Tomczak, J.M., Welling, M.: Sylvester normalizing flows for variational inference. arXiv preprint arXiv:1803.05649 (2018)
2. Brown, R.A.: Building a balanced k-d tree in \( O(kn \log n) \) time. J. Comput. Graph. Tech. 4(1), (2015)
3. Ceccato, A., Frezzato, D.: Remarks on the chemical Fokker–Planck and Langevin equations: nonphysical currents at equilibrium. J. Chem. Phys. 148, 064114 (2018)
4. Chen, X., Yang, L., Duan, J., Karniadakis, G.E.: Solving inverse stochastic problems from discrete particle observations using the Fokker–Planck equation and physics-informed neural networks. SIAM J. Sci. Comput. 43(3), B811–B830 (2021)
5. Drozdov, A.N.: Accurate path integral representations of the Fokker–Planck equation with a linear reference system: comparative study of current theories. Phys. Rev. E 57, 146–158 (1998)
6. Escobedo, M., Herrero, M., Velazquez, J.: A nonlinear Fokker–Planck equation modelling the approach to thermal equilibrium in a homogeneous plasma. Trans. Am. Math. Soc. 350(10), 3837–3901 (1998)
7. Foster, T., Lei, C.L., Robinson, M., Gavaghan, D., Lambert, B.: Model evidence with fast tree based quadrature. Stat 1050, 22 (2020)
8. Furioli, G., Pulvirenti, A., Terraneo, E., Toscani, G.: Fokker–Planck equations in the modeling of socio-economic phenomena. Math. Models Methods Appl. Sci. 27, 44 (2016)
9. Galán, R.F., G Bard, E., Urban, N.N.: Stochastic dynamics of uncoupled neural oscillators: Fokker-Planck studies with the finite element method. Phys. Rev. E 76, 056110 (2007)
10. Han, J., Jentzen, A., Weinan, E.: Solving high-dimensional partial differential equations using deep learning. Proc. Natl. Acad. Sci. 115, 8505–8510 (2018)
11. Honeycutt, R.L.: Stochastic Runge-Kutta algorithms. i. white noise. Phys. Rev. A 45, 600 (1992)
12. Jeffrey, N., Wandelt, B.D.: Solving high-dimensional parameter inference: marginal posterior densities & moment networks. arXiv preprint arXiv:2011.05991 (2020)
13. Jiang, Y.: A new analysis of stability and convergence for finite difference schemes solving the time fractional Fokker-Planck equation. Appl. Math. Model. 39, 1163–1171 (2015)
14. Lepage, G.P.: Adaptive multidimensional integration: vegas enhanced. J. Comput. Phys. 439, 110386 (2021)
15. Li, J., Chen, J., Li, B.: Gradient-optimized physics-informed neural networks (GOPINNs): a deep learning method for solving the complex modified KdV equation. Nonlinear Dyn. pp. 1–12 (2021)
16. Ma, J., Xu, Y., Li, Y., Tian, R., Chen, G., Kurths, J.: Precursor criteria for noise-induced critical transitions in multi-stable systems. Nonlinear Dyn. 101(1), 21–35 (2020)
17. Náprstek, J., Král, R.: Finite element method analysis of uncoupled neural oscillators: Fokker-Planck studies with the finite element method. Appl. Math. Comput. 262, 187–190 (2014)
18. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. 378, 686–707 (2019)
19. Risken, H.: Fokker-Planck equation. In: The Fokker-Planck Equation, pp. 63–95. Springer (1996)
20. Sepehrian, B., Radpoor, M.K.: Numerical solution of non-linear Fokker-Planck equation using finite differences method and the cubic spline functions. Appl. Math. Comput. 262, 187–190 (2015)
21. Sirignano, J., Spiliopoulos, K.: DGM: a deep learning algorithm for solving partial differential equations. J. Comput. Phys. 375, 1339–1364 (2018)
22. Wang, Z., Xu, Y., Li, Y., Kurths, J.: \( \alpha \)-stable noise-induced coherence on a spatially extended Fitzhugh Nagumo system. J. Stat. Mech. Theory Exp. 2019(10), 103501 (2019)
23. Xiao, L.: A finite-time recurrent neural network for solving online time-varying Sylvester matrix equation based on a new evolution formula. Nonlinear Dyn. 90(3), 1581–1591 (2017)
24. Xu, Y., Li, Y., Zhang, H., Li, X., Kurths, J.: The switch in a genetic toggle system with Lévy noise. Sci. Rep. 6(1), 31505–31505 (2016)
25. Xu, Y., Liu, Q., Guo, G., Xu, C., Liu, D.: Dynamical responses of airfoil models with harmonic excitation under uncertain disturbance. Nonlinear Dyn. 89(3), 1579–1590 (2017)
26. Xu, Y., Zan, W., Jia, W., Kurths, J.: Path integral solutions of the governing equation of SDEs excited by Lévy white noise. J. Comput. Phys. 394, 41–55 (2019)
27. Xu, Y., Zhang, H., Li, Y., Zhou, K., Liu, Q., Kurths, J.: Solving Fokker-Planck equation using deep learning. Chaos 30(1), 013133 (2020)
28. Zhai, J., Dobson, M., Li, Y.: A deep learning method for solving Fokker-Planck equations. arXiv preprint arXiv:2012.10696 (2020)
29. Zhang, H., Xu, Y., Li, Y., Kurths, J.: Statistical solution to SDEs with α-stable Lévy noise via deep neural network. Int. J. Dyn. Control 8(4), 1129–1140 (2020)
30. Zhang, R.F., Bilige, S.: Bilinear neural network method to obtain the exact analytical solutions of nonlinear partial differential equations and its application to p-gBKP equation. Nonlinear Dyn. 95(4), 3041–3048 (2019)
31. Zhang, X., Xu, Y., Liu, Q., Kurths, J., Grebogi, C.: Rate-dependent bifurcation dodging in a thermoacoustic system driven by colored noise. Nonlinear Dyn. 104(3), 2733–2743 (2021)

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.