frontiers in Artificial Intelligence

ORIGINAL RESEARCH
published: 02 August 2021
doi: 10.3389/frai.2021.668215

Solution of the Fokker–Planck Equation by Cross Approximation Method in the Tensor Train Format

Andrei Chertkov and Ivan Oseledets*

Skolkovo Institute of Science and Technology, Moscow, Russia

We propose the novel numerical scheme for solution of the multidimensional Fokker–Planck equation, which is based on the Chebyshev interpolation and the spectral differentiation techniques as well as low rank tensor approximations, namely, the tensor train decomposition and the multidimensional cross approximation method, which in combination makes it possible to drastically reduce the number of degrees of freedom required to maintain accuracy as dimensionality increases. We demonstrate the effectiveness of the proposed approach on a number of multidimensional problems, including Ornstein-Uhlenbeck process and the dumbbell model. The developed computationally efficient solver can be used in a wide range of practically significant problems, including density estimation in machine learning applications.

Keywords: fokker-planck equation, probability density function, tensor train format, cross approximation, chebyshev polynomial, ornstein-uhlenbeck process, dumbbell model

OPEN ACCESS

Edited by:
Evangelos Papalexakis, University of California, Riverside, United States

Reviewed by:
Devin Matthews, Southern Methodist University, United States
Prakash Vedula, University of Oklahoma, United States

****Correspondence:***
Andrei Chertkov
a.chertkov@skoltech.ru

Specialty section:
This article was submitted to Machine Learning and Artificial Intelligence, a section of the journal Frontiers in Artificial Intelligence

Received: *15 February 2021*
Accepted: *29 June 2021*
Published: *02 August 2021*

Citation:
Chertkov A and Oseledets I (2021) Solution of the Fokker–Planck Equation by Cross Approximation Method in the Tensor Train Format. Front. Artif. Intell. 4:668215. doi: 10.3389/frai.2021.668215

1 INTRODUCTION

Fokker–Planck equation (FPE) is an important in studying properties of the dynamical systems, and has attracted a lot of attention in different fields. In recent years, FPE has become widespread in the machine learning community in the context of the important problems of density estimation (Grathwohl et al., 2018) for neural ordinary differential equation (ODE) (Chen et al., 2018; Chen and Duvenaud, 2019), generative models (Kidger et al., 2021), etc.

Consider a stochastic dynamical system which is described by stochastic differential equation (SDE) of the form[1]

$$d\boldsymbol{x} = \mathbf{f}(\boldsymbol{x}, t)\, dt + S(\boldsymbol{x}, t)\, d\boldsymbol{\beta}, \quad d\boldsymbol{\beta}\, d\boldsymbol{\beta}^{\top} = Q(t)\, dt, \quad \boldsymbol{x} = \boldsymbol{x}(t) \in \mathbb{R}^d, \tag{1}$$

where $d\boldsymbol{\beta}$ is a q-dimensional space-time white noise, $\mathbf{f}$ is a known d-dimensional vector-function and $S \in \mathbb{R}^{d\times q}$, $Q \in \mathbb{R}^{q\times q}$ are known matrices. The FPE for the corresponding probability density function (PDF) $\rho(\boldsymbol{x}, t)$ of the spatial variable $\boldsymbol{x}$ has the form

$$\frac{\partial \rho(\boldsymbol{x}, t)}{\partial t} = \sum_{i=1}^{d}\sum_{j=1}^{d} \frac{\partial}{\partial \boldsymbol{x}_i}\frac{\partial}{\partial \boldsymbol{x}_j}\left[D_{ij}(\boldsymbol{x}, t)\rho(\boldsymbol{x}, t)\right] - \sum_{i=1}^{d}\frac{\partial}{\partial \boldsymbol{x}_i}\left[\mathbf{f}_i(\boldsymbol{x}, t)\rho(\boldsymbol{x}, t)\right], \tag{2}$$

[1]Vectors and matrices are denoted hereinafter by lower case bold letters ($\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \ldots$) and upper case letters ($A, B, C, \ldots$) respectively. We denote the (i_1, i_2) th element of an $N_1 \times N_2$ matrix A as $A[i_1, i_2]$ and assume that $1 \le i_1 \le N_1$, $1 \le i_2 \le N_2$. For vectors we use the same notation: $\boldsymbol{a}[i]$ is the i-th element of the vector $\boldsymbol{a}$ ($i = 1, 2, \ldots, N$). In addition, for a compact representation of an i-th ($i = 1, 2, \ldots, d$, where $d \ge 1$) element of a vector function $\mathbf{f} = [f_1, f_2, \ldots, f_d]^{\top}$, we will use the notation $\mathbf{f}_i$, which means $\mathbf{f}_i(\,\cdot\,) = \mathbf{f}_i(\,\cdot\,)[i]$.

where $D(\boldsymbol{x},t)=\frac{1}{2}S(\boldsymbol{x},t)Q(t)S^{\top}(\boldsymbol{x},t)$ is a diffusion tensor.

One of the major complications in solution of the FPE is the high dimensionality of the practically significant computational problems. Complexity of using grid-based representation of the solution grows exponentially with d, thus some low-parametric representations are required. One of the promising directions is the usage of low-rank tensor methods, studied in (Dolgov et al., 2012). The equation is discretized on a tensor-product grid, such that the solution is represented as a d-dimensional tensor, and this tensor is approximated in the low-rank tensor train format (TT-format) (Oseledets, 2011). Even with such complexity reduction, the computations often take a long time. In this paper we propose another approach of using low-rank tensor methods for the solution of the FPE, based on its intimate connection to the dynamical systems.

The key idea can be illustrated for $S=0$, i.e. in the deterministic case. For this case the evolution of the PDF along the trajectory is given by the formula

$$\frac{\partial\rho(\boldsymbol{x},t)}{\partial t}=-\mathrm{Tr}\left(\frac{\partial\mathbf{f}(\boldsymbol{x},t)}{\partial\boldsymbol{x}}\right)\rho(\boldsymbol{x},t), \tag{3}$$

where $\mathrm{Tr}(\cdot)$ is a trace operation for the matrix. Hence, to compute the value of $\rho(\boldsymbol{x},t)$ at the specific point $\boldsymbol{x}=\widehat{\boldsymbol{x}}$, it is sufficient to find a preimage $\widehat{\boldsymbol{x}}_0$ such that if it is used as an initial condition for **eq. 1**, then we arrive to $\widehat{\boldsymbol{x}}$. To find the preimage, we need to integrate the **eq. 1** backwards in time, and then to find the PDF value, we integrate a system of **eqs 1**, **3**. Since we can evaluate the value of $\rho(\boldsymbol{x},t)$ at any $\widehat{\boldsymbol{x}}$, we can use the cross approximation method (CAM) (Oseledets and Tyrtyshnikov, 2010; Savostyanov and Oseledets, 2011; Dolgov and Savostyanov, 2020) in the TT-format to recover a supposedly low-rank tensor from its samples. In this way we do not need to have any compact representation of $\mathbf{f}$, but only numerically solve the corresponding ODE. For $S\neq 0$ the situation is more complicated, but we develop a splitting and multidimensional interpolation schemes that allow us effectively recompute the values of the density from some time moment t to the next step $t+h$.

To summarize, main contributions of our paper are the following:

- we derive a formula to recompute the values of the PDF on each time step, using the second order operator splitting, Chebyshev interpolation and spectral differentiation techniques;
- we propose to use a TT-format and CAM to approximate the solution of the FPE which makes it possible to drastically reduce the number of degrees of freedom required to maintain accuracy as dimensionality increases;
- we implement FPE solver, based on the proposed approach, as a publicly available python code[2], and we test our approach on several examples, including multidimensional Ornstein-Uhlenbeck process and dumbbell model, which demonstrate its efficiency and robustness.

2 COMPUTATION OF THE PROBABILITY DENSITY FUNCTION

For ease of demonstration of the proposed approach, we suppose that the noise $\boldsymbol{\beta}\in\mathbb{R}^q$ has the same dimension as the spatial variable $\boldsymbol{x}\in\mathbb{R}^d$ ($q=d$), and the matrices in **eq. 1** and **eq. 2** have the form[3]

$$Q(t)\equiv I_d,\quad S(\boldsymbol{x},t)\equiv\sqrt{2\chi}I_d,\quad D(\boldsymbol{x},t)\equiv\chi I_d, \tag{4}$$

where $\chi\geq 0$ is a scalar diffusion coefficient. Then **eqs 1**, **2** can be rewritten in a more compact form

$$d\boldsymbol{x}=\mathbf{f}(\boldsymbol{x},t)\,dt+\sqrt{2\chi}d\boldsymbol{\beta},\quad d\boldsymbol{\beta}\,d\boldsymbol{\beta}^{\top}=I_d\,dt, \tag{5}$$

$$\frac{\partial\rho}{\partial t}=\chi\Delta\rho-\mathrm{div}\left[\mathbf{f}(\boldsymbol{x},t)\rho\right], \tag{6}$$

where d-dimensional spatial variable $\boldsymbol{x}=\boldsymbol{x}(t)\in\Omega\subset\mathbb{R}^d$ has the corresponding PDF $\rho(\boldsymbol{x},t)$ with initial conditions

$$\boldsymbol{x}(0)=\boldsymbol{x}_0\sim\rho(\boldsymbol{x},0),\quad \rho(\boldsymbol{x},0)=\rho_0(\boldsymbol{x}). \tag{7}$$

To construct the PDF at some moment τ ($\tau>0$) for the known initial distribution $\rho_0(\boldsymbol{x})$, we discretize **eqs 5**, **6** on the uniform time grid with M ($M\geq 2$) points

$$t_m=mh,\quad h=\frac{\tau}{M-1},\quad m=0,1,\ldots,M-1, \tag{8}$$

and introduce the notation $\boldsymbol{x}_m=\boldsymbol{x}(t_m)$ for value of the spatial variable at the moment t_m and $\rho_m(\cdot)=\rho(\cdot,t_m)$ for values of the PDF at the same moment.

2.1 Splitting Scheme

Let $\widehat{V}$ and $\widehat{W}$ be diffusion and convection operators from the **eq. 6**

$$\widehat{V}v\equiv\chi\Delta v,\quad \widehat{W}w\equiv-\mathrm{div}\left[\mathbf{f}(\boldsymbol{x},t)w\right], \tag{9}$$

then on each time step m ($m=0,1,\ldots,M-2$) we can integrate equation

$$\frac{\partial\rho}{\partial t}=\left(\widehat{V}+\widehat{W}\right)\rho,\quad \rho(\cdot,t_m)=\rho_m(\cdot), \tag{10}$$

on the interval (t_m,t_m+h), to find ρ_{m+1} for the known value ρ_m from the previous time step. Its solution can be represented in the form of the product of an initial solution with the matrix exponential

$$\rho_{m+1}=e^{h\left(\widehat{V}+\widehat{W}\right)}\rho_m, \tag{11}$$

and if we apply the standard second order operator splitting technique (Glowinski et al., 2017), then

$$\rho_{m+1}\approx e^{\frac{h}{2}\widehat{V}}e^{h\widehat{W}}e^{\frac{h}{2}\widehat{V}}\rho_m, \tag{12}$$

which is equivalent to the sequential solution of the following equations

[2]The code is publicly available from https://github.com/AndreiChertkov/fpcross.

[3]We use notation I_k for the $k\times k$ ($k=1,2,\ldots$) identity matrix.

$$\frac{\partial v^{(1)}}{\partial t} = \chi \Delta v^{(1)}, \quad v^{(1)}(\,\cdot\,, t_m) = \rho_m(\,\cdot\,), \tag{13}$$

$$\frac{\partial w}{\partial t} = -\mathrm{div}\left[\mathbf{f}(\boldsymbol{x}, t) w\right], \quad w(\,\cdot\,, t_m) = v^{(1)}\left(\,\cdot\,, t_m + \frac{h}{2}\right), \tag{14}$$

$$\frac{\partial v^{(2)}}{\partial t} = \chi \Delta v^{(2)}, \quad v^{(2)}(\,\cdot\,, t_m) = w(\,\cdot\,, t_m + h), \tag{15}$$

with the final approximation of the solution $\rho_{m+1}(\,\cdot\,) = v^{(2)}(\,\cdot\,, t_m + \frac{h}{2})$.

2.2 Interpolation of the Solution

To efficiently solve the convection **eq. 14**, we need the ability to calculate the solution of the diffusion **eq. 13** at arbitrary spatial points, hence the natural choice for the discretization in the spatial domain are Chebyshev nodes, which makes it possible to interpolate the corresponding function on each time step by the Chebyshev polynomials (Trefethen, 2000).

We introduce the d-dimensional spatial grid $\mathbb{X}^{(g)}$ as a tensor product of the one-dimensional grids[4]

$$\boldsymbol{x}_k^{(g)} \in \mathbb{R}^{N_k}, \quad \boldsymbol{x}_k^{(g)}[n_k] = \cos\frac{\pi \cdot (n_k - 1)}{N_k - 1}, \quad n_k = 1, 2, \ldots, N_k, \tag{16}$$

where N_k ($N_k \geq 2$) is a number of points along the kth spatial axis ($k = 1, 2, \ldots, d$), and the total number of the grid points is $N = N_1 \cdot N_2 \cdot \ldots \cdot N_d$. Note that this grid can be also represented in the flatten form as a following matrix

$$X^{(g)} \in \mathbb{R}^{d \times N}, \quad X^{(g)}[k, n] = \boldsymbol{x}_k^{(g)}[\boldsymbol{mind}(n)[k]], \tag{17}$$

where $n = 1, 2, \ldots, N$, $k = 1, 2, \ldots, d$ and by $\boldsymbol{mind}(n) = [\boldsymbol{n_1}, \boldsymbol{n_2}, \ldots, \boldsymbol{n_d}]^\top$ we denoted an operation of construction of the multi-index from the flatten long index according to the big-endian convention

$$n = n_d + (n_{d-1} - 1) N_d + \ldots + (n_1 - 1) N_2 N_3 \ldots N_d. \tag{18}$$

Suppose that we calculated PDF ρ_m on some time step m ($m \geq 0$) at the nodes of the spatial grid $\mathbb{X}^{(g)}$ [note that for the case $m = 0$, the corresponding values come from the known initial condition $\rho_0(\boldsymbol{x})$]. These values can be collected as elements of a tensor[5] $\mathcal{R}_m \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_d}$ such that

$$\mathcal{R}_m[n_1, n_2, \ldots, n_d] = \rho_m\left(\boldsymbol{x}_1^{(g)}[n_1], \boldsymbol{x}_2^{(g)}[n_2], \ldots, \boldsymbol{x}_d^{(g)}[n_d]\right), \tag{19}$$

where $n_k = 1, 2, \ldots, N_k$ ($k = 1, 2, \ldots, d$).

Let us interpolate PDF ρ_m via the system of orthogonal Chebyshev polynomials of the first kind

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x) \text{ for } k = 1, 2, \ldots, \tag{20}$$

in the form of the naturally cropped sum

$$\rho_m(\boldsymbol{x}) \approx \widetilde{\rho_m}(\boldsymbol{x}) = \sum_{n_1=1}^{N_1} \sum_{n_2=1}^{N_2} \cdots \sum_{n_d=1}^{N_d} \mathcal{A}_m[n_1, n_2, \ldots, n_d] T_{n_1-1}(x_1) T_{n_2-1}(x_2) \ldots T_{n_d-1}(x_d), \tag{21}$$

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$ is some spatial point and interpolation coefficients are elements of the tensor $\mathcal{A}_m \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_d}$. For construction of this tensor we should set equality in the interpolation nodes **eq. 16**

$$\widetilde{\rho_m}\left(\boldsymbol{x}_1^{(g)}[n_1], \boldsymbol{x}_2^{(g)}[n_2], \ldots, \boldsymbol{x}_d^{(g)}[n_d]\right) = \rho_m\left(\boldsymbol{x}_1^{(g)}[n_1], \boldsymbol{x}_2^{(g)}[n_2], \ldots, \boldsymbol{x}_d^{(g)}[n_d]\right) \tag{22}$$

for all combinations of $n_k = 1, 2, \ldots, N_k$ ($k = 1, 2, \ldots, d$).

Therefore the interpolation process can be represented as a transformation of the tensor $\mathcal{R}_m$ to the tensor $\mathcal{A}_m$ according to the system of **eq. 22**. If the Chebyshev polynomials and nodes are used for interpolation, then a good way is to apply a fast Fourier transform (FFT) (Trefethen, 2000) for this transformation. However the exponential growth of computational complexity and memory consumption with the growth of the number of spatial dimensions makes it impossible to calculate and store related tensors for the multidimensional case in the dense data format. Hence in the next sections we present an efficient algorithm for construction of the tensor $\mathcal{A}_m$ in the low-rank TT-format.

2.3 Solution of the Diffusion Equation

To solve the diffusion **eqs 13**, **15** on the Chebyshev grid, we discretize Laplace operator using the second order Chebyshev differential matrices [see, for example, (Trefethen, 2000)] $D_k \in \mathbb{R}^{N_k \times N_k}$ such that $D_k = \tilde{D}_k \tilde{D}_k$, where for each spatial dimension $k = 1, 2, \ldots, d$

$$\tilde{D}_k[i, j] = \begin{cases} \dfrac{2(N_k - 1)^2 + 1}{6}, & i = j = 1, \\ \dfrac{-\boldsymbol{x}_k^{(g)}[j]}{2\left(1 - \left(\boldsymbol{x}_k^{(g)}[j]\right)^2\right)}, & i = j = 2, 3, \ldots, N_k - 1, \\ \dfrac{c_i}{c_j} \dfrac{(-1)^{i+j}}{\boldsymbol{x}_k^{(g)}[i] - \boldsymbol{x}_k^{(g)}[j]}, & i \neq j, \quad i, j = 2, 3, \ldots, N_k - 1, \\ -\dfrac{2(N_k - 1)^2 + 1}{6}, & i = j = N_k, \end{cases} \tag{23}$$

with $c_i = 2$ if $i = 1$ or $i = N_k$ and $c_i = 1$ otherwise, and one dimensional grid points $\boldsymbol{x}_k^{(g)}$ defined from **eq. 16**. Then discretized Laplace operator has the form[6]

[4] We suppose that for each spatial dimension the variable x varies within $[-1, 1]$. In other cases, an appropriate scaling can be easily applied.

[5] By tensors we mean multidimensional arrays with a number of dimensions d ($d \geq 1$). A two-dimensional tensor ($d = 2$) is a matrix, and when $d = 1$ it is a vector. For tensors with $d > 2$ we use upper case calligraphic letters ($\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots$). The $(n_1, n_2, \ldots, n_d)$ th entry of a d-dimensional tensor $\mathcal{A} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_d}$ is denoted by $\mathcal{A}[n_1, n_2, \ldots, n_d]$, where $n_k = 1, 2, \ldots, N_k$ ($k = 1, 2, \ldots, d$) and N_k is a size of the k-th mode, and mode-k slice of such tensor is denoted by $\mathcal{A}[n_1, \ldots, n_{k-1}, :, n_{k+1}, \ldots, n_d]$.

[6] Note that for the case $N_1 = N_2 = \ldots = N_d \equiv N_0$, we have only one matrix $D_1 = D_2 = \ldots = D_d \equiv D_0 \in \mathbb{R}^{N_0 \times N_0}$ which greatly simplifies the computation process.

$$\Delta = D_1 \otimes I_{N_2} \otimes \ldots \otimes I_{N_d} + I_{N_1} \otimes D_2 \otimes \ldots \otimes I_{N_d} + \ldots + I_{N_1} \otimes I_{N_2} \otimes \ldots \otimes D_d. \tag{24}$$

Let $\mathcal{V}_m \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_d}$ be the known initial condition for the diffusion equation on the time step m ($t_m = mh$), then for the solution $\mathcal{V}_{m+\frac{1}{2}}$ at the moment $t_m + \frac{h}{2}$ we have

$$\boldsymbol{vec}\left(\mathcal{V}_{m+\frac{1}{2}}\right) = e^{\frac{h}{2}\chi\Delta}\boldsymbol{vec}\left(\mathcal{V}_m\right), \tag{25}$$

where an operation $\boldsymbol{vec}(\cdot)$ constructs a vector from the tensor by a standard reshaping procedure like **eq. 18**. And finally due to the well known property of the matrix exponential, we come to

$$\boldsymbol{vec}\left(\mathcal{V}_{m+\frac{1}{2}}\right) = \left(e^{\frac{h}{2}\chi D_1} \otimes e^{\frac{h}{2}\chi D_2} \otimes \ldots \otimes e^{\frac{h}{2}\chi D_d}\right)\boldsymbol{vec}\left(\mathcal{V}_m\right). \tag{26}$$

If we can represent the initial condition $\mathcal{V}_m$ in the form of Kronecker product of the one-dimensional tensors (for example, in terms of the TT-format in the form of the Kronecker products of the TT-cores, as will be presented below in this work), then we can efficiently evaluate the formula **eq. 26** to obtain the desired approximation for solution $\boldsymbol{vec}\left(\mathcal{V}_{m+\frac{1}{2}}\right)$.

2.4 Solution of the Convection Equation

Convection **eq. 14** can be reformulated in terms of the FPE without diffusion part, when the corresponding ODE has the form

$$d\boldsymbol{x} = \mathbf{f}(\boldsymbol{x}, t)\, dt, \quad \boldsymbol{x} = \boldsymbol{x}(t) \in \mathbb{R}^d, \quad \boldsymbol{x} \sim \rho(\boldsymbol{x}, t). \tag{27}$$

If we consider the differentiation along the trajectory of the particles, as was briefly described in the Introduction, then

$$\left(\frac{\partial w}{\partial t}\right)_{\boldsymbol{x}=\boldsymbol{x}(t)} = \sum_{k=1}^{d} \frac{\partial w}{\partial \boldsymbol{x}_k}\frac{\partial \boldsymbol{x}_k}{\partial t} + \frac{\partial w}{\partial t} = \sum_{k=1}^{d} \frac{\partial w}{\partial \boldsymbol{x}_k}\frac{\partial \boldsymbol{x}_k}{\partial t} - \operatorname{div}[\mathbf{f} w] = = \sum_{k=1}^{d} \frac{\partial w}{\partial \boldsymbol{x}_k}\mathbf{f}_k - \sum_{k=1}^{d} \frac{\partial \mathbf{f}_k}{\partial \boldsymbol{x}_k} w - \sum_{k=1}^{d} \mathbf{f}_k \frac{\partial w}{\partial \boldsymbol{x}_k} = -\sum_{k=1}^{d} \frac{\partial \mathbf{f}_k}{\partial \boldsymbol{x}_k} w, \tag{28}$$

where we replaced the term $\frac{\partial w}{\partial t}$ by the right hand side of **eq. 14** and $\frac{\partial \boldsymbol{x}_k}{\partial t}$ by the right hand side of the corresponding equation in **eq. 27**.

Hence equation for w may be rewritten in terms of the trajectory integration of the following system

$$\begin{cases} \dfrac{\partial \boldsymbol{x}}{\partial t} = \mathbf{f}(\boldsymbol{x}, t), \\ \dfrac{\partial w}{\partial t} = -\operatorname{Tr}\left(\dfrac{\partial \mathbf{f}}{\partial \boldsymbol{x}}(\boldsymbol{x}, t)\right) w \end{cases} \tag{29}$$

Let us integrate **eq. 29** on a time step m ($m = 0, 1, \ldots, M-2$). If we set any spatial grid point $\boldsymbol{x}^* = X^{(g)}[:, n]$ ($n = 1, 2, \ldots, N$) as initial condition for the spatial variable, then we'll obtain solution $\widehat{w}_{m+1}$ for some point $\widehat{\boldsymbol{x}}_{m+1}$ outside the grid (see **Figure 1** with the illustration for the two-dimensional case). Hence we should firstly solve **eq. 27** backward in time to find the corresponding spatial point $\widehat{\boldsymbol{x}}_m$ that will be transformed to the grid point $\boldsymbol{x}^*$ by the step $m+1$. If we select this point $\widehat{\boldsymbol{x}}_m$ and the related value $\widehat{w}_m = w(\widehat{\boldsymbol{x}}_m, t_m)$ as initial conditions for the system **eq. 29**, then its solution w_{m+1} will be related to the point of interest $\boldsymbol{x}^*$.

Note that, according to our splitting scheme, we solve the convection part **eq. 14** after the corresponding diffusion **eq. 13**, and hence the initial condition w_m is already known and defined as a tensor $\mathcal{W}_m \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_d}$ on the Chebyshev spatial grid. Using this tensor, we can perform interpolation according to the formula **eq. 22** and calculate the tensor of interpolation coefficients $\mathcal{A}_m$. Then we can evaluate the approximated value at the point $\widehat{\boldsymbol{x}}_m$ as $\widetilde{w_m}(\widehat{\boldsymbol{x}}_m)$ according to **eq. 21**.

Hence our solution strategy for convection equation is the following. For the given spatial grid point $\boldsymbol{x}^* = X^{(g)}[:, n]$ we integrate equation

$$\frac{\partial \boldsymbol{x}}{\partial t} = \mathbf{f}(\boldsymbol{x}, t), \quad \boldsymbol{x}(t_{m+1}) = \boldsymbol{x}^*, \tag{30}$$

backward in time to find the corresponding point $\widehat{\boldsymbol{x}}_m = \boldsymbol{x}(t_m)$. Then we find the value of w at this point, using interpolation $\widetilde{w_m}$, and then we solve the system **eq. 29** on the time interval $(t_m, t_m + h)$ with initial condition $(\widehat{\boldsymbol{x}}_m, \widetilde{w_m}(\widehat{\boldsymbol{x}}_m))$ to obtain the value w_{m+1} at the point $\boldsymbol{x}^*$. The described process should be repeated for each grid point ($n = 1, 2, \ldots, N$) and, ultimately, we'll obtain a tensor $\mathcal{W}_{m+1} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_d}$ which is the approximated solution of convection part **eq. 14** of the splitting scheme on the Chebyshev spatial grid.

An important contribution of this paper is an indication of the possibility and a practical implementation of the usage of the multidimensional CAM in the TT-format to recover a supposedly low-rank tensor $\mathcal{W}_{m+1}$ from computations on only a part of specially selected spatial grid points. This scheme will be described in more details later in the work after setting out the fundamentals of the TT-format.

3 LOW-RANK REPRESENTATION

There has been much interest lately in the development of data-sparse tensor formats for high-dimensional problems. A very promising tensor format is provided by the tensor train (TT) approach (Oseledets and Tyrtyshnikov, 2009; Oseledets, 2011), which was proposed for compact representation and approximation of high-dimensional tensors. It can be computed via standard decompositions (such as SVD and QR-decomposition) but does not suffer from the curse of dimensionality[7].

[7]By the full format tensor representation or uncompressed tensor we mean the case, when one calculates and saves in the memory all tensor elements. The number of elements of an uncompressed tensor (hence, the memory required to store it) and the amount of operations required to perform basic operations with such tensor grows exponentially in the dimensionality, and this problem is called the curse of dimensionality.

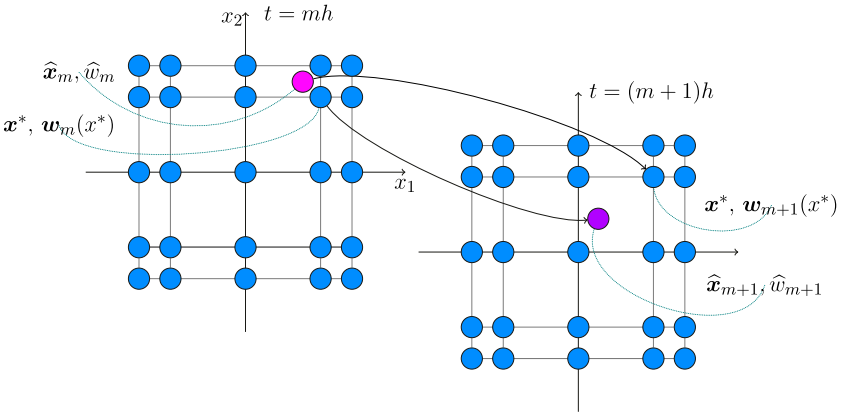

FIGURE 1 | Evolution of the spatial variable and the corresponding PDF for two consecutive time steps related to the fixed Chebyshev grid in the case of two dimensions.

In many analytical considerations and practical cases a tensor is given implicitly by a procedure enabling us to compute any of its elements, so the tensor appears rather as a black box. For example, to construct the convection part of PDF (i.e., the tensor $\mathcal{W}_m$ introduced above), we should compute the corresponding function for all possible sets of indices. This process requires an extremely large number of operations and can be time-consuming, so it may be useful to find some suitable low-parametric approximation of this tensor using only a small portion of all tensor elements. CAM (Oseledets and Tyrtyshnikov, 2010) which is a widely used method for approximation of high-dimensional tensors looks appropriate for this case.

In this section we describe the properties of the TT-format and multidimensional CAM that are necessary for efficient solution of our problem, as well as the specific features of the practical implementation of interpolation by the Chebyshev polynomials in terms of the TT-format and CAM.

3.1 Tensor Train Format

A tensor $\mathcal{R} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_d}$ is said to be in the TT-format (Oseledets, 2011), if its elements are represented by the formula

$$\mathcal{R}[n_1, n_2, \ldots, n_d] = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_{d-1}=1}^{R_{d-1}} \mathcal{G}_1[1, n_1, r_1]\mathcal{G}_2[r_1, n_2, r_2] \ldots \mathcal{G}_{d-1}[r_{d-2}, n_{d-1}, r_{d-1}]\mathcal{G}_d[r_{d-1}, n_d, 1] \tag{31}$$

where $n_k = 1, 2, \ldots, N_k$ $(k = 1, 2, \ldots, d)$, three-dimensional tensors $\mathcal{G}_k \in \mathbb{R}^{R_{k-1} \times N_k \times R_k}$ are named TT-cores, and integers $R_0, R_1, \ldots, R_d$ (with convention $R_0 = R_d = 1$) are named TT-ranks. The latter formula can be also rewritten in a more compact form

$$\mathcal{R}[n_1, n_2, \ldots, n_d] = G_1(n_1)G_2(n_2)\ldots G_d(n_d), \tag{32}$$

where $G_k(n_k) = \mathcal{G}_k[:, n_k, :]$ is an $R_{k-1} \times R_k$ matrix for each fixed n_k (since $R_0 = R_d = 1$, the result of matrix multiplications in **eq. 32** is a scalar). And a vector form of the TT-decomposition looks like

$$\boldsymbol{vec}(\mathcal{R}) = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_{d-1}=1}^{R_{d-1}} \mathcal{G}_1[1, :, r_1] \otimes \mathcal{G}_2[r_1, :, r_2] \otimes \ldots \otimes \mathcal{G}_d[r_{d-1}, :, 1], \tag{33}$$

where the slices of the TT-cores $\mathcal{G}_k$ are vectors of length N_k $(k = 1, 2, \ldots, d)$.

The benefit of the TT-decomposition is the following. Storage of the TT-cores $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_d$ requires less or equal than $d \times \max_{1 \le k \le d}(N_k R_k^2)$ memory cells (instead of $N = N_1 N_2 \ldots N_d \sim N_0^d$ cells for the uncompressed tensor, where N_0 is an average size of the tensor modes), and hence the TT-decomposition is free from the curse of dimensionality if the TT-ranks are bounded.

The detailed description of the TT-format and linear algebra operations in terms of this format[8] is given in works (Oseledets and Tyrtyshnikov, 2009; Oseledets, 2011). It is important to note that for a given tensor $\widehat{\mathcal{R}}$ in the full format, the TT-decomposition (compression) can be performed by a stable TT-SVD algorithm. This algorithm constructs an approximation $\mathcal{R}$ in the TT-format to the given tensor $\widehat{\mathcal{R}}$ with a prescribed accuracy ϵ_{TT} in the Frobenius norm[9]

$$\left\|\mathcal{R} - \widehat{\mathcal{R}}\right\|_F \le \epsilon_{TT} \cdot \left\|\widehat{\mathcal{R}}\right\|_F, \tag{34}$$

[8]All basic operations in the TT-format are implemented in the ttpy python package https://github.com/oseledets/ttpy and its MATLAB version https://github.com/oseledets/TT-Toolbox.

[9]An exact TT-representation exists for the given full tensor $\widehat{\mathcal{R}}$, and TT-ranks of such representation are bounded by ranks of the corresponding unfolding matrices (Oseledets, 2011). Nevertheless, in practical applications it is more useful to construct TT-approximation with a prescribed accuracy ϵ_{TT}, and then carry out all operations (summations, products, etc) in the TT-format, maintaining the same accuracy ϵ_{TT} of the result.

but a procedure of the tensor approximation in the full format is too costly, and is even impossible for large dimensions due to the curse of dimensionality. Therefore more efficient algorithms like CAM are needed to quickly construct the tensor in the low rank TT-format.

Algorithm 1 Cross approximation in the TT-format on the Chebyshev grid.

Data: function $\mathsf{r}(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathbb{R}^d$ is any d-dimensional spatial point inside $[-1,1]^d$; initial guess $\mathcal{R}_0 \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_d}$ in the TT-format; the accuracy of approximation ϵ_{CA}.

Result: TT-tensor $\mathcal{R} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_d}$, which collect the function values on the multidimensional Chebyshev grid.

Function $\mathtt{func}(\widehat{N})$**:**
  // Return a list of values for the set of indices $\widehat{N} \in \mathbb{R}^{I \times d}$ $(I \geq 1)$.
  Create vector: $\boldsymbol{r} \in \mathbb{R}^I$.
  for $i = 1$ **to** I **do**
    Set multi-index: $\boldsymbol{n} = \widehat{N}[i,:]$.
    Create vector: $\boldsymbol{x} \in \mathbb{R}^d$.
    Construct grid points: $\boldsymbol{x}[k] = \cos\frac{\pi \cdot (\boldsymbol{n}[k]-1)}{N_k - 1}$ for $k = 1, 2, \ldots, d$.
    Evaluate function value: $\boldsymbol{r}[i] = \mathsf{r}(\boldsymbol{x})$.
  end
  return $\boldsymbol{r}$
Calculate: $\mathcal{R} = \mathsf{rect\_cross}(\mathsf{func}, \mathcal{R}_0, \epsilon_{CA})$.

3.2 Cross Approximation Method

The CAM allows to construct a TT-approximation of the tensor with prescribed accuracy ϵ_{CA}, using only part of the full tensor elements. This method is a multi-dimensional analogue of the simple cross approximation method for the matrices (Tyrtyshnikov, 2000) that allows one to approximate large matrices in $\mathcal{O}(N_0 R^2)$ time by computing only $\mathcal{O}(N_0 R)$ elements, where N_0 is an average size of the matrix modes and R is the rank of the matrix. The CAM and the TT-format can significantly speed up the computation and reduce the amount of consumed memory as will be illustrated in the next sections on the solution of the model equations.

The CAM constructs a TT-approximation $\mathcal{R}$ to the tensor $\widehat{\mathcal{R}}$, given as a function $f(n_1, n_2, \ldots, n_d)$, that returns the $(n_1, n_2, \ldots, n_d)$ th entry of $\widehat{\mathcal{R}}$ for a given set of indices. This method requires only

Algorithm 2 Multidimensional polynomial interpolation in the TT-format.

Data: TT-tensor $\mathcal{R} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_d}$; the approximation accuracy ϵ.

Result: TT-tensor $\mathcal{A} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_d}$ of interpolation coefficients.

Extract and copy TT-cores $(\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_d)$ of the TT-tensor $\mathcal{R}$.
for $k = 1$ **to** d **do**
  // $\mathcal{G}_k \in \mathbb{R}^{R_{k-1} \times N_k \times R_k}$ with TT-ranks R_{k-1} and R_k.
  Set: $R_k^* = R_{k-1} \cdot R_k$.
  Change the axis order: $\mathcal{G}_k = \mathsf{swapaxes}(\mathcal{G}_k, 1, 2)$.
  Reshape to the matrix: $G_k = \mathsf{reshape}(\mathcal{G}_k, (N_k, R_k^*))$.
  for $r^* = 1$ **to** R_k^* **do**
    Set: $\boldsymbol{g} = G_k[:, r^*]$.
    Create vector: $\widehat{\boldsymbol{g}} \in \mathbb{R}^{2N_k - 2}$.
    Set: $\widehat{\boldsymbol{g}}[i] = \boldsymbol{g}[i]$ for $i = 1, 2, \ldots, N_k$.
    Set: $\widehat{\boldsymbol{g}}[i] = \boldsymbol{g}[2N_k - i]$ for $i = N_k + 1, N_k + 2, \ldots, 2N_k - 2$.
    Compute the FFT (real part): $\widehat{\boldsymbol{g}} = \mathsf{fft}(\widehat{\boldsymbol{g}})$.
    Set: $\boldsymbol{g}[i] = \widehat{\boldsymbol{g}}[i]$ for $i = 1, 2, \ldots, N_k$.
    Scale boundary items: $\boldsymbol{g}[1] = \frac{\boldsymbol{g}[1]}{2}$, $\boldsymbol{g}[N_k] = \frac{\boldsymbol{g}[N_k]}{2}$.
    Set: $G_k[:, r^*] = \frac{1}{N_k - 1}\boldsymbol{g}$.
  end
  Reshape back: $\mathcal{G}_k = \mathsf{reshape}(G_k, (N_k, R_{k-1}, R_k))$.
  Change back the axis order: $\mathcal{G}_k = \mathsf{swapaxes}(\mathcal{G}_k, 1, 2)$.
end
Construct TT-tensor $\mathcal{A}$ from the TT-cores $(\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_d)$.
Round up TT-tensor $\mathcal{A}$ to ϵ: $\mathcal{A} = \mathsf{tt\_round}(\mathcal{A}, \epsilon)$ // Optional.

$\mathcal{O}(d \times \max_{1 \leq k \leq d}(N_k R_k^3))$ operations for the construction of the approximation with a prescribed accuracy ϵ_{CA}, where $R_0, R_1, \ldots, R_d$ $(R_0 = R_d = 1)$ are TT-ranks of the tensor $\mathcal{R}$ [see detailed discussion of the CAM in (Oseledets and Tyrtyshnikov, 2010)]. It should be noted that TT-ranks can depend on the value of selected accuracy ϵ_{CA}, but for a wide class of practically interesting tasks the TT-ranks are bounded or depend polylogarithmically on ϵ_{CA} [(Oseledets, 2010; Oseledets, 2011) for more details and examples]. In **Algorithm 1** the description of the process of construction of the tensor in the TT-format on the Chebyshev grid by the CAM is presented (we'll call it as a function $\mathsf{cross}_{\mathbb{X}}(\cdot)$ below). We prepare function func, which transforms given indices into the spatial grid points and return an array of the corresponding values of the target $\mathsf{r}(\cdot)$. Then this function is passed as an argument to the standard rank adaptive method tt_rectcross from the ttpy package. The CAM is described in more detail in the original papers (Oseledets and Tyrtyshnikov, 2010; Savostyanov and Oseledets, 2011), as well as in a recent work (Dolgov and Savostyanov, 2020), which formulates a computationally efficient parallel implementation of the algorithm.

3.3 Multidimensional Interpolation

As was discussed in the previous sections, we discretize the FPE on the multidimensional Chebyshev grid and interpolate solution of the first diffusion equation in the splitting scheme **eq. 13** by the Chebyshev polynomials to obtain its values on custom spatial points (different from the grid nodes) and then perform efficient trajectory integration of the convection **eq. 14**.

The desired interpolation may be constructed from solution of the system of **eq. 22** in terms of the FFT (Trefethen, 2000), but for the high dimension numbers we have the exponential growth of computational complexity and memory consumption, hence it is very promising to construct tensor of the nodal values and the corresponding interpolation coefficients in the TT-format.

Consider a TT-tensor $\mathcal{R} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_d}$ with the list of TT-cores $[\mathcal{G}_1, \mathcal{G}_2, \ldots \mathcal{G}_d]$, which collects PDF values on the nodes of the Chebyshev grid at some time step (the related function is $\mathsf{r}(\boldsymbol{x})$, and this tensor is obtained, for example, by the CAM or according to TT-SVD procedure from the tensor in the full format). Then the corresponding TT-tensor $\mathcal{A} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_d}$ of interpolation coefficients with the TT-cores $[\tilde{\mathcal{G}}_1, \tilde{\mathcal{G}}_2, \ldots \tilde{\mathcal{G}}_d]$ can be constructed according to the scheme, which is presented in **Algorithm 2** [we'll call it as a function $\mathsf{interpolate}(\cdot)$ below].

In this Algorithm we use standard linear algebra operations swapaxes and reshape, which rearrange the axes and change the dimension of the given tensor respectively, function fft for construction of the one-dimensional FFT for the given vector, and function tt_round from the ttpy package, which round the given tensor to the prescribed accuracy ϵ. Note that the inner loop in **Algorithm 2** for r^* may be replaced by the vectorized computations of the corresponding two-dimensional FFT.

For the known tensor $\mathcal{A}$ we can perform a fast computation of the function value at any given spatial point $\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_d^\top]$ by a matrix product of the convolutions of the TT-cores of $\mathcal{A}$ with appropriate column vectors of Chebyshev polynomials

$$\mathsf{r}(\boldsymbol{x}) \approx \sum_{r_1=1}^{R_1}\sum_{r_2=1}^{R_2}\dots\sum_{r_{d-1}=1}^{R_{d-1}}\left(\sum_{n_1=1}^{N_1}\tilde{\mathcal{G}}_1[1,n_1,r_1]T_{n_1-1}(x_1)\right)\left(\sum_{n_2=1}^{N_2}\tilde{\mathcal{G}}_2[r_1,n_2,r_2]T_{n_2-1}(x_2)\right)\dots\left(\sum_{n_d=1}^{N_d}\tilde{\mathcal{G}}_d[r_{d-1},n_d,1]T_{n_d-1}(x_d)\right), \tag{35}$$

We'll call the corresponding function as $\mathsf{inter\_eval}(\mathcal{A}, X)$ below. This function constructs a list of $r(\,\cdot\,)$ values for the given set of I points $X \in \mathbb{R}^{d\times I}$ $(I \geq 1)$, using interpolation coefficients $\mathcal{A}$ and sequentially applying the formula **eq. 35** for each spatial point.

4 DETAILED ALGORITHMS

In **Algorithms 3**, **4** and **5** we combine the theoretical details discussed in the previous sections of this work and present the final calculation scheme for solution of the multidimensional FPE in the TT-format, using CAM (function $\mathsf{cross}_{\mathbb{X}}$, **Algorithm 1**)[10] and interpolation by the Chebyshev polynomials (function $\mathsf{interpolate}$ from **Algorithm 2** that constructs interpolation coefficients and function $\mathsf{inter\_eval}$ that evaluates interpolation result at given points according to the formula **eq. 35**).

We denote by einsum the standard linear algebra operation that evaluates the Einstein summation convention on the operands (see, for example, the numpy python package). Function vstack stack arrays in sequence vertically, function $\mathsf{ode\_solve}(\mathsf{rhs}, t_1, t_2, Y_0)$ (where t_1 and t_2 are initial and final times, rhs is the right hand side of equations, and matrix Y_0 collects initial conditions) solves a system of ODE with vectorized initial condition by the one step of the fourth order Runge-Kutta method.

Algorithm 3 Solution of the FPE in the TT-format.

Data: time grid parameters (final time τ and number of points $M \geq 2$); spatial grid parameters (dimension $d \geq 1$ and numbers of points $N_1 \geq 2, N_2 \geq 2, \dots, N_d \geq 2$ for each dimension); d-dimensional vector-function $\mathbf{f}(\boldsymbol{x}, t)$; functions $\frac{\partial}{\partial \boldsymbol{x}_i}\mathbf{f}_i(\boldsymbol{x}, t)$ $(i = 1, 2, \dots, d)$; function for the initial condition $\mathsf{r}_0(\boldsymbol{x})$; scalar diffusion coefficient χ; approximation accuracy ϵ.

Result: approximated solution $\mathcal{R} \in \mathbb{R}^{N_1\times N_2\times\dots\times N_d}$ of the FPE at the moment τ in the TT-format on the nodes of the Chebyshev grid.

Calculate the time step: $h = \frac{\tau}{M-1}$.
Generate random TT-tensor of rank-1: $\mathbb{Q} \in \mathbb{R}^{N_1\times N_2\times\dots\times N_d}$.
Compute a TT-tensor with initial PDF values: $\mathcal{R} = \mathsf{cross}_{\mathbb{X}}(\mathsf{r}_0, \mathbb{Q}, \epsilon)$.
Set initial guess (in terms of CAM) for convection term: $\mathcal{W}_0 = \mathcal{R}$.
for $k = 1$ **to** d **do**
  Construct the second order differential matrix D_k according to (23).
  Calculate the matrix exponential: $Z_k = e^{\frac{h}{2}\chi D_k}$.
end
for $m = 0$ **to** $M-2$ **do**
  Solve: $\mathcal{R}, \mathcal{W}_0 = \mathsf{step}(\mathcal{R}, \mathcal{W}_0, Z_1, Z_2, \dots, Z_d, h, m, \mathbf{f}, \frac{\partial}{\partial \boldsymbol{x}_1}\mathbf{f}_1, \frac{\partial}{\partial \boldsymbol{x}_2}\mathbf{f}_2, \dots, \frac{\partial}{\partial \boldsymbol{x}_d}\mathbf{f}_d)$
  // See Algorithm 4 with the implementation of a function step.
end

Algorithm 4 One computational step of solution of the FPE.

Data: variables from the namespace of Algorithm 3.

Result: approximated solution $\mathcal{R} \in \mathbb{R}^{N_1\times N_2\times\dots\times N_d}$ of the FPE at the current time moment on the nodes of the Chebyshev grid and updated initial guess $\mathcal{W}_0 \in \mathbb{R}^{N_1\times N_2\times\dots\times N_d}$ for convection term.

Set current time: $t = m \cdot h$.
// Update TT-cores of the TT-tensor $\mathcal{R}$ to compute the diffusion action from equation (13):
for $k = 1$ **to** d **do**
  Set: $\mathcal{G}_k = \mathsf{einsum}(ij, sjq \to siq, Z_k, \mathcal{G}_k)$ for $k = 1, 2, \dots, d$.
end
Round up TT-tensor $\mathcal{R}$ to ϵ: $\mathcal{R} = \mathsf{tt\_round}(\mathcal{R}, \epsilon)$ // Optional.
Calculate interpolation coefficients: $\mathcal{A} = \mathsf{interpolate}(\mathcal{R}, \epsilon)$.
Compute convection action (see equation (14)): $\mathcal{R} = \mathsf{cross}_{\mathbb{X}}(\mathsf{func}, \mathcal{W}_0, \epsilon)$.
// See Algorithm 5 with the implementation of a function func.
Perform normalization for $\mathcal{R}$ on $[-1, 1]$.
Set: $\mathcal{W}_0 = \mathcal{R}$.
// Update TT-cores of the TT-tensor $\mathcal{R}$ to compute the diffusion action from equation (13):
for $k = 1$ **to** d **do**
  Set: $\mathcal{G}_k = \mathsf{einsum}(ij, sjq \to siq, Z_k, \mathcal{G}_k)$ for $k = 1, 2, \dots, d$.
end
Round up TT-tensor $\mathcal{R}$ to ϵ: $\mathcal{R} = \mathsf{tt\_round}(\mathcal{R}, \epsilon)$ // Optional.

Algorithm 5 Function that solves convection term of the FPE.

Data: the set of points $X \in \mathbb{R}^{d\times I}$ $(I \geq 1)$; variables from the namespace of Algorithm 4.

Result: a list of function values $\boldsymbol{w} \in \mathbb{R}^I$.

Solve (27) backward in time: $X^* = \mathsf{ode\_solve}(\mathbf{f}, t+h, t, X)$.
Find interpolated values: $\boldsymbol{w}^* = \mathsf{inter\_eval}(\mathcal{A}, X^*)$.
Set initial condition for (29): $Z^* = \mathsf{vstack}([X^*, \boldsymbol{w}^*])$.
Function $\mathsf{rhs}\ (Y)$**:**
  // Return the rhs of (29) for the list of points $Y \in \mathbb{R}^{(d+1)\times I}$ $(I \geq 1)$.
  Set: $X = Y[1:-1, :]$.
  Set: $\boldsymbol{w} = Y[-1, :]$.
  Set: $F_0 = \mathbf{f}(X, t)$.
  Set: $F_1 = \sum_{i=1}^{d}\frac{\partial}{\partial \boldsymbol{x}_i}\mathbf{f}_i(X, t)$.
  return $\mathsf{vstack}([F_0, -F_1\boldsymbol{w}])$
Solve (29) and get the last variable: $\boldsymbol{w} = \mathsf{ode\_solve}(\mathsf{rhs}, t, t+h, Z^*)[-1, :]$.

5 NUMERICAL EXAMPLES

In this section we illustrate the proposed computational scheme, which was presented above, with the numerical experiments. All calculations were carried out in the Google Colab cloud interface[11] with the standard configuration (without GPU support).

Firstly we consider an equation with a linear convection term—Ornstein-Uhlenbeck process (OUP) (Vatiwutipong and Phewchean, 2019) in one, three and five dimensions. For the one-dimensional case, which is presented for convention, we only solve equation using the dense format (not TT-format), hence the corresponding results are used to verify the general correctness and convergence properties of the proposed algorithm, but not its efficiency. In the case of the multivariate problems we use the proposed tensor based solver, which operates in accordance with the algorithm described above. To check the results of our computations, we use the known analytic stationary solution for the OUP, and for the one-dimensional case we also perform comparison with constructed analytic solution at any time moment.

Then we consider more complicated dumbbell problem (Venkiteswaran and Junk, 2005) which may be represented as a three-dimensional FPE with a nonlinear convection term. For this case we consider the Kramer expression and compare our computation results with the results from another works for the same problem.

In the numerical experiments we consider the spatial region Ω such that PDF is almost vanish on the boundaries $\rho(\boldsymbol{x}, t)|_{\partial\Omega} \approx 0$, and the initial condition is selected in the form of the Gaussian function

[10]Note that in **Algorithm 4** we use the solution of the convection part from the previous time step $(k-1)$ as an initial guess $\mathcal{W}_0$ for CAM on the next time step (k). As it was found empirically It may seem more logical to use the solution of the diffusion part from the same time step (k) as an initial guess, but we found empirically that it leads to higher TT-ranks of the result.

[11]Actual links to the corresponding Colab notebooks are available in our public repository https://github.com/AndreiChertkov/fpcross.

$$\rho(\boldsymbol{x},0)=\rho_0(\boldsymbol{x})=(2\pi s)^{-\frac{d}{2}}\exp\Big[-\frac{1}{2s}||\boldsymbol{x}||^2\Big],\quad s\in\mathbb{R},\quad s>0, \tag{36}$$

where parameter s is selected as $s=1$. To estimate the accuracy of the obtained PDF ($\boldsymbol{\rho}$) we use the relative 2-norm of deviation from the exact value ($\boldsymbol{\rho}_{exact}$)

$$e=\frac{||\boldsymbol{\rho}-\boldsymbol{\rho}_{exact}||_2}{|\boldsymbol{\rho}_{exact}||_2}. \tag{37}$$

We compute the value $\boldsymbol{\rho}_{exact}$ through the given function, using a CAM with an accuracy parameter two orders of magnitude higher, than the one that was used in the solver.

5.1 Numerical Solution of the Ornstein-Uhlenbeck Process

Consider FPE of the form **eq. 6** in the d-dimensional case with

$$\mathbf{f}(\boldsymbol{x},t)=A(\boldsymbol{\mu}-\boldsymbol{x}(t)),\quad \chi=\frac{1}{2},\quad \boldsymbol{x}\in\Omega=[x_{min},x_{max}]^d,\quad t\in[0,\tau], \tag{38}$$

where $A\in\mathbb{R}^{d\times d}$ is invertible real matrix, $\boldsymbol{\mu}\in\mathbb{R}^d$ is the long-term mean, $x_{min}\in\mathbb{R}$ and $x_{max}\in\mathbb{R}$ ($x_{min}<x_{max}$), $\tau\in\mathbb{R}$ ($\tau>0$). This equation is a well known multivariate OUP with the following properties [see for example (Singh et al., 2018; Vatiwutipong and Phewchean, 2019)]:

- mean vector is

$$\boldsymbol{M}(t,\boldsymbol{x}_0)=e^{-At}\boldsymbol{x}_0+(I_d-e^{-At})\boldsymbol{\mu}; \tag{39}$$

- covariance matrix is

$$\Sigma(t)=\int_0^t e^{A(s-t)}SS^{\top}e^{A^{\top}(s-t)}ds, \tag{40}$$

and, in our case as noted above $S=\sqrt{2\chi}I_d$;

- transitional PDF is

$$\rho(\boldsymbol{x},t,\boldsymbol{x}_0)=\frac{\exp\Big[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{M}(t,\boldsymbol{x}_0))^{\top}\Sigma^{-1}(t)(\boldsymbol{x}-\boldsymbol{M}(t,\boldsymbol{x}_0))\Big]}{\sqrt{|2\pi\Sigma(t)|}}; \tag{41}$$

- stationary solution is

$$\rho_{st}(\boldsymbol{x})=\frac{\exp\Big[-\frac{1}{2}\boldsymbol{x}^{\top}W^{-1}\boldsymbol{x}\Big]}{\sqrt{(2\pi)^d det(W)}}, \tag{42}$$

where matrix $W\in\mathbb{R}^{d\times d}$ can be found from the following equation

$$AW+WA^{\top}=2\chi I_d; \tag{43}$$

- the (multivariate) OUP at any time is a (multivariate) normal random variable;

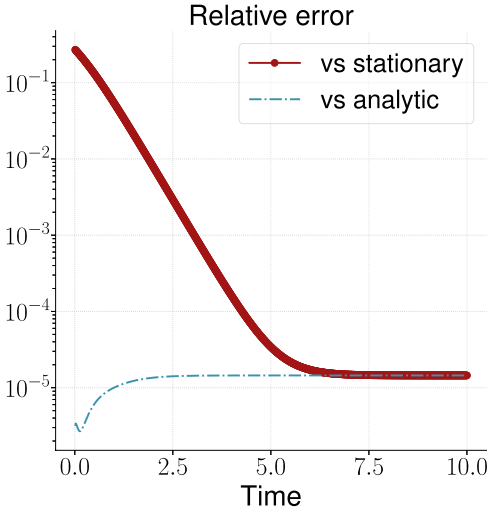

FIGURE 2 | Relative error of the calculated solution vs known analytic and stationary solutions for the one-dimensional OUP.

- the OUP is mean-reverting (the solution tends to its long-term mean $\boldsymbol{\mu}$ as time t tends to infinity) if all eigenvalues of A are positive (if $A>0$ in the one-dimensional case).

5.1.1 One-Dimensional Process

Let consider the one-dimensional ($d=1$) OUP with

$$A=1,\quad \mu=0,\quad x_{min}=-5,\quad x_{max}=5,\quad \tau=10. \tag{44}$$

We can calculate the analytic solution in terms of only spatial variable and time via integration of the transitional PDF **eq. 41**

$$\rho(x,t)=\int_{-\infty}^{\infty}\rho(x,t,x_0)\rho_0(x_0)\,dx_0. \tag{45}$$

Accurate computations lead to the following formula

$$\rho(x,t)=\frac{1}{\sqrt{2\pi(\Sigma(t)+se^{-2At})}}\exp\Big[-\frac{x^2}{2(\Sigma(t)+se^{-2At})}\Big], \tag{46}$$

where $\Sigma(t)$ is defined by **eq. 40** and for the one-dimensional case may be represented in the form

$$\Sigma(t)=\frac{1-e^{-2At}}{2A}. \tag{47}$$

Using the formulas **eq. 42** and **eq. 43** we can represent a stationary solution for the one-dimensional case in the explicit form

$$\rho_{stat}(x)=\sqrt{\frac{A}{\pi}}e^{-Ax^2}. \tag{48}$$

We perform computation for $N_1=50$ spatial points and $M=1000$ time points and compare the numerical solution with the known analytic **eq. 46** and stationary **eq. 48** solution. In the **Figure 2** we present the corresponding result. Over time, the error of the numerical solution relative to the analytical

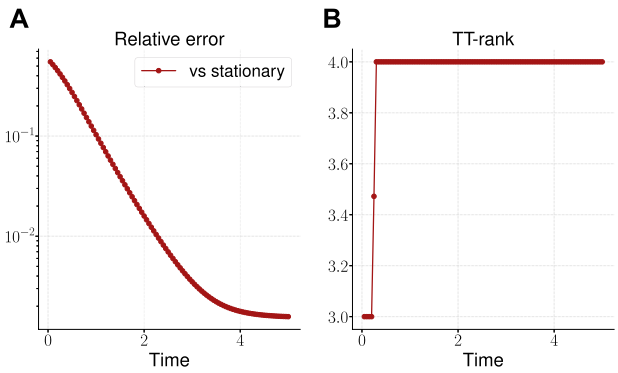

FIGURE 3 | Relative error of the calculated solution vs known stationary solution **(A)** and the effective TT-rank **(B)** for the three-dimensional OUP.

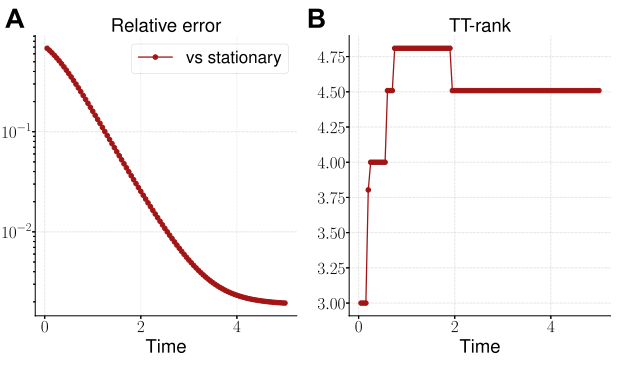

FIGURE 4 | Relative error of the calculated solution vs known stationary solution **(A)** and the effective TT-rank **(B)** for the five-dimensional OUP.

solution first increases slightly, and then stabilizes at approximately 10^{-5}. At the same time, the numerical solution approaches the stationary one, and the corresponding error at large times also becomes approximately 10^{-5}. Note that the time to build the solution was about 5 s.

5.1.2 Three-Dimensional Process

Our next example is the three-dimensional ($d = 3$) OUP with the following parameters

$$A = \begin{bmatrix} 1.5 & 1 & 0 \\ 0 & 1 & 0 \\ 0.5 & 0.3 & 1 \end{bmatrix}, \quad \boldsymbol{\mu} = 0, \quad x_{min} = -5, \quad x_{max} = 5, \quad \tau = 5. \tag{49}$$

When carrying out numerical calculation, we select 10^{-4} as the accuracy of the CAM, 100 as a total number of time points and 30 as a number of points along each of the spatial dimensions. The computation result is compared with the stationary solution **eq. 42** which was obtained as solution of the related matrix **eq. 43** by a standard solver for Lyapunov equation.

The result is shown in **Figure 3**. As can be seen, the TT-rank[12] remains limited, and the accuracy of the solution over time grows, reaching 10^{-3} by the time $t = 5$. The time to build the solution was about 26 s.

To evaluate the efficiency of the proposed algorithm in the TT-format, we also solve these three-dimensional OUP, using dense format (as for the one-dimensional case, all arrays are presented in its full form). The corresponding calculation took about 376 s, so in this case we have an acceleration of calculations by more than an order of magnitude.

5.1.3 Five-Dimensional Process

This multidimensional case is considered in the same manner as the previous one. We select the following parameters

$$A = \begin{bmatrix} 1.5 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0.5 & 0.3 & 0.2 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{\mu} = 0, \quad x_{min} = -5, \quad x_{max} = 5, \quad \tau = 5. \tag{50}$$

We select the same values as in the previous example for the CAM accuracy (10^{-4}), the number of time points (100) and the number of spatial points (30), and compare result of the computation with the stationary solution from **eq. 42** and **eq. 43**.

The results are presented on the plots on **Figure 4**. The TT-rank of the solution remains limited and reaches the value 4.5 at the end time step, and the solution accuracy reaches almost 10^{-3}. The time to build the solution was about 100 s.

5.2 Numerical Solution of the Dumbbell Problem

Now consider a more complex non-linear example corresponding to the three-dimensional ($d = 3$) dumbbell model of the form **eq. 6** with[13]

$$\mathbf{f}(\boldsymbol{x}, t) = A\boldsymbol{x} - \frac{1}{2}\nabla\phi, \quad A = \beta \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \phi = \frac{||\boldsymbol{x}||^2}{2} + \frac{\alpha}{p^3} e^{-\frac{||\boldsymbol{x}||^2}{2p^2}}, \tag{51}$$

where

[12]Hereinafter, we present effective TT-rank of the computation result. For TT-tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_d}$ with TT-ranks $R_0, R_1, \ldots, R_d$ ($R_0 = R_d = 1$) the effective TT-rank $\widehat{R}$ is a solution of quadratic equation $N_1\widehat{R} + \sum_{\alpha=2}^{d} N_\alpha \widehat{R}^2 + N_d\widehat{R} = \sum_{\alpha=1}^{d} N_\alpha R_{\alpha-1} R_\alpha$. The representation with a constant TT-rank $\widehat{R}$ ($\widehat{R}_0 = 1$, $\widehat{R}_1 = \widehat{R}_2 \ldots = \widehat{R}_{d-1} = \widehat{R}$, $\widehat{R}_d = 1$) yields the same total number of parameters as in the original decomposition of the tensor $\mathcal{X}$.

[13]This choice of parameters corresponds to the problem of polymer modeling from the work (Venkiteswaran and Junk, 2005). In the corresponding model, the molecules of the polymer are represented by beads and interactions are indicated by connecting springs. Accordingly, for the case of only two particles we come to the dumbbell problem, which can be mathematically written in the form of the FPE.

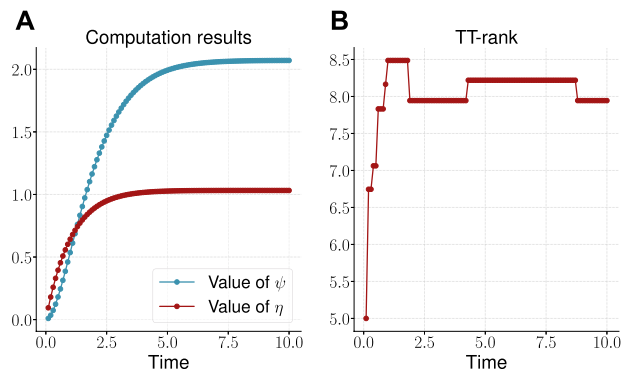

FIGURE 5 | Computed values **(A)** and the effective TT-rank **(B)** for the three-dimensional dumbbell problem.

$$\chi = \frac{1}{2}, \quad \boldsymbol{x} \in \Omega = [-10, 10]^3, \quad t \in [0, 10], \quad \alpha = 0.1, \quad \beta = 1, \quad p = 0.5. \tag{52}$$

Making simple calculations (taking into account the specific form of the matrix A), we get explicit expressions for the function and the required partial derivatives ($k = 1, 2, 3$)

$$\mathbf{f} = \beta \begin{bmatrix} \boldsymbol{x}_2 \\ 0 \\ 0 \end{bmatrix} - \frac{1}{2}\boldsymbol{x} + \frac{\alpha}{2p^5} e^{-\frac{\|\boldsymbol{x}\|^2}{2p^2}} \boldsymbol{x}, \tag{53}$$

$$\frac{\partial \mathbf{f}_k}{\partial \boldsymbol{x}_k} = -\frac{1}{2} + \frac{\alpha}{2p^5} e^{-\frac{\|\boldsymbol{x}\|^2}{2p^2}} - \frac{\alpha}{2p^7} e^{-\frac{\|\boldsymbol{x}\|^2}{2p^2}} \boldsymbol{x}_k^2. \tag{54}$$

Next, we consider the Kramer expression

$$\tau(t) = \int \rho(\boldsymbol{x}, t)[\boldsymbol{x} \otimes \nabla\phi] d\boldsymbol{x}, \tag{55}$$

and as the values of interest (as in the works (Venkiteswaran and Junk, 2005; Dolgov et al., 2012)) we select

$$\psi(t) = \frac{\tau_{11}(t) - \tau_{22}(t)}{\beta^2} = \frac{1}{\beta^2} \rho(\boldsymbol{x}, t) \left(\boldsymbol{x}_1 \frac{\partial \phi}{\partial \boldsymbol{x}_1} - \boldsymbol{x}_2 \frac{\partial \phi}{\partial \boldsymbol{x}_2} \right), \tag{56}$$

$$\eta(t) = \frac{\tau_{12}(t)}{\beta} = \frac{1}{\beta} \rho(\boldsymbol{x}, t) \boldsymbol{x}_1 \frac{\partial \phi}{\partial \boldsymbol{x}_2}. \tag{57}$$

During the calculations we used the following solver parameters:

- the accuracy of the CAM is 10^{-5};
- the number of time grid points is 100;
- the number of grid points along each of the spatial dimensions is 60.

The results are presented on the plots on **Figure 5**. The time to build the solution was about 200 s (also additional time was required to calculate the values $\psi(t)$ and $\eta(t)$ from **eq. 56** and **eq. 57** respectively). As can be seen, the TT-rank remains limited, and its stationary value is about 8. We compared the obtained stationary values of the $\psi(t)$ and $\eta(t)$ variables:

$$\psi(t = 10) = 2.0707, \quad \eta(t = 10) = 1.0318, \tag{58}$$

with the corresponding results from (Dolgov et al., 2012)[14], and we get the following values for relative errors

$$\epsilon_\psi = 1.9 \times 10^{-4}, \quad \epsilon_\eta = 9.7 \times 10^{-4}. \tag{59}$$

6 RELATED WORKS

The problem of uncertainty propagation through nonlinear dynamical systems subject to stochastic excitation is given by the FPE, which describes the evolution of the PDF, and has been extensively studied in the literature. A number of numerical methods such as the path integral technique (Wehner and Wolfer, 1983; Subramaniam and Vedula, 2017), the finite difference and the finite element method (Kumar and Narayanan, 2006; Pichler et al., 2013) have been proposed to solve the FPE.

These methods inevitably require mesh or associated transformations, which increase the amount of computation. The problem becomes worse when the system dimension increases. To maintain accuracy in traditional discretization based numerical methods, the number of degrees of freedom of the approximation, i.e. the number of unknowns, grows exponentially as the dimensionality of the underlying state-space increases.

On the other hand, the Monte Carlo method, that is common for such kind of problems (Kikuchi et al., 1991; Küchlin and Jenny, 2017), has slow rate of convergence, causing it to become computationally burdensome as the underlying dimensionality increases. Hence, the so-called curse of dimensionality fundamentally limits the use of the FPE for uncertainty quantification in high dimensional systems.

In recent years, low-rank tensor approximations have become especially popular for solving multidimensional problems in various fields of knowledge (Cichocki et al., 2016). However, for the FPE, this approach is not yet widely used. We note the works (Dolgov et al., 2012; Sun and Kumar, 2014; Sun and Kumar, 2015; Dolgov, 2019; Fox et al., 2020) in which the low-rank TT-decomposition was proposed for solution of the multidimensional FPE. In these works, the differential operator and the right-hand side of the system are represented in the form of TT-tensor. Moreover, in paper (Dolgov et al., 2012) the joint discretization of the solution in space-time is considered. The difference of our approach from these works is its more explicit iterative form for time integration, as well as the absence of the need to represent the right hand side of the system in a low-rank format, which allows to use this approach in machine learning applications.

[14]As values for comparison, we used the result of the most accurate calculation from work (Dolgov et al., 2012), within which $\widehat{\psi}(t = 10) = 2.071143$, and $\widehat{\eta}(t = 10) = 1.0328125$.

7 CONCLUSION

In this paper we proposed the novel numerical scheme for solution of the multidimensional Fokker–Planck equation, which is based on the Chebyshev interpolation and spectral differentiation techniques as well as low rank tensor approximations, namely, the tensor train decomposition and cross approximation method, which in combination make it possible to drastically reduce the number of degrees of freedom required to maintain accuracy as dimensionality increases.

The proposed approach can be used for the numerical analysis of uncertainty propagation through nonlinear dynamical systems subject to stochastic excitations, and we demonstrated its effectiveness on a number of multidimensional problems, including Ornstein-Uhlenbeck process and dumbbell model.

As part of the further development of this work, we plan to conduct more rigorous estimates of the convergence of the proposed scheme, as well as formulate a set of heuristics for the optimal choice of number of time and spatial grid points and tensor train rank. Another promising direction for further research is the application of established approaches and developed solver to the problem of density estimation for machine learning models.

DATA AVAILABILITY STATEMENT

Program code, input data and calculation results can be found here: https://github.com/AndreiChertkov/fpcross

AUTHOR CONTRIBUTIONS

IO contributed to general formulation of the problem and formulation of the preliminary version of the algorithm. AC contributed to development of the final version of algorithms and program code, carrying out numerical experiments.

FUNDING

The work was supported by Ministry of Science and Higher Education grant No. 075-10-2021-068.

REFERENCES

Chen, R. T., and Duvenaud, D. (2019). *Neural Networks with Cheap Differential Operators*. New York, NY: arXiv preprint arXiv:1912.03579.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural Ordinary Differential Equations. *Adv. Neural Inf. Process. Syst.*, 6571–6583.

Cichocki, A., Lee, N., Oseledets, I., Phan, A.-H., Zhao, Q., and Mandic, D. P. (2016). Tensor Networks for Dimensionality Reduction and Large-Scale Optimization: Part 1 Low-Rank Tensor Decompositions. *FNT Machine Learn.* 9, 249–429. doi:10.1561/2200000059

Dolgov, S., and Savostyanov, D. (2020). Parallel Cross Interpolation for High-Precision Calculation of High-Dimensional Integrals. *Comp. Phys. Commun.* 246, 106869. doi:10.1016/j.cpc.2019.106869

Dolgov, S. V. (2019). A Tensor Decomposition Algorithm for Large Odes with Conservation Laws. *Comput. Methods Appl. Math.* 19, 23–38. doi:10.1515/cmam-2018-0023

Dolgov, S. V., Khoromskij, B. N., and Oseledets, I. V. (2012). Fast Solution of Parabolic Problems in the Tensor Train/Quantized Tensor Train Format with Initial Application to the Fokker--Planck Equation. *SIAM J. Sci. Comput.* 34, A3016–A3038. doi:10.1137/120864210

Fox, C., Dolgov, S., Morrison, M. E., and Molteno, T. C. (2020). Grid Methods for Bayes-Optimal Continuous-Discrete Filtering and Utilizing a Functional Tensor Train Representation. *Inverse Probl. Sci. Eng.*, 1–19. doi:10.1080/17415977.2020.1862109

Glowinski, R., Osher, S. J., and Yin, W. (2017). *Splitting Methods in Communication, Imaging, Science, and Engineering*. Springer.

Grathwohl, W., Chen, R. T., Bettencourt, J., Sutskever, I., and Duvenaud, D. (2018). *Ffjord: Free-form Continuous Dynamics for Scalable Reversible Generative Models*. arXiv preprint arXiv:1810.01367.

Kidger, P., Foster, J., Li, X., Oberhauser, H., and Lyons, T. (2021). *Neural Sdes as Infinite-Dimensional gans*. arXiv preprint arXiv:2102.03657.

Kikuchi, K., Yoshida, M., Maekawa, T., and Watanabe, H. (1991). Metropolis Monte Carlo Method as a Numerical Technique to Solve the Fokker-Planck Equation. *Chem. Phys. Lett.* 185, 335–338. doi:10.1016/s0009-2614(91)85070-d

Küchlin, S., and Jenny, P. (2017). Parallel Fokker-Planck-DSMC Algorithm for Rarefied Gas Flow Simulation in Complex Domains at All Knudsen Numbers. *J. Comput. Phys.* 328, 258–277. doi:10.1016/j.jcp.2016.10.018

Kumar, P., and Narayanan, S. (2006). Solution of Fokker-Planck Equation by Finite Element and Finite Difference Methods for Nonlinear Systems. *Sadhana* 31, 445–461. doi:10.1007/bf02716786

Oseledets, I., and Tyrtyshnikov, E. (2010). Tt-cross Approximation for Multidimensional Arrays. *Linear Algebra its Appl.* 432, 70–88. doi:10.1016/j.laa.2009.07.024

Oseledets, I. V. (2010). Approximation of $2d\times2d$ Matrices Using Tensor Decomposition. *SIAM J. Matrix Anal. Appl.* 31, 2130–2145. doi:10.1137/090757861

Oseledets, I. V. (2011). Tensor-train Decomposition. *SIAM J. Sci. Comput.* 33, 2295–2317. doi:10.1137/090752286

Oseledets, I. V., and Tyrtyshnikov, E. E. (2009). Breaking the Curse of Dimensionality, or How to Use Svd in many Dimensions. *SIAM J. Sci. Comput.* 31, 3744–3759. doi:10.1137/090748330

Pichler, L., Masud, A., and Bergman, L. A. (2013). "Numerical Solution of the Fokker-Planck Equation by Finite Difference and Finite Element Methods-A Comparative Study," in *Computational Methods in Stochastic Dynamics* (Springer), 69–85. doi:10.1007/978-94-007-5134-7_5

Savostyanov, D., and Oseledets, I. (2011). "Fast Adaptive Interpolation of Multi-Dimensional Arrays in Tensor Train Format," in *The 2011 International Workshop on Multidimensional (nD) Systems* (IEEE), 1–8. doi:10.1109/nds.2011.6076873

Singh, R., Ghosh, D., and Adhikari, R. (2018). Fast Bayesian Inference of the Multivariate Ornstein-Uhlenbeck Process. *Phys. Rev. E* 98, 012136. doi:10.1103/PhysRevE.98.012136

Subramaniam, G. M., and Vedula, P. (2017). A Transformed Path Integral Approach for Solution of the Fokker-Planck Equation. *J. Comput. Phys.* 346, 49–70. doi:10.1016/j.jcp.2017.06.002

Sun, Y., and Kumar, M. (2015). A Numerical Solver for High Dimensional Transient Fokker-Planck Equation in Modeling Polymeric Fluids. *J. Comput. Phys.* 289, 149–168. doi:10.1016/j.jcp.2015.02.026

Sun, Y., and Kumar, M. (2014). Numerical Solution of High Dimensional Stationary Fokker-Planck Equations via Tensor Decomposition and Chebyshev Spectral Differentiation. *Comput. Math. Appl.* 67, 1960–1977. doi:10.1016/j.camwa.2014.04.017

Trefethen, L. N. (2000). *Spectral Methods in MATLAB, Vol. 10*. Philadelphia, PA: Siam.

Tyrtyshnikov, E. (2000). Incomplete Cross Approximation in the Mosaic-Skeleton Method. *Computing* 64, 367–380. doi:10.1007/s006070070031

Vatiwutipong, P., and Phewchean, N. (2019). Alternative Way to Derive the Distribution of the Multivariate Ornstein-Uhlenbeck Process. *Adv. Differ. Equ* 2019, 276. doi:10.1186/s13662-019-2214-1

Venkiteswaran, G., and Junk, M. (2005). A QMC Approach for High Dimensional Fokker-Planck Equations Modelling Polymeric Liquids. *Mathematics Comput. Simulation* 68, 43–56. doi:10.1016/j.matcom.2004.09.002

Wehner, M. F., and Wolfer, W. G. (1983). Numerical Evaluation of Path-Integral Solutions to Fokker-Planck Equations. *Phys. Rev. A.* 27, 2663–2670. doi:10.1103/physreva.27.2663

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.