Journal of Computational Physics 511 (2024) 113110

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Solving high-dimensional Fokker-Planck equation with functional hierarchical tensor ☆

Xun Tang *, Lexing Ying

Stanford University, Stanford, CA, 94305, USA

ARTICLE INFO

Keywords:
High-dimensional Fokker-Planck equation
Functional tensor network
Hierarchical tensor network
Curse of dimensionality

ABSTRACT

This work is concerned with solving high-dimensional Fokker-Planck equations with the novel perspective that solving the PDE can be reduced to independent instances of density estimation tasks based on the trajectories sampled from its associated particle dynamics. With this approach, one sidesteps error accumulation occurring from integrating the PDE dynamics on a parameterized function class. This approach significantly simplifies deployment, as one is free of the challenges of implementing loss terms based on the differential equation. In particular, we introduce a novel class of high-dimensional functions called the functional hierarchical tensor (FHT). The FHT ansatz leverages a hierarchical low-rank structure, offering the advantage of linearly scalable runtime and memory complexity relative to the dimension count. We introduce a sketching-based technique that performs density estimation over particles simulated from the particle dynamics associated with the equation, thereby obtaining a representation of the Fokker-Planck solution in terms of our ansatz. We apply the proposed approach successfully to three challenging time-dependent Ginzburg-Landau models with hundreds of variables.

1. Introduction

Solving high-dimensional partial differential equations (PDE) remains one of the main challenges in scientific computing, with applications in molecular dynamics, quantum mechanics, and high-dimensional control theory. A key element underlying the challenges in solving high-dimensional PDE is the fact that the computational resources needed to carry out traditional numerical techniques often scale exponentially in the dimension of the problem [1]. This paper focuses on solving the Fokker-Planck equation:

$$\partial_t p = \nabla \cdot (p \nabla V) + \frac{1}{\beta} \Delta p, \quad x \in \Omega \subset \mathbb{R}^d, t \in [0, T], \tag{1}$$

which governs the distribution function $p(t, x)$ of the particle system governed by the Langevin dynamics $dX_t = -\nabla V(X_t)dt + \sqrt{2\beta^{-1}} dB_t$ over a potential function V with inverse temperature β.

Solving Fokker-Planck equations in low dimensions is a well-developed topic. This paper is concerned with the case of hundreds or even thousands of dimensions. One prevalent type of high-dimensional Fokker-Planck equation comes from the discretization of an infinite-dimensional functional equation into finite but very large dimensions. Throughout this text, we use the motivating example

☆ Code implementation can be found in https://github.com/Xun-Tang123/Fokker_planck_with_FHT.

* Corresponding author.
E-mail address: xuntang@stanford.edu (X. Tang).

https://doi.org/10.1016/j.jcp.2024.113110
Received 9 December 2023; Received in revised form 18 April 2024; Accepted 12 May 2024

Available online 22 May 2024

of a time-dependent Ginzburg-Landau (G-L) model used for studying the phenomenological theory of superconductivity [2–5]. In the 2D G-L model, one can intuitively think of a particle taking the form of a field $x(a): [0,1]^2 \to \mathbb{R}$, where the potential function is a functional $V(x)$ defined as follows:

$$V(x) = \frac{\lambda}{2} \int_{[0,1]^2} |\nabla_a x(a)|^2 \, da + \frac{1}{4\lambda} \int_{[0,1]^2} |1 - x(a)^2|^2 \, da, \tag{2}$$

where λ is a parameter balancing the relative importance of the first term and the second term in (2). In the numerical treatment, one considers the discretization of the unit square $[0,1]^2$ into a grid of $d = m^2$ points $\{(ih, jh)\}$, for $h = \frac{1}{m+1}$ and $1 \le i, j \le m$. The discretization of the field at this grid is the d-dimensional vector $x = (x_{(i,j)})_{1 \le i,j \le m}$, where $x_{(i,j)} = x(ih, jh)$. Under the discretization scheme, the potential energy is defined as

$$V(x) = V(x_{(1,1)}, \dots, x_{(m,m)}) := \frac{\lambda}{2} \sum_{v \sim w} \left(\frac{x_v - x_w}{h} \right)^2 + \frac{1}{4\lambda} \sum_v \left(1 - x_v^2\right)^2, \tag{3}$$

where v and w are Cartesian grid points and $v \sim w$ if and only if they are adjacent. The evolution of the distribution over discretized fields satisfies the Fokker-Planck equation (1) with the discretized potential function $V(x)$. Even a moderate grid resolution results in a model very large in dimension, which is a recurring theme for discretizing infinite-dimensional functional differential equations. The associated functional differential equation is the equation for the functional $p(t,x)$ encoding the probability density at the function x in time t.

1.1. Background and contribution

Solving Fokker-Planck equations without time-stepping Traditionally, numerical solvers for the high-dimensional Fokker-Planck equations rely on performing time integration of a given initial condition $p|_{t=0}$ along a chosen parametric function class. The time integration can be done either by directly performing the time stepping of the solution or it can be done by including the PDE loss in the regression target. A common feature of the time integration scheme is error accumulation, as errors from previous time steps tend to build up. Moreover, compressing the differential operator $\nabla \cdot (p \nabla V)$ in (1) is often performed with heuristics, leading to quite significant implementation challenges both in the design of the compression scheme and in its development into code.

The key insight in this paper is that solving the high-dimensional time-dependent Fokker-Planck equation can be reduced to a series of density estimation tasks. The reasoning is rather simple: one can approximate the Fokker-Planck solution $p(t,x)$ at any time $t \in [0,T]$ through density estimation over the sampled trajectory at time t. Thus, one can approximate the solution $p(t,x)$ at a collection of time steps $0 = t_0 \le \cdots \le t_K = T$, and the continuous-in-time solution $p(t,x)$ for an arbitrary $t \in [0,T]$ can be approximated through appropriate interpolation. More specifically, we take advantage of the associated particle dynamics to the Fokker-Planck equation:

$$dX_t = -\nabla V(X_t) dt + \sqrt{2\beta^{-1}} dB_t. \tag{4}$$

By simulating the dynamics of (4) through stochastic dynamic equation (SDE) simulation, we obtain a particle-based empirical approximation of $p(t,x)$ by independently sampled trajectory data $\{X_i(t)\}_{t \in [0,T]}$ for $i = 1, \dots, N$, which constitutes the input to the density estimation task. Thus, if $p(t,x)$ belongs to a function class with an efficient denoising or density estimation procedure, we can approximate the Fokker-Planck solution via denoising this empirical distribution. In effect, this approach sidesteps both the error accumulation and the coding challenges of a time integration scheme.

Here, we choose the functional hierarchical tensor as the ansatz of the PDE solution $p(t,x)$. The rationale for this choice is that the diffusive nature of the Fokker-Planck equations tends to keep the tensor rank under control. Compared to most neural network algorithms, the proposed ansatz can efficiently calculate the normalization constant (see Section 1.2 for detailed discussions). Moreover, while directly modeling the density through nonparametric density estimation suffers from the curse of dimensionality [6], our proposed approach effectively performs by denoising the empirical distribution through restriction to the proposed parametric function class, thus avoiding the curse of dimensionality. Among the possible choices for functional tensor networks, while an alternative is to perform density estimation with a functional tensor train [7–11], we choose a functional hierarchical tensor to better account for low-rank structures of more general high-dimensional Fokker-Planck equations.

Functional hierarchical tensor Functional hierarchical tensor is understood in the literature [12–14] to be a suitable tool for reduced-order modeling due to its versatility for multidimensional systems. In particular, the ansatz uses a modeling assumption natural for modeling discretized functional differential equations. The hierarchical tensor network assumes a low-rank structure along a hierarchical bipartition of the variables, which readily applies to discretized infinite-dimensional models. For example, the 2D G-L model has the natural geometry of a 2D grid, and it is natural to construct hierarchical bipartition of the variables through alternatively partitioning the variables along the two axes of the 2D grid, as shown in Fig. 1. In comparison, it is known that the tensor train ansatz assumes the variables $x = (x_1, \dots, x_d)$ satisfy low-rankness according to the 1D structure set by the tensor network [15], which makes it unsuitable for the 2D Ginzburg-Landau model, and the same argument applies to models for which the variables do not obey a 1D topological structure.

Our choice of functional hierarchical tensor over functional tensor train is partially motivated by parallel developments in quantum physics. For the task of modeling quantum systems, the tensor train ansatz is more commonly known as matrix product states

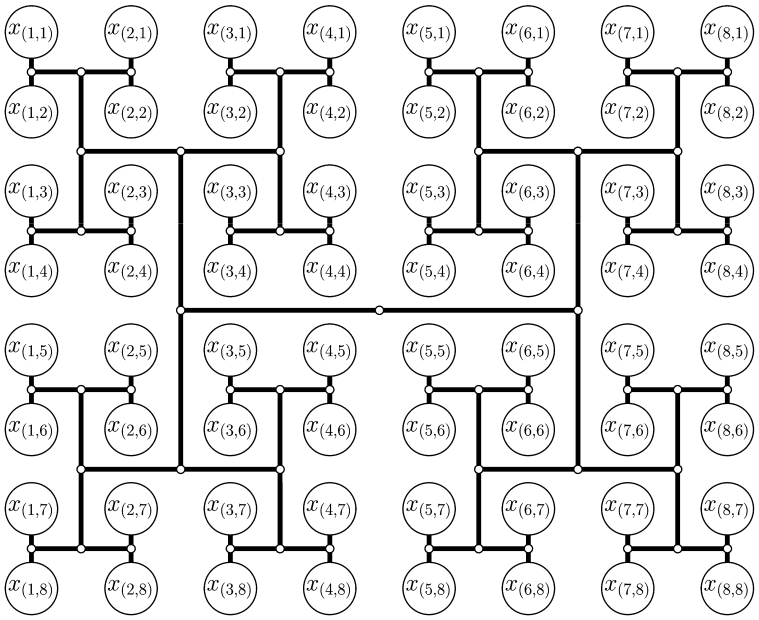


Fig. 1. Illustration of the hierarchical bipartition of a 2D Ginzburg-Landau model discretized to an 8×8 grid.

[16], where it is well-known that it is more suitable for modeling the ground state of a 1D system [17]. For modeling systems with nonlocal interaction, a hierarchical tensor network is often more advantageous [18]. Through the use of a functional tensor network, our work extends the capability of the hierarchical tensor network into modeling continuous distributions.

Density estimation through hierarchical sketching After the trajectories are sampled through particle simulations, we process them through a sketching-based density estimation algorithm, obtaining a hierarchical tensor representation of the Fokker-Planck equation solution. This work's main contribution is an end-to-end procedure that produces a Fokker-Planck solution from sampled trajectories.

To ensure that an accurate approximation can be obtained with a small parameter size, we have worked out practical choices over key design questions. The important ingredients are the structure of the hierarchical bipartition, the sketch function, and the basis to be used for the hierarchical tensor. Using sophisticated numerical techniques, we are able to obtain solutions to Ginzburg-Landau models of very high dimensions, including solving the multidimensional Ginzburg-Landau model in (3) with d in the hundreds.

1.2. Related work

Neural network for solving high-dimensional PDEs Neural network has shown tremendous success in recent years in solving high-dimensional PDEs. The most common approach to solve PDEs with neural networks primarily relies on penalizing the partial-differential equation either in terms of the strong form (as in the physics-informed neural network [19] and the deep Galerkin method [20]) or the variational form (as in the deep Ritz method [21]). Another separate direction is to solve the PDE by creating regression targets special to the problem itself, as can be seen in the deep BSDE method [1] and neural operators [22]. While these algorithms perform well, further improving their effectiveness in high-dimensional PDEs is challenging.

Neural network methods for density estimation As our work primarily focuses on using density estimation to solve the Fokker-Planck equation, our work is related to generative learning algorithms for which density estimation is possible. Examples include normalizing flows [23,24], energy-based models [25,26], and the more recent diffusion models [27,28]. Only normalizing flows allow direct density evaluation. In contrast, the energy-based model and diffusion model only allow for approximating the likelihood function, which renders them inapplicable in our case, as the objective is to obtain the model's associated normalized likelihood function. Moreover, training several neural networks across many time steps might be inefficient, while our sketching-based approach relies on quick linear algebraic subroutines and does not involve iterative optimization.

Monte-Carlo method As one has access to trajectories of the particle dynamics due to Monte-Carlo sampling, one might hope to perform density estimation through nonparametric density estimation directly. However, particle-based density estimations such as kernel density estimation suffer from the curse of dimensionality [6], making it ill-suited to our problem. If the goal is to estimate the statistical moments of the solved Fokker-Planck equation, then indeed the traditional Monte-Carlo approach is well-suited to estimate such quantity with an $O(1/\sqrt{N})$ rate [29], N being the number of samples.

Tensor network for high-dimensional PDE There are several works using tensor networks to solve high-dimensional PDEs. One approach is to use grid discretization of the PDE solution p, resulting in the task of solving the PDE for a d-dimensional tensor, as can be seen in [30]. The functional tensor network approach employs a mesh-free representation of the PDE solution based on a tensor network. The success of representing a general PDE solution by a tensor network depends on whether it has a low-rank representation. General conditions on approximating functions with functional hierarchical tensor can be found in [31]. Along this line of work, the considered tensor network structures include canonical-polyadic (CP) [32] and tensor train [33,14,11,9,34]. The use of hierarchical tensor in solving high-dimensional PDE has also been discussed [35], though quite different from our proposed methodology. The work in [36,7] bears the most resemblance to our work, as they combine particle methods with a functional tensor network structure. The main difference is the fact that our proposed algorithm does not involve a time-stepping component, and therefore, our method is less prone to error accumulation. A similar sketching-based density estimation algorithm for solving the Fokker-Planck equation with a functional tensor train can be done by combining our approach with the sketching methods outlined in [15]. In addition, density estimation under the tensor network ansatz has been considered in [37–39]. Contrary to the aforementioned works, this work is the first to consider sketching-based density estimation under the functional hierarchical tensor representation.

1.3. Contents and notations

We outline the structure of the remainder of the manuscript. Section 2 gives a detailed introduction to functional hierarchical tensor and its sketching-based density estimation algorithm. Section 3 goes through the algorithm for solving the Fokker-Planck equation with the functional hierarchical tensor. Section 4 details the numerical implementations and results from solving Ginzburg-Landau models in multiple dimensions.

For notational compactness, we introduce several shorthand notations for simple derivation. For $n \in \mathbb{N}$, let $[n] := \{1, \dots, n\}$. For an index set $S \subset [d]$, we let x_S stand for the subvector with entries from index set S. We use $\bar{S}$ to denote the set-theoretic complement of S, i.e. $\bar{S} = [d] - S$.

2. Hierarchical functional tensor network

To address the Ginzburg-Landau and general models arising from discretizing high-dimensional functional equations, we propose using the function class of a functional hierarchical tensor based on a binary-tree-based low-rank structure. In this section, we go over the functional hierarchical tensor and its sketching algorithm. The symbol d is reserved for the dimension of the state-space variable, and the symbol N is reserved for the number of samples. Without loss of generality, we shall go over the procedure to perform density estimation over a collection of N samples $\{y^{(i)} \in \mathbb{R}^d\}_{i=1}^N$. One can think of $y^{(i)}$ as the data collected from the i-th trajectory at a fixed time $t \in [0, T]$.

2.1. Hierarchical bipartition

We first describe the functional tensor network representation of a d-dimensional function $p\colon \mathbb{R}^d \to \mathbb{R}$. In general, let $\{\psi_i\}_{i=1}^n$ denote a collection of orthonormal function basis over a single variable, and let $C \in \mathbb{R}^{n^d}$ be the tensor represented by a tensor network. The *functional tensor network* is the d-dimensional function defined by the following equation:

$$p(x) \equiv p(x_1, \dots, x_d) = \sum_{i_1, \dots, i_d = 0}^{n-1} C_{i_1, \dots, i_d} \psi_{i_1}(x_1) \cdots \psi_{i_d}(x_d) = \left\langle C, \bigotimes_{j=1}^{d} \vec{\Psi}(x_j) \right\rangle, \tag{5}$$

where $\vec{\Psi}(x_j) = \left[\psi_1(x_j), \dots, \psi_n(x_j)\right]$ is an n-vector encoding the evaluation of any x_j over the entire single variable function basis set.

In this work, we choose C to be represented by a hierarchical tensor network ansatz so that the associated p takes the form of a functional hierarchical tensor. In addition to density evaluation, this ansatz allows for efficient evaluation of moments and efficient sampling (see Section 3 for more details on potential application).

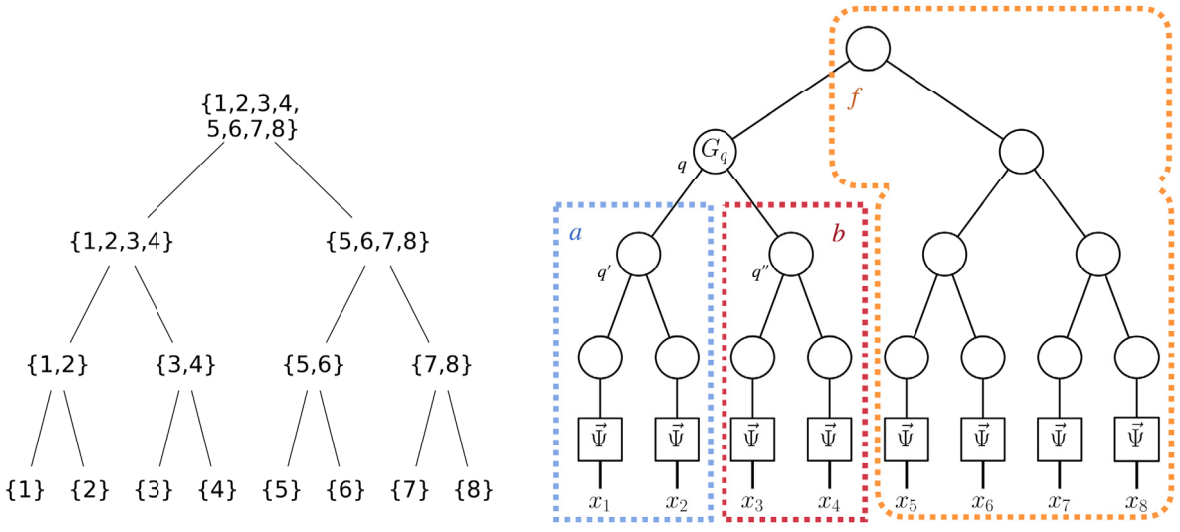
Crucial to the notion of the hierarchical tensor network is a hierarchical bipartition of the variable set. Without loss of generality, let $d = 2^L$ so that the variable set admits exactly L levels of variable bipartition. As illustrated in Fig. 2a, at the l-th level, the variable index set is partitioned according to

$$[d] = \bigcup_{k=1}^{2^l} I_k^{(l)}, \quad I_k^{(l)} := \{2^{L-l}(k-1) + 1, \dots, 2^{L-l}k\}, \tag{6}$$

which in particular implies the recursive relation that $I_k^{(l)} = I_{2k-1}^{(l+1)} \cup I_{2k}^{(l+1)}$.

2.2. Tensor network structure

We introduce the fundamental structure of a functional hierarchical tensor. Importantly, we focus on a single node q, say at level index l and block index $k \in [2^l]$. We take the shorthand $a(q) := I_{2k-1}^{(l+1)}, b(q) := I_{2k}^{(l+1)}$ and $f(q) = [d] - a(q) \cup b(q)$, to be its left branch, right branch, and the rest, respectively, as illustrated in Fig. 2b. While these index sets depend on the node q, in order to simplify the formulae, we refer to them just as a, b, and f.



(a) Binary decomposition of variables (b) Functional hierarchical tensor diagram

Fig. 2. Illustrations of functional hierarchical tensor with $d = 8$.

Intrinsically, a hierarchical tensor network represents a tensor $C \in \mathbb{R}^{n^d}$ with a low-rank structure along every hierarchical bipartition. The bipartition at the node q is defined to be $[d] = \bar{f} \cup f$, where $\bar{f}$ is the complement of f as a subset of $[d]$. With the partition $[d] = \bar{f} \cup f$, there exists $r_f \in \mathbb{N}$ so that the $n^{|\bar{f}|} \times n^{|f|}$ unfolding matrix $C(i_{\bar{f}}; i_f)$ is of rank r_f. In other words, there exist $C_{\bar{f}} : \mathbb{R}^{n^{|\bar{f}|}} \times \mathbb{R}^{r_f}$ and $C_f : \mathbb{R}^{r_f} \times \mathbb{R}^{n^{|f|}}$ that form a low-rank decomposition of C (illustrated in Fig. 3(a)):

$$C(i_1, \ldots, i_d) = \sum_{\theta=1}^{r_f} C_{\bar{f}}(i_{\bar{f}}, \theta) C_f(\theta, i_f). \tag{7}$$

If the low-rankness property in (7) holds for any node q, the exponential-sized tensor C can be described by a mere $O(dr^3)$ number of parameters by a hierarchical tensor network, where $r := \max_q(r_{f(q)})$. The tensor network is parameterized by a collection of tensor cores denoted by $\{G_q\}_q$. As Fig. 2 illustrates, the tensor cores have the same tree structure as in the variable bipartition. Each internal bond of the tensor network is associated with a bond in the binary decomposition of the variables. Each physical bond of the tensor network connects a leaf node with a $\vec{\Psi}(x_j)$ node. In summary, the graphical structure of the hierarchical tensor network is formed by a binary tree, and the physical index is at the leaf node. The fact that such cores exist can be proved by induction using (7), and a proof can be found in [13,40]. In terms of runtime complexity and memory complexity, the functional hierarchical tensor enjoys the same $O(d)$ scaling as that of a hierarchical tensor [40].

2.3. Sketching algorithm

We shall go over how to use a sketch-based method to obtain a functional hierarchical tensor representation from the given collection of samples $\left\{ y^{(i)} := \left(y_1^{(i)}, \ldots, y_d^{(i)} \right) \right\}_{i=1}^N$. For the reader's convenience, important equations in this subsection are included in Fig. 3 in terms of the tensor diagram. Below, we review the main equation behind hierarchical sketching in the functional case. The goal is to solve for the tensor core G_q, where q is the node of the k-th block at level l.

Equations For simplicity, we first assume $0 < l < L$ so that q is neither the root nor the leaf node. Similar as before, we have $a := I_{2k-1}^{(l+1)}, b := I_{2k}^{(l+1)}$ and $f = [d] - a \cup b$. Then, the structural low-rankness property of (7) implies that there exist $r_a, r_b, r_f \in \mathbb{N}$, $C_a : [n^{|a|}] \times [r_a] \to \mathbb{R}$, $C_b : [n^{|b|}] \times [r_b] \to \mathbb{R}$, $C_f : [r_f] \times [n^{|f|}] \to \mathbb{R}$ such that the following equation holds (illustrated in Fig. 3(a)):

$$C(i_1, \ldots, i_d) = \sum_{\alpha, \beta, \theta} C_a(i_a, \alpha) C_b(i_b, \beta) G_q(\alpha, \beta, \theta) C_f(\theta, i_f). \tag{8}$$

Equation (8) is exponential-sized, and therefore, one would not solve this equation directly. The hierarchical sketching algorithm essentially solves the over-determined linear system (8) for G_q with the use of sketch functions. Let $\tilde{r}_a, \tilde{r}_b, \tilde{r}_f$ be integers so that $\tilde{r}_a > r_a, \tilde{r}_b > r_b, \tilde{r}_f > r_f$. Through the sketch functions $S_a : [n^{|a|}] \times [\tilde{r}_a] \to \mathbb{R}, S_b : [n^{|b|}] \times [\tilde{r}_b] \to \mathbb{R}, S_f : [n^{|f|}] \times [\tilde{r}_f] \to \mathbb{R}$, we can contract the linear system in (8) with S_a, S_b, S_f at the variables i_a, i_b, i_f respectively, which leads to the following sketched linear system (illustrated in Fig. 3(b) and Fig. 3(d)):

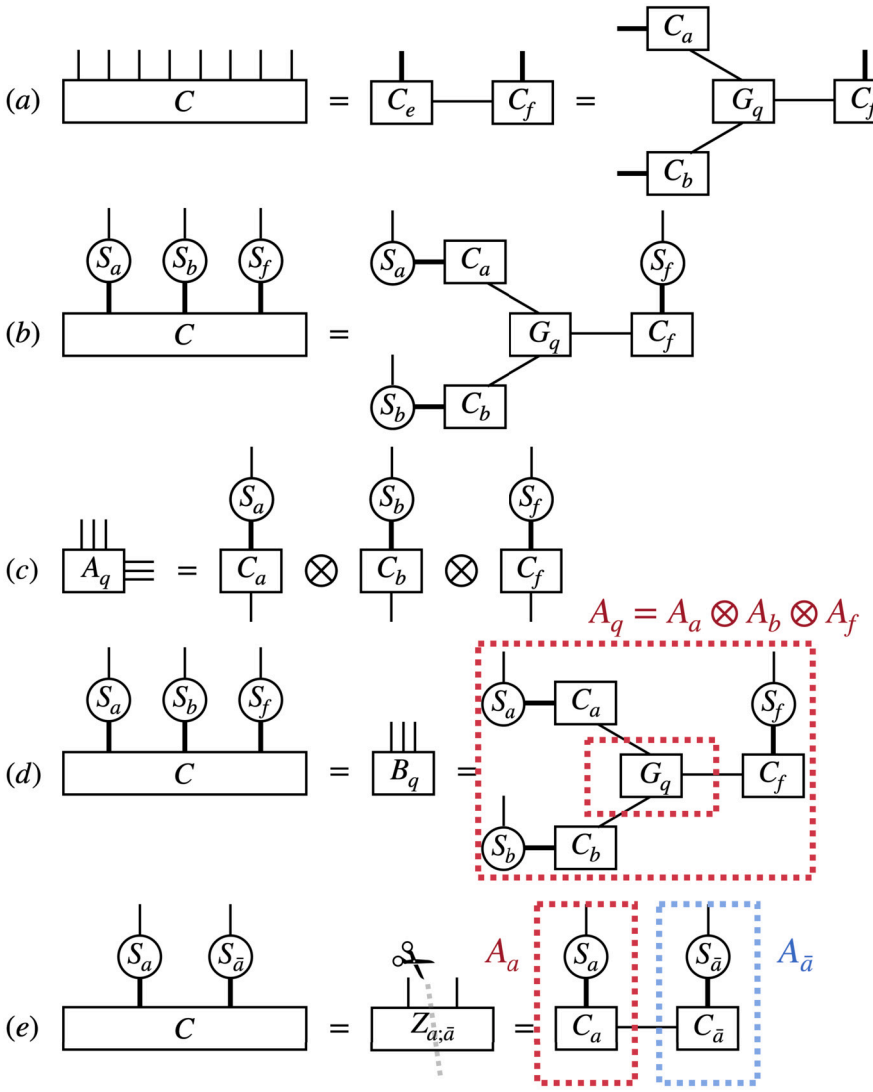


Fig. 3. Tensor diagram representation of main equations used in Section 2.3. Equation (7) and (8) are summarized in (a). Equation (9) is shown in (b). The coefficient term in (9) is shown in (c). The definition of B_q is shown in (d). Equation (10) for $Z_{a;\bar{a}}$ is shown in (e), where the scissor symbol indicates the rank-r_a compression through SVD.

$$B_q(\mu,\nu,\zeta) = \sum_{\alpha,\beta,\theta} A_a(\mu,\alpha)A_b(\nu,\beta)A_f(\zeta,\theta)G_q(\alpha,\beta,\theta), \tag{9}$$

where A_a, A_b, A_f are respectively the contraction of C_a, C_b, C_f by S_a, S_b, S_f and B_q is the contraction of C by $S_a \otimes S_b \otimes S_f$, as illustrated by the first equality of Fig. 3(d). The equation (9) can be seen as the following linear system of $\tilde{r}_a\tilde{r}_b\tilde{r}_f$ equations for the unknown G_q (illustrated in Fig. 3(d)):

$$(A_a \otimes A_b \otimes A_f)G_q = B_q.$$

While obtaining B_q in (9) is done by contracting known tensors, obtaining terms such as A_a is only possible after specifying the gauge degree of freedom for C_a. However, we show that one can obtain A_a without having access to C_a. The structural low-rankness of the ansatz implies the existence of $C_{\bar{a}}$ such that there exists a linear system $C(i_1,\dots,i_d) = \sum_\alpha C_a(i_a,\alpha)C_{\bar{a}}(\alpha,i_{\bar{a}})$. One can sketch this linear system by contracting it with $S_a, S_{\bar{a}}$, which leads to the following linear system (illustrated in Fig. 3(e)):

$$Z_{a;\bar{a}}(\mu,\phi) = \sum_\alpha A_a(\mu,\alpha)A_{\bar{a}}(\alpha,\phi), \tag{10}$$

where $A_{\bar{a}}$ is the contraction of $C_{\bar{a}}$ with $S_{\bar{a}}$. Then, due to a gauge degree of freedom in choosing C_a, there also exists a gauge degree of freedom in choosing A_a. Thus, one can perform singular value decomposition (SVD) on $Z_{a;\bar{a}}$, obtaining $Z_{a;\bar{a}}(\mu,\phi) = \sum_\alpha U(\mu,\alpha)V(\alpha,\phi)$, and the choice of $A_a = U$ and $A_{\bar{a}} = V$ forms a consistent choice of gauge between the pair $(C_a, C_{\bar{a}})$.

Likewise, one can obtain $Z_{b;\bar{b}}$ and the SVD of $Z_{b;\bar{b}}$ results in A_b and $A_{\bar{b}}$. The same holds for $Z_{\bar{f};f}$, A_f, and $A_{\bar{f}}$. To solve for G_q in the linear system (9), one simply contracts B_q with the pseudo-inverse of A_a, A_b, A_f. Thus, one can use (9) to solve for G_q for any $l \neq 0, L$.

Our construction likewise gives rise to the equation of G_q for $l = 0, L$ as special cases. For $l = 0$, one can go through with the same calculation by setting $C_f = S_f = A_f = 1$ in (8) and (9). For $l = L$, one can sketch the linear system in (8) for $f = [d] - \{k\}$ by contraction with S_f. The detail for both cases is described in Algorithm 1.

Approximation via samples We now describe how one can obtain a consistent estimation of G_q through samples. As before, suppose $\{\psi_i\}_{i=1}^n$ is the collection of univariate orthonormal function basis which defines the functional tensor network basis. To the sketch tensor $S_a : [n^{|a|}] \times [\tilde{r}_a] \to \mathbb{R}$, one associates a continuous sketch function $s_a : \mathbb{R}^{|a|} \times [\tilde{r}_a] \to \mathbb{R}$ through the following construction:

$$s_a(x_a, \mu) = \sum_{i_j, j \in a} S_a(i_a, \mu) \prod_{j \in a} \psi_{i_j}(x_j), \tag{11}$$

and one can likewise obtain $s_{\bar{a}}$ through $S_{\bar{a}}$. By definition, one has

$$Z_{a;\bar{a}}(\mu, \phi) = \sum_{i_j, j \in [d]} C(i_1, \ldots, i_d) S_a(i_a, \mu) S_{\bar{a}}(i_{\bar{a}}, \phi) \tag{12}$$

By the orthonormality of the function basis $\{\psi_i\}_{i=1}^n$, it follows that the transformation from a coefficient tensor to a function (see the definition in (5)) is an isometric embedding from $\mathbb{R}^{n^d}$ to $L^2(\mathbb{R}^d)$. Therefore, one has

$$Z_{a;\bar{a}}(\mu, \phi) = \int_{\mathbb{R}^d} p(x_1, \ldots, x_d) s_a(x_a, \mu) s_{\bar{a}}(x_{\bar{a}}, \phi)\, d\, x_1 \ldots x_d = \mathbb{E}_{X \sim p} \left[s_a(X_a, \mu) s_{\bar{a}}(X_{\bar{a}}, \phi) \right],$$

and the formulae for $Z_{b;\bar{b}}$ and $Z_{\bar{f};f}$ follow likewise. Similarly, one can obtain B_q in (9) by

$$B_q(\mu, \nu, \zeta) = \mathbb{E}_{X \sim p} \left[s_a(X_a, \mu) s_b(X_b, \nu) s_f(X_f, \zeta) \right].$$

Since $\{y^{(i)} \in \mathbb{R}^d\}_{i=1}^N$ provides an empirical approximation to p, one obtains a finite sample approximation of $Z_{a;\bar{a}}$ in the following way:

$$Z_{a;\bar{a}}(\mu, \phi) \approx \frac{1}{N} \sum_{i=1}^{N} s_a(y_a^{(i)}, \mu) s_{\bar{a}}(y_{\bar{a}}^{(i)}, \phi) \tag{13}$$

and likewise one can also obtain a sample estimation of B_q and thus approximately solve for G_q. It is easy to check that a consistent estimator of $Z_{a;\bar{a}}$ leads to a consistent estimator of A_a, and therefore one can likewise consistently estimate A_b, A_f, and thus the approximated solution is a consistent estimator for G_q.

Algorithm summary The sketching algorithm is summarized as Algorithm 1, which demonstrates the whole procedure to carry out the functional hierarchical tensor sketching algorithm given a sample collection, including the details for the omitted edge cases of $l = 0, L$.

3. Solving for Fokker-Planck equation

In this section, we detail our approach to solving the Fokker-Planck equation. The given information regarding the PDE includes the potential function $V : \mathbb{R}^d \to \mathbb{R}$, terminal time T, and initial distribution p_0. The main workflow is summarized in the following simple steps:

1. Select time steps $0 = t_0 \leq t_1 \leq \ldots \leq t_K = T$. Then, with a chosen numerical scheme (e.g. Euler-Maruyama), run N independent instance of SDE simulations on the stochastic dynamic equation (4) to time T with the initial condition p_0. Obtain N trajectories and record the trajectory data $\{X^{(i)}(t_j)\}_{j \in [K], i \in [N]}$.
2. For $j = 1, \ldots, K$, obtain an approximated functional hierarchical solution $p_j(x) \approx p(x, t_j)$ by calling Algorithm 1 on the sample $\{X^{(i)}(t_j)\}_{i \in [N]}$.
3. Output the continuous-in-time solution for $p(t, x)$, defined as follows: Suppose that $t \in [0, T]$ satisfies $t \in (t_j, t_{j+1})$, then $p(t, x)$ can be approximated with $p_j(x)$ and $p_{j+1}(x)$ via an appropriate interpolation.

Once the approximated solution p is obtained, one can perform sampling or compute observables based on the ansatz. We remark that the approximated density has its normalization constant approximately equal to one, as (9) automatically controls the scale for each tensor core so that the approximated solution is normalized. As the normalization constant is efficient to compute, one could also rescale the approximate density to ensure the normalization constant is exactly one.

As $p(t, x)$ interpolates between p_j and p_{j+1} for $t \in (t_j, t_{j+1})$, the task of sampling and computing observables reduces to performing such tasks on snapshot solutions p_j and p_{j+1}. Therefore, it suffices to illustrate how to perform such tasks on an obtained snapshot solution p_j.

Algorithm 1 Functional hierarchical tensor sketching.

Require: Sample $\{y^{(i)}\}_{i=1}^{N}$.
Require: Chosen function basis $\{\psi_j\}_{j=1}^{n}$.
Require: Collection of sketch functions $\{s_{I_k^{(l)}}\}, \{s_{[d]-I_k^{(l)}}\}$ and target internal ranks $\{r_{I_k^{(l)}}\}$ for each level index l and block index k.

for each node q on the hierarchical tree **do**
  Set l as the level index of q. Set k as the block index of q.
  if q is not leaf node **then**
    $(a, b) \leftarrow (I_{2k-1}^{(l+1)}, I_{2k}^{(l+1)})$
    Obtain $Z_{a;\bar{a}}, Z_{b;\bar{b}}$ by (13)
    Obtain A_a as the left factor of the best rank r_a factorization of $Z_{a;\bar{a}}$
    Obtain A_b as the left factor of the best rank r_b factorization of $Z_{b;\bar{b}}$.
  end if
  if q is not root node **then**
    $f \leftarrow [d] - I_k^{(l)}$
    Obtain $Z_{\bar{f};f}$ by (13).
    Obtain A_f as the right factor of the best rank r_f factorization of $Z_{\bar{f};f}$.
  end if
  if q is root node **then**
    Obtain $B_q(\mu, \nu) = \frac{1}{N}\sum_i s_a(y_a^{(i)}, \mu) s_b(y_b^{(i)}, \nu)$.
    Obtain G_q by solving the over-determined linear system $(A_a \otimes A_b)G_q = B_q$.
  else if q is leaf node **then**
    Obtain $B_q(j, \zeta) = \frac{1}{N}\sum_i \psi_j(y_k^{(i)}) s_f(y_f^{(i)}, \zeta)$.
    Obtain G_q by solving the over-determined linear system $(A_f)G_q = B_q$.
  else
    Obtain $B_q(\mu, \nu, \zeta) = \frac{1}{N}\sum_i s_a(y_a^{(i)}, \mu) s_b(y_b^{(i)}, \nu) s_f(y_f^{(i)}, \zeta)$.
    Obtain G_q by solving the over-determined linear system $(A_a \otimes A_b \otimes A_f)G_q = B_q$.
  end if
end for

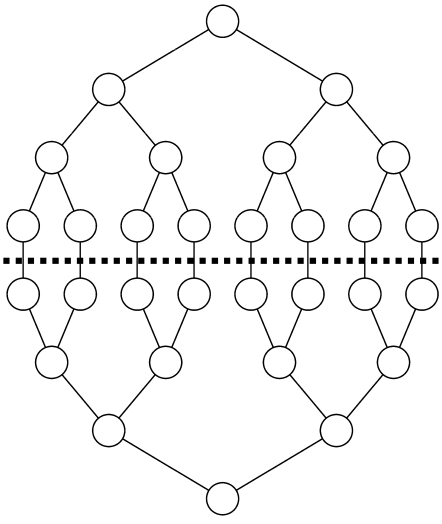


Fig. 4. Tensor diagram representation for the measuring observable $\mathbb{E}_{X\sim p_j}[O(X)]$. The top half of the tensor cores represents the tensor network for p_j, and the bottom half represents the tensor network for O.

Application I: observable estimation For an observable function $O: \mathbb{R}^d \to \mathbb{R}$, note that one has $\mathbb{E}_{X\sim p_j}[O(X)] = \left\langle O(x), p_j(x)\right\rangle_{L^2(\mathbb{R}^d)}$. When O can be written in terms of a finite sum of a monomial over the basis functions, one can compute the inner product efficiently through a tensor contraction, as the orthonormality of the function basis implies the function space inner product will coincide with the inner product between the coefficient tensor. In general, when O is a functional hierarchical tensor, one can compute $\left\langle O(x), p_j(x)\right\rangle_{L^2(\mathbb{R}^d)}$ through tensor diagram contraction, as is shown in Fig. 4. The tensor contraction diagram in Fig. 4 admits efficient evaluation through iterative tensor contractions starting from the middle two layers.

Application II: sample generation The counting-to-sampling [41] perspective reduces the sampling task to the previous one for observable estimation. Using conditional distributions, we have the following decomposition of p_j at time t_j

$$p_j(x_1, \ldots, x_d) = p_j(x_1) \cdot p_j(x_2|x_1) \ldots p_j(x_d|x_1, \ldots, x_{d-1}).$$

One then uses sequential Monte Carlo (see [29] for details) to generate i.i.d. samples: For $k \in [d]$, suppose one has sampled the distribution up to index $k-1$ with $x_i = a_i$ for $i = 1, \dots, k-1$. One can sample the k-th variable through $X_k \sim p_j(x_k = x | x_{[k-1]} = a_{[k-1]})$, where $p_j(x_k = x | x_{k-1} = a_{k-1}, \dots, x_1 = a_1) = \mathbb{E}_{X \sim p_j}[\prod_{i \in [k-1]} \delta(X_i - a_i)\delta(X_k - x))]$.

3.1. Extensions

Though we have been focused on Langevin dynamics for simplicity, our method can be generalized to a wider range of settings. We shall remark a few notable extensions to the current approach.

Fokker-Planck beyond Langevin dynamics First, one can readily see that the approach is agnostic to the simulated SDE structure considered. In particular, one can consider the generalized Fokker-Planck equation,

$$\partial_t p = \nabla \cdot (pf) + \sum_{i,j,k=1}^{d} \partial_{x_i x_j} (\sigma_{ik}\sigma_{jk} p), \quad x \in \Omega \subset \mathbb{R}^d, t \in [0, T],$$

for a general particle system with the following form:

$$dX_t = -f(X_t, t)dt + \sigma(X_t, t)dB_t, \tag{14}$$

where $f : \mathbb{R}^d \times [0, T] \to \mathbb{R}^d$ and $\sigma : \mathbb{R}^d \times [0, T] \to \mathbb{R}^{d \times d}$ are general advection and diffusion terms. While Algorithm 1 does not depend on the exact method used to generate the sample trajectory, it might be advisable to use a higher-order SDE simulation to guarantee sample quality in simulating (14).

Non-linear Fokker-Planck equations Secondly, one can also extend our method to nonlinear Fokker-Planck equations. Using the perspective that the Fokker-Planck equation is the Wasserstein gradient flow on an Energy potential [42], one can consider a functional

$$E(p) + \beta^{-1} \int p(x) \ln p(x) dx$$

with a non-linear $E(p)$. The associated PDE is

$$\partial_t p(t, x) = \nabla \cdot (p \nabla \frac{\delta E}{\delta p}) + \beta^{-1} \Delta p,$$

where $\frac{\delta E}{\delta p}$ is the Frechet derivative of E with respect to p. The particle dynamics is

$$dX = -(\nabla \frac{\delta E}{\delta p})(X) + \sqrt{2\beta^{-1}} dB_t$$

with the drift term now depends on p. Our approach can be readily extended to such nonlinear Fokker-Planck equations if one simulates multiple trajectories jointly with an ensemble method (for detail, see [29]).

4. Application to Ginzburg-Landau model

Our main numerical treatment is dedicated to a discretized Ginzburg-Landau model in varying physical dimensions. The high dimensionality here comes from the discretization of an infinite-dimensional functional equation [43]. For the sake of simplicity, we apply functional hierarchical tensor density estimation only at time $T = 1$. Generally, one runs the algorithm at multiple time slots to solve for the solution across $[0, T]$.

The choice of basis function For the Ginzburg-Landau model, the $\int_a |1 - (x(a))^2| da$ term ensures that it is highly unlikely for $\|x\|_\infty$ to be significantly bigger than 1. For the discretized model, each variable x_j is essentially bounded within an interval $[-B, B]$ with $B = 2.5$. As a result, the samples can be effectively seen as a distribution compactly supported in the domain $[-B, B]^d$. Therefore, we can use the Fourier representation choosing a maximal degree parameter q and picking the first $n = 2q + 1$ Fourier basis functions in the sine-cosine form in $[-B, B]$, denoted $\{\psi_i\}_{i=-q}^{q}$. In effect, if one denotes $p^\star$ as the target density at time $t \in [0, T]$, then our target is to approximate $\mathcal{P}_q p^\star$, which denotes the truncated Fourier series approximation of $p^\star$ restricted to degree q. This relationship is expressed as follows:

$$(\mathcal{P}_q p^\star)(x_1, \dots, x_d) = \sum_{i_1, \dots, i_d = -q}^{q} C^\star_{i_1, \dots, i_d} \psi_{i_1}(x_1) \cdots \psi_{i_d}(x_d), \tag{15}$$

where $C^\star \in \mathbb{R}^{n^d}$ is the coefficient tensor. When the diffusion term in the Fokker-Planck equation has a strong influence, the spectral approximation in (15) provides an accurate approximation with a small q. The goal is then to approximate $p^\star$ through a compressed hierarchical tensor $C \in \mathbb{R}^{n^d}$:

$$p^\star \approx \mathcal{P}_q p^\star \approx p_{\mathrm{FHT}}, \quad p_{\mathrm{FHT}}(x_1,\dots,x_d) = \sum_{i_1,\dots,i_d=-q}^{q} C_{i_1,\dots,i_d}\psi_{i_1}(x_1)\cdots\psi_{i_d}(x_d). \tag{16}$$

Bipartition structure Similar to the proposed hierarchical bipartition for the 2D case, we recursively bipartitions the Cartesian grid along its axes in an alternating fashion. The structure for the two-dimensional case is shown in Fig. 1 and can be done likewise to other dimensions. Explicitly, assume for simplicity that a function u in Δ-dimensions is discretized to the $d = m^\Delta$ points denoted $\{u_{i_1,\dots,i_\Delta}\}_{i_1,\dots,i_\Delta\in[m]}$ with $m = 2^\mu$. For each index i_δ, one performs a length μ binary expansion $i_\delta = a_{\delta 1}\dots a_{\delta\mu}$. The variable $u_{i_1,\dots,i_\Delta}$ is given index k with the length $\mu\Delta$ binary expansion $k = a_{11}a_{21}\dots a_{\Delta-1,\mu}a_{\Delta,\mu}$, and the associated binary decomposition structure follows from the given index according to (6). One can check that the construction exactly corresponds to performing binary partition by sweeping along the Δ dimensions. For intuition on this construction, one can check that the bipartition in Fig. 1 corresponds to the indexing $k = a_{11}a_{21}a_{12}a_{22}a_{13}a_{23}$.

The choice of sketch function In practice, one does not have ready access to a pre-determined sketch function, and a higher quality in the sketch function would mean that one would be better able to capture the spatial correlation of the density function. As can be seen in (13), one needs to choose relatively smooth sketch functions to ensure that an accurate approximation of terms is possible with sample estimation. Moreover, the sketch functions should be designed to capture important features so that terms such as $Z_{a;\bar{a}}$ could provide meaningful linear systems to solve for G_q.

In our work, the sketch functions we choose are of two kinds. The first kind of sketch function comprises monomials of the Fourier basis of low degree. For $a = I_k^{(l)}$ or $a = [d] - I_k^{(l)}$ (see definition in (6)), the chosen sketch functions are of the following form:

$$f(x_a) = \prod_{j\in a}\psi_{i_j}(x_j),$$

for some choice of $-q \le i_j \le q$ for each j. The total Fourier degree can be characterized by $\deg(f) = \sum_{j\in a}|i_j|$, and one typically chooses f with small $\deg(f)$.

The second kind of sketch function is motivated by the renormalization group, in particular by the coarse-graining of the variables. In particular, one chooses a cluster $h \subset a$ and a specific Fourier mode index i. The resultant function $g_{h,i}$ is an averaging of the i-th Fourier mode over variables in h:

$$g_{h,i}(x_a) \equiv \frac{1}{|h|}\sum_{j\in h}\psi_i(x_j).$$

In our particular implementation, for $a = I_k^{(l)}$, a typical example of chosen sketch function would be $h = I_{2k}^{(l+1)}$ and $i = 1$. Then, the resultant sketch function would be

$$g_{h,1}(x_a) \equiv \frac{1}{|h|}\sum_{j\in h}\sin(\pi x_j/B),$$

which can be thought of as coarse-grained information on half of the variables in a. One can similarly define $g_{a-h,1}(x_a)$. One can augment the sketch function set by including terms such as $g_{h,1}^2$ and $g_{a-h,1}^2$. Empirically, including such coarse-grained low-order Fourier modes greatly increases model performance and stability.

4.1. 1D Ginzburg-Landau potential

The first numerical example is a 1D Ginzburg-Landau model. The potential energy is defined as

$$V(x_1,\dots,x_m) := \frac{\lambda}{2}\sum_{i=1}^{m+1}\left(\frac{x_i - x_{i-1}}{h}\right)^2 + \frac{1}{4\lambda}\sum_{i=1}^{m}\left(1 - x_i^2\right)^2, \tag{17}$$

where $h = \frac{1}{m+1}$ and $x_0 = x_{m+1} = 0$. In particular, we fix $m = 256$, $\lambda = 0.01$ and $\beta = \frac{1}{8}$. The initial condition is $p_0(x) = \delta(x - (0,\dots,0))$ and the dimension d is 256.

We perform $N = 6000$ SDE simulations with $T = 1$ and $\delta t = \frac{1}{2000}$, all starting from the initial location $(0,\dots,0)$. The maximal Fourier degree q is taken to be $q = 15$ to account for the localization of the samples, and the maximal internal bond dimension is $r = 6$.

Fig. 5 compares the marginal distribution of (x_{15}, x_{59}) between the empirical distribution and the model obtained by hierarchical tensor sketching. The hierarchical tensor sketching is successful at capturing the correlation between two variables at relatively faraway points.

4.2. 2D Ginzburg-Landau potential

In the second numerical example, we consider the usual 2D Ginzburg-Landau model. The potential energy is defined as

$$V(x_{(1,1)},\dots,x_{(m,m)}) := \frac{\lambda}{2}\sum_{v\sim w}\left(\frac{x_v - x_w}{h}\right)^2 + \frac{1}{4\lambda}\sum_{v}\left(1 - x_v^2\right)^2, \tag{18}$$

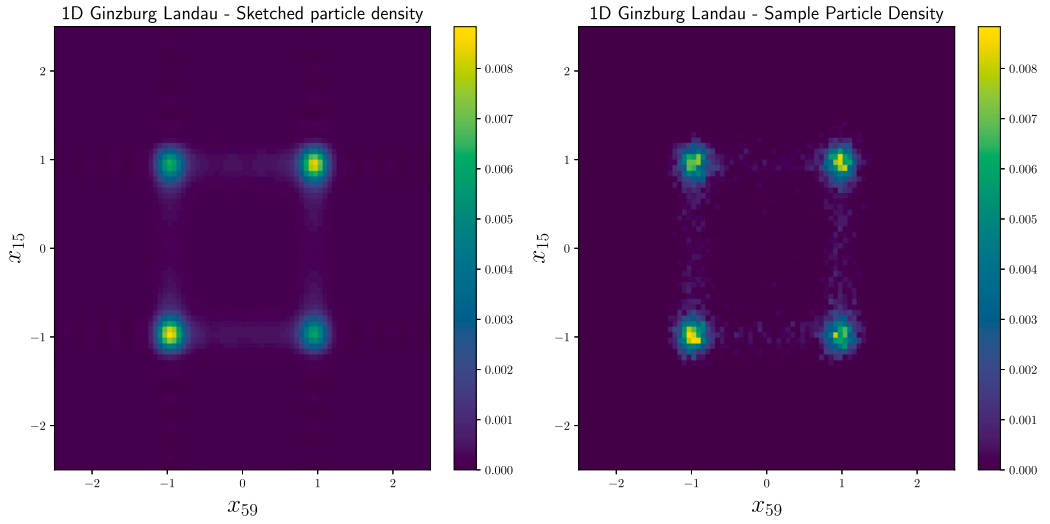


Fig. 5. 1D Ginzburg-Landau model. The plots of the marginal distribution at (x_{19}, x_{59}).

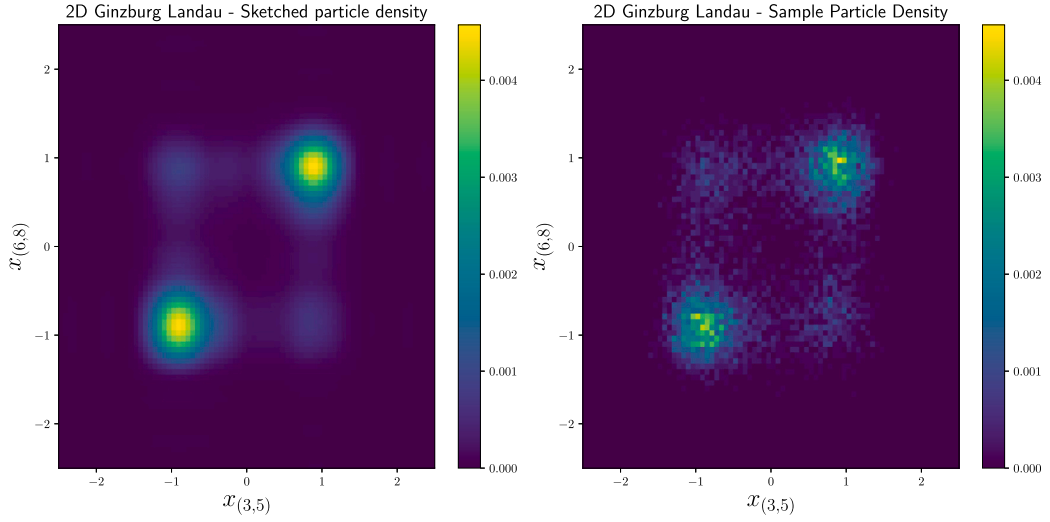


Fig. 6. 2D Ginzburg-Landau model. The plots of marginal distribution at $(x_{(3,5)}, x_{(6,8)})$.

where $h = \frac{1}{m+1}$ and $x_{(l,0)} = x_{(l,m+1)} = x_{(0,l)} = x_{(m+1,l)} = 0$ for $l = 1, \ldots, m$. The parameters are $m = 16$, $\lambda = 0.03$, $\beta = \frac{1}{5}$, and the initial condition is $p_0(x) = \delta(x - (0, \ldots, 0))$. The dimension d is 256.

We perform $N = 6000$ SDE simulation with $T = 1$ and $\delta t = 0.003$, starting from the initial condition $(0, \ldots, 0)$. We take $q = 10$ and a maximal internal bond of $r = 20$. To ensure numerical stability, the internal bond dimension is dynamically chosen according to the singular values information obtained during sketching.

Fig. 6 shows that the obtained marginal distribution of $(x_{(3,5)}, x_{(6,8)})$ closely matches the empirical distributions. To test the accuracy of observable estimation, Fig. 7 plots the predicted two-point correlation function

$$f(i,j) := \mathrm{Corr}\left(x_{(8,8)}, x_{(i,j)}\right) \equiv \frac{\mathrm{Cov}\left[x_{(8,8)}, x_{(i,j)}\right]}{\sigma_{x_{(8,8)}} \sigma_{x_{(i,j)}}},$$

where the result closely matches the ground truth obtained from a Monte-Carlo estimation from 60,000 samples. The accuracy is quite remarkable as the internal bond r is kept at a relatively small level.

4.3. 3D Ginzburg-Landau potential

Here, we consider a 3D Ginzburg-Landau model. Similarly, define a 3D Cartesian grid $D = (Q, E)$, where $Q := \{(i,j,k) \mid i,j,k = 0, \ldots, m+1\}$. The potential energy is defined as

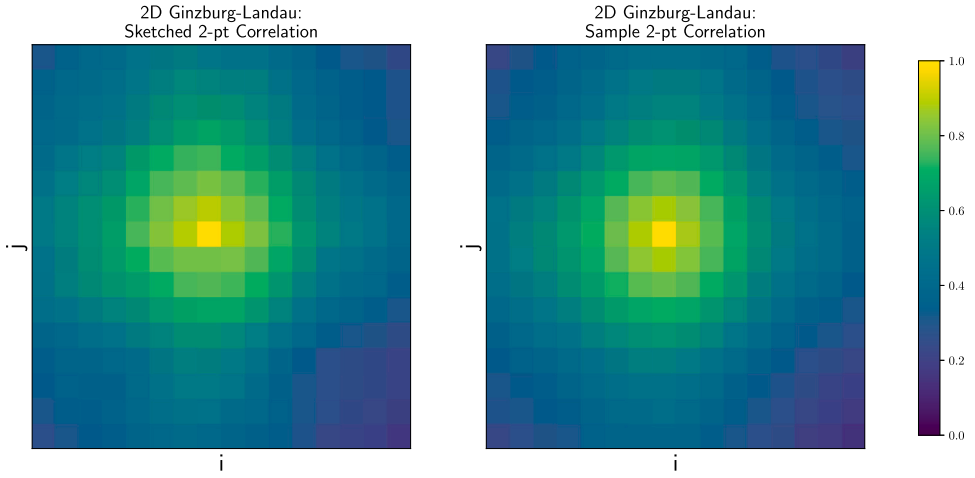


Fig. 7. Two-point correlation function $f(i,j) = \mathrm{Corr}\left(x_{(8,8)}, x_{(i,j)}\right)$ of the sketched functional hierarchical tensor, compared with the ground truth from a Monte-Carlo estimation from 60,000 samples.

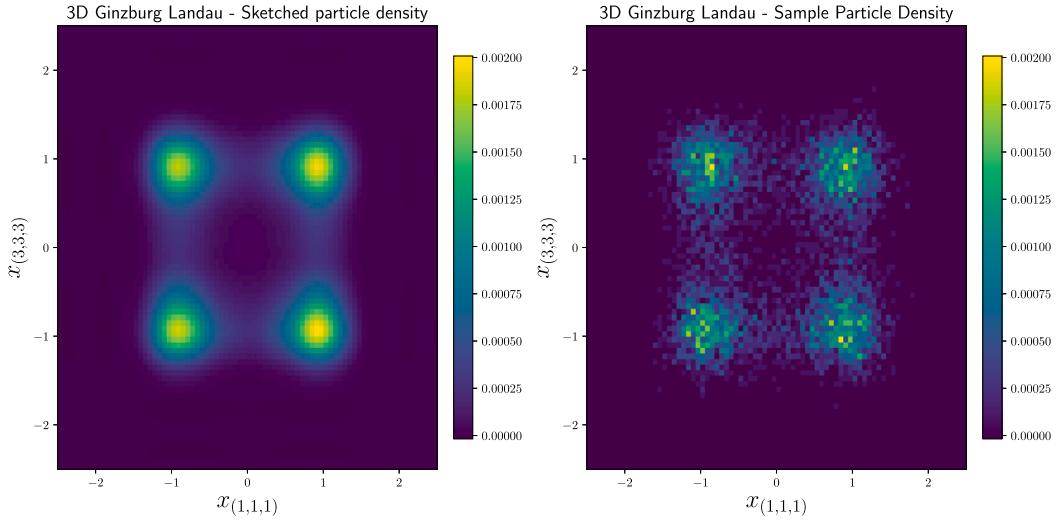


Fig. 8. 3D Ginzburg-Landau model. The plots of marginal distribution at $(x_{(1,1,1)}, x_{(3,3,3)})$.

$$V(x_{(1,1,1)}, \ldots, x_{(m,m,m)}) := \frac{\lambda}{2} \sum_{v \sim w} \left(\frac{x_v - x_w}{h}\right)^2 + \frac{1}{4\lambda} \sum_{v} \left(1 - x_v^2\right)^2, \tag{19}$$

where $h = \frac{1}{m+1}$, and the boundary condition is such that $x_{(i,j,k)} = 0$ if one of i, j, k equals to 0 or $m+1$. For parameters, we fix $m = 8$, $\lambda = 0.01$ and $\beta = \frac{1}{10}$. The initial condition is $p_0(x) = \delta(x - (0, \ldots, 0))$ and the dimension d is 512.

We perform $N = 6000$ SDE simulation with $T = 1$ and $\delta t = 0.002$, starting from the initial condition $(0, \ldots, 0)$. We take $q = 10$ and a maximal internal bond of $r = 20$, where the internal bond is likewise chosen dynamically. Fig. 8 shows the performance on the marginal distribution of $(x_{(1,1,1)}, x_{(3,3,3)})$ between empirical distributions and the model obtained by hierarchical tensor sketching, and similarly the result shows quite good performance.

5. Conclusion

We introduce a novel density estimation approach that combines particle methods with a functional hierarchical tensor network in solving the high-dimensional Fokker-Planck equations. The algorithm is applied to the discretized Ginzburg-Landau model in 1D, 2D, and 3D. This method points to a new direction for tackling general high-dimensional Fokker-Planck equations. Combined with sensible numerical treatment, this approach has the potential for modeling high-dimensional particle dynamics with high fidelity. An open question is whether one can extend a tensor network approach to solving similar equations, such as the Kolmogorov backward equation and the Hamilton-Jacobi equation.

CRediT authorship contribution statement

Xun Tang: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Lexing Ying:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgement

We thank Yuehaw Khoo for the discussions and the reviewers for constructive comments.

References

[1] J. Han, A. Jentzen, W. E, Solving high-dimensional partial differential equations using deep learning, Proc. Natl. Acad. Sci. 115 (2018) 8505–8510.
[2] V.L. Ginzburg, V.L. Ginzburg, L. Landau, On the Theory of Superconductivity, Springer, 2009.
[3] K.-H. Hoffmann, Q. Tang, Ginzburg-Landau Phase Transition Theory and Superconductivity, vol. 134, Birkhäuser, 2012.
[4] P.C. Hohenberg, A.P. Krekhov, An introduction to the Ginzburg–Landau theory of phase transitions and nonequilibrium patterns, Phys. Rep. 572 (2015) 1–42.
[5] W. E, W. Ren, E. Vanden-Eijnden, Minimum action method for the study of rare events, Commun. Pure Appl. Math. 57 (2004) 637–656.
[6] T. Hastie, R. Tibshirani, J.H. Friedman, J.H. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, vol. 2, Springer, 2009.
[7] Y. Chen, Y. Khoo, Combining particle and tensor-network methods for partial differential equations via sketching, preprint, arXiv:2305.17884, 2023.
[8] Y. Ren, H. Zhao, Y. Khoo, L. Ying, High-dimensional density estimation with tensorizing flow, Res. Math. Sci. 10 (2023) 30.
[9] A. Dektor, D. Venturi, Dynamic tensor approximation of high-dimensional nonlinear pdes, J. Comput. Phys. 437 (2021) 110295.
[10] A. Dektor, A. Rodgers, D. Venturi, Rank-adaptive tensor methods for high-dimensional nonlinear pdes, J. Sci. Comput. 88 (2021) 36.
[11] M.B. Soley, P. Bergold, A.A. Gorodetsky, V.S. Batista, Functional tensor-train Chebyshev method for multidimensional quantum dynamics simulations, J. Chem. Theory Comput. 18 (2021) 25–36.
[12] W. Hackbusch, S. Kühn, A new scheme for the tensor representation, J. Fourier Anal. Appl. 15 (2009) 706–722.
[13] W. Hackbusch, Tensor Spaces and Numerical Tensor Calculus, vol. 42, Springer, 2012.
[14] A. Gorodetsky, S. Karaman, Y. Marzouk, A continuous analogue of the tensor-train decomposition, Comput. Methods Appl. Mech. Eng. 347 (2019) 59–84.
[15] Y. Hur, J.G. Hoskins, M. Lindsey, E.M. Stoudenmire, Y. Khoo, Generative modeling via tensor train sketching, Appl. Comput. Harmon. Anal. 67 (2023) 101575.
[16] S. Östlund, S. Rommer, Thermodynamic limit of density matrix renormalization, Phys. Rev. Lett. 75 (1995) 3537.
[17] J.I. Cirac, D. Perez-Garcia, N. Schuch, F. Verstraete, Matrix product states and projected entangled pair states: concepts, symmetries, theorems, Rev. Mod. Phys. 93 (2021) 045003.
[18] G. Vidal, Entanglement renormalization, Phys. Rev. Lett. 99 (2007) 220405.
[19] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[20] J. Sirignano, K. Spiliopoulos, Dgm: a deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1339–1364.
[21] B. Yu, et al., The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (2018) 1–12.
[22] N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, A. Anandkumar, Neural operator: learning maps between function spaces, preprint, arXiv:2108.08481, 2021.
[23] E.G. Tabak, E. Vanden-Eijnden, Density estimation by dual ascent of the log-likelihood, Commun. Math. Sci. 8 (2010) 217–233.
[24] D. Rezende, S. Mohamed, Variational inference with normalizing flows, in: International Conference on Machine Learning, PMLR, 2015, pp. 1530–1538.
[25] G.E. Hinton, Training products of experts by minimizing contrastive divergence, Neural Comput. 14 (2002) 1771–1800.
[26] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, F. Huang, A tutorial on energy-based learning, in: Predicting Structured Data, 2006.
[27] Y. Song, S. Ermon, Generative modeling by estimating gradients of the data distribution, Adv. Neural Inf. Process. Syst. 32 (2019).
[28] Y. Song, C. Durkan, I. Murray, S. Ermon, Maximum likelihood training of score-based diffusion models, Adv. Neural Inf. Process. Syst. 34 (2021) 1415–1428.
[29] J. Liu, Monte Carlo Strategies in Scientific Computing, vol. 75, Springer, 2001.
[30] Y. Chen, J. Hoskins, Y. Khoo, M. Lindsey, Committor functions via tensor networks, J. Comput. Phys. 472 (2023) 111646.
[31] R. Schneider, A. Uschmajew, Approximation rates for the hierarchical tensor format in periodic Sobolev spaces, J. Complex. 30 (2014) 56–71.
[32] B.N. Khoromskij, C. Schwab, Tensor-structured Galerkin approximation of parametric and stochastic elliptic pdes, SIAM J. Sci. Comput. 33 (2011) 364–385.
[33] D. Bigoni, A.P. Engsig-Karup, Y.M. Marzouk, Spectral tensor-train decomposition, SIAM J. Sci. Comput. 38 (2016) A2405–A2439.
[34] M. Eigel, J. Neumann, R. Schneider, S. Wolf, Non-intrusive tensor reconstruction for high-dimensional random pdes, Comput. Methods Appl. Math. 19 (2019) 39–53.
[35] M. Bachmayr, R. Schneider, A. Uschmajew, Tensor networks and hierarchical tensors for the solution of high-dimensional partial differential equations, Found. Comput. Math. 16 (2016) 1423–1472.
[36] L. Richter, L. Sallandt, N. Nüsken, Solving high-dimensional parabolic pdes using the tensor train format, in: International Conference on Machine Learning, PMLR, 2021, pp. 8998–9009.
[37] S. Cheng, L. Wang, T. Xiang, P. Zhang, Tree tensor networks for generative modeling, Phys. Rev. B 99 (2019) 155131.
[38] X. Tang, Y. Hur, Y. Khoo, L. Ying, Generative modeling via tree tensor network states, Res. Math. Sci. 10 (2023) 19.
[39] E. Grelier, A. Nouy, R. Lebrun, Learning high-dimensional probability distributions using tree tensor networks, preprint, arXiv:1912.07913, 2019.
[40] Y. Peng, Y. Chen, E.M. Stoudenmire, Y. Khoo, Generative modeling via hierarchical tensor sketching, preprint, arXiv:2304.05305, 2023.
[41] M. Jerrum, Counting, Sampling and Integrating: Algorithms and Complexity, Springer Science & Business Media, 2003.
[42] R. Jordan, D. Kinderlehrer, F. Otto, The variational formulation of the Fokker–Planck equation, SIAM J. Math. Anal. 29 (1998) 1–17.
[43] G. Rosen, Functional calculus theory for incompressible fluid turbulence, J. Math. Phys. 12 (1971) 812–820.