COMMUN. MATH. SCI.
Vol. 20, No. 3, pp. 803–827

AN EFFICIENT DATA-DRIVEN SOLVER FOR FOKKER-PLANCK EQUATIONS: ALGORITHM AND ANALYSIS[*]

MATTHEW DOBSON[†], YAO LI[‡], AND JIAYU ZHAI[§]

Abstract. Computing the invariant probability measure of a randomly perturbed dynamical system usually means solving the stationary Fokker-Planck equation. This paper studies several key properties of a novel data-driven solver for low-dimensional Fokker-Planck equations proposed in [Y. Li, Commun. Math. Sci., 17(4):1045–1059, 2019]. Based on these results, we propose a new "block solver" for the stationary Fokker-Planck equation, which significantly improves the performance of the original algorithm. Some possible ways of reducing numerical artifacts caused by the block solver are discussed and tested with examples.

Keywords. Fokker-Planck equation; Monte Carlo simulation; data-driven method.

AMS subject classifications. 65N99; 65C05; 37M25.

1. Introduction

Random perturbations to deterministic dynamical systems are ubiquitous in models used in physics, biology and engineering. The steady state of a randomly perturbed dynamical system is of critical interest in the study of these physical, biological or chemical systems and their applications. From a dynamical systems point of view, the interplay of dynamics and noise is both interesting and challenging, especially if the underlying dynamics is chaotic. Characteristics of the steady state distribution also help us to understand asymptotic effects of random perturbations to deterministic dynamics.

The evolution of the probability density function of a randomly perturbed system is described by the Fokker-Planck equation [21]. Consider a stochastic dynamical system

$$\mathrm{d}X_t = f(X_t)\mathrm{d}t + \sigma(X_t)\mathrm{d}W_t, \tag{1.1}$$

where f is a vector field in $\mathbb{R}^n$, σ is a coefficient matrix, and $\mathrm{d}W_t$ is an n-dimensional white noise. The corresponding Fokker-Planck equation, which is also known as the Kolmogorov forward equation, is

$$u_t = \mathcal{L}u = -\sum_{i=1}^{n}(f_i u)_{x_i} + \frac{1}{2}\sum_{i,j=1}^{n}(D_{i,j}u)_{x_i x_j}, \tag{1.2}$$

where $D = \sigma^T\sigma$, $u(x,t)$ denotes the probability density at time t, and subscripts t and x_i denote partial derivatives. In this paper, we focus on the invariant probability measure of (1.1), whose density function satisfies the stationary Fokker-Planck equation

$$\mathcal{L}u = 0 \qquad \int_\Omega u\,dx = 1.$$

Detailed assumptions about Equations (1.1) and (1.2) will be given in Section 2.1.

*Received: January 30, 2021; Accepted (in revised form): September 13, 2021. Communicated by Jianfeng Lu.

†Department of Mathematics and Statistics, University of Massachusetts Amherst, Amherst, MA, 01002, USA (dobson@math.umass.edu).

‡Department of Mathematics and Statistics, University of Massachusetts Amherst, Amherst, MA, 01002, USA (yaoli@math.umass.edu).

§Department of Mathematics and Statistics, University of Massachusetts Amherst, Amherst, MA, 01002, USA (zhai@math.umass.edu).

For Langevin dynamics, the invariant probability measure is given by the Gibbs distribution which can be computed up to the unknown normalizing constant; however, in general, the Fokker-Planck equation can not be solved analytically. Rigorous estimations of the invariant probability density function are challenging as well. Most known results are proved by large deviations techniques [8], which unfortunately only shows tail properties when the noise is asymptotically small. Some concentration properties of the invariant probability measure can be proved by assuming some dissipative conditions. For example, it was shown in [4, 16] that such concentration in the vicinity of a strong attractor is "Gaussian-like". However, these theoretical results can rarely give a satisfactory quantitative description of the invariant probability measure. Therefore, numerical solution techniques are necessary to further study these randomly perturbed dynamical systems. Numerically solving a steady state Fokker-Planck equation in an unbounded domain is nontrivial, and additional challenges are present in systems with high dimensionality, chaotic underlying dynamics, and multiscale coefficient terms.

One difficulty of solving the Fokker-Planck equation numerically is the conflict between the need for high-resolution local solutions and the necessity to handle large spatial domains. On one hand, in many applications, what we need is a high-resolution local numerical solution. It is known that the invariant probability measure tends to concentrate at the vicinity of the global attractor, and for such systems we are interested in the distribution in a local region of the phase space. In addition, if the strength of noise σ is small, it is proved that the probability density function is concentrated in an $O(\sigma)$ neighborhood of the global attractor [16]. So in order to obtain a meaningful solution and to avoid numerical artifacts, the grid size needs to be small enough. On the other hand, the Fokker-Planck equation in $\mathbb{R}^d$ is defined on an unbounded domain with zero value at infinity. The lack of a local boundary condition makes the problem computationally challenging. The existing methods usually solve the Fokker-Planck equation in a region that is large enough to cover all attractors.

In [15], a hybrid method is proposed to partially resolve the difficulties. The method deals with the local Fokker-Planck equation and completely removes the unknown boundary condition, which makes the resultant linear system underdetermined. To solve this underdetermined problem, Monte-Carlo simulation is used to provide a reference solution for the numerical solver (finite difference or finite element). The reference solution itself has low accuracy and lots of undesired fluctuations. The algorithm then projects the "noisy" reference solution onto the kernel of the discretized numerical solver. This minimizes the distance between the collection of possible numerical solutions (without knowing the boundary condition) and the reference solution from the Monte Carlo simulation. This method can solve the problem in any local area even if it doesn't cover any attractor. It also smooths the oscillation caused by the Monte Carlo sampling. By reducing the computational cost from non-locality, it can provide a high resolution solution in a local area.

Paper [15] only introduced the algorithm without proof. Analysis of this algorithm is carried out in this paper. We prove that the hybrid method introduced in [15] can significantly reduce the error of the reference solution produced by the Monte Carlo simulation. The heuristic reason is that the error term of this random reference solution is very close to an i.i.d. random vector. The expected norm of this random vector is dramatically reduced when projecting it to a lower dimensional subspace. In addition, we use a combination of rigorous analysis and numerical computations to show that the error term of this projection concentrates on the boundary of the numerical domain. In other words, the empirical performance of the hybrid algorithm is actually much better than what can be rigorously proved. In numerical examples, we show that our Fokker-Planck solver can tolerate very high level of spatially uncorrelated error in the reference solution. Hence the algorithm does not require a large sample size, but it is important to choose the correct Monte Carlo sampler that has low spatial correlation.

The other goal of this paper is to improve the performance of this hybrid method by introducing block solvers. This improvement is motivated by the locality of the hybrid method. Since the hybrid method does not rely on local boundary conditions, we can divide the numerical domain into a large number of small blocks and apply the hybrid method to each block. The global solution is a collage of local solutions on these blocks. This divide-and-conquer strategy is very efficient. Consider a d-dimensional problem with N grid points in each dimension. The classical numerical PDE solver needs to solve a large linear system with N^d variables. Assume the cost of solving a linear system with n variables is $O(n^p)$. Then the total cost is $O(N^{dp})$, which is considerably large if for instance $d=3$ and $N=1000$. However, if we divide the grid into many blocks with only m grid points in each dimension. The total cost of solving the Fokker-Planck equation on $(N/m)^d$ blocks becomes $m^{pd}\times(N/m)^d=m^{(p-1)d}N^d$. Empirically m can be as small as $20-30$. This dramatically reduces the total computational cost, unless the linear solver can achieve a linear complexity (which usually does not happen). In addition, parallelizing these block solvers is much easier than computing a large linear system in parallel. Instead of a local solution in a small subset of the phase space demonstrated in [15], the block solver now allows us to compute the full invariant probability density function of 3D or 4D systems, as demonstrated later in this paper.

The idea of using local blocks is supported by our analytical results in the first half of this paper. Theoretically, using larger blocks gives better reduction of error terms from Monte Carlo simulations, as proved in Theorem 2.1. But the analysis in this paper shows that the error tends to concentrate at the boundary of blocks. Hence the size of blocks does not need to be very large to make the accuracy of solutions in the interior of blocks acceptable, and the error on the boundary can be repaired by algorithms. Since the error of numerical solution mainly concentrates on the boundary, a naive block solver has visible interface errors between blocks. We then develop methods to reduce this interface error. Two different approaches, namely the overlapping blocks method and the shifting blocks method, are introduced and tested with several examples.

In this paper, we mainly consider low-dimensional systems up to dimension 3 or 4, where traditional grid-based numerical methods still work. For systems in much higher dimensions, all traditional grid-based methods of solving the Fokker-Planck equation, such as finite difference method or finite elements method, are not feasible any more. Direct Monte Carlo simulation also greatly suffers from the curse-of-dimensionality. There are several techniques introduced to deal with certain multi-dimensional Fokker-Planck equations, such as the truncated asymptotic expansion, splitting method, orthogonal functions, and tensor decompositions [6,7,17,22–24]. These methods usually need some additional assumptions on the solution, and error analyses for these methods are difficult in general. In addition, many of those methods still need a large domain to use a zero boundary condition. It is worth mentioning that an efficient high-dimensional sampling technique is introduced in [2,3]. This method works well for a class of stochastic differential equations in which most variables are conditionally linear with respect to a few nonlinear variables. The main idea is to use the conditional Gaussian property to give a conditional probability density function of "linear" variables conditioning on the state of "nonlinear" variables. Therefore, one doesn't need to sample the stochastic differential equation on the entire state space. In the future, we can incorporate these high-dimensional sampling techniques to the mesh-free version of our hybrid algorithm [26].

The rest of this paper is organized as follows. In Section 2, we review the hybrid method in [15] and rigorously analyze the convergence of the method. We also show that the error will concentrate on the boundary of the domain. A directed block solver is proposed in Section 3. Two possible methods to repair interface error between blocks are studied in Section 4. In Section 5, we use three example systems to test our algorithms and error reduction methods. Section 6 is the conclusion.

2. Analysis of data-driven Fokker-Planck solver

2.1. Algorithm description. Consider a stochastic differential equation

$$\mathrm{d}X_t = f(X_t)\mathrm{d}t + \sigma(X_t)\mathrm{d}W_t, \tag{2.1}$$

where $X_t \in \mathbb{R}^n$, $f:\mathbb{R}^n \to \mathbb{R}^n$ is a continuous vector field, σ is an $n\times n$ matrix-valued function, and $\mathrm{d}W_t$ is the white noise in $\mathbb{R}^n$. We assume that f and σ have enough regularity such that Equation (2.1) admits a unique solution X_t that is a Markov process with a transition kernel $P^t(x,\cdot)$. Similar as in [15], we further assume that X_t admits a unique invariant probability measure π such that

$$\pi P^t(A) = \int_{\mathbb{R}^n} P^t(x,A)\pi(\mathrm{d}x)$$

for any measurable set A. In addition, we assume π is absolutely continuous with respect to the Lebesgue measure, and $P^t(x,\cdot)$ converges to π for any $x\in\mathbb{R}^n$. We refer to [1, 10–13, 19, 20, 25] for the detailed conditions that lead to the existence of solutions of (2.1), the existence of an invariant probability measure, and the convergence to the invariant probability measure.

Let u be the probability density function of π. It is well known that u satisfies the stationary Fokker-Planck equation

$$0 = \mathcal{L}u = -\sum_{i=1}^{n}(f_i u)_{x_i} + \frac{1}{2}\sum_{i,j=1}^{n}(D_{i,j}u)_{x_i x_j}, \tag{2.2}$$

where $D=\sigma^T\sigma$. In addition, because of the convergence, we have

$$u(x) = \lim_{T\to\infty}\frac{1}{T}\int_0^T u(x,t)\mathrm{d}t,$$

where $u(x,t)$ is the probability density function of X_t.

For the sake of simplicity we assume $n=2$ when introducing the algorithm, but our algorithm works for any dimension. Now assume that we would like to solve u numerically on a 2D domain $D=[a_0,b_0]\times[a_1,b_1]$. To do this, an $N\times M$ grid is constructed on D with grid size $h=(b_0-a_0)/N=(b_1-a_1)/M$. Since u is the density function, we approximate it at the center of each of the $N\times M$ boxes $\{O_{i,j}\}_{i=1,j=1}^{i=N,j=M}$ with $O_{i,j}=[a_0+(i-1)h, a_0+ih]\times[a_1+(j-1)h, a_1+jh]$. Let $\mathbf{u}=\{u_{i,j}\}_{i=1,j=1}^{i=N,j=M}$ be this numerical solution on D that we are interested in. $\mathbf{u}$ can be considered as a vector in $\mathbb{R}^{NM}$. Throughout this paper, we still denote this vector by $\mathbf{u}$ when it does not lead to confusion. An entry of $\mathbf{u}$, denoted by $u_{i,j}$, approximates the probability density function u at the center of the (i,j)-box with coordinate $(ih+a_0-h/2, jh+a_1-h/2)$. Now, we consider u as the solution to the boundary-free PDE (2.2) and discretize the operator $\mathcal{L}$ on D with respect to all $(N-2)(M-2)$ interior boxes. The discretization of the Fokker-Planck equation with respect to each center point gives a linear relation among $\{u_{i,j}\}$. This produces a linear constraint for $\mathbf{u}$, denoted as

$$\mathbf{A}\mathbf{u}=\mathbf{0},$$

where $\mathbf{A}$ is an $(N-2)(M-2)\times(NM)$ matrix. $\mathbf{A}$ is said to be the discretized Fokker-Planck operator.

Then we need the Monte Carlo simulation to produce a reference solution. Let $\{\mathbf{X}_n\}_{n=1}^{\mathbf{N}}$ be a long numerical trajectory of the time-δ sample chain of X_t, i.e., $\mathbf{X}_n = X_{n\delta}$, where $\delta>0$ is the time step size of the Monte Carlo simulation. Let $\mathbf{v}=\{v_{i,j}\}_{i=1,j=1}^{i=N,j=M}$ such that

$$v_{i,j}=\frac{1}{\mathbf{N}h^2}\sum_{n=1}^{\mathbf{N}}\mathbf{1}_{O_{i,j}}(X_n).$$

It follows from the ergodicity of (2.1) that $\mathbf{v}$ is an approximate solution of (2.2). Again, we denote the $N\times M$ vector reshaped from $\mathbf{v}$ by $\mathbf{v}$ as well.

As introduced in [15], we look for the solution of the following optimization problem

$$\min\|\mathbf{u}-\mathbf{v}\|_2 \qquad (2.3)$$
$$\text{subject to } \mathbf{A}\mathbf{u}=\mathbf{0}.$$

This is called the least norm problem. Vector

$$\mathbf{u}=\mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}(-\mathbf{A}\mathbf{v})+\mathbf{v} \qquad (2.4)$$

solves the optimization problem (2.3).

2.2. Error analysis through projections. The aim of this section is to show that the solution $\mathbf{u}$ to the optimization problem (2.3) is a good approximation of the global analytical solution u on $\mathbb{R}^2$. Let $\mathbf{u}^{\text{ext}}=\{u^{\text{ext}}_{i,j}\}=u(ih+a_0-h/2, jh+a_1-h/2)$ be the values of the exact solution u at the centres of the boxes. We assume that the Monte Carlo simulation produces an unbiased sample $\mathbf{v}$ that approximates $\mathbf{u}^{\text{ext}}$. We note that this assumption is usually not exactly satisfied because the invariant probability measure of the numerical scheme that produces $\{\mathbf{X}_n\}$ is only an approximation of π. We refer [14] for known results about the difference between the two invariant measures for Langevin dynamics and [18] for that of generic stochastic differential equations. However, when $\mathbf{N}$ is large (greater than 10^7 in our simulations), $\mathbf{v}-\mathbf{u}^{\text{ext}}$ is usually "noisy" enough to be treated as a vector of i.i.d. random numbers. Improving the quality of sampling is extremely important to this algorithm. As seen in the numerical examples, the algorithm can be used with far fewer samples if the error terms at different grid points are independent. We will address sampling methods in our subsequent work.

In order to make the rigorous proof, we need the following assumption.

(H)

(a) For $i=1,\dots,N, j=1,\dots,M$, $v_{i,j}-u^{\text{ext}}_{i,j}$ are i.i.d random variables with expectation 0 and variance ζ^2.

(b) The finite difference scheme for Equation (2.2) is convergent for the boundary value problem on $[a_0,b_0]\times[a_1,b_1]$ with L^∞ error $O(h^p)$.

The performance of the algorithm is measured by $h\mathbb{E}[\|\mathbf{u}-\mathbf{u}^{\text{ext}}\|_2]$, which is the L^2 numerical integration of the error term. Before solving the optimization problem (2.3), we have $h\mathbb{E}[\|\mathbf{v}-\mathbf{u}^{\text{ext}}\|_2]=O(\zeta hN)=O(\zeta)$.

THEOREM 2.1. *Assume* **(H)** *holds. We have the following bound for the* L^2 *error*

$$h\mathbb{E}[\|\mathbf{u}-\mathbf{u}^{\text{ext}}\|_2] \leq O(h^{1/2}\zeta) + O(h^p).$$

Proof. In order to proceed, we need an auxiliary vector that satisfies the linear constraint in Equation (2.3). Consider the Fokker-Planck equation on the extended domain $\tilde{D} = [a_0 - h, b_0 + h] \times [a_1 - h, b_1 + h]$ with boundary condition

$$\begin{cases} \mathcal{L}w = 0 & (x,y) \in \tilde{D} \\ w(x,y) = u(x,y) & (x,y) \in \partial\tilde{D} \end{cases}, \tag{2.5}$$

where h is the mesh size. This problem is well-posed and has a unique solution $u(x,y), (x,y) \in \tilde{D}$.

Consider the discretization of (2.5) by finite difference method. It is of the following form

$$\begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{B} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{u}^{\text{lin}} \\ \mathbf{u_0} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix},$$

where the extended equations

$$\begin{bmatrix} \mathbf{B} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{u}^{\text{lin}} \\ \mathbf{u_0} \end{bmatrix} = \mathbf{0}$$

are the equations for the variables on the boundary ∂D of D, and $\mathbf{u_0}$ are the values of $u(x,y)$ at grid points on the boundary $\partial\tilde{D}$ of $\tilde{D}$. This boundary value problem gives a solution $\mathbf{u}^{\text{lin}}$ that satisfies the linear constraint. By assumption **(H)**, we have

$$\|\mathbf{u}^{\text{lin}} - \mathbf{u}^{\text{ext}}\|_\infty = O(h^p).$$

By the triangle inequality, it is sufficient to estimate

$$\|\mathbf{u} - \mathbf{u}^{\text{lin}}\|_2.$$

Let P be the projection matrix to $\text{Ker}(\mathbf{A})$. Then equation (2.3) implies $\mathbf{u} = P\mathbf{v}$. Since $\mathbf{u}^{\text{lin}} \in \text{Ker}(A)$, we have

$$\mathbf{u} - \mathbf{u}^{\text{lin}} = P\mathbf{v} - \mathbf{u}^{\text{lin}} = P(\mathbf{v} - \mathbf{u}^{\text{lin}}) = P(\mathbf{v} - \mathbf{u}^{\text{ext}}) + P(\mathbf{u}^{\text{ext}} - \mathbf{u}^{\text{lin}}).$$

Take the L^2 norm on both sides and apply the triangle inequality to get

$$h\|\mathbf{u} - \mathbf{u}^{\text{lin}}\|_2 \leq h\|P(\mathbf{v} - \mathbf{u}^{\text{ext}})\|_2 + h\|P(\mathbf{u}^{\text{ext}} - \mathbf{u}^{\text{lin}})\|_2. \tag{2.6}$$

The second term is easy to bound because

$$\begin{aligned} h\|P(\mathbf{u}^{\text{ext}} - \mathbf{u}^{\text{lin}})\|_2 &\leq h\|\mathbf{u}^{\text{ext}} - \mathbf{u}^{\text{lin}}\|_2 \\ &\leq hN\|\mathbf{u}^{\text{ext}} - \mathbf{u}^{\text{lin}}\|_\infty = O(h^p). \end{aligned} \tag{2.7}$$

By assumption **(H)**, $\mathbf{w} = \mathbf{v} - \mathbf{u}^{\text{ext}}$ is a random vector with i.i.d. entries, and P projects $\mathbf{w}$ from $\mathbb{R}^{N\times N}$ to $\text{Ker}(\mathbf{A})$. Note that the dimension of $\text{Ker}(\mathbf{A})$ is $4N-4$. Let $S \in SO(N^2)$ be an orthogonal matrix such that the first $4N-4$ columns of S^T form an orthonormal basis of $\text{Ker}(\mathbf{A})$. Let $\mathbf{s}_1, \cdots, \mathbf{s}_{N^2}$ be column vectors of S^T. Then S is a change-of-coordinate matrix such that $\text{Ker}(\mathbf{A})$ is spanned by $\mathbf{s}_1, \cdots, \mathbf{s}_{4N-4}$.

Let

$$S\mathbf{w} = [\hat{w}_1, \cdots, \hat{w}_{N^2}]^T.$$

We have

$$P\mathbf{w} = \sum_{i=1}^{4N-4} \hat{w}_i \mathbf{s}_i.$$

This implies

$$\mathbb{E}[\|P\mathbf{w}\|_2] = \mathbb{E}[\left(\sum_{i=1}^{4N-4} \hat{w}_i^2 \right)^{1/2}] \le \left(\sum_{i=1}^{4N-4} \mathbb{E}[\hat{w}_i^2] \right)^{1/2},$$

because $\{\mathbf{s}_{i=1}^{N^2}\}$ are orthonormal vectors.

We have

$$\hat{w}_i = \sum_{j=1}^{N^2} S_{ji} w_i,$$

where w_i is the i-th entry of $\mathbf{w}$. S is orthogonal hence

$$\sum_{j=1}^{N^2} S_{ji}^2 = 1.$$

Recall that entries of $\mathbf{w}$ are i.i.d. random variables with expectation zero and variance ζ^2. This implies

$$\mathbb{E}[\hat{w}_i^2] = \zeta^2 \sum_{j=1}^{N^2} S_{ji}^2 = \zeta^2.$$

Hence

$$\mathbb{E}[\|\mathbf{v} - \mathbf{u}^{\text{ext}}\|_2] = \mathbb{E}[\|P\mathbf{w}\|_2] \le \sqrt{4N-4} \cdot \zeta. \tag{2.8}$$

The proof is completed by combining Equations (2.6), (2.7), and (2.8). $\square$

2.3. Concentration of errors. The empirical performance of our algorithm is actually much better than the theoretical bound given in Theorem 2.1. This is because the error term $\mathbf{u} - \mathbf{u}^{\text{ext}}$ usually concentrates at the boundary of the domain. To see this, we can calculate the basis of Ker($\mathbf{A}$). In 1D, Ker($\mathbf{A}$) only contains two linear functions, which has very little error concentration. In 2D, for the case of the Laplacian, $f = 0$ on the unit square domain with $M = N$, a basis of Ker($\mathbf{A}$) can be explicitly given. The following proposition follows easily from some elementary calculations.

PROPOSITION 2.1. *Let* $\mathbf{A}$ *be the discretized Laplacian on a square domain with* $N \times N$ *grids. Without loss of generality assume* N *is odd. Let* $(x,y) = (\frac{i-1}{N-1}, \frac{j-1}{N-1})$ *and* $h = 1/(N-1)$. *Define vectors* $\mathbf{u}^{k,p} \in \mathbb{R}^{N \times N}, p = 1, \cdots, 8$ *by*

$$\begin{aligned}
u_{i,j}^{k,1} &= b_k^1 \sin(2k\pi x) e^{c_k y}, & u_{i,j}^{k,2} &= b_k^2 \cos(2k\pi x) e^{c_k y} \\
u_{i,j}^{k,3} &= b_k^3 \sin(2k\pi x) e^{c_k (1-y)}, & u_{i,j}^{k,4} &= b_k^4 \cos(2k\pi x) e^{c_k (1-y)}, \\
u_{i,j}^{k,5} &= b_k^5 \sin(2k\pi y) e^{c_k x}, & u_{i,j}^{k,6} &= b_k^6 \cos(2k\pi y) e^{c_k x}, \\
u_{i,j}^{k,7} &= b_k^7 \sin(2k\pi y) e^{c_k (1-x)}, & u_{i,j}^{k,8} &= b_k^8 \cos(2k\pi y) e^{c_k (1-x)},
\end{aligned} \tag{2.9}$$

where $c_k = h^{-1}cosh^{-1}(2-cos(2k\pi h))$, $k=1,\cdots,(N-3)/2$ *for* $u^{k,1},u^{k,3},u^{k,5},u^{k,7}$ *and* $k=1,\cdots,(N-1)/2$ *for* $u^{k,2},u^{k,4},u^{k,6},u^{k,8}$, *and* b_k^p *are normalizers to make* $\|\mathbf{u}^{k,p}\|_2 = 1$. *Further define vectors* $\mathbf{u}^{l,p} \in \mathbb{R}^{N\times N}$, $p=1,2,3,4$ *such that*

$$u^{l,1}_{i,j}=1, \quad u^{l,2}_{i,j}=x, \quad u^{l,3}_{i,j}=y, \quad u^{l,4}_{i,j}=xy. \tag{2.10}$$

Then

$$\mathcal{B}=\{\{\mathbf{u}^{k,p}\}_{p=1}^{8},\{\mathbf{u}^{l,p}\}_{p=1}^{4}\}$$

is a basis of $\mathrm{Ker}(A)$.

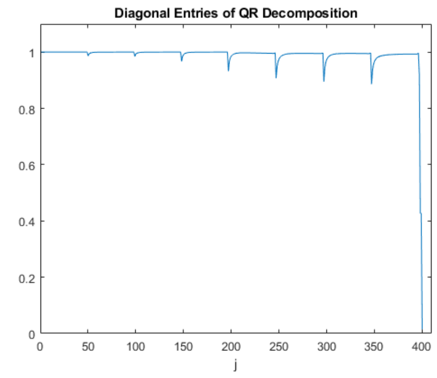

FIG. 2.1. *Diagonal entries from the* R *matrix of the* QR *decomposition of the basis* $\mathcal{B}$ *in* (2.9) *and* (2.10) *respectively arranged in lexicographical order. The basis functions are nearly orthogonal. The mesh size* N *is* 101. *Value of last diagonal entry is* 0.015.

The basis $\mathcal{B}$ in Proposition 2.1 is nearly an orthonormal basis in most directions. For example, a QR decomposition of vectors in $\mathcal{B}$ shows that most diagonal terms are nearly equal to 1 even for large N, as shown in Figure 2.1. The only exception is $\mathbf{u}^{l,4}$, whose corresponding diagonal term is only 0.015. The exponential terms in vectors $\mathbf{u}^{k,p}$ indicate exponential decay of the solution away from the boundary. Since $4N-8$ out of $4N-4$ vectors in $\mathcal{B}$ have significant concentration at the boundary, we expect the concentration of error at the boundary with a high probability.

The basis of $\mathrm{Ker}(\mathbf{A})$ for the general case can not be explicitly given. Instead, we can compute principal angles between $\mathrm{Ker}(\mathbf{A})$ and Θ_D, where Θ_D is the subspace spanned by coordinate vectors corresponding to boundary layer with thickness D. In other words,

$$\Theta_D = \mathrm{span}\{\mathbf{e}_{i,j}\,|\, i\le D \text{ or } j\le D \text{ or } i\ge N-D \text{ or } j\ge N-D\}.$$

If most principal angles are small, $\mathrm{Ker}(\mathbf{A})$ is almost parallel with Θ_D, and the projection of a random vector to Θ_D preserves most of its length. In other words, we see a concentration of error terms at the boundary of the domain.

Principal angles $0\le\theta_1\le\cdots\le\Theta_{4N-4}\le\pi/2$ are a sequence of angles that describe the angle between $\mathrm{Ker}(\mathbf{A})$ and Θ_D. The first one is

$$\theta_1 = \min\{\arccos(\frac{\boldsymbol{\alpha}\cdot\boldsymbol{\beta}}{\|\boldsymbol{\alpha}\|\|\boldsymbol{\beta}\|})\,|\,\boldsymbol{\alpha}\in\mathrm{Ker}(\mathbf{A}),\boldsymbol{\beta}\in\Theta_D\}=\angle(\boldsymbol{\alpha}_1,\boldsymbol{\beta}_1).$$

Other angles are defined recursively with

$$\theta_i = \min\{\arccos(\frac{\boldsymbol{\alpha}\cdot\boldsymbol{\beta}}{\|\boldsymbol{\alpha}\|\|\boldsymbol{\beta}\|}) \,|\, \boldsymbol{\alpha}\in \mathrm{Ker}(\mathbf{A}), \boldsymbol{\beta}\in\Theta_D, \boldsymbol{\alpha}\perp\boldsymbol{\alpha}_j, \boldsymbol{\beta}\perp\boldsymbol{\beta}_j, \forall 1\le j\le i-1\},$$

such that $\angle(\boldsymbol{\alpha}_i,\boldsymbol{\beta}_i)=\theta_i$. Without loss of generality assume $\|\boldsymbol{\alpha}_i\|=1$ for all $1\le i\le 4N-4$. Since $\dim(\Theta_D)\ge\dim(\mathrm{Ker}(\mathbf{A}))$, it is easy to see that $\{\boldsymbol{\alpha}_1,\cdots,\boldsymbol{\alpha}_{4N-4}\}$ forms an orthonormal basis of $\mathrm{Ker}(A)$. Recall that $\mathbf{u}\in\mathrm{Ker}(\mathbf{A})$ and that the error $\mathbf{u}-\mathbf{u}^{\mathrm{ext}}$ is approximated by the projection of a random vector $\mathbf{w}$ with i.i.d. entries to the subspace $\mathrm{Ker}(\mathbf{A})$. Hence we can further assume that $\boldsymbol{\xi}=\mathbf{u}-\mathbf{u}^{\mathrm{ext}}$ is approximated by a random vector

$$\boldsymbol{\xi}=\sum_{i=1}^{4N-4} c_i\boldsymbol{\alpha}_i, \tag{2.11}$$

where c_i are i.i.d. random variables with zero mean and variance ζ^2. Define

$$p_D(\boldsymbol{\xi})=\frac{\mathbb{E}[\|P_{\Theta_D}\boldsymbol{\xi}\|]}{\mathbb{E}[\|\boldsymbol{\xi}\|]}$$

as the mean weight of ξ projected on to the boundary layer, where P_{Θ_D} is the projection matrix to Θ_D. Assume ξ satisfies Equation (2.11). It is easy to see that

$$p_D(\boldsymbol{\xi})=\frac{1}{4N-4}\sum_{i=1}^{4N-4}\cos(\theta_i).$$

Therefore, $p_D(\boldsymbol{\xi})$ measures the degree of concentration of errors on the boundary layer with thickness D.

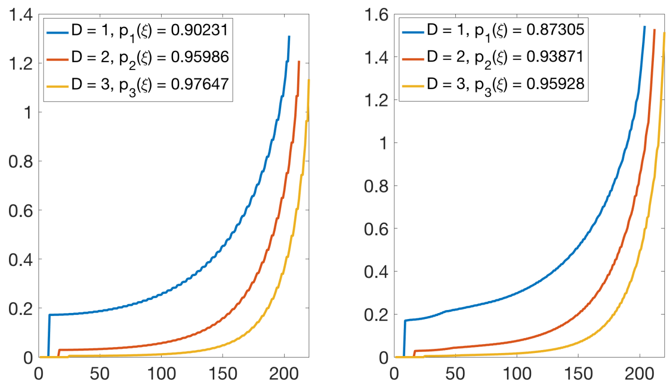

FIG. 2.2. *Principal angles between* $\mathrm{Ker}(\mathbf{A})$ *and* Θ_D *for* $D=1,2,3$. *Left: Diffusion process without drift. Right: Fokker-Planck equation as given in Example 5.1.*

Principal angles can be numerically computed by an SVD decomposition. In Figure 2.2, we list all principal angles for $D=1,2,3$. The matrices in Figure 2.2 are given by discretization of 2D Fokker-Planck Equations (2.2) for $f=0$ (left panel) and f as in Equation 5.1 (right panel). The size of a block is $(50+D)\times(50+D)$. We can see that the mean weight of $\boldsymbol{\xi}$ projected to Θ_D is very large. In other words most of the error term $\mathbf{u}-\mathbf{u}^{\text{ext}}$ concentrates at the boundary layer. We also remark that the degree of concentration of error terms increases with the dimension.

The 2D case is demonstrated in Figure 2.2, and our computation shows that the error concentration is even more significant in 3D.

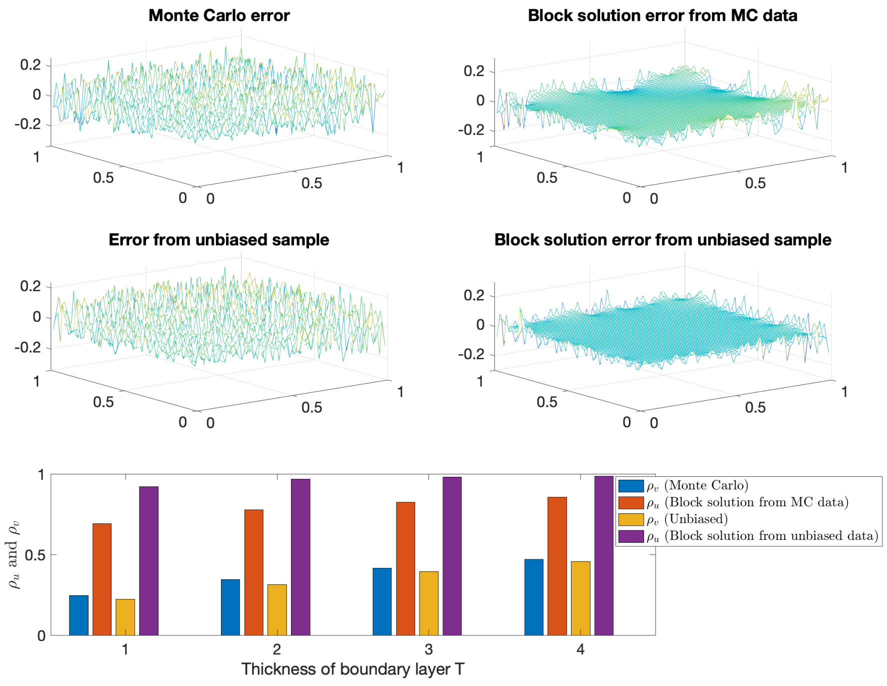

FIG. 2.3. *Empirical spatial distribution of error term for the ring density function as in Section 5.1. Top left:* $\mathbf{v}-\mathbf{u}^{ext}$. *Top right:* $\mathbf{u}-\mathbf{u}^{ext}$. *Middle left:* $\mathbf{v}'-\mathbf{u}^{ext}$. *Middle right:* $\mathbf{u}'-\mathbf{u}^{ext}$. *Bottom: Comparison of* ρ_v, ρ_u, $\rho_{v'}$, $\rho_{u'}$ *for* $D=1,2,3,4$.

Figure 2.3 shows an empirical test of the spatial distribution of error terms. The Fokker-Planck equation is still from the ring density function as in Section 5.1. We choose a 64×64 block on $[0,1]\times[0,1]$ and solve the Fokker-Planck equation with our hybrid solver. The Monte Carlo simulation uses 10^6 sample points. The numerical solutions $\mathbf{v}$ and $\mathbf{u}$ are compared with the exact solution $\mathbf{u}^{\text{ext}}$ in the top left and right panel, respectively. As a comparison, we also produce 10^6 unbiased samples from the invariant density itself, denoted by $\mathbf{v}'$. The solution of the hybrid solver from $\mathbf{v}'$ is denoted by $\mathbf{u}'$. We can clearly see that most of the error of $\mathbf{u}$ and $\mathbf{u}'$ concentrates at the boundary of the domain. The bottom panel compares the relative weight of error concentrating on the boundary layer for $D=1,2,3,4$. The relative weights ρ_u and ρ_v are given by

$$\rho_v=\frac{\|P_{\Theta_D}(\mathbf{v}-\mathbf{u}^{\text{ext}})\|}{\|\mathbf{v}-\mathbf{u}^{\text{ext}}\|} \quad \text{and} \quad \rho_u=\frac{\|P_{\Theta_D}(\mathbf{u}-\mathbf{u}^{\text{ext}})\|}{\|\mathbf{u}-\mathbf{u}^{\text{ext}}\|},$$

respectively. $\rho_{u'}$ and $\rho_{v'}$ are also defined analogously.

From Figure 2.3, the spatial concentration of $\mathbf{u}-\mathbf{u}^{\text{ext}}$ on the boundary layer is less than the theoretical prediction given before, mainly because the sample itself has bias. However, we can still see a significant concentration of error on the boundary layer. The error concentration of $\mathbf{u}'-\mathbf{u}^{\text{ext}}$ is much better. Almost all errors of $\mathbf{u}'$ are concentrated on the two boundary layers. It is worth mentioning that although the unbiased sample $\mathbf{v}'$ has little visual difference from the Monte Carlo data $\mathbf{v}$, the resultant solution $\mathbf{u}'$ has significant better performance in terms of error concentration on the boundary. Hence this example also demonstrates the importance of choosing a good Monte Carlo sampler. We will compare a few different samplers in Section 5.1.

3. Block Fokker-Planck solver

Since we can use the hybrid method to compute the Fokker-Planck equation on any region in the phase space, a straightforward improvement is to apply the divide-and-conquer strategy. We can divide the interested numerical domain into small blocks and then combine the results on these blocks to generate the solution on the original big domain. As discussed in the introduction, assume we divide an N^d mesh into many m^d blocks, where $m \ll N$. If the linear solver to an $n\times n$ matrix has $O(n^p)$ complexity (usually $p>1$), the total computational cost is reduced from N^{pd} to $m^{(p-1)d}N^d$. In addition, this block solver significantly simplifies parallel computing, since all blocks are independent and satisfy the same Fokker-Planck equation. We can also change the grid size for each block based on whether the data is dense or sparse in a subregion to further reduce the computational cost. Moreover, we can apply our method to problems with irregular domains by dividing it into many small rectangular blocks.

For simplicity, we still use a rectangular domain $D=[a,a']\times[b,b']$ to describe our algorithm, and assume that we want to solve u in D. We divide D into $K\times L$ blocks $\{D_{k,l}\}_{k=1,l=1}^{k=K,l=L}$ with $D_{k,l}=[a_{k-1},a_k]\times[b_{l-1},b_l]$, where $a_k=a+k(a'-a)/K$ and $b_l=b+l(b'-b)/L$.

Following the algorithm presented in Section 2.1, we construct an $N\times M$ grid on $D_{k,l}$ and discretize the Fokker-Planck equation. This gives a linear constraint

$$\mathbf{A_{k,l}u^{k,l}}=\mathbf{0}$$

on $D_{k,l}$, where $\mathbf{A_{k,l}}$ is an $(N-2)(M-2)\times(NM)$ matrix. Then we obtain a reference solution $\mathbf{v^{k,l}}$ from the Monte-Carlo simulation by picking up the corresponding values $\{v_{i,j}^{k,l}\}_{i=1,j=1}^{i=N,j=M}$ from the global simulation result $\mathbf{v}$, such that

$$v_{i,j}^{k,l}=v((k-1)N+i,(l-1)M+j)$$

for $k=1,\dots,K, l=1,\dots,L, i=1,\dots,N, j=1,\dots,M$. This gives an optimization problem on $D_{k,l}$

$$\begin{gathered}\min\|\mathbf{u}-\mathbf{v_{k,l}}\|_2 \\ \text{subject to } \mathbf{A_{k,l}u}=\mathbf{0}.\end{gathered} \tag{3.1}$$

We denote the solution to (3.1) by $\mathbf{u_{k,l}}$, which can be obtained by calculating

$$\mathbf{u_{k,l}}=\mathbf{A_{k,l}}^T(\mathbf{A_{k,l}}\mathbf{A_{k,l}}^T)^{-1}(-\mathbf{A_{k,l}}\mathbf{v_{k,l}})+\mathbf{v_{k,l}}.$$

Now, the (i,j) coordinate $u_{i,j}^{k,l}$ of $\mathbf{u_{k,l}}$ is an approximation of u at the point $(ih+a_{k-1}-h/2, jh+b_{l-1}-h/2)$, where $h=(a_k-a_{k-1})/N=(b_l-b_{l-1})/M$ is the grid size when we divide $D_{k,l}$ into $N\times M$ boxes. It remains to combine all local solutions $\{\mathbf{u_{k,l}}\}_{k=1,l=1}^{k=K,l=L}$ on all blocks by collaging them together, i.e.,

$$u(ih+a_{k-1}-h/2,jh+b_{l-1}-h/2)=u_{i,j}^{k,l},$$

$k=1,\dots,K,l=1,\dots,L,i=1,\dots,N,j=1,\dots,M$. The collage numerically solves the Fokker-Planck Equation (2.2) on the whole domain D.

4. Reducing interface error

As discussed in Section 2.3, the optimization problem (2.3) projects most error terms to the boundary of the domain. For the block algorithm, the solution is less accurate near the boundary of each block. The error on the boundary usually looks noisy because it inherits the randomness from Monte Carlo simulations. As a result, there are visible fluctuations on the interface of two adjacent blocks. To make the block solver applicable, modifications to the solution on the interface of blocks are necessary.

In this section, we provide two different methods to reduce the interface error, i.e., the overlapping blocks method and the shifting blocks method. The overlapping blocks method expands each block locally, and keeps only the interior portion which has much lower observed errors. The shifting blocks method makes several smoothing passes, shifting the block boundaries each time so that portions previously on the edges are now in block interiors. Advantages and limitations of these methods will also be discussed.

4.1. Overlapping blocks. Since the numerical solution of the hybrid solver has much higher accuracy at interior points than on the boundary, the most natural approach is to discard the boundary layer. When applying the block solver, we can enlarge the blocks by one or two layers of boxes. Then we apply the algorithm in Section 2.1 on the enlarged block. The interior solution restricted to the original block is the new output of the block solver. This is called the *overlapping blocks* method.

More precisely, recall that we first divide $D=[a,a']\times[b,b']$ into $K\times L$ blocks $\{D_{k,l}\}_{k=1,l=1}^{k=K,l=L}$, then divide each block $D_{k,l}=[a_{k-1},a_k]\times[b_{l-1},b_l]$ into $N\times M$ boxes $O_{i,j}^{k,l}=[a_{k-1}+(i-1)h,a_{k-1}+ih]\times[b_{l-1}+(j-1)h,b_{l-1}+jh]$, where $h=(a_k-a_{k-1})/N=(b_l-a_{l-1})/M$. Instead of D, now we work on the extended domain $\tilde{D}_{k,l}=[a_{k-1}-\iota h,a_k+\iota h]\times[b_{l-1}-\iota h,b_l+\iota h]$, where $\iota=1$ or 2. Then the Monte-Carlo simulation is used to get the reference solution $\tilde{\mathbf{v}}$ on the enlarged domain $\tilde{D}=[a-\iota h,a'+\iota h]\times[b-\iota h,b'+\iota h]$ of D.

Instead of disjoint blocks $D_{k,l}$, we construct an $(N+2\iota)\times(M+2\iota)$ grid on $\tilde{D}_{k,l}$ and generate the discretized Fokker-Planck equation

$$\mathbf{\tilde{A}_{k,l}u=0}$$

on $\tilde{D}_{k,l}$, where $\mathbf{\tilde{A}_{k,l}}$ is a $(N+2\iota-2)(M+2\iota-2)\times((N+2\iota)(M+2\iota))$ matrix. Then a local reference solution $\tilde{\mathbf{v}}^{\mathbf{k,l}}$ is obtained by picking up the corresponding value $\tilde{v}_{i,j}^{k,l}$ from the global simulation vector $\tilde{\mathbf{v}}$. For each block, we solve the local optimization problem (3.1), and keep only the values at the interior points, $D_{k,l}$ to create the global approximation u.

The advantage of this overlapping block method is that it is very easy to implement. No additional treatment is necessary besides discarding one or two boundary layers. But in higher dimension, a significant proportion of grid points will be on the boundary of blocks. For example, if $\iota=2$, $M=N=30$, the percentage of unused grid points is 13%

in 1D, 28% in 2D, 46% in 3D, and 65% in 4D. Also, as seen in Figure 2.3, visible error inherited from the reference solution $\mathbf{v}$ can easily penetrate through $4-5$ boundary layers. Hence the output of solutions from the overlapping block method usually still has some visible residual interface error.

4.2. Shifting blocks. The idea of *shifting block* is also motivated by the concentration of error of the solution of (2.3). To resolve the interface fluctuation between blocks, one can simply move the interface to the interior by shifting all blocks and recalculate the solution. Since the solution has much higher accuracy in the interior of a block, this can easily smooth the interface error. More precisely, after applying the block solver, we make a "half-block" shift of the blocks so that boundaries of the original blocks are now in the interior of new blocks. Then we solve optimization problems (2.3) again on newly shifted blocks. The reference solution fed into the optimization problem (2.3) is the numerical solution from the first round. If necessary, one can carry out this shifting block for several rounds to cover all grid points and to improve the accuracy.

Divide the domain $D=[a,a']\times[b,b']$ into $K\times L$ blocks $\{D_{k,l}\}_{k=1,l=1}^{k=K,l=L}$ with $D_{k,l}=[a_{k-1},a_k]\times[b_{l-1},b_l]$, where $a_k=a+k(a'-a)/K$ and $b_l=b+l(b'-b)/L$. Then we make half-block shifts to get the shifted blocks $D'_{k,l}=[a'_{k-1},a'_k]\times[b'_{l-1},b'_l]$, where $a'_k=a_k+(a'-a)/2K=a+(k+1/2)(a'-a)/K$ and $b'_l=b_l+(b'-b)/2L=b+(l+1/2)(b'-b)/L$. The Monte Carlo data needs to cover all blocks $D_{k,l}$ and $D'_{k,l}$.

Now construct $N\times M$ grids both on $D_{k,l}$ and $D'_{k,l}$, and generate the discretized Fokker-Planck equations

$$\mathbf{A_{k,l}u}=\mathbf{0} \qquad \text{and} \qquad \mathbf{A'_{k,l}u}=\mathbf{0}$$

on $D_{k,l}$ and $D'_{k,l}$ respectively, where $\mathbf{A_{k,l}}$ and $\mathbf{A'_{k,l}}$ are $(N-2)(M-2)\times(NM)$ matrices.

We first use the original block solver to solve the optimization problem on each $D_{k,l}$, as described in Section 3. This gives approximated solutions $\mathbf{u}_{k,l}$ on each block. The first approximation $\mathbf{u}^1\in\mathbb{R}^{MN}$ is obtained by collaging $\mathbf{u}_{k,l}$ from all blocks.

Then we generate the reference solution $\mathbf{v}'^{\mathbf{k,l}}$ on shifted blocks $D'_{k,l}$ by using the corresponding values in $\mathbf{u}^1$ whenever available. More precisely we have

$$\mathbf{v}'^{k,l}_{i,j}=\mathbf{u}^1_{(k-1/2)N+i,(l-1/2)M+j}$$

for $k=1,\dots,K-1, l=1,\dots,L-1, i=1,\dots,N, j=1,\dots,M$. When $k=K$ or $l=L$, we use Monte Carlo data to produce $\mathbf{v}'^{k,l}_{i,j}$ if $\mathbf{u}_1$ data is not available. Then we solve the optimization problem (3.1) on the shifted block $D'_{k,l}$ to get a numerical solution $\mathbf{u'_{k,l}}$ on $D'_{k,l}$.

Now $\mathbf{u}'_{k,l}$ are computed on shifted blocks. We use data from $\mathbf{u}'_{k,l}$ to produce the global solution whenever possible, that is, let

$$u((k-1)N+ih+a_{k-1}-h/2,(l-1)M+jh+b_{l-1}-h/2)=u'^{k,l}_{i,j}$$

for $k=2,\dots,K, l=2,\dots,L, i=1,\dots,N, j=1,\dots,M$. If $k=1$ or $l=1$, we use values from $\mathbf{u}^1$ if the data from $\mathbf{u}'_{k,l}$ is not available.

We remark that in practice one does not have to shift the block by exactly one half. This shifting block method can be implemented repeatedly, such that the solution $\mathbf{u}'$ from last round is used as the reference solution for the next round. We find that one efficient way of implementation is to shift the block by $1/3$ for two times to get two solutions $\mathbf{u}'$ and $\mathbf{u}''$ on shifted blocks. Then we feed $\mathbf{u}''$ back to the original block solver as the reference solution. This implementation covers all grid points by interiors of blocks. Using an iterative linear solver can significantly accelerate the shifting blocks method. Because from the second round, we have $\mathbf{u}_{k,l} \approx \mathbf{v}_{k,l}$ at all interior grid points. Hence $\mathbf{0}$ is a good initial guess when solving $(\mathbf{A}_{k,l}\mathbf{A}_{k,l}^T)^{-1}(-\mathbf{A}_{k,l}\mathbf{v}_{k,l})$ in the optimization problem (3.1). Empirically, the total computation time of three shifts is roughly similar to the time needed for the first round, if the conjugated gradient linear solver is used.

5. Numerical examples

In this section, we consider the following three numerical examples to test the performance of our methods.

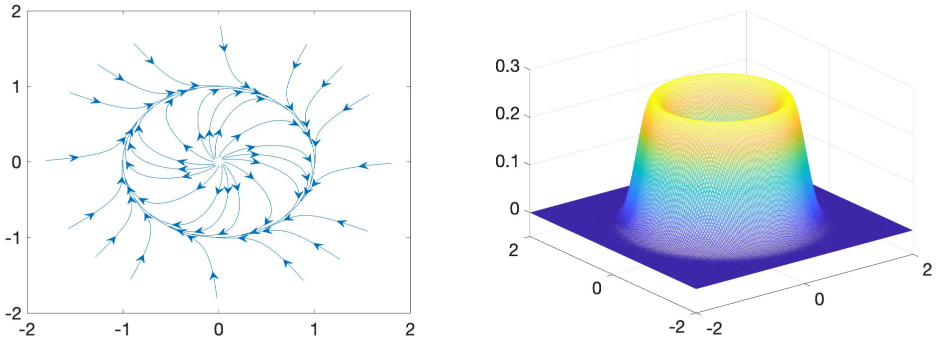

FIG. 5.1. *Left: Some trajectories of the deterministic part of Equation* (5.1). *Right: Exact solution of the Fokker-Planck equation for* (5.1).

5.1. Ring density function.

Consider the following stochastic differential equation:

$$\begin{cases} dx = \big(-4x(x^2+y^2-1)+y\big)\,dt + \varepsilon\, dW_t^x \\ dy = \big(-4y(x^2+y^2-1)-x\big)\,dt + \varepsilon\, dW_t^y \end{cases}, \tag{5.1}$$

where W_t^x and W_t^y are independent Wiener processes. To compare the performance of different solvers in this paper, we fix the strength of white noise to be $\varepsilon = 1$. The deterministic part of Equation (5.1) is a gradient system plus a perpendicular rotation term, where the potential function of the gradient component is

$$V(x,y) = (x^2+y^2-1)^2.$$

See Figure 5.1 Left for selected trajectories of Equation (5.1). The rotation term does not change the invariant probability density function. Therefore, the deterministic part of Equation (5.1) admits a limit circle $x^2+y^2=1$, and the invariant probability measure of (5.1) has density function

$$u(x,y) = \frac{1}{K} e^{-2V/\varepsilon^2},$$

where $K = \pi \int_{-1}^{\infty} e^{-2t^2/\varepsilon^2}\,dt$ is the renormalization constant. Therefore, the stationary Fokker-Planck equation corresponding to (5.1) has an analytic global solution $u(x,y)$ on $\mathbb{R}^2$ (Figure 5.1 Right). We use this example to test a few different aspects of our data-driven Fokker-Planck solver.

A. Comparison of samplers. As stated in Section 2.3, the optimization problem (2.3) can significantly reduce the error in the Monte Carlo sampler if its error terms have spatially uncorrelated entries. We first compare the performance of different samplers without using block solvers. In this example, we look at the approximation obtained from Monte-Carlo simulation with 256×256 mesh points on the domain $D=[-2,2]\times[-2,2]$. Unless otherwise specified, the simulation is done by running the Euler-Maruyama method with a step size $dt=0.002$. We compare the same optimization problem (2.3) using the following six different reference solutions $\mathbf{v}$: (1) Only 10^5 samples are selected from a long trajectory with 10^7 steps. One sample is collected for every 100 steps of the Euler-Maruyama scheme. Then we run a kernel density estimator (KDE) to get the probability density $\mathbf{v}$ at each grid point. (2) All points of the long trajectory with 10^7 samples are used to approximate $\mathbf{v}$. (3) Only 10^5 samples are selected from a long trajectory with 10^7 steps to approximate $\mathbf{v}$. One sample is collected for every 100 steps of the Euler-Maruyama scheme. (4) Eight parallel long trajectories with 1.25×10^6 steps are used to approximate $\mathbf{v}$. (5) Instead of Monte Carlo, 10^6 samples are independently sampled directly from the known invariant probability density function using rejection sampler. (6) Same rejection sampler as (5) with only 10^5 samples. In cases (2)-(6), we use the direct "box counting" method in Section 2.1 to approximate $\mathbf{v}$. In Figure 5.2, we can see a clear advantage of the KDE method, while the quality of the reference solution $\mathbf{v}$ is very low when there are only 10^5 samples.

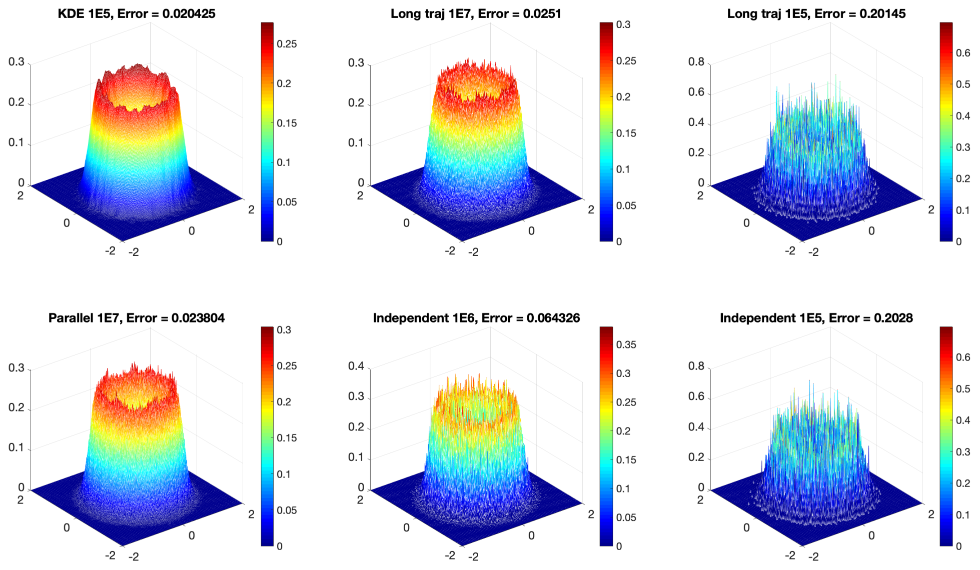

FIG. 5.2. *Top left to bottom right: Reference solution* $\mathbf{v}$ *obtained from (1) KDE approximation based on* 10^5 *samples from one long trajectory of* 10^7 *steps, with long sampling interval of* 100 *steps per sample, (2) Box counting approximation based on* 10^7 *samples from one long trajectory of* 10^7 *steps, with short interval of* 1 *step per sample, (3) Box counting approximation based on* 10^5 *samples from one long trajectory of* 10^7 *samples, with long sampling interval of* 100 *steps per sample, (4) Box counting approximation based on* 10^7 *samples from* 8 *parallel trajectories of* 1.25×10^6 *steps, with short interval of* 1 *step per sample, (5) Box counting approximation based on* 10^6 *samples from the true invariant probability measure, and (6) Box counting approximation based on* 10^5 *samples from the true invariant probability measure.*

However, the advantage of the KDE method does not translate to a more accurate solution to the optimization problem (2.3). We solve the optimization problem (2.3) using the aforementioned six reference solutions $\mathbf{v}$. The difference between the solution $\mathbf{u}$ to the optimization problem (2.3) and the true invariant probability measure is compared in Figure 5.3. We can see that although the KDE method gives a more accurate reference solution $\mathbf{v}$, it does not make the solution $\mathbf{u}$ more accurate. In contrast, the reference solution from independent samples has low accuracy, but gives the most accurate $\mathbf{u}$. Interestingly, fewer samples from a long trajectory with longer intervals in between can slightly improve the accuracy of the solution $\mathbf{u}$ over using all steps of the long trajectory, and a parallel sampler can also improve the accuracy. This result tells us that the optimization problem (2.3) can tolerate a very high level of spatially uncorrelated error in the reference solution $\mathbf{v}$. It is more important to improve the independence of samples between different grid points than reducing the error of the reference solution itself. More independent samples can be achieved by either longer intervals between samples or parallel trajectories. Since longer intervals between samples does not reduce the computational cost (because we still need to simulate the equation for the same number of steps), in practice the parallel sampler is the best choice over other methods. Furthermore, the benefit of using a kernel density estimator does not justify its extra computational cost.

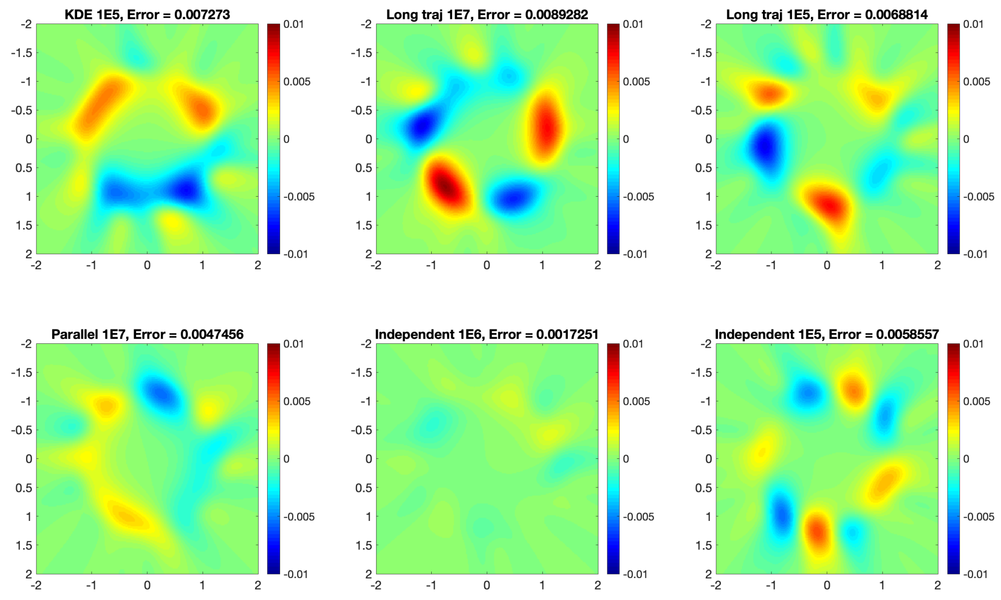

FIG. 5.3. *Top left to bottom right: Error of the solution* $\mathbf{u}$ *to the optimization problem* (2.3) *when the reference solution is obtained from (1) KDE approximation based on* 10^5 *samples from one long trajectory of* 10^7 *steps, with long sampling interval of* 100 *steps per sample, (2) Box counting approximation based on* 10^7 *samples from one long trajectory of* 10^7 *steps, with short interval of* 1 *step per sample, (3) Box counting approximation based on* 10^5 *samples from one long trajectory of* 10^7 *samples, with long sampling interval of* 100 *steps per sample, (4) Box counting approximation based on* 10^7 *samples from* 8 *parallel trajectories of* 1.25×10^6 *steps, with short interval of* 1 *step per sample, (5) Box counting approximation based on* 10^6 *samples from the true invariant probability measure, and (6) Box counting approximation based on* 10^5 *samples from the true invariant probability measure.*

B. Comparison of block solvers. Then we test the performance of block solvers on the same numerical domain $D=[-2,2]\times[-2,2]$ and the same 256×256 grid. The goal is to compare the effect of two error reduction methods proposed in Section 4. We further divide D into 8×8 blocks, each of which thus has 32×32 mesh points. Figure 5.4 Left is the approximation given by the naive block solver described in Section 3. As expected in Section 2.3 and explained at the beginning of Section 4, the error term of the Monte-Carlo simulation data $\mathbf{v}$ (see Figure 5.4 Left) is spread from the interior of each block to its boundary because of the projection, which causes visible interface fluctuation.

The next step is to implement two different error reduction methods introduced in Section 4 and compare their performances. Figure 5.4 Middle shows the solution given by overlapping blocks with 1 layer of box overlap, that is, $\iota=1$ (see Section 4.1). We can see that the interface fluctuation is reduced, especially at the places with high probability density function and high interface fluctuation. The right panel of Figure 5.4 is obtained by iterating the shifting block solver (see Section 4.2) for three repeats. We can see a dramatic reduction of the interface error. The solution looks quite smooth after implementing the shifting block solver.

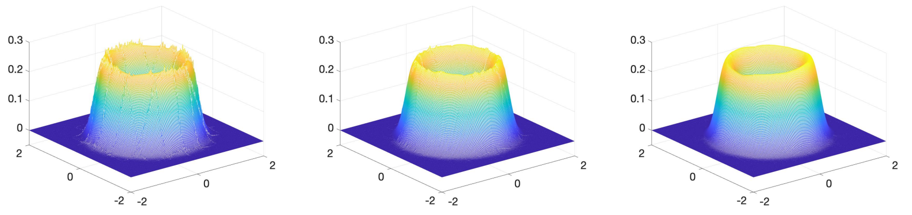

FIG. 5.4. *(**Ring density**) The approximation computed by the basic block solver (**left**),* 1-*overlapping block solver (**middle**), and the shifting block solver (**right**) with* 256×256 *mesh points,* 8×8 *blocks and* 10^7 *samples.*

C. Comparison of performance. Here we compare numerical solutions of the invariant probability measure of Equation (5.1), which is explicitly known. Figure 5.5 shows a comparison of error terms for solutions obtained by different error reduction methods, in both discrete $L^2(D)$ norm and discrete $H^1(D)$ norm. To make a fair comparison, we let the number of samples change with the grid size. Examples with mesh sizes $N=64,128,256,512,1024$, and 2048 are tested and compared. The block size is 32×32 in all tests. The total number of Monte Carlo samples is chosen to be $390.625N^2$. Monte Carlo samples are collected by running 8 parallel long trajectories. Solving local solutions in different blocks is also parallelized. From Figure 5.5, we can see that the L^2 error of the Monte Carlo data is stabilized as expected, because the average sample count per box (and per grid) is constant.

The performance of two error reduction methods are compared in Figure 5.5. We can see that the plain block solver reduces the error significantly compared to the Markov chain data, but the error does not seem to converge to zero. This is not a surprise because all blocks are 32×32, and Theorem 2.1 says that the error should be proportional to $-1/2$ power of the block size. Both error reduction methods reduce the L^2 error from the plain block solver to some degree. The shifting blocks method has better performance, but also a higher computational cost. We can see that the empirical rate of error decay for the shifting block method is roughly $N^{-1/2}$, which is better than the theoretical result in Theorem 2.1.

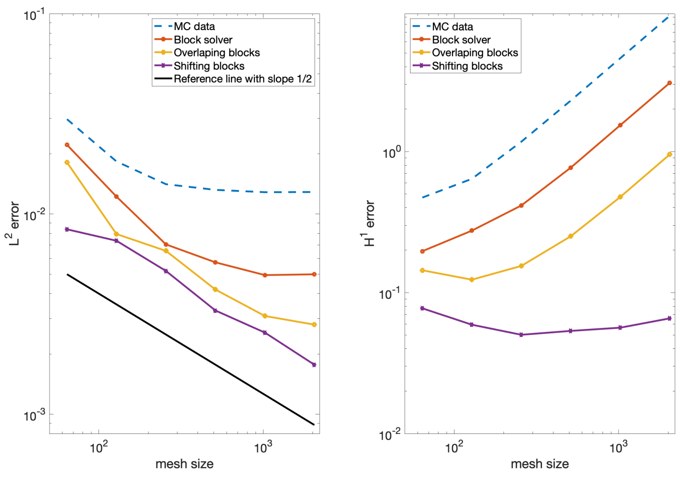

FIG. 5.5. ***(Ring density)*** *Left: Discrete $L^2(D)$ error of solutions produced by different methods. Right: Discrete H^1 error of solutions produced by different methods.*

First order derivatives in the discrete $H^1(D)$ norm are calculated by taking finite differences with respect to nearest grid points. With a constant mean sample size per box, the $H^1(D)$ error of the reference solution from Monte Carlo data diverges when N increases. This is because local fluctuations are roughly unchanged with the mesh size, while the grid size h becomes smaller. Therefore, the derivative of the reference solution is $O(\zeta h^{-1})$, where ζ is the standard deviation of number of samples per box. In other words, all algorithms based on Monte Carlo simulations are expected to have poor performance in $H^1(D)$ error. The divergence of $H^1(D)$ error is alleviated by the overlapping block method, and partially overturned by the shifting blocks method. As we see in Figure 5.5 Right, when $N=2048$, the shifting block method gives a solution whose $H^1(D)$ error is >150 times less than that of the Monte Carlo data.

We can see that due to the lack of interaction between blocks, the information of the reference solution obtained by the Monte-Carlo simulation is not transferred to a neighboring blocks. So if a block is over-sampled, while the others are under-sampled, then after the block solutions are pasted together, the graph is not "flat" at the places where it should be. We can see that the shifting block method has better performance in terms of improving the regularity. This is because it significantly increases interactions between the neighbourhood blocks, and transfers the information between neighborhood blocks. Applying the shifting block method repeatedly can make the result more close to the global solver or the exact solution. But it also incurs some extra computational cost, as seen in Table 5.1.

Finally, we show a comparison of computation time in Table 5.1. In Table 5.1, "Sampling" means the Monte Carlo sampling time (including a burn-in time, which is the waiting time before collecting samples). "Plain" means the plain block solver proposed in Section 3. "Overlapping" means overlapping blocks method with $\iota=1$ in Section 4.1. "Shifting" means the shifting blocks method in Section 4.2. We shift blocks twice by 1/3 and 2/3, and feed the new solution to the original solver as the reference solution. And "Old Version" means the algorithm proposed in [15], where no block is used. We can see that the Monte Carlo sampling actually takes most of the time, and

Mesh	Sampling	Plain	Overlapping	Shifting	Old Version
64	0.02471	0.003822	0.004112	0.01244	0.017317
128	0.09462	0.009741	0.01050	0.04142	0.124361
256	0.4549	0.03165	0.03891	0.1331	0.8035
512	1.8692	0.1172	0.1378	0.5338	10.9225
1024	10.278	0.4626	0.5393	2.0902	61.5952
2048	52.2137	2.0632	2.2946	8.547	781.52

TABLE 5.1. *CPU time (in seconds) for different algorithms and mesh sizes.*

all versions of block-based solvers are very fast. When the mesh size is 2048, the plain block solver is hundreds times faster than solving a large optimization problem (2.3) without dividing the domain.

5.2. Chaotic attractor. In this subsection, we apply our solver to a non-trivial 3D example. Consider the Rossler oscillator with small random perturbations

$$\begin{cases} dx = (-y - z)\,dt + \varepsilon\, dW_t^x \\ dy = (x + ay)\,dt + \varepsilon\, dW_t^y \\ dz = \big(b + z(x - c)\big)\,dt + \varepsilon\, dW_t^z \end{cases}, \tag{5.2}$$

where $a = 0.2$, $b = 0.2$, $c = 5.7$ $\varepsilon = 0.1$, and W_t^x, W_t^y and W_t^z are independent Wiener processes. This system is a representative example of chaotic ODE systems appearing in many applications of physics, biology and engineering. Figure 5.6 shows a trajectory in the corresponding deterministic system and its projection onto the xy-plane.

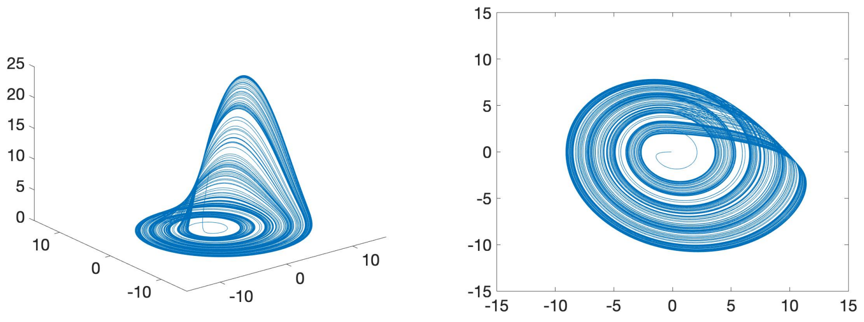

FIG. 5.6. ***(Rossler)*** *A trajectory in the Rossler system* (5.2) ***(left)*** *and its projection on the xy-plane* ***(right)***.

It is natural to imagine that the invariant density of (5.2) has a similar shape to Figure 5.6. We use the block solver together with 3 repetitions of the shifting blocks method on $D = [-15, 15] \times [-15, 15] \times [-1.5, 1.5]$ with $1024 \times 1024 \times 128$ mesh points. The grid is further divided into $32 \times 32 \times 4$ blocks. The reference solution is generated by a Monte Carlo simulation with 3.2×10^{10} samples produced by eight parallel long trajectories. (In practice we let the CPU time of sampling roughly equal the CPU time of linear algebra solver. But the algorithm can use a much smaller sample size.) Four "slices" of the solution, as seen in Figure 5.7, are then projected to the xy-plane for the sake of easier demonstration. Projection of the whole solution to the xy-plane is shown in

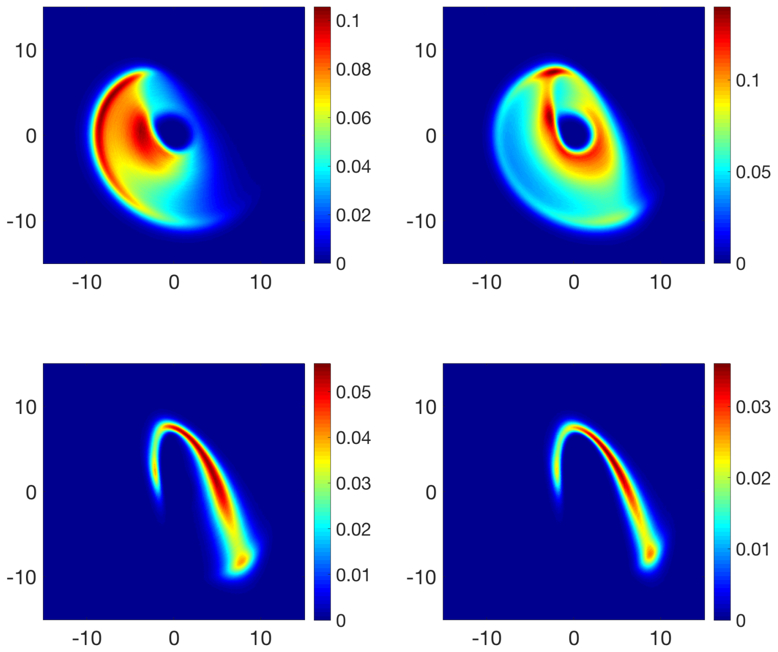

FIG. 5.7. ***(Rossler)*** *Projections of* 4 *"slices" of the invariant density of the Rossler system* (5.2) *to the xy-plane. z-coordinates of* 4 *slices are* $[-0.09375, 0.02344]$, $[0.023440.1406]$, $[0.1406, 0.2578]$, *and* $[0.2578, 0.375]$. *The solution is obtained by a half-block shift solver on* $[-15,15]\times[-15,15]\times[-1.5,2.25]$ *with* $1024\times1024\times128$ *mesh points,* $32\times32\times4$ *blocks, and* 3.2×10^{10} *samples.*

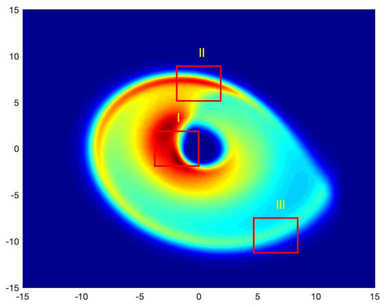

FIG. 5.8. ***(Rossler)*** *The projection of the whole solution of the Rossler system* (5.2) *to the xy-plane. Three different local regions I-III (the red boxes) are used for comparing block solvers with different block sizes. See Figure* 5.9 *also.*

Figure 5.8. In addition to the expected similar shape of the distribution, we can see that many fine local structures of the deterministic system are preserved by the invariant probability measure.

To demonstrate the performance of our algorithm, we apply the data-driven solver without blocks to three local regions with different characteristics (see Figure 5.8). In each region, we use the data-driven solver on $128\times128\times128$ mesh points without dividing the domain into blocks.

In Region I, $[-15/4,0]\times[-15/8,15/8]$, the projection of the solution has both dense and sparse parts that are clearly divided. In the first figure of 5.9, we can see that a similar resolution is preserved when using much smaller block sizes. Both solutions provide high resolution to demonstrate the influence of strong chaos on the invariant distribution. The only difference is the local solver with smaller blocks has higher error on the left and bottom boundary, because the half-shift method does not touch this part. The discrete L^2 norm of the difference between the restriction of global solution on Region I and the local solution is $\varepsilon_{\mathrm{I}}\approx 0.0032$. In Region II, $[-15/8,15/8]\times[165/32,285/32]$, the solution includes an outer "ring" with high density. Outside this "ring", the density function decays quickly. We can see that both solutions show the decay of the density around this ring. The discrete L^2 norm of this difference between the two solutions in Region II is $\varepsilon_{\mathrm{II}}\approx 0.0028$. In Region III, $[75/16,135/16]\times[-45/4,15/2]$, the local solution has much lower density. The local solver is still accurate when the entries of $\mathbf{v}$ are much smaller. The discrete L^2 norm of the difference between the global solution and the local solution in this region is $\varepsilon_{\mathrm{III}}\approx 0.0018$.

Overall, the solution from the block solver has little difference from the one obtained over a large mesh, and the solver can provide desired resolution in both settings. Empirically, we find that a block size of $30-35$ is a good balance of performance and accuracy for most 2D and 3D problems.

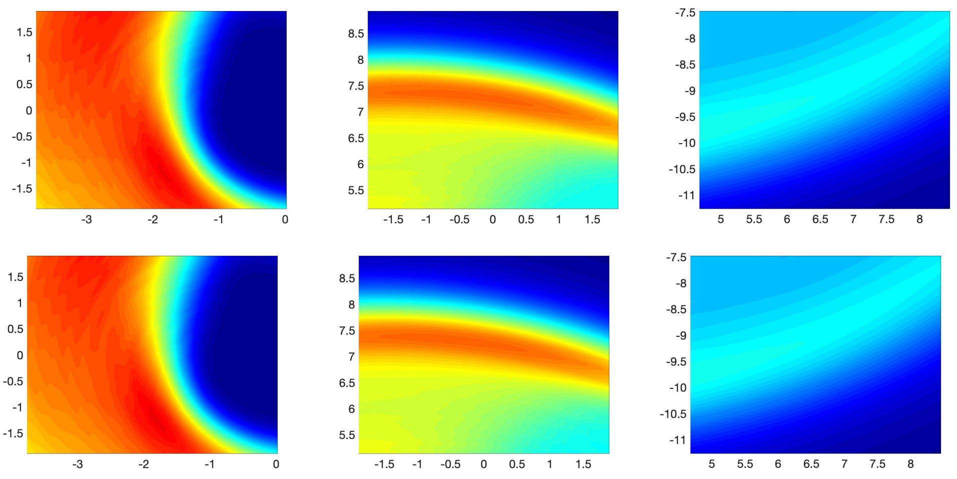

FIG. 5.9. ***(Rossler)*** *The local restrictions of the global solution in Region I–III (the first row), and the projections onto the xy-plane of solutions by the local solver in the three local regions (the second row).*

5.3. Mixed mode oscillation. In this example, we consider another non-trivial 3D system of mixed mode oscillation (MMO) with small random perturbations

$$\begin{cases} dx=\frac{1}{\eta}(y-x^2-x^3)\,dt+\varepsilon\,dW_t^x \\ dy=(z-x)\,dt+\varepsilon\,dW_t^y \\ dz=(-\nu-ax-by-cz)\,dt+\varepsilon\,dW_t^z \end{cases}, \tag{5.3}$$

where $\eta=0.01$, $\nu=0.0072168$, $a=-0.3872$, $b=-0.3251$, $c=1.17$, and W_t^x, W_t^y and W_t^z are independent Wiener processes. The strength of noise is chosen to be $\varepsilon=0.1$.

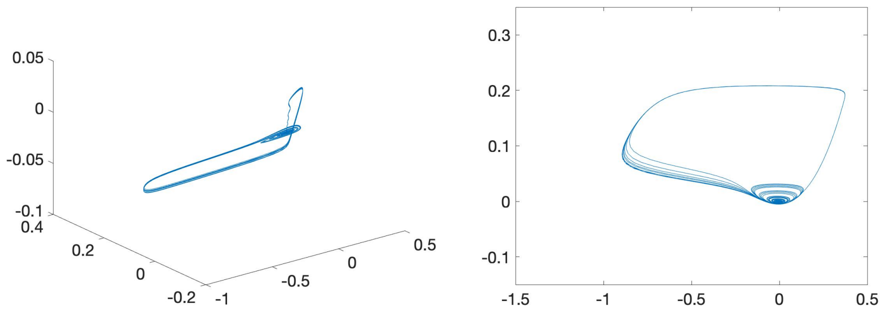

FIG. 5.10. ***(MMO)*** *A trajectory in the system of mixed mode oscillation* (5.3) ***(left)*** *and its projection on the xy-plane* ***(right)***.

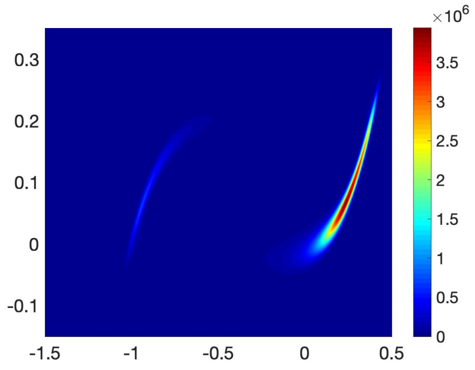

FIG. 5.11. ***(MMO)*** *The projection of the invariant density of the system* (5.3) *of mixed mode oscillation onto the xy-plane by a half-block shift solver on* $[-1.5,0.5]\times[-0.15,0.35]\times[-0.1,0.15]$ *with* $2048\times512\times256$ *mesh points,* $32\times32\times32$ *blocks and* 10^9 *samples.*

Figure 5.10 provides one trajectory of the corresponding deterministic system and its projection on the xy-plane. The deterministic part of Equation (5.3) has a critical manifold $y=x^2+x^3$, at which the derivative of the fast variable vanishes. We can see that oscillations with different amplitudes occur near the fold of the critical manifold, where the attracting and repelling sheet of the critical manifold meet. This is called the mixed mode oscillation (MMO) [5]. The mechanism of mixed mode oscillations is similar as that of the canard explosion, which means the trajectory can follow the unstable sheet of the critical manifold for some time [9]. It was observed in [15] that the canard explosion can be destroyed by a small random perturbation. This motivates us to explore the characteristics of MMO under random perturbations.

We again use the half-block shift solver with $2048\times512\times256$ mesh points, $64\times16\times8$ blocks and 10^9 samples (produced by 8 parallel long trajectories) to get the invariant measure on $D=[-1.5,0.5]\times[-0.15,0.35]\times[-0.1,0.15]$. The entire computation (sampling, applying block solver, and applying three repeats of shifting block solver) takes 3560.52s on the author's laptop. In comparison, the traditional finite difference method on a much smaller grid with $96\times24\times12$ mesh points takes 3978.53s. We did further performance tests on grids with $8m\times2m\times m$ mesh points for $m=8,9,10,11,12$ to get the computational times $T(m)$, which are $113.69, 289.75, 734.53, 1730.85$, and

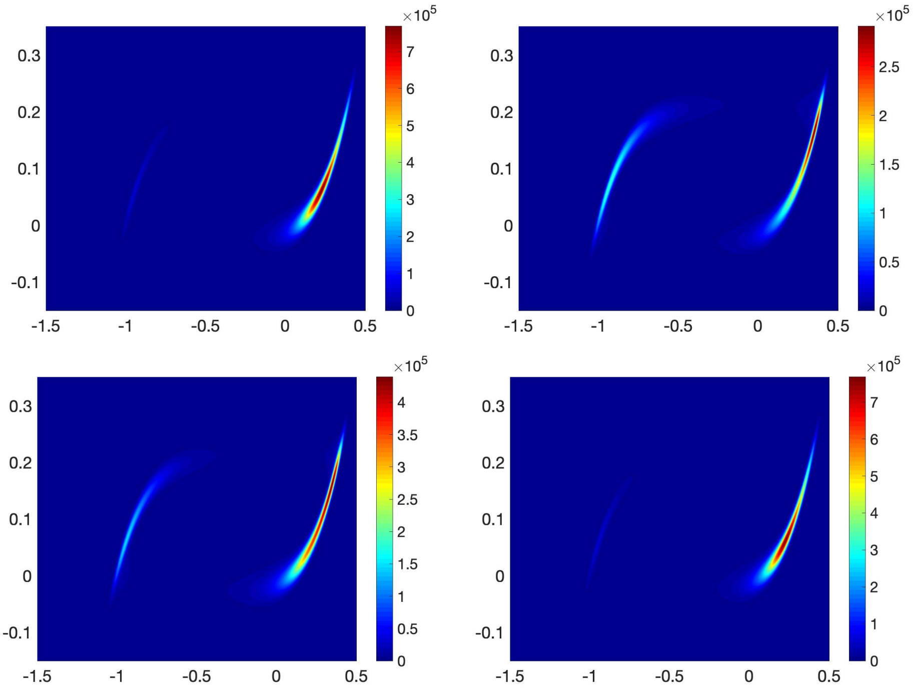

FIG. 5.12. *(MMO) The projections onto the xy-plane of solutions by the local solver in the four z-layers).*

3978.53, respectively. A linear regression on the log-log plot gives an empirical relation $T(m)=e^{-13.57}m^{8.778}$. So, theoretically, when $m=256$ (which is our case), a traditional Fokker-Planck solver would take 1.76×10^{15} seconds on the same computer to solve this problem.

The numerical result is still projected to the xy-plane (see Figure 5.11). We can see that the invariant measure is mainly supported by the neighborhood of the stable sheets of the critical manifold. Deterministic oscillations with small amplitude are eliminated by the random perturbation. In other words, similar to the canard explosion, MMO can not survive a small random perturbation. The mechanism of this phenomenon is still not clear. It is also not known how small the noise should be in order to see MMO in Equation (5.3).

To corroborate the performance of the solver on local regions, in this example, we apply it to four 'z-layers', that is, the region in the phase space of the form $[-1.5,0.5]\times[-0.15,0.35]\times I$, where $I=[-0.1,-0.1+d],[-0.05,-0.05+d],[0.05,0.05+d]$, and $[0.1,0.1+d]$ respectively with $d=1/32$. In each layer, we apply an iterated shifting blocks solver with $2048\times512\times32$ mesh points, $64\times16\times1$ blocks, and 10^9 samples.

Figure 5.12 shows the invariant distribution in these four local layers when projected to the xy-plane. We can see the invariant density function on each z-layer. Similar as in Figure 5.11, most invariant density concentrates at two stable sheets of the invariant manifold, and no small amplitude oscillations can be seen from the invariant probability density function.

6. Conclusion

A data-driven method for computing the invariant probability measure of the Fokker-Planck equation was proposed in [15]. The key idea is to generate a reference solution from Monte Carlo simulation to partially replace the role of boundary conditions. In this paper, we rigorously proved the convergence of this hybrid method. The data-driven method can tolerate very high level of spatially uncorrelated error in the reference solution, which is consistent with the analytical result. The concentration of error is also investigated analytically and numerically. We found that the error tends to concentrate on the boundary of the numerical domain, which makes the empirical performance much better than our theoretical result. Motivated by these results and the divide-and-conquer strategy, we proposed a block version of this hybrid method. It dramatically reduces the computational cost for problems up to dimension 4. This method makes the computation of invariant probability measures possible for many stochastic differential equations arising in different fields, especially for researchers with limited computing resources. Finally, to repair the interface error appearing at the interface between adjacent blocks, two different methods are proposed and tested with several numerical examples.

The block solver studied in this paper can be extended into several directions. A natural extension is the time-dependent Fokker-Planck equations. As discussed in [15], one only needs to slightly modify the optimization problem (2.3) to solve a time-dependent Fokker-Planck equation. This data-driven framework also works for other PDEs with available data from stochastic simulations, such as reaction-diffusion equations. It is well known that a chemical reaction system with diffusions can be computed by both the stochastic simulation algorithm (SSA) and the reaction-diffusion equation. This is similar to the case of the Fokker-Planck equation. In addition, mesh-free version of this block solver can be developed to solve higher dimensional problems. Some high-dimensional sampling methods [2,3] can be adopted to improve the quality of sampling.

REFERENCES

[1] V. Bogachev and M. Röckner, *A generalization of Khasminskii's theorem on the existence of invariant measures for locally integrable drifts*, Theory Probab. Appl., 45:363–378, 2001. 2.1

[2] N. Chen and A.J. Majda, *Beating the curse of dimension with accurate statistics for the Fokker-Planck equation in complex turbulent systems*, Proc. Natl. Acad. Sci., 114(49):12864–12869, 2017. 1, 6

[3] N. Chen and A.J. Majda, *Efficient statistically accurate algorithms for the Fokker-Planck equation in large dimensions*, J. Comput. Phys., 354:242–268, 2018. 1, 6

[4] M.V. Day and T.A. Darden, *Some regularity results on the Ventcel-Freidlin quasi-potential function*, Appl. Math. Optim., 13(1):259–282, 1985. 1

[5] M. Desroches, J. Guckenheimer, B. Krauskopf, C. Kuehn, H.M. Osinga, and M. Wechselberger, *Mixed-mode oscillations with multiple time scales*, SIAM Rev., 54(2):211–288, 2012. 5.3

[6] G.-K. Er, *Methodology for the solutions of some reduced Fokker-Planck equations in high dimensions*, Ann. Phys., 523(3):247–258, 2011. 1

[7] G.-K. Er and V.P. Iu, *State-space-split method for some generalized Fokker-Planck-Kolmogorov equations in high dimensions*, Phys. Rev. E, 85(6):067701, 2012. 1

[8] M.I. Freidlin and A.D. Wentzell, *Random Perturbations of Dynamical Systems*, Springer, 15–43, 1998. 1

[9] J. Guckenheimer and R. Haiduc, *Canards at folded nodes*, Moscow Math. J., 5(1):91–103, 2005. 5.3

[10] M. Hairer and J.C. Mattingly, *Ergodicity of the 2D Navier-Stokes equations with degenerate stochastic forcing*, Ann. Math., 164:993–1032, 2006. 2.1

[11] W. Huang, M. Ji, Z. Liu, and Y. Yi, *Steady states of Fokker-Planck equations: I. Existence*, J. Dyn. Diff. Eqs., 27(3-4):721–742, 2015. 2.1

[12] I. Karatzas and S. Shreve, *Brownian Motion and Stochastic Calculus*, Springer Science & Business

Media, 113, 2012. 2.1

[13] R. Khasminskii, *Stochastic Stability of Differential Equations*, Springer Science & Business Media, 66, 2011. 2.1

[14] T. Lelievre and G. Stoltz, *Partial differential equations and stochastic methods in molecular dynamics*, Acta Numer., 25:681–880, 2016. 2.2

[15] Y. Li, *A data-driven method for the steady state of randomly perturbed dynamics*, Commun. Math. Sci., 17(4):1045–1059, 2019. 1, 2.1, 2.1, 5.1, 5.3, 6

[16] Y. Li and Y. Yi, *Systematic measures of biological networks I: Invariant measures and entropy*, Commun. Pure Appl. Math., 69(9):1777–1811, 2016. 1

[17] A.J. Majda, I. Timofeyev, and E.V. Eijnden, *A mathematical framework for stochastic climate models*, Commun. Pure Appl. Math., 54(8):891–974, 2001. 1

[18] J.C. Mattingly, A.M. Stuart, and M.V. Tretyakov, *Convergence of numerical time-averaging and stationary measures via Poisson equations*, SIAM J. Numer. Anal., 48(2):552–577, 2010. 2.2

[19] S.P. Meyn and R.L. Tweedie, *Markov Chains and Stochastic Stability*, Springer Science & Business Media, 2012. 2.1

[20] B. Øksendal, *Stochastic Differential Equations*, Springer, 65–84, 2003. 2.1

[21] H. Risken, *The Fokker-Planck Equation*, Springer, 63–95, 1996. 1

[22] Y. Sun and M. Kumar, *Numerical solution of high dimensional stationary Fokker-Planck equations via tensor decomposition and Chebyshev spectral differentiation*, Comput. Math. Appl., 67:1960–1977, 2014. 1

[23] Y. Sun and M. Kumar, *A numerical solver for high dimensional transient Fokker-Planck equation in modeling polymeric fluids*, J. Comput. Phys., 289:149–168, 2015. 1

[24] U. von Wagner and W.V. Wedig, *On the calculation of stationary solutions of multi-dimensional Fokker-Planck equations by orthogonal functions*, Nonlinear Dyn., 21(3):289–306, 2000. 1

[25] E.C. Zeeman, *Stability of dynamical systems*, Nonlinearity, 1(1):115–155, 1988. 2.1

[26] J. Zhai, M. Dobson, and Y. Li, *A deep learning method for solving Fokker-Planck equations*, Proc. Mach. Learn. Res., to appear. 1