Computers and Structures 273 (2022) 106896

Contents lists available at ScienceDirect

Computers and Structures

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

Systematic matrix formulation for efficient computational path integration

Henrik T Sykora [b,*], Rachel Kuske [a], Daniil Yurchenko [b]

[a] School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30313, USA
[b] Institute of Sound and Vibration Research, University of Southampton, SO17 1BJ, UK

ARTICLE INFO

Article history:
Received 9 May 2022
Accepted 24 August 2022
Available online 7 September 2022

Keywords:
Path Integration Method
Numerical Method
Chapman Kolmogorov Equation
Probability Density Function
Random Dynamical System
Convergence
Stochastic Differential Equations

ABSTRACT

In this work we introduce a novel methodological treatment of the numerical path integration method, used for computing the response probability density function of stochastic dynamical systems. The method is greatly accelerated by transforming the corresponding Chapman-Kolmogorov equation to a matrix multiplication. With a systematic formulation we split the numerical solution of the Chapman-Kolmogorov equation into three separate parts: we interpolate the probability density function, we approximate the transitional probability density function of the process and evaluate the integral in the Chapman-Kolmogorov equation. We provide a thorough error and efficiency analysis through numerical experiments on a one, two, three and four dimensional problem. By comparing the results obtained through the Path Integration method with analytical solutions and with previous formulations of the path integration method, we demonstrate the superior ability of this formulation to provide accurate results. Potential bottlenecks are identified and a discussion is provided on how to address them.

1. Introduction

The probability density function (PDF) is an important characteristic of the response statistics of dynamical systems to random effects. Through the time evolution or the steady-state of the response PDF of these dynamical systems we can investigate various phenomena in a wide range of application areas, including engineering, finance, physical and life sciences. By obtaining a response PDF, one can predict important future characteristics of the dynamic system, such as n^{th} order moments and reliability, and can improve decision making process, where appropriate. The use of stochastic differential equations (SDEs), where the described phenomenon is governed by established laws, is a concise and efficient tool for the description and the quantitative modelling of such systems.

In the case of utilising SDEs to analyse stochastic dynamics, there are well established methods to obtain the PDF of the state variables, such as using the Monte-Carlo (MC) method, or solving the corresponding Fokker–Planck equation. During the MC method we use path-wise approximations obtained through the numerical integration of the SDE. As it is a stochastic approach, usually we need a large number of approximated realisations to obtain a good approximation for the PDF. The advantage of this method is its relative simplicity and that it can be generalised for a large set of problem classes. However, due to the stochastic nature of the MC method, we might need a large number of sample paths to properly characterise the statistics of the investigated system, and some uncertainty may still remain in the results. Alternatively, in using the Fokker–Planck equation we have to solve a partial differential equation to obtain the response PDF. In general, there is no explicit analytical solution for this equation, except for a small number of special cases, thus requiring a numerical approximation, such as finite element [4,22] or finite difference [27] methods. The main advantage of this approach over the MC method is that the results are deterministic, and as such, it is free from the perturbations that stochastic simulations introduce. However, due to conditions on continuity the Fokker–Planck equation does not generalise to certain non-smooth settings, such as time-varying impacts.

In this paper we develop a novel approach within the path integral (PI) formulation, to track the PDF time evolution of the solution of an SDE. It is based on the law of total probability captured by the Chapman-Kolmogorov (CK) equation. An essential component of the CK equation is the evaluation of an integral involving the transitional PDF (TPDF), that describes the transition from one state to another over a time interval. Since the exact TPDF is not available in most cases, this has to be approximated. One

* Corresponding author.
E-mail addresses: h.t.sykora@soton.ac.uk (H.T Sykora), rachel@math.gatech.edu (R. Kuske), d.iurchenko@soton.ac.uk (D. Yurchenko).

https://doi.org/10.1016/j.compstruc.2022.106896
0045-7949/

possible approach to estimate the TPDF is a generalised cell mapping (GCM), where the region of interest is divided into cells, and the probability of moving from one cell to another is computed through MC simulations [15,30]. This method generalises well for a wide range of systems, but it introduces stochastic perturbations. Another approach, the Wiener Path Integral method, uses a variational formulation and the most probable path to approximate the TPDF [14,13,23,26]. The difficulty of this method is that it requires a Lagrangian functional that must be determined for each individual case, so that it has been applied to only a limited number of specific systems. The difficulty of this method is that it requires the solution of a boundary value problem corresponding to a Lagrangian functional that must be determined for each type of sytem that one wants to analyise. However, when the Lagrangian functional is available, this method is efficient at approximating the TPDF and thus can be used directly to determine the steady state PDF. This has been shown for number of engineering mechanics systems exhibiting diverse nonlinear, hysteretic behaviors [24], even with fractional derivative terms [25]. An approach that balances generalisability and efficiency uses the PDF of an explicit numerical time step to approximate the TPDF, thus removing the need for discrete cells and the negative effects of the stochastic perturbations in the GCM, while utilizing the straightforward determination of the PDF for a large number of system classes, including non-smooth dynamical systems. Hence this method is used in numerous implementations of the PI method, as well as in this work.

Most formulations of the numerical evaluation of PI solutions to the Chapman-Kolmogorov equations lead to a computationally expensive iterative method [2,5,6,19–21], where the integral in the CK is evaluated directly while using an interpolation for the spatial discretisation of the PDF. Recently there were two major efforts in order to reduce the computation time of the iterations within a PI formulation. One approach uses GPUs [2,8] to evaluate an iteration, greatly increasing the performance of the method as it is a highly parallelisable task without restrictions on the problem investigated by this approach. The other approach reduced the computational cost of the calculations by an order of magnitude by reformulating the CK equation as a convolution and using FFT to transform that convolution to a matrix multiplication [9,10,17] in the Fourier space. However, there are a few restrictions with this FFT-based approach: one can apply it to systems subjected to additive noise excitation only, and it also requires that the response PDF is well described in the Fourier space. Furthermore, the previous formulations coupled the discretisation of the PDF and the evaluation of the integral in the CK equation, resulting in inefficient sampling of the TPDF and restrictive dependencies between the spatial and temporal resolutions. Related limitations are observed in the FFT-based method, which is compatible with resolutions that are powers of two, namely $N = 2^m, m \in \mathbb{N}_+$.

In this work we propose a new systematic formulation of the PI approach, the Step Matrix Multiplication PI method (SMM-PI). We transform the CK equation to a matrix multiplication, without the restrictions of the FFT-based approach. This is especially useful in the case of time-invariant systems, since then the matrix corresponding to the CK equation is computed only once and then used iteratively to advance the response PDF in time. An additional advantage is that there are no restrictions on the types of SDE systems that can be investigated within this formulation, since it maintains the general structure of the original PI method. Thus it can be used for a wide range of problem classes, such as systems with parametric noise or non-smooth systems. Several advantages follow from our formulation of SMM-PI, where the numerical solution of the CK equation is separated into three main tasks: interpolation of the PDF, approximation of the TPDF of the process and evaluation of the integral of the CK equation. This structure allows the efficient treatment of each of these computations as well as the flexibility to incorporate different interpolation and approximation methods for improved performance, depending on the problem. The approach also lends itself to straightforward application of further efficiencies, such as parallelisation. We conduct, for the first time, a systematic error and computational cost analysis of the PI method through numerical experiments on one, two, three and a four dimensional problems by comparing the results obtained through the PI method against analytical solutions. This provides a clear understanding of the improvements that can increase the performance and efficiency of the method. As previous studies of the PI-based methods did not include a detailed discussion of the error and performance of the method, providing limited evidence for convergence, here we compare the new approach with the direct and the FFT-based evaluations of the CK equation. This systematic evaluation leads to the clear understanding of the improvements to be made to increase the efficiency of the SMM-PI, and how to achieve them. Finally, we discuss the limitations of this formulation of the PI method and how they can be addressed.

The applications of the proposed SMM-PI include a variaty of stochastic systems described by SDE's. These include nonlinear systems with time-dependent drift and diffusion coefficients, moreover, the SMM-PI can be generalised to non-smooth systems as well. The SMM-PI works well for systems with a wide range of noise sources, including correlated noise. Also, we can have noise directly affecting each coordinate (including correlated noise), or only some, such as in the case of engineering models, where noise tends to appear in the second order equations describing, e.g. a motion. The SMM-PI is especially useful in the analysis of systems that have a steady-state behaviour and have stationary or attracting periodic response PDF's. Based on the general nature of the SMM-PI method, we can use it in biology, physics, engineering, finance or other fields of science.

The paper is organised as follows: in Section 2.1 we introduce and discretise the Chapman-Kolmogorov equation and construct the step matrix. Next, in Section 3 we analyse the error convergence, the performance and the efficiency of the proposed method through numerical examples. Then in Section 4 we provide a discussion on the choice of resolutions for the path integration method, finally, we draw conclusions in Section 5.

2. New Path Integration Formulation

2.1. Chapman-Kolmogorov equation

We consider the following stochastic differential equation

$$\mathrm{d}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t), t)\,\mathrm{d}t + \mathbf{g}(\mathbf{x}(t), t)\,\mathrm{d}\mathbf{W}(t), \tag{1}$$

where $\mathbf{x} = [x_1 \quad \dots \quad x_d]^\top$ is the $\mathbb{R}^d$-valued stochastic state variable, $\mathbf{W}(t)$ is the $\mathbb{R}^m$-valued Wiener process (Brownian motion), $\mathbf{f} : \mathbb{R}^d \times [0,T] \mapsto \mathbb{R}^d, \mathbf{g} : \mathbb{R}^d \times [0,T] \mapsto \mathbb{R}^{d\times m}$ and t is the independent (time) variable.

We arrange the state space $\mathbf{x}$ in such a way that the diffusion term $\mathbf{g}$ has the following structure:

$$g_{ij}(\mathbf{x}(t), t) \equiv 0 \qquad \text{for } i < k \leqslant d \qquad \text{and } j = 1, 2, \dots, m, \tag{2}$$

namely, we formulate the dymanics (1) so that the first $1, \dots, k-1$ states are not subjected to direct noise excitation. Note that it is not uncommon that one or more components of $\mathbf{g}$ vanish, e.g., when (1) gives equations of motions representing a mechanical or electrical system and the components of $\mathbf{x}$ represent acceleration and velocity. We characterise the behaviour of the stochastic dynamical sys-

tem (1) by computing the time dependent PDF $p(\mathbf{x},t)$ using the PI method.

The central part of any formulation of the PI method is the Chapman-Kolmogorov equation

$$p(\mathbf{x},t_{n+1}) = \int_{\mathbb{R}^d} p(\mathbf{x},t_{n+1}|\mathbf{y},t_n)p(\mathbf{y},t_n)\mathrm{d}\mathbf{y}. \tag{3}$$

It describes the time evolution of the probability density function $p(\mathbf{x},t)$ between the discrete times t_n and t_{n+1} using transitional probability density function, conditioned on the initial position at t_n. Since the solution of the Chapman-Kolmogorov equation is generally not available in a closed analytical form, we have to numerically evaluate (3) for each discrete time step t_n. Our systematic treatment of the Chapman-Kolmogorov equation has three key elements:

- Approximation of the transitional probability density function (TPDF) $p(\mathbf{x},t_{n+1}|\mathbf{y},t_n)$,
- Spatial discretization of the PDF $p(\mathbf{x},t_n)$,
- Quadrature computation of the integral.

This approach leads to a number of advantages in computing the time evolution of the PDF $p(\mathbf{x},t)$. The formulation allows us to separately investigate the effect of the discretisation of each component and isolate the potential bottlenecks of the method. A critical feature of our approach is the evaluation of the integral in (3) in terms of multiplication by a step matrix. In the case when the SDE in (1) describes a time-invariant (or time periodic) solution, it is necessary to compute the step matrix (or a finite sequence of step matrices) only once. Furthermore, the systematic treatment of the PI method allows us to analyse each component's influence on the error convergence and computational performance, and opens the door to additional efficient implementations in the future by highlighting the potential bottlenecks of the PI method.

2.2. Approximation of the TPDF

We approximate the transitional probability density function $p(\mathbf{x},t_{n+1}|\mathbf{y},t_n)$ by using the probability density function of the Euler–Maruyama scheme with $\mathbf{x}(t_n)=\mathbf{y}$:

$$\begin{aligned}\mathbf{x}(t_{n+1}) &\approx \mathbf{y} + \mathbf{f}(\mathbf{y},t_n)\Delta t_n + \mathbf{g}(\mathbf{y},t_n)\Delta\mathbf{W}_n :\\ &= \boldsymbol{\eta}(\mathbf{y},t_n,t_{n+1}) + \mathbf{g}(\mathbf{y},t_n)\Delta\mathbf{W}_n,\end{aligned} \tag{4}$$

where $\Delta t_n := t_{n+1}-t_n$ is the length of the time step, and $\Delta\mathbf{W}_n = W(t_{n+1})-W(t_n)$ is the Wiener increment of the process $\mathbf{W}(t)$ on the time interval $[t_n,t_{n+1}]$, with normal distribution $\mathcal{N}(0,\sqrt{\Delta t_n})$.

Since only the states from x_k through x_d are subjected to direct noise excitation, the approximate evolution of the states x_1 through x_{k-1} in (4) is deterministic, while the evolution of the remaining states is stochastic. Thus it is useful to split the state space $\mathbf{x}$ into $\mathbf{x}_I$ and $\mathbf{x}_{II}$:

$$\mathbf{x} = \begin{bmatrix}\mathbf{x}_I\\ \mathbf{x}_{II}\end{bmatrix}, \qquad \text{where } \mathbf{x}_I := \begin{bmatrix}x_1\\ \vdots\\ x_{k-1}\end{bmatrix}, \qquad \mathbf{x}_{II} := \begin{bmatrix}x_k\\ \vdots\\ x_d\end{bmatrix}. \tag{5}$$

The corresponding approximate dynamics is also separated as

$$\boldsymbol{\eta}(\mathbf{x},t_n,t_{n+1}) = \begin{bmatrix}\boldsymbol{\eta}_I(\mathbf{x}_I,\mathbf{x}_{II},t_n,t_{n+1})\\ \boldsymbol{\eta}_{II}(\mathbf{x}_I,\mathbf{x}_{II},t_n,t_{n+1})\end{bmatrix} \quad \text{and } \mathbf{g}(\mathbf{x},t_n) = \begin{bmatrix}\mathbf{0}\\ \mathbf{g}_{II}(\mathbf{x}_I,\mathbf{x}_{II},t_n)\end{bmatrix}. \tag{6}$$

In the following we use the subscripts I and II to denote decompositions based on (5) and (6). The probability density function of the deterministic variables $\mathbf{x}_I$ is given in terms of a Dirac-delta function $\delta(.)$, which is used to write the probability density function of (4) in terms of the decomposition,

$$p(\mathbf{x},t_{n+1}|\mathbf{y},t_n) = \delta(\mathbf{x}_I - \boldsymbol{\eta}_I(\mathbf{y}_I,\mathbf{y}_{II},t_n,t_{n+1}))\cdot p_{II}(\mathbf{x}_{II},\mathbf{y}_I,\mathbf{y}_{II},t_n,t_{n+1}). \tag{7}$$

Since (4) is the Euler–Maruyama step for (1) with increment (4), the function p_{II} is the PDF $p_{\mathcal{N}}$ of a multivariate normal distribution with parameters $\boldsymbol{\mu}=\boldsymbol{\eta}_{II}$ and $\boldsymbol{\sigma}=\mathbf{g}_{II}\sqrt{\Delta t_n}$:

$$p_{II}(\mathbf{x}_{II},\mathbf{y}_I,\mathbf{y}_{II},t_n,t_{n+1}) := p_{\mathcal{N}}\left(\mathbf{x}_{II},\eta_{II}(\mathbf{y}_I,\mathbf{y}_{II},t_n,t_{n+1}),\mathbf{g}_{II}(\mathbf{y}_I,\mathbf{y}_{II},t_n)\sqrt{\Delta t_n}\right), \tag{8}$$

where

$$p_{\mathcal{N}}(\mathbf{x}_{II},\boldsymbol{\mu},\boldsymbol{\mu}) = \frac{e^{-\frac{1}{2}(\mathbf{x}_{II}-\boldsymbol{\mu})^\top(\boldsymbol{\mu}\boldsymbol{\mu}^\top)(\mathbf{x}_{II}-\boldsymbol{\mu})}}{\sqrt{(2\pi)^{d-k+1}\det(\boldsymbol{\mu}\boldsymbol{\mu}^\top)}}. \tag{9}$$

Substituting (7) into (3), we evaluate the integral of the Dirac-delta of a function $\delta(\gamma(\mathbf{x}))$ [11] using

$$\int_{\mathbb{R}^n}\boldsymbol{\beta}(\mathbf{x})\delta(\boldsymbol{\gamma}(\mathbf{x}))\mathrm{d}\mathbf{x} = \frac{\boldsymbol{\beta}(\xi)}{|\det\mathbf{J}_\gamma(\xi)|}, \quad \text{where} \quad \boldsymbol{\gamma}(\xi)=\mathbf{0} \quad \text{and } \mathbf{J}_\gamma := \frac{\partial\boldsymbol{\gamma}}{\partial\mathbf{x}}, \tag{10}$$

which yields

$$p(\mathbf{x},t_{n+1}) = \int_{\mathbb{R}^{d-k+1}} \frac{p_{II}(\mathbf{x}_{II},\boldsymbol{\psi},\mathbf{y}_{II},t_n,t_{n+1})}{\left|\det\mathbf{J}_{\gamma_I}(\boldsymbol{\psi},\mathbf{y}_{II},t_n,t_{n+1})\right|} p\left(\begin{bmatrix}\boldsymbol{\psi}^\top & \mathbf{y}_{II}^\top\end{bmatrix}^\top,t_n\right)\mathrm{d}\mathbf{y}_{II}. \tag{11}$$

Here $\boldsymbol{\psi}$ denotes the solution of

$$\boldsymbol{\gamma}_I(\boldsymbol{\psi},\mathbf{y}_{II},t_n,t_{n+1}) = \mathbf{x}_I - \boldsymbol{\eta}_I(\boldsymbol{\psi},\mathbf{y}_{II},t_n,t_{n+1}) = \mathbf{0}, \tag{12}$$

namely, where the argument of the multivariate Dirac-delta function $\delta(\mathbf{x})$ in (7) is zero. In general, the solution of (12) is not available in a closed analytical form; thus, we obtain it using the Newton–Raphson iteration. The Jacobian $\det\mathbf{J}_{\gamma_I}$ originates from the change of variables in the integration of the Dirac-delta function $\delta(g(\mathbf{x}))$ of a function $g(\mathbf{x})$, detailed in (10). It is defined as

$$\mathbf{J}_{\gamma_I}(\boldsymbol{\psi},\mathbf{y}_{II},t_n,t_{n+1}) := \frac{\partial}{\partial\mathbf{y}_I}\boldsymbol{\gamma}_I(\mathbf{y}_I,\mathbf{y}_{II},t_n,t_{n+1})\Big|_{\mathbf{y}_I=\boldsymbol{\psi}}. \tag{13}$$

We note that instead of (4) we can use other methods to approximate the TPDF of the process (1), such as higher order numerical schemes, e.g. using an explicit 4th order Rungke-Kutta (RK4) step to approximate the drift $\boldsymbol{\eta}$ or using the Milstein method to approximate the diffusion. Further demonstration and discussion is given in Section 3.1.1.

2.3. Spatial discretisation of the PDF

We approximate the PDF $p(\mathbf{x},t_n)$ at discrete times t_n using an interpolation $\bar{p}(\mathbf{x},t_n)$ defined over the finite spatial region $\mathscr{I} := \prod_{j=1}^d \mathscr{I}_j \subset \mathbb{R}^d$, where $\mathscr{I}_j = [a_j,b_j]\subset\mathbb{R}$, given by

$$p(\mathbf{x},t_n) \approx \begin{cases}\bar{p}(\mathbf{x},t_n) & \mathbf{x}\in\mathscr{I},\\ 0 & \text{otherwise.}\end{cases} \tag{14}$$

In (14) we assume that the PDF $p(\mathbf{x},t_n)$ is nonzero on the finite region $\mathscr{I}$, which is reasonable for most stochastic processes for which a steady-state solution $p_{\mathrm{st}}(\mathbf{x})$ exists. Here, $\bar{p}$ denotes an interpolation function defined by

$$\bar{p}(\mathbf{x},t_n) := \sum_{i_1=1}^{N_1}\cdots\sum_{i_d=1}^{N_d}\left(q_{i_1,\ldots,i_d,n}\prod_{j=1}^d\phi_{j,i_j}(x_j)\right) = \langle\boldsymbol{\phi}(\mathbf{x}),\mathbf{q}_n\rangle. \tag{15}$$

The interpolation (15) is constructed by a Cartesian product of interpolations along each dimension labeled by j, where $j = 1, \ldots, d$, where we use N_j for the number of interpolation nodes along the j-th dimension.

For concise notation of the interpolation we use $\langle ., . \rangle$ for a Euclidean inner product [7] of two rank d tensors $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{N_1 \times \ldots \times N_d}$, which is given by

$$\langle \mathbf{A}, \mathbf{B} \rangle := \sum_{i_1=1}^{N_1} \cdots \sum_{i_d=1}^{N_d} A_{i_1,\ldots,i_d} B_{i_1,\ldots,i_d}. \tag{16}$$

The elements of the rank d tensors $\boldsymbol{\phi}(\mathbf{x})$ and $\mathbf{q}_n$ in the product form of the interpolation are given by

$$[\boldsymbol{\phi}(\mathbf{x})]_{i_1,\ldots,i_d} = \prod_{j=1}^{d} \phi_{j,i_j}(x_j) \quad \text{and} \quad [\mathbf{q}_n]_{i_1,\ldots,i_d} = q_{i_1,\ldots,i_d,n}. \tag{17}$$

This formulation allows us to decouple the interpolation functions ϕ_{j,i_j} from the values $q_{i_1,\ldots,i_d}$, that are recorded at the nodes of the interpolation grid. The actual form of the base functions for ϕ_{j,i_j}, depends on the type of the interpolation and the number of nodes N_j that we use along the j-th direction. In Appendix B.3 we provide a simple demonstrative example on how the interpolation (15) is constructed for $d = 2$.

In this work we compare the linear, cubic, quintic, barycentric (Lagrangian), and trigonometric interpolations. The base functions ϕ_{j,i_j} of these interpolations are listed in Appendix B. We refer to the linear, cubic, and quintic interpolations as sparse interpolations, as their base functions $\phi_{j,i_j}(x_j)$ are all zero, except for two, four or six of them, respectively. Similarly, we will refer to the barycentric and trigonometric interpolations as dense, as all their base functions $\phi_{j,i_j}(x_j)$ have nonzero values. Also we implement the barycentric interpolation as Chebyshev interpolation, emphasizing that we record the data $q_{i_1,\ldots,i_d}$ at the Chebyshev points. This is to avoid the Runge phenomenon, an oscillation typically appearing at the edges of the interpolation interval for higher order interpolation approximations based on equally spaced nodes.

A common property of each interpolation approach is that at the nodes $\mathbf{X}_{(i_1,\ldots,i_d)} := \begin{bmatrix} x_{i_1} & \ldots & x_{i_d} \end{bmatrix}^\top$ we have

$$p(\mathbf{X}_{(i_1,\ldots,i_d)}, t_n) = \bar{p}(\mathbf{X}_{(i_1,\ldots,i_d)}, t_n) = q_{i_1,\ldots,i_d,n}. \tag{18}$$

In order to compute the interpolated PDF $p(\mathbf{x}, t_{n+1})$ at the next discrete time t_{n+1} we utilise property (18) to compute the data values $q_{i_1,\ldots,i_d,n+1}$ as

$$q_{i_1,\ldots,i_d,n+1} = \int_{\mathscr{I}} p(\mathbf{X}_{(i_1,\ldots,i_d)}, t|\mathbf{y}, t_n) \bar{p}(\mathbf{y}, t_n) \mathrm{d}\mathbf{y}. \tag{19}$$

For the remainder of this paper, as an abuse of notation, we use a single (i) to index quantities that correspond to quantities previously indexed with $i_1, \ldots, i_d$, e.g.: $q_{(i),n} := q_{i_1,\ldots,i_d,n+1}$ and $\mathbf{X}_{(i)} := \mathbf{X}_{(i_1,\ldots,i_d)}$. Substituting the TPDF (8), (11) and the definition (15) of $\bar{p}$ into (19) yields

$$q_{(i),n+1} = \int_{\mathscr{I}_{(k,d)}} \frac{p_{II}(\mathbf{X}_{(i)}, \boldsymbol{\Psi}_{(i)}, \mathbf{y}_{II}, t_n, t_{n+1})}{\left|\det \mathbf{J}_{\boldsymbol{\gamma}_I}(\boldsymbol{\Psi}_{(i)}, \mathbf{y}_{II}, t_n, t_{n+1})\right|} \left\langle \boldsymbol{\phi}\left(\begin{bmatrix} \boldsymbol{\Psi}_{(i)}^\top & \mathbf{y}_{II}^\top \end{bmatrix}^\top \right), \mathbf{q}_n \right\rangle \mathrm{d}\mathbf{y}_{II}. \tag{20}$$

Here $\mathbf{q}_n$ behaves as a constant with respect to $\mathbf{y}_{II}$, thus we can interchange the integral and the Euclidean inner product leading to

$$q_{(i),n+1} = \langle \boldsymbol{\Phi}_{(i),n}, \mathbf{q}_n \rangle, \tag{21}$$

with

$$\boldsymbol{\Phi}_{(i),n} = \int_{\mathscr{I}_{(k,d)}} \frac{p_{II}(\mathbf{X}_{(i)}, \boldsymbol{\Psi}_{(i)}, \mathbf{y}_{II}, t_n, t_{n+1})}{\left|\det \mathbf{J}_{\boldsymbol{\gamma}_I}(\boldsymbol{\Psi}_{(i)}, \mathbf{y}_{II}, t_n, t_{n+1})\right|} \boldsymbol{\phi}\left(\begin{bmatrix} \boldsymbol{\Psi}_{(i)}^\top & \mathbf{y}_{II}^\top \end{bmatrix}^\top \right) \mathrm{d}\mathbf{y}_{II} \tag{22}$$

where $\boldsymbol{\Psi}_{(i)}$ is defined as the solution of $\mathbf{X}_{(i),I} - \boldsymbol{\eta}_I(\boldsymbol{\Psi}_{(i)}, \mathbf{y}_{II}, t_n, t) = \mathbf{0}$ (similarly to (12)) and $\mathscr{I}_{(k,d)} := \prod_{j=k}^{d} \mathscr{I}_j$.

As (21) gives a single $q_{(i),n+1}$, we repeat (19) for each node value $q_{(i),n+1}$ and organise each resulting $\boldsymbol{\Phi}_{(i),n}$ into a step matrix $\mathbf{S}_n$ to obtain the matrix multiplication

$$\mathrm{vec}(\mathbf{q}_{n+1}) = \mathbf{S}_n \mathrm{vec}(\mathbf{q}_n) \quad \text{where} \quad \mathbf{S}_n = \begin{bmatrix} \vdots \\ \mathrm{vec}(\boldsymbol{\Phi}_{(i),n})^\top \\ \vdots \end{bmatrix}. \tag{23}$$

This form is equivalent to the numerical evaluation of (3) and represents a time step in the approximation of the PDF $p(\mathbf{x}, t)$. The operator $\mathrm{vec}(\mathbf{A})$ reshapes the d-dimensional tensor $\mathbf{A} \in \mathbb{R}^{N_1 \times \ldots \times N_d}$ as a one-dimensional column vector $\mathrm{vec}(\mathbf{A}) \in \mathbb{R}^{\bar{d}}$, where $\bar{d} = \prod_{j=1}^{d} N_j$.

The decoupling of the interpolation function $\boldsymbol{\phi}(\mathbf{x})$ from the recorded node values $\mathbf{q}_n$ in (15) allows us to transform the evaluation of the Chapman-Kolmogorov Eq. (3) to the matrix multiplication in (23). In the case that (1) is time-invariant ($\mathbf{f}(\mathbf{x}, t) \equiv \mathbf{f}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x}, t) \equiv \mathbf{g}(\mathbf{x})$) and we want to investigate the evolution or the steady-state solution of the PDF $p(\mathbf{x}, t)$, we have to compute the step matrix $\mathbf{S}_n \equiv \mathbf{S}$ once, given that we use a constant time step Δt for each n. Then advancing the PDF $p(\mathbf{x}, t)$ by a single time-step is a very efficient operation, as it is achieved via a matrix–vector multiplication defined in (23). Similarly, in the case that (1) is a T_p-periodic system ($\mathbf{f}(\mathbf{x}, t) = \mathbf{f}(\mathbf{x}, t + T_p)$ and $\mathbf{g}(\mathbf{x}, t) = \mathbf{g}(\mathbf{x}, t + T_p)$), then we have to compute the set of $\mathbf{S}_n$, $n = 1, \ldots, p$ (where $p = T_p/\Delta t$ is the discrete period), and then just successively evaluate the matrix multiplication (23) to advance the PDF $p(\mathbf{x}, t)$ in time.

The main computational cost for the SMM-PI is in finding $\mathbf{S}_n$, since the matrix multiplication is one of the best optimised numerical task. Given that the step matrix $\mathbf{S}_n$ is already computed, the matrix multiplication (23) is one of the fastest methods to approximate (3). This is especially useful when investigating time invariant or periodic systems, that require a large number of time steps to reach their steady-state.

Note that we have to choose the limits of the interpolated region $\mathscr{I}$ in such a way that it covers the entire region where the PDF $p(\mathbf{x}, t)$ takes nonzero values during the investigated time frame $t \in [0, T]$.

2.4. Approximation of the integral

The final component of the solution of the Chapman-Kolmogorov equaiton (3) is to evaluate the integral (22). Depending on the time step Δt_n and the state $\mathbf{X}_{(i)}$, there is only a small $\bar{\mathscr{I}}_{(k,d,i)} \subset \mathscr{I}_{(k,d)}$ region where the TPDF p_{II}, and thus the kernel of (22), is nonzero. To minimise the number of subnodes required for an accurate numerical approximation of $\boldsymbol{\Phi}_{(i),n}$, we aim to evaluate the kernel (22) where it is relevant, namely, over the region $\bar{\mathscr{I}}_{(k,d,i)}$. In order to determine $\bar{\mathscr{I}}_{(k,d,i)}$, we first identify the $\boldsymbol{\xi}$ initial state of the drift step from where the system evolves to $\mathbf{X}_{(i)}$ by solving the equation

$$\mathbf{X}_{(i)} - \boldsymbol{\eta}(\boldsymbol{\xi}, t_n, t_{n+1}) = \mathbf{0}. \tag{24}$$

Next, we take the covariance matrix $\mathbf{\Sigma}(\xi, t_n, t_{n+1}) = \mathbf{g}(\xi, t_n)\mathbf{g}(\xi, t_n)^\top \Delta t_n$ and use its spectral radius to compute the largest standard deviation σ, i.e.,

$$\sigma^2 = \rho(\mathbf{\Sigma}), \text{ where } \rho(\mathbf{\Sigma}) = \max_z \{\text{abs}(z),\ z \in: \det(\mathbf{\Sigma} - z\mathbf{I}) = 0\}. \tag{25}$$

With the help of σ we determine the new region $\bar{\mathscr{I}}_{(k,d,i)} := \mathscr{I}^{(\sigma)}_{(k,d,i)} \cap \mathscr{I}_{(k,d)}$ over the integral in (22) is evaluated by

$$\mathscr{I}^{(\sigma)}_{(k,d,i)} = \prod_{j=k}^{d} [\xi_j - s\cdot\sigma, \xi_j + s\cdot\sigma]. \tag{26}$$

Since we use a small time step Δt_n, we assume that the distribution in $\mathbf{y}$ is approximately Gaussian, thus we choose $s = 6$ to ensure that the whole region is covered where the TPDF p_{II} is nonzero. Using the region $\bar{\mathscr{I}}_{(k,d,i)}$ instead of $\mathscr{I}_{(k,d)}$ we restrict the integration to where the TPDF is non-negligible.

In order to evaluate the integral (22) over $\bar{\mathscr{I}}_{(k,d,i)}$ with high accuracy and performance, we use the Gauss–Legendre (GL) quadrature [1]. For integrating a function $f : \mathbb{R} \mapsto \mathbb{R}^{N_1\times\ldots\times N_d}$ (e.g., in case $k = d$) over $[-1, 1]$ the GL quadrature is defined as

$$\int_{-1}^{1} f(z)\mathrm{d}z \approx \sum_{i=1}^{m} w_i f(z_i). \tag{27}$$

Here, the nodes $z_i, i = 1, \ldots, m$, are defined by the roots of the m-th Legendre polynomial, where m is the number of sample points used, and the weights w_i are given by the formula

$$w_i = \frac{2}{(1 - z_i^2)\left(Q_m^2(z_i)\right)}, \quad \text{where} \quad Q_m(z_i) = \frac{\partial}{\partial z} P_m(z)\bigg|_{z=z_i}, \tag{28}$$

and $P_m(z)$ is the m-th Lagrange polynomial. In the case of $k = d$ we can apply (27) directly by rescaling the integration lattice values z_i and weights w_i appropriately for the chosen interval. To apply the GL quadrature to integrate over the interval $\bar{\mathscr{I}}_{(k,d,i)}, k \neq d$, we need to rescale the integration lattice values z_i and weights w_i in each dimension $j = k, \ldots, d$ constructing $\bar{\mathscr{I}}_{(k,d,i)}$ and take their Cartesian product in a manner similar to (15).

The formulations of the PI method in previous works [17,19,20] relied on using uniformly distributed node values in a cubic interpolation to directly evaluate the integral. Utilising the GL quadrature approximation combined with the integral over the region $\bar{\mathscr{I}}_{(k,d,i)}$ greatly increases the accuracy and performance of the (approximate) evaluation of the integral (22) [29]. We discuss this further in Section 4.

3. Numerical experiments

In this section we test the error convergence and the computational efficiency of the PI method through numerical experiments. The first example we investigate is a simple nonlinear scalar system with additive noise:

$$\mathrm{d}x(t) = \left(x(t) - x^3(t)\right)\mathrm{d}t + \sqrt{2}\mathrm{d}W(t) \qquad x(0) \sim \mathcal{N}(0, 1), \tag{29}$$

where $\mathcal{N}(0, 1)$ refers to the standard normal distribution. The steady-state PDF of (29) shown in Fig. 1a is given by

$$p_{\text{st}}^{\text{A},1}(x) = C^{-1} e^{x^2 - x^4}, \qquad C = \frac{e^{1/8}\pi}{2}\left(I_{-\frac{1}{4}}\left(\frac{1}{8}\right) + I_{\frac{1}{4}}\left(\frac{1}{8}\right)\right), \tag{30}$$

where $I_\alpha(z)$ is the modified Bessel function of the first kind.

The second system we discuss is a second order system ($d = 2$), namely an oscillator with cubic nonlinearity:

$$\begin{aligned} \mathrm{d}x(t) &= v(t)\,\mathrm{d}t, \\ \mathrm{d}v(t) &= \left(-2\zeta v(t) + x(t) - \lambda x^3(t)\right)\mathrm{d}t + \sigma\,\mathrm{d}W(t). \end{aligned} \tag{31}$$

Here we use the states x and v, that represent the displacement and velocity of the oscillator, respectively. Here the state vector is defined as $\mathbf{x} = [x\ v]^\top$ and the initial distribution is a two-dimensional standard normal distribution, i.e., $\mathbf{x}(0) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The stationary PDF $p_{\text{st}}(\mathbf{x})$ corresponding to (31) is

$$p_{\text{st}}^{\text{A},2}(\mathbf{x}) := p_{\text{st}}^{\text{A},2}(x, v) = C_1 e^{C_0\left(\frac{1}{2}\left(x^2 - v^2\right) - \frac{1}{4}\lambda x^4\right)}, \quad \text{where} \quad C_0 = \frac{4\zeta}{\sigma^2}, \tag{32}$$

and C_1 is the normalisation constant. For the tests we choose $\zeta = 0.15$, $\lambda = 0.25$ and $\sigma^2 = 0.075$, which results in a symmetric steady-state PDF $p_{\text{st}}(\mathbf{x})$ with two well separated peaks as Fig. 1b shows. Then the approximation of $p_{\text{st}}^{\text{PI}}(\mathbf{x})$ with dense interpolations is challenging, as the contrast of large flat regions with the rapid increase of the peaks make the interpolations prone to high frequency components.

To benchmark the performance of the PI method for $d = 3$ and $d = 4$, we apply the method to a linear oscillator subjected to first order filtered noise and to a coupled oscillator. The governing equation of the linear oscillator for $d = 3$ is

$$\begin{aligned} \mathrm{d}x(t) &= v(t)\,\mathrm{d}t, \\ \mathrm{d}v(t) &= \left(-2\zeta v(t) - x(t) + \sigma Z(t)\right)\mathrm{d}t, \\ \mathrm{d}Z(t) &= -\mu_1 Z(t)\mathrm{d}t + \sqrt{2\mu_1}\,\mathrm{d}W(t). \end{aligned} \tag{33}$$

Here the states x and v represent the displacement and velocity, respectively, while the state Z is the first order filtered noise. For the numerical tests use $\zeta = 0.05, \sigma = 0.5$, and $Z(t)$ as the standard Gaussian noise excitation ($Z(t) \sim \mathcal{N}(0, 1)$) with $\mu_1 = 2$. The governing equations of the coupled oscillator for $d = 4$ are

$$\begin{aligned} \mathrm{d}x(t) &= v_x(t)\mathrm{d}t, \\ \mathrm{d}y(t) &= v_y(t)\mathrm{d}t, \\ \mathrm{d}v_x(t) &= \left(-0.4\,v_x(t) - 0.2\,v_y(t) - 2x(t) + y(t)\right)\mathrm{d}t, \\ \mathrm{d}v_y(t) &= \left(0.2\left(v_x(t) - v_y(t)\right) - \left(x(t) - y(t)\right)\right)\mathrm{d}t + \mathrm{d}W(t). \end{aligned} \tag{34}$$

Here the states x and y represent the displacements, and the v_x and v_y the corresponding velocities.

In the following sections we test the accuracy and performance in the case of the scalar system ($d = 1$) and the nonlinear oscillator ($d = 2$), while for the $d = 3$ and $d = 4$ dimensional systems we approximate the CPU time required to compute the step matrix $\mathbf{S}_n$ and the matrix multiplication (equivalent to taking a time step). Furthermore, as all the systems described by Eqs. (29)–(34) are time invariant, we omit the notation of the time step n, where it is unnecessary, i.e. $\mathbf{S}_n \equiv \mathbf{S}$ and $\Phi_{(i),n} \equiv \Phi_{(i)}$.

3.1. Error of the path integration method

First, we test the error convergence of the PI method for different temporal and spatial resolutions for the systems described by (29) ($d = 1$) and (31) ($d = 2$). Throughout this section the error ε_1 of the PI solution is defined as

$$\varepsilon_1 := \int_{\mathscr{I}_{(k,d)}} \left|p_{\text{st}}^{\text{A},d}(\mathbf{x}) - p_{\text{st}}^{\text{PI},d}(\mathbf{x})\right|\mathrm{d}\mathbf{x}, \tag{35}$$

where $p_{\text{st}}^{\text{A},d}$ and $p_{\text{st}}^{\text{PI},d}$ denote the analytical and the numerically determined steady-state PDFs ($p_{\text{st}}(\mathbf{x}) := \lim_{t\to\infty} p(\mathbf{x}, t)$), respectively. We ensure, that $p(\mathbf{x}, t_n)$ has converged to the steady-state solution $p_{\text{st}}(\mathbf{x})$ according to the criterion

$$\int_{\mathscr{I}_{(k,d)}} \left|p^{\text{PI},d}(\mathbf{x}, t_n) - p^{\text{PI},d}(\mathbf{x}, t_{n+1})\right|\mathrm{d}\mathbf{x} < \kappa\Delta t, \tag{36}$$

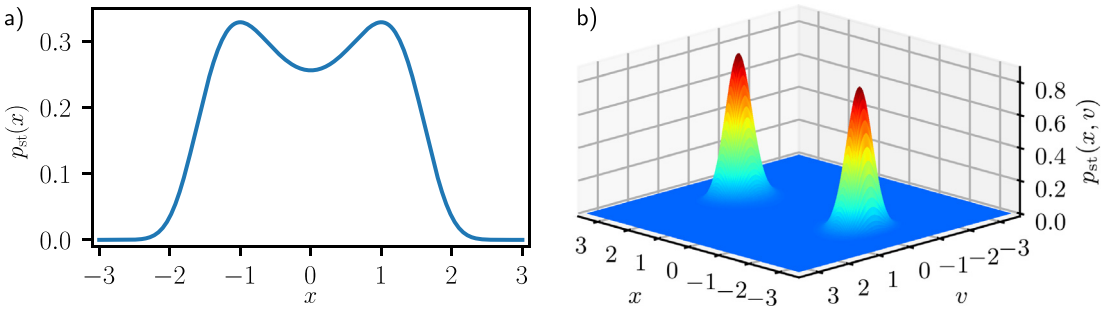

Fig. 1. Steady-state response PDF's of the systems described by (a) Eq. (29) and (b) Eq. (31) with parameters $\zeta = 0.15, \lambda = 0.25$ and $\sigma^2 = 0.075$.

for an appropriate choice of κ, i.e. $\kappa < \varepsilon_1$. Since the error we achieved in our tests cases is $\varepsilon_1 > 10^{-6}$, we choose $\kappa = 10^{-8}$. For the scalar system (29) we satisfy (36) with $\kappa = 10^{-8}$ by computing to a final unitless time $t = T = 6.8$ at which we reach the steady-state PDFs (30), for each investigated spatial and temporal resolution. Likewise at $T = 38.4$ the second order system (31) reaches (32) at each resolution. Thus we conservatively choose $p^{\mathrm{PI}}_{\mathrm{st}}(\mathbf{x}) = p^{\mathrm{PI}}(\mathbf{x}, 10)$ for the benchmarks for $d = 1$ and $p^{\mathrm{PI}}_{\mathrm{st}}(\mathbf{x}) = p^{\mathrm{PI}}(\mathbf{x}, 40)$ for $d = 2$. For the evaluation of the integral in the CK Eq. (3) discussed in Section 2.4, we use 31 Gauss–Legendre quadrature nodes.

3.1.1. Error due to the temporal discretisation

First we consider the effect of the temporal resolution Δt on the error ε_1 of the steady-state PDF approximated with the PI method. Fig. 2 shows the error ε_1 of the PI method as a function of the time step Δt for different time stepping methods: the Euler–Maruyama scheme described in (4) (labeled *Euler* in the figure) and the 4th order Runge–Kutta-Maruyama method (labeled *RK4* in the figure). In the Runge–Kutta-Maruyama method the drift is approximated with the 4th order Runge–Kutta scheme (detailed in Appendix A), while the diffusion is approximated with the Maruyama scheme.

Utilising the RK4 method adds additional complexity to the computation required to solve (12) and determine J_l. For $d > 1$ this might translate to better approximations for the interaction between the state variables, however, it does not increase the error convergence rate as $\Delta t_n \to 0$ as it is limited by the weak order of the diffusion step, which is $O(\Delta t)$ (or $\varepsilon_1 \propto \Delta t$) for the Maruyama approximation of the diffusion. The argument against the use of the Milstein method for the approximation of the diffusion is similar. The computation of the distribution of the Milstein step is very complex in the general case, with $k \neq d$ and non-diagonal diffusion, while not providing better convergence properties: both the Milstein and the Maruyama approaches yield a weak $O(\Delta t)$ approximation for the diffusion [12].

For the scalar system (31) ($d = 1$), Fig. 2a shows that we achieve almost the same accuracy, regardless of the time stepping methods used. In contrast, as Fig. 2b shows, for the second order system (31) ($d = 2$) the use of the *RK4* method provides higher accuracy, because it better approximates the dynamics of the state variables described by the drift. However, the two time stepping methods have the same error convergence rate, which is limited by the approximation of the noise term. This is illustrated by the auxiliary line that corresponds to a first order error convergence ($\varepsilon_1 \propto \Delta t^1$).

3.1.2. Error due to spatial discretisation

Next, we consider the effect of the spatial resolution N and the choice of the interpolation method on the error ε_1 of the steady-state PDF approximated with the PI method. For the scalar system (29) we choose the interpolated region as $x \in [-3, 3]$, while for the second order system (31) we choose $x, v \in [-3.5, 3.5]$ and use the same spatial resolution N for the interpolation along both dimensions x and v. Fig. 3 shows the errors of the different interpolation methods as functions of the number of nodes N used for the interpolation, and compares different time steps Δt.

Based on the results obtained by numerical experiments, we see that the linear interpolation is the least reliable spatial discretisation method. Even though the solution corresponding to the linear interpolation converges with $O(N^{-2})$, it does so after a plateau. For both the scalar $d = 1$ and second order $d = 2$ systems, this plateau persists for larger node numbers N as we increase Δt, even for the entire range of N investigated. The cubic and quintic interpolations display a faster convergence rate of $O(N^{-3})$ and $O(N^{-5})$, respectively, and are less susceptible to resolution errors generated by the decrease of the time step Δt. The Chebyshev and trigonometric interpolations show an exponential convergence for all time resolutions Δt, so that for smaller values of N, the maximum achievable accuracy is attained, as dictated by the time stepping method, This is particularly clear for the scalar system ($d = 1$): in Fig. 3a the Chebyshev interpolation achieves this minimum error ε_1 at $N = 25$ for $\Delta t = 10^{-3}$ and $N = 37$ for $\Delta t = 10^{-5}$, while for the cubic interpolations it is achieved for $N > 100$ ($\Delta t = 10^{-3}$), or not at all in the investigated region ($\Delta t = 10^{-5}$). The accuracy obtained with the

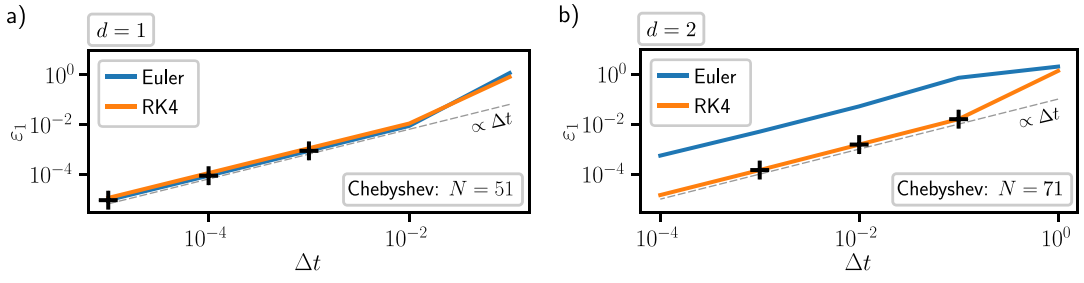

Fig. 2. Error ε_1 of the PI approximation of the PDFs (a) $p^{\mathrm{A},1}_{\mathrm{st}}(x)$ in (30) and (b) $p^{\mathrm{A},2}_{\mathrm{st}}(x, v)$ in (32) as a function of the time step Δt. The black crosses denote the minimum achievable errors ε_1 corresponding to the temporal step size Δt in Fig. 3.

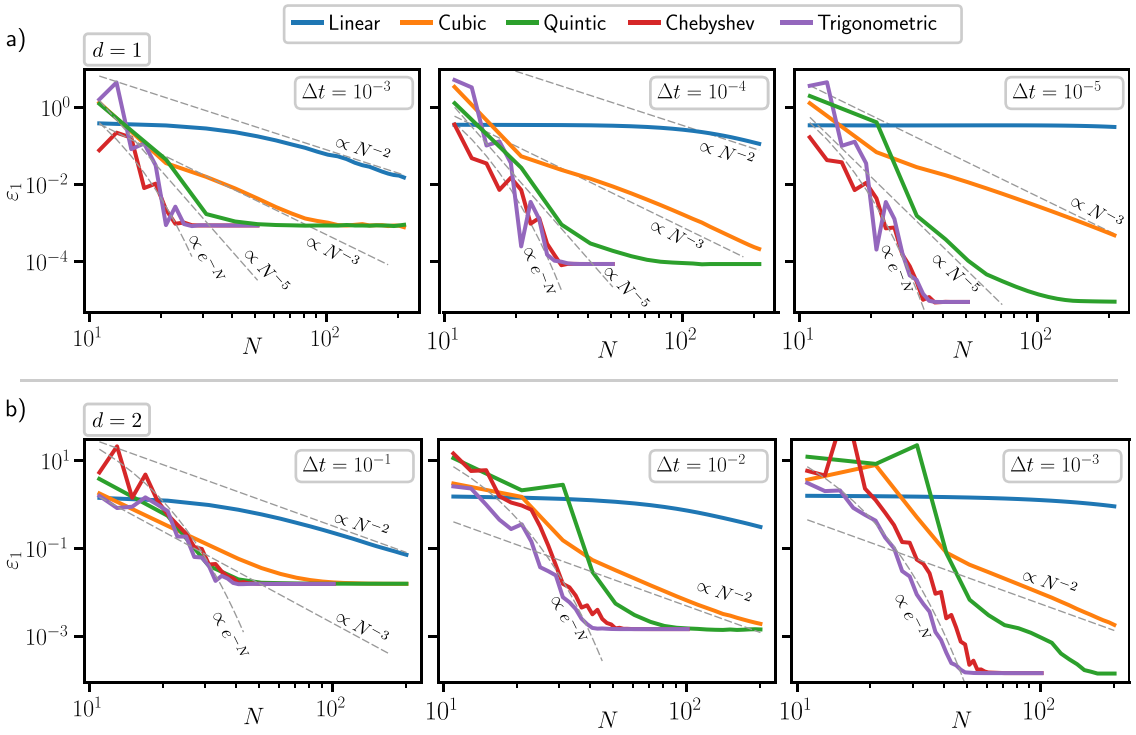

Fig. 3. Comparison of the error ε_1 as a function of the number N of the interpolation nodes, for the different interpolation methods used in the PI approximation of the PDFs (a) $p_{\mathrm{st}}^{\mathrm{A},1}(x)$ in (30) and (b) $p_{\mathrm{st}}^{\mathrm{A},2}(x,v)$ in (32).

quintic interpolation lies between the cubic and the dense interpolations.

In contrast, for the second order system $d = 2$ (Fig. 3b) the use of sparse interpolations can produce results with ε_1 error similar to that obtained by the dense interpolations for larger time steps ($\Delta t = 10^{-1}$). The quintic interpolation performs especially well, with its convergence rate appearing exponential for the largest time step $\Delta t = 10^{-1}$, similar to the dense interpolations. However, as we decrease the time step Δt, the dense interpolations unequivocally produce more accurate results for larger spatial resolutions N, until the accuracy is limited by the time discretisation.

In Fig. 3 we observe another phenomenon, i.e., as we decrease the time step Δt, the error can increase if we do not increase the number of nodes N for the interpolations. Fig. 4 illustrates how the interactions of the resolutions for $d = 1$ can influence the solutions obtained through the SMM-PI method for $\Delta t = 10^{-2}, 10^{-3}, 10^{-5}$ and $N = 15, 25, 51$ compared to the analytical solution $p_{\mathrm{st}}^{\mathrm{A}}(x)$ in (30). We see, that increasing the number N of the interpolation nodes indeed increases the approximation accuracy, up to a point where the approximated solutions are almost perfectly aligned with the analytical solution. However, decreasing the time step Δt can lead to less accurate results or to false solutions, if the number N is not adjusted accordingly. For the scalar system (29) the sparse interpolations (linear, cubic or quintic) are more susceptible to this phenomenon than the dense interpolations (Chebyshev and trigonometric). The figure shows both the failure of the linear interpolation to capture small variations in the PDF for smaller Δt, and the instability for higher order interpolations with limited N.

3.2. Performance benchmarks

In this section we benchmark the performance of the PI method in terms of CPU time and memory requirements. These results are complementary to those from Section 3.1, which characterise the numerical performance of the interpolation and thus the mathematical algorithm but may not translate directly to the time needed to compute accurate results. The measures we provide in this section reflect that the different interpolation methods result in different densities of the step matrices $\mathbf{S}$, which in turn can influence the time it takes to compute a single time step. Our approach of the PI method is implemented in Julia (version 1.7 with libLLVM-12.0.1) and the tests were completed on a thin and light laptop with an Intel® Core™ i7 8565U CPU, with 24 GB of RAM and a Linux kernel 5.14. During the performance tests the step matrix $\mathbf{S}$ is represented as a dense matrix for the dense interpolations for $d = 1$ and $d = 2$, and as a compressed sparse row (CSR) sparse matrix in case of both the sparse and dense interpolations for $d = 3$ and $d = 4$. For comparison we included the benchmark results of two FORTRAN implementations [16] of the PI method: the direct integration [18] of the CK Eq. (3) and the approach where the CK equation is evaluated using the FFT [17]. We refer to these methods as the direct and FFT-based methods, respectively. Both utilise cubic B-spline interpolations for the spatial discretisation.

3.2.1. Computational time and spatial discretisation

In Fig. 5 we summarise the CPU time requirements to compute the matrix $\mathbf{S}$ and to take a single time step as a function of the spatial resolution N. During the tests we used $\Delta t = 10^{-5}$ for $d = 1$, $\Delta t = 10^{-3}$ for $d = 2$, $\Delta t = 10^{-2}$ for $d = 3$ and $\Delta t = 10^{-1}$ for $d = 4$. The dashed lines represent approximated computation times. The computation time of $\mathbf{S}$ was approximated based on the average time needed to compute a single row $\mathrm{vec}(\mathbf{\Phi}_{(i),n})$ of the step matrix $\mathbf{S}$: specifically, we individually measured and averaged the computation time of 1000 randomly selected rows and then multiplied this average by the number of rows N^d. For the approximation of the time required to evaluate a time step (a matrix multiplication) we use the previously computed rows and individually measured the computation time of multiplying each row with a dense column vector representing the node value vec-

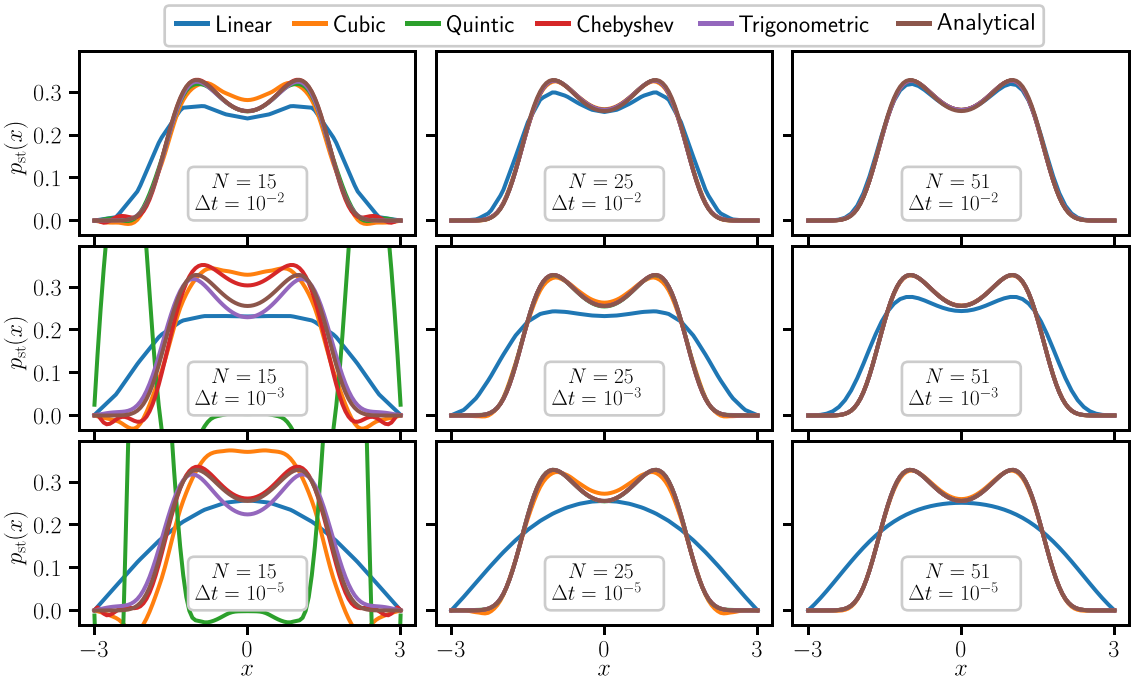

Fig. 4. The PDF $p_{st}(\mathbf{x})$ of (29) approximated by the PI method, compared to the reference solution defined by (30). The curves not visible are in agreement with and covered by the analytical solution.

tor $\mathbf{q}_n$. We averaged these measurements over the rows, and multiplied this average by the number of rows N^d to obtain the estimate of the CPU time of a single time step.

In Fig. 5 the left column presents the CPU time necessary to compute the step matrix $\mathbf{S}$, the right column presents the CPU time necessary to compute a single time step, while the rows correspond to the different d-dimensions of the systems used for the examples.

For the scalar problem ($d = 1$) defined by (29), we see that the use of a sparse matrix for representing the step matrix $\mathbf{S}$ is beneficial only for larger spatial discretisations N. When we consider the exact computation time needed (continuous lines), we see that for small spatial resolutions N the computation of the step matrix is almost constant and the computation time gradually increases as we increase N. This is due to the significant overhead time to prepare the computations of the step matrix $\mathbf{S}$, i.e., we need to preallocate the memory space for $\mathbf{S}$ and prepare the Newton–Raphson iteration to get $\mathbf{\Psi}_{(i)}$, all of which has comparable computational cost to the computations of the rows $\Phi_{(i)}$. As we increase the spatial resolution N, the effect of this constant overhead becomes negligible compared to the time required to compute $\mathbf{S}$, as the approximated time converges to the exact time requirement. Recall that for the approximation we consider only the computations of the rows $\Phi_{(i)}$.

We also see a significant difference in the time it takes to compute the matrix $\mathbf{S}$ for the sparse interpolations (almost perfectly overlapping each other), the Chebyshev interpolation and for the trigonometric interpolation. This is due to the difference of the complexity of the two interpolation classes: during the sparse interpolation we need to compute two (linear interpolation), four (cubic interpolation) or six (quintic interpolation) weights, while for the Chebyshev and trigonometric interpolations require weights at all N points. Moreover, the computation of the weight matrix $\boldsymbol{\phi}$ for the trigonometric interpolation takes longer than for that of the Chebyshev interpolation, given that the trigonometric functions are more computationally expensive than the polynomials used in the Chebyshev case.

For $d = 2$ there is again a small overhead computation time in the computation of the step matrix $\mathbf{S}$, related to preallocations for small spatial resolutions N. When comparing the exact time measurements (continuous lines) with the approximation (dashed lines), we see that this overhead becomes negligible compared to the computation of the rows, as the two sets of lines converge to each other. For the dense interpolations, the computation of $\mathbf{S}$ is more costly (approximately by a factor of 2), but the time complexity of the computation of $\mathbf{S}$ is the same $O(N^4)$ for both sets of interpolation methods. Again, the computations times are grouped together for the sparse and dense interpolations.

For $d = 3$ and $d = 4$ we give an approximation of the CPU time for computing $\mathbf{S}$. We do not include the curve corresponding to exact measurements given the length of time to compute them for larger N, e.g. approximately 277 h for $d = 3$ and $N = 201$, or more than 28000 h (or 3.2 years) for $d = 4$ and $N = 81$. The approximations give a time complexity of $O(N^6)$ and $O(N^8)$ for $d = 3$ and $d = 4$, respectively. The black pluses ("+") indicate exactly measured CPU times for $d = 3$ ($N = 24$) and $d = 4$ ($N = 32$) with cubic interpolation.

Considering the CPU time cost of a single time step for $d = 1$, Fig. 5 shows a performance gain from a sparse representation only for matrix $\mathbf{S}$ sizes larger than approximately 101×101. As in Fig. 3, for $d = 1$ the dense interpolations reach the maximum accuracy at $N = 37$ even for $\Delta t = 10^{-5}$, thus larger matrix sizes are not necessary to accurately compute the PDF using dense interpolations. The figure confirms, that the computation of an accurate steady-state PDF p_{st} with dense interpolations is orders of magnitudes faster than the computation with sparse interpolations for $d = 1$.

For $d = 2$ representing $\mathbf{S}$ as a sparse matrix is beneficial for the CPU cost of a time step, particularly for larger spatial discretisation values of N. There the time step computation is 3 orders of magnitude faster for the sparse representation than for the dense methods. For $d = 3$ and $d = 4$ the approximated CPU time cost of a time step is again orders of magnitude faster for the sparse interpolations than it is for the dense interpolations. Here we see again that

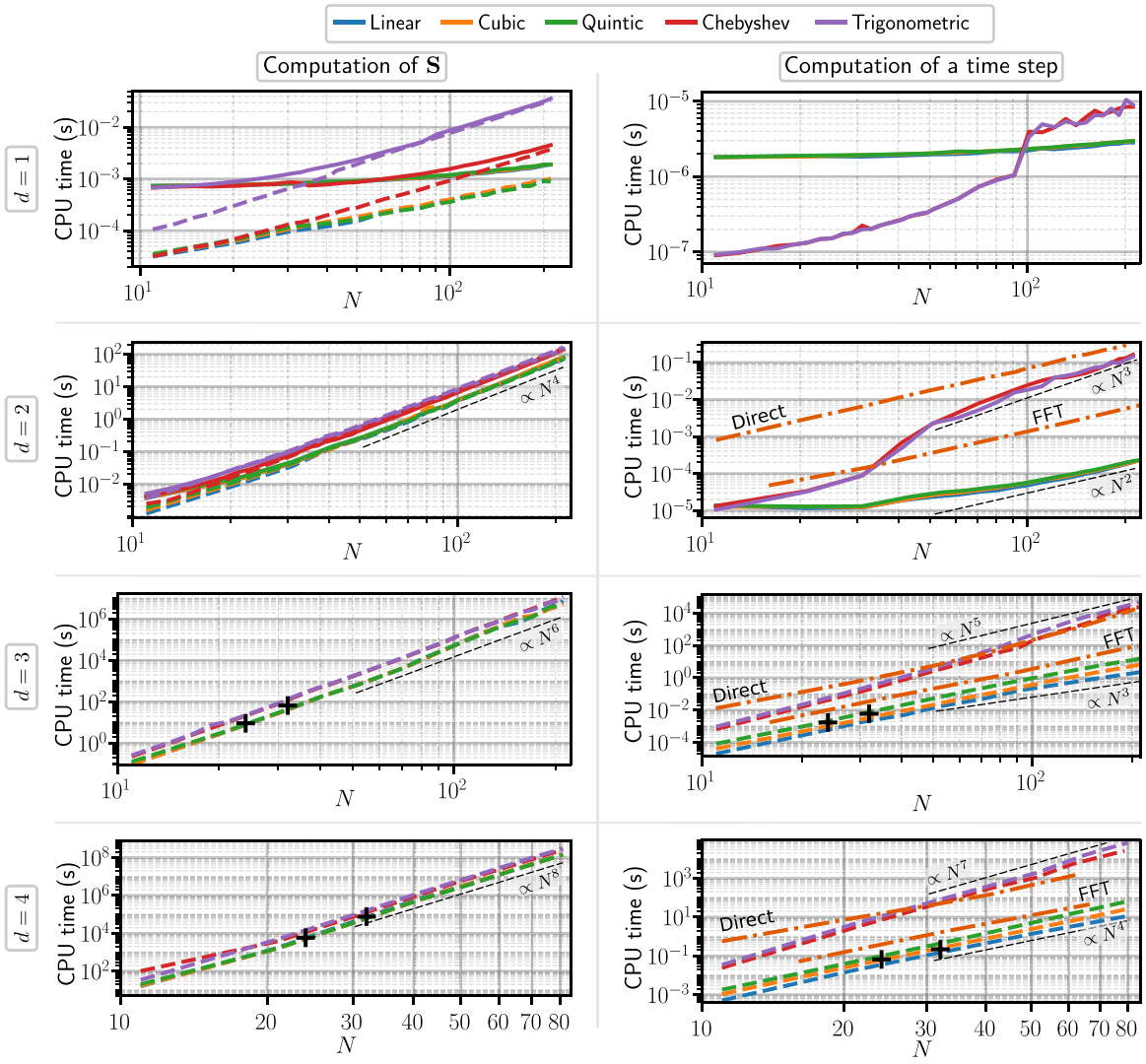

Fig. 5. The CPU time required to compute the step matrix **S** and a single time step for $d = 1, 2, 3$ and 4. The dashed lines represent approximated computational times, while the black crosses ("+") for $d = 3$ and 4 are validating measurements at $N = 24$ and 32 with cubic interpolation. Note that some lines are overlapping due to nearly identical results.

both the sparse and dense interpolations are separated into two groups in terms of the computation time of **S**.

Next, in Fig. 5 we compare the requirements of computing a time step with the SMM-PI with the time required to compute a time step with the direct and the FFT-based approaches for $d > 1$. In the case where the step matrix **S** has already been computed based on sparse interpolations, computing a single time step by multiplying the node value vector $\mathrm{vec}(\mathbf{q}_n)$ with the step matrix **S** is order of magnitudes faster than the direct approach and significantly faster than the FFT based approach. For the dense interpolation, the matrix multiplication can be slower even than the direct approach, since the number of operations in matrix multiplication (even for a sparsely represented step matrix obtained with dense interpolation) negatively impacts the performance of a time step. Also, analogous to the computation time of the step matrix **S**, the computation results are split into two groups based on the interpolation class (sparse or dense).

To summarise, we observe in Fig. 5 that for $d = 2, 3$ and 4 the time complexity of computing the step matrix is approximately $O(N^{2d})$; while the time complexity of computing a time step with dense interpolations is approximately $O(N^{2d-1})$ and $O(N^d)$ for the dense and sparse interpolations, respectively. Thus we assume, that this holds for higher $d > 4$ dimensions as well. As the computational time necessary to compute **S** scales with d exponentially, the accurate computation of the PDF for larger systems is challenging using the SMM-PI method without using other efficiencies, e.g. parallelisation. We note that the systematic framework we present provides a solid foundation for implementing such efficiencies.

3.2.2. Numerical efficiency

In this section we investigate the efficiency of the PI method, defined as the computational time necessary to obtain a result with error ε_1. Fig. 6 combines the results in Figs. 3 and 5, thus providing the computational cost of achieving a desired error ε_1 utilising different interpolation methods for $d = 1$ and $d = 2$. Similarly to Section 3.1.2 we use $p^{\mathrm{PI}}_{\mathrm{st}}(\mathbf{x}) = p^{\mathrm{PI}}(\mathbf{x}, 10)$ for the benchmarks for $d = 1$ and $p^{\mathrm{PI}}_{\mathrm{st}}(\mathbf{x}) = p^{\mathrm{PI}}(\mathbf{x}, 40)$ for $d = 2$. Furthermore, in the case of the dense interpolations we do not investigate resolutions $N > 51$ ($d = 1$) and $N > 101$ ($d = 2$) since those methods minimise the errors for smaller N values, as the plateau in Fig. 3 shows.

In Fig. 6a we see that in the case of $d = 1$ the dense interpolation methods are capable of producing accurate results orders of magnitude faster than the sparse interpolation methods, especially compared to the direct and FFT-based evaluation approaches. As

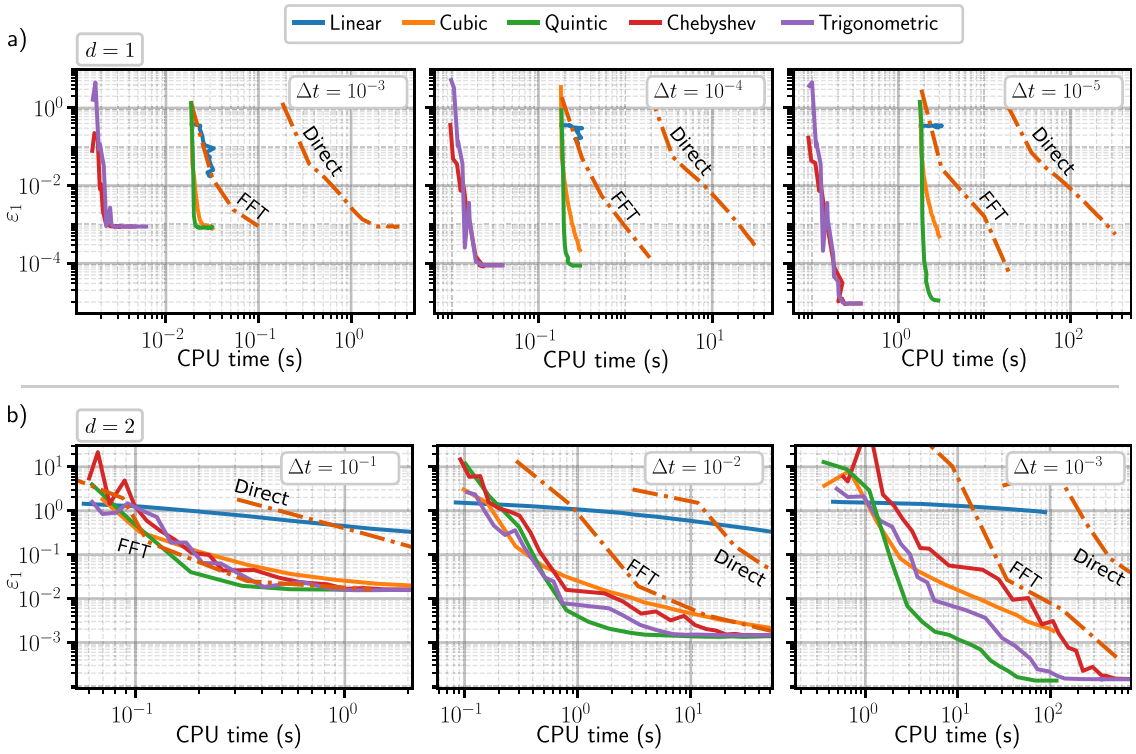

Fig. 6. Error ε_1 of the PI approximation of the PDF $p_{\text{st}}(x)$ in (30) as the function of the CPU time required to approximate the steady-state solution $p_{\text{st}}(x)$ wit.h the PI method.

already seen in Fig. 4, the linear interpolation is unable to capture the structure of the PDF $p_{\text{st}}(x)$ for small time steps Δt, and thus its poor performance can not be improved with increased CPU time. The large differences between the time necessary to obtain the dense and sparse interpolated approximation of the steady-state PDF p_{st} are due to the representation of the step matrix $\mathbf{S}$. We reach the maximum accuracy with the dense interpolations at $N = 37$ and $\Delta t = 10^{-5}$, while the performance of the matrix multiplication with a sparse matrix is only better for $N > 101$, as shown in Fig. 5. We also see the efficiency benefits in this case from transforming the CK Eq. (3) to a matrix multiplication. This new approach computes an accurate approximation of the PDF of the system (29) that is at least an order of magnitude faster than the FFT-based computation and almost three orders of magnitude faster than the direct approach.

In the case of $d = 2$ shown in Fig. 6b, the efficiency advantage is not as obvious as in $d = 1$. For the largest time step size $\Delta t = 10^{-1}$ shown, the computational cost of setting up $\mathbf{S}$ combined with the small number of time steps (400 steps to reach $T = 40$) results in similar efficiency for each interpolation method, except the linear interpolation. As we decrease the time step to increase accuracy, the number of steps needed for convergence to steady-state increases. The SMM-PI approach is again more efficient than the FFT-based evaluation of the CK equation. The sparse quintic interpolation outperforms every other method, with a greater advantages observed for the smallest investigated time step $\Delta t = 10^{-3}$. Even though the dense interpolations use a smaller number of interpolation nodes to reach the desired accuracy, it is more expensive to compute the time evolution and the steady-state PDF of a dynamical system. This is due to the complexity difference of the sparse and dense interpolations, and the larger number of nonzero elements in the corresponding sparse step matrices $\mathbf{S}$ for the dense methods, that leads to slower matrix multiplications as well. This observation implies that for problems with large step matrices $\mathbf{S}$ (e.g. problems with $d > 2$) the accuracy benefit of the dense interpolation is outweighed by the slower performance of the matrix multiplication corresponding to the time stepping.

3.2.3. Memory requirements

Another important aspect of the PI method is the memory required to store the step matrix $\mathbf{S}$, as its size increases exponentially with d. For example, in this section the step matrix is $\mathbf{S} \in \mathbb{R}^{N^d \times N^d}$, meaning that $\mathbf{S}$ has $v_{\mathbf{S},\text{full}} = N^{2d}$ elements. To reduce the memory needed to store the step matrix we use compressed sparse row (CSR) sparse matrices. The density $\rho_{\mathbf{S}} \leqslant 1$ of the step matrix $\mathbf{S}$ is defined by the ratio of the nonzero elements and the total number of elements in the matrix $\mathbf{S}$. Due to their nature of using all the grid values participating in the interpolation, the dense interpolations always produce a full step matrix $\mathbf{S}$ ($\rho_{\mathbf{S}} = 1$), thus we consider the elements that satisfy the condition

$$|S_{ij}| < \max_{i,j}(|S_{ij}|) \times 10^{-8} \tag{37}$$

as zero elements. Here S_{ij} denotes a single element of $\mathbf{S}$, while $\max_{i,j}(|S_{ij}|)$ denotes the element with the maximum absolute value.

The number of elements in a CSR matrix with a density $\rho_{\mathbf{S}}$ is the sum of the $\rho_{\mathbf{S}} N^{2d}$ number of nonzero elements, the $\rho_{\mathbf{S}} N^{2d}$ number of the corresponding row indices, and the $N^d + 1$ column pointers, that assign the row indices to columns. Then the required number of stored numbers is

$$v_{\mathbf{S},\text{sparse}} = 2\rho_{\mathbf{S}} N^{2d} + N^d + 1. \tag{38}$$

This means that if $v_{\mathbf{S},\text{sparse}} < v_{\mathbf{S},\text{full}} = N^{2d}$, then it is beneficial to store $\mathbf{S}$ as a sparse matrix, as far as memory requirements are concerned. This limit density $\rho_{\mathbf{S}}^{\text{lim}}(N)$ where $v_{\mathbf{S},\text{sparse}} = v_{\mathbf{S},\text{full}}$ is given by

$$\rho_{\mathbf{S}}^{\text{lim}}(N) = \frac{N^{2d} - N^d - 1}{2N^{2d}} \quad \text{and} \quad \rho_{\mathbf{S},\infty}^{\text{lim}} = \lim_{N\to\infty} \rho_{\mathbf{S}}^{\text{lim}}(N) = \frac{1}{2}. \tag{39}$$

Fig. 7 presents the density, the number of elements $v_{\mathbf{S}} = \max(v_{\mathbf{S},\text{full}}, v_{\mathbf{S},\text{sparse}})$ and the net memory requirements of storing the step matrix **S**. When approximating the memory requirement of the step matrix **S** we assume, that each element or index is represented by a 64 bit double precision floating point number or a 64 bit integer. For $d = 1$ and $d = 2$ we measured the memory requirement directly, while for $d = 3$ and $d = 4$ we approximated them using the average density based of the 1000 rows previously used to find the computational time approximations (hence we plot as dashed lines). For $d = 3$ and $d = 4$ the black crosses ("+") denote exact measurements of the density and the number of elements/memory requirement for two specific cases with quintic interpolation.

We see that for $d = 1, 2$, the dense interpolations produce step matrices **S** with density over or at the limit $\rho_{\mathbf{S}}^{\lim}(N)$ throughout the range of N investigated, indicating that the use of a dense matrix representation is reasonable. For $d = 3$ and 4, the sparse interpolations produce sparse step matrices for most d and N combinations shown, thus we use sparse matrices to store **S**.

Even there is a steady decline in the density of the step matrices **S**, the number of its nonzero elements grows, along with the memory needed to store the matrix. The density and thus the memory requirement of the dense and sparse interpolations are separated into two clearly distinguishable groups. This separation is due to the different approaches employed by the two types of interpolation: in dense interpolation we assign a weight for each N^d interpolation grid value, while in sparse interpolations we assign a weight for a fixed number of node values (two for the linear, four for the cubic and six for the quintic interpolation). Even though condition (37) is used for the dense interpolations to exclude certain elements of the matrix **S**, the number of nonzero elements in the resulting step matrix **S** is multiple orders of magnitude larger than the number of elements in the step matrix obtained using the sparse interpolations.

3.3. Comparison with FFT approach for higher dimensional systems

In the previous sections we compared the efficiency for $d = 1$ and $d = 2$. In this section we provide a comparison of the SMM-PI and the FFT-based approaches for the higher order systems (33) ($d = 3$) and (34) ($d = 4$). We focus the comparison on the error ε_1, errors for the marginal densities, and total CPU time.

As neither (33) nor (34) is subjected to parametric noise, their response PDFs can be well approximated in the Fourier space (the response PDFs have decaying tails in each direction). Thus the FFT-based approach [17] is a potential candidate to consider along with the SMM-PI approach presented in this paper. Through this comparison, we consider whether the computation of the step

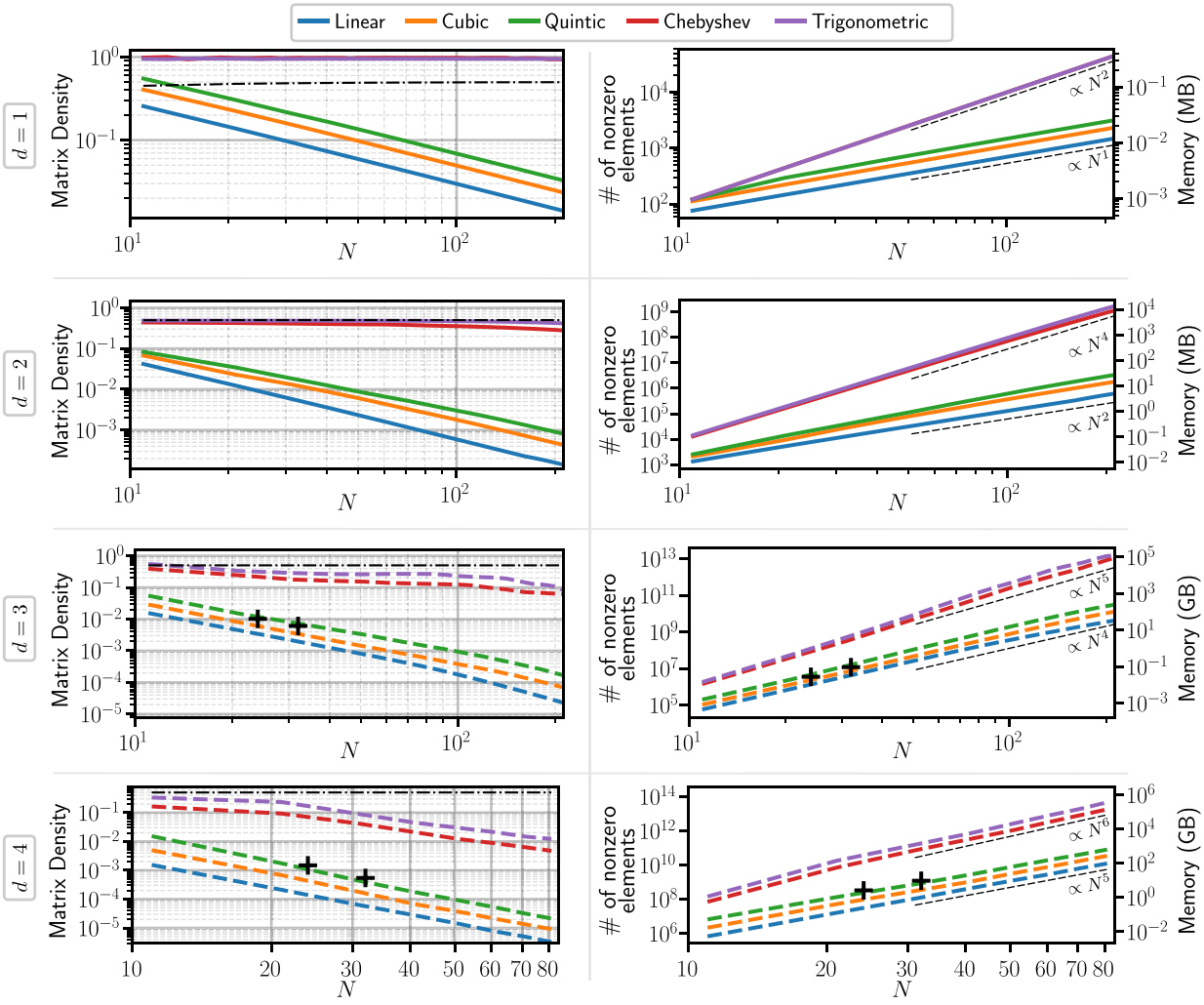

Fig. 7. The matrix density, the number of nonzero elements and the memory requirement of the step matrix **S** for different interpolation methods. The continuous lines denote exact measurements and the dashed lines correspond to approximations. The black dash-dotted line corresponds to $\rho_{\mathbf{S}}^{\lim}(N)$. The black crosses ("+") for $d = 3$ and 4 are validating measurements at $N = 24$ and 32 with cubic interpolation.

matrix $\mathbf{S}$ is indeed worth the computational overhead, as it may be computationally expensive.

We recall that the FFT-based method is compatible only with resolutions that are powers of two, namely $N = 2^m, m \in \mathbb{N}_+$, and that the TPDF $p(\mathbf{x}, t_{n+1}|\mathbf{y}, t_n)$ is sampled at the interpolation nodes when the integration is implemented. This can cause issues when using a small time step Δt, resulting in a small discretised diffusion term $g_k(\mathbf{x}, t)\sqrt{\Delta t_n}$ for which the TPDF $p(\mathbf{x}, t_{n+1}|\mathbf{y}, t_n)$ is narrow, with no or limited overlap with the interpolation nodes. To properly sample the interaction between the PDF and the TPDF, a high and computationally expensive spatial resolution $N = 2^m$ is necessary, even though a lower resolution N would be enough to characterise the PDF. In contrast, for the SSM-PI method we narrow the integration region to $\bar{\mathscr{I}}_{(k,d,i)}$, namely we omit the regions from the integration where we approximate that the TPDF is zero, as described in Section 2.4. In Tables 1 and 2 we present the error ε_1 and total computation time of the steady-state PDF of (33) ($d = 3$) and (34) ($d = 4$) for different resolutions N for the SMM-PI and FFT-based approaches. For the PI computations we use the Runga-Kutta-Maruyama method with a time step $\Delta t = 10^{-2}$ for $d = 3$ and $\Delta t = 10^{-1}$ for $d = 4$, quintic interpolation for the SMM-PI, and cubic B-spline interpolation for the FFT-based approach. The error ε_1 was computed by comparing the results obtained by the PI method to the Gaussian PDF characterised by (D.2) in Appendix D.

For $d = 3$ we see that the FFT-based approach yields numerically unstable solutions, which exhibit large oscillations for increasing N. This instability stems from the fact that the FFT-based approach cannot capture the interaction between the TPDF and PDF for this example, even at higher spatial resolutions $N = 128$. The large error ε_1 for the FFT-based method are due to large oscillations in the solutions. In contrast, the SMM-PI approach was able to produce reasonably accurate results even for $N = 21$, with ε decreasing with N. Fig. 8 compares the steady-state marginal PDFs obtained with the SMM-PI and the FFT-based PI method with the analytical results. Once again the FFT-based method shows oscillations in the solution at $N = 128$, while the SMM-PI captures the steady-state marginal PDFs of (33) already at $N = 21$. Fig. 8c shows that the FFT-based method is able to capture the interaction between the TPDF $p(\mathbf{x}, t_{n+1}|\mathbf{y}, t_n)$ and PDF $p(\mathbf{x}, t_n)$ for a larger time step $\Delta t = 10^{-1}$, thus eliminating the oscillations in the solution. However, in this case the large time step causes a different numerical instability; thus we get almost uniform marginal distributions shown for $p_{v,\mathrm{st}}(v)$ and $p_{Z,\mathrm{st}}(Z)$.

When considering the error ε_1 for $d = 4$ in Table 2 we see that it gradually decreases with increasing N for the SMM-PI approach. In contrast, for the FFT-based apporach ε_1 is not consistently decreasing as N increases, and is substantially larger than the error of the solution obtained through the SMM-PI approach. Fig. 9a presents the steady-state marginal PDFs of (34) compared to the analytical solution based on Appendix D. We see that with $N = 17$ the SMM-PI approach yields a false solution, while the steady-state marginal PDFs are well approximated with $N = 21$. In Fig. 9b we see, that with the FFT-based approach we get a solution with seemingly improving accuracy for the marginal PDFs as we increase N, even though the error measure ε_1 does not decrease significantly.

To better understand the apparent discrepancy between ε_1 and the behavior of the marginal PDFs in Fig. 9b we inspect the error distribution of the method by calculating the marginal errors $\varepsilon_{1,x}(x), \varepsilon_{1,y}(y), \varepsilon_{1,V_x}(V_x), \varepsilon_{1,V_y}(V_y)$ in Fig. 10, defined analogously to the marginal PDF:

$$\begin{aligned}
\varepsilon_{1,x}(x) &:= \int_{\mathbb{R}^3} \left| p_{\mathrm{st}}^{\mathrm{A},4}([x,y,V_x,V_y]^\top) - p_{\mathrm{st}}^{\mathrm{PI},4}([x,y,V_x,V_y]^\top) \right| \mathrm{d}y\,\mathrm{d}V_x\,\mathrm{d}V_y,\\
\varepsilon_{1,y}(y) &:= \int_{\mathbb{R}^3} \left| p_{\mathrm{st}}^{\mathrm{A},4}([x,y,V_x,V_y]^\top) - p_{\mathrm{st}}^{\mathrm{PI},4}([x,y,V_x,V_y]^\top) \right| \mathrm{d}x\,\mathrm{d}V_x\,\mathrm{d}V_y,\\
\varepsilon_{1,V_x}(V_x) &:= \int_{\mathbb{R}^3} \left| p_{\mathrm{st}}^{\mathrm{A},4}([x,y,V_x,V_y]^\top) - p_{\mathrm{st}}^{\mathrm{PI},4}([x,y,V_x,V_y]^\top) \right| \mathrm{d}x\,\mathrm{d}y\,\mathrm{d}V_y,\\
\varepsilon_{1,V_y}(V_y) &:= \int_{\mathbb{R}^3} \left| p_{\mathrm{st}}^{\mathrm{A},4}([x,y,V_x,V_y]^\top) - p_{\mathrm{st}}^{\mathrm{PI},4}([x,y,V_x,V_y]^\top) \right| \mathrm{d}y\,\mathrm{d}x\,\mathrm{d}V_x.
\end{aligned} \tag{40}$$

These quantities give us an insight on the cumulative errors in each direction for the two approaches with different resolutions. We see, that even though the marginal PDFs in Fig. 9a and b appear very similar, the error ε_1 along with the marginal errors $\varepsilon_{1,x}(x), \varepsilon_{1,y}(y), \varepsilon_{1,V_x}(V_x), \varepsilon_{1,V_y}(V_y)$ are significantly larger in the case of the FFT method. This means, that the response PDF $p^{\mathrm{PI},4}(\mathbf{x})$ obtained through the FFT method differs significantly from the analytical solution $p^{\mathrm{A},4}(\mathbf{x})$, even though they have similar marginal PDFs. Thus we see the importance of having a systematic PI approach that can reliably address computational error. An inspection of the shape of the marginal PDFs alone may lead us to incorrectly conclude that the full solution is correct, thus leading to false conclusions when this steady-state PDF is used in a practical problem. Even though the FFT-based method is able to produce results quickly one has to verify the results before accepting them, while the SMM-PI method does not have this issue.

Additionally, the FFT-based approach cannot address systems described by Eq. (1) in full generality, e.g., in case the system is subjected to parametric noise excitation or it is an impacting system with non-symmetric response PDFs. Thus in general, we should use the SMM-PI approach, unless the step matrix $\mathbf{S}$ cannot fit into the memory of the computation. For $d = 4$ this is a legitimate concern, as in the example above, even for $N = 31$ we need 6.9 GB memory for the step matrix obtained with quintic interpolation. If that resolution is not sufficient, and we need a resolution of $N = 41$ or 51, the step matrix $\mathbf{S}$ uses 31.6 GB or 277 GB of memory, respectively. If this large amount of memory is not available and if the FFT-based method is not efficiently applicable, we should utilise the direct approach. However, we should consider the direct approach as a last resort, as according to the benchmarks it is the least efficient method to evaluate the CK Eq. (3).

4. Discussion of choosing appropriate spatial and temporal resolutions

The transformation of the CK Eq. (3) to a matrix multiplication has great potential for increasing the efficiency of the PI method. At the same time, given the results in the previous sections, one has to consider a few things when using the method to obtain the time evolution or the steady state of the PDF of a stochastic dynamical system.

Table 1
Error ε_1 and total computation times of the approximation of the steady-state response PDF $p_{\mathrm{st}}(\mathbf{x}) \approx p(\mathbf{x}, 110)$ of the system (33) ($d = 3$) with time step $\Delta t = 10^{-2}$. For the SMM-PI approach the quintic interpolation and for the FFT approach cubic B-spline interpolation was used. We emphasize that the results obtained using FFT are numerically unstable with the abbreviation NU.

	SMM-PI			FFT		
N	21	25	31	32	64	128
Error ε_1	0.0145	0.0113	0.0102	3.3×10^{10} (NU)	3.7×10^{12} (NU)	6.8 (NU)
Total CPU time	24.10 s	44.29 s	105.82 s	187.17 s	1531.2 s	2.97 h

Table 2
Error ε_1 and total computation times of the approximation of the steady-state response PDF $p_{\text{st}}(\mathbf{x}) \approx p(\mathbf{x}, 40)$ of the system (34) ($d = 4$) with time step $\Delta t = 10^{-1}$. For the SMM-PI approach the quintic interpolation and for the the FFT-based approac cubic B-spline interpolation was used.

	SMM-PI						FFT		
N	15	17	19	21	25	31	16	32	64
Error ε_1	0.8607	0.4402	0.2164	0.1195	0.06105	0.0367	1.112	1.066	1.064
Total CPU time	278 s	802 s	0.584 h	1.18 h	4.70 h	29.42 h	13.626 s	312.8 s	1.687 h

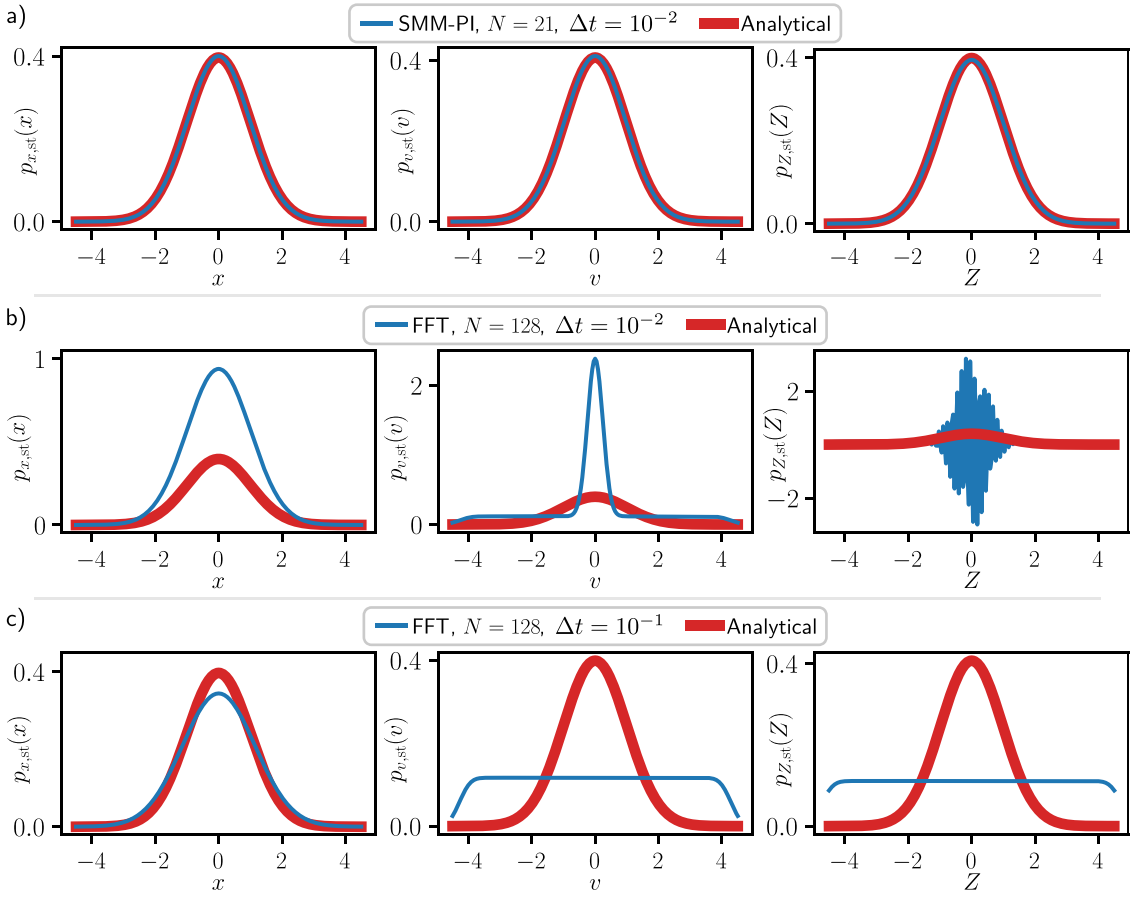

Fig. 8. Comparison of the marginal steady-state response PDFs of the system (33) ($d = 3$) with the approximated solutions obtained by (a) the matrix mulitplication approach with quintic interpolation, (b)-(c) the FFT-based approach with cubic B-splin.e interpolation.

First of all, it is necessary to choose the interpolation region $\mathscr{I}$. In general, if there is no analytical solution or approximation that provide the relevant moments or the PDF evolution, preliminary Monte-Carlo simulations are required for an initial estimation for $\mathscr{I}$. Based on the simulated realisations, we can choose a proper interpolation region based on the empirical response histogram of the system, or the steady-state central second moment of the state variables, e.g. by taking a 6 standard deviation region of the process. For the examples (29)–(34) we had the analytical solution for the steady state PDF: for $d = 1$ and $d = 2$ the solutions are presented in (30) and (32) while for $d = 3$ and 4 the steady state PDF is provided in Appendix D.

After we have the interpolation region $\mathscr{I}$, we choose proper temporal discretisation methods and resolutions. First, we need to choose the time stepping method and size Δt_n for which the discretised SDE (4) captures the dynamics of the continuous SDE (1) accurately and without introducing numerical instability over the region $\mathscr{I}$. A small Δt allows a more accurate approximation of both the drift and diffusion, however, a very small Δt can lead to numerical issues, as the TPDF $p(\mathbf{x}, t_{n+1}|\mathbf{y}, t_n)$ with a small time step is highly concentrated near $\mathbf{y}$ for that step, so that integration of the TPDF can magnify the interpolation errors without sufficient spatial resolution N.

While there are various considerations to take into account, e.g. balancing N and Δt, as were considerations in previous approaches, this new approach provides a systematic way of implementing the PI method and addressing potential sources of computational error, in contrast to previous methods that did not. In Fig. 4 we demonstrated the effect of different temporal and spatial discretisation, namely, that even though the spatial discretisation was sufficient for a given Δt_n, decreasing the time step may increase the error for the same N, as mentioned above. In the case of linear interpolation the method is not able to accurately resolve small fluctuations in the density for a small time step Δt, and instead tends to smooth spatial variation of $p(\mathbf{x}, t)$. At the same time, the higher order (cubic, quintic and dense) interpolations can introduce well-known oscillatory instabilities if the resolution is too low. In this paper we do not provide a strict analysis of this effect; however, we note that this oscillatory instability is related to the case where the largest eigenvalues of the step matrix $\mathbf{S}$ take multi-

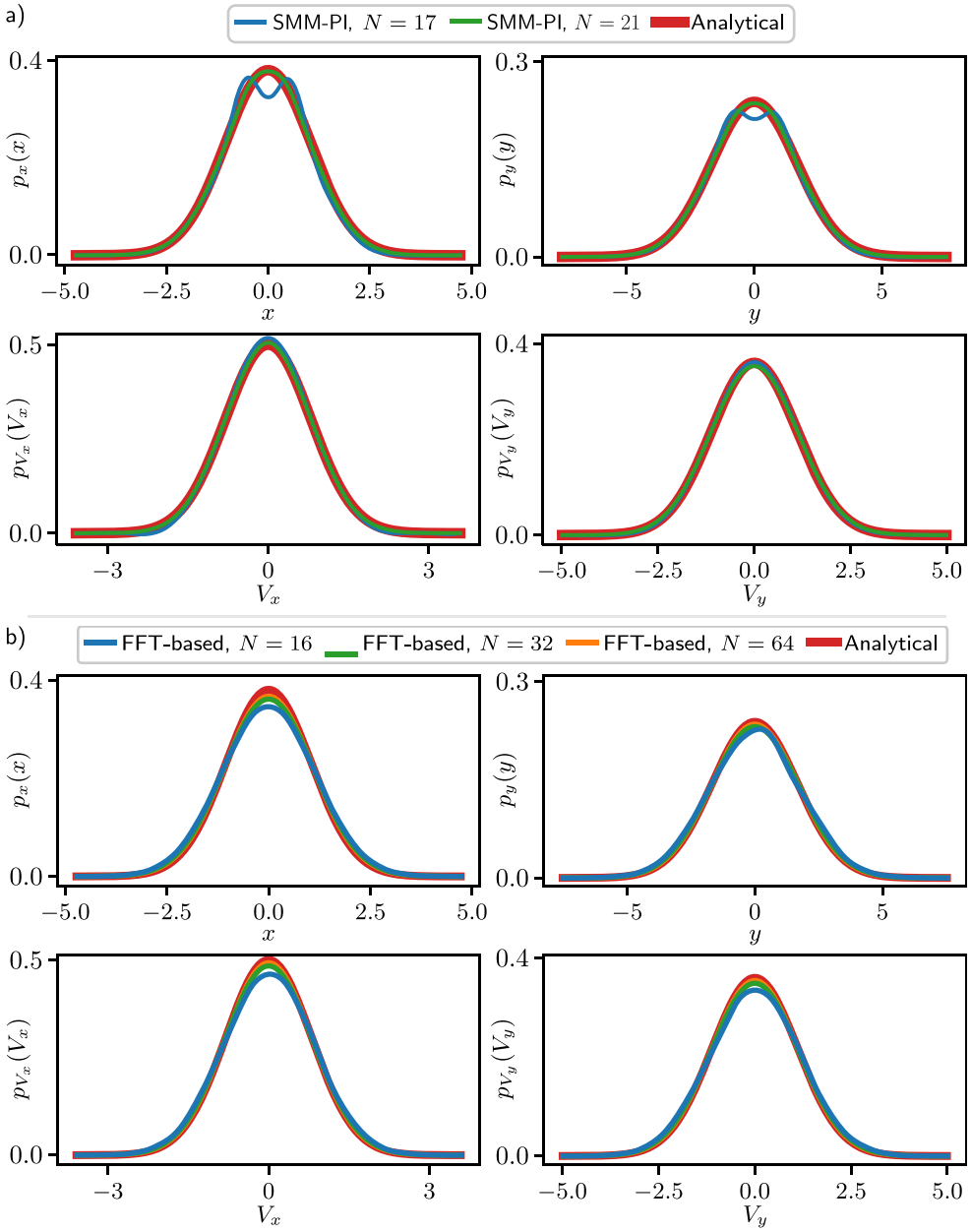

Fig. 9. Comparison of the marginal steady-state response PDFs of the system (34) ($d = 4$) with the approximated solutions obtained (a) by the SMM-PI approach with quintic interpolation and (b) by the FFT-based approach with cubic B-spline interpolation. The time step is $\Delta t = 10^{-1}$.

ple, usually complex, values. This spectral structure leads to false solutions, as the steady state $\lim_{n\to\infty}\text{vec}(\mathbf{q}_n)$ in (23) converges to the corresponding eigenvector.

In essence, the SMM-PI formulation requires an adjustment of the resolution of the spatial and temporal discretisation to assure sufficient accuracy both for the approximation of the PDF $p(\mathbf{x}, t_n)$ and its interaction with the TPDF $p(\mathbf{x}, t_{n+1}|\mathbf{y}, t_n)$ in the evaluation of the CK. The new SMM-PI formulation, based on separate treatment of the three parts of interpolation of the PDF, approximation of the TPDF of the process and evaluation of the CK integral, provides insight on how choose these resolutions. For example, for a d-dimensional system with a single noise excitation ($k = d$) it is a good strategy to initially set the time step Δt_n and spatial resolution N in a way that the ratio of $\frac{b_k - a_k}{N}$ and $g_k(\mathbf{x}, t)\sqrt{\Delta t_n}$ is close to one or smaller, and test the resulting PDF against a histogram obtained through coarse MC simulations. Then we can make adjustments to the resolutions based on comparing the PDF obtained through the SMM-PI with e.g. the one obtained through the MC simulations: we can increase accuracy by decreasing Δt or increasing N, or decide to decrease computational cost by decreasing N. Note that the above recommendations on the initial ratio of Δt and N are based on empirical results, as we did not conduct the analysis on the complex interaction between the multiple sources of error (time evolution, spatial discretisation and numerical integration) to obtain the exact error convergence rates and numerical stability criteria.

For small dimensional stochastic systems (e.g. $d = 1$ or 2) increasing N can be done at a low computational cost: if the spatial resolution N is not sufficient for the chosen time step Δt_n, then we can increase the resolution N at a small performance penalty. As the size of the investigated system increases (e.g., $d > 3$) one has to carefully choose a balance between Δt and N giving the corre-

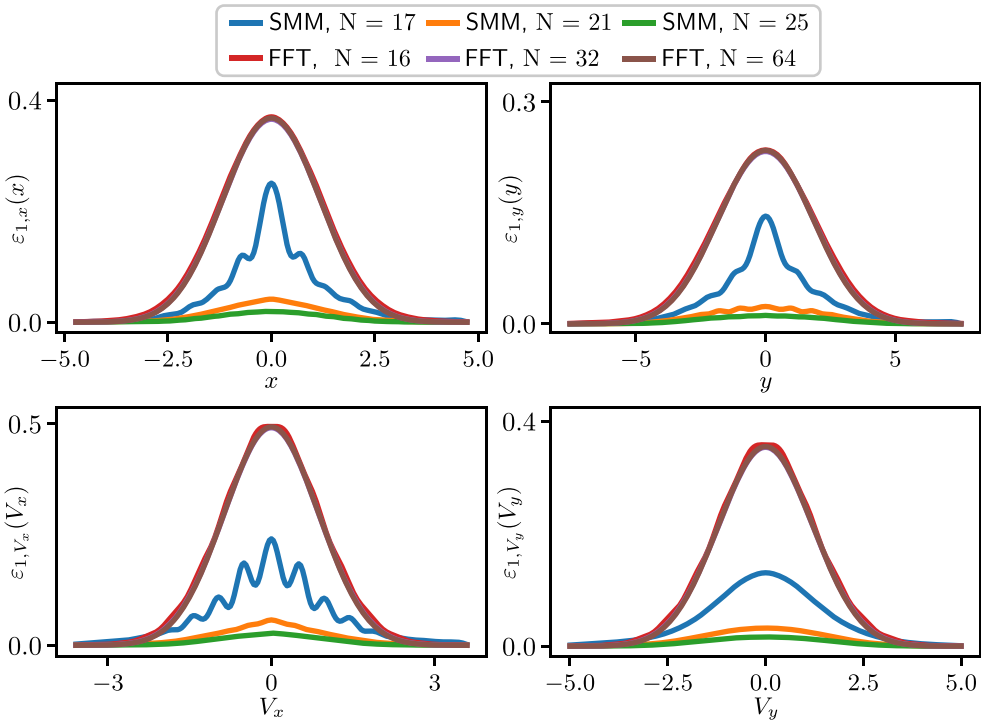

Fig. 10. Comparison of the marginal errors $\varepsilon_{1,\cdot}$ of the system (34) ($d = 4$) with the approximated solutions obtained by SMM-PI with quintic interpolations and FFT-based (FFT) approaches with cubic B-spline interpolation. The time step is $\Delta t = 10^{-1}$.

sponding spatial resolution, since the computation of the step matrix **S** becomes very expensive in terms of both computational time and memory.

To summarise, the SMM-PI method is especially useful when analysing the models of low-dimensional nonlinear systems ($d \leqslant 3$) and investigate the effect of parameters on the time evolution of the PDF.

5. Conclusions

In this paper we provided a novel systematic formulation of the PI method as an approximate method to solve the Chapman-Kolmogorov (CK) Eq. (3). The CK equation models the evolution of the PDF for a stochastic process in general, and we apply it to systems described by SDE's. There are three key elements of the new PI construction: the approximation of the TPDF $p(\mathbf{x}, t_{n+1}|\mathbf{y}, t_n)$, the discretisation of the PDF $p(\mathbf{x}, t_n)$, and the quadrature evaluation of the integral. Since the TPDF is the fundamental component that advances the PDF in time we approximate it using the PDF of a numerical time stepping scheme applied to the SDE (1). For the discretisation of the PDF $p(\mathbf{x}, t_n)$ we use different interpolation methods, and to evaluate the integral in (3) that combines components, we use the Gauss–Legendre quadrature. Partitioning the PI method in this manner allows us to separately investigate the performance of the approximation for each component and to isolate the potential bottlenecks within the method. Furthermore, through this formulation we transform the Chapman-Kolmogorov equation to multiplication by step matrix that captures the full PDF evolution, thus boosting the performance of the PI method.

To test the accuracy, computational requirements and the efficiency of the PI method with the different temporal and spatial approximation methods and resolutions, we conducted numerical experiments on a set of nonlinear and linear examples. We note that this study appears to be the first systematic performance evaluation of any PI method. The results presented in Fig. 6 show, that in the case of the scalar system ($d = 1$) the most efficient approximations for the PDF use dense interpolation approaches, for which the method yields accurate results even with relatively small size step matrices $\mathbf{S}_n$. For dimension $d = 2$, the efficiency of the quintic interpolation method surpasses that of the dense interpolations, and our tests indicate that this conclusion holds for higher d-dimensional systems as well. The numerical tests also confirm, that for $d = 1, 2, 3$, transforming the CK Eq. (3) to a matrix multiplication significantly increases the efficiency of the PI method, compared to previous approaches, even before considering any parallelisation.

The computationally most expensive part of the SMM-PI approach is the generation of the step matrix $\mathbf{S}_n$. Nevertheless, since the new formulation provides the advantage that the row vectors $\mathrm{vec}(\mathbf{\Phi}_{(i),n})^\top$ of $\mathbf{S}_n$ defined in (22) are independent of each other, we can greatly reduce the computation time necessary to obtain $\mathbf{S}_n$ by the parallel computation of the individual rows, further increasing the efficiency of the SMM-PI method. However, there is a single caveat to this approach based on Fig. 7: a large amount of memory is necessary to store $\mathbf{S}_n$, as shown for $d \geqslant 4$, even when a sparse interpolation is applied. This is a major barrier to be overcome in order to generalise this approach to higher d-dimensional systems. Nonetheless, the method presented in this paper is proved to be remarkably efficient and is a potential candidate for the high-performance computational solution of nonlinear (time-dependent) stochastic systems (subjected to various noise sources) that are used for modelling in biology, physics, engineering, finance or other fields of science, even on thin and light laptops with moderate computational capabilities.

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.

Acknowledgements

The authors gratefully acknowledge partial funding for this work from NSF-CMMI 2009270 and EPSRC EP/V034391/1.

Appendix A. Runge–Kutta-Maruyama methods

To increase the accuracy of the approximation of the TPDF $p(\mathbf{x}, t_{n+1}|\mathbf{y}, t_n)$ we can use a Runge–Kutta step for the drift term. In this case (4) changes to

$$\mathbf{x}(t_{n+1}) \approx \mathbf{x}(t_n) + \hat{\mathbf{f}}(\mathbf{x}(t_n), t_n)\Delta t_n + \mathbf{g}(\mathbf{x}(t_n), t_n)\Delta \mathbf{W}_n. \tag{A.1}$$

We use the explicit 4th order Runge–Kutta approximation [28] $\hat{\mathbf{f}}(\mathbf{x}(t_n), t_n)$ of the drift term, that is

$$\hat{\mathbf{f}}(\mathbf{x}(t_n), t_n) = \frac{1}{6}(\mathbf{k}_1 + \mathbf{k}_2 + \mathbf{k}_3 + \mathbf{k}_4), \tag{A.2}$$

where

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{f}(\mathbf{x}(t_n), t_n), \\ \mathbf{k}_2 &= \mathbf{f}\big(\mathbf{x}(t_n) + \tfrac{1}{2}\mathbf{k}_1\Delta t_n, t_n + \tfrac{1}{2}\Delta t_n\big), \\ \mathbf{k}_3 &= \mathbf{f}\big(\mathbf{x}(t_n) + \tfrac{1}{2}\mathbf{k}_2\Delta t_n, t_n + \tfrac{1}{2}\Delta t_n\big), \\ \mathbf{k}_4 &= \mathbf{f}\big(\mathbf{x}(t_n) + \mathbf{k}_3\Delta t_n, t_n + \Delta t_n\big). \end{aligned} \tag{A.3}$$

Note that with the help of Butcher tableaus we can construct the approximation $\hat{\mathbf{f}}(\mathbf{x}(t_n), t_n)$ using different explicit Runge–Kutta methods.

Appendix B. Interpolation functions

Here we collect the weight functions $\phi_{j,l}$ of the interpolation methods used to construct the interpolations $\bar{p}(\mathbf{x}, t_n)$ in (15). We differentiate two groups of interpolations: sparse and dense interpolations. In case of the sparse interpolations we assign a non-zero weight only to a fixed number of node values $q_{(i),n}$ independent of the number of nodes N_j along the $j-th$ dimension, while in case of dense interpolations we assign weights for each N_j node value $q_{(i),n}$. Also, as the interpolation functions are described only along the single j-th dimension, as abuse of notation we omit j from the subscripts, and refer to the resolution as N, to the limits of the interpolation region as a and b, to the grid locations as x_l, to the node values as q_l and to the weight functions as $\phi_i(x)$.

For each interpolations we interpolate the function $p : \mathbb{R} \mapsto \mathbb{R}$ with an interpolation function $\bar{p}(x)$ in the form

$$p(x) \approx \bar{p}(x) = \begin{cases} \sum_{i=1}^{N} q_i\phi_i(x) & \text{if } x \in \mathscr{I} = [a, b], \\ 0 & \text{otherwise.} \end{cases} \tag{B.1}$$

Here q_i denotes the values taken by the function $p(x)$ in the interpolation nodes, namely $q_i = p(x_i)$.

B.1. Sparse interpolations

Throughout this section we use the equidistant grid over the j-th dimension with nodes

$$x_l = a + (l-1)\frac{b-a}{N-1}, \tag{B.2}$$

and the local variable θ_l defined as

$$\theta_l(x) := \begin{cases} \frac{x-x_l}{h} & \text{if } x_l < x < x_{l+1}, \\ 0 & \text{otherwise,} \end{cases} \tag{B.3}$$

where $h = x_{l+1} - x_l = \frac{b-a}{N-1}$.

B.1.1. Linear interpolation

Even though the linear interpolation is well-known, we demonstrate the thought-process of how we derive the sparse interpolations used in this paper. During the linear interpolation we approximate the function $p(x)$ on $x_l < x < x_{l+1}$ with the interpolation functions

$$\bar{p}(x) = \alpha_{l,1}\theta_l(x) + \alpha_{l,0}, \tag{B.4}$$

with conditions

$$\begin{aligned} \bar{p}(x_l) &= \alpha_{l,0} = q_l, \\ \bar{p}(x_{l+1}) &= \alpha_{l,1} + \alpha_{l,0} = q_{l+1}, \end{aligned} \tag{B.5}$$

namely, the interpolation function in the nodes x_i return the node values q_i. We solve (B.5) for $\alpha_{l,1}$ and $\alpha_{l,0}$, and substitute them back to (B.4) we get

$$\bar{p}(x) = (q_{l+1} - q_l)\theta_l(x) + q_l. \tag{B.6}$$

Collecting the coefficients of q_i, where $i = 1, \ldots, N$, we obtain the weight functions $\phi_i(x)$ as

$$\begin{aligned} \phi_l(x) &= 1 - \theta_l(x), \\ \phi_{l+1}(x) &= \theta_l(x), \\ \phi_i(x) &= 0 \quad \text{for all} \quad i \neq l,\ i \neq l+1. \end{aligned} \tag{B.7}$$

Note that in case of the linear interpolation function $\bar{p}$ is C^0 on $\mathscr{I}$.

B.1.2. Cubic interpolation

In case of the cubic interpolation we approximate the $p(x)$ on $x_l < x < x_{l+1}$ with the cubic polynomial

$$\bar{p}(x) = \alpha_{l,3}\theta_l^3(x) + \alpha_{l,2}\theta_l^2(x) + \alpha_{l,1}\theta_l(x) + \alpha_{l,0}, \tag{B.8}$$

with boundary conditions conditions for the values

$$\begin{aligned} \bar{p}(x_l) &= \alpha_{l,0} = q_l, \\ \bar{p}(x_{l+1}) &= \alpha_{l,3} + \alpha_{l,2} + \alpha_{l,1} + \alpha_{l,0} = q_{l+1}, \end{aligned} \tag{B.9}$$

and for the first derivatives

$$\begin{aligned} \bar{p}\prime(x_l) &= \alpha_{l,1}/h = \Delta_{h,1}^{(3)}\{p\}(l), \\ \bar{p}\prime(x_{l+1}) &= (3\alpha_{l,3} + 2\alpha_{l,2} + \alpha_{l,1})/h = \Delta_{h,1}^{(3)}\{p\}(l+1), \end{aligned} \tag{B.10}$$

where $\Delta_{h,1}^{(3)}\{p\}(l)$ denotes the 3-point finite difference approximation of the first derivative $p\prime(x_l)$, namely,

$$\Delta_{h,1}^{(3)}\{p\}(l) = \begin{cases} \frac{-q_3+4q_2-3q_1}{2h} & \text{if } l = 1, \\ \frac{-q_N+4q_{N-1}-3q_{N-2}}{2h} & \text{if } l = N-1, \\ \frac{q_{l+1}-q_{l-1}}{2h} & \text{otherwise.} \end{cases} \tag{B.11}$$

To obtain ϕ_i we solve the linear equation system (B.9) and (B.10) for $\alpha_{l,m}, m = 0, 1, 2, 3$, substitute the solution back to (B.8) and collect the coefficients of q_i.

In case of a general l the non-zero weight functions ϕ_i are

$$\begin{aligned} \phi_{l-1}(x) &= \tfrac{1}{2}\theta_l^3(x) + \theta_l^2(x) - \tfrac{1}{2}\theta_l(x), \\ \phi_l(x) &= \tfrac{1}{2}\theta_l^3(x) - \tfrac{5}{2}\theta_l^2(x) + 1, \\ \phi_{l+1}(x) &= -\tfrac{3}{2}\theta_l^3(x) + 2\theta_l^2(x) + \tfrac{1}{2}\theta_l(x), \\ \phi_{l+2}(x) &= \tfrac{1}{2}\theta_l^3(x) - \tfrac{1}{2}\theta_l^2(x). \end{aligned} \tag{B.12}$$

In case of $l = 1$ $(x_1 < x < x_2)$ the non-zero weight functions ϕ_i are

$$\begin{aligned} \phi_1(x) &= \tfrac{1}{2}\theta_1^2(x) - \tfrac{3}{2}\theta_1(x) + 1, \\ \phi_2(x) &= -\theta_1^2(x) + 2\theta_1(x), \\ \phi_3(x) &= \tfrac{1}{2}\theta_1^2(x) - \tfrac{1}{2}\theta_1(x), \end{aligned} \tag{B.13}$$

and in case of $l = N-1$ $(x_{N-1} < x < x_N)$ the non-zero weight functions are written as

$$\begin{aligned}\phi_{N-2}(x) &= \tfrac{1}{2}\theta_{N-1}^2(x) - \tfrac{1}{2}\theta_{N-1}(x),\\ \phi_{N-1}(x) &= -\theta_{N-1}^2(x) + 1,\\ \phi_N(x) &= \tfrac{1}{2}\theta_{N-1}^2(x) + \tfrac{1}{2}\theta_{N-1}(x).\end{aligned} \tag{B.14}$$

Note that in case of the cubic interpolation function $\bar{p}$ is C^1 on $\mathscr{I}$.

B.1.3. Quintic interpolation

In case of the quintic interpolation we use the quintic polynomial

$$\bar{p}(x) = \alpha_{l,5}\theta_l^5(x) + \alpha_{l,4}\theta_l^4(x) + \alpha_{l,3}\theta_l^3(x) + \alpha_{l,2}\theta_l^2(x) + \alpha_{l,1}\theta_l(x) + \alpha_{l,0} \tag{B.15}$$

With the boundary conditions for the values

$$\begin{aligned}\bar{p}(x_l) &= \alpha_{l,0} = q_l,\\ \bar{p}(x_{l+1}) &= \alpha_{l,5} + \alpha_{l,4} + \alpha_{l,3} + \alpha_{l,2} + \alpha_{l,1} + \alpha_{l,0} = q_{l+1},\end{aligned} \tag{B.16}$$

for the first derivatives

$$\begin{aligned}\bar{p}\prime(x_l) &= \alpha_{l,1}/h = \Delta_{h,1}^{(5)}\{p\}(l),\\ \bar{p}\prime(x_{l+1}) &= (5\alpha_{l,5} + 4\alpha_{l,4} + 3\alpha_{l,3} + 2\alpha_{l,2} + \alpha_{l,1})/h = \Delta_{h,1}^{(5)}\{p\}(l+1),\end{aligned} \tag{B.17}$$

and the second derivatives

$$\begin{aligned}\bar{p}\prime\prime(x_l) &= 2\alpha_{l,2}/h^2 = \Delta_{h,2}^{(5)}\{p\}(l),\\ \bar{p}\prime\prime(x_{l+1}) &= (20\alpha_{l,5} + 12\alpha_{l,4} + 6\alpha_{l,3} + 2\alpha_{l,2})/h^2 = \Delta_{h,2}^{(5)}\{p\}(l+1).\end{aligned} \tag{B.18}$$

Here $\Delta_{h,1}^{(5)}\{p\}(l)$ denotes the 5-point finite difference approximation of the first derivative $p\prime(x_l)$, namely,

$$\Delta_{h,1}^{(5)}\{p\}(l) = \begin{cases} \frac{-25q_1+48q_2-36q_3+16q_4-3q_5}{12h} & \text{if } l=1,\\ \frac{-3q_1-10q_2+18q_3-6q_4+q_5}{12h} & \text{if } l=2,\\ \frac{-q_{N-4}+6q_{N-3}-18q_{N-2}+10q_{N-1}+3q_N}{12h} & \text{if } l=N-1,\\ \frac{3q_{N-4}-16q_{N-3}+36q_{N-2}-48q_{N-1}+25q_N}{12h} & \text{if } l=N,\\ \frac{q_{l-2}-8q_{l-1}+8q_{l+1}-q_{l+2}}{12h} & \text{otherwise,}\end{cases} \tag{B.19}$$

while $\Delta_{h,2}^{(5)}\{p\}(l)$ denotes the 5-point finite difference approximation of the second derivative $p\prime\prime(x_l)$, namely,

$$\Delta_{h,2}^{(5)}\{p\}(l) = \begin{cases} \frac{35q_1-104q_2+114q_3-56q_4+11q_5}{12h^2} & \text{if } l=1,\\ \frac{11q_1-20q_2+6q_3+4q_4-q_5}{12h^2} & \text{if } l=2,\\ \frac{-q_{N-4}+4q_{N-3}+6q_{N-2}-20q_{N-1}-11q_N}{12h^2} & \text{if } l=N-1,\\ \frac{11q_{N-4}-56q_{N-3}+114q_{N-2}-104q_{N-1}+35q_N}{12h^2} & \text{if } l=N,\\ \frac{-q_{l-2}+16q_{l-1}-30q_l+16q_{l+1}-q_{l+2}}{12h^2} & \text{otherwise.}\end{cases} \tag{B.20}$$

In case of a general l the non-zero weight functions ϕ_i are

$$\begin{aligned}\phi_{l-2}(x) &= -\tfrac{5}{24}\theta_l^5(x) + \tfrac{13}{24}\theta_l^4(x) - \tfrac{3}{8}\theta_l^3(x) - \tfrac{1}{24}\theta_l^2(x) + \tfrac{1}{12}\theta_l(x),\\ \phi_{l-1}(x) &= \tfrac{25}{24}\theta_l^5(x) - \tfrac{8}{3}\theta_l^4(x) + \tfrac{13}{8}\theta_l^3(x) + \tfrac{2}{3}\theta_l^2(x) - \tfrac{2}{3}\theta_l(x),\\ \phi_l(x) &= -\tfrac{25}{12}\theta_l^5(x) + \tfrac{21}{4}\theta_l^4(x) - \tfrac{35}{12}\theta_l^3(x) - \tfrac{5}{4}\theta_l^2(x) + 1,\\ \phi_{l+1}(x) &= \tfrac{25}{12}\theta_l^5(x) - \tfrac{31}{6}\theta_l^4(x) + \tfrac{11}{4}\theta_l^3(x) + \tfrac{2}{3}\theta_l^2(x) + \tfrac{2}{3}\theta_l(x),\\ \phi_{l+2}(x) &= -\tfrac{25}{24}\theta_l^5(x) + \tfrac{61}{24}\theta_l^4(x) - \tfrac{11}{8}\theta_l^3(x) - \tfrac{1}{24}\theta_l^2(x) - \tfrac{1}{12}\theta_l(x),\\ \phi_{l+3}(x) &= \tfrac{5}{24}\theta_l^5(x) - \tfrac{1}{2}\theta_l^4(x) + \tfrac{7}{24}\theta_l^3(x).\end{aligned} \tag{B.21}$$

In case of $l=1$ ($x_1 < x < x_2$) the non-zero weight functions ϕ_i are

$$\begin{aligned}\phi_1(x) &= \tfrac{1}{24}\theta_1^4(x) - \tfrac{5}{12}\theta_1^3(x) + \tfrac{35}{24}\theta_1^2(x) - \tfrac{25}{12}\theta_1(x) + 1,\\ \phi_2(x) &= -\tfrac{1}{6}\theta_1^4(x) + \tfrac{3}{2}\theta_1^3(x) - \tfrac{13}{3}\theta_1^2(x) + 4\theta_1(x),\\ \phi_3(x) &= \tfrac{1}{4}\theta_1^4(x) - 2\theta_1^3(x) + \tfrac{19}{4}\theta_1^2(x) - 3\theta_1(x),\\ \phi_4(x) &= -\tfrac{1}{6}\theta_1^4(x) + \tfrac{7}{6}\theta_1^3(x) - \tfrac{7}{3}\theta_1^2(x) + \tfrac{4}{3}\theta_1(x),\\ \phi_5(x) &= \tfrac{1}{24}\theta_1^4(x) - \tfrac{1}{4}\theta_1^3(x) + \tfrac{11}{24}\theta_1^2(x) - \tfrac{1}{4}\theta_1(x).\end{aligned} \tag{B.22}$$

In case of $l=2$ ($x_2 < x < x_3$) the non-zero weight functions ϕ_i are

$$\begin{aligned}\phi_1(x) &= \tfrac{1}{24}\theta_2^4(x) - \tfrac{1}{4}\theta_2^3(x) + \tfrac{11}{24}\theta_2^2(x) - \tfrac{1}{4}\theta_2(x),\\ \phi_2(x) &= -\tfrac{1}{6}\theta_2^4(x) + \tfrac{5}{6}\theta_2^3(x) - \tfrac{5}{6}\theta_2^2(x) - \tfrac{5}{6}\theta_2(x) + 1,\\ \phi_3(x) &= \tfrac{1}{4}\theta_2^4(x) - \theta_2^3(x) + \tfrac{1}{4}\theta_2^2(x) + \tfrac{3}{2}\theta_2(x),\\ \phi_4(x) &= -\tfrac{1}{6}\theta_2^4(x) + \tfrac{1}{2}\theta_2^3(x) + \tfrac{1}{6}\theta_2^2(x) - \tfrac{1}{2}\theta_2(x),\\ \phi_5(x) &= \tfrac{1}{24}\theta_2^4(x) - \tfrac{1}{12}\theta_2^3(x) - \tfrac{1}{24}\theta_2^2(x) + \tfrac{1}{12}\theta_2(x).\end{aligned} \tag{B.23}$$

In case of $l=N-2$ ($x_{N-2} < x < x_{N-1}$) the non-zero weight functions are

$$\begin{aligned}\phi_{N-4}(x) &= \tfrac{1}{24}\theta_{N-2}^4(x) - \tfrac{1}{12}\theta_{N-2}^3(x) - \tfrac{1}{24}\theta_{N-2}^2(x) + \tfrac{1}{12}\theta_{N-2}(x),\\ \phi_{N-3}(x) &= -\tfrac{1}{6}\theta_{N-2}^4(x) + \tfrac{1}{6}\theta_{N-2}^3(x) + \tfrac{2}{3}\theta_{N-2}^2(x) - \tfrac{2}{3}\theta_{N-2}(x),\\ \phi_{N-2}(x) &= \tfrac{1}{4}\theta_{N-2}^4(x) - \tfrac{5}{4}\theta_{N-2}^2(x) + 1,\\ \phi_{N-1}(x) &= -\tfrac{1}{6}\theta_{N-2}^4(x) - \tfrac{1}{6}\theta_{N-2}^3(x) + \tfrac{2}{3}\theta_{N-2}^2(x) + \tfrac{2}{3}\theta_{N-2}(x),\\ \phi_N(x) &= \tfrac{1}{24}\theta_{N-2}^4(x) + \tfrac{1}{12}\theta_{N-2}^3(x) - \tfrac{1}{24}\theta_{N-2}^2(x) - \tfrac{1}{12}\theta_{N-2}(x).\end{aligned} \tag{B.24}$$

In case of $l=N-1$ ($x_{N-1} < x < x_N$) the non-zero weight functions are

$$\begin{aligned}\phi_{N-4}(x) &= \tfrac{1}{24}\theta_{N-1}^4(x) + \tfrac{1}{12}\theta_{N-1}^3(x) - \tfrac{1}{24}\theta_{N-1}^2(x) - \tfrac{1}{12}\theta_{N-1}(x),\\ \phi_{N-3}(x) &= -\tfrac{1}{6}\theta_{N-1}^4(x) - \tfrac{1}{2}\theta_{N-1}^3(x) + \tfrac{1}{6}\theta_{N-1}^2(x) + \tfrac{1}{2}\theta_{N-1}(x),\\ \phi_{N-2}(x) &= \tfrac{1}{4}\theta_{N-1}^4(x) + \theta_{N-1}^3(x) + \tfrac{1}{4}\theta_{N-1}^2(x) - \tfrac{3}{2}\theta_{N-1}(x),\\ \phi_{N-1}(x) &= -\tfrac{1}{6}\theta_{N-1}^4(x) - \tfrac{5}{6}\theta_{N-1}^3(x) - \tfrac{5}{6}\theta_{N-1}^2(x) + \tfrac{5}{6}\theta_{N-1}(x) + 1,\\ \phi_N(x) &= \tfrac{1}{24}\theta_{N-1}^4(x) + \tfrac{1}{4}\theta_{N-1}^3(x) + \tfrac{1}{}24\theta_{N-1}^2(x) + \tfrac{1}{4}\theta_{N-1}(x).\end{aligned} \tag{B.25}$$

Note that in case of the cubic interpolation function $\bar{p}$ is C^2 on $\mathscr{I}$.

B.2. Dense interpolations

B.2.1. Barycentric interpolation on a Chebyshev grid

If we use the barycentric interpolation on a Chebyshev grid with N number of nodes x_i, $i=1,2,\ldots,N$, then ϕ_i is written as

$$\phi_i(x) = \begin{cases} 1 & \text{if } x = x_i,\\ 0 & \text{if } x = x_l,\ l \neq i,\\ \frac{(-1)^{i-1}c_{i-1}}{x-x_i} \Big/ \sum_{l=0}^{N-1} \frac{(-1)^l c_l}{x-x_{l+1}}, & \text{otherwise,}\end{cases} \tag{B.26}$$

$$c_i = \begin{cases} 1/2 & \text{if } i=0 \text{ or } i=N-1,\\ 1 & \text{otherwise,}\end{cases} \tag{B.27}$$

with nodes

$$x_i = \frac{a+b}{2} + \frac{b-a}{2}\cos\left(\frac{i-1}{N-1}\pi\right). \tag{B.28}$$

B.2.2. Trigonometric interpolation

If we use trigonometric interpolation along the j-th dimension with N_j number of equidistant nodes x_l, $l=0,1,\ldots,N_j-1$, then $\phi_{j,i}$ is. In case N is odd:

$$\phi_i(x) = \begin{cases} 1 & \text{if } x = x_i,\\ 0 & \text{if } x = x_l,\ l \neq i,\\ \frac{\sin\left(\frac{\pi}{b-a}N(x-x_i)\right)}{N\sin\left(\frac{\pi}{b-a}(x-x_i)\right)}, & \text{otherwise.}\end{cases} \tag{B.29}$$

In case N is even:

$$\phi_i(x) = \begin{cases} 1 & \text{if } x = x_i \\ 0 & \text{if } x = x_l,\ l \neq i \\ \frac{\sin\left(\frac{\pi}{b-a} N_j (x - x_i)\right)}{N_j \tan\left(\frac{\pi}{b-a}(x - x_i)\right)}, & \text{otherwise,} \end{cases} \tag{B.30}$$

During the trigonometric interpolation we the uniform grid

$$x_i = a + (i-1)\frac{(b-a)}{N}, \quad l = 0, \ldots, N_j - 1. \tag{B.31}$$

B.3. Two-dimensional linear interpolation

In this section we demonstrate the construction of the linear interpolation of the function $p(\mathbf{x}), \mathbf{x} \in \mathbb{R}^2,\ p : \mathbb{R}^2 \mapsto \mathbb{R}$ on $\mathbf{x} \in \mathscr{I}$. As p is a 2-dimensional function, we use a two-variable notation, i.e., $p(x,y) := p([x,y]^\top)$, where we use $x := x_1 \in [a_1, b_1]$ and $y := x_2 \in [a_2, b_2]$. We construct the linear interpolation along the two dimensions and show that it indeed results in the form provided in (15).

The function $p(x,y)$ is available in the points (x_{i_1}, y_{i_2}) of the equidistant grid in (B.2) defined along both dimensions, namely,

$$q_{i_1,i_2} = p(x_{i_1}, y_{i_2}) \quad \text{where} \quad i_1 = 1, \ldots, N_1, \text{and} \, i_2 = 1, \ldots, N_2. \tag{B.32}$$

During the interpolation process we approximate $p(x,y)$ between the gridpoints, namely, $x \in [x_{l_1}, x_{l_2}]$ and $y \in [y_{l_2}, y_{l_2}]$. First, we interpolate along x using the weight functions defined in (B.7)

$$\begin{aligned} p(x, y_{l_2}) &= \phi_{1,l_1}(x) q_{l_1,l_2} + \phi_{1,l_1+1}(x) q_{l_1+1,l_2}, \\ p(x, y_{l_2+1}) &= \phi_{1,l_1}(x) q_{l_1,l_2+1} + \phi_{1,l_1+1}(x) q_{l_1+1,l_2+1}. \end{aligned} \tag{B.33}$$

Then we interpolate along y

$$\begin{aligned} p(x,y) &= \phi_{2,l_2}(y) p(x, y_{l_2}) + \phi_{2,l_2+1}(y) p(x, y_{l_2+1}) \\ &= \phi_{2,l_2}(y)\, \phi_{1,l_1}(x)\, q_{l_1,l_2} \\ &\quad + \phi_{2,l_2}(y)\, \phi_{1,l_1+1}(x)\, q_{l_1+1,l_2} \\ &\quad + \phi_{2,l_2+1}(y)\, \phi_{1,l_1}(x)\, q_{l_1,l_2+1} \\ &\quad + \phi_{2,l_2+1}(y)\, \phi_{1,l_1+1}(x)\, q_{l_1+1,l_2+1} \\ &= \langle \boldsymbol{\phi}(x,y), \mathbf{q} \rangle. \end{aligned} \tag{B.34}$$

The matrices corresponding to the product weight functions $\phi_{1,i_1}(x)\, \phi_{2,i_2}(y)$ and the node values q_{i_1,i_2} are

$$\boldsymbol{\phi}(x,y) = \begin{pmatrix} 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & dots & \vdots & \vdots & dots & 0 \\ 0 & \cdots & \phi_{1,l_1}(x)\, \phi_{2,l_2}(y) & \phi_{1,l_1+1}(x)\, \phi_{2,l_2}(y) & \cdots & 0 \\ 0 & \cdots & \phi_{1,l_1}(x)\, \phi_{2,l_2+1}(y) & \phi_{1,l_1+1}(x)\, \phi_{2,l_2+1}(y) & \cdots & 0 \\ \vdots & dots & \vdots & \vdots & dots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 \end{pmatrix} \in \mathbb{R}^{N_1 \times N_2} \tag{B.35}$$

and

$$\mathbf{q} = \begin{pmatrix} 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & dots & \vdots & \vdots & dots & 0 \\ 0 & \cdots & q_{l_1,l_2} & q_{l_1,l_2+1} & \cdots & 0 \\ 0 & \cdots & q_{l_1+1,l_2} & q_{l_1+1,l_2+1} & \cdots & 0 \\ \vdots & dots & \vdots & \vdots & dots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 \end{pmatrix} \in \mathbb{R}^{N_1 \times N_2}, \tag{B.36}$$

respectively.

Substituting the weight functions from (B.7) to (15) leads to the same result as (B.34).

Appendix C. Equation of motion of the coupled oscillator

The equation describing the motion of two linear oscillator coupled is given by

$$\begin{aligned} \mathrm{d}x &= v_x \mathrm{d}t, \\ \mathrm{d}y &= v_y \mathrm{d}t, \\ \mathrm{d}v_x &= \left(-2\omega_1(\xi_1 + \xi_{12})v_x - 2\xi_{12}\omega_1 v_y - (\omega_1^2 + \omega_{12}^2)x + \omega_{12}^2 y\right)\mathrm{d}t, \\ \mathrm{d}v_y &= \left(-2\omega_2\xi_2(v_y - v_x) - \omega_2^2(y - x)\right)\mathrm{d}t + \sigma \mathrm{d}W(t). \end{aligned} \tag{C.1}$$

We substitute the parameters $\omega_1 = \omega_2 = \omega_{12} = \sigma = 1$ and $\xi_1 = \xi_2 = \xi_{12} = 0.1$ to obtain (34) for the numerical experiments.

Appendix D. Steady-state PDF of linear time-invariant stochastic systems

The theorem (8.2.12) in [3] states that the steady-state solution of the equation

$$\mathrm{d}\mathbf{x}(t) = \mathbf{A}\mathbf{x}(t)\mathrm{d}t + \mathbf{b}\mathrm{d}W(t), \tag{D.1}$$

where $\mathbf{A} \in \mathbb{R}^{d \times d}$ and $\mathbf{b} \in \mathbb{R}^d$, is a stationary Gaussian process on the condition that the eigenvalues of $\mathbf{A}$ have negative real values, i.e. $\lim_{t\to\infty} \mathbf{x}(t) \sim \mathscr{N}(\mathbf{0}, \mathbf{K})$, where

$$\mathbf{K} = \lim_{t\to\infty} \int_0^t e^{\mathbf{A}^\top (t-s)} \mathbf{b}\mathbf{b}^\top e^{\mathbf{A}^\top (t-s)} \mathrm{d}s. \tag{D.2}$$

In the case of (33) the coefficients of (D.1) are

$$\mathbf{x} = \begin{pmatrix} x \\ v \\ Z \end{pmatrix}, \mathbf{A} = \begin{pmatrix} 0 & 0 & 1 \\ -1 & -2\zeta & \sigma \\ 0 & 0 & -\mu \end{pmatrix}, \mathbf{b} = \begin{pmatrix} 0 \\ 0 \\ \sqrt{2\mu} \end{pmatrix}, \tag{D.3}$$

while in the case of (C.1) the coefficients of (D.1) are

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix},$$

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -(\omega_1^2 + \omega_{12}^2) & \omega_{12}^2 & -2\omega_1(\xi_1 + \xi_{12}) & -2\xi_{12}\omega_1 \\ \omega_2^2 & -\omega_2^2 & 2\omega_2\xi_2 & -2\omega_2\xi_2 \end{pmatrix},$$

$$\mathbf{b} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \sigma \end{pmatrix}. \tag{D.4}$$

References

[1] Abramowitz M, Stegun IA. Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, Dover, New York, 9th dover printing, 10th gpo printing ed.; 1964.

[2] Alevras P, Yurchenko D. GPU computing for accelerating the numerical path integration approach 2016;171:46–53.

[3] Arnold L. Stochastic Differential Equations: Theory and Applications. Munich: R. Oldenbourg Verlag; 1973.

[4] Bergman LA, Heinrich JC. Solution of the pontriagin-vitt equation for the moments of time to first passage of the randomly accelerated particle by the finite element method. Int J Numer Meth Eng 1980;15:1408–12.

[5] Chai W, Dostal L, Naess A, Leira BJ. A comparative study of the stochastic averaging method and the path integration method for nonlinear ship roll motion in random beam seas, 2017;23 : 854–865.

[6] Chen L, Jakobsen ER, Naess A. On numerical density approximations of solutions of SDEs with unbounded coefficients. Adv Comput Math 2017;44:693–721.

[7] Friedland S, Lim L-H. Nuclear norm of higher-order tensors. Math Comput 2017;87:1255–81.

[8] Gaidai O, Dou P, Naess A, Dimentberg M, Cheng Y, Ye R. Nonlinear 6d response statistics of a rotating shaft subjected to colored noise by path integration on GPU. Int J Non-Linear Mech 2019;111:142–8.
[9] Gaidai O, Naess A, Dimentberg M. Response statistics of rotating shaft with non-linear elastic restoring forces by path integration. J Sound Vib 2017;400:113–21.
[10] Gaidai O, Yurchenko D, Ye R, Xu X, Wang J. Offshore crane non-linear stochastic response: novel design and extreme response by a path integration. Ships Offshore Struct 2021:1–7.
[11] Gel'fand IM, Shilov G. Generalized functions, 1–6, Academic Press; 1966–1968.
[12] P.E. Kloeden, E. Platen, and H. Schurz, Numerical Solution of SDE Through Computer Experiments, Universitext, Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.
[13] I.A. Kougioumtzoglou, A.D. Matteo, P.D. Spanos, A. Pirrotta, and M.D. Paola, An efficient wiener path integral technique formulation for stochastic response determination of nonlinear MDOF systems, 82 (2015).
[14] I.A. Kougioumtzoglou and P.D. Spanos, Nonstationary stochastic response determination of nonlinear systems: A wiener path integral formalism, 140 (2014), p. 04014064.
[15] Li C, Xu W, Yue X. Stochastic response of a vibro-impact system by path integration based on generalized cell mapping method. Int J Bifur Chaos 2014;24:1450129.
[16] Mo E. Path integration automatic programmer.
[17] E. Mo and A. Naess, Efficient path integration by FFT, in Applications of Statistics and Probability in Civil Engineering: Proceedings of the 10th International Conference, J. Kanda, T. Takada, and H. Furuta, eds., Tokyo, Japan, July 2007, CRC Press.
[18] E. Mo and A. Naess, Nonsmooth dynamics by path integration: An example of stochastic and chaotic response of a meshing gear pair, 4 (2009).
[19] Naess A, Johnsen J. Response statistics of nonlinear, compliant offshore structures by the path integral solution method. Probab Eng Mech 1993;8:91–106.
[20] Naess A, Moe V. Efficient path integration methods for nonlinear dynamic systems 2000;15:221–31.
[21] Paola MD, Alotta G. Path integral methods for the probabilistic analysis of nonlinear systems under a white-noise process. ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg 2020;6.
[22] Peskov N. Finite element solution of the fokker–planck equation for single domain particles. Phys B 2020;599:412535.
[23] I. Petromichelakis and I.A. Kougioumtzoglou, Addressing the curse of dimensionality in stochastic dynamics: a wiener path integral variational formulation with free boundaries, 476 (2020), p. 20200385.
[24] Petromichelakis I, Psaros AF, Kougioumtzoglou IA. Stochastic response determination of nonlinear structural systems with singular diffusion matrices: A wiener path integral variational formulation with constraints. Probab Eng Mech 2020;60:103044.
[25] , Stochastic response analysis and reliability-based design optimization of nonlinear electromechanical energy harvesters with fractional derivative elements, ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg, 7; 2021.
[26] Psaros AF, Kougioumtzoglou IA, Petromichelakis I. Sparse representations and compressive sampling for enhancing the computational efficiency of the wiener path integral technique 2018;111: 87–101.
[27] Roberts J. First-passage time for randomly excited non-linear oscillators. J Sound Vib 1986;109:33–50.
[28] Süli E, Mayers D. An Introduction to Numerical Analysis. Cambridge: Cambridge University Press; 2003.
[29] Trefethen LN. Approximation theory and approximation practice. Philadelphia: SIAM, Society for Industrial and Applied Mathematics; 2020.
[30] Yue X, Xu W. Stochastic bifurcation of an asymmetric single-well potential duffing oscillator under bounded noise excitation. Int J Bifur Chaos 2010;20:3359–71.