Long Time Propagation of Stochasticity by Dynamical Polynomial Chaos Expansions

Hasan Cagan Ozen

Submitted in partial fulfillment of the

requirements for the degree

of Doctor of Philosophy

in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2017

ABSTRACT

Long Time Propagation of Stochasticity by Dynamical Polynomial Chaos Expansions

Hasan Cagan Ozen

Stochastic differential equations (SDEs) and stochastic partial differential equations (SPDEs) play an important role in many areas of engineering and applied sciences such as atmospheric sciences, mechanical and aerospace engineering, geosciences, and finance. Equilibrium statistics and long-time solutions of these equations are pertinent to many applications. Typically, these models contain several uncertain parameters which need to be propagated in order to facilitate uncertainty quantification and prediction. Correspondingly, in this thesis, we propose a generalization of the Polynomial Chaos (PC) framework for long-time solutions of SDEs and SPDEs driven by Brownian motion forcing.

Polynomial chaos expansions (PCEs) allow us to propagate uncertainties in the coefficients of these equations to the statistics of their solutions. Their main advantages are: (i) they replace stochastic equations by systems of deterministic equations; and (ii) they provide fast convergence. Their main challenge is that the computational cost becomes prohibitive when the dimension of the parameters modeling the stochasticity is even moderately large. In particular, for equations with Brownian motion forcing, the long-time simulation by PC-based methods is notoriously difficult as the dimension of stochastic variables increases with time.

With the goal in mind to deliver computationally efficient numerical algorithms for stochastic equations in the long time, our main strategy is to leverage the intrinsic sparsity in the dynamics by identifying the influential random parameters and construct spectral approximations to the solutions in terms of those relevant variables. Once this strategy is employed dynamically in time, using online constructions, approximations can retain their sparsity and accuracy; even for long times. To this end, exploiting Markov property of

Brownian motion, we present a restart procedure that allows PCEs to expand the solutions at future times in terms of orthogonal polynomials of the measure describing the solution at a given time and the future Brownian motion. In case of SPDEs, the Karhunen–Loeve expansion (KLE) is applied at each restart to select the influential variables and keep the dimensionality minimal. Using frequent restarts and low degree polynomials, the algorithms are able to capture long-time solutions accurately. We will also introduce, using the same principles, a similar algorithm based on a stochastic collocation method for the solutions of SDEs.

We apply the methods to the numerical simulation of linear and nonlinear SDEs, and stochastic Burgers and Navier–Stokes equations with white noise forcing. Our methods also allow us to incorporate time-independent random coefficients such as a random viscosity. We propose several numerical simulations, and show that the algorithms compare favorably with standard Monte Carlo methods in terms of accuracy and computational times. To demonstrate the efficiency of the algorithms for long-time simulations, we compute invariant measures of the solutions when they exist.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First and foremost I offer my most sincere gratitude to my thesis supervisor, Professor Guillaume Bal, for his support and guidance throughout my research work. I am deeply grateful for all the advice, critical comments, and insightful discussions. Without his clear mathematical intuition, this work would not have been achieved.

I would like to thank the thesis committee members, Professor Marc Spiegelman, Professor Qiang Du, Professor Kyle Mandli, and Professor Georg Stadler, for taking time to review my thesis.

I am also grateful to Professor Fatih Ecevit for his support both before and during my Ph.D. study. His encouragement motivated me to pursue my Ph.D. and his passion for mathematics has always inspired me.

I wish to thank Kevin Carlberg, Khachik Sargsyan, and Mohammad Khalil for giving me the opportunity to work with them during my internship at Sandia National Laboratories.

My thanks to my friends, Alex Watson, Chenxi Guo, Roshan Sharma, Owen Evans, Hande Öztürk, Olgun Adak, and Jak Akdemir, with whom I have shared many meals, drinks, music, and fruitful discussions throughout my study.

I would like to thank to my beloved parents Birol Özen and Emine Özen for their endless support and having confidence in me.

Last but not least, my lovely wife, Filiz Carus Özen who has supported me for many long arduous years, deserves a special thanks. Without her company and support, I could not have survived this journey. This thesis is dedicated to her and my parents.

Chapter 1

Introduction

1.1 Overview

The modeling and numerical simulation of complex real-world problems requires addressing several sources of uncertainties such as model discrepancies, parametric input uncertainties, measurement errors and uncertainties, and numerical errors. Accordingly, the following two questions are usually posed in uncertainty quantification (UQ) framework: How accurate are the models representing the true physics? What are the effects of the uncertainties on the quantity of interest? UQ aims to answer these questions in order to achieve accurate predictive simulations for applications.

Many physical and engineering phenomena can be modeled using ordinary differential equations (ODEs) and partial differential equations (PDEs). In the presence of uncertainties, these models incorporate stochastic processes and/or random variables to represent the uncertain sources. The source of uncertainty typically includes uncertain physical parameters, uncertain initial and boundary conditions, and random forcing terms. Stochastic variants of ODEs and PDEs are called stochastic differential equations (SDEs) and stochastic partial differential equations (SPDEs), respectively.

Typical UQ work flow is demonstrated in Figure 1.1 in a Bayesian framework. Prior knowledge of the uncertain parameters is updated through Bayes likelihood using experiment data and as a result, a posterior is obtained. This is the model calibration step. Once the model is calibrated and probability distributions of the input variables are characterized, the input variables $\boldsymbol{\xi}$ are propagated through the model, which in turn yields the quantity of interest $u(\boldsymbol{\xi})$ parametrized by the input. This step requires simulation of a possibly high dimensional model $\mathcal{M}$, which represents either a SDE or a SPDE here. The forward propagation involves computation of statistical properties of the response u such as realizations, probability density functions, prediction intervals, and moments. Statistical knowledge of the response is then utilized in predictions. If necessary, a post-processing step involving a sensitivity analysis can be carried out to characterize the impact of the input on the output variability and re-calibrate the model for ensuing uncertainty propagation [110].

In this thesis, we are primarily interested in the forward problem and developing efficient numerical algorithms to be used in propagating uncertainties in the coefficients of SDEs and SPDEs. We will focus on stochastic evolution equations which describe the evolution of an uncertain dynamics in time depending on the input stochasticity. The type of stochasticity that we propagate includes random initial conditions, non-forcing physical parameters, and high-dimensional, time-dependent stochastic forcing terms.

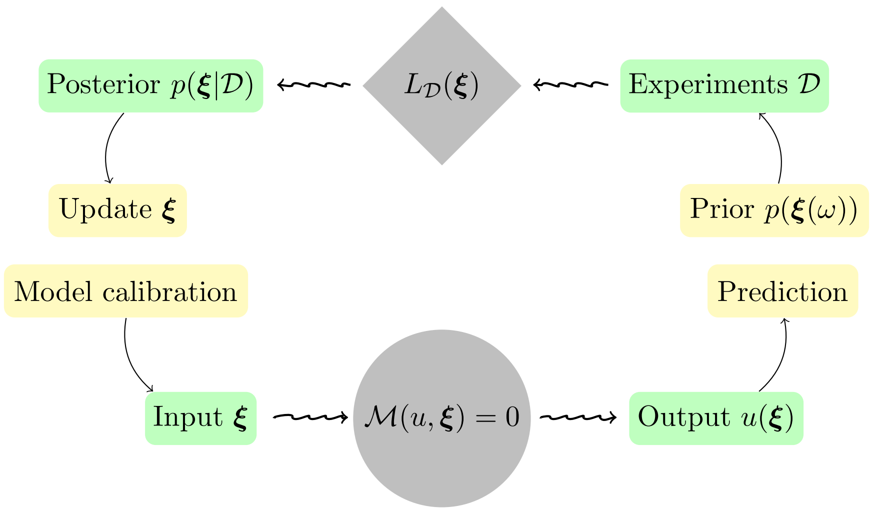

Posterior $p(\boldsymbol{\xi}|\mathcal{D})$
$L_{\mathcal{D}}(\boldsymbol{\xi})$
Experiments $\mathcal{D}$
Update $\boldsymbol{\xi}$
Prior $p(\boldsymbol{\xi}(\omega))$
Model calibration
Prediction
Input $\boldsymbol{\xi}$
$\mathcal{M}(u,\boldsymbol{\xi})=0$
Output $u(\boldsymbol{\xi})$

Figure 1.1: Forward and Inverse UQ.

1.2 Existing Probabilistic Methods

In the following, we discuss probabilistic methods to solve time-dependent stochastic evolution equations by focusing on two popular methods: Monte Carlo (MC) and Polynomial

Chaos (PC).

There are several techniques designed for solving stochastic evolution equations and propagating statistical information of the solution in time. For instance, the method of moments aims to derive a system of coupled differential equations for moments of the solution. Each equation describes the evolution of a certain moment. Typically, lower order moment equations depend on higher order moments yielding infinite set of equations. Moment-closure techniques then "close" these equations by truncating the moment equations into a low-dimensional system based upon physical considerations or sometimes heuristics. In the simplest settings, underlying distributions are approximated by Gaussian distributions and only second order moments of the solution are propagated. However, for nonlinear systems, it is difficult to find good closures and therefore, low-dimensional approximations to the moment equations might yield considerable errors [79; 16; 65].

Fokker-Planck equation-based methods offer a way to propagate the probability density function (pdf) of the random solution. For Itô diffusions, under appropriate assumptions, one can derive a PDE (called Fokker-Planck or Kolmogorov forward equation), which describes the evolution of the transition probabilities of the Markov semigroup. Numerical approximation requires solving a PDE of spatial dimension the same as the dimension of the SDE system. Thus, for high-dimensional SDEs, pdf-based techniques become prohibitively expensive. For SPDEs driven by white noise, Fokker-Planck equation becomes an infinite-dimensional system and has been mostly a theoretical tool in stochastic analysis [23]; see also [106; 27] for some recent numerical approximations. Although Fokker-Planck equations will not be the main focus of this thesis, we will utilize them to compare the accuracy and convergence of different methods.

Undoubtedly, Monte Carlo method has been one of the most popular methods to simulate stochastic equations. The popularity is due to its robustness and easy-to-implement nature. For SDEs and SPDEs, it is usually of interest to compute expectations (integrals with respect to a probability distribution) of functionals of the solutions. MC method relies on the law of large numbers and estimates these expectations by considering realizations of the solution. For each realization, one can use existing deterministic numerical methods to easily estimate expectations.

The accuracy of MC methods depends on the sample size and is not dependent on the dimensionality of the stochastic system. Although MC methods are applicable to high-dimensional systems, their efficiency for complicated systems is limited by their slow convergence. The typical convergence rate of MC methods is $O(1/\sqrt{M_{samp}})$, where M_{samp} is the sample size of the realizations of the source of randomness. The convergence can be accelerated by quasi-Monte Carlo methods, which can push the convergence rate to almost $O(1/M_{samp})$. There are also variance reduction techniques such as antithetic variables, control variates, importance sampling, and stratified sampling, which aim to reduce the sample-size-independent constant in the error to obtain a better precision. These acceleration techniques usually assume that the distribution of the quantity of interest is known.

For stochastic evolution equations, MC methods have to be coupled with time-integration methods. Due to the nature of stochastic integrals, straightforward application of existing time-integration methods usually yields low order convergence in terms of the time-step compared to the convergence behavior of deterministic methods. With special care of random forcing terms, higher order convergence can be achieved in the strong (pathwise) and weak (distributional) senses, though the numerical implementation gets complicated [64; 61]. The convergence rate of the law of large numbers still applies and limits the efficiency of standard MC methods for complicated dynamics. There has been a growing interest in designing multilevel Monte Carlo methods, which use a hierarchy of grid discretization (similar to multigrid methods) and optimize the number of samples at each level to accelerate the convergence. These methods offer substantial speed-ups compared to standard MC methods provided that the variance of multilevel corrections decay sufficiently fast with the refinement [43]. However, divergence of these methods has been noted for SDEs with nonlinear coefficients if an inappropriate time-stepping method is incorporated [59]. Thus, a judicious design of time-integration methods is needed; see also [9] and references therein.

Polynomial chaos is a popular spectral method that is used to propagate uncertainties in the coefficients of differential equations to the statistics of their solutions. The method discretizes the random space using global spectral basis of polynomials. It originated from the works of Wiener [121], and Cameron and Martin [17] on the decomposition of functionals of Brownian motion in a basis of Hermite polynomials of Gaussian random variables. Applications of such a framework to random flows and turbulence theory are examined in [82; 92; 21]. More recently, works of Ghanem et al. [41; 42] combined the standard Hermite PC method with the Karhunen–Loeve expansion (KLE) [62; 72] to study structural mechanics problems. They used finite element bases in spatial variables and a spectral basis obtained by the KLE discretization of the stochastic input in the random space. A deterministic system of equations for the PC coefficients is obtained by the Galerkin projection in the Hilbert space. The method is called stochastic finite elements and it revived the interest in spectral methods for stochastic equations. Xiu and Karniadakis [125; 128; 122] next extended the Hermite PC to a set of non-Gaussian random parameters, and studied numerical convergence and efficiency in flow simulations. Their method is called *generalized polynomial chaos* (gPC). Rather than classical Hermite polynomials, gPC uses tailored orthogonal polynomials associated to the distribution of the uncertainty to provide optimal representations. The works [73; 83; 84] laid down theoretical foundations of the Hermite PC applied to SPDEs driven by Brownian motion. Numerical approximations to fluid dynamics equations of this framework are considered in [57; 75]. For other extensions and discussions of the PC framework, we refer to [67; 112; 7; 26; 33; 81; 119; 118; 85; 66; 37; 30; 4; 16].

The main advantage of the PC method is that it allows us to propagate stochasticity by providing expansions of quantities of interest in terms of appropriate uncertainties while in effect replacing the stochastic equations by a system of deterministic equations. Once these deterministic equations are solved, statistical properties of the solution including higher order moments can be readily inferred from the coefficients of the expansion, which facilitates uncertainty quantification. In contrast to MC methods, the PC method leverages the regularity of the solution in the input stochasticity and offers fast convergence when it is coupled with Galerkin projection. For instance, exponential convergence with respect to the degree of polynomials has been usually observed in simulations [41; 125; 128; 119]. All these advantages have made the PC method a viable alternative to MC methods in engineering applications.

1.3 Motivation and Objective

Polynomial chaos expansions (PCEs) provide an explicit expression of quantities of interests as functionals of the underlying uncertain parameters and in some situations allow us to perform uncertainty propagation and quantification at a considerably lower computational cost than Monte Carlo methods [41; 125; 128; 57; 16; 93; 8; 94]. However, they suffer from the curse of dimensionality and thus typically work efficiently for systems involving low-dimensional uncertainties. The efficiency of the method is reduced because of the large number of terms that appear in the expansion. A related major drawback appears in the long-time integration of evolution equations. The presence of a temporal random forcing is a serious challenge to the PC method as the number of stochastic variables increases with time, which hinders the capability of the PC method for long-time computations [57; 16; 93; 94]. Moreover, standard PCE utilizes orthogonal polynomials of the initial distribution, and as time evolves, the dynamics deviate from the initial data substantially, e.g., due to nonlinearities, and the solution may become poorly represented in the initial basis.

In this thesis, focusing on SDEs and SPDEs driven by additive Brownian motion forcing, our main objective is to offer efficient numerical algorithms which alleviate the two major drawbacks of the PC method: (i) curse of dimensionality; and (ii) loss of optimality in long-time integration. For Markovian systems, one can leverage the intrinsic sparsity of the dynamics in the sense that a sparse representation of the solution at a future time can be obtained in terms of the solution variables (or a compressed version of the solution variables) at the current time and the random variables that represent the random forcing for the future. In other words, one can "forget" about the past and as a consequence keep the dimension of the uncertainty fixed and independent of time. Such a forgetting strategy requires construction of dynamical algorithms which adapt to evolving, arbitrary probability measures of the solution. It also involves selecting the most influential variables automatically on-the-fly and constructing approximations based on them. This time-dependent adaptation is in some sense similar to and extends the gPC method, which constructs optimal representations by taking into account the distribution of the uncertainty. To this end, the algorithms we propose here carry in time essential information, e.g., orthogonal polynomials and in some cases quadrature nodes, to characterize the pertaining measures.

1.4 Outline of the Thesis

Chapter 2 reviews the theory of the Karhunen-Loeve expansion and polynomial chaos expansions, discusses non-intrusive and intrusive implementations focusing on sparse grid collocation method, and establishes the notations used in the subsequent chapters.

In Chapter 3, we propose a PC-based method, called *Dynamical generalized Polynomial Chaos* (DgPC), for long-time evolution of SDEs with Brownian motion forcing. The method constructs evolving chaos expansions based on polynomials of projections of the time-dependent solution and the random forcing through a restarting mechanism. More precisely, chaos expansions at each restart are constructed based on the knowledge of the moments of underlying distributions at a given time. In the setting of dynamics satisfying a Markov property, we crucially exploit this feature to introduce projections of the solution at prescribed time steps that allow us to forget the variables in the past and keep reasonably sparse random bases. Orthogonal bases that the method propagates in time are the optimal ones associated to the solution variables. Thus, chaos bases are adapted to the evolving dynamics with the following consequences: (i) DgPC retains its optimality for long times; and (ii) the curse of dimensionality is mitigated. Notably, we establish, with appropriate modifications, theoretical convergence analysis which sheds light on our numerical findings. Further, asymptotic analysis for computational complexity will also be discussed. Inspired by examples of [38; 16], we will apply our method to a nonlinear coupled system of stochastic differential equations and its variants.

Chapter 4 builds upon the method introduced in Chapter 3 for SDEs and extends it to SPDEs driven by white noise. While solutions to SPDEs are, in general, high-dimensional random fields, they may lend themselves in some cases to low-dimensional representations [41; 12; 60; 107; 57; 28; 105; 20]. Armed with this fact, we propose at each restart to use the KLE to compress the solution into a representation involving a finite number of random modes. In cases where the modeling equations contain non-forcing random inputs other than Brownian forcing, such as a random viscosity, the KLE is applied to the solution and the random parameters together so that the algorithm automatically selects the intrinsic variables, which have the largest influence on the solution. The KLE is a computationally expensive procedure as it requires solving a large eigenvalue problem. We offer different methods such as Krylov subspace methods and low-rank approximations to large covariance matrices to mitigate the computational cost. A few dominating random KLE modes are then chosen and incorporated into PCE to represent the future solution. The computation of orthogonal polynomials of multivariate distributions is based upon estimation of multivariate moments using a sampling procedure. To keep the number of terms in DgPC small, we make use of a sparse truncation technique for multi-indices [57; 75; 13]. We present both short- and long-time computations for a randomly forced 1D Burgers equation and a stochastic 2D stochastic Navier–Stokes (SNS) system. In some cases, we provide a purely PCE-based numerical verification of the convergence of the process to its invariant measure.

Chapter 5 goes back to the setting of SDEs and proposes a stochastic non-intrusive method based on sparse grid collocation (SGC) for long-time simulations. The method uses pre-determined sparse quadrature rules for the forcing term and constructs evolving set of sparse quadrature rules for the solution variables in time. In contrast to methods developed in the previous chapters, this method propagates deterministic samples of the distribution of the solution. We carry out a similar restart scheme to keep the dimension of the random variables for the forcing term, therefore also the number of quadrature points, independent of time. At each restart, a sparse quadrature rule for the solution variables is constructed from the knowledge of previous quadrature rules through an optimization procedure. In this way, the method allows us to accurately capture the long-time solutions using small degrees of freedom. We apply the algorithm to low-dimensional nonlinear SDEs and demonstrate its ability to reach accurate long-time simulations numerically.

Finally, Chapter 6 concludes with a summary of the thesis and offers several future work directions.

Chapter 2

Preliminaries

2.1 Karhunen–Loeve Expansion

Let $G \in \mathbb{R}^d$ be a compact spatial domain. Given a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω is a sample space equipped with the sigma-algebra $\mathcal{F}$ and the probability measure $\mathbb{P}$, we denote by $L^2(G \times \Omega)$ the Hilbert space of square integrable random fields on G, i.e. second order stochastic processes. For a random field $u \in L^2(G \times \Omega)$, we define the expectation

$$\bar{u}(x) := \mathbb{E}[u(x, \omega)] = \int_\Omega u(x, \omega)\mathbb{P}(d\omega),$$

and the covariance

$$\mathrm{Cov}_u(x, y) := \mathbb{E}[(u(x, \omega) - \bar{u}(x))\,(u(y, \omega) - \bar{u}(y))'], \quad x, y \in G, \tag{2.1}$$

where $'$ denotes the transpose. We denote by $\langle \cdot, \cdot \rangle_{L^2(G)}$ the spatial inner product on $L^2(G)$.

Associated to a continuous covariance, there is a linear integral operator, called covariance kernel, which is a self-adjoint, positive semidefinite Hilber-Schmidt operator on $L^2(G)$ [58]. Then, by Mercer's theorem, the covariance admits the following representation

$$\mathrm{Cov}_u(x, y) = \sum_{l=1}^{\infty} \lambda_l\, \phi_l(x)\, \phi_l(y),$$

where λ_l's and ϕ_l's are the eigenvalues and the eigenfunctions of the associated covariance kernel, i.e.

$$\langle \mathrm{Cov}_u(x, \cdot)\,, \phi_l \rangle_{L^2(G)} = \lambda_l\, \phi_l(x), \quad x \in G.$$

The eigenvalues are non-negative and the eigenfunctions constitute a complete orthonormal set in $L^2(G)$; $\langle \phi_l, \phi_k \rangle_{L^2(G)} = \delta_{lk}$, where δ_{lk} denotes the Kronecker delta function.

Any random field $u \in L^2(G \times \Omega)$ with a continuous covariance admits the following spectral expansion, which is called the Karhunen–Loeve expansion (KLE) [41]:

$$u(x, \omega) = \bar{u}(x) + \sum_{l=1}^{\infty} \sqrt{\lambda_l}\, \eta_l(\omega)\, \phi_l(x), \tag{2.2}$$

where random variables η_l are mean zero and given by the projection onto eigenfunctions

$$\eta_l(\omega) = \lambda_l^{-1/2} \langle [u(\cdot, \omega) - \bar{u}]\, , \phi_l \rangle_{L^2(G)}. \tag{2.3}$$

The random modes also satisfy orthogonality, i.e. $\mathbb{E}[\eta_l\, \eta_k] = \delta_{lk}$; see also [62; 72].

The major feature of the KLE is that the truncation after a finite number, denoted by D hereafter, of terms is optimal in L^2 sense, i.e. the error resulting from projecting the stochastic process onto any other orthogonal spatial set using (2.3) is always greater than the error of the truncation of the KLE (2.2). How D should be chosen obviously depends on the spectrum of the covariance kernel. When the process shows a high degree of correlation, then typically D may be chosen relatively small due to the rapid decay of the eigenvalues [41; 66; 107]. This property makes the KLE a useful dimensionality reduction technique in many applications and will play a crucial role in Chapter 4 to compress the dimensionality of solutions of SPDEs.

The L^2-norm of the error of D-term truncation of the KLE (2.2) goes to zero as the degrees of freedom goes to infinity:

$$\sum_{l>D}^{\infty} \lambda_l \to 0, \quad D \to \infty.$$

Then, the energy retained in the expansion can be defined as the ratio

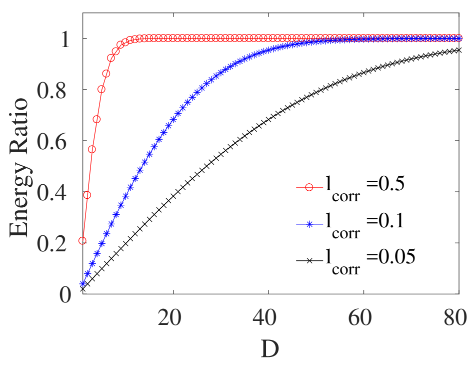
$$\frac{\sum_{l=1}^{D} \lambda_l}{\sum_{l=1}^{\infty} \lambda_l}, \tag{2.4}$$

which is an indication of how fast the eigenvalues decay.

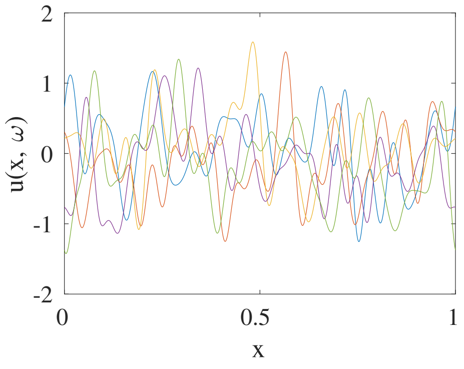
We consider a simple demonstration to show the rate of decay of the eigenvalues for a mean zero process (2.2) with the periodic exponential covariance function

$$\mathrm{Cov}_u(x, y) = \exp\left(-\frac{2}{l_{corr}^2} \sin^2(\pi(x - y)) \right), \quad x, y \in [0, 1],$$

where l_{corr} is the correlation length [102]; and see also Section 4.3.1. The random modes in the KLE are selected as uniformly distributed independent random variables on $[-1, 1]$. The energy ratio (2.4) for different values of the correlation length, $l_{corr} = 0.5, 0.1$, and 0.05, is depicted in Figure 2.1a. We observe that the smaller the correlation length, the more terms needed to retain the same energy level. Specifically, to capture 95% of the total energy, 9, 40, and 78 terms are needed in each scenario, respectively. Figure 2.1b shows 5 different realizations of the process $u(x, \omega)$ with the correlation length $l_{corr} = 0.1$. The realizations are obtained by sampling the random KLE modes using 40 terms. The periodicity of realizations is mandated by the form of the covariance function and the period is 1 in this case.



(a) Energy ratio vs D



(b) Realizations

Figure 2.1: Energy ratios for different values of the truncation parameter D and 5 realizations of the process.

2.2 Polynomial Chaos Expansions

Consider $L^2(\Omega, \mathcal{F}, \mathbb{P})$, the space of real-valued random variables with finite second order moments. Let $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots)$ be a countable collection of independent and identically distributed (i.i.d.) standard Gaussian random variables belonging to the probability space, and $\mathcal{F} = \sigma(\boldsymbol{\xi})$. Then, we define the Wick polynomials by the tensor product

$$T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) := \prod_{k=1}^{\infty} H_{\alpha_k}(\xi_k),$$

where $\boldsymbol{\alpha}$ belongs to set of multi-indices with a finite number of nonzero components

$$\mathcal{J} = \{\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots) \,|\, \alpha_k \in \mathbb{N}_0,\, |\boldsymbol{\alpha}| = \sum_{k=1}^{\infty} \alpha_k < \infty\},$$

and $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$. H_n is the nth order normalized one-dimensional Hermite polynomial given by the formula

$$H_n(x) = \frac{1}{\sqrt{n!}} (-1)^n \, e^{x^2/2} \, \frac{d^n}{dx^n} \, e^{-x^2/2}.$$

Note that the Wick polynomials are orthogonal to each other with respect to the measure induced by $\boldsymbol{\xi}$:

$$\mathbb{E}[T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) T_{\boldsymbol{\beta}}(\boldsymbol{\xi})] = \delta_{\boldsymbol{\alpha}\boldsymbol{\beta}}.$$

The Cameron and Martin theorem [17] establishes that the Wick polynomials form a complete orthonormal basis in $L^2(\Omega, \mathcal{F}, \mathbb{P})$. This means that any functional $u(\cdot, \boldsymbol{\xi}) \in L^2$, can be expanded as

$$u(\cdot, \boldsymbol{\xi}) = \sum_{\boldsymbol{\alpha} \in \mathcal{J}} u_{\boldsymbol{\alpha}}(\cdot) \, T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}), \quad u_{\boldsymbol{\alpha}}(\cdot) = \mathbb{E}[u(\cdot, \boldsymbol{\xi}) T_{\boldsymbol{\alpha}}(\boldsymbol{\xi})], \tag{2.5}$$

and the sum converges in L^2. Here $(\cdot)$ notation represents deterministic independent arguments; e.g. spatial and temporal variables. The spectral expansion (2.5) is called *polynomial chaos expansion* (PCE) [121; 41]. Here we emphasize that the expansion is convergent for general random variables provided second order moments exist and the measurability condition is satisfied. The expansion can be seen as a sum of a Gaussian approximation and a non-Gaussian part; the terms that satisfy $|\boldsymbol{\alpha}| \leq 1$ and $|\boldsymbol{\alpha}| > 1$, respectively. The decay rate of the coefficients $u_{\boldsymbol{\alpha}}$ depends on the smoothness of the solution in the random parameters and typically, low-order coefficients dominate high-order ones in most applications. In the subsequent chapters, we will also use the term Hermite PCE to emphasize that the expansion utilizes Gaussian random variables.

One of the notable features of PCE is that it separates the randomness in u such that the coefficients $u_{\boldsymbol{\alpha}}$ are deterministic and all statistical information is contained in the coefficients. In particular, the first two moments are given by

$$\mathbb{E}[u] = u_{\mathbf{0}} \quad \text{and} \quad \mathbb{E}[u^2] = \sum_{\boldsymbol{\alpha}\in\mathcal{J}} |u_{\boldsymbol{\alpha}}|^2.$$

Higher order moments may then be computed using the triple products of Wick polynomials:

$$\mathbb{E}[T_{\boldsymbol{\alpha}}(\boldsymbol{\xi})T_{\boldsymbol{\beta}}(\boldsymbol{\xi})T_{\boldsymbol{\gamma}}(\boldsymbol{\xi})] = \prod_{k=1}^{\infty} \mathbb{E}[H_{\alpha_k}(\xi_k)H_{\beta_k}(\xi_k)H_{\gamma_k}(\xi_k)] = \prod_{k=1}^{\infty} \frac{\sqrt{\alpha_k!\,\beta_k!\,\gamma_k!}}{(m-\alpha_k)!\,(m-\beta_k)!\,(m-\gamma_k)!},$$

for even $m = (\alpha_k + \beta_k + \gamma_k)/2$ and $m \geq \alpha_k, \beta_k, \gamma_k$; otherwise the product is zero [122; 123]. For instance, PCE coefficients for u^2 and u^3 are given by

$$(u^2)_{\boldsymbol{\alpha}} = \mathbb{E}[u^2\, T_{\boldsymbol{\alpha}}(\boldsymbol{\xi})] = \sum_{\boldsymbol{\beta}\in\mathcal{J}}\sum_{\boldsymbol{\gamma}\in\mathcal{J}} u_{\boldsymbol{\beta}}\, u_{\boldsymbol{\gamma}}\, \mathbb{E}[T_{\boldsymbol{\alpha}}(\boldsymbol{\xi})T_{\boldsymbol{\beta}}(\boldsymbol{\xi})T_{\boldsymbol{\gamma}}(\boldsymbol{\xi})],$$
$$(u^3)_{\boldsymbol{\alpha}} = \mathbb{E}[u^3\, T_{\boldsymbol{\alpha}}(\boldsymbol{\xi})] = \sum_{\boldsymbol{\beta}\in\mathcal{J}}\sum_{\boldsymbol{\gamma}\in\mathcal{J}} (u^2)_{\boldsymbol{\beta}}\, u_{\boldsymbol{\gamma}}\, \mathbb{E}[T_{\boldsymbol{\alpha}}(\boldsymbol{\xi})T_{\boldsymbol{\beta}}(\boldsymbol{\xi})T_{\boldsymbol{\gamma}}(\boldsymbol{\xi})].$$

Moments can also be computed by repeated application of the Hermite product formula [84; 75]:

$$u^2 = \sum_{\boldsymbol{\alpha}\in\mathcal{J}}\sum_{\boldsymbol{\gamma}\in\mathcal{J}}\sum_{0\leq\boldsymbol{\beta}\leq\boldsymbol{\alpha}} C(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma})\, u_{\boldsymbol{\alpha}-\boldsymbol{\beta}+\boldsymbol{\gamma}}\, u_{\boldsymbol{\beta}+\boldsymbol{\gamma}}\, T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}), \tag{2.6}$$

where

$$C(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) = \left[\binom{\boldsymbol{\alpha}}{\boldsymbol{\beta}}\binom{\boldsymbol{\beta}+\boldsymbol{\gamma}}{\boldsymbol{\gamma}}\binom{\boldsymbol{\alpha}-\boldsymbol{\beta}+\boldsymbol{\gamma}}{\boldsymbol{\gamma}}\right]^{1/2}.$$

In numerical computations, the doubly infinite expansion (2.5) is truncated so that it becomes a finite expansion

$$u \approx u_{K,N}(\cdot,\xi_1,\ldots,\xi_K) := \sum_{|\boldsymbol{\alpha}|\leq N} u_{\boldsymbol{\alpha}}(\cdot) \prod_{k=1}^{K} H_{\alpha_k}(\xi_k), \tag{2.7}$$

where we used polynomials up to degree N in the variables $(\xi_1, \xi_2, \ldots, \xi_K)$. We define the corresponding multi-index set

$$\mathcal{J}_{K,N} := \{\boldsymbol{\alpha} = (\alpha_1,\ldots,\alpha_K) \,|\, \alpha_k \in \mathbb{N}_0,\, |\boldsymbol{\alpha}| \leq N\}. \tag{2.8}$$

The equation (2.7) is nothing but an orthogonal projection of u onto polynomial basis and leads to

$$M := \binom{K+N}{K}$$

terms in the approximation. Throughout the thesis, we use graded lexicographic ordering for multi-indices [123]; unless otherwise stated. An example of this ordering with two variables is given in Table 2.1.

$\lvert\boldsymbol{\alpha}\rvert$	$\boldsymbol{\alpha}$	$T_{\boldsymbol{\alpha}}(\boldsymbol{\xi})$
0	(0,0)	1
1	(1,0)	ξ_1
	(0,1)	ξ_2
2	(2,0)	$\frac{1}{\sqrt{2}}(\xi_1^2-1)$
	(1,1)	$\xi_1\xi_2$
	(0,2)	$\frac{1}{\sqrt{2}}(\xi_2^2-1)$
3	(3,0)	$\frac{1}{\sqrt{6}}(\xi_1^3-3\xi_1)$
	(2,1)	$\frac{1}{\sqrt{2}}(\xi_1^2-1)\xi_2$
	(1,2)	$\frac{1}{\sqrt{2}}\xi_1(\xi_2^2-1)$
	(0,3)	$\frac{1}{\sqrt{6}}(\xi_2^3-3\xi_2)$
$\vdots$	$\vdots$	$\vdots$

Table 2.1: An ordering for the multi-index set and Hermite basis.

In the context of white noise–driven SDEs, the solution $u(t,\omega)$, $t \in [0,T]$ and $\omega \in \Omega$, is a functional of paths of Brownian motion $W(t,\omega)$

$$u = u(t; \{W(\tau), 0 \le \tau \le t\}).$$

For the sake of brevity, we sometimes omit ω-dependence and write $W(t)$ for $W(t,\omega)$. Throughout the thesis, all stochastic integrals are considered in the Ito sense [90].

The random variables ξ_k can be obtained by the projection

$$\xi_k(\omega) = \int_0^T m_k(t)\, dW(t,\omega),$$

where $m_k(t)$ is a deterministic, complete orthonormal system in $L^2[0,T]$. Then, $\boldsymbol{\xi} = (\xi_k)_k$ is comprised of i.i.d. standard Gaussian random variables and the expansion

$$\sum_{k=1}^{\infty} \xi_k \int_0^t m_k(\tau) d\tau, \tag{2.9}$$

converges in L^2 to Brownian motion for all $t \leq T$, i.e.

$$\mathbb{E}\left[W(t) - \sum_{k=1}^{K} \xi_k \int_0^t m_k(\tau) d\tau\right]^2 \to 0, \quad K \to \infty, \tag{2.10}$$

[10, Chapter 6]. Here, Brownian motion $\{W(t),\, 0 \leq t \leq T\}$ is projected onto $L^2[0,T]$ for a fixed time $T > 0$ so that the corresponding PCE basis $T_{\boldsymbol{\alpha}}(\boldsymbol{\xi})$ depends implicitly on T. In this thesis, Brownian motion and its "derivative", white noise, are interpreted through the infinite linear combination (2.9).

Typical examples of m_k's are trigonometric functions and wavelets [15; 78; 57; 10; 31]. For instance, if we choose the orthonormal basis

$$m_k(t) = \sqrt{\frac{2}{T}} \cos\left(\frac{(k-1/2)\pi t}{T}\right), \quad k = 1, 2, \ldots$$

then the expansion (2.9) is the KLE of Brownian motion on $[0,T]$:

$$W(t) = \sum_{k=1}^{\infty} \frac{\sqrt{2T}}{(k-1/2)\pi} \sin\left(\frac{(k-1/2)\pi t}{T}\right) \xi_k.$$

Another choice of orthonormal set of functions is

$$m_1(t) = \frac{1}{\sqrt{T}},\ m_k(t) = \sqrt{\frac{2}{T}} \cos\left(\frac{(k-1)\pi t}{T}\right), \quad k = 2, 3, \ldots.$$

Under this choice, the expansion becomes

$$W(t) = \frac{t}{\sqrt{T}} \xi_1 + \sum_{k=2}^{\infty} \frac{\sqrt{2T}}{(k-1)\pi} \sin\left(\frac{(k-1)\pi t}{T}\right) \xi_k, \tag{2.11}$$

and the K-term truncation L^2-error (2.10) can be derived as

$$\begin{aligned}\sum_{k>K} \left(\int_0^t m_k(\tau) d\tau\right)^2 &= \sum_{k=K}^{\infty} \frac{2T}{k^2\pi^2} \sin^2\left(\frac{k\pi t}{T}\right), \\ &= \begin{cases} O(TK^{-1}), & \text{if } 0 < t < T, \\ 0, & \text{if } t = T, \end{cases}\end{aligned}$$

see also [75]. It is interesting to note that both representations violate Markov property of Brownian motion for times $0 < t < T$ as they use global bases on $[0, T]$. It is also possible to filter $T_{\boldsymbol{\alpha}}(\boldsymbol{\xi})$ with respect to the σ-algebra generated by Brownian motion up to time t. Resulting basis will not violate Markov property. However, since it is not orthogonal, this basis will not be considered here; see [73; 84; 75].

2.3 Estimation of Expectations

Approximation of an output $u(\boldsymbol{\xi}) \in \mathbb{R}^d$ of a stochastic model parametrized by a stochastic input variables $\boldsymbol{\xi} \in \mathbb{R}^K$ typically involves computation of expectations of functionals of u. For instance, as we saw in the preceding section, PCE projects the output u onto orthogonal polynomial basis $T_{\boldsymbol{\alpha}}(\boldsymbol{\xi})$ and the coefficients of the expansion are given in the form of expectations with respect to the probability measure of $\boldsymbol{\xi}$; see (2.5). The dimension K of the stochastic input variables $\boldsymbol{\xi}$ is usually large in most applications, which in turn requires efficient estimations of high-dimensional integrals.

There are mainly two types of methods available for computing expectations: intrusive and non-intrusive methods. Intrusive methods, also called Galerkin methods, are based on the orthogonality between the projected solution and the residual to form deterministic governing equations for expansion coefficients. On the other hand, non-intrusive methods rely on realizations of the input variables $\boldsymbol{\xi}$ to estimate averages. Random sampling methods, for instance, use pseudo-random number sequences in the estimation. An alternative way is to compute a sequence of deterministic sampling points by considering quadrature rules associated to the measure of input variables. The latter is usually called stochastic collocation method, or sometimes non-intrusive spectral projection [124; 66].

We defer detailed discussions of intrusive Galerkin methods to the ensuing chapters and focus on quadrature-based collocation methods in the next section.

2.3.1 Sparse Grid Collocation Method

Assuming the probability distribution of the input variables $\boldsymbol{\xi}$ is known, e.g. given in the Askey family [125], and the components are independent, one can construct multi-dimensional quadrature rules by employing tensorization of one-dimensional quadrature rules. For the sake of brevity, we consider evaluation of K-dimensional integrals of the output $u(\boldsymbol{\xi})$ with respect to a Gaussian probability measure $p_{\boldsymbol{\xi}}(\boldsymbol{\xi}) \propto \exp(-||\boldsymbol{\xi}||_2/2)$.

In one dimension, we define the Gauss-Hermite quadrature rules I_Q consisting of the weights and nodes $\{w^q, \xi^q\}_{q=1}^{Q}$, $Q \in \mathbb{N}$ and $\xi \in \mathbb{R}$:

$$I_Q(u)(\xi) := \sum_{q=1}^{Q} w^q \, u(\xi^q),$$

where ξ^q are the roots of the Hermite polynomials H_Q and the weights $w^q = 1/(Q^2(H_{Q-1}(\xi^q))^2)$. It is known that I_Q is exact if u is a polynomial of degree less than or equal to $2Q-1$; [25].

For the multi-dimensional case, the integral $\mathbb{E}[u(\boldsymbol{\xi})]$ can be approximated by the tensor product formula

$$\mathbb{E}[u(\boldsymbol{\xi})] \approx I_Q^{\otimes K} := \sum_{\alpha_1=1}^{Q} \dots \sum_{\alpha_K=1}^{Q} u(\xi_1^{\alpha_1}, \dots, \xi_K^{\alpha_K}) \, w_1^{\alpha_1} \dots w_K^{\alpha_K}.$$

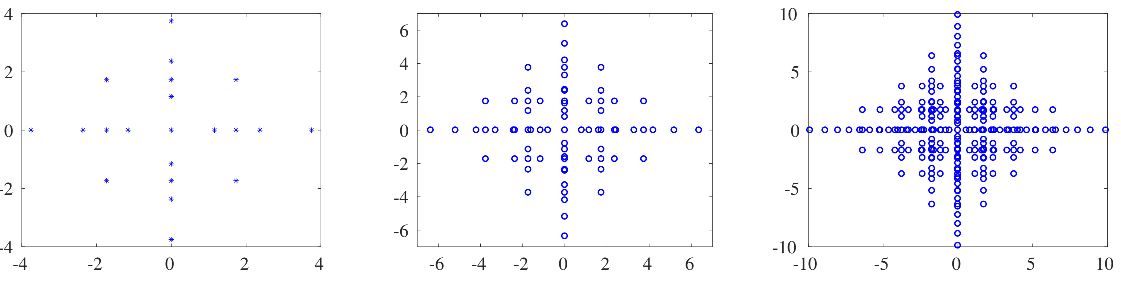
Here we use the multi-index notation: $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_K) \in \mathbb{N}^K$ with $|\boldsymbol{\alpha}| = \sum_{k=1}^{K} \alpha_k$. We denote by $Q_{\boldsymbol{\xi}}$ the total number of resulting quadrature nodes. Approximations based on this tensor product suffer from curse of dimensionality and computational costs scale exponentially with dimension K, i.e. $Q_{\boldsymbol{\xi}} = Q^K$.

If the dimension of the random variables is moderately high, a sparse quadrature rule first proposed by Smolyak, can be used to reduce the number of quadrature nodes while maintaining the accuracy [111]. Following [120; 40], we write the sparse grid approximation to the multi-dimensional integral with the level λ

$$\mathbb{E}[u] \approx \sum_{\lambda \le |\boldsymbol{\alpha}| \le \lambda+K-1} (-1)^{\lambda+K-|\boldsymbol{\alpha}|-1} \binom{K-1}{|\boldsymbol{\alpha}|-\lambda} (I_{\alpha_1} \otimes \dots \otimes I_{\alpha_K})(u), \tag{2.12}$$

where $\boldsymbol{\alpha} \geq \mathbf{1}$. This quadrature rule is exact for multivariate polynomials of total degree up to $2\lambda - 1$ and greatly reduces the number of evaluations compared to the tensor product rule above [40; 89]. In this work, we employ isotropic Smolyak sparse grid quadrature rules for Gaussian measures, meaning that the level λ is the same for each dimension. We also note that the weights of this sparse quadrature rule can be negative.

An illustration of the quadrature nodes in two dimensions with different levels λ is given in Figure 2.2. One-dimensional Gauss-Hermite rules are used to construct sparse quadrature rules.



(a) $\lambda = 2$ (b) $\lambda = 3$ (c) $\lambda = 4$

Figure 2.2: A demonstration of sparse quadrature nodes for different levels using one-dimensional Gauss-Hermite rule.

Chapter 3

Dynamical gPC for SDEs

3.1 Related Work and Motivation

Applicability of polynomial chaos expansions to differential equations with stochastic parameters have been shown in numerous works [41; 42; 57; 122; 125; 118; 33; 81]. Two major advantages of PCEs are: (i) they provide deterministic means to compute statistical information about the response of a random system; (ii) they take advantage of smoothness of the response parametrized by random parameters to achieve fast convergence. The efficiency of PCE applied to equations containing low-dimensional random parameters has been noted in the literature.

For moderate- to high-dimensional random spaces, the efficiency of the Hermite PCE typically deteriorates. As we noted before in (2.7), if an equation contains K random variables and a polynomial basis of total order N is used, then the total number of terms in the approximation becomes

$$\binom{K+N}{K}.$$

Thus, to maintain a level of accuracy, the number of terms in the expansion should scale exponentially with K. Then the computational cost increases rapidly with high dimensionality, which in turn decreases the efficiency of PCE. This is the "curse of dimensionality" in this context.

Another related major problem of PC is that expansions may converge slowly and even fail to converge for long time evolutions [78; 119; 118; 57; 16]. The optimality of the fixed, initial PC basis diminishes in time if the dynamics exhibit considerable deviations from the initial conditions. These problems led to numerous extensions of the Hermite PCE, which we now briefly discuss.

The paper [125] proposed a method called generalized polynomial chaos (gPC), in which the random variables ξ have specific, non-Gaussian, distributions in the Askey family [125; 122]. Although, the Hermite PCE (2.5) converges for any L^2 functional, the optimal convergence is usually achieved when the underlying uncertainty is nearly Gaussian [122; 125]. The main idea of [125] is then to represent the randomness in the system in a sparser way using a polynomial basis which is orthogonal with respect to the distribution of the input random parameters. Table 3.1 summarizes the optimal choices of orthogonal polynomials for different one-dimensional continuous probability distributions [122; 125].

distribution	density function	polynomials
Gaussian	$\frac{1}{\sqrt{2\pi}}e^{-\xi^2/2}$	Hermite
U(a,b)	$\frac{1}{b-a}$	Legendre
$\beta(a,b)$	$\frac{\xi^{a-1}(1-\xi)^{b-1}}{B(a,b)}$	Jacobi
$\Gamma(a,b)$	$\frac{b^a}{\Gamma(a)}\xi^{a-1}e^{-b\xi}$	Laguerre

Table 3.1: Association between continuous probability distributions and orthogonal polynomials.

Further generalizations to arbitrary probability measures beyond the Askey family were proposed in [119; 118], where the probability space is decomposed into multiple elements and local chaos expansions are employed in each sub-element. This approach allows PC to adapt to different regions in the random space and extends the valid integration time of gPC for low-dimensional systems. The approach taken in [37] is to use a restart procedure, where the chaos expansion is restarted at different time-steps in order to mitigate the long-time integration issues of chaos expansions. Another generalization toward arbitrary distributions is presented in [91], using only moment information of the involved distribution with a data-driven approach.

Although most of the extensions offer considerable improvements over the Hermite PCE, problems related to high dimensionality and long-time integration of spectral expansions still persist in most applications. For instance, in case of evolution equations with complex stochastic forcing (e.g. white noise), the limitations of PCE are discussed in [57; 16]. Both manuscripts noted that a rapidly increasing number of terms in PCE is needed to get a reasonable representation of the solution as time evolves. In this connection, we propose a method which addresses these drawbacks and offers a viable way to compute long-time solutions of evolution equations driven by white noise.

The plan of the rest of the chapter is as follows. Our methodology, Dynamical generalized Polynomial Chaos, is described in detail in Section 3.2. Numerical experiments comparing DgPC to Hermite PC and Monte Carlo simulations are presented in Section 3.3.

3.2 Description of the Proposed Method

Our key idea is to adapt the PCE to the dynamics of the system. Consider for concreteness the following SDE:

$$du(t) = \mathcal{L}(u)\,dt + \sigma\,dW(t), \quad u(0) = u_0, \quad t \in [0,T], \tag{3.1}$$

where $\mathcal{L}(u)$ is a general function of u (and possibly other deterministic or stochastic parameters), $W(t)$ is a Brownian motion, $\sigma > 0$ is a constant, and u_0 is an initial condition with a prescribed probability density. We assume that a solution exists and is unique on $[0,T]$. We also use W_t and $W(t)$ interchangeably.

As the system evolves, a fixed polynomial chaos basis adapted to u_0 and dW may not be optimal to represent the solution $u(t)$ for long times. Moreover, the dimension of the representation of dW increases with time t, which renders the PCE method computationally intractable even for moderate values of T. We will therefore introduce an increasing sequence of restart times $0 < t_j < t_{j+1} < T$ and construct a new PCE basis at each t_j based on the solution $u(t_j)$ and all additional random variables that need to be accounted for. A very similar methodology with $\sigma = 0$ was considered earlier in [37; 56]. When random forcing is present and satisfies the Markov property as in the above example, the restarting strategy allows us to "forget" past random variables that are no longer necessary and focus on a significantly smaller subset of random variables that influence the future evolution. As an example of application, we will show that the restarting strategy allows us to capture the invariant measure of the above SDE, when such a measure exists.

To simplify notation, we present our algorithm on the above scalar SDE with $\mathcal{L}(u)$ a function of u, knowing that all results also apply to systems of SDEs with minor modifications; see sections 3.2.2.4 and 3.3.

3.2.1 Formulation

First, we notice that the solution $u(t)$ of (3.1) is a random process depending on the initial condition and the paths of Brownian motion up to time t:

$$u = u(t;\, \{W_\tau,\, 0 \le \tau \le t\}, u_0).$$

Therefore, recalling the expansion for $W(t)$ (2.9), the solution at time t can be seen as a nonlinear functional of u_0 and the countably infinite variables $\boldsymbol{\xi}$. As previously noticed in [37; 56], the solution $u(t)$ can be represented as a linear chaos expansion in terms of the polynomials in itself and therefore, for sufficiently small later times $t+\varepsilon$, $\varepsilon > 0$, the solution $u(t+\varepsilon)$ can be efficiently captured by low order chaos expansions in $u(t)$ everything else being constant. Moreover, the solution $u(t+\varepsilon)$ on the interval $0 < \varepsilon < \varepsilon_0$ depends on $W_{[t,t+\varepsilon_0]}$ and not on values of W outside of this interval. This significantly reduces the number of random variables in $\boldsymbol{\xi}$ that need to be accounted for. This crucial observation clearly hinges upon the Markovian property of the dynamics. Hence, dividing the time horizon into small pieces and iteratively employing PCE offer a possible way to alleviate both curse of dimensionality and long-time integration problems.

We decompose the time horizon $[0,T]$ into n subintervals; namely $[0,t_1], [t_1,t_2], \ldots, [t_{n-1},T]$ where $0 = t_0 < t_1 < \ldots < t_n = T$. The idea is then to employ polynomial chaos expansion in each subinterval and restart the approximation depending on the distributions of both $\boldsymbol{\xi}_j$ and $u(t_j)$ at each t_j, $1 \le j < n$; see Figure 3.1. Here, $\boldsymbol{\xi}_j$ denotes the Gaussian random variables required for Brownian forcing on the interval $[t_j, t_{j+1}]$. Throughout the chapter, we utilize the term $T_{\boldsymbol{\alpha}}$ to represent orthonormal chaos basis involving its arguments. In order to establish evolution equations for chaos basis $T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_j)$, triple products $\mathbb{E}[T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_j)T_{\boldsymbol{\beta}}(\boldsymbol{\xi}_j)T_{\boldsymbol{\gamma}}(\boldsymbol{\xi}_j)]$ are needed. This procedure basically corresponds to computing the coefficients and indices in the Hermite product formula (3.3).

The probability distribution of $u(t_j)$ does not belong to any classical family such as the Askey family in general. The construction of the orthogonal polynomials in $u(t_j)$ is therefore fairly involved computationally and is based on the general PCE results mentioned earlier in the section. At each time step, we compute the moments of $u(t_j)$ using its previously obtained chaos representation and incorporate them in a modified Gram–Schmidt method. This is a computationally expensive step. Armed with the orthogonal basis, we then compute the triple products in $u(t_j)$ to perform the necessary Galerkin projections onto spaces spanned by this orthogonal basis and thus obtain evolution equations for the deterministic expansion coefficients [41; 68]. Letting $u_j := u(t_j)$, equations for the coefficients $(u_{j+1})_{\boldsymbol{\alpha}}$ of u_{j+1} are given in the general form

$$d(u_{j+1})_{\boldsymbol{\alpha}} = \mathbb{E}\left[T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_j, u_j)\, L\left(\sum_{\boldsymbol{\beta}} (u_{j+1})_{\boldsymbol{\beta}}\, T_{\boldsymbol{\beta}}(\boldsymbol{\xi}_j, u_j)\right)\right] ds + \sigma\, \mathbb{E}\left[T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_j, u_j)\, dW_t\right],$$

where $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathcal{J}$. Before integration of these expansion coefficients in time, u_j is represented in terms of its own orthogonal polynomials, which provides a description of the initial conditions on that interval. We then perform a high order time integration. Cumulants of the resulting solution are then computed to obtain relevant statistical information about the underlying distribution.

Remark 3.1. The proposed methodology does not require the computation of any probability density function (pdf) and rather depends only on its moments at each restart time. The statistical information contained in such moments provides useful data for decision making in modeling and, for determinate distributions, moments characterize the distribution uniquely. This aspect will be essential in the proof of convergence in section 3.2.4.

Comparing our method with probability distribution function methods, e.g., the Fokker–Planck equation, we note that it essentially evolves the coefficients of orthogonal polynomials of the projected solution in time instead of evolving the pdf.

Let Z_j represents the nonlinear chaos expansion mapping $(u_{j-1}, \boldsymbol{\xi}_{j-1})$ to u_j. Our scheme may be demonstrated by Figure 3.1.

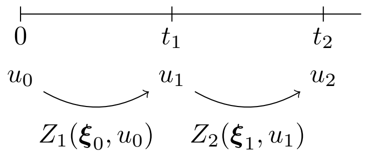

0
t_1
t_2
u_0
u_1
u_2
$Z_1(\boldsymbol{\xi}_0, u_0)$
$Z_2(\boldsymbol{\xi}_1, u_1)$

Figure 3.1: Propagation of chaos expansions in DgPC.

Mathematically, we have

$$u_j = Z_j(\boldsymbol{\xi}_{j-1}, u_{j-1}) = \sum_{\boldsymbol{\alpha} \in \mathcal{J}} (u_j)_{\boldsymbol{\alpha}} \, T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_{j-1}, u_{j-1}), \quad j \in \mathbb{N}. \tag{3.2}$$

In simulations, $\boldsymbol{\xi}_j$ is truncated with finite dimension $K \in \mathbb{N}$. Let also $N \in \mathbb{N}$ denote the maximum degree of the polynomials in the variables $(\boldsymbol{\xi}_j, u(t_j))$. Here, to simplify, K and N are chosen independently of j. The multi-indices having K dimensions and degree N belong to the set $\mathcal{J}_{K,N} := \{\boldsymbol{\alpha} = (\alpha_1, \alpha_2, .., \alpha_K) \,|\, \alpha_k \in \mathbb{N}_0, \, |\boldsymbol{\alpha}| \leq N\}$.

We now present our algorithm, called *Dynamical generalized Polynomial Chaos* (DgPC), in one dimension for concreteness; see also section 3.2.2.4.

Algorithm 1 Dynamical generalized Polynomial Chaos (DgPC) for SDEs

```
Decompose time domain $[0,T] = [t_0, t_1] \cup \ldots \cup [t_{n-1}, T]$
Initialize degrees of freedom $K, N$
Compute coefficients/indices in Hermite product formula for $\boldsymbol{\xi}_0$
for each time-step $t_j \geq 0$ do
    calculate moments $\mathbb{E}[(u_j)^m]$
    construct orthogonal polynomials $T_k(u_j)$
    compute triple products $\mathbb{E}[T_k(u_j)\, T_l(u_j)\, T_m(u_j)]$
    perform Galerkin projection onto span$\{T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_j, u_j)\}$
    set up initial conditions for the coefficients $(u_j)_{\boldsymbol{\alpha}}$
    evolve the expansion coefficients $(u_j)_{\boldsymbol{\alpha}}$
    compute cumulants
end for
```

Several remarks are in order. First, since our stochastic forcing has identically distributed independent increments, we observe that at each subinterval the distribution of $\boldsymbol{\xi}_j$ is the same. Computing and storing (in sparse format) the coefficients for the product formula (3.3) only once drastically reduces the computational time.

At each iteration, the random variable $u(t_j) = u_j$ is projected onto a finite dimensional chaos space and the next step is initialized with the projected random variable. For a d-dimensional SDE system, this projection leads to $\binom{K+N}{N} \times \binom{d+N}{N}$ terms in the basis for the subinterval $[t_j, t_{j+1}]$. Therefore, the total number of degrees of freedom used in $[0, t]$ becomes $n \times \binom{K+N}{N} \times \binom{d+N}{N}$; see also section 3.2.3. We emphasize that small values of K, N are utilized in each subinterval such that computations can be carried out quickly.

The idea of iteratively constructing chaos expansions for arbitrary measures was considered earlier in [37; 56; 99; 4; 5; 131]. In [4; 5], an iterative method is proposed to solve coupled multiphysics problems, where a dimension reduction technique is used to exchange information between iterations while allowing the construction of PC in terms of arbitrary (compactly supported) distributions at each iteration. A similar iterative procedure for Hermite PC was suggested for stochastic partial differential equations (SPDEs) with Brownian forcing in the conclusion section of [57] without any mathematical construction or numerical example. In [131], a multi-stage PC-based algorithm is presented based on the recursive formulation in [73] to compute second order moments of linear SPDEs. We also stress an important difference between our approach and [37]: our scheme solely depends on the statistical information, i.e., moments, which are available through chaos expansion, whereas [37] either requires the pdf of u_j at t_j or uses mappings to transform u_j back to the original input random variables, which in our problem is high dimensional, including the ξ variables.

3.2.2 Implementation

We now describe the implementation of our algorithm.

3.2.2.1 Moments

Provided that the distribution of the initial condition u_0 is known, several methods allow us to compute moments: analytic formulas, quadrature methods, or Monte Carlo sampling. Also, in the case of limited data, moments can be generated from raw data sets. Throughout the chapter, we assume that the moments of the initial condition can be computed to some large finite order so that the algorithm can be initialized.

We first recall the Hermite product formula, which establishes the multiplication of two PCEs (2.5) for random variables u and v. The chaos expansion for the product becomes

$$uv = \sum_{\boldsymbol{\alpha}\in\mathcal{J}} \sum_{\boldsymbol{\gamma}\in\mathcal{J}} \sum_{0\le\boldsymbol{\beta}\le\boldsymbol{\alpha}} C(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}) u_{\boldsymbol{\alpha}-\boldsymbol{\beta}+\boldsymbol{\gamma}} v_{\boldsymbol{\beta}+\boldsymbol{\gamma}}\, T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}), \tag{3.3}$$

where C is defined in (2.6); [75; 57]. As our applications include nonlinear terms, this formula will be used repeatedly to multiply Hermite chaos expansions.

To compute moments of the random variable u_j (recall (3.2)) at time step t_j, $j = 1, \dots, n$, we use its chaos representation and previously computed triple products recursively as follows. Due to the Markov property of the solution, the normalized chaos basis in $\boldsymbol{\xi}_{j-1}$ and u_{j-1} becomes a tensor product, and thus we can write

$$u_j = \sum_{\boldsymbol{\alpha}'} (u_j)_{\boldsymbol{\alpha}'}\, T_{\alpha'}(\boldsymbol{\xi}_{j-1}, u_{j-1}) = \sum_{\boldsymbol{\alpha},k} (u_j)_{\boldsymbol{\alpha},k}\, T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_{j-1})\, T_k(u_{j-1}),$$

where $\boldsymbol{\alpha}, \boldsymbol{\alpha}' \in \mathcal{J}$ and $k \in \mathbb{N}_0$. Then for an integer $m > 1$,

$$u_j^m = \sum_{\boldsymbol{\alpha},k} (u_j)_{\boldsymbol{\alpha},k}\, T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_{j-1})\, T_k(u_{j-1}) \sum_{\boldsymbol{\beta},l} (u_j^{m-1})_{\boldsymbol{\beta},l}\, T_{\boldsymbol{\beta}}(\boldsymbol{\xi}_{j-1})\, T_l(u_{j-1}).$$

Here, $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathcal{J}$ and $k, l \in \mathbb{N}_0$. Hence, using (3.3), the coefficients $(u_j^m)_{\boldsymbol{\gamma},r}$, where $\boldsymbol{\gamma} \in \mathcal{J}$ and $r \in \mathbb{N}_0$, can be calculated using the expression

$$(u_j^m)_{\boldsymbol{\gamma},r} = \sum_{\boldsymbol{\alpha}\in\mathcal{J}} \sum_{0\le\theta\le\boldsymbol{\gamma}} \sum_{k,l} C(\boldsymbol{\gamma},\theta,\boldsymbol{\alpha})\, \mathbb{E}[T_r(u_{j-1})T_k(u_{j-1})T_l(u_{j-1})]\, (u_j^{m-1})_{\boldsymbol{\gamma}-\theta+\boldsymbol{\alpha},l}\, (u_j)_{\theta+\boldsymbol{\alpha},k}, \tag{3.4}$$

which allows us to compute moments by simply taking the first coefficient, i.e. $\mathbb{E}[u_j^m] = (u_j^m)_{\mathbf{0},0}$. Even though the coefficients $C(\boldsymbol{\gamma},\theta,\boldsymbol{\alpha})$ can be computed offline, the triple products $\mathbb{E}[T_r(u_j)T_k(u_j)T_l(u_j)]$ must be computed at every step as the distribution of u_j evolves.

3.2.2.2 Orthogonal Polynomials

Given the real random variable u_j obtained by the orthogonal projection of the solution u onto homogeneous chaos at t_j, the Gram–Schmidt orthogonalization procedure can be used to construct the associated orthogonal polynomials of its continuous distribution. For theoretical reasons, we assume that the moment problem for u_j is uniquely solvable; i.e., the distribution is nondegenerate and uniquely determined by its moments [2; 36].

To construct orthogonal polynomials, the classical Gram–Schmidt procedure uses the following recursive relation

$$T_0 = 1, \quad T_m(u_j) = u_j^m - \sum_{l=0}^{m-1} a_l^m \, T_l(u_j), \quad m \geq 1,$$

where the coefficients are given by $a_l^m = \mathbb{E}[u_j^m \, T_l(u_j)] \left(\mathbb{E}[T_l^2(u_j)]\right)^{-1}$. Note that $\mathbb{E}$ denotes the Lebesgue–Stieltjes integral with respect to distribution of u_j. We observe that the coefficient a_l^m requires moments up to degree $2m-1$ as T_l is at most of degree $m-1$. Thus, normalization would need first $2m$ moments for orthonormal polynomials up to degree m.

After the construction of orthonormal polynomials of u_j, the basis $T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_j, u_j)$ can be generated by tensor products since $\boldsymbol{\xi}_j$ and u_j are independent. Moreover, triple products $\mathbb{E}[T_k(u_j) \, T_l(u_j) \, T_m(u_j)]$, where $k, l, m = 0, \ldots, N$, can be found by simply noting that this expectation is a triple sum involving moments up to order $3N$. Hence, the knowledge of moments yields not only the orthonormal set of polynomials but also the triple products required at each iterative step.

In our numerical computations, we prefer to utilize a modified Gram–Schmidt algorithm as the classical approach is overly sensitive to roundoff errors in cases of high dimensionality [44]. We refer the reader to [36] for other algorithms, e.g., Stieltjes and modified Chebyshev methods.

3.2.2.3 Initial Conditions

When the algorithm is reinitialized at the restart point t_j, it needs to construct the initial condition in terms of chaos variables for the next iteration on $[t_j, t_{j+1}]$. In other words, we need to find the coefficients $(u_j)_\alpha$ in terms of u_j and $\boldsymbol{\xi}_j$. To this end, we observe that the first two polynomials in u_j are given by $T_0(u_j) = 1$ and $T_1(u_j) = \sigma_{u_j}^{-1}(u_j - \mathbb{E}[u_j])$, where

σ_{u_j} is the standard deviation of the random variable u_j. Notice that $\sigma_{u_j} > 0$, because we assumed in the preceding section that the distribution was nondegenerate. Hence, the initial condition can be easily found by the relation

$$u_j = \mathbb{E}[u_j]T_0(u_j) + \sigma_{u_j}T_1(u_j) = \sum_{\boldsymbol{\alpha}\in\mathcal{J},\, k\in\mathbb{N}_0} (u_j)_{\boldsymbol{\alpha},k}\, T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_j)\, T_k(u_j).$$

3.2.2.4 Extension to Higher Dimensions

We now extend our algorithm to higher dimensions, and for concreteness, we consider two dimensions. Let $u_{j+1} = (v_1, v_2) \in \mathbb{R}^2$ be a two-dimensional random variable obtained by projection of the solution u onto homogeneous chaos space at time t_{j+1}.

The two-dimensional case requires the calculation of the mixed moments $\mathbb{E}[v_1^l\, v_2^m]$, $l, m \in \mathbb{N}\cup\{0\}$. In the case of independent components v_1, v_2, the moments become products of marginal moments and the basis $T_{\boldsymbol{\alpha}}(v_1, v_2)$ is obtained by tensor products of the one-dimensional bases. For a nonlinear system of equations, the components of the solution typically do not remain independent as time evolves. Therefore, we need to extend the procedure to correlated components.

Denoting by $Z_{j+1,1}$ and $Z_{j+1,2}$ the corresponding chaos expansions on $[t_j, t_{j+1}]$ so that $v_1 = Z_{j+1,1}(\boldsymbol{\xi}_j, u_j)$ and $v_2 = Z_{j+1,2}(\boldsymbol{\xi}_j, u_j)$, we compute the mixed moments by the change of variables formula

$$\mathbb{E}[v_1^l\, v_2^m] = \mathbb{E}_{(v_1,v_2)}[v_1^l\, v_2^m] = \mathbb{E}_{(\boldsymbol{\xi}_j,u_j)}\left[Z_{j+1,1}^l\, Z_{j+1,2}^m\right], \tag{3.5}$$

where $\mathbb{E}_{(\cdot)}$ represents the expectation with respect to the joint distribution induced by the subscript. Note that the latter integral in (3.5) can be computed with the help of previously computed triple products of $\boldsymbol{\xi}_j$ and u_j. Incidentally, similar transformation methods were used previously in [56; 4; 5].

Since the components may not be independent, the tensor product structure is lost but the construction of orthogonal polynomials is still possible. Based on the knowledge of marginal moments, we first compute the tensor product $T_{\boldsymbol{\alpha}}(v_1, v_2)$. However, this set is not orthogonal with respect to the joint probability measure of (v_1, v_2). Therefore, we further orthogonalize it via a Gram–Schmidt method. Note that this procedure requires only mixed moments $E_{(v_1,v_2)}[v_1^l v_2^m]$, which have already been computed in the previous step of the algorithm. It is worth noting that the resulting set of orthogonal polynomials is not unique as it depends on the ordering of monomials [130]. In applications, we consider the same ordering used for the set of multi-indices; see Table 2.1. Finally, calculation of triple products and initial conditions can be extended in an obvious way.

3.2.3 Computational Complexity

In this section, we discuss the computational costs of our algorithm using a vector-valued version of the SDE (3.1) in $\mathbb{R}^d$ for a fixed $d \in \mathbb{N}$. For each scalar SDE, we assume that the nonlinearity is proportional to mth order monomial with $m \in \mathbb{N}$. Furthermore, we compare costs of DgPC and Hermite PC in the case where both methods attain a similar order of error.

Throughout this section, K, N will denote the dimension and the degree in $\boldsymbol{\xi} = (\xi_1, \dots, \xi_K)$ for Hermite PC. Here $\boldsymbol{\xi}$ represents the truncation of d-dimensional Brownian motion; therefore, $K \gg d$. The time interval is fixed and given by $[0, 1]$. We also recall $M(K, N) = \binom{K+N}{K}$, the degrees of freedom for the simple truncation (2.7). For the DgPC method, we divide the interval into $n > 1$ identical subintervals so that $\Delta t = n^{-1}$. We now slightly change the notation of previous sections and denote by K_*, N_* the dimension and degree of polynomials of $\boldsymbol{\xi}_j = (\xi_{jK_*+1}, \dots, \xi_{jK_*+K_*})$ used in DgPC approximation for each $1 \leq j < n$. With additional d variables at each restart, the dimension of the multi-index set for DgPC becomes $M(K_*, N_*)M(d, N_*)$ due to the tensor product structure. Further, let $h < 1$ and $\zeta \geq 1$ denote the time step and global order of convergence for the time integration method employed in Hermite PC, respectively.

3.2.3.1 Computational costs

With the above notation, we summarize the computational costs in Table 3.2. All terms should be understood in big O notation.

The estimates in Table 3.2 are obtained as follows. Triple products for $\boldsymbol{\xi}$ with $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma} \in$

Flops	DgPC	Hermite PC
Offline	$K_* \times M(K_*, N_*)^3$	$K \times M(K, N)^3$
Time evolution	$d \times h^{-1} \times n^{1/\zeta} \times (M(K_*, N_*)M(d, N_*)) \times [1 + (m-1) \times (M(K_*, N_*)M(d, N_*))^2]$	$d \times h^{-1} \times M(K, N) \times [1 + (m-1) \times M(K, N)^2]$
Moments	$n \times (M(K_*, N_*)M(d, N_*))^3 \times [(6d - 3d^2) \times N_* + (d-1) \times M(d, 3N_*)]$	
Gram–Schmidt	$n \times M(d, N_*)^3$	
Triple products	$n \times M(d, N_*)^6$	
Initials	$n \times d \times (M(K_*, N_*)M(d, N_*))$	

Table 3.2: Comparison of computational costs for Hermite PC and DgPC.

$\mathcal{J}_{K,N}$ can be calculated by the equation

$$\begin{aligned}\mathbb{E}[T_{\boldsymbol{\alpha}}(\boldsymbol{\xi})T_{\boldsymbol{\beta}}(\boldsymbol{\xi})T_{\boldsymbol{\gamma}}(\boldsymbol{\xi})] &= \mathbb{E}\left[\prod_{i=1}^{K} H_{\alpha_i}(\xi_i)\prod_{j=1}^{K} H_{\beta_j}(\xi_j)\prod_{k=1}^{K} H_{\gamma_k}(\xi_k)\right], \\ &= \prod_{k=1}^{K}\mathbb{E}[H_{\alpha_k}(\xi_k)H_{\beta_k}(\xi_k)H_{\gamma_i}(\xi_k)].\end{aligned}$$

Thus, the offline cost is of order $K \times M(K, N)^3$ assuming that one-dimensional triple products can be read from a precomputed table.

Since a time discretization with error $O(h^\zeta)$ is employed for Hermite PC, then for DgPC in each subinterval, time steps of order $h^{-1}n^{(1-\zeta)/\zeta}$ should be used to attain the same global error. Without nonlinearity, the total cost in Hermite PC for evolution of a d-dimensional system becomes $d \times h^{-1} \times M(K, N)$. Due to the functional $\mathcal{L}(u) \propto u^m, m > 1$, the cost should also include the computation of u^m at each time integration step. This requires additional $d \times h^{-1} \times (m-1) \times M(K, N)^3$ work (see the computation of moments below). Hence, the corresponding total cost of evolution becomes the sum of these costs.

The coefficients of the kth power of a single projected variable u_j are given by

$$(u_j^k)_\gamma = \sum_{\boldsymbol{\alpha},\boldsymbol{\beta}}(u_j^{k-1})_{\boldsymbol{\alpha}}\,(u_j)_{\boldsymbol{\beta}}\,\mathbb{E}[T_{\boldsymbol{\alpha}}T_{\boldsymbol{\beta}}T_{\boldsymbol{\gamma}}], \quad 2 \le k \le 3N_*,$$

assuming that, after each multiplication, the variable is projected onto the PC basis. Here,

$\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma} \in \mathcal{J}_{K_*,N_*} \otimes \mathcal{J}_{d,N_*}$. Thus, the computation of marginal moments of all variables requires $d \times 3N_* \times (M(K_*, N_*)M(d, N_*))^3$ calculations at each restart time in DgPC. Since we use the same ordering for multidimensional moments as $\mathcal{J}_{d,3N_*}$, computing joint moments further needs $(d-1) \times (M(d, 3N_*) - 3dN_*) \times (M(K_*, N_*)M(d, N_*))^3$ amount of work.

The Gram–Schmidt procedure costs are cubic in the size of the Gram matrix, which has dimension $M(d, N_*)$ in DgPC. Each triple product $\mathbb{E}[T_{\boldsymbol{\alpha}}(u_j)T_{\boldsymbol{\beta}}(u_j)T_{\boldsymbol{\gamma}}(u_j)], \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma} \in \mathcal{J}_{d,N_*}$, is a triple sum; therefore, the total cost is proportional to $M(d, N_*)^6$. Also, initial conditions can be computed by orthogonal projection onto the DgPC basis, which costs $M(K_*, N_*)M(d, N_*)$ per dimension.

Note that although we employed simple truncation (2.7) for multi-indices, sparse truncation techniques can be utilized to further reduce the costs of both computing moments and evolution; see [75; 57; 66].

3.2.3.2 Error bounds and cost comparison

We now discuss the error terms in both algorithms. We posit that the Hermite PCE converges algebraically in N and K and that the error at time T satisfies

$$||u - u_{pce}||_{L^2} \lesssim T^{\delta} \, (N^{-\eta} + K^{-\lambda}), \tag{3.6}$$

for some constants $\eta, \lambda > 0$ and $\delta > 1$ depending on the SDE. Here $A \lesssim B$ denotes that $A \leq cB$ for $c > 0$ with $A, B \geq 0$. This assumption enforces an increase in the degrees of freedom N, K if one wants to maintain the accuracy in the long term. Algebraic convergence in K and the term T^{δ} stems from the convergence (2.10). We do not show errors resulting from time discretization in (3.6) since we already fixed those errors to the same order in both methods. Incidentally, even though we assumed algebraic convergence in N, exponential numerical convergence, depending on the regularity of the solution in $\boldsymbol{\xi}$, is usually observed in the literature [122; 123; 41; 75; 125; 118].

Although it is usually hard to quantify the constants η, λ, δ, this may be done for simple cases. For instance, for the Ornstein–Uhlenbeck (OU) process (3.24), using the exact solution and Fourier basis of cosines, we can obtain the parameters $\delta = 2$ and $\lambda = 3/2$ for short times. The parameter η does not play a role in this case since the SDE stays Gaussian and hence we take $N = 1$. We also note that for complex dynamics, η may depend on N; see [75; 118] in the case of the stochastic Burgers equation and advection equations.

Now, we fix the error terms in (3.6) to the same order $O(\varepsilon)$, $\varepsilon > 0$, by choosing $N = O(K^{\lambda/\eta})$. Since DgPC uses truncated $\boldsymbol{\xi}$ of dimension K_* and polynomials of degree N_* in each subinterval of size $\Delta t = n^{-1}$, the L^2 error at $T = 1$ has the form

$$||u - u_{dgpc}||_{L^2} \lesssim \frac{n}{n^\delta}\left(N_*^{-\eta} + K_*^{-\lambda}\right). \tag{3.7}$$

Thus, to maintain the same level ε of accuracy, we can choose

$$n^{1-\delta}K_*^{-\lambda} \cong K^{-\lambda} \quad \text{and} \quad n^{1-\delta}N_*^{-\eta} \cong K^{-\lambda}.$$

From Table 3.2, we observe that to minimize costs for DgPC, we can take $K_* = d$ and $N_* = 1$ and maximize n according to the previous equation as

$$n \cong K^{\frac{\lambda}{\delta-1}}. \tag{3.8}$$

With these choices of parameters, the total computational cost for DgPC is of algebraic order in K and in general, is dominated by the computation of moments.

For Hermite PC with large K and $N = O(K^{\lambda/\eta})$ (or $N = O(\lambda/\eta \log K)$ in the case of exponential convergence in N), both offline and evolution stages include the cubic term $M(K,N)^3$. Both costs increase exponentially in K. Therefore, the asymptotic analysis suggests that DgPC can be performed with substantially lower computational costs using frequent restarts (equation (3.8)) in nonlinear systems driven by high dimensional random forcing.

3.2.4 Convergence Results

We now consider the convergence properties of our scheme as degrees of freedom tend to infinity.

3.2.4.1 Moment problem and density of polynomials

There is an extensive literature on the moment problem for probability distributions; see [2; 11; 97; 34] and their references. We provide the relevant background in the analysis of DgPC.

Definition 3.2. (Hamburger moment problem) *For a probability measure μ on $(\mathbb{R}, \mathcal{B}(\mathbb{R}))$, the moment problem is uniquely solvable provided moments of all orders $\int_{\mathbb{R}} x^k \mu(dx)$, $k \in \mathbb{N}_0$, exist and they uniquely determine the measure. In this case, the distribution μ is called determinate.*

There are several sufficient conditions for a one-dimensional distribution to be determinate in Hamburger sense: these are compact support, exponential integrability and Carleman's moment condition [2; 11]. For instance, Gaussian and uniform distributions are determinate, whereas the lognormal distribution is not. The moment problem is intrinsically related to the density of associated orthogonal polynomials. Indeed, if the cumulative distribution $\mathbb{F}_u$ of a random variable u is determinate, then the corresponding orthogonal polynomials constitute a dense set in $L^2(\mathbb{R}, \mathbb{B}(\mathbb{R}), \mathbb{F}_u)$ and therefore also in $L^2(\Omega, \sigma(u), \mathbb{P})$ [2; 11; 34]. Additionally, finite dimensional distributions on $\mathbb{R}^d$ with compact support are determinate; see [97].

Now, let $\boldsymbol{\zeta} = (\zeta_i)_{i\in\mathbb{N}}$ denote an independent collection of general random variables, where each ζ_i (not necessarily identically distributed) has finite moments of all orders and its cumulative distribution function is continuous. Under these assumptions, [30] proves the following theorem.

Theorem 3.3. *For any random variable $\eta \in L^2(\Omega, \sigma(\boldsymbol{\zeta}), \mathbb{P})$, gPC converges to η in L^2 if and only if the moment problem for each ζ_k, $k \in \mathbb{N}$, is uniquely solvable.*

This result generalizes the convergence of Hermite PCE to general random variables whose laws are determinate. Notably, to prove L^2 convergence of chaos expansions, it is enough to check one of the determinacy conditions mentioned above for each one-dimensional ζ_i.

We now consider the relation between determinacy and distributions of SDEs. Consider the following diffusion

$$du_t = b(u_t)\, dt + \sigma(u_t)\, dW_t, \quad u(0) = u_0, \quad t \in [0, T], \tag{3.9}$$

where W_t is d-dimensional Brownian motion, $u_0 \in \mathbb{R}^d$, and $b : \mathbb{R}^d \to \mathbb{R}^d, \sigma : \mathbb{R}^d \to \mathbb{R}^{d\times d}$ are globally Lipschitz and satisfy the usual linear growth bound. These conditions imply that the SDE has a unique strong solution with continuous paths [90]. Determinancy of the distribution of u_t is established by the following theorem; see [97; 113; 32] for details.

Theorem 3.4. *The law of the solution of* (3.9) *is determinate if* $\sup_x ||\sigma\sigma^T(x)||$ *is finite.*

3.2.4.2 L^2 convergence with a finite number of restarts

Consider the solution of the SDE (3.9) written as

$$u_t = F_t(u_0, \boldsymbol{\xi}_0^t), \tag{3.10}$$

where $F_t : \mathbb{R}^d \times \mathbb{R}^\infty \to \mathbb{R}^d$ is the exact evolution operator mapping the initial condition u_0 and $\boldsymbol{\xi}_0^t = (\xi_1^t, \xi_2^t, \dots)$ to the solution u_t at time $t > 0$. Here, with a slight change of notation, $\boldsymbol{\xi}_0^t$ represents Brownian motion on the interval $[0, t]$. Note that future solutions $u_{t+\tau}$, $\tau > 0$, are obtained by $F_t(u_\tau, \boldsymbol{\xi}_\tau^{t+\tau})$, where $\boldsymbol{\xi}_\tau^{t+\tau}$ denotes Brownian motion on the interval $[\tau, t+\tau]$.

Even though the distribution of the exact solution u_t is determinate under the hypotheses of Theorem 3.4, it is not clear that this feature still holds for the projected random variables. To address this issue, we introduce, for $R \in \mathbb{R}_+$, a truncation function $\chi_R \in C_c^\infty(\mathbb{R}^d \to \mathbb{R}^d)$

$$\chi_R(u) := \begin{cases} u, & \text{when } u \in A_R, \\ 0, & \text{when } u \in \mathbb{R}^d \setminus A_{3R}, \end{cases}$$

where $A_R := \prod_{i=1}^d [-R, R] \subset \mathbb{R}^d$, and such that $\chi_R(u)$ decays smoothly and sufficiently slowly on A_R^c such that the Lipschitz constant $\mathrm{Lip}(\chi_R)$ of χ_R equals 1; i.e., for $|\cdot|$, the Euclidean distance in $\mathbb{R}^d$, $|\chi_R(u) - \chi_R(v)| \le |u - v|$. Such a function is seen to exist by appropriate mollification for each coordinate using the continuous, piecewise smooth, and odd function $\chi(u_1)$ defined by $R - |u_1 - R|$ on $[0, 2R]$, and extended by 0 outside $[-2R, 2R]$. This truncation is a theoretical tool that allows us to ensure that all distributions with support properly restricted on compact intervals remain determinate; see [97].

We can now introduce the following approximate solution operators for $R \in \mathbb{R}_+$ and $M \in \mathbb{N}_0$:

$$F_t^R(u, \boldsymbol{\xi}_0^t) := \chi_R \circ F_t(u, \boldsymbol{\xi}_0^t) : \mathbb{R}^d \times \mathbb{R}^\infty \to A_{3R}, \tag{3.11}$$

$$F_t^{M,R}(u, \boldsymbol{\xi}_0^t) := P^M \circ F_t^R(u, \boldsymbol{\xi}_0^t) = \sum_{i=0}^{M-1} F_{i,t}^R T_i(u, \boldsymbol{\xi}_0^t), \tag{3.12}$$

where P^M denotes the orthogonal projection onto polynomials in u and $\boldsymbol{\xi}_0^t$, and M is the total number of degrees of freedom used in the expansion. Throughout this section, we use a linear indexing to represent polynomials in both u (restricted on A_{3R}) and the random variables $\boldsymbol{\xi}$.

The main assumptions we imposed on the solution operator F_t are summarized as follows:

i) $\mathbb{E}_{\boldsymbol{\xi}_0^t}|F_t(0,\boldsymbol{\xi}_0^t)|^p \le C_t$, where $0 < C_t < \infty$ and $p = 2+\epsilon$ with $\epsilon > 0$.

ii) $\mathbb{E}_{\boldsymbol{\xi}_0^t}|F_t(u,\boldsymbol{\xi}_0^t) - F_t(v,\boldsymbol{\xi}_0^t)|^p \le \rho_t^p\,|u-v|^p$, where $u,v \in \mathbb{R}^d$, $0 < \rho_t < \infty$, and $p = 2+\epsilon$ with $\epsilon > 0$.

iii) For $\varepsilon > 0$ and $R > 0$, there is $M(\varepsilon,R,t)$ so that $\mathbb{E}_{\boldsymbol{\xi}_0^t}|F_t^R(u,\boldsymbol{\xi}_0^t) - F_t^{M,R}(u,\boldsymbol{\xi}_0^t)|^2 \le \varepsilon$, where $u \in A_{3R}$.

Assumption i) is a stability estimate ensuring that the chain remains bounded in pth norm starting from a point in $\mathbb{R}^d$ (here 0 without loss of generality owing to ii)). Assumption ii) is a Lipschitz growth condition controlled by a constant ρ_t. These assumptions involve the L^p norm, and hence are slightly stronger than control in the L^2 sense. Note that F_t^R also satisfies assumption ii) with the same constant ρ_t since $\mathrm{Lip}(\chi_R) = 1$. Finally, assumption iii) is justified by the Weierstrass approximation theorem for the u variable in A_{3R} and by the Cameron–Martin theorem to handle the $\boldsymbol{\xi}$ variable.

Consider now the following versions of the Markov chains:

$$u_{j+1} := F_t(u_j,\boldsymbol{\xi}_j), \quad j = 0,\dots,n-1, \tag{3.13}$$

$$u_{j+1}^R := F_t^R(u_j^R,\boldsymbol{\xi}_j), \quad u_0^R = \chi_R(u_0), \quad j = 0,\dots,n-1, \tag{3.14}$$

$$u_{j+1}^{M,R} := F_t^{M,R}(u_j^{M,R},\boldsymbol{\xi}_j), \quad u_0^{M,R} = \chi_R(u_0), \quad j = 0,\dots,n-1, \tag{3.15}$$

where $u_0 \in \mathbb{R}^d$ (possibly random and independent of the variables $\boldsymbol{\xi}$) and $\boldsymbol{\xi}_j = \boldsymbol{\xi}_{jt}^{(j+1)t}$ is a sequence of i.i.d. Gaussian random variables representing Brownian motion on the interval $[jt,(j+1)t]$. We also use the notation $\boldsymbol{\xi}_0^{jt}$ to denote Brownian motion on the interval $[0,jt]$. Then we have the first result.

Lemma 3.5. *Assume i), ii), and iii) hold, and let $n \in \mathbb{N}$ be finite. Suppose also that $\mathbb{E}|u_0|^p = C_0 < \infty$. Then, for each $\varepsilon > 0$, there exists $R \in \mathbb{R}_+$ depending on n, t, and C_0 such that $\mathbb{E}|u_j - u_j^R|^2 \leq \varepsilon$ for each $0 \leq j \leq n$.*

Proof. Let $\varepsilon > 0$ be fixed. Using properties i) and ii), we observe that

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\xi}_0^{(j+1)t}}|F_t(u_j,\boldsymbol{\xi}_j)|^p &\leq 2^{p-1}\left(\mathbb{E}_{\boldsymbol{\xi}_0^{jt}}\mathbb{E}_{\boldsymbol{\xi}_j}|F_t(u_j,\boldsymbol{\xi}_j) - F_t(0,\boldsymbol{\xi}_j)|^p + \mathbb{E}_{\boldsymbol{\xi}_0^{jt}}\mathbb{E}_{\boldsymbol{\xi}_j}|F_t(0,\boldsymbol{\xi}_j)|^p\right),\\ &\leq 2^{p-1}\left(\rho_t^p\,\mathbb{E}_{\boldsymbol{\xi}_0^{jt}}|u_j|^p + C_t\right),\end{aligned}$$

where for all j, $\boldsymbol{\xi}_j$ and $\boldsymbol{\xi}_0^t$ are identically distributed. Then, by induction, we obtain

$$\mathbb{E}_{u_0,\boldsymbol{\xi}_0^{(j+1)t}}|u_{j+1}|^p \leq (2^{p-1}\rho_t^p)^{j+1}\,\mathbb{E}|u_0|^p + C_{t,n}^p \leq C_{t,n,C_0}, \quad 0 \leq j \leq n-1, \tag{3.16}$$

where the constant C_{t,n,C_0} is bounded. The last inequality indicates that pth norm of the solution grows with j, possibly exponentially, but remains bounded.

For $u_j = F_{jt}(u_0, \boldsymbol{\xi}_0^{jt})$, we note that $\mathbb{E}|u_j|^2 = \mathbb{E}_{u_0,\boldsymbol{\xi}_0^{jt}}|u_j|^2$ since u_j is independent of future forcing. By Markov's inequality and (3.16), we get

$$\mathbb{P}(u_j \in A_R^c) \leq \frac{\mathbb{E}|u_j|^2}{R^2} \leq C_{t,n,C_0}\,R^{-2}, \quad 1 \leq j \leq n. \tag{3.17}$$

Let $\mathbb{1}_A$ be the indicator function of the set A. Using (3.16) and (3.17), we compute the following error bound

$$\begin{aligned}\mathbb{E}|u_j - \chi_R u_j|^2 &= \mathbb{E}[(\mathbb{1}_{\{u_j\in A_R\}} + \mathbb{1}_{\{u_j\in A_R^c\}})|u_j - \chi_R u_j|^2], \quad 1 \leq j \leq n,\\ &\leq 4\,\mathbb{E}[\mathbb{1}_{\{u_j\in A_R^c\}}|u_j|^2] \leq 4\,(\mathbb{E}|u_j|^p)^{2/p}\,\mathbb{P}(u_j \in A_R^c)^{1/q}, \quad p = 2+\epsilon, q = 1 + 2\epsilon^{-1},\\ &\leq C_{t,n,C_0}\,R^{-2/q}.\end{aligned} \tag{3.18}$$

Then, using property ii) gives

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\xi}_{j-1}}|u_j - u_j^R|^2 &= \mathbb{E}_{\boldsymbol{\xi}_{j-1}}|F_t(u_{j-1},\boldsymbol{\xi}_{j-1}) - F_t^R(u_{j-1}^R,\boldsymbol{\xi}_{j-1})|^2,\\ &\leq (1+\delta^{-1})\mathbb{E}_{\boldsymbol{\xi}_{j-1}}|u_j - \chi_R u_j|^2\\ &\qquad + (1+\delta)\mathbb{E}_{\boldsymbol{\xi}_{j-1}}|\chi_R F_t(u_{j-1},\boldsymbol{\xi}_{j-1}) - F_t^R(u_{j-1}^R,\boldsymbol{\xi}_{j-1})|^2, \quad \delta > 0,\\ &\leq (1+\delta^{-1})\mathbb{E}_{\boldsymbol{\xi}_{j-1}}|u_j - \chi_R u_j|^2 + (1+\delta)\rho_t^2|u_{j-1} - u_{j-1}^R|^2.\end{aligned}$$

Taking expectation with respect to remaining measures and using (3.18) yields

$$\begin{aligned}\mathbb{E}|u_j - u_j^R|^2 &\le (1+\delta)\rho_t^2\,\mathbb{E}|u_{j-1} - u_{j-1}^R|^2 + (1+\delta^{-1})C_{t,n,C_0}\,R^{-2/q},\\ &\le ((1+\delta)\rho_t^2)^j\mathbb{E}|u_0 - u_0^R|^2 + (1+\delta^{-1})C_{t,n,C_0}\,R^{-2/q}\sum_{i=0}^{j-1}((1+\delta)\rho_t^2)^i. \qquad (3.19)\end{aligned}$$

Setting $\delta = 1$ in (3.19) entails

$$\mathbb{E}|u_j - u_j^R|^2 \le 2^j\rho_t^{2j}\,\mathbb{E}|u_0 - u_0^R|^2 + \,C_{t,n,C_0}\,R^{-2/q}.$$

Since $\mathbb{E}|u_0|^p < \infty$, we also have $\mathbb{E}|u_0 - u_0^R|^2 = O(R^{-2/q})$. Thus, we can choose $R(n,t,C_0) \in \mathbb{R}_+$ large enough such that $\mathbb{E}|u_j - u_j^R|^2 \le \varepsilon$ for each $0 \le j \le n$. $\square$

Based on the preceding lemma, we prove that the chain (3.15) converges to the solution of the chain (3.13) in the setting of a finite number of restarts.

Theorem 3.6. *Under the assumptions of Lemma 3.5, for each $\varepsilon > 0$, there exists $R(n,t,C_0) \in \mathbb{R}_+$ and then $M(R,n,t) \in \mathbb{N}_0$ such that $\mathbb{E}|u_j - u_j^{M,R}|^2 \le \varepsilon$ for each $0 \le j \le n$.*

Proof. Let $\varepsilon > 0$ be fixed. By the previous lemma, choose $R > 0$ such that $\mathbb{E}|u_j - u_j^R|^2 \le \varepsilon/4$ for $0 \le j \le n$. Then, using calculations similar to those above, by ii) we get

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\xi}_{j-1}}|u_j^R - u_j^{M,R}|^2 &\le (1+\delta)\mathbb{E}_{\boldsymbol{\xi}_{j-1}}|F_t^R(u_{j-1}^R,\boldsymbol{\xi}_{j-1}) - F_t^R(u_{j-1}^{M,R},\boldsymbol{\xi}_{j-1})|^2\\ &\qquad + (1+\delta^{-1})\mathbb{E}_{\boldsymbol{\xi}_{j-1}}|F_t^R(u_{j-1}^{M,R},\boldsymbol{\xi}_{j-1}) - F_t^{M,R}(u_{j-1}^{M,R},\boldsymbol{\xi}_{j-1})|^2, \delta > 0,\\ &\le 2\rho_t^2\,|u_{j-1}^R - u_{j-1}^{M,R}|^2\\ &\qquad + 2\,\mathbb{E}_{\boldsymbol{\xi}_{j-1}}|F_t^R(u_{j-1}^{M,R},\boldsymbol{\xi}_{j-1}) - F_t^{M,R}(u_{j-1}^{M,R},\boldsymbol{\xi}_{j-1})|^2,\ \delta = 1,\\ &= \rho_*\,|u_{j-1}^R - u_{j-1}^{M,R}|^2 + 2\,\mathbb{E}_{\boldsymbol{\xi}_{j-1}}|F_t^R(u_{j-1}^{M,R},\boldsymbol{\xi}_{j-1}) - F_t^{M,R}(u_{j-1}^{M,R},\boldsymbol{\xi}_{j-1})|^2,\end{aligned}$$

where $j \ge 1$ and $\rho_* := 2\rho_t^2$. Then by iii), we choose $M(R,n,t)$ sufficiently large so that

$$\mathbb{E}_{\boldsymbol{\xi}_{j-1}}|u_j^R - u_j^{M,R}|^2 \le \rho_*\,|u_{j-1}^R - u_{j-1}^{M,R}|^2 + \frac{\varepsilon}{4}\frac{\rho_* - 1}{\rho_*^n - 1}. \qquad (3.20)$$

The last inequality can be rewritten as

$$\mathbb{E}|u_j^R - u_j^{M,R}|^2 \le \rho_*^j\,\mathbb{E}|u_0^R - u_0^{M,R}|^2 + \varepsilon/4 = \varepsilon/4.$$

Therefore, $\mathbb{E}|u_j - u_j^{M,R}|^2 \le \varepsilon$ for each $1 \le j \le n$. $\square$

3.2.4.3 Convergence to invariant measures and long time evolution

Consider the setting of the solution operator to the SDE given in (3.10). As T increases to ∞, the random variable $u(T)$ may converge in distribution to a limiting random variable u_∞, whose distribution is the invariant measure of the evolution equation (3.10). Although there are many efficient ways to analyze and compute such invariant measures (see for instance [108]), we wish to show that our iterative algorithm also converges to that invariant measure as degrees of freedom tend to infinity. In other words, our PCE-based method allows us to remain accurate for long-time evolutions.

Now we iterate each discrete chain in (3.13), (3.14) and (3.15) for an arbitrary $j \in \mathbb{N}_0$. In order to ensure long-time convergence, we need a stricter condition than ii) and impose instead the following contraction condition

ii') $\mathbb{E}_{\boldsymbol{\xi}_0^t}|F_t(u,\boldsymbol{\xi}_0^t) - F_t(v,\boldsymbol{\xi}_0^t)|^p \leq \rho_t^p\,|u-v|^p$, where $0 < \rho_t < 1$, $p = 2+\epsilon$, $\epsilon > 0$, and $u, v \in \mathbb{R}^d$.

We first prove the following result about the existence and uniqueness of an invariant measure for the original chain (3.13).

Lemma 3.7. *Under conditions i) and ii'), there exists a unique invariant measure ν of the chain* (3.13) *with bounded pth moment. Moreover, if $\mathbb{E}|u_0|^p < \infty$, then $\mathbb{E}|u_j|^p$ is bounded uniformly in j.*

Proof. For $u \in \mathbb{R}^d$ (and using that $|a+b|^p \leq (1+\delta)^p|a|^p + (1+\delta^{-1})^p|b|^p$), we compute

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\xi}_0^t}|F_t(u,\boldsymbol{\xi}_0^t)|^p &\leq (1+\delta)^p\,\mathbb{E}_{\boldsymbol{\xi}_0^t}|F_t(u,\boldsymbol{\xi}_0^t) - F_t(0,\boldsymbol{\xi}_0^t)|^p + (1+\delta^{-1})^p\,\mathbb{E}_{\boldsymbol{\xi}_0^t}|F_t(0,\boldsymbol{\xi}_0^t)|^p, \quad \delta > 0,\\ &\leq ((1+\delta)\rho_t)^p\,|u|^p + (1+\delta^{-1})^p\,C_t,\\ &= \rho_*\,|u|^p + C,\end{aligned}$$

where δ is chosen so that $\rho_* := ((1+\delta)\rho_t)^p < 1$ and $C < \infty$. Thus, there exists a Lyapunov function $V(u) = |u|^p$ with a constant $\rho_* < 1$. This in turn implies the existence of an invariant measure ν with finite pth moment for the process (3.13); see [51, Corollary 4.23]. Uniqueness also follows from assumption ii'); see [51, Theorem 4.25].

Let v_0 be a random variable with law ν and independent of u_0. Consider the chain $v_{j+1} = F_t(v_j, \boldsymbol{\xi}_j)$ started from v_0. Note $v_j \sim \nu$ for all $j \in \mathbb{N}_0$. Then, from the bound

$$\mathbb{E}|u_j - v_j|^p \leq (\rho_t^p)^j \mathbb{E}|u_0 - v_0|^p,$$

we conclude that $\sup_j \mathbb{E}|u_j|^p < \infty$. □

The following theorem establishes the exponential convergence of the PC chain (3.15) to the chain (3.13) as time j increases.

Theorem 3.8. *Assume i), ii') and iii) hold, and $\mathbb{E}|u_0|^p < \infty$. Then, for each $\varepsilon > 0$, there exists $R > 0$ independent of j, and then $M(R,t) \in \mathbb{N}_0$, such that $\mathbb{E}|u_j - u_j^{M,R}|^2 \leq \varepsilon$ for all $j \in \mathbb{N}_0$.*

Proof. Let $\varepsilon > 0$ be given. By Lemma 3.7, $\mathbb{E}|u_j|^p$ is bounded uniformly in j. The inequality (3.18) holds with a constant independent of j. Then, choosing $\delta > 0$ so that $(1+\delta)\rho_t^2 < 1$ in (3.19) implies that there exists a constant $R(t,\nu) > 0$ independent of j such that $\mathbb{E}|u_j - u_j^R|^2 \leq \varepsilon/4$.

As in the proof of Theorem 3.6, we choose $\delta > 0$ so that $\rho_* := (1+\delta)\rho_t^2 < 1$. Then, with $M(R,t) \in \mathbb{N}_0$ large enough, (3.20) is replaced by

$$\mathbb{E}_{\boldsymbol{\xi}_{j-1}}|u_j^R - u_j^{M,R}|^2 \leq \rho_* \, |u_{j-1}^R - u_{j-1}^{M,R}|^2 + \frac{\varepsilon}{4}(1-\rho_*), \quad j \geq 1.$$

Therefore,

$$\mathbb{E}|u_j^R - u_j^{M,R}|^2 \leq \rho_*^j \, \mathbb{E}|u_0^R - u_0^{M,R}|^2 + \frac{\varepsilon}{4}(1-\rho_*)(1-\rho_*)^{-1} = \varepsilon/4, \quad j \geq 1.$$

This concludes our proof of convergence of the DgPC method to the invariant measure ν. □

Remark 3.9. It would be desirable to obtain that the chain (corresponding to $R = \infty$, or equivalently no support truncation) $u_{j+1}^M = F_t^M(u_j^M, \boldsymbol{\xi}_j)$ remains determinate. We were not able to do so and instead based our theoretical results on the assumption that the true distributions of interest were well approximated by compactly supported distributions. In practice, the range of M has to remain relatively limited for at least two reasons. First, large M rapidly involves very large computational costs; and second, the determination of measures from moments becomes exponentially ill-posed as the degree of polynomials N increases. For these reasons, the support truncation has been neglected in the following numerical section since, heuristically, for large R and limited N, the computation of (low order) moments of distributions with rapidly decaying density at infinity is hardly affected by such a support truncation.

3.3 Numerical Experiments

In this section, we present several numerical simulations of our algorithm and discuss the implementation details. We consider several of the equations described in [16; 38] to show that PCE-based simulations may perform well in such settings. We mostly consider the following two-dimensional nonlinear system of coupled SDEs:

$$\begin{aligned} du(t) &= -(b_u + a_u v(t))u(t)\,dt + f(t)\,ds + \sigma_u\,dW_u(t), \\ dv(t) &= -(b_v + a_v u(t))\,v(t)\,dt + \sigma_v\,dW_v(t), \end{aligned} \tag{3.21}$$

where $a_u, a_v \geq 0$, $b_u, b_v > 0$ are damping parameters, $\sigma_u, \sigma_v > 0$ are constants, and W_u and W_v are two real independent Wiener processes.

The system (3.21) was proposed in [38] to study filtering of turbulent signals which exhibit intermittent instabilities. The performance of Hermite PCE is analyzed in various dynamical regimes by the authors in [16], who conclude that truncated PCE struggle to accurately capture the statistics of the solution in the long term due to both the truncated expansion of the white noise and neglecting higher order terms, which become crucial because of the nonlinearities. For a review of the different dynamical regimes that (3.21) exhibits, we refer the reader to [38; 16].

For the rest of the section, $T \in \mathbb{R}_+$ stands for the endpoint of the time interval while $\Delta t = T/n$ denotes the time-step after which restarts occur at $t_j = j\Delta t$. Moreover, K denotes the number of basic random variables ξ_k used in the truncation of the expansion (2.9). In the presence of multiple Wiener processes it will denote the total number of variables. We slightly change the previous notation and let N and L denote the maximum degree of the polynomials in $\boldsymbol{\xi}$ and u_j, respectively. Recall that u_j is the projected PC solution at t_j.

Furthermore, following [75; 57] we choose the orthonormal bases for $L^2[t_{j-1}, t_j]$ as

$$m_{j,1}(t) = \frac{1}{\sqrt{t_j - t_{j-1}}}, \; m_{j,k}(t) = \sqrt{\frac{2}{t_j - t_{j-1}}} \cos\left(\frac{(k-1)\pi(t - t_{j-1})}{t_j - t_{j-1}}\right), \; k \geq 2, \; t \in [t_{j-1}, t_j]. \tag{3.22}$$

Other possible options include sine functions, a combination of sines and cosines, and wavelets. We refer the reader to [78] for details on the use of wavelets to deal with discontinuities for random inputs.

Assuming that the solution of (3.21) is square integrable, we utilize intrusive Galerkin projections in order to establish the following equations for the coefficients of the PCE:

$$\begin{aligned}
\dot{u}_{\boldsymbol{\alpha}} &= -b_u\, u_{\boldsymbol{\alpha}} - a_u\, (uv)_{\boldsymbol{\alpha}} + f\,\delta_{\boldsymbol{\alpha}\mathbf{0}} + \sigma_u\, E[\dot{W}_u T_{\boldsymbol{\alpha}}],\\
\dot{v}_{\boldsymbol{\alpha}} &= -b_v\, v_{\boldsymbol{\alpha}} - a_v\, (uv)_{\boldsymbol{\alpha}} + \sigma_v\, E[\dot{W}_v T_{\boldsymbol{\alpha}}].
\end{aligned}$$

This system of ODEs is then solved by either a second- or a fourth-order time integration method. Finally, we note that other methods are also available to compute the PCE coefficients such as nonintrusive projection and collocation methods [124; 66; 41; 123; 112].

In most of our numerical examples, the dynamics converge to an invariant measure. To demonstrate the convergence behavior of our algorithm, we compare our results to exact second order statistics or Monte Carlo simulations with sufficiently high sampling rate M_{samp} (e.g., Euler–Maruyama or weak Runge–Kutta methods [64]) where the exact solution is not available. Comparisons involve the following relative pointwise errors:

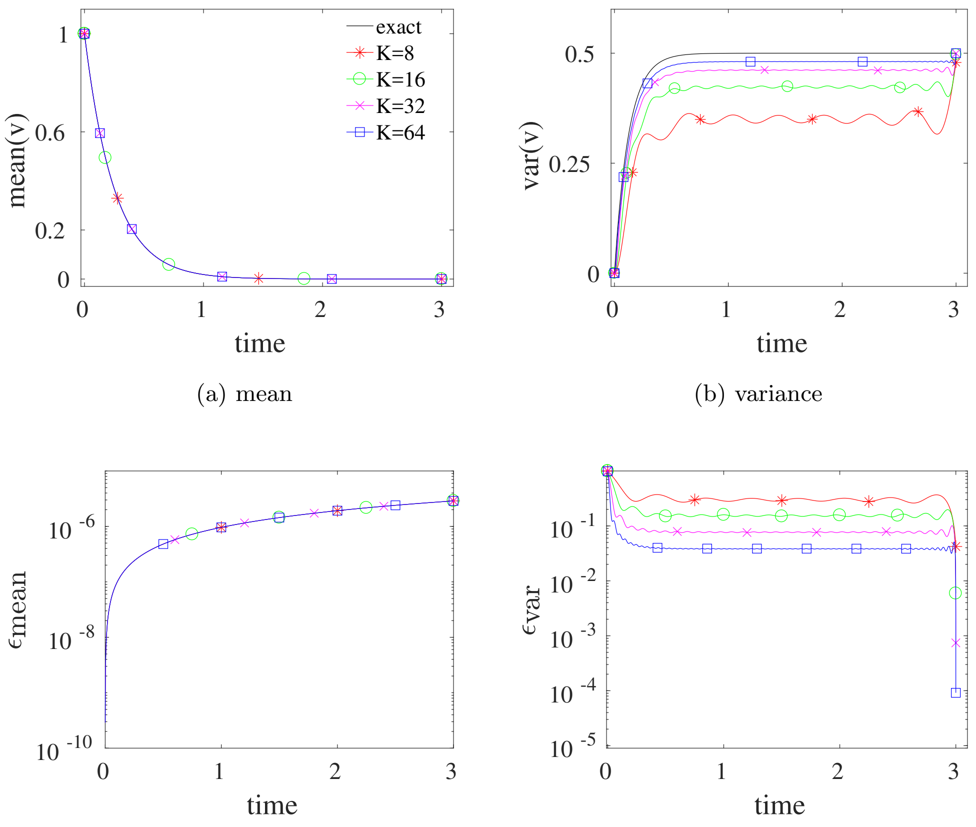
$$\epsilon_{\text{mean}}(t) := \left|\frac{\mu_{\text{pce}}(t) - \mu_{\text{exact}}(t)}{\mu_{\text{exact}}(t)}\right|, \quad \epsilon_{\text{var}}(t) := \left|\frac{\sigma^2_{\text{pce}}(t) - \sigma^2_{\text{exact}}(t)}{\sigma^2_{\text{exact}}(t)}\right|, \tag{3.23}$$

where μ, σ^2 represents the mean and variance. In the following figures, we plot the evolution of mean, variance, ϵ_{mean} and ϵ_{var} using the same legend. In addition, we exhibit the evolution of higher order cumulants, which will be denoted by κ to demonstrate the convergence to steady state as time grows. Finally, in our numerical computations, we make use of the C++ library UQ Toolkit [26].

Example 3.10. As a first example, we consider the one-dimensional Ornstein-Uhlenbeck (OU) process

$$dv(t) = -b_v\, v(t)\, dt + \sigma_v\, dW_v(t), \quad t \in [0, T], \quad v(0) = v_0, \tag{3.24}$$

on $[0, 3]$ with the parameters $b_v = 4, \sigma_v = 2, v_0 = 1$. We first present the results of Hermite PCE.



(a) mean (b) variance

(c) ϵmean (d) ϵvar

Figure 3.2: Hermite PC for the Ornstein-Uhlenbeck process.

Figure 3.2 shows that Hermite PC captures the mean accurately but approximation for the variance is accurate only at the endpoint. This is a consequence of the expansion in (2.11), which violates the Markov property of Brownian motion and is inaccurate for all $0 < t < T$, while it becomes (spectrally) accurate at the endpoint T. Using frequent restarts in DgPC, these oscillations are significantly attenuated as expected; see Figure 3.3b. This oscillatory behavior could also be alleviated by expanding the Brownian motion as in (2.11) on subintervals of $(0,1)$. Note that the global basis functions $T_{\boldsymbol{\alpha}}(\boldsymbol{\xi})$ which implicitly depend on the endpoint T may also be filtered by taking the conditional expectation $\mathbb{E}[T_{\boldsymbol{\alpha}}(\boldsymbol{\xi})|\mathcal{F}_t^W]$, where $\mathcal{F}_t^W$ is the σ-algebra corresponding to Brownian motion up to time t [75; 84]. We do not pursue this issue here but note that the accuracy of the different methods should be compared at the end of the intervals of discretization of Brownian motion.

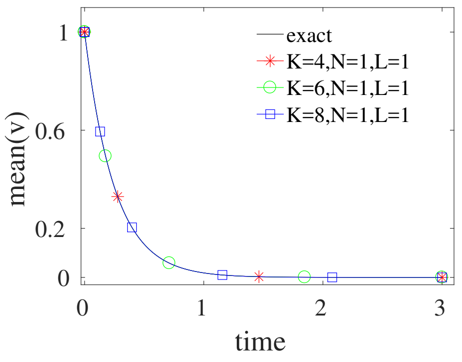
Next, we demonstrate numerical results for DgPC taking $N=1$, $L=1$ and a varying $K=4,6,8$. Figure 3.3 shows that as the exact solution converges to a steady state, our algorithm captures the second order statistics accurately even with a small degrees of freedom utilized in each subinterval. Moreover, although we do not show it here, it is possible, by increasing L, to approximate the higher order zero cumulants $\kappa_i, i=3,4,5,6$, with an error on the order of machine precision, which implies that the algorithm converges numerically to a Gaussian invariant measure.

In Figure 3.4, we illustrate an algebraic convergence of the variance in terms of the dimension K for $T=3$ and $T=15$. DgPC uses the same size of interval $\Delta t=0.2$ for both cases. For comparison, we also include the convergence behavior of Hermite PCE. Values of K are taken as $(2,4,6,8)$ for Figures 3.4a and 3.4c and $K=(8,16,32,64)$ for Figures 3.4b and 3.4d. We deduce that our algorithm maintains an algebraic convergence of order $O(K^{-3})$ for both $T=3$ and $T=15$ whereas the convergence behavior of Hermite PCE drops from $O(K^{-3})$ to $O(K^{-2})$ as time increases. This confirms the fact that the degrees of freedom required for standard PCE to maintain a desired accuracy should increase with time.

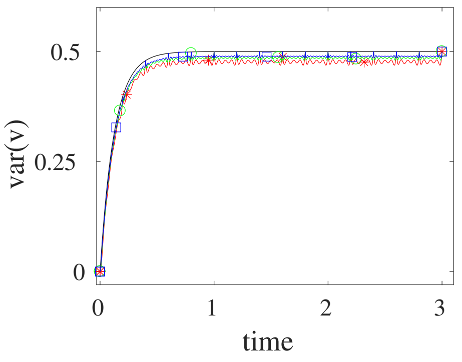
Example 3.11. We now introduce a nonlinearity in the equation so that the damping term includes a cubic component:

$$dv = -(v^2+1)v\,dt + \sigma_v\,dW_v, \quad v(0)=1. \tag{3.25}$$

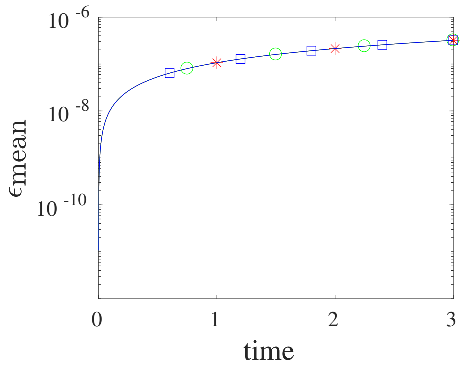
Figure 3.5 displays several numerical simulations for the degree of polynomials in $\boldsymbol{\xi}$ given by $N=1,2,3$ and $\sigma_v=2$. This is compared with the weak Runge–Kutta method using $M_{samp}=200000$ and $dt=0.001$. We observe that increasing N improves the accuracy of the solution to this nonlinear equation as expected. Moreover, Table 3.3 presents a comparison



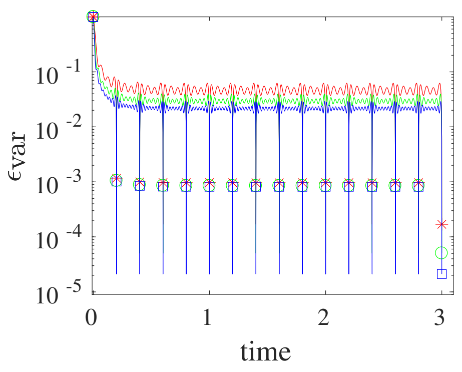
(a) mean



(b) variance



(c) ϵmean



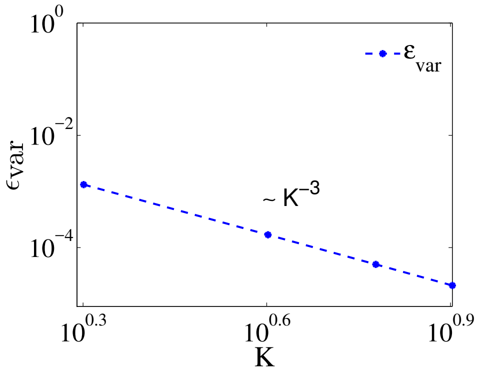
(d) ϵvar

Figure 3.3: DgPC with $\Delta t = 0.2$ and varying K for the Ornstein-Uhlenbeck process.

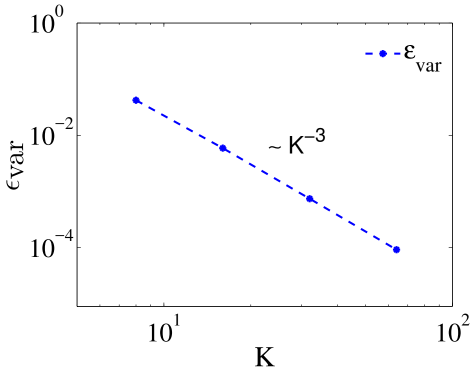
for statistical cumulants between our method with $K = 5$, $N = 2$, $L = 4$ and Monte Carlo at time $T = 4$. The (stationary) cumulants of the invariant measure are also estimated by solving a standard Fokker–Planck (FP) equation [90; 35] for the invariant measure. We conclude this example by noting that the level of accuracy of approximations in DgPC for cumulants is similar to that of the MC method.

Example 3.12. As a third example we consider an OU process (3.24) in which the damping parameter is random and uniformly distributed in $[1, 3]$, i.e. $b_v \sim U(1, 3)$. This is an example of non-Gaussian dynamics that may be seen as a coupled system for (v, b_v) with $db_v = 0$.

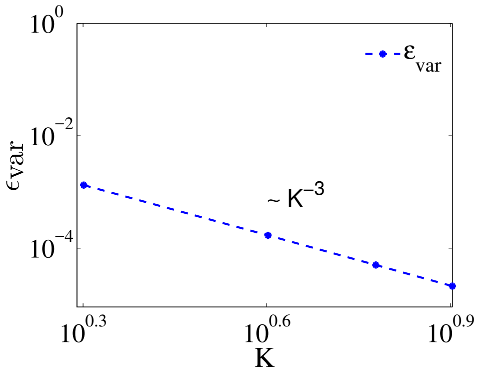
We consider a time domain $[0, 8]$ and divide it into $n = 40$ subintervals. The initial condition is normally distributed $v_0 \sim N(1, 0.04) \perp\!\!\!\perp W_v$ and $\sigma_v = 2$. In the next figure,



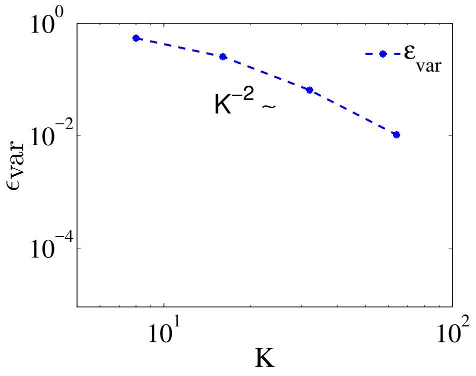
(a) DgPC at $T = 3$



(b) Hermite PC at $T = 3$



(c) DgPC at $T = 15$



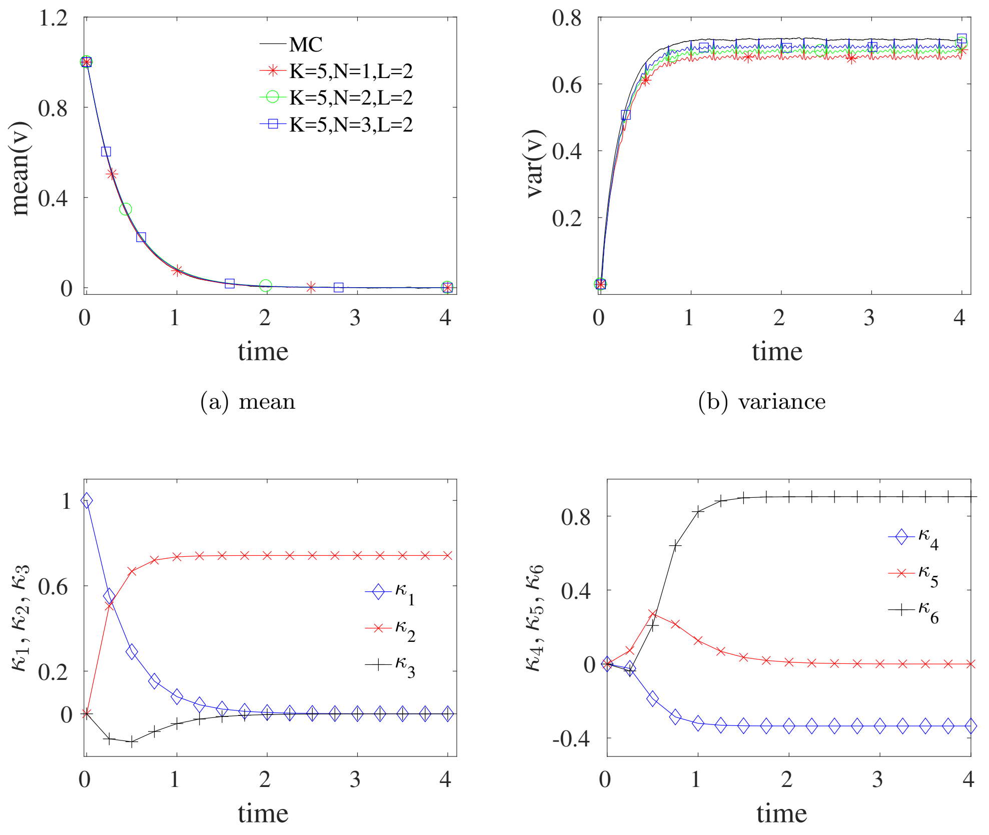
(d) Hermite PC at $T = 15$

Figure 3.4: Comparison of K-convergence of Hermite PC and DgPC at different times.

	κ_1	κ_2	κ_3	κ_4	κ_5	κ_6
DgPC	3.58E-5	7.33E-1	-2.08E-5	-3.37E-1	6.02E-5	9.35E-1
MC	-2.53E-3	7.33E-1	7.85E-4	-3.38E-1	-3.34E-3	9.58E-1
FP	0	7.33E-1	0	-3.39E-1	0	9.64E-1

Table 3.3: Cumulants at $T = 4$ obtained by three different methods.

we compare second order statistics obtained by our method to Monte Carlo method for which we use the Euler–Maruyama scheme with the time step $dt = 0.002$ and sampling rate $M_{samp} = 1000 \times 1000$ implying 10^6 samples in total. We stress again that this problem is essentially two-dimensional since the damping is random.



(a) mean

(b) variance

(c) cumulants

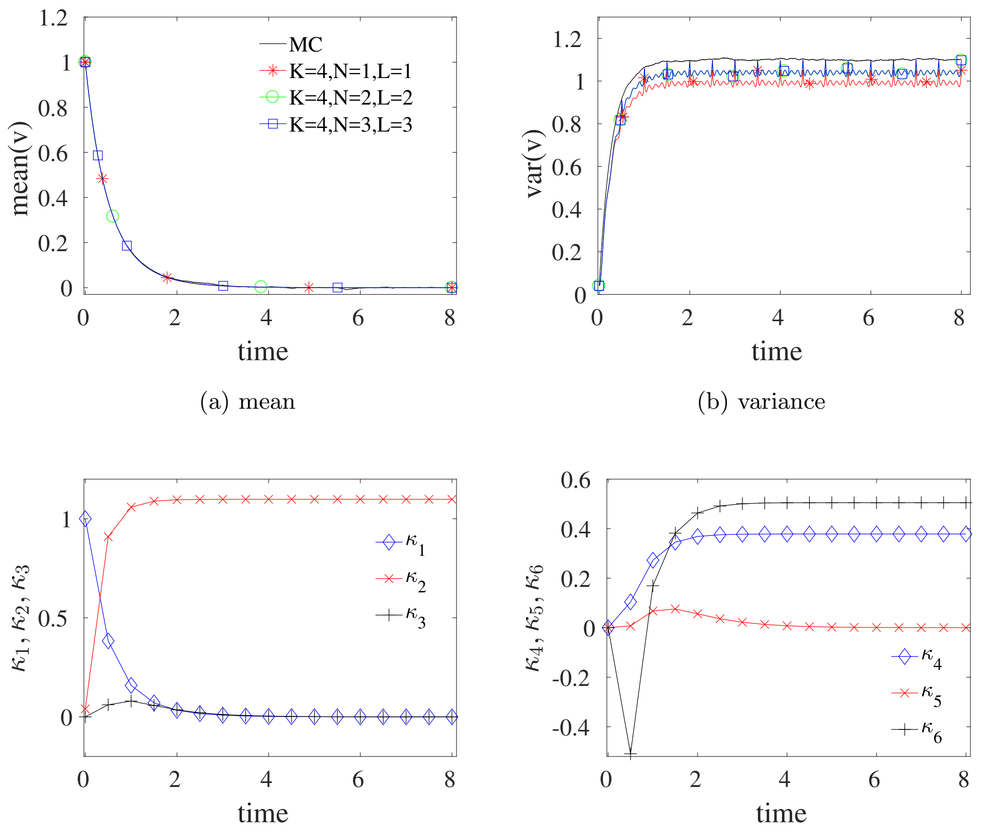
(d) cumulants

Figure 3.5: DgPC with $\Delta t = 0.25$ and increasing N for (3.25) with cubic nonlinearity.

As expected, the mean decreases monotonically and is approximated accurately by the algorithm. The estimation of the variance becomes more accurate as N increases. Furthermore, Figures 3.6c and 3.6d show that the cumulants become stationary for long times indicating that the numerical approximations converge to a measure which is non-Gaussian.

Table 3.4 compares the first six cumulants obtained by our algorithm at time $T = 8$ with $K = 4$, $N = 3$, $L = 4$ to the Monte Carlo method. For further comparison, we also provide cumulants obtained by averaging Fokker–Planck density with respect to the known, explicit, distribution of the damping. It can be observed from the following table that both our algorithm and Monte Carlo capture cumulants reasonably well although the accuracy degrades at higher orders.

The above calculations provide an example of stochasticity with two distinct compo-



(a) mean

(b) variance

(c) cumulants

(d) cumulants

Figure 3.6: DgPC with $\Delta t = 0.2$, and increasing N and L for the Ornstein-Uhlenbeck process with a uniformly distributed damping.

	κ_1	κ_2	κ_3	κ_4	κ_5	κ_6
DgPC	1.68E-5	1.10	3.80E-5	3.78E-1	8.72E-5	5.04E-1
MC	-1.78E-4	1.10	-3.25E-3	3.70E-1	-6.34E-2	4.65E-1
FP	0	1.10	0	3.79E-1	0	5.29E-1

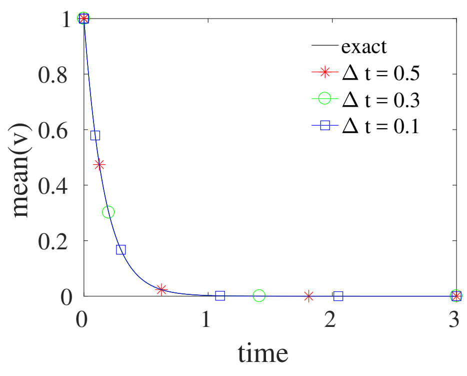
Table 3.4: Cumulants at $T = 8$ obtained by three different methods.

nents: the variables ξ that are Markov and can be projected out and the non-Markov variable b_v that has long-time effects on the dynamics. The variables (v, b_v) are strongly correlated for positive times. PCE thus need to involve orthogonal polynomials that reflect these correlations and cannot be written as tensorized products of orthogonal polynomials in v and b_v.

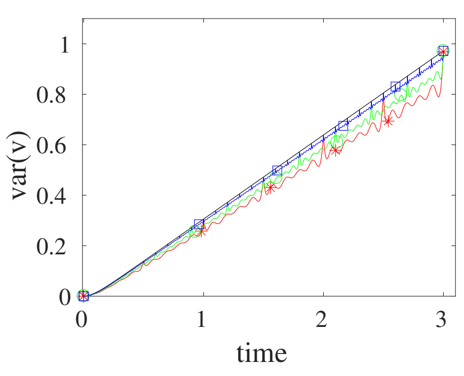
Example 3.13. We demonstrate that our method is not limited to equations forced by Brownian motion. We apply it to an equation where the forcing is a nonlinear function of Brownian motion:

$$dv(t) = -b_v\, v(t)\, dt + \sigma_v\, d(W^2(t) - t), \quad v(0) = 1,$$

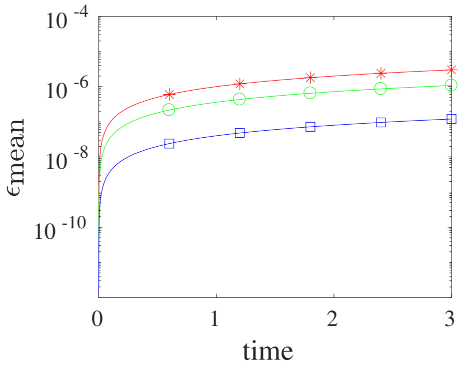
with parameters $b_v = 6$, $\sigma_v = 1$. To depict the effect of different choices of number of restarts, we take $\Delta t = 0.5, 0.3, 0.1$ using $K = 5$, $N = 2$, $L = 2$ in each expansion.



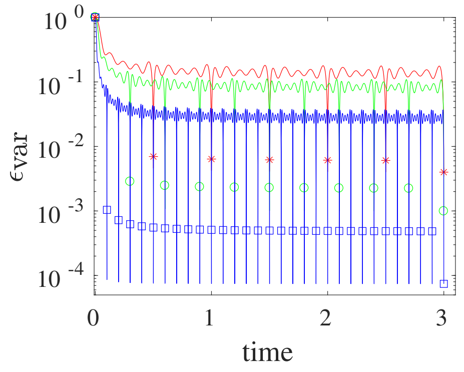
(a) mean



(b) variance



(c) ϵmean



(d) ϵvar

Figure 3.7: DgPC using $\Delta t = 0.5, 0.3$ and 0.1. The random forcing is a nonlinear function of Brownian motion.

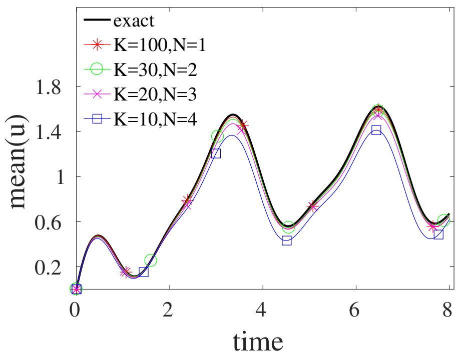
From Figure 3.7, we observe that second order statistics are captured accurately and an increasing number of restarts provides better approximations as anticipated. Although the system does not converge to a steady state (variance of $v(t)$ increases linearly), this example illustrates that DgPC is able to capture behaviors of solutions where the forcing is a nonlinear function of Wiener process.

Example 3.14. This example concerns the nonlinear system (3.21), where the dynamics of u exhibit intermittent non-Gaussian behavior. A time dependent deterministic periodic forcing is considered with the second equation being an OU process, i.e. $a_v = 0$ in (3.21), and the parameters are taken as $b_u = 1.4$, $b_v = 10$, $\sigma_u = 0.1$, $\sigma_v = 10$, $a_u = 1$, $f(t) = 1+1.1\cos(2t+1)+0.5\cos(4t)$, with initial conditions $u_0 = N(0, \sigma_u^2/8b_u) \perp\!\!\!\perp v_0 = N(0, \sigma_v^2/8b_v)$; see also [16]. The initial variables are independent of each other and of the stochastic forcing. Intermittency is introduced by using an OU process, which acts as a multiplicative noise fluctuating in the damping. The second order statistics for this case ($a_v = 0$) can be derived analytically [38] and we plot them in black in the following figures using quadrature methods with sufficiently high number of quadrature points.

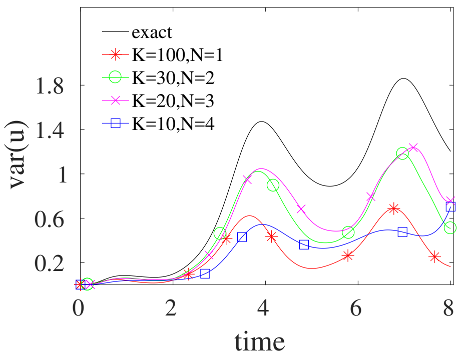
We first depict the results for u obtained by standard Hermite PCE in Figure 3.8. It can be observed that truncated PC approximations are accurate for short times. However, they quickly lose their accuracy as time grows. This is consistent with the simulations presented in [16]. Figure 3.9 shows that the accuracy is vastly improved in DgPC using $\Delta t = 0.1$, and second order statistics are accurate up to three or four digits. Moreover, errors in the variance oscillate around $O(10^{-3})$ for $L = 3$ throughout the time length suggesting that the approximation retains its accuracy in the long term, which is consistent with theoretical predictions. Our algorithm easily outperforms standard PCE in capturing the long-time behavior of the nonlinear coupled system (3.21) in this scenario.

Example 3.15. For this scenario, we consider the system parameters as $a_u = 1$, $a_v = 0$, $b_u = 1.2$, $b_v = 0.5$,$\sigma_u = 0.5$, $\sigma_v = 0.5$ without the deterministic forcing $f = 0$ and with the initials $u_0 = N(1, \sigma_u^2/8b_u) \perp\!\!\!\perp v_0 = N(0, \sigma_v^2/8b_v)$. In this regime, the dynamics of $u(s)$ are characterized by unstable bursts of large amplitude [16].

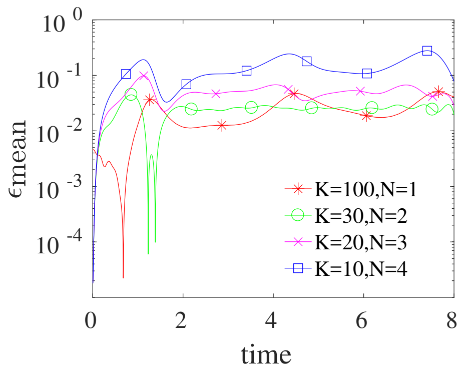
Second order statistics obtained by Hermite PCE are presented in Figure 3.10. We observe from Figure 3.10b and Figure 3.10d that the relative errors for standard Hermite expansions are unacceptably high. On the other hand, as the degrees of freedom is increased



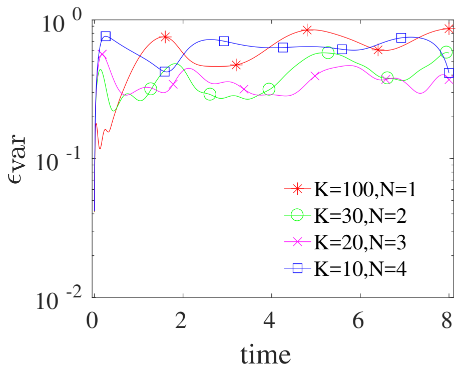
(a) Hermite PC mean(u)



(b) Hermite PC var(u)



(c) Hermite PC ϵ_{mean}

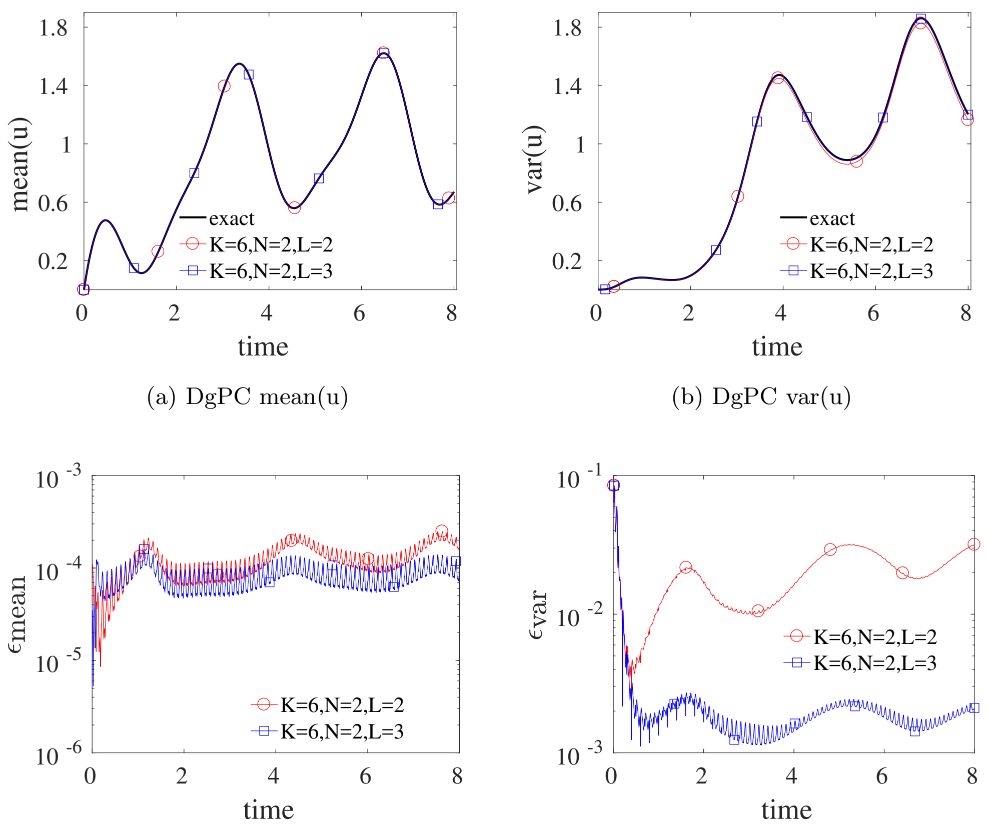


(d) Hermite PC ϵ_{var}

Figure 3.8: Hermite PC for the nonlinear system (3.21) with periodic deterministic forcing.

in DgPC, there is a clear pattern of convergence of second order statistics; see Figures 3.11c and 3.11d. Note, however, that short-time accuracy is better than accuracy for long times. This behavior may be explained by the onset of intermittency as nonlinear effects kick in after short times. Further, Figure 3.12 shows the error behavior of variance at $T = 10$ as N and L vary, and K is fixed. The rate of convergence is consistent with exponential convergence in N and L as the logarithmic plot is almost linear. For similar convergence behaviors, see [122; 75; 125].

Finally, in Figure 3.13, we present the evolution of the first few cumulants obtained by DgPC, which shows that the system converges to a steady state distribution as time increases. Bivariate cumulants are ordered so that first and second subscripts correspond to u and v, respectively. Figure 3.13f shows kurtosis excess for u, v and clearly indicates



(a) DgPC mean(u)

(b) DgPC var(u)

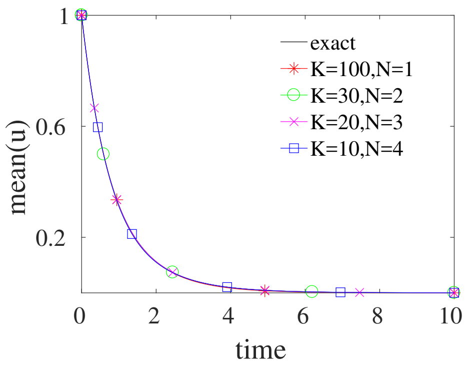
(c) DgPC ϵ_{mean}

(d) DgPC ϵ_{var}

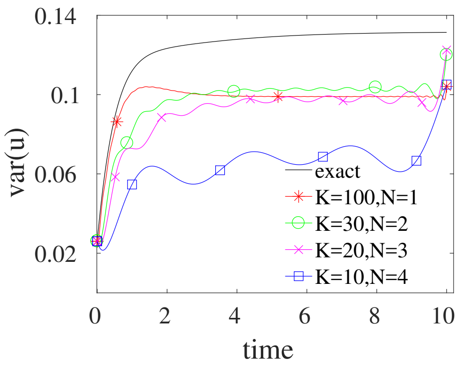
Figure 3.9: DgPC for the nonlinear system (3.21) with periodic deterministic forcing.

that the dynamics of v stay Gaussian, whereas those of u converge to a non-Gaussian state.

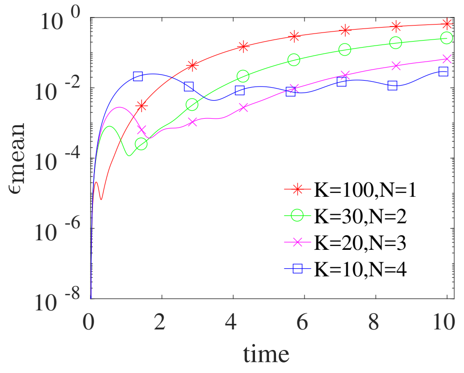
Example 3.16. As a final example, using the same set of parameters of the previous example, we introduce a small perturbation to the second equation and set $a_v = 0.03$ in (3.21). Another choice of perturbation as $a_v = 0.01$ was considered before in [16]. Comparing Figures 3.13 and 3.14, we see that evolutions of higher order cumulants are perturbed in most cases as expected. In this case, it took a longer transient time to converge to an invariant measure as additional nonlinearity was introduced into the system.



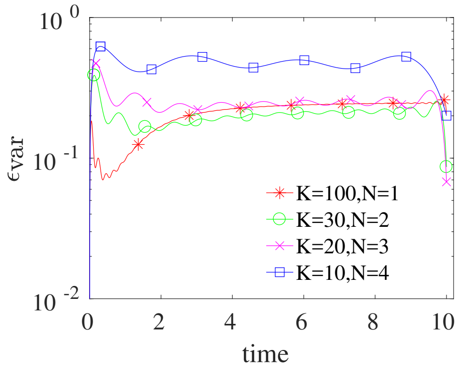
(a) Hermite PC mean(u)



(b) Hermite PC var(u)

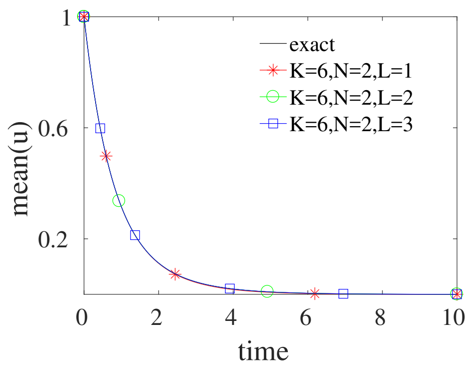


(c) Hermite PC ϵ_{mean}

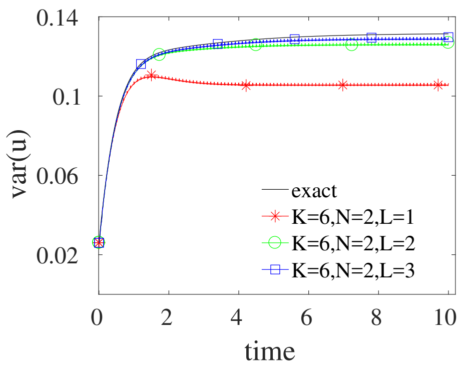


(d) Hermite PC ϵ_{var}

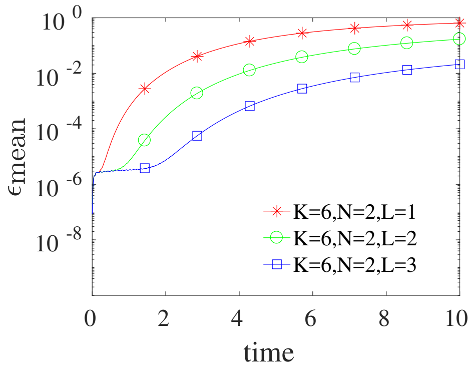
Figure 3.10: Hermite PC for the nonlinear system (3.21) with zero deterministic forcing.



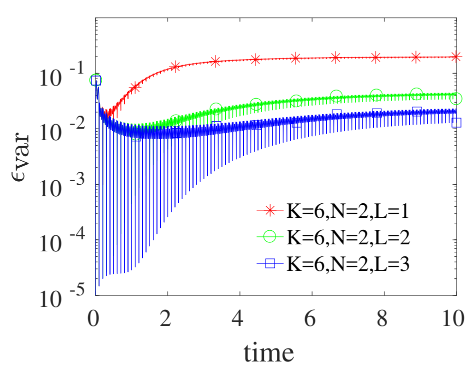
(a) DgPC mean(u)



(b) DgPC var(u)



(c) DgPC ϵ_{mean}



(d) DgPC ϵ_{var}

Figure 3.11: DgPC using $\Delta t = 0.1$ for the nonlinear system (3.21) with zero deterministic forcing.

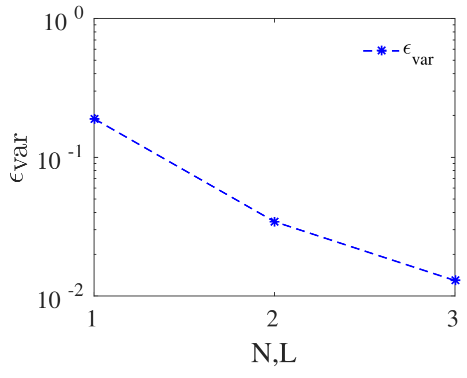


Figure 3.12: N- and L-convergence in the variance of DgPC at $T = 10$.

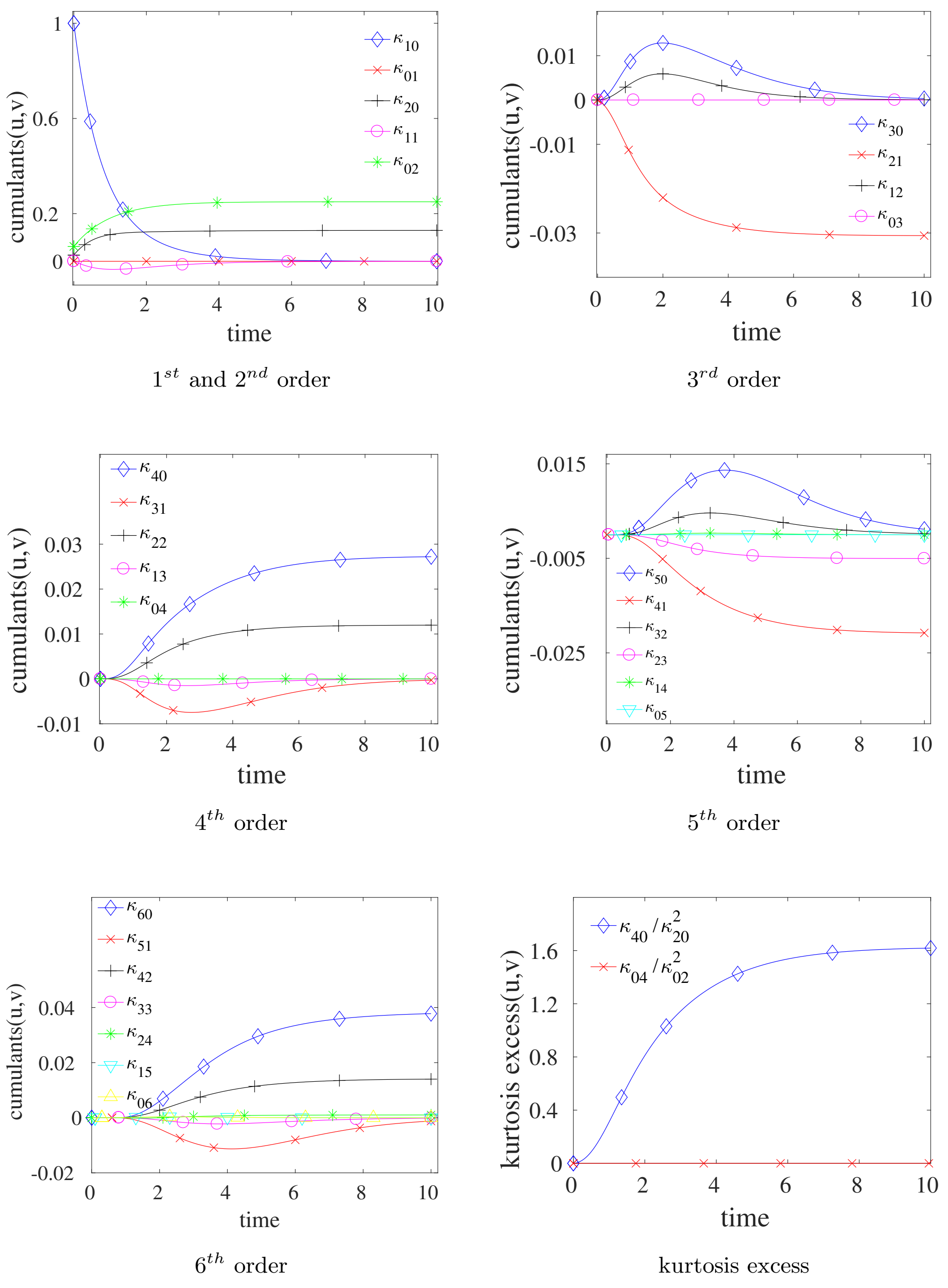
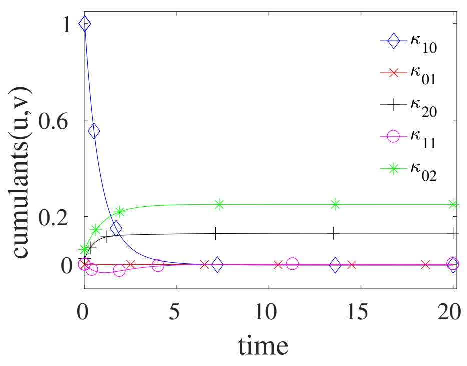
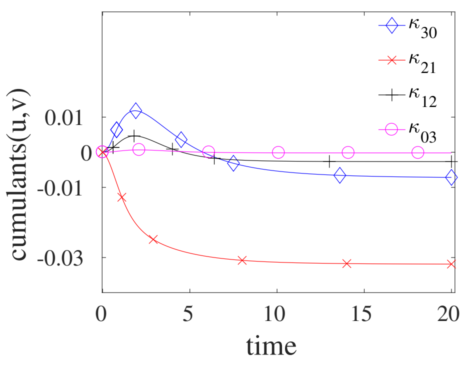


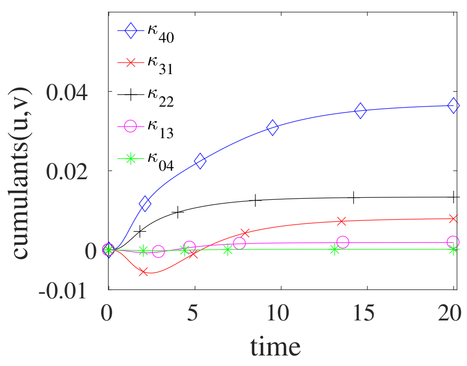
Figure 3.13: Bivariate cumulants and kurtosis excess obtained by DgPC for the nonlinear system (3.21) with zero deterministic forcing and $a_v = 0$.



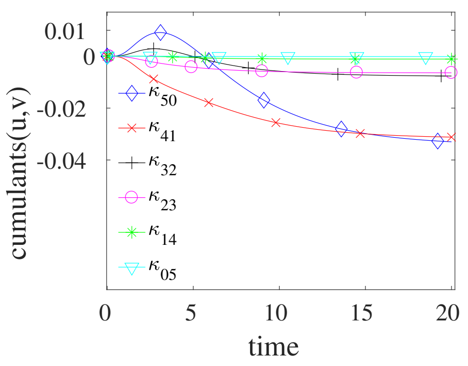
(a) 1^{st} and 2^{nd} order



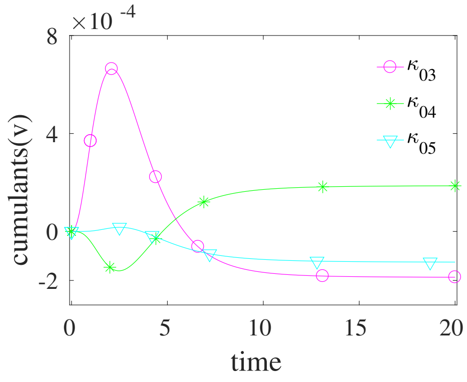
(b) 3^{rd} order



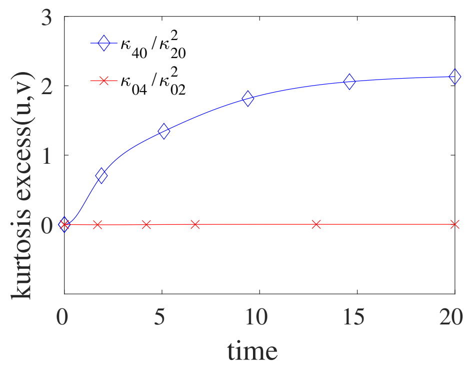
(c) 4^{th} order



(d) 5^{th} order



(e) cumulants of v



(f) kurtosis excess

Figure 3.14: Bivariate cumulants and kurtosis excess obtained by DgPC for the nonlinear system (3.21) with zero deterministic forcing and a perturbation $a_v = 0.03$.

Chapter 4

Dynamical gPC for SPDEs

4.1 Introduction

Although the DgPC algorithm performs well for low-dimensional SDEs, extension to larger systems is challenging and requires serious modifications. In this chapter, we extend the DgPC method to the framework of SPDEs driven by white noise.

The Karhunen–Loeve expansion is a popular technique to reduce the dimensionality in the random space. SPDE solutions are high-dimensional random fields and in some cases, they enjoy low-dimensional representations which can be provided by the KLE. However, these low-dimensional representations of solutions are time-dependent [105; 20; 19]. To provide optimal PCE representations in time, we propose to project the solution at each restart onto a lower dimensional manifold using its KLE. Since the KLE is known to be optimal in the mean square sense, at each restart point in time, only a few dominating, most energetic random modes are chosen and incorporated into PCE to represent the future solution. For equations with non-forcing random parameters, we apply the KLE to the combination of the solution and random parameters. Although the random variables for the random forcing can be forgotten, the effects of non-Markov time-independent random variables have to be incorporated into evolving PCE representations.

The combination of the random KLE modes and the random forcing variables brings about high dimensionality. The computational challenges then become: (i) computing orthogonal polynomials of arbitrary multivariate distributions; and (ii) keeping the number of terms in the expansion as small as possible.

The construction of orthogonal polynomials of evolving multivariate distributions is possible by estimating their statistical moments [36; 93], which is, in general, a computationally intensive procedure; see also the previous chapter. In this chapter, we estimate the moments of the solution using its PCE through a sampling approach to greatly reduce computational cost; see [4] for a similar sampling methodology. We also show that the method is robust with respect to re-sampling.

The second challenge (ii) is a major problem for all PC-based methods. The Karhunen-Loeve expansion is computationally expensive. For problems of moderate size, we find that the eigenvalue problem is solved efficiently by using a Krylov subspace method. For larger problems, in order to mitigate both memory requirements and computational cost of the KLE, we find low-rank approximations to the covariance matrices based on their PC representations and without assembling them. The algorithm leverages randomized projection methods as introduced in [80; 54], and uses appropriate random matrices to obtain fast and accurate solutions of large eigenvalue problems. After selecting the dominating modes in the KLE, we make use of the sparse truncation technique proposed in [57; 75] to further reduce the number of terms in a PCE. For long-time computations, we also develop an adaptive restart scheme, which adapts the time lag between restarts based on the nonlinear effects.

The use of compression techniques to exploit intrinsic lower dimensional structures of solutions of SPDEs is not new and is in fact necessary in many contexts; see [12; 28; 117; 105; 20; 19]. The novelty of our approach is that a lower dimensional representation of the solution is learned online and integrated into a PCE to integrate future random forcing and represent future solutions. This procedure is computationally viable and the combination of the aforementioned ideas allows us to attain a reasonable accuracy in the long-time evolutions of SPDEs for a reasonable computational cost.

As we are interested in the long-time evolution of SPDEs, we restrict ourselves here to dynamics with a dissipation mechanism. Equilibrium statistics and asymptotic properties of the solutions are relevant in many applications and have been extensively studied in the literature [92; 21; 109; 57; 52; 85; 16; 98]. Based on these motivations, we provide numerical experiments for a 1D randomly forced Burgers equation and a 2D stochastic Navier–Stokes (SNS) system. All equations are driven by white noise and satisfy periodic (spatial) boundary conditions. To demonstrate the efficiency of our algorithm, we present both short- and long-time computations. In some cases, we model the viscosity as a time-independent random process. Statistical moments obtained by the algorithm are compared to the standard MC methods for short times to assess the accuracy of the algorithm. Results show promising speed-ups over standard MC methods. We exhibit convergence behavior in terms of the degree of the expansion, the number of random modes retained in the KLE, and the (adaptive) restart time. Verifications of invariant measures in the long-time are also demonstrated in some cases.

4.2 Description of the Methodology

Throughout this chapter, we consider the following time-dependent stochastic partial differential equation driven by white noise $\dot{W}(t,\omega)$:

$$\begin{cases} \partial_t u(x,t,\omega) = \mathcal{L}(u(x,t,\omega)) + \sigma(x)\,\dot{W}(t,\omega), \quad x \in G \subset \mathbb{R}^d,\, \omega \in \Omega,\, t \in [0,T], \\ u(x,0,\omega) = u_0(x,\omega), \quad x \in G,\, \omega \in \Omega, \end{cases} \tag{4.1}$$

where, for concreteness, G is the d-dimensional torus so that u and its derivatives are periodic functions in the variable x. The DgPC algorithm may easily be extended to more general boundary conditions and geometries. Above, $\mathcal{L}$ is a possibly non-linear differential operator in the spatial variables. The solution takes values in $\mathbb{R}^p$. In the numerical simulations, the parameters d and p are set to either 1 or 2. We present the algorithmic details of the DgPC method applied to the general SPDE (4.1) in the following.

Let a decomposition $0 = t_0 < t_1 < \ldots < t_n = T$ be given. Following our discussion in section 3.2.1 and using the Markov property, the solution $u(x,t_{j+1},\omega)$, $0 \le j < n$, can be represented in a PC expansion in terms of $u(x,t_j,\omega)$ and $\boldsymbol{\xi}_j$, where $\boldsymbol{\xi}_j = (\xi_{j,1},\xi_{j,2},\ldots)$ denotes the Gaussian random variables required for Brownian forcing on the interval $[t_j,t_{j+1}]$. Let $u_j(x,t_j,\omega)$ denote the projection of the solution $u(x,t_j,\omega)$ onto the polynomial chaos space. We will also use the shorthand notation u_j to denote this projection. To construct a PCE in terms of polynomials of u_j, we separate the spatial dependence and randomness via the KLE (2.2):

$$u_j = \bar{u}_j(x) + \sum_{l=1}^{\infty} \sqrt{\lambda_{j,l}}\, \eta_{j,l}(\omega)\, \phi_{j,l}(x). \tag{4.2}$$

Let $\boldsymbol{\eta}_j := (\eta_{j,1}, \eta_{j,2}, \dots)$ denote the random modes. Since the solution $u(x,t,\omega)$, $t \in [t_j, t_{j+1}]$, is a functional of the random forcing $\boldsymbol{\xi}_j$ and modes $\boldsymbol{\eta}_j$, the next step PCE $u_{j+1}(x,t,\omega)$ is given by

$$u_{j+1}(x,t,\omega) = \sum_{\boldsymbol{\alpha}\in\mathcal{J}} u_{j+1,\boldsymbol{\alpha}}(x,t)\, T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_j(\omega), \boldsymbol{\eta}_j(\omega)), \quad t \in [t_j, t_{j+1}], \tag{4.3}$$

with the notation $u_{j+1}(x,t_j,\omega) = u_j$, where $T_{\boldsymbol{\alpha}}$ denotes an orthonormal basis in its arguments. The expansion dynamically needs a PC basis depending on the random forcing and evolving random KLE modes of the solution. The coefficients $u_{j+1,\boldsymbol{\alpha}}(x,t)$ satisfy a PDE system obtained by Galerkin projection of the SPDE (4.1) onto the space spanned by $T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_j, \boldsymbol{\eta}_j)$. Statistical properties can be retrieved after solving the induced PDE system.

Here are the computational bottlenecks of this approach: (i) the simple truncation (2.7) yields a large number of terms in the expansion, which leads to long computational times to solve the deterministic evolution equations; (ii) estimating the terms appearing in the KLE (4.2) is a major computational bottleneck, especially in higher spatial dimensions; and (iii) computation of the orthogonal polynomials $T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_j, \boldsymbol{\eta}_j)$ may also require intensive amount of computation. In the following, we address these issues in turn.

4.2.1 Sparse Truncation

The number of terms in the simple truncation (2.7) in the Hermite PCE increases rapidly with respect to N and K. Given sufficient regularity of the solution, the expansion coefficients decay both in the number of Gaussian variables K and the degree of polynomials N. This observation led the authors [57] to introduce a sparse truncation of the multi-index set and retain a truncated random basis, which keeps lower (higher) order polynomials in ξ_i with larger (smaller) subscripts. This truncation can be quantified using an estimate for the decay rate of the coefficients; see [75; 57; 129; 33; 13]. Following [75; 57; 33], we introduce a sparse index

$$r = (r_1, \dots, r_K), \quad N \geq r_1 \geq r_2 \geq \ldots \geq r_K,$$

and define the corresponding sparse multi-index set

$$J^r_{K,N} := \{\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K),\ |\boldsymbol{\alpha}| \le N, \alpha_k \le r_k\}. \tag{4.4}$$

Basically, the index r keeps track of how much degree we want in each variable ξ_i. Using (4.4), one can define the corresponding version of the PC expansion (2.5) which might have drastically reduced number of terms. This is possible by a suitable choice of the index r so that ineffective cross terms in high degree polynomials are eliminated.

4.2.2 Karhunen–Loeve Expansion

At each restart t_j, we employ the KLE (4.2) for the projected random field u_j, and the expansion is truncated after a finite number of D terms. The decomposition yields the eigenvalues $\lambda_{j,l}$ and the eigenfunctions $\boldsymbol{\eta}_j = (\eta_{j,l})$ for $l = 1, \ldots, D$. Therefore, in addition to the Gaussian variables $\boldsymbol{\xi}_j$, we have the random modes $\boldsymbol{\eta}_j$ in the PCE (4.3) so that the total number of random variables becomes $K + D$. To accommodate $\boldsymbol{\eta}_j$, we extend the multi-index set (4.4) to $\mathcal{J}^r_{K+D,N}$. This can also be done by the tensorization $\mathcal{J}^r_{K,N} \otimes \mathcal{J}^r_{D,N}$ of the multi-index sets since $\boldsymbol{\xi}_j$ and $\boldsymbol{\eta}_j$ are independent. However, since tensorization yields higher number of terms in the PCE for most values of K and D, it is not considered in the following.

Assuming that the orthonormal basis $T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_{j-1}, \boldsymbol{\eta}_{j-1})$ is constructed in the previous step, the solution $u(x, t_j, \omega)$ is approximated using the truncated PCE

$$u_j = \sum_{\boldsymbol{\alpha} \in \mathcal{J}^r_{K+D,N}} u_{j,\boldsymbol{\alpha}}(x)\, T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_{j-1}, \boldsymbol{\eta}_{j-1}), \tag{4.5}$$

where the time dependence of the coefficients $u_{j,\boldsymbol{\alpha}}(x)$ is omitted for brevity. To avoid confusion, we note that both the projection (4.3) and its truncation (4.5) will be denoted by u_j. Armed with this approximation, using the orthogonality of random bases, the covariance of u_j is easily estimated by

$$\mathrm{Cov}_{u_j}(x, y) = \sum_{\boldsymbol{\alpha} > 0} u_{j,\boldsymbol{\alpha}}(x)\, u_{j,\boldsymbol{\alpha}}(y)', \quad x, y \in G, \tag{4.6}$$

where $\boldsymbol{\alpha} \in \mathcal{J}^r_{K+D,N}$.

In practice, we consider a discretization of the spatial domain G with an even mesh parameter $M_x \in \mathbb{N}$ and grid points x_m, $m = 1, \ldots, M_x^d$. Denote by C the resulting covariance matrix. In general, we can use a spectral method, e.g., Fourier series in the case of periodic functions, to approximate the coefficients $u_{j,\boldsymbol{\alpha}}$ as

$$u_{j,\boldsymbol{\alpha}}(x) \approx \sum_{k=1}^{M_x^d} \hat{u}_{j,\boldsymbol{\alpha}}(k)\, \varphi_k(x), \tag{4.7}$$

where φ_k are orthogonal global basis functions on G and $\hat{u}_{j,\boldsymbol{\alpha}}(k) = \langle u_{j,\boldsymbol{\alpha}}, \varphi_k \rangle_{L^2(G)}$; see [55]. Thus, $u_{j,\boldsymbol{\alpha}}(x)$ is approximated by a vector $(u_{j,\boldsymbol{\alpha}}(x_m))_{m=1}^{M_x^d}$ on the grid. Therefore, the dimension of the covariance matrix C becomes of order $\mathcal{O}(M_x^d \times M_x^d)$.

After computing the covariance matrix, the corresponding eigenvalue problem can be solved for the first D largest eigenvalues. Then, the random modes $\boldsymbol{\eta}_j$ are given as

$$\eta_{j,l} = \frac{1}{\sqrt{\lambda_{j,l}}} \langle \, (u_j - \mathbb{E}[u_j]) \, , \phi_{j,l} \rangle_{L^2(G)} = \frac{1}{\sqrt{\lambda_{j,l}}} \sum_{\boldsymbol{\alpha}>0} \langle u_{j,\boldsymbol{\alpha}}, \phi_{j,l} \rangle_{L^2}\, T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_{j-1}, \boldsymbol{\eta}_{j-1}), \quad l = 1, \ldots, D. \tag{4.8}$$

This representation yields the random modes $\boldsymbol{\eta}_j$ as a function of $\boldsymbol{\xi}_{j-1}$ and the previous modes $\boldsymbol{\eta}_{j-1}$. Here, we assume that the integrals $\langle u_{j,\boldsymbol{\alpha}}, \phi_{j,l} \rangle_{L^2}$ can be computed by an accurate quadrature method.

Remark 4.1. When the solution u is more than one-dimensional, several implementations of the KLE can be considered. For instance, we may apply the KLE to each component of the solution u separately and incorporate the resulting individual random modes into one PCE. Although this approach certainly makes the KLE step faster, we found that it yielded inaccurate results in DgPC and needed a large number of variables in the PCE to represent the solution accurately since cross covariance structures between the components of the solution are lost. Therefore, in the following, we implement the KLE directly to the multi-dimensional solution and produce one set of random modes $\boldsymbol{\eta}$ which represents all of its components.

Depending on the resolution of the discretization of the domain G and the dimension d, assembling the covariance matrix and solving the corresponding eigenvalue problem may prove dauntingly expensive. Several methods have been devised to reduce computational costs, such as fast Fourier techniques (e.g., in the case of stationary covariance kernels) or sparse matrix approximations together with Krylov subspace solvers [107; 44; 29; 63].

Here, the assembly of the covariance matrix is performed at each restart via the summation formula (4.6). In our one dimensional simulations, with $d = 1$, this assembly can be carried out reasonably fast. Since the solution of the eigenvalue problem is required only for the number $D \ll M_x^d$ of eigenvalues and eigenfunctions, Krylov subspace methods [104] perform well. We utilize the implicitly restarted Arnoldi method to efficiently find the few largest eigenvalues and corresponding eigenfunctions; [3; 69].

In the higher dimensional simulations, when $d > 1$, computing and storing such large covariance matrices become challenging. Covariance matrices are, in general, not sparse and require $\mathcal{O}(M_x^{2d})$ units of storage in the memory. Moreover, assembling a large matrix at every restart is computationally very expensive for long-time simulations. The problem of computing and storing a large covariance matrix resulting from the KLE was addressed before in [107; 29; 63; 19]. It was noted that although the covariance matrices were dense, they were usually of low-rank; [107; 29]. Based on this observation, we next introduce an approximation approach, which leverages low-rank structures and avoids assembling large matrices.

A low-rank approximation $AB \approx C \in \mathbb{R}^{M_x^d \times M_x^d}$ tries to capture most of the action of the matrix C by a product of two low-rank matrices $A \in \mathbb{R}^{M_x^d \times l}$ and $B \in \mathbb{R}^{l \times M_x^d}$. Several efficient algorithms, e.g., the fast multipole method and $\mathcal{H}$-matrices, depend on low-rank approximations [49; 50]. We approximate eigenvalues and eigenvectors of the correlation matrix C by using low-rank approximations as follows.

Given a low rank approximation $Q(Q'C)$ of the symmetric covariance matrix C, where the matrix Q is of size $\mathbb{R}^{M_x^d \times l}$ with $l \geq D$ orthonormal columns, the eigenvalue problem of C can be approximated efficiently by applying the QR or SVD algorithm to the much smaller matrix $Q'CQ$. In the DgPC setting (4.6), this amounts to computing

$$\mathbb{R}^{l \times l} \ni Q'CQ = \sum_{\boldsymbol{\alpha} > 0} (Q'u_{\boldsymbol{\alpha}})\,(Q'u_{\boldsymbol{\alpha}})'. \tag{4.9}$$

The explicit assembly of the covariance matrix C is avoided by computing only the matrix-vector product $Q'u_\alpha \in \mathbb{R}^{l \times 1}$. An approximate eigenvalue decomposition $C \approx U \Lambda U'$ is deduced from the eigenvalue decomposition of the smaller matrix $Q'CQ = V\Lambda V'$ by setting $U = QV$.

The crucial step of the computation is the construction of a low-rank matrix Q with $l \ll M_x^d$ orthonormal columns that accurately describes C. We tried an approach based on the discrete unitary Fourier transform to map the coefficients u_α to the frequency space and retain only the lowest frequencies. Although this approach allowed us to obtain reasonable compressions of the covariance matrix and enabled faster computations, the following approach consistently yielded much better results.

Following [80; 54], we construct the matrix Q using random projections. Algorithm 4.1 in [54] draws an $M_x^d \times l$ Gaussian random matrix O and forms the matrix $Y = CO \in \mathbb{R}^{M_x^d \times l}$. The matrix Q with l orthonormal columns is then obtained by the QR factorization $Y = QR$. Note again that we do not assemble the matrix C. Rather, the matrix-matrix product CO is computed as in equation (4.9). Since we require the largest D eigenvalues, the target low-rank becomes D. As indicated in [54], we use an oversampling parameter p by setting $l = D + p$. Typical values of p are 5 or 10. Since eigenvalues decay rapidly in our applications, we found $p = 10$ to be accurate. With overwhelming probability, the spectral error $||C - QQ'C||_2$ is bounded within a small polynomial perturbation of the theoretical minimum, the $(l+1)^{th}$ singular value of C; for relevant theoretical details, see [54, Section 10].

In practice, we found this randomized approach to be highly accurate in our computational simulations. Moreover, since assembly of large covariance matrices are avoided, running times and memory requirements for the KLE in $\mathbb{R}^2$ are reduced drastically compared to the previously described methods; see section 4.3.2.

4.2.3 Additional Non-forcing Random Inputs

In this subsection, we consider the case in which the differential operator $\mathcal{L}$ in (4.1) contains additional random input parameters, i.e., $\mathcal{L} = \mathcal{L}(u(x,t,\omega),\omega)$. The random inputs will be denoted by the process $Z(x,\omega)$ of a dimension D_Z. A typical case is that of a random viscosity, e.g., depending on a set of uniformly distributed random variables. We assume that the process Z is independent of time and that the corresponding orthogonal polynomials are available; for instance in the Askey family [125].

We first observe that the solution $u(x,t,\omega)$ is now a functional of Brownian motion W and the random process Z. Therefore, assuming L^2 integrability, it can be written as a PCE in terms of the associated orthogonal polynomials of W and Z. At the restart t_j, there are two options to carry out the KLE: (i) apply the KLE to only the solution u_j and keep the basis variables for Z in addition to $\boldsymbol{\eta}_j$ in the next PCE; and (ii) compress the combined random variable (u_j, Z) using the KLE and denote by $\boldsymbol{\eta}_j$ the combined random modes which represent both u_j and Z. The first approach will yield PCE which provide functional representations of the solution in terms of W and Z for each time t_j. In the second approach, the functional dependence of the solution in terms of Z is lost in the first KLE step. However, the moments of the solution can still be computed through the combined random KLE modes. In many UQ settings, rather than a functional dependence, it is statistical information of the underlying solution, e.g., moments of the invariant measure in the long-time, that we are after. Moreover, the second approach can be seen as a dimensionality reduction technique, which compresses u_j and the process Z together, thereby further reducing the number of terms in PCE. When additional random parameters appear in the equation, we found it reasonable to implement the second approach to reduce cost while the first approach may be used as a reference computation to assess accuracy.

Remark 4.2. It is useful to note that by combining the random fields u_j and Z, the algorithm automatically chooses the important part of the random process Z that influences the solution while keeping the moments of the solution accurate; see section 4.3.

4.2.4 Moments and Orthogonal Polynomials

After obtaining the random modes $\boldsymbol{\eta}_j$, $j \in \mathbb{N} \cup \{0\}$, we need to construct the following orthonormal basis:

$$\{T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_j, \boldsymbol{\eta}_j) : |\boldsymbol{\alpha}| \leq N, \boldsymbol{\alpha} \in \mathcal{J}^r_{K+D,N}\}. \tag{4.10}$$

Notice that since $\boldsymbol{\xi}_j$ is Gaussian and identically distributed for each j, the corresponding orthonormal polynomials of $\boldsymbol{\xi}_j$ are known to be the Hermite polynomials for each j. However, the probability distribution of $\boldsymbol{\eta}_j$ is arbitrary and changes at each restart. Therefore, the computation of orthonormal polynomials is computationally intensive and can be performed using the Gram–Schmidt method as follows.

We note that the set (4.10) can be computed based on the knowledge of moments of the variables $\boldsymbol{\xi}_j$ and $\boldsymbol{\eta}_j$. Following [45; 4], we assemble the Gram matrix H^j with the entries

$$H^j_{kl} = \mathbb{E}[(\boldsymbol{\xi}_j, \boldsymbol{\eta}_j)^{\boldsymbol{\alpha}_k + \boldsymbol{\alpha}_l}], \quad \boldsymbol{\alpha}_k, \boldsymbol{\alpha}_l \in \mathcal{J}^r_{K+D,N}. \tag{4.11}$$

The matrix H^j is a $|\mathcal{J}^r_{K+D,N}|$-dimensional, square and symmetric matrix. For theoretical reasons, we assume that the moments up to $2N$ exist and the measure of $(\boldsymbol{\xi}_j, \boldsymbol{\eta}_j)$ is non-degenerate. Then, the Cholesky factorization is employed to H^j and the polynomials $T_{\boldsymbol{\alpha}_l}$ is found by inverting the resulting upper triangular matrix as

$$T_{\boldsymbol{\alpha}_l}(\boldsymbol{\xi}_j, \boldsymbol{\eta}_j) = \sum_{\boldsymbol{\alpha}_k \leq \boldsymbol{\alpha}_l} a_{kl}\, (\boldsymbol{\xi}_j, \boldsymbol{\eta}_j)^{\boldsymbol{\alpha}_k}, \tag{4.12}$$

where a_{kl} are real coefficients.

Remark 4.3. The KLE yields uncorrelated random variables $\boldsymbol{\eta}_j$. If the underlying process is Gaussian, it is known that these variables are also independent. However, in general, marginals of $\boldsymbol{\eta}_j$ are dependent variables. Multi-index operations can still be used to construct the polynomial set (4.10) with respect to the joint distribution, although the estimation of multivariate moments of $\boldsymbol{\eta}_j$ becomes necessary because of such a dependency. In this case, it is known that orthogonal polynomials are not unique and depend on the ordering imposed on the multi-index set; see [130; 4; 30]. In all computations, we use the graded lexicographic ordering for multi-indices; see Table 2.1.

Remark 4.4. The completeness of the orthogonal polynomials $T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_j, \boldsymbol{\eta}_j)$ is closely related to the moment problem of the random variables $\boldsymbol{\xi}_j$ and $\boldsymbol{\eta}_j$. In particular, if the moment problem is uniquely solvable, i.e., the measure is determinate, then the orthogonal polynomials are dense in L^2 [11; 97; 36; 30]; see also the previous chapter. Some basic conditions that guarantee determinacy of the measure of a continuous random variable on a finite dimensional space are compact support and exponential integrability. Gaussian measures are determinate and the Hermite PCE converges by the Cameron-Martin theorem. However, in general, whether the distribution of $\boldsymbol{\eta}_j$ is determinate or not is unknown. This problem is addressed in the previous chapter in the case of finite dimensional SDE systems; see section 3.2.4. Theoretical results are applied in the setting where the solutions are approximated by compactly supported distributions under appropriate assumptions. In the following, we assume that the measures associated to $\boldsymbol{\eta}_j$ are determinate so that convergence is guaranteed, which is consistent with our numerical simulations; see section 4.3.

Remark 4.5. A quick summary of the measures and corresponding random variables appearing in this chapter is as follows. The solution to the SPDE with white noise forcing and possibly random coefficients generates a time-dependent measure on an abstract infinite dimensional probability space. A finite dimensional approximation is obtained first by spatial discretization and second, at each restart of the algorithm, KLE. This gives rise to a still complicated joint distribution of $\boldsymbol{\eta}_j \in \mathbb{R}^D$, which we never explicitly construct. As indicated in the preceding remarks, we assume that this measure is determinate or well approximated by a determinate measure so that its orthogonal polynomials are dense in the L^2 sense. What the algorithm propagates is a finite dimensional set of (approximately) orthogonal polynomials for this measure, with the assumption that in the limit of infinite order polynomials, infinite KLE, and vanishing spatial discretization mesh, such a set would characterize the SPDE solution. If an explicit, approximate construction of the distribution of $\boldsymbol{\eta}_j \in \mathbb{R}^D$ is needed at any particular time, a possible way to do so would be by means of moment-constrained maximum entropy methods such as those described in [1] and references therein.

Based on the above discussion, the orthonormal basis (4.10) requires the computation of the moments (4.11). The exact moments of the Gaussian variables $\boldsymbol{\xi}_j$ are computed by analytical formulas and then stored during the offline stage. However, the distribution of $\boldsymbol{\eta}_j$ is varying with j. Therefore, the computation of moments should be carried out based on information provided by the PCE.

Several methods are available to compute moments of probability distributions in the PC context such as, e.g., quadrature methods, Monte Carlo sampling, or a pure PCE approach. This procedure is notoriously expensive and ill-posed [36]. The pure PCE approach computes the moments of $\boldsymbol{\eta}_j$ by repeatedly multiplying its PCE and taking expectation; see [26]. This approach is discussed in detail in the preceding chapter and works reasonably well for a low dimensional SDE systems. However, it becomes prohibitively expensive if the dimension of the random variables in the PC basis is even moderate; see section 3.2.3. Therefore, in this chapter, we consider an alternative approach using Monte Carlo sampling, which drastically reduces the computational cost for computing moments compared to the pure PCE approach.

We assume that independent samples of the initial condition (therefore, the samples of $\boldsymbol{\eta}_0$) are provided so that the algorithm can be initialized. To construct the set (4.10), based on (4.11), we need to compute the first $2N$ moments of the joint random variable $(\boldsymbol{\xi}_j, \boldsymbol{\eta}_j)$. Moreover, since the triple products will be required for the evolution of the PCE coefficients, the first $3N$ moments need to be computed as well; see section 4.2.5. Using the same ordering of the multi-index set $\mathcal{J}^r_{K+D,3N}$, we require the computation of the following moments:

$$\mathbb{E}[(\boldsymbol{\xi}_j, \boldsymbol{\eta}_j)^{\boldsymbol{\alpha}}] = \mathbb{E}[\boldsymbol{\xi}_j^{(\alpha_1,\dots,\alpha_K)}]\,\mathbb{E}[\boldsymbol{\eta}_j^{(\alpha_{K+1},\dots,\alpha_{K+D})}], \quad \boldsymbol{\alpha} \in \mathcal{J}^r_{K+D,3N}, \tag{4.13}$$

where we used the independence of $\boldsymbol{\xi}_j$ and $\boldsymbol{\eta}_j$.

Let $\boldsymbol{\eta}_j(\omega_i) := (\eta_{j,1}(\omega_i), \dots, \eta_{j,D}(\omega_i))$ denote independent samples of the random modes for $\omega_i \in \Omega$, where $i = 1, \dots, S \in \mathbb{N}$. Then, provided the samples $\boldsymbol{\eta}_j(\omega_i)$ are given, the moments of $\boldsymbol{\eta}_j$ can be approximated by

$$\mathbb{E}[\boldsymbol{\eta}_j^{(\alpha_{K+1},\dots,\alpha_{K+D})}] \approx \frac{1}{S}\sum_{i=1}^{S}(\boldsymbol{\eta}_j(\omega_i))^{(\alpha_{K+1},\dots,\alpha_{K+D})} = \frac{1}{S}\sum_{i=1}^{S}\prod_{l=1}^{D}(\eta_{j,l}(\omega_i))^{\alpha_{K+l}},$$

where we used the usual multi-index notation for powers. Therefore, multivariate moments (4.13) are computed by a combination of the analytical formulas for $\boldsymbol{\xi}_j$ and a sampling approximation for $\boldsymbol{\eta}_j$. Note that in applications, we use small values of N with a sufficiently large number of samples S to guarantee accuracy.

Although we discussed computing moments based on samples, we have not explained how the samples of $\boldsymbol{\eta}_j$ are acquired except for $\boldsymbol{\eta}_0$. The distribution of $\boldsymbol{\eta}_j$, $j \geq 1$, is evolving in time. However, owing to the PCE (4.8) of the each component $\eta_{j,l}$, we can write

$$\boldsymbol{\eta}_j = \sum_{\boldsymbol{\alpha}} \boldsymbol{\eta}_{j,\boldsymbol{\alpha}} T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_{j-1}, \boldsymbol{\eta}_{j-1}), \quad j \in \mathbb{N}.$$

This representation gives a natural way to sample from the distribution of $\boldsymbol{\eta}_j$ by the recursion

$$\boldsymbol{\eta}_j(\omega_i) = \sum_{\boldsymbol{\alpha}} \boldsymbol{\eta}_{j,\boldsymbol{\alpha}} T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_{j-1}(\omega_i), \boldsymbol{\eta}_{j-1}(\omega_i)), \quad i = 1, \ldots, S, \tag{4.14}$$

assuming that we obtained samples of $\boldsymbol{\eta}_{j-1}(\omega_i)$ at the previous restart t_{j-1}. Independent samples $\boldsymbol{\xi}_{j-1}(\omega_i)$ are obtained through sampling a multivariate Gaussian distribution. Therefore, on the subinterval $[t_{j-1}, t_j]$, PCE acts like a transport map which maps previously obtained samples of $\boldsymbol{\eta}_{j-1}$ and new samples of $\boldsymbol{\xi}_{j-1}$ to the samples of the new random modes $\boldsymbol{\eta}_j$.

Remark 4.6. Note that $\boldsymbol{\eta}_j$ is a function of the variables $\boldsymbol{\xi}_{j-1}$ and $\boldsymbol{\eta}_{j-1}$. Therefore, the number of samples of $\boldsymbol{\eta}_j$ should be ideally S^2 provided the same of number of samples S is used for each $\boldsymbol{\xi}_{j-1}$ and $\boldsymbol{\eta}_{j-1}$. However, in practice, this is not feasible in our method as the number of samples grows in time. Instead, the equation (4.14) keeps only the diagonal terms in the sample space. In the numerical simulations, we use large values of S so that the loss of accuracy incurred from discarding some samples would be minimal. Moreover, it is important to notice that (4.14) entails samples from the joint distribution of $\boldsymbol{\eta}_j$ so that Monte Carlo method is used to approximate the expectations while preserving the dependence structure of marginals.

Remark 4.7. The method blends Monte Carlo sampling into a PC approach to exploit the virtues of the both methods, namely, rapid computation of expectations and spectral accuracy provided by the MC and PC methods, respectively. Although the method is utilizing samples for the computation of moments, samples are not used in the evolution stage. The algorithm essentially propagates moments of the measures between successive times, where moments are computed using a sampling technique. To test the robustness of the method with respect to sampling, imagine that the algorithm starts with an infinite supply of independent samples of $\boldsymbol{\eta}_0$. We discard the first sample after using it to construct the corresponding orthogonal polynomials at the end of the first time interval and propagate the remaining samples with the PCE map to construct (an infinite supply of) samples of $\boldsymbol{\eta}_1$. The algorithm iteratively estimates the distribution, hence the moments, of each $\boldsymbol{\eta}_j$ while samples are discarded at each restart. We tested this idea by starting with a set of

n independent samples of $\boldsymbol{\eta}_0$ and propagated them by PCE for a maximum of n restarts. We found that the accuracy of the calculations was not affected by such a re-sampling tool; see numerical Example 4.8. This comparison showed the robustness of the algorithm under changes of samples. Since in practice, such re-sampling increases the computational costs compared to (4.14), it is not considered in the numerical experiments presented in the next sections. We also emphasize that the sampling approach readily returns samples of the approximated solution at the endpoint T through its KLE without a further sampling procedure. These samples can also be useful in uncertainty quantification to estimate further statistical properties of the solution such as probabilities on prescribed sets or probability density functions.

4.2.5 Galerkin Projection and Local Initial Conditions

Once an orthonormal basis is obtained, the algorithm performs a Galerkin projection onto the space spanned by the basis, and this requires the computation of triple products. Using the representation (4.12), the required triple products can be written as

$$\mathbb{E}[T_{\boldsymbol{\alpha}_k} T_{\boldsymbol{\alpha}_l} T_{\boldsymbol{\alpha}_m}] = \sum_{\boldsymbol{\alpha}_{k'} \leq \boldsymbol{\alpha}_k} \sum_{\boldsymbol{\alpha}_{l'} \leq \boldsymbol{\alpha}_l} \sum_{\boldsymbol{\alpha}_{m'} \leq \boldsymbol{\alpha}_m} a_{k'k}\, a_{l'l}\, a_{m'm}\, \mathbb{E}[(\boldsymbol{\xi}_j, \boldsymbol{\eta}_j)^{\boldsymbol{\alpha}_{k'} + \boldsymbol{\alpha}_{l'} + \boldsymbol{\alpha}_{m'}}], \tag{4.15}$$

where all multi-indices belong to the set $\mathcal{J}^r_{K+D,N}$. Thus this formula can be computed by the knowledge of moments of order up to $3N$.

Depending on the choice of the sparse index r, the multi-index $\boldsymbol{\alpha}_{k'}+\boldsymbol{\alpha}_{l'}+\boldsymbol{\alpha}_{m'} \in \mathcal{J}_{K+D,3N}$ might not be an element of $\mathcal{J}^r_{K+D,3N}$. Therefore, once we fix the multi-index set $\mathcal{J}^r_{K+D,N}$ in the offline stage, we also compute and store the indices that are elements of $\mathcal{J}^r_{K+D,3N}$.

Finally, we perform Galerkin projection of the SPDE (4.1) and obtain the following PDE system for the coefficients $u_{j+1,\boldsymbol{\alpha}}(x,t)$ of (4.3), $t \in [t_j, t_{j+1}]$:

$$\partial_t(u_{j+1,\boldsymbol{\alpha}}) = \mathbb{E}\left[T_\alpha(\boldsymbol{\xi}_j, \boldsymbol{\eta}_j)\, \mathcal{L}\left(\sum_{\boldsymbol{\beta}} u_{j+1,\boldsymbol{\beta}}\, T_{\boldsymbol{\beta}}(\boldsymbol{\xi}_j, \boldsymbol{\eta}_j)\right)\right] + \sigma(x)\, \mathbb{E}[T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_j, \boldsymbol{\eta}_j)\dot{W}(t)]. \tag{4.16}$$

The first expectation in the above line is computed with the aid of the triple products (4.15) and the second using the representation (2.10).

Note that the initial conditions $u_{j+1,\boldsymbol{\alpha}}(x,t_j)$ can be obtained by noticing that the representation (4.2) of u_j is nothing but a sum involving linear polynomials in $\eta_{j,l}$. It can therefore be rewritten in the basis $T_\alpha(\boldsymbol{\xi}_j, \boldsymbol{\eta}_j)$ with the help of Galerkin projections. Hence, the only coefficients that survive in (4.3) at t_j are the mean and the ones which correspond to the first degree polynomials in $\boldsymbol{\eta}_j$. Then, the PDE system (4.16) can be solved in time using a time-integration method combined with the aforementioned spectral method (4.7). If the initial condition u_0 of (4.1) is deterministic, we employ the Hermite PCE on the first interval $[0, t_1]$, which does not necessitate the computation of the KLE.

4.2.6 Adaptive Restart Scheme

So far, the method uses a predetermined restart time Δt. For long-term simulations, an adaptive restart scheme that sets the restarts online depending on the properties of the solution can reduce the computational cost.

We propose to adapt the restart time based on the following two criteria: (i) preserve the accuracy of the representation (2.10) of the random forcing; and (ii) mitigate the effect the nonlinearities in the accuracy of the polynomial expansions. For a prescribed number of dimensions to describe the random forcing, the algorithm can not take too large steps to preserve accuracy. Also, nonlinearities force the solution to be less accurately described by low-degree polynomials in the initial condition as time increases. In both cases, we wish Δt to be as large as possible for a given accuracy in mind.

To this end, let Δt_j denote the adaptive time-step starting from time t_j. To ensure an accurate representation of the forcing term, we set a maximum value $\Delta t_{\max}$ for Δt_j for all j, i.e. $\Delta t_j \le \Delta t_{\max}$. In practice, $\Delta t_{\max}$ is based on the error analysis of random forcing by a finite dimensional approximation; see for instance [57]. To address nonlinearities, we consider the following ratio for the PC coefficients $u_{j,\boldsymbol{\alpha}}$:

$$\rho(t) := \frac{|| \sum_{|\boldsymbol{\alpha}|>1} u_{j,\boldsymbol{\alpha}}^2(\cdot, t)||_{L^1}}{|| \sum_{|\boldsymbol{\alpha}|>0} u_{j,\boldsymbol{\alpha}}^2(\cdot, t)||_{L^1}}, \quad t \in [t_{j-1}, t_{j-1} + \Delta t_{j-1}]. \tag{4.17}$$

The condition measures the norm ratio of the nonlinear terms in the variance to the norm of the variance. In applications, the ratio is computed at each time integration point. Similar other conditions were used in different settings in [37; 56].

Consider a threshold value $\epsilon \in (0, 1)$. We propose the following conditions for adaptive time-steps using $t \in [t_{j-1}, t_{j-1} + \Delta t_{j-1}]$:

i) if $\rho(t) \leq 3\epsilon$ then set the next time-step $\Delta t_j = \min(t_* - t_{j-1}, \Delta t_{\max})$ for $\rho_p(t_*) = 2\epsilon$,

ii) if $\rho(t) > 3\epsilon$ then go back to t_{j-1} and set $\Delta t_{j-1} = \min(t_* - t_{j-1}, \Delta t_{\max})$ for $\rho(t_*) = 2\epsilon$,

where ρ_p is a polynomial approximation to $\rho(t)$, which can be found by fitting a p-degree polynomial to $\rho(t)$ on the interval $[t_{j-1}, t_{j-1} + \Delta t_{j-1}]$. This approximation is only required for time values t satisfying $\rho(t) < 2\epsilon$.

The time-steps Δt_j, $j > 0$ are set adaptively. For short time-steps, we do not expect dynamics to change drastically between successive intervals. Therefore, condition i) verifies whether the ratio is smaller than 3ϵ on the current interval, and then sets the adaptive time-step for the next interval. When $\rho(t) \leq \epsilon$, then the algorithm selects a bigger time-step by finding the root of $\rho_p(t_*) = 2\epsilon$. Note that the current evolution on the interval $[t_{j-1}, t_{j-1} + \Delta t_{j-1}]$ is not prematurely stopped at the end point. Although PCEs converge at any point inside the interval, errors, however, are known to wildly oscillate inside the interval and become spectrally accurate only at the end point; see the previous chapter and [16]. Condition ii) essentially verifies whether the ratio becomes too large (i.e. $> 3\epsilon$), and when this happens, forces the evolution to restart from the current initial point t_{j-1}. This control ensures that the algorithm does not take too large steps.

Our procedure is summarized in Algorithm 2, where, for simplicity, we only present the version which uses a predetermined number of restarts.

4.3 Numerical Simulations

We now present numerical simulations for the Burgers equation in one spatial dimension and a Navier–Stokes system in two spatial dimensions both driven by white noise. We consider these equations for two reasons. First, the statistical behavior of solutions of these equations is of importance in statistical mechanics and turbulence theory; see, e.g., [41; 84; 52; 53]. Second, they serve as challenging test beds for the PCE methodology.

We illustrate convergence results in terms of the degrees of freedoms D, N, and the time-step Δt, and consider both short-time and long-time evolutions. The convergence of

Algorithm 2 Dynamical generalized Polynomial Chaos (DgPC) for SPDEs

```
Decompose the time domain $[0,T] = [0,t_1] \cup \ldots \cup [t_{n-1},T]$
Initialize the degrees of freedom $K, N, D, S$
Choose the sparse index $r = (r_1, \ldots, r_{K+D})$
Compute the indices used in the triple-product formula (4.15)
Compute moments of $\boldsymbol{\xi}_0$
for each time-step $t_j \geq 0$ do
    apply the KLE to $u_j$ and obtain $\boldsymbol{\eta}_j = (\eta_{j,1}, \ldots, \eta_{j,D})$
    compute the moments $\mathbb{E}[(\boldsymbol{\xi}_j, \boldsymbol{\eta}_j)^{\boldsymbol{\alpha}}]$
    construct orthogonal polynomials $T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_j, \boldsymbol{\eta}_j)$
    compute the associated triple products
    perform Galerkin projection onto span$\{T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_j, \boldsymbol{\eta}_j)\}$
    set up initial conditions for $u_{j+1}$
    evolve the PCE coefficients of $u_{j+1}$
end for
```

the method in terms of K is treated in detail in Chapter 3. As a general comment, we do not recommend using large values of N since the computation of orthogonal polynomials is quite ill-posed. In the following, we use polynomials of degree up to $N = 3$. The algorithm mitigates the ill-posedness by choosing frequent restarts and small degree; i.e. small Δt and N. The settings we consider here closely follow those addressed in the manuscripts [57; 75].

4.3.1 Burgers Equation

We consider the following one dimensional viscous stochastic Burgers equation for $u(x,t,\omega)$:

$$\begin{cases} \partial_t u + \frac{1}{2}\partial_x u^2 = \nu\, \partial_x^2 u + \sigma(x)\dot{W}(t), \\ u(x,0,\cdot) = u_0(x), \quad u(0,t,\cdot) = u(1,t,\cdot), \quad (x,t) \in [0,1] \times [0,T], \end{cases} \tag{4.18}$$

where $W(t)$ is a Brownian motion in time, $\nu > 0$ is the viscosity (which will be either deterministic or random), the initial condition u_0 is deterministic, and the solution itself is periodic in the spatial domain $[0,1]$.

Following [75; 57], we choose cosine functions as an orthonormal basis $m_{j,k}(t), k \in \mathbb{N}$, for $L^2[t_j, t_{j+1}]$. Employing the equation (2.10) for each subinterval $[t_j, t_{j+1}]$ and using Galerkin projection, we derive the governing equations for the PC coefficients $u_{j+1,\boldsymbol{\alpha}}(x,t)$ of u_{j+1}:

$$\partial_t(u_{j+1,\boldsymbol{\alpha}}) + \frac{1}{2}\partial_x\,(u_{j+1}^2)_{\boldsymbol{\alpha}} = \nu\,\partial_x^2\,u_{j+1,\boldsymbol{\alpha}} + \sigma(x)\sum_{k=1}^{K} m_{j,k}(t)\,\mathbb{E}[\xi_{j,k}\,T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_j, \boldsymbol{\eta}_j)],$$

where $\boldsymbol{\xi}_j = (\xi_{j,1}, \xi_{j,2}, \ldots)$. Since the initial condition is deterministic, we employ Hermite PCE in the subinterval $[t_0, t_1]$. The PC coefficients $(u_{j+1}^2)_{\boldsymbol{\alpha}}$ of the nonlinearity u_{j+1}^2 are computed by multiplying the corresponding PCEs with the help of pre-computed triple products.

Since the coefficients $u_{j+1,\boldsymbol{\alpha}}(x,t)$ are periodic in the physical space, we utilize a truncated Fourier series:

$$u_{j+1,\boldsymbol{\alpha}}(x,t) \approx \sum_{k=-M_x/2+1}^{M_x/2} \hat{u}_{j+1,\boldsymbol{\alpha}}(k,t)\,e^{2\pi\,i\,k\,x}, \quad x \in [0,1],$$

with the even number of frequencies M_x to be chosen. Further, using the equidistant partition for the spatial domain $[0,1]$ and Fast Fourier Transform (FFT), we can compute the Fourier coefficients $\hat{u}_{j+1,\boldsymbol{\alpha}}$ with a reasonable computational cost. This procedure gives rise to an ODE system which is then integrated in time using a second order predictor-corrector method combined with an exponential integrator for the stiff term [70].

In the implementation of the Burgers equation, the algorithm assembles the covariance matrix at each restart as discussed in section 4.2.2. A large memory is not required for such a one dimensional spatial problem. We found that using the random projection technique described in section 4.2.2 did not result in significantly shorter total computational times because the computation of the eigenvalue problem is already efficient in this case by means of Krylov subspace methods.

Example 4.8. For this numerical simulation, we choose the spatial part of the random forcing as $\sigma(x) = \frac{1}{2}\cos(2\pi x)$, the initial condition as $u_0(x) = \frac{1}{2}\sin(2\pi x)$ and set the viscosity $\nu = 0.01$. Under these sets of parameters, it has been proved that there exists a unique invariant measure, which is the long-time limit of the dynamics [109, Theorem 2]. Thus, the main aim of the following simulations is to demonstrate the efficiency of the algorithm in the long-time setting.

For the parameters of DgPC, we take $K = 2$, $N = 2$, and vary the number of the KLE modes D. Final time is $T = 3$ and the interval divided into 30 pieces by taking $\Delta t = 0.1$. The spatial mesh size M_x is set to be 2^7 and the number of samples S to compute moments is taken as 10^5. Finally, sparse indices r are chosen as follows:

i) if $D = 3$, then $r = (2, 2, 2, 1, 1)$, and if $|\boldsymbol{\alpha}| = 2$, we set $r = (2, 2, 2, 1, \cdot)$ resulting in 15 terms in the expansion.

ii) if $D = 4$, then $r = (2, 2, 2, 2, 1, 1)$, and if $|\boldsymbol{\alpha}| = 2$, we set $r = (2, 2, 2, 2, 1, \cdot)$ resulting in 21 terms in the expansion.

iii) if $D = 5$, then $r = (2, 2, 2, 2, 2, 1, 1)$, and if $|\boldsymbol{\alpha}| = 2$, we set $r = (2, 2, 2, 2, 2, 1, \cdot)$ resulting in 28 terms in the expansion.

Note that the first K indices in r correspond to degrees of polynomials in $\boldsymbol{\xi}$ and the remaining to $\boldsymbol{\eta}$. Choosing $r = (2, 2, 2, 2, 1, 1)$ means using only first degree polynomials is η_3 and η_4. Also, setting $r = (2, 2, 2, 2, 1, \cdot)$ for $|\boldsymbol{\alpha}| = 2$ eliminates the cross terms involving first degree polynomials of η_4 in the second order terms. Note that using a sparse index not only reduces the number of terms in PCE but also alleviates the computation of moments.

To compare our algorithm, we use a second order weak Runge-Kutta scheme since the exact solution is not available. To make a fair comparison, both algorithms (Monte Carlo & DgPC) use the same time-step $dt = 0.001$ and the same mesh size $M_x = 2^7$. The number of samples used in MC algorithms are $M_{samp} = 10^4, 5 \times 10^4$ and 10^5 and the corresponding algorithms will be denoted by MC1, MC2 and MC3, respectively. The exact solution is taken as the result of MC3 and the relative L^2 error $||\mathbb{E}[u_{dgpc}] - \mathbb{E}[u_{mc}]||_2/||\mathbb{E}[u_{mc}]||_2$ of the mean is computed. Errors for higher order centered moments are computed similarly.

From Figure 4.1, we observe that all errors grow with time in the initial stages and in particular, the degree of freedom $D = 3$ is the least accurate, which is expected as the dynamics change rapidly during initial stages. Increasing the number of KLE modes entails more accurate expansion up to some order. It can be observed that all the error levels stabilize for moderate times while $D = 5$ is the most accurate. This phenomenon is explained by the convergence to a stationary measure such that statistics do not change considerably after some time.

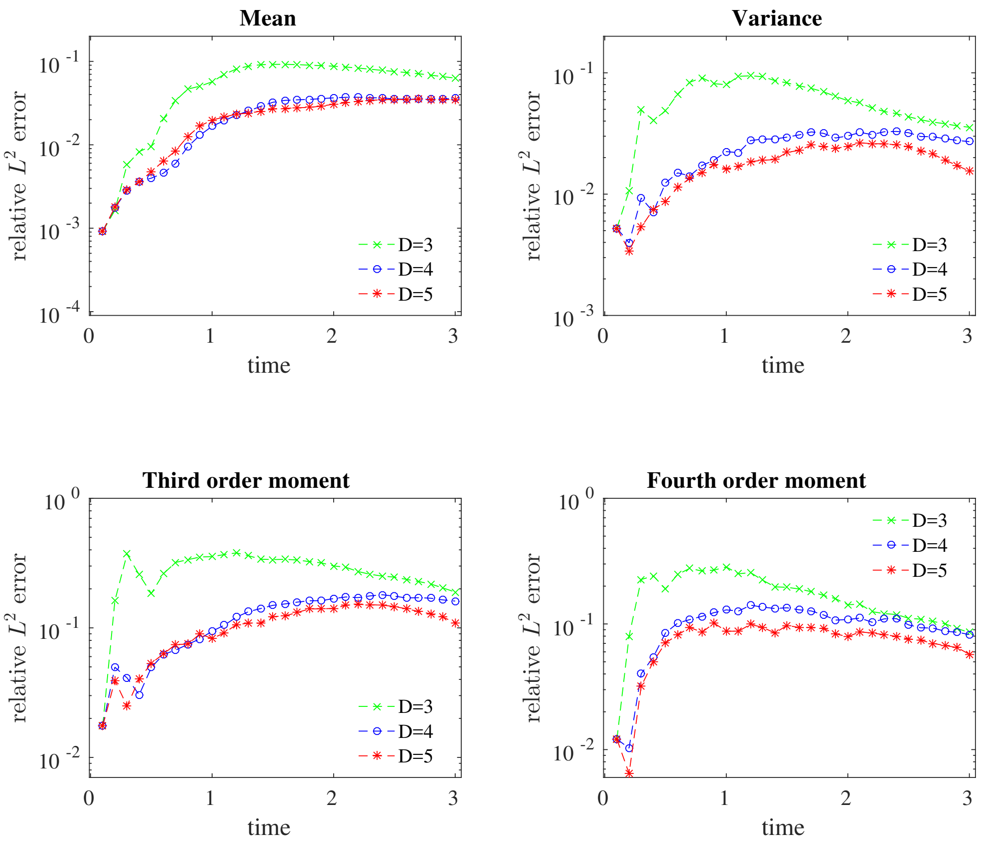


Figure 4.1: Relative errors of centered moments obtained by DgPC with $T = 3$ and $\Delta t = 0.1$. Exact solution is computed by MC3.

To demonstrate that our method is robust with respect to sampling changes and does not really require the propagation of the same initial set of samples in time, we perform the following calculations; see Remark 4.7. Let us start with a set containing $n = 30$ different independent sets of samples $\boldsymbol{\eta}_0(w_{i,1}), \ldots, \boldsymbol{\eta}_0(w_{i,n})$, $i = 1, \ldots, S$, of $\boldsymbol{\eta}_0$. The orthogonal polynomials at the first restart stage $T_\alpha(\boldsymbol{\xi}_0, \boldsymbol{\eta}_0)$ are constructed using the first samples $\boldsymbol{\eta}_0(w_{i,1})$. To calculate the next set of samples $\boldsymbol{\eta}_1(w_{i,1}), \ldots, \boldsymbol{\eta}_1(w_{i,n-1})$ of $\boldsymbol{\eta}_1$ at the next restart, we propagate the remaining samples $\boldsymbol{\eta}_0(w_{i,2}), \ldots, \boldsymbol{\eta}_0(w_{i,n})$ and the samples $\boldsymbol{\xi}_0(w_{i,2}), \ldots, \boldsymbol{\xi}_0(w_{i,n})$ of $\boldsymbol{\xi}_0$ through the now available PCE. This approach, which is computationally challenging for n large and here mostly to show robustness with respect to sample changes, allows us to use different samples to compute orthogonal polynomials at each restart and then propagate the remaining samples of the next variable; see Figure 4.2 for a visualization of this approach.

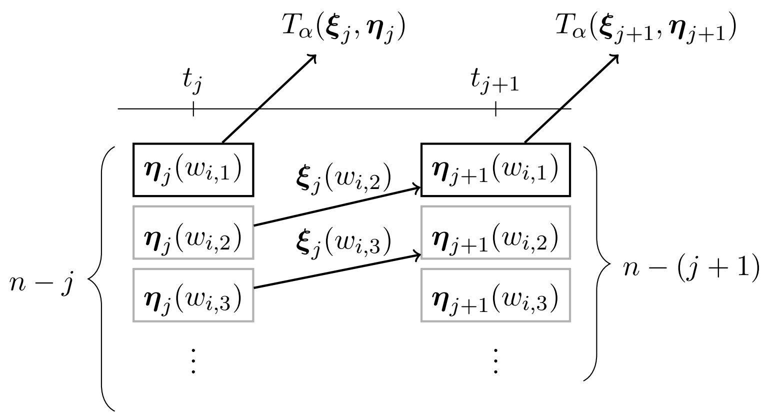

$T_\alpha(\boldsymbol{\xi}_j, \boldsymbol{\eta}_j)$
$T_\alpha(\boldsymbol{\xi}_{j+1}, \boldsymbol{\eta}_{j+1})$
t_j
t_{j+1}
$\boldsymbol{\eta}_j(w_{i,1})$
$\boldsymbol{\eta}_j(w_{i,2})$
$\boldsymbol{\eta}_j(w_{i,3})$
$\boldsymbol{\xi}_j(w_{i,2})$
$\boldsymbol{\xi}_j(w_{i,3})$
$\boldsymbol{\eta}_{j+1}(w_{i,1})$
$\boldsymbol{\eta}_{j+1}(w_{i,2})$
$\boldsymbol{\eta}_{j+1}(w_{i,3})$
$n - j$
$n - (j + 1)$

Figure 4.2: Robustness under the change of samples.

With this idea, we re-ran the simulation and report the results in Figure 4.3. This figure indicates that the latter approach is very comparable in terms of accuracy to the method described in the paper where the initial set of samples is kept fixed. Our method therefore appears to be very robust with respect to changes of samples of the distribution of $\boldsymbol{\eta}_j$. We conclude that there seems to be no need to keep propagating the various set of samples.

We now increase the final time to $T = 6$. DgPC algorithms use the following parameters: $K = 4$, $N = 2$, $D = 3, 4, 5$, $S = 10^5$, and $\Delta t = 0.1$. The corresponding total number of terms for each subinterval becomes 18, 24 and 31. The mesh size is taken as $M_x = 2^8$, which offers better spatial resolution. Table 4.1 summarizes the relative L^2 errors of the DgPC algorithms with different degrees of freedom and the MC methods. All errors are computed by taking MC3 as the exact solution. The time ratio column is computed as the total time required by the each algorithm divided by the elapsed time of MC3 with $M_{samp} = 10^5$ and $dt = 0.0005$. The parameters for MC1 and MC2 algorithms remain the same as above; the algorithms are executed a few times and the resulting errors are averaged. We also include the elapsed times for the offline computation in DgPC algorithms. In practice, the required data for the offline step can be computed once and stored for further executions of the algorithm to speed-up the running time.

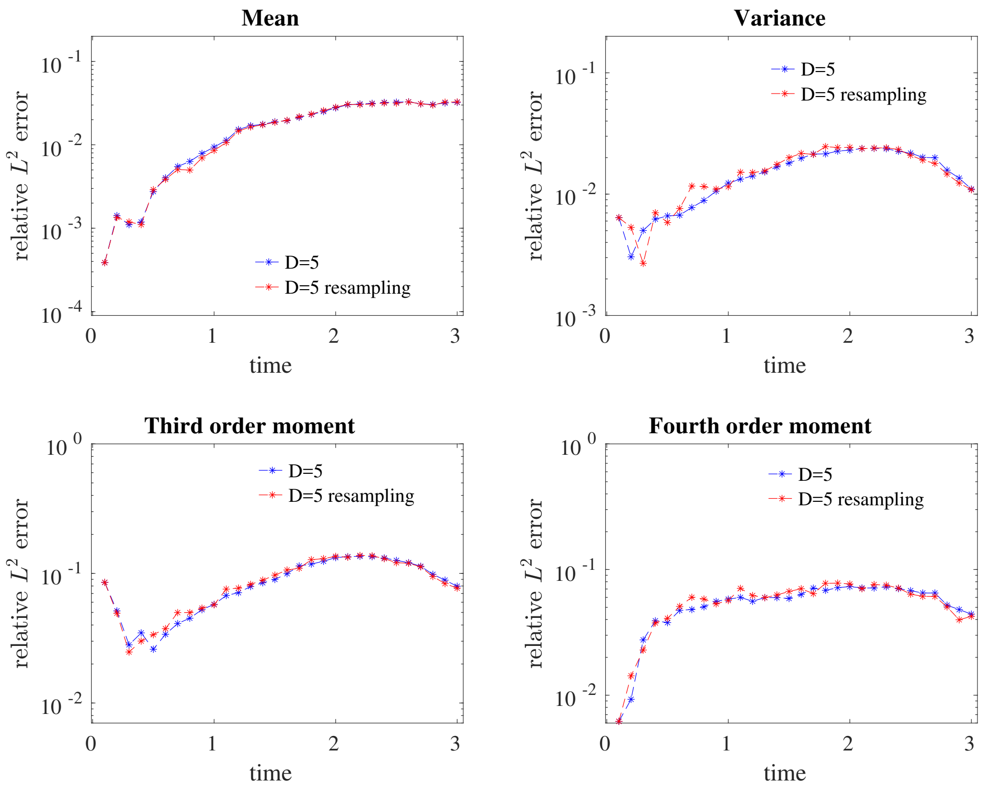


Figure 4.3: Relative errors of moments obtained by DgPC using current sampling and re-sampling approaches.

	Mean	Variance	3rd order	4th order	Time ratio
DgPC: $D = 3$	2.21E-2	1.18E-2	5.20E-2	4.04E-2	0.003
DgPC: $D = 4$	1.86E-2	5.4E-3	3.45E-2	2.41E-2	0.007
DgPC: $D = 5$	1.67E-2	4.0E-3	1.43E-2	1.09E-2	0.02
MC1	2.29E-2	1.16E-2	4.53E-2	2.27E-2	0.05
MC2	1.17E-2	4.0E-3	1.82E-2	9.4E-3	0.25

Table 4.1: Relative errors of centered moments by DgPC and MC methods at T=6. Each time ratio is computed by comparing to MC3.

Table 4.1 demonstrates the idea of using low degree polynomials and small number of terms in PC expansion combined with frequent restarts seems to pay off. DgPC with 31 terms in the expansion (i.e. $D = 5$) attains comparable accuracy as MC2 (i.e. $M_{samp} = 5 \times 10^4$ and $dt = 0.001$) with a computational time which is only eight percent of that Monte Carlo algorithm. Also, we observe that all errors recovered to a level of $O(10^{-2})$, which is an acceptable accuracy for long-time simulations.

The evolution of the energy ratio (2.4) in the KLE is depicted in Figure 4.4 for different values of D. We observe that for the degree of freedom $D = 5$, the method captures at least 99% of the total energy for all times. Moreover, as the dynamics converge to the invariant measure, all energy ratios become very close to 100%, which indicates that the invariant measure lives in a low-dimensional space and the solution can be represented in the KLE form with a small number of terms (at most $D = 3$).

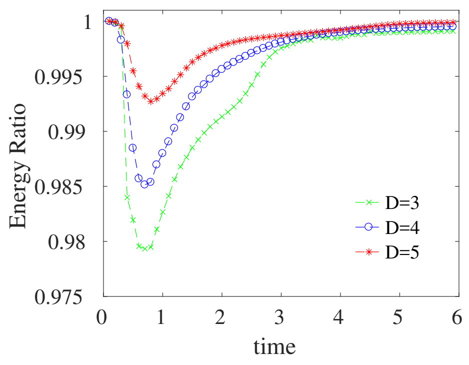


Figure 4.4: Evolution of the retained energy in KLE for different values of degrees of freedom.

Fixing the degree of freedom as $D = 5$, we now employ the adaptive time stepping (4.17) approach. To probe the sensitivity of the algorithm on the threshold parameter ϵ, we choose $\epsilon = 0.005, 0.01, 0.02$ and the initial time-step $\Delta t_0 = 0.1$. Also, we set $\Delta t_{max} = 0.4$ to get $O(10^{-2})$ accuracy for the truncation (2.10) of the forcing term using $K = 4$; see [57, Theorem 5.1]. We utilize quadratic polynomials as our ansatz to approximate the ratio (4.17); see condition i) below (4.17).

Using $\epsilon = 0.005, 0.01$ and 0.02 results in $66, 44$ and 29 number of restarts with the time ratios 0.024, 0.018, and 0.015 compared to MC3, respectively. As expected, decreasing the threshold value implies longer computational time and a larger number of restarts. Furthermore, from Figure 4.5, we observe that errors corresponding to $\epsilon = 0.02$ are the

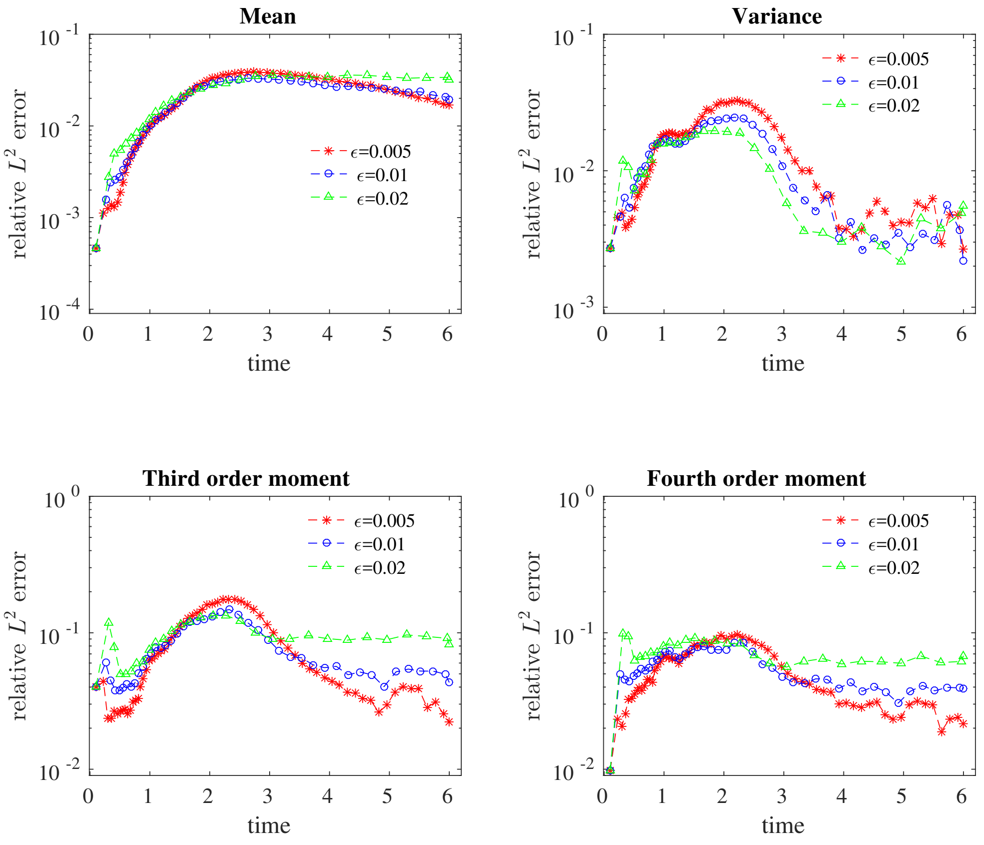


Figure 4.5: Evolution of relative errors of moments with adaptive time-stepping using different threshold values ϵ.

largest in the initial stages and the long-term while errors for the smallest value $\epsilon = 0.005$ correspond the most accurate behavior in the long-term.

We make the following remarks: (i) optimal values of ϵ should be chosen according to the computational time and error level, and for this calculation, $\epsilon \in (0.005, 0.01)$ seems optimal; (ii) earlier stages of the evolution should be analyzed carefully since using large values of the threshold value may result in a loss of accuracy; (iii) using very small values of ϵ may result in accumulation of errors if errors at each restart are significant, e.g., when a small number of degrees of freedom is used in the KLE. Finally, we note that fitting a linear polynomial for (4.17) yields similar results.

To show that the algorithm captures the invariant measure for a long time $T = 6$, we consider the following three different initial conditions

$$u_0(x) = 0.15\sin(2\pi x) \quad \& \quad u_0(x) = 0.5\cos(4\pi x) \quad \& \quad u_0(x) = 0.25(\sin(4\pi x) + \cos(8\pi x)),$$

and compute the moments at time $T = 6$. We see from Figure 4.6 that the dynamics converge to the unique invariant measure, which is a global attractor.

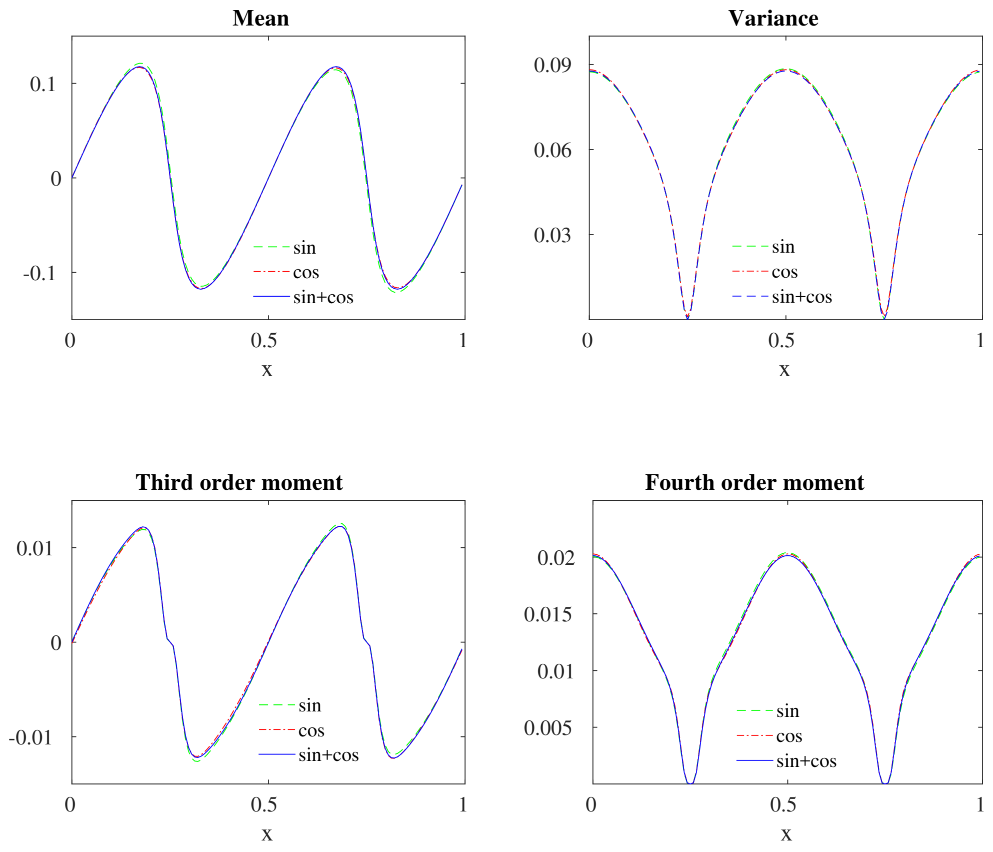


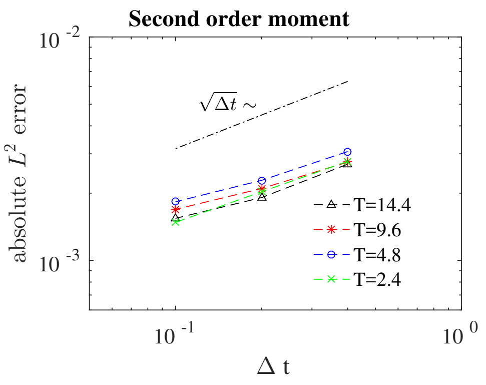
Figure 4.6: Moments of the invariant measure of Burgers equation at $T = 6$ obtained by DgPC.

Example 4.9. The purpose of the following numerical verification is to display the rate of convergence as Δt varies for long time simulations. To this end, we take the initial condition and the forcing

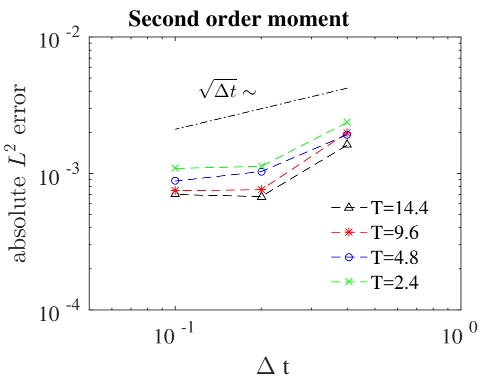
$$u_0(x) = 0.5(\exp(\cos(2\pi x)) - 1.5)\sin(2\pi(x + 0.37)) \quad \& \quad \sigma(x) = 0.5\cos(4\pi x),$$

where the initial condition has several nonzero frequency components in the Fourier space. The viscosity is set to be $\nu = 0.005$. Note that there is no stationary state in the long-term.

Using the same setting of Example 4.8 for parameters of DgPC, we apply DgPC with the varying values $\Delta t = 0.1, 0.2$ and 0.4 for each final time $T = 2.4, 4.8, 9.6$ and 14.4. All simulations use the same time-step $dt = 0.001$ for time-integration. Figure 4.7 demonstrates that the order of convergence in long-time is varying between $O(\Delta t^{0.4})$ and $O(\sqrt{\Delta t})$. This behavior is consistent with the claims made in [57, Theorem 5.1] in the setting of the Hermite PCE.



(a) $D = 4$



(b) $D = 5$

Figure 4.7: Convergence behavior of errors of the second order moment using $\Delta t = 0.1$, 0.2, and 0.4.

Example 4.10. In this example, we test the accuracy of the algorithm against an exact solution. When $\sigma(x) = \sigma$ is constant, the exact moments of the stochastic Burgers equation can be computed by solving the deterministic Burgers equation and estimating appropriate integrals by numerical quadratures.

We set $\sigma = 0.1$ in (4.18) and take the initial condition

$$u_0(x) = 0.1 - 4\nu\pi\cos(2\pi x)/(3 + \sin(2\pi x)).$$

In this case, the exact solution for the deterministic Burgers equation becomes

$$u_{det}(x,t) = 0.1 - 4\nu\pi\exp(-4\nu\pi^2 t)\cos(2\pi(x-0.1t))/(3+\exp(-4\nu\pi^2 t)\sin(2\pi(x-0.1t))),\ t \geq 0.$$

The moments of the stochastic solution $u(x,t)$ are then computed by the following integrals:

$$\mathbb{E}[u(x,t)^n] = \int_{\mathbb{R}^2} (u_{det}(x-z,t)+y)^n p(y,z)\, dydz,$$

where $p(y,z) = \frac{\sqrt{3}}{\pi\sigma^2 t^2} \exp\left(-\frac{2y^2}{\sigma^2 t} + \frac{6yz}{\sigma^2 t^2} - \frac{6z^2}{\sigma^2 t^3}\right)$ [57, equation (3.13)]. We use a large number of quadrature points and the periodicity to compute the moments of the solution accurately.

To perform convergence analysis in terms of degree of polynomials, we take $N = 1, 2$ and 3, and set $K = 3$, $D = 4$, $S = 3 \times 10^5$. This setting results in $8, 18$ and 38 number of terms in the expansion for each time interval. Figure 4.8 demonstrates the resulting relative errors for the moments of the solution. As expected, we observe that increasing the degrees of freedom N helps to reduce the errors and an error accuracy of $O(10^{-3})$ is attained. Since $\sigma = 0.1$ is held constant, the forcing term continuously forces the zeroth order spatial modes of the high statistical moments, which are not damped by the viscosity and grow with time.

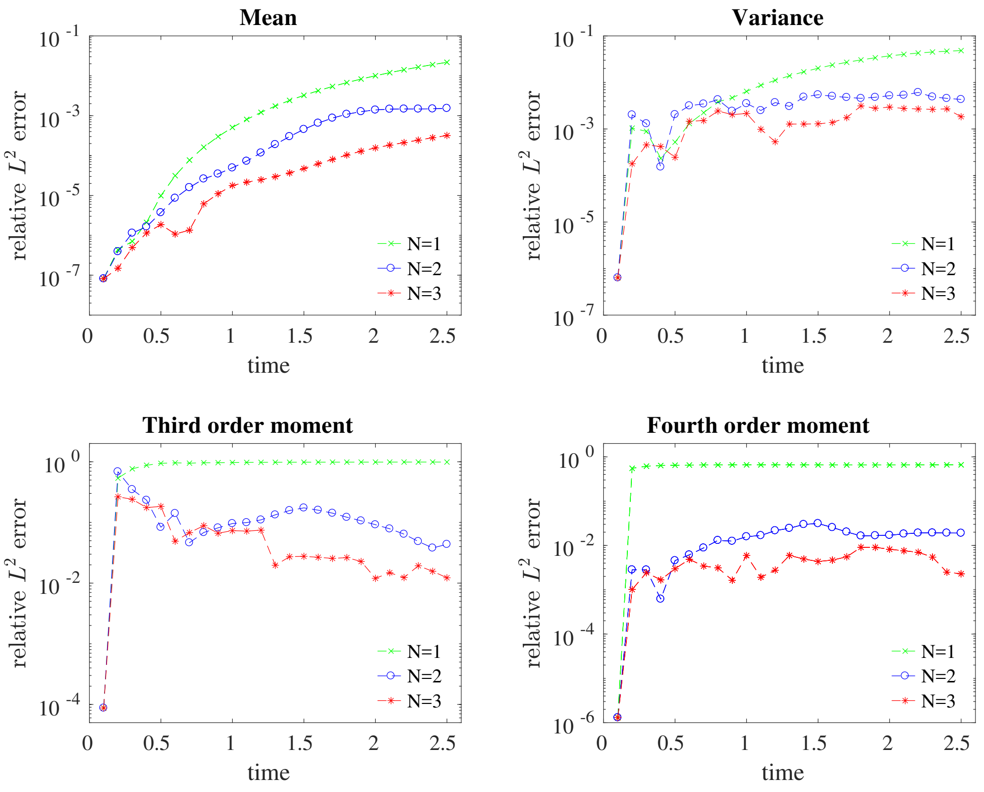


Figure 4.8: Convergence behavior of errors of moments using polynomial degrees $N = 1$, 2, and 3.

Example 4.11. For this simulation, we model the viscosity as an uncertain parameter with a spatial dependence, which can be useful for quantifying uncertainties in applications [41; 66; 85]. To this end, using the same notation of section 4.2.3, the covariance kernel of the underlying random process $Z(x,\omega)$ is assumed to be the following periodic exponential kernel

$$\mathrm{Cov}_Z(x,y) = \sigma_Z^2 \exp\left(-\frac{2}{l_Z^2}\sin^2(\pi(x-y))\right), \quad x,y \in [0,1],$$

where σ_Z is the amplitude and l_Z is the correlation length. We then compute the truncated KLE of the mean zero process $Z(x,\omega)$ and construct the viscosity as a function of Z as

$$\nu(x,\omega) = a_1 + Z^2(x,\omega), \qquad \text{where} \qquad Z(x,\omega) = \sum_{l=1}^{D_Z} \sqrt{\lambda_l}\, U_l(\omega)\, \phi_l(x),$$

with independent uniform random variables $U_l \sim U(-1,1)$ and $a_1 > 0$. Armed with this viscosity, we consider the diffusion term in the Burgers equation (4.18) in divergence form, i.e., $\partial_x(\nu(x,\omega)\,\partial_x u)$.

We utilize Hermite and Legendre polynomial bases on the first subinterval $[0,t_1]$ to expand both Brownian motion and the random viscosity. Although the viscosity does not change in time, its PC representation changes as the PC bases differ at each restart time. Thus, we keep track of the PCEs of both $u(x,t,\omega)$ and $\nu(x,\omega)$. Moreover, we compress both the solution and the random viscosity at each restart, i.e., the KLE is applied to the couple (u_j, Z); see section 4.2.3. A reference calculation is computed by keeping the uniform random variables U_l, $l = 1,\dots,D_Z$, at each restart in a PCE together with $\boldsymbol{\xi}_j$ and $\boldsymbol{\eta}_j$. Relative errors are computed with respect to this reference calculation.

For the following simulation, the parameters are as follows: $K = 2$, $N = 2$, $D = 8$, $D_Z = 3$, $S = 3\times 10^5$. The correlation length of the periodic kernel is set to $l_Z = 2$. To avoid confusion, we slightly change the notation and denote by σ_W the spatial part of the random forcing and consider three different scenarios:

i) $\sigma_Z = 0.04$ and $\sigma_W(x) = 0.1\cos(2\pi x)$,

ii) $\sigma_Z = 0.1$ and $\sigma_W(x) = 0.1\cos(2\pi x)$,

iii) $\sigma_Z = 0.1$ and $\sigma_W(x) = 0.04\cos(2\pi x)$, with the same initial condition $u_0(x) = 0.5\cos(4\pi x)$. These parameters correspond to different relative influences between the viscosity and the random forcing.

We present the evolution of the relative errors for moments of the solution u and the random viscosity ν for $t = 4$ in Figure 4.9. Each moment is averaged over distributions of the random process Z and Brownian motion automatically by the PCE. First, we observe from the first two subfigures that the relative errors of the moments of the solution are of $O(10^{-2})$ in the long-time while the most accurate ones correspond to scenario i). In the same scenario, we see from the rightmost subfigure that the accuracy in the variance of ν decreases and stabilizes in time, which substantiates the observation that the algorithm selects the important part of the moments of the viscosity while keeping the solution accurate. Nevertheless, if slight changes in the moments of the viscosity become significant and accuracy needs to be improved, the KLE can be carried out using correlation matrices rather than covariance matrices.

Regarding the computational time, we note that the DgPC with $K + D = 10$ number of variables requires almost one eighth of the run time of the reference calculation which utilizes $K + D + D_Z = 13$ number of variables in each PCE. Therefore, in cases where random parameters in the equation are high dimensional, applying the KLE to combined random variables is advantageous in terms of speed given a prescribed accuracy.

Figures 4.10a and 4.10b show forty snapshots of the moments of the solution in time for the scenario iii), where black curves indicate the initial states for each moment. Note that since the viscosity is random, for each realization of the viscosity there is a unique invariant measure, and what these figures exhibit is the steady state, which is obtained by averaging these measures over the distribution of the viscosity. We also see that the moments of this averaged measure differ in magnitude compared to those in Figure 4.6 as the relative influences of viscosity and Brownian motion are changed. The rightmost sub-figure plots the one-point probability density function corresponding to this steady state. The density function is easily obtained using the samples of the approximated random field via a kernel density estimation procedure; see Remark 4.7 and [14]. We see that for each point $x \in [0, 1]$, the density function is unimodal and has peaks near the points x, where the variance is minimum. We finally note that the cross covariance structure of the solution on the spatial

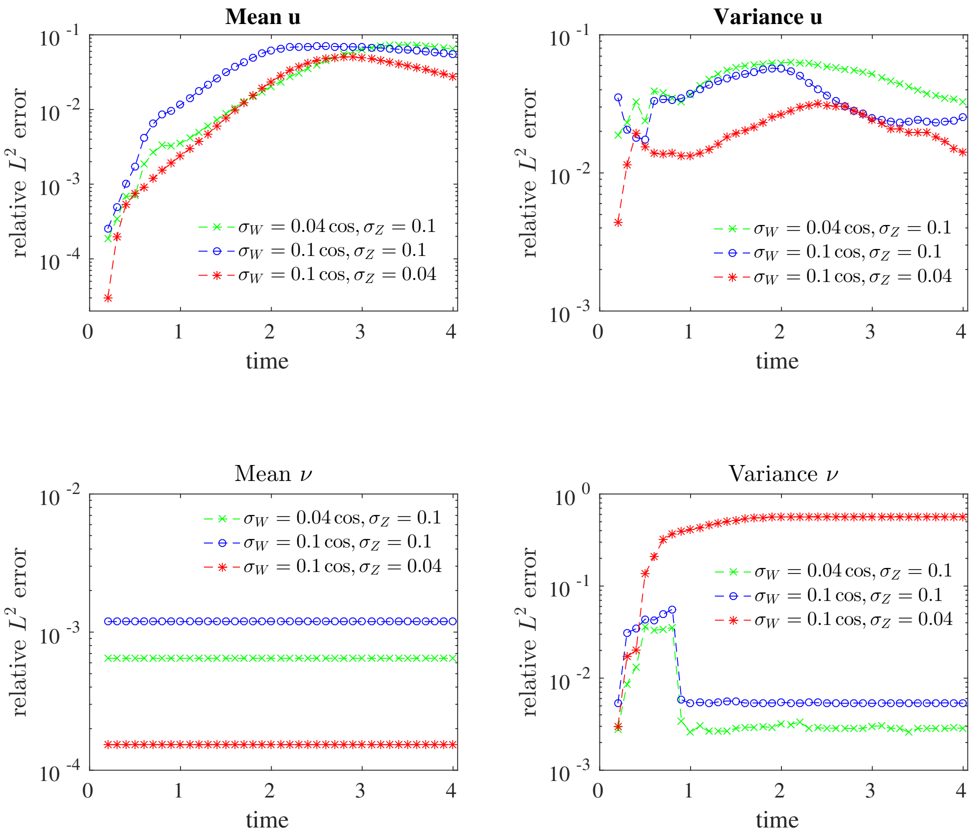


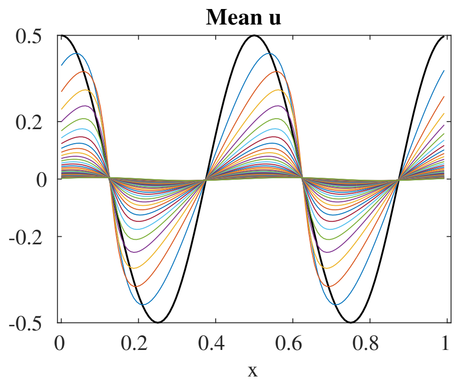
Figure 4.9: Relative errors of moments of the solution and the random viscosity. Errors are computed by comparing DgPC with $D = 8$ to a reference calculation which uses $D = 8$ and $D_Z = 3$.

mesh can also be easily deduced from the algorithm if needed.

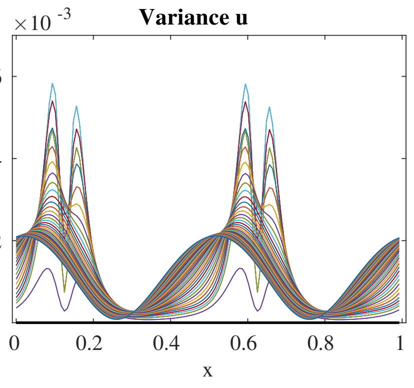
4.3.2 Navier–Stokes Equations

In this section, we provide applications of our algorithm to solve a two-dimensional system of stochastic Navier–Stokes equations (SNS). We consider the following coupled system:

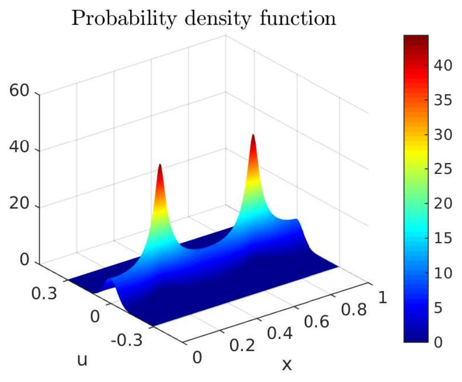
$$\begin{cases} \theta_t + \mathbf{u}\cdot\nabla\theta = \mu\,\Delta\theta, \\ \mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u} = \nu\Delta\mathbf{u} - \nabla P + \sigma\dot{\mathbf{W}}(t), \\ \nabla\cdot\mathbf{u} = 0, \end{cases} \tag{4.19}$$



(a) Mean over time



(b) Variance over time



(c) One-point pdf at final time

Figure 4.10: Snapshots of the second order moments of Burgers equation on $[0, T] = [0, 4]$ and one-point probability density function at $T = 4$.

where θ, the temperature, is convected by the stochastic velocity field $\mathbf{u} = (u, v)$, which is forced by a Brownian motion $\mathbf{W} = (W_1, W_2)$ with independent components and the spatial part $\sigma(x, y) = \mathrm{diag}(\sigma_1(x, y), \sigma_2(x, y))$, $x, y \in [0, 1]$. The temperature diffusivity and fluid viscosity are denoted by μ and ν, respectively. Notice that equations are coupled only through velocity term and the temperature is convected by the random velocity passively.

We take the computational domain $[0, 1] \times [0, 1] \subset \mathbb{R}^2$, and assume that θ and $\mathbf{u}$ are doubly periodic. It is then possible to introduce the stream function ψ such that $\mathbf{u} =$

$(\psi_y, -\psi_x)$, define the vorticity $w = v_x - u_y$ and rewrite the system (4.19) as:

$$\begin{cases} \theta_t + (u\theta)_x + (v\theta)_y = \mu\,\Delta\theta, \\ w_t + (uw)_x + (vw)_y = \nu\Delta w + (\sigma_2)_x \dot{W}_2 - (\sigma_1)_y \dot{W}_1, \\ -\Delta\psi = w, \quad u = \psi_y, \quad v = -\psi_x. \end{cases} \tag{4.20}$$

We also suppose that the stream function is periodic. Following [57; 75], the initial condition for the vorticity w is chosen to be

$$w(x,y,0) = C - \frac{1}{2\delta}\exp\left(-\frac{I(x)(y-0.5)^2}{2\delta^2}\right), \tag{4.21}$$

where $I(x) = 1 + \varepsilon(\cos(\gamma\, 2\pi x) - 1)$ with $\gamma \in \mathbb{N}$, and C is a constant to make the initial condition mean zero on $[0,1]^2$. This choice corresponds to a flat shear layer of width δ concentrated at $y = 0.5$. The width is perturbed sinusoidally with the amplitude ε and spatial frequency γ. In our numerical experiments, we will also consider the reflected initial condition $w(y,x,0)$ for which the layer is concentrated vertically; see Figures 4.11 and 4.14.

We consider the following initial condition for the temperature:

$$\theta(x,y,0) = \begin{cases} H_\delta(y-0.25), & \text{if } y \le 0.4, \\ 1 - 2H_\delta(y-0.5), & \text{if } 0.4 < y < 0.6, \\ -H_\delta(0.75-y), & \text{if } y \ge 0.6, \end{cases},$$

where

$$H_\delta(x) = \begin{cases} 0, & \text{if } x < -\delta, \\ \frac{x+\delta}{2\delta} + \frac{\sin(\pi x/\delta)}{2\pi}, & \text{if } |x| \le \delta, \\ 1, & \text{if } x > \delta \end{cases}$$

is the mollified Heaviside function. This formulation yields an initial condition, which consists of four connected layers, where interfaces between layers have thickness δ. As discussed in [57; 75], setting small values to δ creates a sharp shear layer and temperature interface. As a consequence of Kelvin-Helmholtz instability, the fluid then will roll-up, and the temperature will be convected and mixed up [74].

The Hermite PCE is effectively applied to stochastic Navier–Stokes equations (in most cases without Brownian motion forcing) in various manuscripts [67; 122; 128; 119; 57; 75].

The presence of Brownian motion forcing makes the system very hard to solve even for short times due to the overwhelming number of random variables needed in the Hermite PCE. Therefore, we now apply the DgPC algorithm to the system (4.20).

We choose the same orthonormal system $m_{j,k}(t)$ on $[t_j, t_{j+1}]$ as in the previous section and project each component $W_1(t)$ and $W_2(t)$ of the Brownian motion to obtain $\boldsymbol{\xi}_j$. The total number of ξ_k's will be denoted by K, where the first (last) $K/2$ variables correspond to the first (second) component of $\mathbf{W}(t)$. Assuming the variables $w, \theta, \mathbf{u}$ and ψ admit PCEs, the method keeps track of the corresponding expansions. At each time-step t_j, the KLE is applied to (w_j, θ_j) and the mode $\boldsymbol{\eta}_j$ is obtained. Then, Galerkin projection of (4.20) onto the space span$\{T_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_j, \boldsymbol{\eta}_j) : \alpha \in \mathcal{J}^r_{K+D,N}\}$ yields the following equations:

$$\begin{cases} \partial_t(\theta_{j+1,\boldsymbol{\alpha}}) + \partial_x(u_{j+1}\theta_{j+1})_{\boldsymbol{\alpha}} + \partial_y(v_{j+1}\theta_{j+1})_{\boldsymbol{\alpha}} = \mu\,\Delta\theta_{j+1,\boldsymbol{\alpha}}, \\ \partial_t(w_{j+1,\boldsymbol{\alpha}}) + \partial_x(u_{j+1}w_{j+1})_{\boldsymbol{\alpha}} + \partial_y(v_{j+1}w_{j+1})_{\boldsymbol{\alpha}} = \nu\,\Delta w_{j+1,\boldsymbol{\alpha}} \\ \qquad +(\sigma_2)_x \displaystyle\sum_{k=K/2+1}^{K} m_{j,k-K/2}(t)\,\mathbb{E}[\xi_{j,k}\,T_{\boldsymbol{\alpha}}] - (\sigma_1)_y \sum_{k=1}^{K/2} m_{j,k}(t)\,\mathbb{E}[\xi_{j,k}\,T_{\boldsymbol{\alpha}}], \\ -\Delta\psi_{j+1,\boldsymbol{\alpha}} = w_{j+1,\boldsymbol{\alpha}}, \quad u_{j+1,\boldsymbol{\alpha}} = \partial_y(\psi_{j+1,\boldsymbol{\alpha}}), \quad v_{j+1,\boldsymbol{\alpha}} = -\partial_x(\psi_{j+1,\boldsymbol{\alpha}}). \end{cases} \tag{4.22}$$

The resulting deterministic PDE system (4.22) is solved utilizing a truncated Fourier series and FFT in two dimensions on a mesh of size $M_x \times M_x$.

As we discussed in Section 4.2.2, there are two main methods to compute KLE: (i) assemble the full covariance matrix using (4.6) and use a Krylov subspace method to find largest eigenvalues (as was done in the previous section); or (ii) use the random projection technique to accelerate the computation by equation (4.9) and find eigenvalues of the resulting small matrix. To show the computational savings incurred by the second method, some SNS systems were solved using both methods and accuracies are compared.

Example 4.12. This simulation concerns the short-time accuracy and the computational time of the DgPC algorithm, which will be assessed using comparisons with Monte Carlo methods with a sufficiently high number of samples.

We set $\mu, \nu = 0.0002$ and take the spatial part of forcing as

$$(\sigma_1)_y = 0.1\pi\cos(2\pi x)\cos(2\pi y), \quad (\sigma_2)_x = 0.1\pi\cos(2\pi x)\sin(2\pi y).$$

The parameters $\delta = 0.025, \varepsilon = 0.3$ and $\gamma = 2$ give rise to the initial conditions which are depicted in Figure 4.11. Similar parameters can be found in [57, Section 4.1].

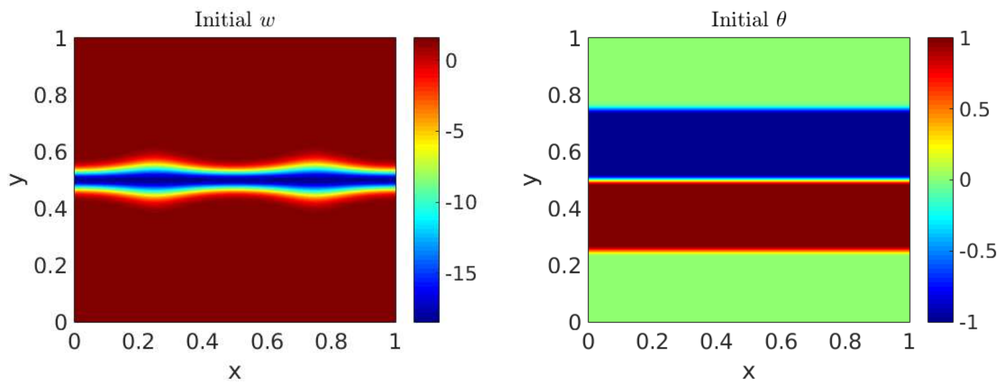


Figure 4.11: Initial conditions for vorticity w and temperature θ.

We apply the DgPC with the following parameters: $K = 4$, $N = 2$, $S = 2 \times 10^5$, $T = 1$, $\Delta t = 0.1$, $M_x = 2^7$ and $D = 4, 6, 8$. Sparse indices r are chosen as:

i) if $D = 4$, then $r = (2, 1, 2, 1, 2, 2, 1, 1)$, and if $|\alpha| = 2$, we set $(2, \cdot, 2, \cdot, 2, 2)$. This results in 19 coefficients in the PCE.

ii) if $D = 6$, then $r = (2, 1, 2, 1, 2, 2, 1, 1, 1, 1)$, and if $|\alpha| = 2$, we set $(2, \cdot, 2, \cdot, 2, 2, 1, 1)$. This results in 30 coefficients in the PCE.

iii) if $D = 8$, then $r = (2, 1, 2, 1, 2, 2, 2, 2, 2, 1, 1, 1)$, and if $|\alpha| = 2$, we set $(2, \cdot, 2, \cdot, 2, 2, 2, 2, 2)$. This results in 41 coefficients in the PCE.

We note that sparser sets of indices can be chosen more aggressively in applications to provide faster offline and online computations.

The first four moments of vorticity and temperature are plotted in Figure 4.12 and Figure 4.13. Higher order moments are centered. Roll-up of the fluid is clearly observed in the mean temperature. Due to the Kelvin-Helmholtz instability and the structure of the initial vorticity, the thin shear layer evolves, rolls up and eventually forms two vortices concentrated at the same locations of sinusoidal perturbations; see [74; 57] for the previous results and discussions.

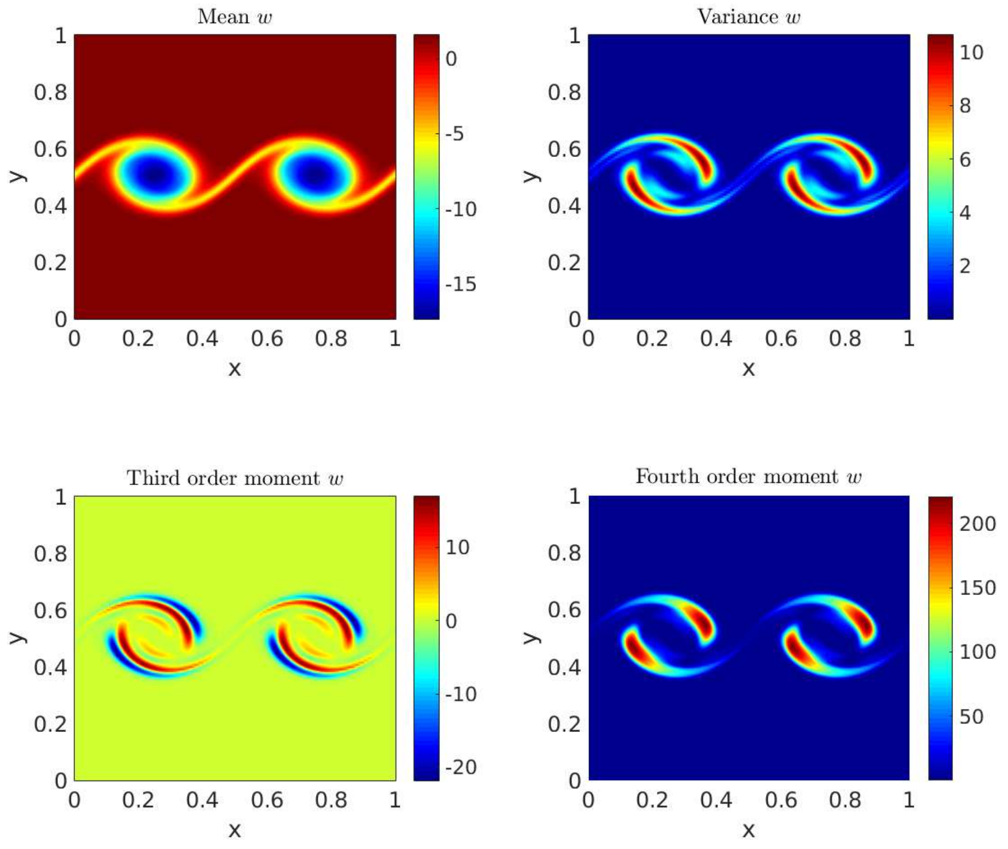


Figure 4.12: Moments of vorticity w obtained by DgPC with $D = 8$ at $T = 1$.

To make comparisons as meaningful as possible, we use second order integration methods, namely weak Runge-Kutta and predictor corrector, with the same time-step $dt = 0.001$ in both Monte Carlo and DgPC algorithms. Diffusion terms are integrated analytically by an exponential integrator scheme. The number of samples used in MC are $M_{samp} = 1000, 5000, 10000,$ and the corresponding algorithms will be denoted by MC1, MC2 and MC3, respectively. Relative L^2 errors and computational times are computed using the algorithm MC3 as the "exact" solution.

Relative errors for the moments of the vorticity w are given in Table 4.2. In this implementation, the algorithm assembles the full covariance matrix at each restart using (4.6) and uses the Arnoldi method to compute the largest D eigenvalues. We found that error levels and convergence behavior for the moments of the temperature θ were very

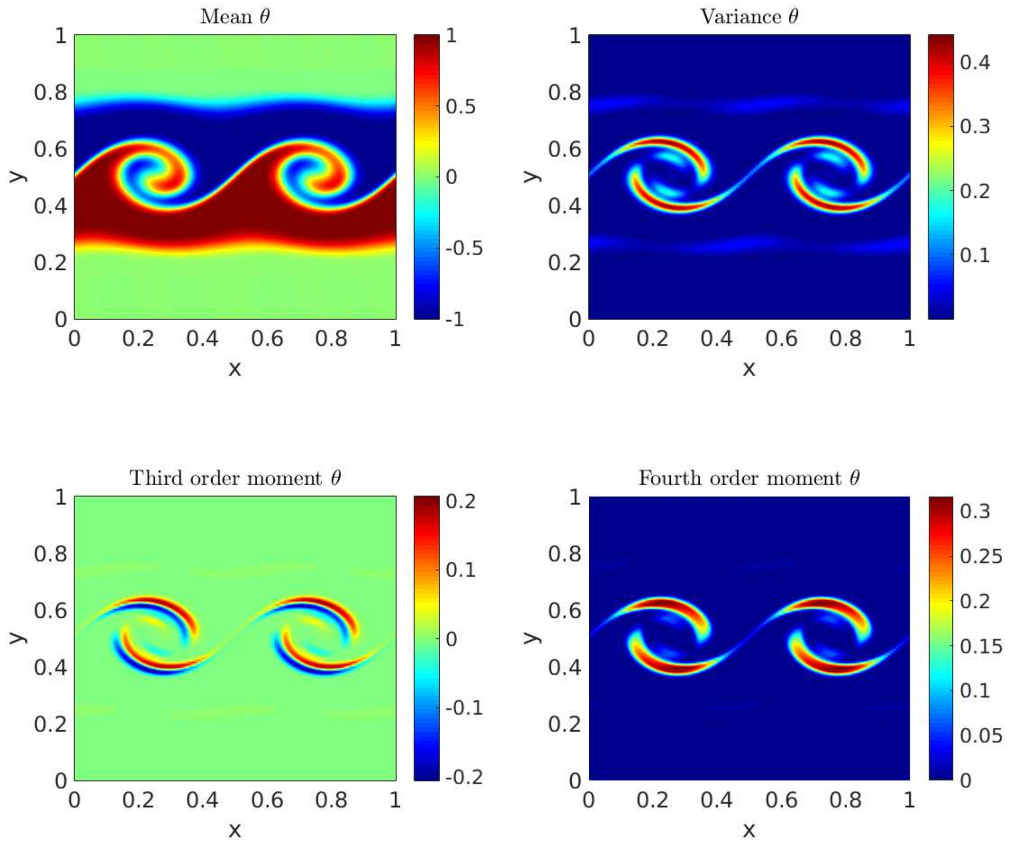


Figure 4.13: Moments of temperature θ obtained by DgPC with $D = 8$ at $T = 1$.

similar and hence are not displayed. It can be observed that using $D = 8$ random modes in DgPC with second degree polynomials yields a similar accuracy to that achieved in MC2. Convergence order of DgPC in terms of the parameter D seems to be at least quadratic whereas convergence of MC is approximately of $O(1/\sqrt{M_{samp}})$; especially for higher order moments.

For comparison, we now apply the random matrix technique discussed in Section 4.2.2 to compute the KLE at each restart. Recall that this technique does not require assembling any covariance matrix and is therefore memory efficient. Table 4.3 shows the resulting errors of the DgPC algorithm using this technique with the target rank parameter $l = D+p$, where we used $p = 10$ for the oversampling parameter. Comparing Table 4.2 and 4.3, we observe that the error levels remain comparable while the computational costs are not. Elapsed

Algorithm	Mean	Variance	3rd order	4th order	Time ratio
DgPC: $D = 4$	8.2E-3	1.58E-1	7.50E-1	4.90E-1	0.025
DgPC: $D = 6$	2.7E-3	3.57E-2	1.55E-1	8.34E-2	0.041
DgPC: $D = 8$	2.1E-3	2.64E-2	7.01E-2	5.98E-2	0.073
MC1	4.3E-3	7.26E-2	1.42E-1	1.16E-1	0.1
MC2	3.4E-3	2.90E-2	5.40E-2	4.65E-2	0.5

Table 4.2: Relative errors of moments of vorticity w at $T = 1$ and timing. Exact solution is taken as algorithm MC3.

times are approximately divided by 8, 4 and 2 for the degrees of freedom $D = 4, 6$ and $D = 8$, respectively. For $D = 8$, this shows that the (total) algorithm now runs in about half the time. Specifically for the KLE step, we found computational times are reduced by approximately 1000. This simulation confirms that the computation of the KLE becomes a serious computational bottleneck in high dimensions when the covariance matrices are assembled (accounting for half of the time of the full algorithm). Thanks to the random projection method, the KLE no longer constitutes a computational bottleneck for the SNS simulations presented here. The computational costs are now mostly dominated by time evolution and restart procedures.

Algorithm	Mean	Variance	3rd order	4th order	Time ratio
DgPC: $D = 4$	8.2E-3	1.59E-1	7.57E-1	4.98E-1	0.003
DgPC: $D = 6$	2.8E-3	3.73E-2	1.58E-1	8.25E-2	0.010
DgPC: $D = 8$	2.1E-3	2.58E-2	6.67E-2	5.90E-2	0.035

Table 4.3: Elapsed times and relative errors of moments of vorticity w at $T = 1$. The random projection technique with the parameter $l = D + 10$ is used to accelerate computation of the KLE.

Example 4.13. Using the same scenario as in the previous example, we consider a stochastic viscosity $\nu = U(0.0002, 0.0004)$ and set μ to be the same random variable, i.e. $\mu = \nu$. Since the Monte Carlo simulation takes a very large amount of time to compute, we restrict ourselves to the final time $T = 0.5$ and the mesh size $M_x = 2^6$. Monte Carlo algorithms MC1, MC2 and MC3, are executed with the number of samples 100×100, 200×200 and 300×300, respectively. These samples correspond to the samples of Brownian motion and the viscosity. The parameters of DgPC remain the same except we increase S to 3×10^5 as there is an additional random coefficient in the system.

Algorithm	Mean	Variance	3rd order	4th order	Time ratio
DgPC: $D = 4$	3.26E-4	2.42E-2	3.05E-1	5.77E-2	0.0009
DgPC: $D = 6$	2.85E-4	1.63E-2	1.95E-1	4.89E-2	0.0025
DgPC: $D = 8$	2.74E-4	4.45E-3	6.27E-2	3.30E-2	0.0067
MC1	2.60E-3	2.29E-2	9.59E-2	5.25E-2	0.11
MC2	1.11E-3	8.51E-3	4.10E-2	2.07E-2	0.44

Table 4.4: Relative errors of moments of vorticity w at $T = 0.5$. Elapsed times are compared to Algorithm MC3.

Table 4.4 exhibits the relative errors of DgPC for the vorticity using the random matrix approach for the KLE. Comparing Table 4.4 and 4.3, we see that relative elapsed times of DgPC with respect to MC3 are further improved. Additional randomness for MC means an extra dimension to sample from, whereas for DgPC, it means an extra variable that needs to be compressed into the modes $\boldsymbol{\eta}$. Since the dynamics crucially depend on the behavior of the viscosity, using few realizations for viscosity sampling in MC is not recommended. In this setting, we found that our MC simulations demanded a high CPU time compared to DgPC. Note, however, that viscosity sampling could clearly be performed in parallel in a MC framework—something that is not as easily feasible in the PCE setting.

Example 4.14. The preceding simulations were concerned with short time evolutions of SNS and comparisons of the proposed algorithm with a Monte Carlo method. Numerical results for reasonably short time computations indicated that our algorithm achieved a similar accuracy compared to MC typically for a smaller computational cost.

We are now interested in long time simulations and convergence to steady states. Since there is no random forcing acting upon the temperature equation in (4.20), the (uncoupled) temperature diffuses to zero quickly. Therefore, we only solve the vorticity equation in the system (4.20).

The following numerical experiment considers the vorticity equation with a deterministic viscosity $\nu = 0.00055$ and a spatial forcing as described in Example 4.12. The parameters of the simulation are $M_x = 2^6$, $S = 3 \times 10^5$, $T = 288$ and $\Delta t = 0.12$. PC expansions with thirty number of terms are employed on each subinterval. The four-step Adams predictor-corrector method is used for the time integration.

Figure 4.14 shows three different initial conditions for the vorticity. The first layer is supported around $x = 0.5$ while the others are aligned horizontally. Widths of all layers are widened and different sinusoidal perturbations are considered.

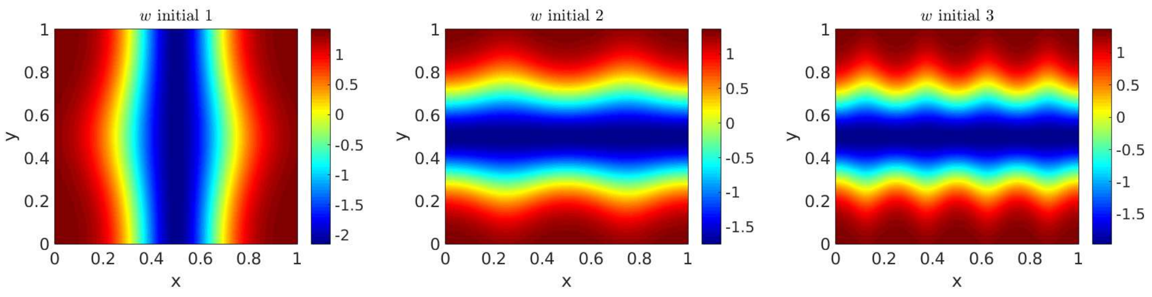


Figure 4.14: Different initial conditions for vorticity w.

In Figure 4.15, we show the L^2-norm of the successive differences of the first two moments in time. Each column represents one of the initial conditions presented in the corresponding column in Figure 4.14. After a (very) long time, the norms of the successive differences drop below $O(10^{-3})$, which (numerically) indicates that statistical moments no longer change significantly in time. In all cases, we found that the dynamics converged to the same state, which is shown in Figure 4.16. Notice that the invariant measure is a non-Gaussian random field and the moments have oscillations in the x variable. We also see that high variance regions correspond to where the mean fields display peaks. Based on these findings for this scenario, we assert that the dynamics converge to an invariant measure which is numerically captured in the long-time by the DgPC algorithm.

Remark 4.15. In long time computations, the MC method usually requires the propagation

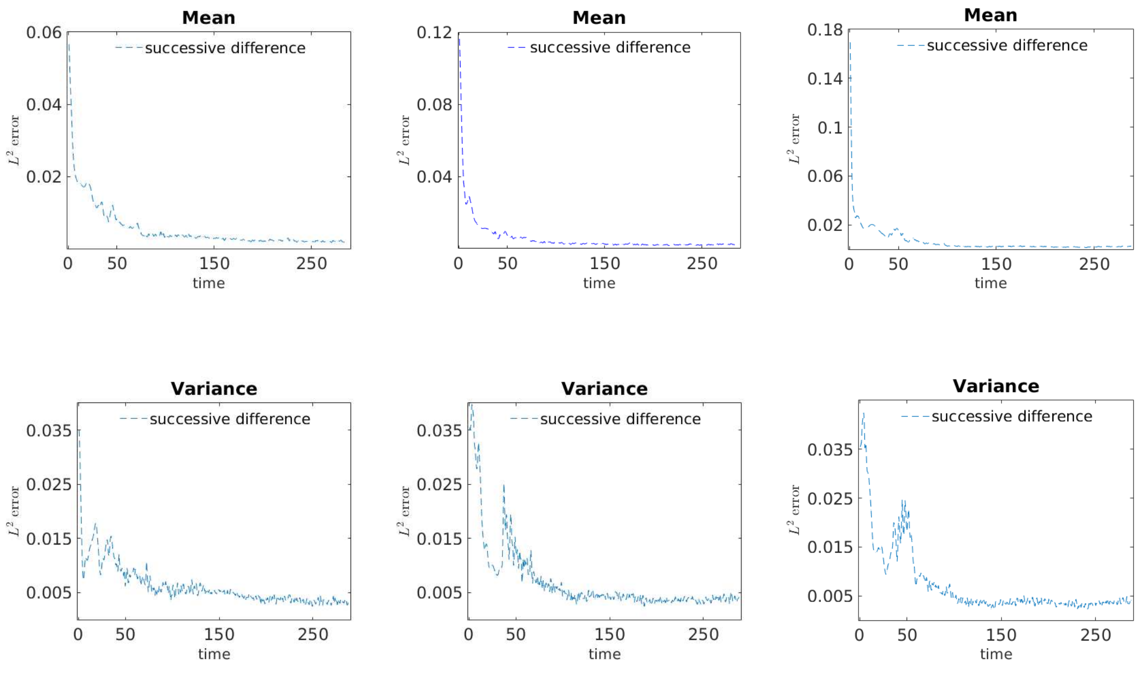


Figure 4.15: L^2-norm of successive differences of moments using three different initial conditions. Each column corresponds to an initial condition depicted in Figure 4.14.

of many realizations in time, which renders the method hardly affordable in some cases. However, if the dynamical system possesses a unique ergodic invariant measure, the MC method may be carried out to sample such a measure by considering a single, very long time MC realization, which repeatedly visits the whole state space. While carrying out such a sampling is also computationally expensive, it is likely to compare favorably to our DgPC algorithm in this case.

In general, ergodicity or uniqueness of an invariant measure may not be known or not hold for complicated physical dynamics (e.g., invariant measures parametrized by a random parameter as in Example 4.13). In such cases, our algorithm offers a viable alternative to the MC method to capture the long-term dynamics by providing statistical information resulting from the expansion coefficients.

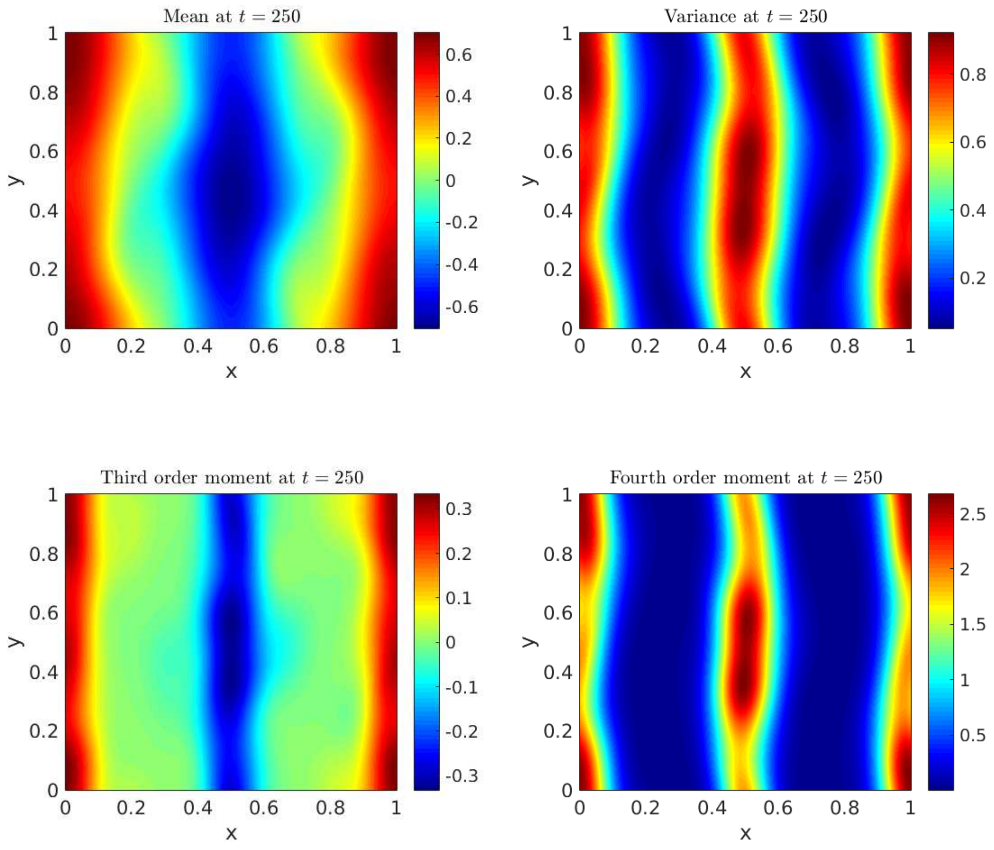


Figure 4.16: Statistical moments of the invariant measure of the vorticity at time $t = 250$ obtained by DgPC.

Chapter 5

Dynamical SGC method for SDEs

5.1 Introduction

In this chapter, we focus on stochastic collocation methods which use deterministic quadrature nodes in the random space to approximate expectations of functionals of solutions. These methods aim to achieve the ease of implementation of MC methods and fast convergence behavior of stochastic Galerkin methods at the same time. Similar to Galerkin methods, their convergence depends on the smoothness of the solution in the random input variables. Their effectiveness relies upon the dimensionality of the random parameter space and they work well if the stochastic system involves random moderate dimensions. It has been shown, especially in uncertainty quantification literature, that they provide a strong alternative to MC methods for differential equations with time-independent random parameters; see e.g. [124; 28; 126; 6; 87; 86; 127; 66].

For equations driven by time-dependent noise, collocation methods with pre-determined quadrature rules have appeared in the recent literature [40; 39; 132; 133; 77; 71]. Manuscripts [132; 133] combined a sparse grid collocation (SGC) method with weak sense time-integration methods to compute second order moments of the solutions of SDEs and SPDEs driven by white noise. It has been proved and observed numerically in [132] that straightforward application of collocation methods leads to failure in long-time integration. As we noted and observed numerically in the previous chapters, the underlying reason of this failure is that in the presence of random forcing time, the number of stochastic variables needed to maintain a prescribed accuracy increases with time. This means, for collocation-based methods, that the number of collocation points should grow with time. The manuscript [132] then introduces a recursive approach for long times based on the SGC method to compute second order moments of linear SPDEs; see also [73; 84]. On the other hand, optimal quantization methods [95; 76; 96] aim to find optimal discrete approximations, e.g. Voronoi quantizations, to the solutions of SDEs, which are adapted to the dynamics. These quantizations are obtained by approximating Brownian motion by a finite-dimensional random process, e.g. truncated KLE of Brownian motion, and deriving ODE systems for the quantizers.

In the following, we propose a dynamical collocation-based method in time to alleviate long-time integration problems in the setting of SDEs. The method propagates optimal quadrature rules for the solution in time and uses pre-determined quadrature rules for the random forcing. In this sense, it can be considered as an extended combination of the proposed methods in [132; 95]. Using a similar restarting strategy of the preceding chapters and a time-integration method, the method constructs sparse quadrature rules for the solution variables on-the-fly. It then estimates expectations of functionals of the future solution variables by using sparse quadrature rules of the solution variables at the current time and the random forcing variables. By constructing quadrature rules with a small number of nodes and employing frequent restarts, the algorithm can utilize small and time-independent degrees of freedom at each restart, while maintaining the accuracy in the long time. We demonstrate the efficiency of the proposed method numerically using low-dimensional nonlinear SDEs.

5.2 A Simple Stochastic Collocation Method

Following the approaches given in the previous chapters, we introduce a sparse quadrature-based collocation method for the d-dimensional SDE (3.1) with $\sigma > 0$ as follows.

First, we consider a time discretization of the interval $[0, T]$

$$\tau_i = i\,dt, \quad i = 0, \ldots, M_T,$$

where $dt = T/M_T$ and approximate the solution $u(t; u_0, \{W(\tau); \tau \leq t\})$ of (3.1) in the weak-sense by the Euler-Maruyama scheme

$$u(\tau_{i+1}) = u(\tau_i) + \mathcal{L}(u(\tau_i))\,dt + \sigma(W(\tau_{i+1}) - W(\tau_i)).$$

Then using the convergence property of the expansion (2.10) in L^2, we can approximate Brownian motion increments by finite dimensional variables $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_K)$:

$$u(\tau_{i+1}; u_0, \boldsymbol{\xi}) = u(\tau_i; u_0, \boldsymbol{\xi}) + \mathcal{L}(u(\tau_i; u_0, \boldsymbol{\xi}))\,dt + \sigma \sum_{k=1}^{K} \xi_k \int_{\tau_i}^{\tau_{i+1}} m_k(\tau) d\tau, \tag{5.1}$$

where $u(\tau_0; u_0, \boldsymbol{\xi}) = u_0$.

Let now $\{w_0^p, u_0^p\}_{p=1}^{Q_{u_0}}$ and $\{\mathbf{w}^q, \boldsymbol{\xi}^q\}_{q=1}^{Q_{\boldsymbol{\xi}}}$ be pre-determined quadrature rules for the random variables u_0 and $\boldsymbol{\xi}$ with the corresponding levels λ_{u_0} and $\lambda_{\boldsymbol{\xi}}$, respectively. Here these quadrature rules denote any enumerations of their multi-dimensional versions. Then the equation (5.1) naturally gives rise to a non-intrusive approximation to the SDE by the following equation:

$$u(\tau_{i+1}; u_0^p, \boldsymbol{\xi}^q) = u(\tau_i; u_0^p, \boldsymbol{\xi}^q) + \mathcal{L}(u(\tau_i; u_0^p, \boldsymbol{\xi}^q))\,dt + \sigma \sum_{k=1}^{K} \xi_k^q \int_{\tau_i}^{\tau_{i+1}} m_k(\tau) d\tau, \tag{5.2}$$

for $p = 1, \ldots, Q_{u_0}$ and $q = 1, \ldots, Q_{\boldsymbol{\xi}}$. This equation dictates the evolution of the initial particles u_0^p under the trajectories of the forcing particles $\boldsymbol{\xi}^q$. In contrast to Monte Carlo methods, the random forcing is sampled deterministically and approximated by its finite-dimensional approximation via the spectral projection (2.10), and the samples of u_0 are taken as quadrature points. Thus, the method is sample-error free.

Remark 5.1. The method introduces three level of approximations. First, SDE is discretized in time. Then, Brownian motion increments are approximated by their finite-dimensional approximations. Finally, we approximate the continuous equation (5.1) by its discrete approximation (5.2) using quadrature rules. Hence, there are three degrees of freedom that are of interest: dt, K, and λ.

Remark 5.2. Note that although we used the Euler method in the formulation, this is not required. Any higher order method can be used to discretize SDE in time. Moreover, the noise amplitude σ can be a function of the solution u; see numerical Example 5.11.

Remark 5.3. For any fixed K, the finite dimensional approximation (2.10) of Brownian motion entails a continuous finite-variation process on $[0, T]$. Thus, the integral with respect to this finite-variation process can be understood in the Stieltjes sense. Then the main questions are when and in what sense the approximate solution (5.2) converges to the true solution. Unfortunately, answers to these questions are beyond the scope of this thesis. For theoretical discussions on the convergence of approximations for SDEs with smooth coefficients in a similar setting, we refer to [95; 76; 96]. Nevertheless, the numerical experiments show clear convergence in moments; see Section 5.4.

The approximate solution $u(t; u_0^p, \boldsymbol{\xi}^q)$ of (5.2) readily entails (approximate) statistical moments by computing

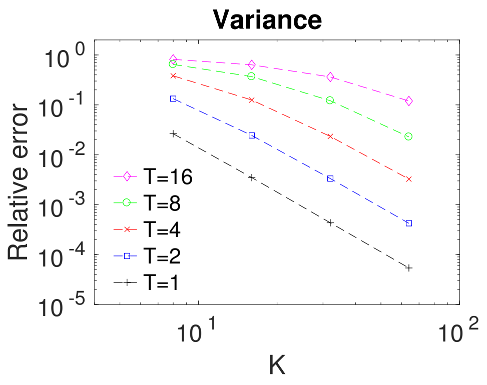
$$\mathbb{E}[u(t; u_0, \boldsymbol{\xi})^{\boldsymbol{\alpha}}] \approx \sum_{p=1}^{Q_{u_0}} \sum_{q=1}^{Q_{\boldsymbol{\xi}}} w_0^p \, \mathbf{w}^q \, u(t; u_0^p, \boldsymbol{\xi}^q)^{\boldsymbol{\alpha}},$$

where $\boldsymbol{\alpha}$ is a multi-index, see (2.8). Here, we use Smolyak sparse grid quadrature (2.12) for Gaussian $\boldsymbol{\xi}$ and for u_0, any accurate quadrature rule can be considered.

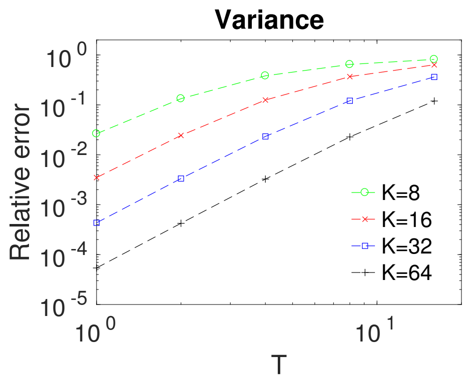
A similar collocation strategy in a Monte-Carlo setting using weak-integration is employed in [132; 133] to compute the second order moments of the solution. It is noted that the efficiency of the collocation strategy depends on the strength of the noise and the length of the time interval. Indeed, in order to maintain a prescribed accuracy the expansion (2.10) requires the number of stochastic variables to be increasing with time. Thus, the number of quadrature points needed to maintain an accuracy quickly becomes overwhelming for long times.

Here is a simple motivating demonstration of the long-time integration problem in case of the Ornstein-Uhlenbeck process. We set $\mathcal{L}(u) = 10(0.1 - u)$ and $\sigma = 4$, and take a deterministic initial condition $u_0 = 1$. We also take $K = 8, 16, 32, 64$, and consider different final times $T = 1, 2, 4, 8, 16$. Since the solution stays Gaussian, we use a sparse Gauss-Hermite rule for $\boldsymbol{\xi}$ with level $\lambda_{\boldsymbol{\xi}} = 1$. To make a fair comparison, we use the same time discretization method (second order Runge-Kutta method) with the time step $dt = 1\text{E-}3$ in each scenario. Figure 5.1 shows that as time increases from $T = 1$ to $T = 16$, the convergence behavior in the second moment of this simple method drops from $O(K^{-3})$

to $O(K^{-1})$. This clearly indicates that the degrees of freedom required for this simple collocation method to maintain a desired accuracy should increase with time.



(a) Relative error versus K



(b) Relative error versus T

Figure 5.1: Relative errors of the variance for different times $T = 1, 2, 4, 8, 16$ and different number of random variables $K = 8, 16, 32, 64$.

5.3 Proposed Methodology

5.3.1 Formulation

To provide an efficient non-intrusive method for SDEs, we propose to evolve quadrature rules of the solution $u(t)$ and the variables $\boldsymbol{\xi}$ to represent the solution $u(t+\epsilon)$ at future times with a sufficiently small number of quadrature points. As we discussed before in Section 3.2.1, $u(t+\epsilon)$ can be captured by low order polynomials in $u(t)$ provided $\epsilon > 0$ is small, therefore, the quadrature level required to integrate polynomials in $u(t)$ can be selected small. To be able to leverage this sparsity in time, we need a similar restarting mechanism proposed in Chapter 3 and Chapter 4.

Let a sequence of restart times $0 < t_j < t_{j+1} < T$, with $\Delta t = t_{j+1} - t_j$, be given. Then the number of quadrature points for $\boldsymbol{\xi}$ on each time interval $[t_j, t_{j+1}]$ can also be made small by selecting a short time horizon, i.e. $\Delta t = \epsilon$. By properties of Brownian motion, quadrature rule for $\boldsymbol{\xi}$ can be read from tables or computed only once in the offline stage. The challenge is then to compute efficient, sparse quadrature rules for the solution $u(t)$ in time. These rules are not straightforward to compute and have to be computed online since the probability distribution of $u(t_j)$ is arbitrary and evolving.

Remark 5.4. This approach is similar to DgPC (Algorithm 2), which computes orthogonal polynomials of the solution in time to perform Galerkin projection. The major difference is that DgPC propagates orthogonal polynomials of the solution whereas this approach propagates quadrature formulas corresponding to the measures of the solution in time. For classical Gauss quadratures, quadrature formulas and associated orthogonal polynomials are closely related [36]. Both polynomial-based and quadrature-based approaches leverage the regularity of the solution in the input randomness while propagating characteristic information of the measures.

We denote by u_j the approximation of $u(t_j)$ entailed by the algorithm at t_j and by $\boldsymbol{\xi}_j = (\xi_{j,1}, \ldots, \xi_{j,K})$ the variables for the random forcing on the interval $[t_j, t_{j+1}]$. Suppose for now that sparse quadrature rules $\{w_j^p, u_j^p\}_{p=1}^{Q_{u_j}}$ and $\{\mathbf{w}_j^q, \boldsymbol{\xi}_j^q\}_{q=1}^{Q_{\boldsymbol{\xi}_j}}$ have already been constructed for u_j and $\boldsymbol{\xi}_j$ at time t_j, respectively. An analog of the equation (5.2) can be written for the approximate solution $u_{j+1}(t; u_j, \boldsymbol{\xi}_j)$ for $t \in [t_j, t_{j+1}]$

$$\begin{aligned} u_{j+1}(\tau_{j,i+1}; u_j^p, \boldsymbol{\xi}_j^q)) = u_{j+1}(\tau_{j,i}; u_j^p, \boldsymbol{\xi}_j^q)) + \mathcal{L}(u_{j+1}(\tau_{j,i}; u_j^p, \boldsymbol{\xi}_j^q))\, dt \\ + \sigma \sum_{k=1}^{K} \xi_{j,k}^q \int_{\tau_{j,i}}^{\tau_{j,i+1}} m_{j,k}(\tau)\, d\tau, \end{aligned} \tag{5.3}$$

where $m_{j,\cdot}(t)$ is a complete orthonormal system for $L^2[t_j, t_{j+1}]$ and $\tau_{j,\cdot}$'s denote a time discretization for the interval $[t_j, t_{j+1}]$.

The question here is how to construct an efficient quadrature rule for the next solution variable u_{j+1} using (5.3). The evolution of the particles $\{u_j^p, \boldsymbol{\xi}_j^q\}$ via the equation (5.3) entails a set $\{u_{j+1}^{p\times q}\}_{p,q}$ of particles of the approximate solution u_{j+1}, i.e. the quadrature nodes at t_j follow the trajectories of the dynamics and give rise to an initial set of nodes at t_{j+1}. The key challenge is then to find a small subset of these nodes and corresponding weights which accurately integrate polynomials in u_{j+1}. Following [4; 5] and [103], we construct such a sparse quadrature for u_{j+1} by using the following L^1 optimization procedure.

Let the particles $u_{j+1}^{p\times q} := u_{j+1}(t_{j+1}; u_j^p, \boldsymbol{\xi}_j^q)$, where $p = 1, \ldots, Q_{u_j}$ and $q = 1, \ldots, Q_{\boldsymbol{\xi}_j}$, serve as an initial set of quadrature nodes for u_{j+1}. Let also the set $\{T_{\boldsymbol{\alpha}}(u) : \boldsymbol{\alpha} \in \mathcal{J}_{d,N}\}$ be any orthonormal basis of polynomials up to degree N in dimension d. Then to extract a sparse quadrature rule at t_{j+1}, we solve the convex optimization problem:

$$\min_{w \in \mathbb{R}^{\tilde{Q}_{u_{j+1}}}} \quad ||w||_1, \quad \text{subject to } Aw = b. \tag{5.4}$$

Here $w \in \mathbb{R}^{\tilde{Q}_{u_{j+1}}}$ with $\tilde{Q}_{u_{j+1}} = Q_{u_j} \times Q_{\boldsymbol{\xi}_j}$, and the constraints $Aw = b$ necessitate the exactness of the quadrature rule up to total degree $|\boldsymbol{\alpha}| \leq N$. The corresponding quadrature level λ_u in Gaussian quadrature sense is $(N+1)/2$ assuming N is odd.

We enumerate the basis $T_{\boldsymbol{\alpha}}(u)$ and denote it by $\{T_k(u) : k = 0, \ldots, M\}$, where $M = |\mathcal{J}_{d,N}|$, and then define

$$A := \begin{bmatrix} T_0(u_{j+1}^1) & \ldots & T_0(u_{j+1}^{\tilde{Q}_{u_{j+1}}}) \\ \vdots & & \vdots \\ T_M(u_{j+1}^1) & \ldots & T_M(u_{j+1}^{\tilde{Q}_{u_{j+1}}}) \end{bmatrix},$$

and the right-hand side vector consisting of the exact moments

$$b := \begin{bmatrix} \mathbb{E}[T_0(u_{j+1})] & \ldots & \mathbb{E}[T_M(u_{j+1})] \end{bmatrix}^T.$$

We assume that the moments $\mathbb{E}[u_{j+1}^{\boldsymbol{\alpha}}]$, for each t_{j+1} and $|\boldsymbol{\alpha}| \leq N$, are finite and we typically have that M is much smaller than $\tilde{Q}_{u_{j+1}}$.

Then a sparse subset denoted by $\{w_{j+1}^p, u_{j+1}^p\}_{p=1}^{Q_{u_{j+1}}}$, with $Q_{u_{j+1}} \leq M \ll \tilde{Q}_{j+1}$, can be extracted having at most M nodes from the solution of the optimization procedure (5.4); see Section 5.3.2 for possible implementations. Once a quadrature rule for u_{j+1} is constructed, the algorithm restarts and evolves the quadrature nodes on the next time interval according to (5.3); see Algorithm 3.

Remark 5.5. It is useful to note that finding a quadrature rule which is exact for polynomials up to degree N amounts to computing a discrete approximation to the measure of u_j which has the same moments, i.e. the summation of Dirac measures $\sum_{p=1}^{Q_j} w_j^p \, \delta_{u_j^p}$ and the continuous measure of u_j have the same moments up to degree N.

Remark 5.6. For probability measures on $\mathbb{R}^d$, the existence of a quadrature rule with positive weights and a degree of exactness $2\lambda - 1$ is guaranteed; [115; 100; 24; 36]. In general,

Algorithm 3 Dynamical Sparse Grid Collocation (DSGC) method for SDEs

```
Decompose the time domain $[0,T] = [0,t_1] \cup \ldots \cup [t_{n-1},T]$
Select a time-integration method
Initialize the degrees of freedom $K, N$
Compute quadrature rules for $\boldsymbol{\xi}_0$ and $u_0$
for each time-step $t_j$, $j > 0$, do
    evolve the quadrature nodes $u_{j-1}^p$ and $\boldsymbol{\xi}_{j-1}^q$ by (5.3)
    obtain the nodes $u_j^{p\times q}$, $p = 1,\ldots,Q_{u_{j-1}}$ and $q = 1,\ldots,Q_{\boldsymbol{\xi}_{j-1}}$
    solve the optimization procedure (5.4)
    extract a sparse quadrature rule $\{w_j^p, u_j^p\}_{p=1}^{Q_{u_j}}$
end for
```

we do not enforce positivity condition for the weights w since a sparse optimal solution with positive weights may not exist; see Section 5.3.2 for further details. We note that (5.4) is not the only construction to find optimal quadrature rules; see also [46; 24; 4; 5] and references therein. Furthermore, the convergence of exact quadrature rules for compactly supported probability measures has been studied extensively in classical literature [25; 36].

Remark 5.7. We do not tensorize quadrature rules for each component of d-dimensional random vector u_j. It is quite possible that components of u_j exhibit correlation; therefore tensorization is not optimal. However, since we impose constraints on multivariate moments $\mathbb{E}[u_j^{\boldsymbol{\alpha}}]$, the algorithm automatically establishes a quadrature rule for the full vector u_j. Moreover, if the dimension d is high, the number of constraints M in (5.4) can be reduced by considering a sparse version of the multi-index set $\mathcal{J}_{d,N}$; see (4.4).

Remark 5.8. This remark concerns the differences and similarities of our approach to the methods in [132; 133] and optimal quantization methods in [95; 76].

Our method uses pre-determined quadrature rules for the random forcing variables and does not a have sampling error, which are similar to the method in [132]. The main difference is that the paper first discretizes the stochastic equations in time and then considers quadrature rules for the random forcing variables in each time-step, i.e. the dimension of randomness depends on the resolution of the time-integration and might grow rapidly with fine discretization. In contrast, our method discretizes in the random space by considering the L^2-approximation (2.10) of Brownian motion with a fixed degree of freedom K. Although a different restarting mechanism is used in [132], their method can only compute moments up to second order, whereas approximations to higher order moments are available in our method by quadrature rules provided higher moments converge.

Our method finds optimal quadrature rules adapted to the evolving solution in a similar sense to optimal quantization methods. Quantization methods [95; 76] aim to discretize the paths of an infinite dimensional randomness by a random vector in finite dimension. Finite dimensional approximations are obtained by the solution of a minimization procedure and are typically given by Voronoi cell collocation. For Brownian motion, quantizers based on its KLE are considered. Then evolution of these quantizers for SDEs is obtained by solving a corresponding integral equation, which is similar to (5.3). The typical convergence order is logarithmic in the number of quantizers, which is a poor rate of convergence for practical applications. In contrast, our method utilizes Gaussian-type quadratures tailored for the solution and the random forcing, which leverage the smoothness of the response to provide fast convergence. Moreover, frequent restarts allow us to mitigate dimensionality and use low-dimensional approximations to Brownian motion forcing.

5.3.2 Implementation

5.3.2.1 Offline stage

In the offline stage, certain quadratures need to be computed. First, we compute sparse quadrature rule for Gaussian $\boldsymbol{\xi}_0$ by using the Smolyak sparse grid with the level $\lambda_{\boldsymbol{\xi}_0}$, which builds upon the standard 1D Gauss-Hermite rule; see Section 2.3.1. By independent increment property of Brownian motion, all $\boldsymbol{\xi}_j$, $j \geq 0$, may have the same quadrature rule assuming K and the level are fixed. Note that although it is not necessary, we keep the number of variables in $\boldsymbol{\xi}$ the same throughout the evolution. If the distribution of the initial condition u_0 is known, a sparse Gauss quadrature is constructed with the level λ_{u_0}. If its distribution is arbitrary, then the optimization procedure (5.4) can be used with Monte Carlo initialization. We make use of the C++ library "UQ Toolkit" to compute Gauss rules [26].

5.3.2.2 Moments and orthogonal polynomials

At the restart time t_{j+1}, estimation of the right-hand side vector in the constraints in the optimization problem (5.4) requires the calculation of the multivariate moments $\mathbb{E}[u_{j+1}^{\boldsymbol{\alpha}}]$, where $u_{j+1} = (u_{j+1,1}, \ldots, u_{j+1,d})$ and $|\boldsymbol{\alpha}| \le N$. These moments are computed using already available quadrature rules $\{w_j^p, u_j^p\}_{p=1}^{Q_{u_j}}$ and $\{\mathbf{w}_j^q, \boldsymbol{\xi}_j^q\}_{q=1}^{Q_{\boldsymbol{\xi}_j}}$ from time t_j:

$$\mathbb{E}[u_{j+1}^{\boldsymbol{\alpha}}] = \mathbb{E}_{(u_j,\boldsymbol{\xi}_j)}[u_{j+1}^{\boldsymbol{\alpha}}(u_j, \boldsymbol{\xi}_j)] \approx \sum_{p=1}^{Q_{u_j}} \sum_{q=1}^{Q_{\boldsymbol{\xi}_j}} w_j^p \, \mathbf{w}_j^q \prod_{i=1}^{d} (u_{j+1,i}(u_j^p, \boldsymbol{\xi}_j^q))^{\alpha_i}.$$

The optimization procedure does not depend on the particular choice of the set $\{T_{\boldsymbol{\alpha}}(u) : \boldsymbol{\alpha} \in \mathcal{J}_{d,N}\}$ of polynomials, e.g. even monomials can be used. However, the choice of $T_{\boldsymbol{\alpha}}$ certainly affects the condition number of the constraint matrix, which in turn affects the stability of the numerical minimization algorithm. To better condition the constraint matrix and improve the convergence of the optimization algorithm, we make use of couple of linear transformations as preconditioning steps. Similar transformation techniques are applied before in the setting of moment-constrained maximum entropy problems [1].

A linear transformation is applied to the solution u_j so that it becomes mean zero. Then a further transformation makes its components uncorrelated, i.e. its covariance matrix becomes identity. Even with these transformations, a scaling issue related to moments arises in the constraint equations. For instance, for a standard Gaussian random variable ξ, we have $\mathbb{E}[\xi^{12}]/\mathbb{E}[\xi^2] = 10395$, i.e. the twelfth moment is larger than the second moment by 5 orders magnitude. To alleviate this scaling issue, we further scale u by its largest moment so that maximum moment becomes 1.

A more direct preconditioning can be applied by a judicious selection of the orthonormal basis $T_{\boldsymbol{\alpha}}$. A basis can be selected using an educated guess in the offline stage, which does not require any online computation. However, since the measure of u_j is evolving in time, this may not be optimal in the long-time in Algorithm 3. An optimal choice for $T_{\boldsymbol{\alpha}}$ is the set of polynomials which are orthogonal with respect to the measure of u_j. Unfortunately, corresponding orthogonal system for u_j is not available a priori in the algorithm, but it can be computed online if further preconditioning is required. Although the computation of orthogonal polynomials is an ill-posed problem, a Gram-Schmidt procedure based on the knowledge of multivariate moments can be used in the computation; see (4.11). In numerical simulations, the choice of the orthonormal system will be explicitly stated.

Here is a simple demonstration of the effects of these transformations. Let ξ_1 and ξ_2 be two independent $N(3,1)$ variables and the maximum degree be $N = 8$. Then, the number of constraints becomes $M = 45$. We use 500 samples from normal distribution to initialize the optimization procedure and keep the samples same for each scenario to make a fair comparison. A sparse quadrature rule with M nodes is extracted according to the algorithm discussed in the next section, and afterwards, the right-hand side vector b is computed numerically using this quadrature rule to check the accuracy. The numerical approximation of b is denoted by $\tilde{b}$ in the following.

Table 5.1 shows condition numbers of the linear system in (5.4) and the accuracy $||b - \tilde{b}||_\infty/||b||_\infty$ of the associated quadrature rule. First two scenarios use monomials as the polynomial basis $T_{\boldsymbol{\alpha}}$ and the last one uses Hermite polynomials, which are the associated orthogonal system in this example. Note finally that condition numbers are independent of the sparse extraction procedure. Clearly, scaling transformations or a careful selection of the polynomials basis leads to at least 5-digit gain of accuracy in this example.

	Without scaling	With scaling	Hermite poly.
cond(A)	7.01E+9	1.91E+3	2.98E+2
$\lvert\lvert b - \tilde{b}\rvert\rvert_\infty/\lvert\lvert b\rvert\rvert_\infty$	3.25E-8	3.95E-13	3.19E-14

Table 5.1: Comparison of the accuracy of quadrature rules for two independent Gaussian variables using different transformations.

5.3.2.3 Sparse quadrature rules

Algorithm 3 constructs dynamical quadrature rules in time for the solution u_j. However, implementation of the optimization algorithm (5.4) to construct an efficient quadrature rule is not straightforward. From (5.4), we observe that at each restart t_j, the optimization procedure is initialized with $\tilde{Q}_{u_j} = Q_{u_{j-1}} \times Q_{\boldsymbol{\xi}_{j-1}}$ number of nodes for u_j. Therefore, the number of quadrature nodes may grow with the number of restarts. The challenge is then to compute a sparse quadrature rule containing a smaller set of nodes and weights while keeping the exactness of the original quadrature rule. To this end, after finding the optimal solution of (5.4), we further employ an extraction routine.

A straightforward sparsification of the optimal solution of (5.4) would be cutting-off the weights that are greater than a certain threshold. Depending on the numerical minimization algorithm, this may not be possible. As discussed in [4; 5], an application of the simplex algorithm yields a sparse optimal solution whereas interior-point methods give a fully populated solution [88]. We choose to use CVX, a package for specifying and solving convex programs [48; 47]. Under the hood, CVX uses SDPT3 which employs interior-point methods to compute the optimal solution [116]. The following procedure is used to extract a sparse quadrature rule; see [4; 5; 88].

At time t_j, the constraints matrix A is of dimension $M \times \tilde{Q}_{u_j}$, where $\tilde{Q}_{u_j}$ is much bigger than M. Thus, the dimension of the nullspace of A is at least $\tilde{Q}_{u_j} - M$. The key observation here is that any vector $z \in \text{null}(A)$ can be added to the weights without changing the equality constraints, i.e. $A(w+z) = b$. Thus, by selecting vectors carefully, we can construct an iterative routine to make most of the weights zero. Based on these observations, we follow the approach given in [4; 5; 88] and employ the following routine at each restart t_j.

Algorithm 4 Sparse Quadrature Extraction Routine for (5.4)

```
Initialize with the optimal weights $w \in \mathbb{R}^{\tilde{Q}}$
**repeat**
    find the indices $\mathcal{N} := \{k \in \{1, \dots, \tilde{Q}\} \,:\, w_k = 0\}$
    find $z \in \text{null}(A)$ with $z_k = 0$, $k \in \mathcal{N}$
    set $\beta = \min\{|\frac{w_k}{z_k}| \,:\, k \notin \mathcal{N}, \text{sign}(z_k) \neq \text{sign}(w_k)\}$
    set $w = w + \beta z$
**until** the number of nonzeros in $w$ is less than or equal to $M$
```

This routine allows us to find a sparse quadrature rule $\{w_j^p, u_j^p\}_{p=1}^{Q_{u_j}}$ with the number of nodes Q_{u_j} satisfying $Q_{u_j} \leq M \ll \tilde{Q}_{u_j}$. Thus, the number of nodes can be made independent of time and frequent restarts can be used in Algorithm 3. A one way to find a vector z in the nullspace of A is to compute a basis for the nullspace. In the implementation, we make use of the QR method to quickly select a vector at each iteration. Finally, we note that this routine does not necessarily yield nonnegative weights.

An application of this procedure to the two dimensional Gaussian random variable discussed before in Table 5.1 reduces the size of the quadrature rule from 500 to 45 while the accuracy remains almost the same as 3.19E-14.

5.4 Numerical Experiments

In this section, we present several numerical simulations of Algorithm 3 using low-dimensional nonlinear SDE models.

For the rest of the section, $T \in \mathbb{R}$ stands for the endpoint of the time interval while $\Delta t = T/n$ denotes the time-step after which restarts occur at $t_j = j\Delta t$. Furthermore, we choose orthonormal bases for $L^2[t_j, t_{j+1}]$ as cosines (3.22). To solve the equation (5.3), we utilize either a first- or a second-order time integration method.

In our numerical examples, the dynamics converge to an invariant measure. To demonstrate the convergence behavior of our algorithm, we compare our results to exact second order statistics or Monte Carlo simulations with sufficiently high number of samples M_{samp}. We also demonstrate the evolution of relative pointwise errors (3.23) computed at each restart. In some cases, we give estimations of first six cumulants of the invariant measure.

Example 5.9. As a first example we consider an OU process

$$du(t) = b_u\,(\mu - u(t))\,dt + \sigma_u\,dW(t), \quad t \in [0,T], \quad u(0) = u_0, \tag{5.5}$$

where the damping parameter is random and uniformly distributed in $[1,3]$, i.e. $b_u \sim U(1,3)$. This is an example of 2-dimensional non-Gaussian dynamics that may be seen as a coupled system for (u, b_u) with $db_u = 0$; see also Example 3.12.

For the first simulation, we consider the time domain $[0,4]$. The mean-return parameter μ is set to be 0.2. The initial condition is normally distributed $u_0 \sim N(1, 0.04) \perp\!\!\!\perp W(t)$ and $\sigma_u = 4$. We use the Gauss-Hermite rule for the initial condition with the level $\lambda_{u_0} = 3$, whereas for the damping parameter b_u, we use the Gauss-Legendre quadrature rule with a varying λ_{b_u}. For Brownian motion, we use 2-dimensional approximation with the product Gauss-Hermite rule of level $\lambda_{\boldsymbol{\xi}} = 2$. We also take the set $T_{\boldsymbol{\alpha}}(u)$ as Hermite polynomials. For time integration, we utilize a second-order weak Runge-Kutta method with $dt = 5\text{E-}4$.

In Figure 5.2, we compare second order statistics obtained by our method to the exact solutions using $N = 1, 2$, and 3, $\lambda_{b_u} = 2, 4$, and 8, and $\Delta t = 0.4, 0.2, 0.1$, and 0.05. The results are obtained by calculating the moments in u variable and then taking averaging with respect to the measure of b_u. The rows of the figure correspond to N-, λ_{b_u}-, and Δt-convergence of the method while keeping the other two degrees of freedom constant. For each different restart step Δt, we keep the time-integration step dt the same.

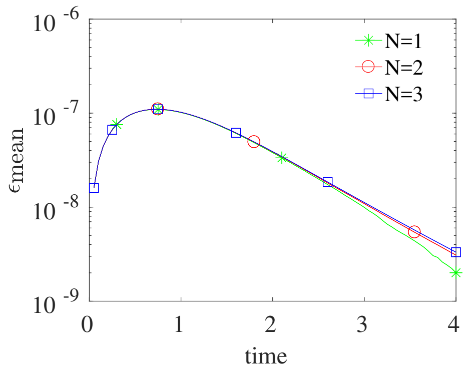
N-convergence of the method in Figure 5.3a and Figure 5.3b shows that the first two moments can be captured accurately with $N = 2$. The level of accuracies are $O(10^{-9})$ and $O(10^{-5})$ for the mean and the variance, respectively. We notice that using a larger quadrature level λ_{b_u} and more frequent restarts also help to reduce the relative errors. We also observe that the convergence behavior of the method in terms of the size of time interval is at least quadratic in the variance; i.e. $O(\Delta t^2)$.

Table 5.2 exhibits the first six cumulants of the equation (5.5) with $\mu = 0.0$ in the long-time $T = 8$. The limiting stationary measure can be obtained by solving the corresponding standard Fokker-Planck equation; see [16; 64; 90]. The first 6 cumulants, κ_i, $i = 1, \ldots 6$, are obtained by computing moments in the solution variable and then averaging with respect to the damping parameter. The table shows that increasing the degree N of polynomials in the constraints in (5.4) clearly helps to accurately capture the higher cumulants in the long-time. We also note that the approximations for the cumulants κ_5 and κ_6 become accurate when $N = 6$ is used while they are inaccurate for $N = 4$. This type of convergence behavior is related to the fact that the equation is linear in this case.

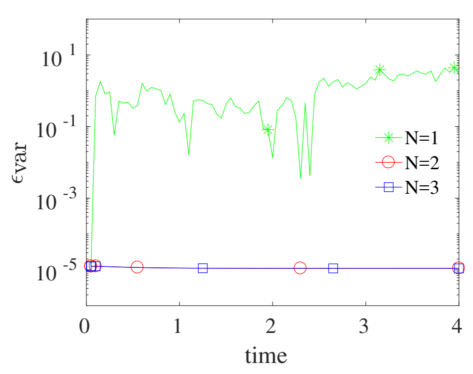
Example 5.10. We now consider a nonlinearity in the equation so that the damping term includes a cubic term:

$$du(t) = -(u^2(t) + 1)\, u(t)\, dt + \sigma_u\, dW(t), \quad u(0) = 1.$$

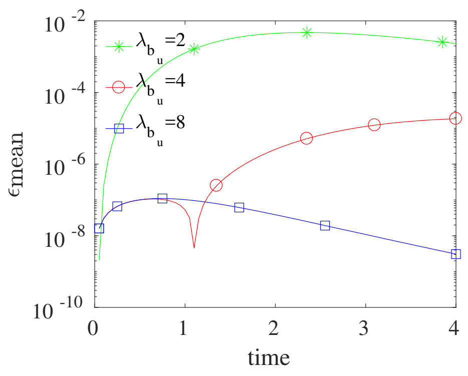
We take $T = 4$, $\Delta t = 0.04$, and $\sigma_u = 2$. For Brownian motion we use $K = 2$ dimensional vector with a sparse Gauss-Hermite rule of level $\lambda_{\boldsymbol{\xi}} = 3$. We also take the set $T_{\boldsymbol{\alpha}}(u)$ as



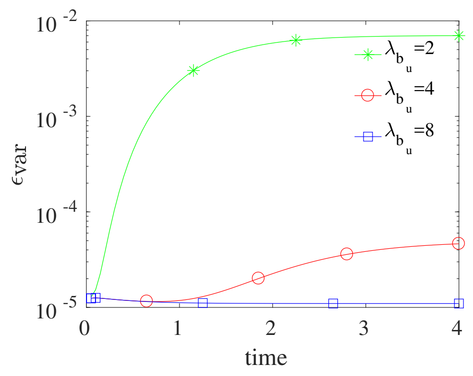
(a) $\lambda_{b_u} = 8$, $\Delta t = 0.05$



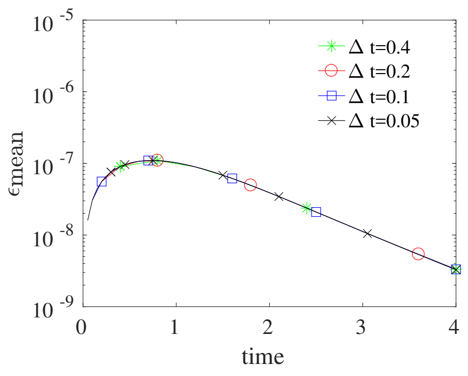
(b) $\lambda_{b_u} = 8$, $\Delta t = 0.05$



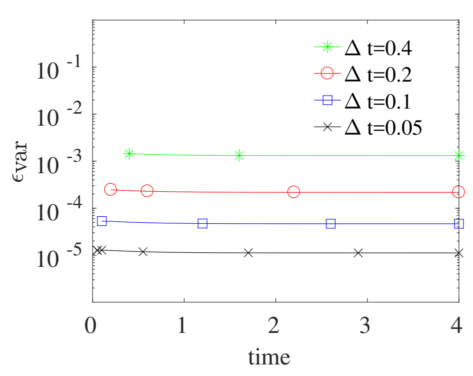
(c) $N = 2$, $\Delta t = 0.05$



(d) $N = 2$, $\Delta t = 0.05$



(e) $\lambda_{b_u} = 8$, $N = 2$



(f) $\lambda_{b_u} = 8$, $N = 2$

Figure 5.2: Convergence behaviors in N, λ_{b_u}, and Δt for the Ornstein-Uhlenbeck process with random damping.

	κ_1	κ_2	κ_3	κ_4	κ_5	κ_6
DSGC: $N = 4$	2.09E-5	4.39	1.75E-4	6.06	-7.38	76.00
DSGC: $N = 6$	2.09E-5	4.39	1.75E-4	6.06	1.96E-3	33.85
Fokker-Planck	0	4.39	0	6.06	0	33.85

Table 5.2: Cumulants obtained by Algorithm 3 and Fokker-Planck equation at $T = 8$ for the Ornstein-Uhlenbeck process with random damping.

Hermite polynomials.

In this case, we observe from Table 5.3 that the Algorithm 3 can be executed accurately in the long-time by increasing the number of degrees of freedom N. Comparing Table 5.3 and Table 5.2, we see that the accuracy in the higher cumulants are slightly decreased and it is harder to capture higher moments as this is a more complicated dynamics having a nonlinearity.

	κ_1	κ_2	κ_3	κ_4	κ_5	κ_6
DSGC: $N = 4$	1.26E-2	7.35E-1	4.12E-2	-3.22E-1	2.08E-1	4.75E-1
DSGC: $N = 6$	-2.01E-3	7.34E-1	8.20E-3	-3.39E-1	-7.27E-2	9.35E-1
DSGC: $N = 8$	3.48E-4	7.33E-1	-2.91E-3	-3.39E-1	2.40E-3	9.52E-1
Fokker-Planck	0	7.33E-1	0	-3.39E-1	0	9.64E-1

Table 5.3: Cumulants obtained by Algorithm 3 and Fokker-Planck equation at $T = 4$ for cubic nonlinearity.

Example 5.11. In this example, we consider a multiplicative noise term and the Cox-IngerSoll-Ross (CIR) model

$$du(t) = b_u(\mu - u(t))dt + \sigma_u \sqrt{u(t)}\, dW(t), \tag{5.6}$$

which is used to describe the evolution of interest rates [22]. The same model is also used in the Heston model to model random volatility. We impose the condition $2b_u\mu \geq \sigma_u^2$ so that the process stays positive. The process has a stationary distribution in the long time and the second order statistics can be computed analytically. Note also that the noise amplitude is non-Lipschitz.

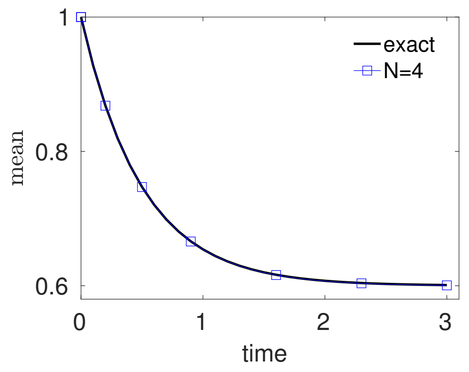
We apply the first-order Milstein method [64]

$$\begin{aligned} u(\tau_{i+1}) &= u(\tau_i) + b_u \left(\mu - u(\tau_i) - \frac{\sigma_u^2}{4b_u} \right) dt \\ &\quad + \sigma_u \sqrt{u(\tau_i)} (W(\tau_{i+1}) - W(\tau_i)) + \frac{\sigma_u^2}{4} (W(\tau_{i+1}) - W(\tau_i))^2, \end{aligned} \tag{5.7}$$

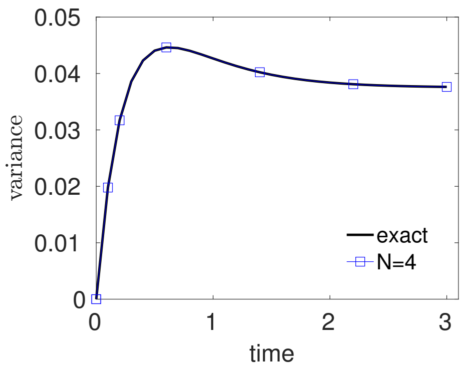
and then approximate Brownian motion increments by their finite-dimensional approximations. Since the noise term involves a function of u, it is necessary to use a time-integration method which takes this dependence into account. We set $T = 3, u_0 = 1$, $\mu = 0.6$, $\sigma_u = 0.5$, $b_u = 2$, $K = 2$, $\lambda_\xi = 4$, and $\Delta t = 0.1$. We compute the associated orthogonal polynomials when we solve the optimization procedure.

The first row of Figure 5.3 shows the evolution of the mean and the variance of the model (5.6) obtained by the analytical solution and Algorithm 3 with $N = 4$. We observe that in the long time, the statistics become stationary and are captured well by the algorithm. The second row shows that the algorithm can be performed accurately in the long-time with a relatively small noise-level $\sigma_u = 0.5$.

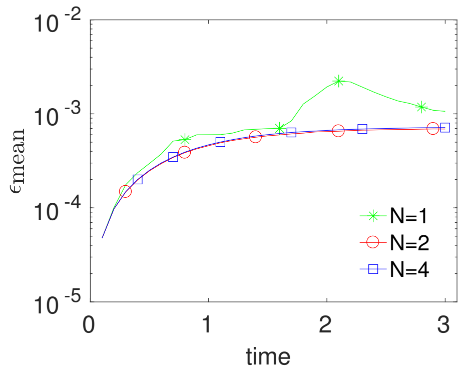
In Table 5.4, we compare the DSGC algorithm and MC method in terms of accuracy and timing. For a fair comparison and to minimize the error of time integration, both methods use Milstein's method with the time step dt = 1E-4. We compare the convergence behaviors of the variance using N and M_{samp} for DSGC and MC, respectively. To assess the robustness of the methods, we take $b_u = 4$ and consider different magnitudes of the noise: $\sigma_u = 0.5, 1$, and 2. Since MC estimates are noisy, we repeat each MC simulation 20 times and average the errors. We first observe that the accuracy of both methods drops when σ_u is increased. Also, if high accuracy is needed, DSGC algorithm can be executed faster than MC in this scenario, e.g., for $\sigma_u = 1$ and $\sigma_u = 2$, MC should use at least M_{samp} = 128E+4 to get to the same level of accuracy of DSGC with $N = 4$, which implies that DSGC is 20 times faster with the same accuracy. We also see that DSGC is very efficient for small magnitude of noise $\sigma_u = 0.5$. The elapsed times for DSGC seem to scale quadratically with N. Note also that the convergence of MC method is guaranteed, however, although we observe numerical convergence in DSGC, we do not have



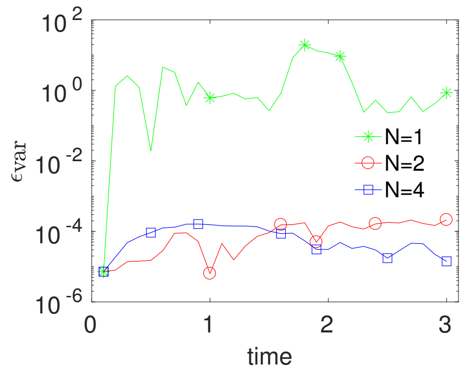
(a) $N = 4$ approximation to the mean



(b) $N = 4$ approximation to the variance



(c) ϵ_{mean} for $N = 1, 2, 4$



(d) ϵ_{var} for $N = 1, 2, 4$

Figure 5.3: Evolution of the mean and the variance of the CIR model, and convergence behaviors in N.

a convergence rigorous result for the approximate solution of (5.7). Similar settings of [95; 96; 76] establish convergence results for SDEs with sufficiently smooth coefficients, which do not apply in this case.

Example 5.12. This example concerns the 2-dimensional nonlinear system

$$du(t) = -(b_u + a_u v(t))u(t)\, dt + \sigma_u\, dW_u(t),$$
$$dv(t) = -b_v\, v(t)\, dt + \sigma_v\, dW_v(t),$$

where $a_u > 0$, $b_u, b_v > 0$ are damping parameters, $\sigma_u, \sigma_v > 0$ are constant noise amplitudes, and W_u and W_v are two real independent Brownian motions. Depending on the regime of

Algorithm	$\sigma_u = 0.5$	$\sigma_u = 1$	$\sigma_u = 2$	Time ratio
DSGC $N = 3$	1.2E-4	2.2E-3	5.7E-3	0.5
DSGC $N = 4$	2.9E-5	1.5E-3	1.9E-3	0.8
DSGC $N = 5$	1.5E-5	1.2E-3	1.3E-3	1.3
DSGC $N = 6$	1.1E-5	6.3E-4	6.4E-4	2.0
MC $M_{\text{samp}} =$1E+4	1.21E-2	2.01E-2	2.46E-2	0.125
MC $M_{\text{samp}} =$2E+4	7.7E-3	1.30E-2	1.74E-2	0.25
MC $M_{\text{samp}} =$4E+4	5.7E-3	1.00E-2	1.20E-2	0.5
MC $M_{\text{samp}} =$8E+4	3.5E-3	6.03E-3	7.8E-3	1.0

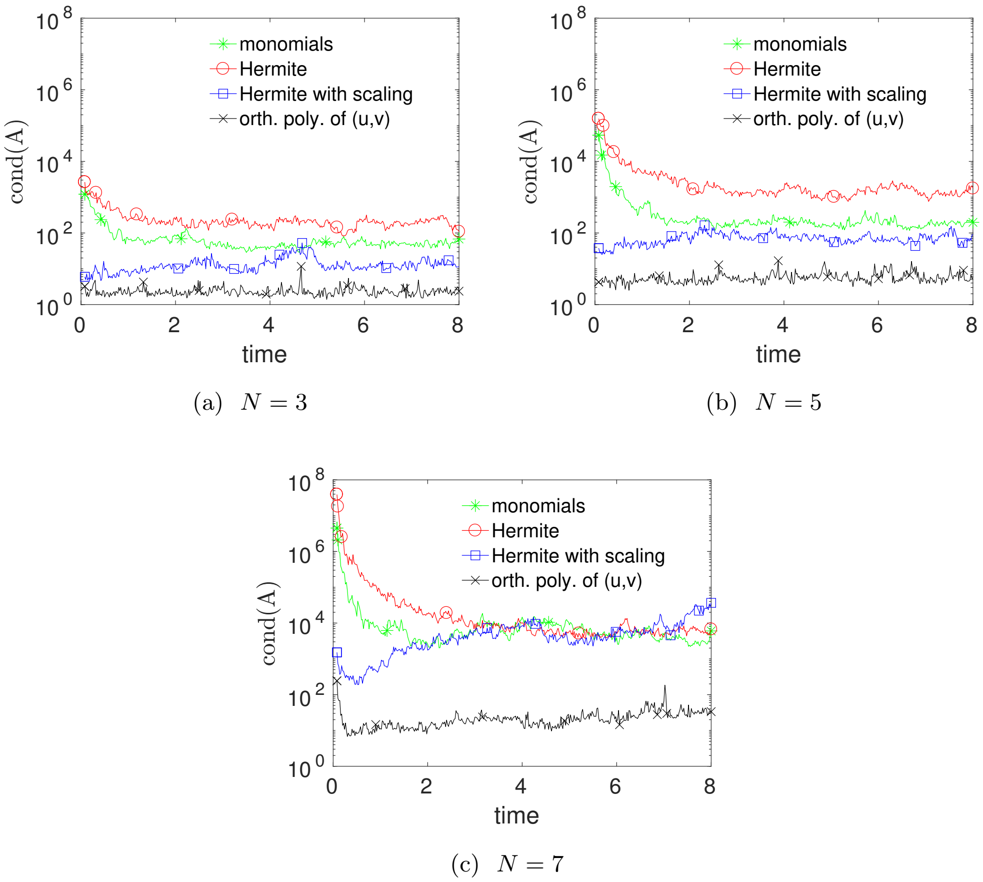
Table 5.4: Errors of the variance at $T = 1$, and relative timings of DSGC and MC methods using different degrees of freedom.

the parameters, the dynamics of the solution u exhibit intermittent non-Gaussian behavior; see detailed discussions in [38; 16] and also Section 3.3.

Following [16], we consider the system parameters as $a_u = 1$, $b_u = 1.2$, $b_v = 0.5$, $\sigma_u = 0.5$, $\sigma_v = 0.5$, and take the initials $u_0 = N(1, \sigma_u^2/8b_u) \perp\!\!\!\perp v_0 = N(0, \sigma_v^2/8b_v)$. In this regime, the dynamics of u are characterized by unstable bursts of large amplitude and possess a fat-tailed long-time distribution.

We consider a long-time $T = 8$ and use $K = 4$ variables for the random forcing terms with the quadrature level $\lambda_{\boldsymbol{\xi}} = 2$. A sparse Gauss-Hermite quadrature rule is used for the random initial conditions. Since we use a small quadrature level $\lambda_{\boldsymbol{\xi}}$, we restart frequently and take $\Delta t = 0.02$ in the following simulations. The ability to use small degrees of freedom with frequent restarts is one of the main advantages of the method.

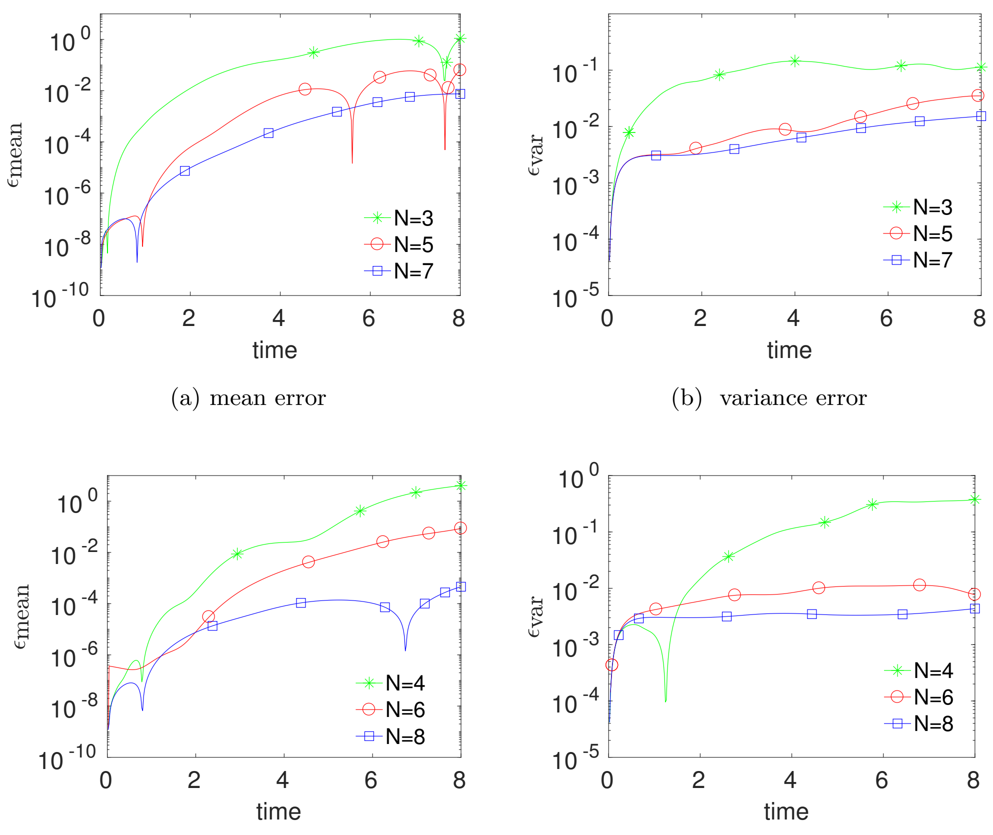
The first simulation concerns the choice of the polynomials $T_{\boldsymbol{\alpha}}(u, v)$. Figure 5.4 shows the condition numbers of the constraint matrix A in (5.4) for different choices of polynomials and varying N. We observe that in each case, the lowest condition numbers correspond to the ones which are obtained by orthogonal polynomials with respect to the joint distribution of (u, v). Although the computation of orthogonal polynomials for large degree of polynomials is unstable, once the computation is carried out, it yields a well-conditioned constraint matrix for fixed N. Moreover, increasing the degree of polynomials N leads to overall larger condition numbers, which might affect the stability of the numerical minimization procedures for large N.



(a) $N = 3$ (b) $N = 5$ (c) $N = 7$

Figure 5.4: Condition numbers of the constraint matrix A in the optimization procedure (5.4) for different degrees of freedom N and different choices of $T_{\boldsymbol{\alpha}}$.

Using polynomials $T_{\boldsymbol{\alpha}}(u, v)$ which are orthogonal with respect to the joint distribution of the solution, we next demonstrate N-convergence of the method in Figure 5.5. We observe from the figure that the mean and the variance can be captured in the long-time up to $O(10^{-3})$ of accuracy. Throughout the time evolution, the number of quadrature points of the joint distribution of (u, v) is $\binom{N+2}{2}$. For instance, for $N = 5, 6, 7,$ and 8, this number becomes $21, 28, 36,$ and 45, respectively. It is useful to note that although increasing N gives better errors, it also makes the numerical optimization procedure relatively unstable.



(a) mean error (b) variance error

(c) mean error (d) variance error

Figure 5.5: N-convergence of the method for the nonlinear system of SDEs.

Finally, we take a relatively short time $T = 1$, and compare the computational times and the accuracy of DSGC, DgPC, and MC methods. All methods use a second order time-integration method with the same time step $dt = $ 2E-4. MC simulations are repeated 20 times to get a stable estimate of the error. Note that it is getting computationally harder to obtain a stable estimate in MC for longer times since the estimates are noisy. Both DSGC and DgPC use the restart step $\Delta t = 0.05$. DgPC algorithm uses the sampling approach discussed in 4.2.

From Table 5.5, we first notice that for the same level of accuracy, DSGC performs better than both methods in terms of computational time. This behavior is due to the fact that the method can utilize a small number of samples and frequent restarts to achieve a high accuracy in a fast manner. We also observe that DgPC with $N = 2$ offers a sufficient accuracy for many applications with a longer computational time in this scenario. Thus, DSGC offers a relatively fast way to propagate samples of the solution in this low-dimensional case.

For high-dimensional cases, we do not expect the current DSGC algorithm outperforms DgPC algorithm. As we established in Chapter 4, although DgPC is an intrusive method, it provides fast and accurate long-time simulations for SPDEs compared to sampling methods with the help of Galerkin projection and propagation of orthogonal polynomials. To achieve a similar long-time accuracy of DgPC, sampling methods need to use a large number of samples. In that case, DSGC has an advantage over MC since it leverages the regularity of the solution. All in all, if a high accuracy is needed, one might prefer DSGC in low-dimensional cases and DgPC can be used for high-dimensional dynamics.

We conclude with a remark about an extension of DSGC to high-dimensional dynamics.

Algorithm	ϵ_{mean}	ϵ_{var}	Time ratio
DSGC $N = 3$	4.6E-4	3.01E-2	0.095
DSGC $N = 5$	9.2E-6	4.5E-3	0.2
DgPC $N = 2$	3.8E-4	3.7E-3	2.28
DgPC $N = 3$	3.2E-4	1.8E-3	6.71
MC $M_{\text{samp}} =$4E+4	4.3E-3	6.2E-3	0.5
MC $M_{\text{samp}} =$8E+4	2.6E-3	4.8E-3	1.0

Table 5.5: Errors of the mean and the variance at $T = 1$, and relative timings of DSGC, DgPC, and MC methods using different degrees of freedom.

Remark 5.13. For high-dimensional SDEs or SPDEs, the DSGC algorithm needs considerable modifications. Dynamical sparse quadrature rules for the solution will have higher degrees of freedom, which, in turn, may hinder the efficiency of the algorithm; especially in case of complex dynamics. Thus, further dimensionality reduction techniques and simple parallelization can be considered. Moreover, the stability of the constrained optimization in high-dimensional cases should also be investigated. We demonstrated such an extension from SDEs to SPDEs in Chapter 4 in case of DgPC. A similar extension can be done for Algorithm by 3 using the KLE of random solutions in time and constructing sparse quadrature rules for the selected finite number of KLE random modes.

Chapter 6

Conclusion

Polynomial chaos expansions applied to the simulation of evolution equations driven by time-dependent stochastic forcing suffer from two drawbacks: the dimension of randomness is too large, and the long-time solution may not be sparsely represented in a fixed PCE basis. In the setting of Markovian random forcing, we proposed a restart method that addresses these two drawbacks. Such restarts of the PCE, which we called here Dynamical generalized Polynomial Chaos, allow us both to keep the number of random variables small and to obtain a solution that remains reasonably sparse in the evolving basis of orthogonal polynomials—things that can not be achieved without restarts.

We applied DgPC to the long-time numerical simulation of various nonlinear stochastic differential equations driven by Brownian motion. To demonstrate the ability of the algorithm to reach long-time solutions, we computed invariant measures for SDEs that admit one, and found a very good agreement between DgPC and other standard methods such as Monte Carlo-based simulations. We also presented a simple theoretical justification for such a convergence in Chapter 3.

The main computational difficulty in DgPC, as in most gPC-based methods, is the estimation of the orthogonal polynomials. Our method is based on estimating moments of the multivariate distribution of interest and constructing orthogonal polynomials of evolving arbitrary measures by a Gram–Schmidt procedure. The bottleneck in such estimations is the cost of computing moments. This is similar to the cost of solving differential equations by PCE methods, where moment estimations are also the most costly step. However, in

DgPC, such estimations cannot be performed offline.

From a theoretical point of view, we need to ensure that the evolving measures remain determinate so that their orthogonal polynomials span square integrable functionals. Since distributions with compact support are determinate, our theoretical results have been applied in the setting where SDE solutions are well approximated by compactly supported distributions. In this connection, we note that the calculation of orthogonal polynomials from knowledge of moments is an ill-posed problem, which is a serious impediment to gPC methods in general.

In Chapter 4, we extended DgPC algorithm to tackle long-time integration and high-dimensional randomness in the context of PDEs with Markovian forcing. To deal with infinite dimensionality, the Karhunen–Loeve expansion is employed at each restart to find a representation of the solution which corresponds to a low-dimensional dynamics of the pertaining physical processes. The most influential modes are then incorporated in a PCE for the future evolution. Using sparse multi-index sets and frequent restarts, the algorithm provides an efficient way to capture the solutions in a fairly sparse random bases in terms of orthogonal polynomials of dynamical measures.

The computational bottlenecks of the algorithm for SPDEs are the simulation of the deterministic evolution equations, the KLE, and the computation of moments. The cost of the deterministic evolution is dictated by the complicated nature of the SPDEs. The KLE is an expensive dimensionality reduction technique. Since the algorithm constructs PCEs online, the KLE (or other dimensionality reduction techniques) is unavoidable at each restart. We found that for large covariance matrices, the KLE cost was drastically reduced when the covariance matrix was estimated by a low-rank approximation obtained by random projections. The estimation of the orthogonal polynomials and corresponding triple products of evolving arbitrary measures is also a costly step.

Using 1D Burgers equation and 2D Navier–Stokes system both driven by Brownian motion, we provided several numerical simulations for both short- and long-time solutions. We compared the accuracy and computational time of the algorithm to the standard Monte Carlo method and found that the proposed algorithm achieved similar error levels for a (generally significant) lower computational cost in most cases. The substantial speed-up of DgPC is especially promising when the equation contains additional, time-independent, random parameter, which is one of the main reasons to use PC–based methods in general. To demonstrate the efficiency of the algorithm for long-time simulations, we computed invariant measures for both equations, which is not a trivial task for two dimensional Navier–Stokes systems.

In Chapter 5, we provided a non-intrusive, MC noise-free sampling method for long-time solutions of SDEs. The method is based on estimating Brownian motion forcing with a finite dimensional approximation, which is also of finite variation in time. Then one hopes to obtain the convergence of the approximate solution to the true solution owing to the convergence of the approximation of Brownian motion. The method uses a restart mechanism, very similar to that of DgPC, to construct sparse quadrature rules for the solution on-the-fly and incorporates these rules along with the quadrature formulas for the random forcing to compute expectations of functionals of the solutions. Rather than propagating orthogonal polynomials, it propagates deterministic samples, i.e. quadrature nodes, of the solution to provide dynamical approximations. One can easily leverage optimized legacy differential equation solvers and carry out the evolution of the solution particles in parallel.

The main computational difficulties are to keep the number of quadrature nodes small in time, and to compute quadrature rules for the solution variables in a stable and efficient manner on-the-fly. We make use of a minimization procedure with the constraints that the quadrature rules integrate orthogonal polynomials up to a certain degrees of freedom exactly. We also discussed how to extract quadrature rules with small number of nodes and presented several different polynomials bases used in the optimization. All these constructions are motivated by the interest of computing expectations of functionals of the solutions with a high accuracy. In the limit of infinite degrees of freedom, the method would approximate the true measure of the solution with a discrete approximation having the exact moments. Numerical results for different low-dimensional nonlinear dynamics confirmed the efficiency of the algorithm in long times and showed that the algorithm compares to standard Monte Carlo method reasonably well.

To be able to compute long-time solutions of SDEs and SPDEs, one has to keep the dimensionality of randomness in check. To this end, we proposed dynamical algorithms, which propagate in time approximations to the true probability measures in one form or another. Specifically, algorithms propagate orthogonal polynomials and quadrature rules associated to the solutions via a restart procedure. The restarting step of our algorithms remains computationally expensive, especially for problems in two or more spatial dimensions. However, the restart mechanism provides a viable means to keep the number of random variables to reasonable levels. Its ability to compute statistical properties of long-time evolutions for fairly complicated equations is quite promising.

6.1 Future Work

Here are a few directions for possible future work.

Localized expansions: Polynomial chaos expansions and their dynamical versions that we developed here provide us characteristic statistical information about the response measures using global orthogonal polynomials in the random space. These global approximations yield spectral accuracy when the response depends smoothly on the random input. For strong nonlinear dependencies on the random input and/or discontinuous stochastic input, the efficiency of the global expansions might deteriorate due to Gibbs-type phenomenon. Moreover, although PCEs give us the means to assess the moments of distributions, the accuracy degrades for higher moments due to the global nature of bases. To better assess tail behaviors and account for possible stochastic discontinuities, DgPC can be coupled with local PCEs, which construct localized basis expansions in the random space; e.g. multi-element gPC by [119], see also [78; 56]. In that case, local orthogonal polynomials are constructed with respect to measure of the solution restricted to smaller subsets of the random space. This construction coupled with dynamical expansions in time would have higher accuracy for tails of distributions and robustness with respect to regularity of inputs.

Non-Markov dynamics: We mainly focused on the dynamics with Markov property and forgot the past when constructing our approximations. In modeling, a Markov property is usually desired for computational efficiency. However, not all stochastic processes have memoryless property and models can be generalized to allow some (long-range or short-range) dependence on the variables in the past. This dependence can be also exploited in forecasting. To give an example, the increments of fractional Brownian motion with Hurst parameter bigger than 1/2 are positively correlated, thus Markov semigroup theory does not apply to SDEs driven by this forcing. If a non-Markov process is of interest then our algorithm might be modified to take into account the variables from the past. The method certainly can not keep all past variables due to high dimensionality. However, for short-range dependence, one might be able to compress the past data into a relatively small number of random variables, and incorporate these and present variables into approximations. In this case, the computation of compression techniques and the estimation of orthogonal polynomials would become more intricate.

Elliptic equations: Although we mainly dealt with stochastic systems for which the randomness is supplied by time-dependent forcing terms, in many engineering applications, the main source of randomness comes from time-independent random terms. For instance, for elliptic problems

$$\nabla_x \cdot (a(x,\omega)\nabla_x u(x,\omega)) + f(x,\omega) = 0, \quad x \in G \subset \mathbb{R}^d,\, \omega \in \Omega, \tag{6.1}$$

the stochastic diffusion coefficient, denoted here by $a(x,\omega)$, can be a source of high dimensional randomness. The diffusion coefficient is typically modeled by a truncated KLE, which naturally gives a way to construct PCEs in terms of a few random KLE modes. Then PC can be coupled with finite element method in G for an approximation of the solution. In case of high-dimensional diffusion coefficient and fine spatial discretization, the dimensionality of the resulting linear system is too large, therefore efficient and scalable methods are needed.

In the deterministic setting, domain decomposition (DD) methods [101] provide an efficient way to solve large problems by introducing a partition of the spatial domain of interest and solving local, small-sized problems in an iterative way. Under appropriate formulations, the iterative solutions converge to the true solution after typically a few iterations. The main advantage of these methods is that they avoid expensive global solves. They also provide natural parallel implementations which improve scalability. DD methods can be easily extended to the stochastic setting by the help of PC expansions. Depending on the dimensionality of the diffusion coefficient, the number of terms in the PCE might be large, which gives rise to a large deterministic linear system. Then DD can be used to construct a preconditioner for this linear system [114]. Another approach would be to derive spatially-local KLEs for random parameters and construct local polynomial expansions in a lower dimensional space [18].

Assume we are interested in a high-dimensional setting where the spatial domain is partitioned into several subdomains and local solves in each iteration of a combination of DD and PC methods are still expensive to carry out. The crucial observation here is that solution on each subdomain depends on the local, low-dimensional representation of the random parameter $a(x, \omega)$ and the local boundary condition. After a few iterations, the local boundary condition will depend not only the randomness on the neighbors but also on all randomness in the system, which leads to high-dimensional representations at the interface. However, the exchanged information at the local boundary depends strongly on the nearest neighbors and weakly on the subdomains that are far away since the incoming information is more or less averaged, i.e. homogenized, throughout iterations. Hence, the local boundary condition is highly compressible and amenable to Karhunen-Loeve-type dimensionality reduction techniques; see [4; 5] for a similar approach in a multi-physics setting. To leverage this sparsity, we can introduce an iterative mechanism similar to that of DgPC. In each iteration of DD, the algorithm would compress the local boundary information, and can combine the leading random modes and random variables corresponding to the local diffusion coefficient into a lower dimensional PCE representation for the local solution. After solving the local problem, the information is passed back to neighbors and the algorithm restarts. In this way, local solves can be estimated in a much faster way although there are additional computational costs associated to the KLE and construction of orthogonal polynomials. If the diffusion coefficient is low dimensional to begin with, then the cost of compression techniques and dynamical constructions in each iteration will offset their computational savings. However, in high-dimensional settings, this proposed algorithm would provide an iterative way to approximate the global high-dimensional solution using low-dimensional and efficient representations.

Parabolic equations: Let us consider a parabolic equation of the form:

$$\partial_t u(x, t, \omega) = \nabla_x \cdot (a(x, \omega) \nabla_x u(x, t, \omega)) + f(x, \omega),$$

with an appropriate boundary condition, where $x \in G \subset \mathbb{R}^d, \omega \in \Omega$, and $t \in [0, T]$.

A standard way based on polynomial approach for this problem is to apply the KLE to the random coefficient $a(x,\omega)$, truncate the expansion, and construct PCE in terms of leading random modes. A fine spatial discretization will give a large system of equations which needs to be propagated in time.

As we observed in Chapter 4, even though there is a high-dimensional randomness in the coefficients of the SPDE, the solution itself might be represented in a fairly sparse basis in time provided dynamical bases are constructed. In other words, the diffusion coefficient can be high dimensional, whereas the solution itself can be low dimensional due to smoothing effects. In this case, the main difference with equations forced by time-dependent forcing is that the randomness $a(x,\omega)$ can not be forgotten in time (as in the case of random viscosity in Chapter 4). Then, at each restart, the KLE should be applied to the solution and the diffusion coefficient together to extract common dominating modes. Based on these modes, PCEs can be constructed to represent both u and a. The efficiency of the algorithm will depend on how fast the eigenvalues of the combined random field (u,a) decay. Moreover, domain decomposition techniques discussed above can also be incorporated into this scheme to solve the elliptic part of the equation efficiently at each time-integration step.

Hybrid PC–MC: In applications, higher order terms in the KLE of the random coefficient $a(x,\omega)$ are usually ignored. However, in some cases, a high-dimensional high-frequency component might have a non-negligible small influence on the solution. This effect might be captured better using Monte Carlo methods with a reasonable number of samples [8]. Then, an ultimate solver would be the one which exploits the advantages of both PC and MC methods by combining them: treat large scale, low-dimensional components by PCE and handle small scale, high-dimensional terms by MC.

Bibliography

[1] Rafail V. Abramov. The multidimensional maximum entropy moment problem: a review on numerical methods. *Commun. Math. Sci.*, 8(2):377–392, 2010.

[2] N. I. Akhiezer. *The classical moment problem and some related questions in analysis.* Translated by N. Kemmer. Hafner Publishing Co., New York, 1965.

[3] W. E. Arnoldi. The principle of minimized iteration in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.*, 9:17–29, 1951.

[4] M. Arnst, R. Ghanem, E. Phipps, and J. Red-Horse. Measure transformation and efficient quadrature in reduced-dimensional stochastic modeling of coupled problems. *Internat. J. Numer. Methods Engrg.*, 92(12):1044–1080, 2012.

[5] M. Arnst, R. Ghanem, E. Phipps, and J. Red-Horse. Reduced chaos expansions with random coefficients in reduced-dimensional stochastic modeling of coupled problems. *Internat. J. Numer. Methods Engrg.*, 97(5):352–376, 2014.

[6] Ivo Babuska, Fabio Nobile, and Raúl Tempone. A stochastic collocation method for elliptic partial differential equations with random input data. *SIAM J. Numer. Anal.*, 45(3):1005–1034, 2007.

[7] Ivo Babuska, Raúl Tempone, and Georgios E. Zouraris. Galerkin finite element approximations of stochastic elliptic partial differential equations. *SIAM J. Numer. Anal.*, 42(2):800–825, 2004.

[8] Guillaume Bal. Propagation of stochasticity in heterogeneous media and applications to uncertainty quantification. In Roger Ghanem, David Higdon, and Houman Owhadi,

editors, *Handbook of Uncertainty Quantification*, pages 1–24. Springer International Publishing, Cham, 2016.

[9] Andrea Barth, Annika Lang, and Christoph Schwab. Multilevel Monte Carlo method for parabolic stochastic partial differential equations. *BIT*, 53(1):3–27, 2013.

[10] Richard F. Bass. *Stochastic processes*, volume 33 of *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge University Press, Cambridge, 2011.

[11] C. Berg and J. Christensen. Density questions in the classical theory of moments. *Ann. Inst. Fourier*, 31(3):vi, 99–114, 1981.

[12] Gal Berkooz, Philip Holmes, and John L. Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. In *Annual review of fluid mechanics, Vol. 25*, pages 539–575. Annual Reviews, Palo Alto, CA, 1993.

[13] Géraud Blatman and Bruno Sudret. Adaptive sparse polynomial chaos expansion based on least angle regression. *J. Comput. Phys.*, 230(6):2345–2367, 2011.

[14] Z. I. Botev, J. F. Grotowski, and D. P. Kroese. Kernel density estimation via diffusion. *Ann. Statist.*, 38(5):2916–2957, 2010.

[15] John P. Boyd. *Chebyshev and Fourier spectral methods.* Dover Publications, Inc., Mineola, NY, second edition, 2001.

[16] M. Branicki and A. J. Majda. Fundamental limitations of polynomial chaos for uncertainty quantification in systems with intermittent instabilities. *Commun. Math. Sci.*, 11(1):55–103, 2013.

[17] R. H. Cameron and W. T. Martin. The orthogonal development of non-linear functionals in series of Fourier-Hermite functionals. *Ann. of Math. (2)*, 48:385–392, 1947.

[18] Yi Chen, John Jakeman, Claude Gittelson, and Dongbin Xiu. Local polynomial chaos expansion for linear differential equations with high dimensional random inputs. *SIAM J. Sci. Comput.*, 37(1):A79–A102, 2015.

[19] Mulin Cheng, Thomas Y. Hou, and Zhiwen Zhang. A dynamically bi-orthogonal method for time-dependent stochastic partial differential equations I: derivation and algorithms. *J. Comput. Phys.*, 242:843–868, 2013.

[20] Minseok Choi, Themistoklis P. Sapsis, and George Em Karniadakis. A convergence study for SPDEs using combined polynomial chaos and dynamically-orthogonal schemes. *J. Comput. Phys.*, 245:281–301, 2013.

[21] Alexandre Joel Chorin. Gaussian fields and random flow. *J. Fluid Mech.*, 63:21–32, 1974.

[22] John C. Cox, Jonathan E. Ingersoll, Jr., and Stephen A. Ross. A theory of the term structure of interest rates. *Econometrica*, 53(2):385–407, 1985.

[23] Giuseppe Da Prato. *Kolmogorov equations for stochastic PDEs.* Advanced Courses in Mathematics. CRM Barcelona. Birkhäuser Verlag, Basel, 2004.

[24] Philip J. Davis. A construction of nonnegative approximate quadratures. *Math. Comp.*, 21:578–582, 1967.

[25] Philip J. Davis and Philip Rabinowitz. *Methods of numerical integration.* Computer Science and Applied Mathematics. Academic Press, Inc., Orlando, FL, second edition, 1984.

[26] B. J. Debusschere, H. N. Najm, P. P. Pébay, O. M. Knio, R. G. Ghanem, and O. P. Le Maître. Numerical challenges in the use of polynomial chaos representations for stochastic processes. *SIAM J. Sci. Comput.*, 26(2):698–719, 2004.

[27] Francisco Delgado-Vences and Franco Flandoli. A spectral-based numerical method for Kolmogorov equations in Hilbert spaces. *Infin. Dimens. Anal. Quantum Probab. Relat. Top.*, 19(3):1650020, 37, 2016.

[28] Alireza Doostan, Roger G. Ghanem, and John Red-Horse. Stochastic model reduction for chaos representations. *Comput. Methods Appl. Mech. Engrg.*, 196(37-40):3951–3966, 2007.

[29] Michael Eiermann, Oliver G. Ernst, and Elisabeth Ullmann. Computational aspects of the stochastic finite element method. *Comput. Vis. Sci.*, 10(1):3–15, 2007.

[30] O. G. Ernst, A. Mugler, H-J. Starkloff, and E. Ullmann. On the convergence of generalized polynomial chaos expansions. *ESAIM Math. Model. Numer. Anal.*, 46(2):317–339, 2012.

[31] Lawrence C. Evans. *An introduction to stochastic differential equations.* American Mathematical Society, Providence, RI, 2013.

[32] D. Filipovic and M. Larsson. Polynomial preserving diffusions and applications in finance. Technical report, Swiss Finance Institute Research Paper, 2014.

[33] Philipp Frauenfelder, Christoph Schwab, and Radu Alexandru Todor. Finite elements for elliptic problems with stochastic coefficients. *Comput. Methods Appl. Mech. Engrg.*, 194(2-5):205–228, 2005.

[34] G. Freud. *Orthogonal Polynomials.* Pergamon Press, New York, 1971.

[35] A. Friedman. *Stochastic differential equations and applications. Vol. 1.* Academic Press, New York-London, 1975.

[36] W. Gautschi. *Orthogonal polynomials: computation and approximation.* Numerical Mathematics and Scientific Computation. Oxford University Press, New York, 2004. Oxford Science Publications.

[37] M. Gerritsma, J-B. van der Steen, P. Vos, and G. E. Karniadakis. Time-dependent generalized polynomial chaos. *J. Comput. Phys.*, 229(22):8333–8363, 2010.

[38] B. Gershgorin, J. Harlim, and A. J. Majda. Test models for improving filtering with model errors through stochastic parameter estimation. *J. Comput. Phys.*, 229(1):1–31, 2010.

[39] Thomas Gerstner. *Sparse Grid Quadrature Methods for Computational Finance.* University of Bonn, 2007.

[40] Thomas Gerstner and Michael Griebel. Numerical integration using sparse grids. *Numer. Algorithms*, 18(3-4):209–232, 1998.

[41] R. G. Ghanem and P. D. Spanos. *Stochastic finite elements: a spectral approach*. Springer-Verlag, New York, 1991.

[42] Roger Ghanem and John Red-Horse. Propagation of probabilistic uncertainty in complex physical systems using a stochastic finite element approach. *Phys. D*, 133(1-4):137–144, 1999. Predictability: quantifying uncertainty in models of complex phenomena (Los Alamos, NM, 1998).

[43] Michael B. Giles. Multilevel Monte Carlo path simulation. *Oper. Res.*, 56(3):607–617, 2008.

[44] G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, fourth edition, 2013.

[45] Gene H. Golub and Gérard Meurant. *Matrices, moments and quadrature with applications*. Princeton Series in Applied Mathematics. Princeton University Press, Princeton, NJ, 2010.

[46] Gene H. Golub and John H. Welsch. Calculation of Gauss quadrature rules. *Math. Comp. 23 (1969), 221-230; addendum, ibid.*, 23(106, loose microfiche suppl):A1–A10, 1969.

[47] Michael Grant and Stephen Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008.

[48] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, March 2014.

[49] Lars Grasedyck and Wolfgang Hackbusch. Construction and arithmetics of H-matrices. *Computing*, 70(4):295–334, 2003.

[50] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comput. Phys.*, 73(2):325–348, 1987.

[51] M. Hairer. Lecture notes on ergodic properties of Markov processes, 2006.

[52] Martin Hairer and Jonathan C. Mattingly. Ergodicity of the 2D Navier-Stokes equations with degenerate stochastic forcing. *Ann. of Math. (2)*, 164(3):993–1032, 2006.

[53] Martin Hairer and Jonathan C. Mattingly. Spectral gaps in Wasserstein distances and the 2D stochastic Navier-Stokes equations. *Ann. Probab.*, 36(6):2050–2091, 2008.

[54] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2):217–288, 2011.

[55] Jan S. Hesthaven, Sigal Gottlieb, and David Gottlieb. *Spectral methods for time-dependent problems*, volume 21 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2007.

[56] V. Heuveline and M. Schick. A hybrid generalized polynomial chaos method for stochastic dynamical systems. *Int. J. Uncertain. Quantif.*, 4(1):37–61, 2014.

[57] T. Y. Hou, W. Luo, B. Rozovskii, and H-M. Zhou. Wiener chaos expansions and numerical solutions of randomly forced equations of fluid mechanics. *J. Comput. Phys.*, 216(2):687–706, 2006.

[58] John K. Hunter and Bruno Nachtergaele. *Applied analysis*. World Scientific Publishing Co., Inc., River Edge, NJ, 2001.

[59] Martin Hutzenthaler, Arnulf Jentzen, and Peter E. Kloeden. Divergence of the multilevel Monte Carlo Euler method for nonlinear stochastic differential equations. *Ann. Appl. Probab.*, 23(5):1913–1966, 2013.

[60] M. Jardak, C.-H. Su, and G. E. Karniadakis. Spectral polynomial chaos solutions of the stochastic advection equation. In *Proceedings of the Fifth International Conference on Spectral and High Order Methods (ICOSAHOM-01) (Uppsala)*, volume 17, pages 319–338, 2002.

[61] A. Jentzen and P. E. Kloeden. The numerical approximation of stochastic partial differential equations. *Milan J. Math.*, 77:205–244, 2009.

[62] Kari Karhunen. Über lineare Methoden in der Wahrscheinlichkeitsrechnung. *Ann. Acad. Sci. Fennicae. Ser. A. I. Math.-Phys.*, 1947(37):79, 1947.

[63] B. N. Khoromskij, A. Litvinenko, and H. G. Matthies. Application of hierarchical matrices for computing the Karhunen-Loève expansion. *Computing*, 84(1-2):49–67, 2009.

[64] P. E. Kloeden and E. Platen. *Numerical solution of stochastic differential equations*, volume 23 of *Applications of Mathematics (New York)*. Springer-Verlag, Berlin, 1992.

[65] Christian Kuehn. Moment closure—a brief review. In *Control of self-organizing nonlinear systems*, Underst. Complex Syst., pages 253–271. Springer, Cham, 2016.

[66] O. P. Le Maître and O. M. Knio. *Spectral methods for uncertainty quantification.* Scientific Computation. Springer, New York, 2010.

[67] Olivier P. Le Maître, Omar M. Knio, Habib N. Najm, and Roger G. Ghanem. A stochastic projection method for fluid flow. I. Basic formulation. *J. Comput. Phys.*, 173(2):481–511, 2001.

[68] O. P. Le Matre and O. M. Knio. PC analysis of stochastic differential equations driven by wiener noise. *Reliability Engineering and System Safety*, 135:107–124, 2015.

[69] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK users' guide: Solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*, volume 6 of *Software, Environments, and Tools.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1998.

[70] Randall J. LeVeque. *Finite difference methods for ordinary and partial differential equations.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2007.

[71] C. Litterer and T. Lyons. High order recombination and an application to cubature on Wiener space. *Ann. Appl. Probab.*, 22(4):1301–1327, 2012.

[72] Michel Loève. *Probability theory.* Springer-Verlag, New York-Heidelberg, 1977.

[73] Sergey Lototsky, Remigijus Mikulevicius, and Boris L. Rozovskii. Nonlinear filtering revisited: a spectral approach. *SIAM J. Control Optim.*, 35(2):435–461, 1997.

[74] J. S. Lowengrub, M. J. Shelley, and B. Merriman. High-order and efficient methods for the vorticity formulation of the Euler equations. *SIAM J. Sci. Comput.*, 14(5):1107–1142, 1993.

[75] W. Luo. *Wiener Chaos Expansion and Numerical Solutions of Stochastic Partial Differential Equations.* ProQuest LLC, Ann Arbor, MI, 2006. California Institute of Technology.

[76] Harald Luschgy and Gilles Pagès. Functional quantization of a class of Brownian diffusions: a constructive approach. *Stochastic Process. Appl.*, 116(2):310–336, 2006.

[77] Terry Lyons and Nicolas Victoir. Cubature on Wiener space. *Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci.*, 460(2041):169–198, 2004.

[78] O. P. Maître, O. M. Knio, H. N. Najm, and R. G. Ghanem. Uncertainty propagation using Wiener-Haar expansions. *J. Comput. Phys.*, 197(1):28–57, 2004.

[79] Andrew J. Majda and Michal Branicki. Lessons in uncertainty quantification for turbulent dynamical systems. *Discrete Contin. Dyn. Syst.*, 32(9):3133–3221, 2012.

[80] Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. A randomized algorithm for the decomposition of matrices. *Appl. Comput. Harmon. Anal.*, 30(1):47–68, 2011.

[81] Hermann G. Matthies and Andreas Keese. Galerkin methods for linear and nonlinear elliptic stochastic partial differential equations. *Comput. Methods Appl. Mech. Engrg.*, 194(12-16):1295–1331, 2005.

[82] William C. Meecham and Armand Siegel. Wiener-Hermite expansion in model turbulence at large Reynolds numbers. *Phys. Fluids*, 7:1178–1190, 1964.

[83] R. Mikulevicius and B. Rozovskii. Linear parabolic stochastic PDEs and Wiener chaos. *SIAM J. Math. Anal.*, 29(2):452–480, 1998.

[84] R. Mikulevicius and B. L. Rozovskii. Stochastic Navier-Stokes equations for turbulent flows. *SIAM J. Math. Anal.*, 35(5):1250–1310, 2004.

[85] Habib N. Najm. Uncertainty quantification and polynomial chaos techniques in computational fluid dynamics. In *Annual review of fluid mechanics. Vol. 41*, volume 41 of *Annu. Rev. Fluid Mech.*, pages 35–52. Annual Reviews, Palo Alto, CA, 2009.

[86] F. Nobile, R. Tempone, and C. G. Webster. An anisotropic sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM J. Numer. Anal.*, 46(5):2411–2442, 2008.

[87] F. Nobile, R. Tempone, and C. G. Webster. A sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM J. Numer. Anal.*, 46(5):2309–2345, 2008.

[88] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, second edition, 2006.

[89] Erich Novak and Klaus Ritter. Simple cubature formulas with high polynomial exactness. *Constr. Approx.*, 15(4):499–522, 1999.

[90] B. Øksendal. *Stochastic differential equations*. Universitext. Springer-Verlag, Berlin, sixth edition, 2003.

[91] S. Oladyshkin and N. Wolfgang. Data-driven uncertainty quantification using the arbitrary polynomial chaos expansion. *Reliability Engineering and System Safety*, 106:179–190, 2012.

[92] S. A. Orszag and L. R. Bissonnette. Dynamical properties of truncated wiener-hermite expansions. *Phys. Fluids*, 10:2603–2613, 1967.

[93] H. Cagan Ozen and Guillaume Bal. Dynamical polynomial chaos expansions and long time evolution of differential equations with random forcing. *SIAM/ASA J. Uncertain. Quantif.*, 4(1):609–635, 2016.

[94] H. Cagan Ozen and Guillaume Bal. A dynamical polynomial chaos approach for long-time evolution of SPDEs. *Journal of Computational Physics*, 343:300 – 323, 2017.

[95] Gilles Pagès and Jacques Printems. Functional quantization for numerics with an application to option pricing. *Monte Carlo Methods Appl.*, 11(4):407–446, 2005.

[96] Gilles Pagès and Afef Sellami. Convergence of multi-dimensional quantized SDE's. In *Séminaire de Probabilités XLIII*, volume 2006 of *Lecture Notes in Math.*, pages 269–307. Springer, Berlin, 2011.

[97] L. C. Petersen. On the relation between the multidimensional moment problem and the one-dimensional moment problem. *Math. Scand.*, 51(2):361–366 (1983), 1982.

[98] Mass Per Pettersson, Gianluca Iaccarino, and Jan Nordström. *Polynomial chaos methods for hyperbolic partial differential equations.* Mathematical Engineering. Springer, Cham, 2015.

[99] Gaël Poëtte and Didier Lucor. Non intrusive iterative stochastic spectral representation with application to compressible gas dynamics. *J. Comput. Phys.*, 231(9):3587–3609, 2012.

[100] Mihai Putinar. A note on Tchakaloff's theorem. *Proc. Amer. Math. Soc.*, 125(8):2409–2414, 1997.

[101] Alfio Quarteroni and Alberto Valli. *Domain decomposition methods for partial differential equations.* Numerical Mathematics and Scientific Computation. The Clarendon Press, Oxford University Press, New York, 1999.

[102] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning.* Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 2006.

[103] Ernest K. Ryu and Stephen P. Boyd. Extensions of Gauss quadrature via linear programming. *Found. Comput. Math.*, 15(4):953–971, 2015.

[104] Yousef Saad. *Numerical methods for large eigenvalue problems*, volume 66 of *Classics in Applied Mathematics.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2011.

[105] Themistoklis P. Sapsis and Pierre F. J. Lermusiaux. Dynamically orthogonal field equations for continuous stochastic dynamical systems. *Phys. D*, 238(23-24):2347–2360, 2009.

[106] Christoph Schwab and Endre Süli. Adaptive Galerkin approximation algorithms for Kolmogorov equations in infinite dimensions. *Stoch. Partial Differ. Equ. Anal. Comput.*, 1(1):204–239, 2013.

[107] Christoph Schwab and Radu Alexandru Todor. Karhunen-Loève approximation of random fields by generalized fast multipole methods. *J. Comput. Phys.*, 217(1):100–122, 2006.

[108] T. Shardlow and A. M. Stuart. A perturbation theory for ergodic Markov chains and application to numerical approximations. *SIAM J. Numer. Anal.*, 37(4):1120–1137, 2000.

[109] Ya. G. Sinaĭ. Two results concerning asymptotic behavior of solutions of the Burgers equation with force. *J. Statist. Phys.*, 64(1-2):1–12, 1991.

[110] Ralph C. Smith. *Uncertainty quantification*, volume 12 of *Computational Science & Engineering.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2014.

[111] S. A. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. *Soviet Math. Dokl.*, 4:240–243, 1963.

[112] C. Soize and R. Ghanem. Physical systems with random uncertainties: chaos representations with arbitrary probability measure. *SIAM J. Sci. Comput.*, 26(2):395–410, 2004.

[113] J. Stoyanov. Moment problems related to the solutions of stochastic differential equations. In *Stochastic theory and control (Lawrence, KS, 2001)*, volume 280 of *Lecture Notes in Control and Inform. Sci.*, pages 459–469. Springer, Berlin, 2002.

[114] Waad Subber and Abhijit Sarkar. A domain decomposition method of stochastic PDEs: an iterative solution techniques using a two-level scalable preconditioner. *J. Comput. Phys.*, 257(part A):298–317, 2014.

[115] Vladimir Tchakaloff. Formules de cubatures mécaniques à coefficients non négatifs. *Bull. Sci. Math. (2)*, 81:123–134, 1957.

[116] Kim-Chuan Toh, Michael J. Todd, and Reha H. Tütüncü. On the implementation and usage of SDPT3—a Matlab software package for semidefinite-quadratic-linear programming, version 4.0. In *Handbook on semidefinite, conic and polynomial optimization*, volume 166 of *Internat. Ser. Oper. Res. Management Sci.*, pages 715–754. Springer, New York, 2012.

[117] Daniele Venturi, Xiaoliang Wan, and George Em Karniadakis. Stochastic low-dimensional modelling of a random laminar wake past a circular cylinder. *J. Fluid Mech.*, 606:339–367, 2008.

[118] X. Wan and G. E. Karniadakis. Long-term behavior of polynomial chaos in stochastic flow simulations. *Comput. Methods Appl. Mech. Engrg.*, 195(41-43):5582–5596, 2006.

[119] X. Wan and G. E. Karniadakis. Multi-element generalized polynomial chaos for arbitrary probability measures. *SIAM J. Sci. Comput.*, 28(3):901–928, 2006.

[120] Grzegorz W. Wasilkowski and Henryk Woźniakowski. Explicit cost bounds of algorithms for multivariate tensor product problems. *J. Complexity*, 11(1):1–56, 1995.

[121] N. Wiener. The Homogeneous Chaos. *Amer. J. Math.*, 60(4):897–936, 1938.

[122] D. Xiu. *Generalized (Wiener-Askey) polynomial chaos.* ProQuest LLC, Ann Arbor, MI, 2004. Thesis (Ph.D.)–Brown University.

[123] D. Xiu. *Numerical methods for stochastic computations.* Princeton University Press, Princeton, NJ, 2010.

[124] D. Xiu and J. S. Hesthaven. High-order collocation methods for differential equations with random inputs. *SIAM J. Sci. Comput.*, 27(3):1118–1139, 2005.

[125] D. Xiu and G. E. Karniadakis. The Wiener-Askey polynomial chaos for stochastic differential equations. *SIAM J. Sci. Comput.*, 24(2):619–644, 2002.

[126] Dongbin Xiu. Efficient collocational approach for parametric uncertainty analysis. *Commun. Comput. Phys.*, 2(2):293–309, 2007.

[127] Dongbin Xiu. Fast numerical methods for stochastic computations: a review. *Commun. Comput. Phys.*, 5(2-4):242–272, 2009.

[128] Dongbin Xiu and George Em Karniadakis. Modeling uncertainty in flow simulations via generalized polynomial chaos. *J. Comput. Phys.*, 187(1):137–167, 2003.

[129] J. Xu and J. Li. Sparse wiener chaos approximations of zakai equation for nonlinear filtering. *in Proceedings of the 21st Annual IEEE International Conference on Chinese Control and Decision Conference (CCDC09)*, page pp. 910913, 2009,.

[130] Y. Xu. On orthogonal polynomials in several variables. In *Special functions, q-series and related topics*, volume 14 of *Fields Inst. Commun.*, pages 247–270. Amer. Math. Soc., Providence, RI, 1997.

[131] Z. Zhang, B. Rozovskii, M. V. Tretyakov, and G. E. Karniadakis. A multistage Wiener chaos expansion method for stochastic advection-diffusion-reaction equations. *SIAM J. Sci. Comput.*, 34(2):A914–A936, 2012.

[132] Z. Zhang, M. V. Tretyakov, B. Rozovskii, and G. E. Karniadakis. A recursive sparse grid collocation method for differential equations with white noise. *SIAM J. Sci. Comput.*, 36(4):A1652–A1677, 2014.

[133] Zhongqiang Zhang, Michael V. Tretyakov, Boris Rozovskii, and George E. Karniadakis. Wiener chaos versus stochastic collocation methods for linear advection-diffusion-reaction equations with multiplicative white noise. *SIAM J. Numer. Anal.*, 53(1):153–183, 2015.