Journal of Scientific Computing (2019) 81:1210–1239
https://doi.org/10.1007/s10915-019-00976-5

Certified Offline-Free Reduced Basis (COFRB) Methods for Stochastic Differential Equations Driven by Arbitrary Types of Noise

Yong Liu[1] · Tianheng Chen[2] · Yanlai Chen[3] · Chi-Wang Shu[2]

Received: 2 April 2019 / Revised: 2 May 2019 / Accepted: 10 May 2019 / Published online: 16 May 2019

Abstract
In this paper, we propose, analyze, and implement a new reduced basis method (RBM) tailored for the linear (ordinary and partial) differential equations driven by arbitrary (i.e. not necessarily Gaussian) types of noise. There are four main ingredients of our algorithm. First, we propose a new space-time-like treatment of time in the numerical schemes for ODEs and PDEs. The second ingredient is an accurate yet efficient compression technique for the spatial component of the space-time snapshots that the RBM is adopting as bases. The third ingredient is a non-conventional "parameterization" of a non-parametric problem. The last is a RBM that is free of any dedicated offline procedure yet is still efficient online. The numerical experiments verify the effectiveness and robustness of our algorithms for both types of differential equations.

Keywords Reduced basis method · Stochastic PDE · Least squares · Greedy algorithms · Offline-free

Y. Liu: Research supported by the China Scholarship Council.
T. Chen: Research supported by ARO Grant W911NF-16-1-0103.
Y. Chen: Research supported by NSF Grant DMS-1719698 and AFOSR Grant FA9550-18-1-0383. This project was conceived when YC visited Brown University on sabbatical in 2017–2018.
C. Shu: Research supported by ARO Grant W911NF-16-1-0103 and NSF Grant DMS-1719410.

✉ Yanlai Chen
yanlai.chen@umassd.edu

Yong Liu
yong123@mail.ustc.edu.cn

Tianheng Chen
tianheng_chen@brown.edu

Chi-Wang Shu
shu@dam.brown.edu

[1] School of Mathematical Sciences, University of Science and Technology of China, Hefei 230026, Anhui, People's Republic of China

[2] Division of Applied Mathematics, Brown University, Providence, RI 02912, USA

[3] Department of Mathematics, University of Massachusetts Dartmouth, 285 Old Westport Road, North Dartmouth, MA 02747, USA

1 Introduction

Polynomial chaos expansion is a deterministic approach to solving stochastic partial differential equations (SPDEs). The main idea is to construct a complete set of L^2 orthogonal basis functions of the underlying probability space. Then the expansion coefficients will satisfy a deterministic system of PDEs, called the *propagator* in [22]. It is usually recognized as the stochastic Galerkin method [33] due to its evident resemblance with the classic spectral Galerkin method. Traditionally, the polynomial chaos expansion approach is well suited for SPDEs driven by Gaussian white noise or Lévy noise [12,19]. One can check [22,23] for theoretical analysis and [20,36,37] for numerical studies. Recently, a distribution-free stochastic Malliavin calculus framework was developed in [25], giving rise to a new class of SPDEs driven by arbitrary type of noise. The numerical aspects of these distribution-free SPDEs were then investigated in [10]. For linear SPDEs, the propagator system has lower triangular and sparse structure, and is independent of the type of noise involved. However, for nonlinear problems, the propagator system is fully coupled, and varies from one kind of noise to another. The Wick–Malliavin approximation was proposed in [24] as a decoupling technique. The authors in [10] also derived an error estimate for the truncation of the propagator system, showing that the mean square truncation error converges at an exponential rate with respect to the polynomial order, and at a cubic rate with respect to the number of random variables included.

In many test cases, solving the deterministic propagator is much faster than Monte Carlo simulation for a desired accuracy level, in that it is free of repeated sampling and the curse of half order convergence. However, the size of the propagator system grows exponentially with respect to both truncation parameters, and the (cubic) convergence rate is not fast enough to compensate for it. We often need to evolve thousands of expansion coefficients sequentially to obtain a high-fidelity approximation. This can be prohibitively expensive, especially for problems with long time integration and/or high space dimensions. It is therefore imperative to design fast algorithms with guaranteed accuracy and practically observable speedup.

Fast algorithms such as the proper orthogonal decomposition (POD) algorithm [3,32] have been an active research area for parametrized partial differential equations (PPDEs) which are ubiquitous in science and engineering. In the "many-query" (when simulations of the PPDE are necessary for many different realizations of the parameter) or "real-time" (when simulation result is sought in negligible time) contexts, the reduced basis method (RBM) [5,9,26,29] is by now a well-known approach. It only queries the PPDE emulator, that is the "truth" solver, for a rather small number of times at some strategically determined locations in the parameter domain in the so-called offline phase to build up a *reduced* solution space. The online phase seeks a surrogate solution for any parameter value as the energy projection of the solution corresponding to that parameter into the reduced solution space. What differentiates RBM from other model reduction approaches is perhaps its explicit error certification which also guides the building of the low-dimensional solution space.

This paper considers the adaptation of RBM for our distribution-free SODEs and SPDEs. Classic RBM usually deals with problems with an explicit parameter dependence [18,27]. Our setting is based on differential equations that are notably free of parameters. It is therefore not immediately clear how to apply RBM directly since, first of all, there is no *parameter-induced* solution manifold thanks to the lack of a parameter. The second challenge originates from the fact that the propagator system must be solved sequentially. This sequential nature means that, even if we find a way to parametrize the propagator system, the parametric problems are inter-connected, a phenomenon that is notably missing from any PPDE.

To address these challenges, we first construct a non-conventional "parameterization" of the propagator system which exploits the multi-index of the polynomial chaos expansion. We then design a RBM that is free of any dedicated offline procedure to accommodate the sequential nature of the problem since the usual offline procedure would have demanded queries of the "truth" solver at any parameter values independent of those at the others. The resulting reduced solver is still online efficient, meaning that its complexity is independent of the size of the "truth" solver. The resulting solver for the stochastic differential equations is a hybrid of "truth" and reduced solvers which is orders of magnitude faster than using the "truth" solver alone. To facilitate the handling of time, we propose a new space-time-like treatment of time in the numerical schemes for ODEs and PDEs. Unlike existing approaches, it is based on time-stepping, thus is free of the hassle of forming the space-time variational formulation, but obtains approximations of the solution in a space-time fashion. When the spatial domain has high dimensions, it is infeasible to handle the full space-time solutions and construct the resulting reduced solver, we design an accurate yet efficient compression technique for the spatial component of the space-time snapshots that RBM is adopting as bases. Obviously this compression procedure is not necessary for ODES. Therefore, we have two versions of the certified offline-free reduced basis (COFRB) methods, COFRB_ODE and COFRB_PDE.

The remainder of the paper is organized as follows. In Sect. 2, we review the stochastic analysis to derive the propagator system for SPDE model problems and the classical reduced basis method. The COFRB methods and the details of their implementation are presented in Sect. 3. Complexity discussions are in Sect. 4. Section 5 is devoted to numerical results. Finally, some concluding remarks and discussions on future work are given in Sect. 6.

2 Background

In this section, we present the necessary background materials for our algorithm, namely the distribution-free stochastic analysis and the classical reduced basis method. To ease readability, Table 1 lists the notations that are most often used in this section and beyond.

2.1 Distribution-Free Stochastic Analysis

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, and $\Xi = \{\xi_k\}_{k=1}^{\infty}$ is a sequence of independent identically distributed random variables with $\mathbb{E}[\xi_k] = 0$ and $\mathbb{E}[\xi_k^2] = 1$ for each $k \geq 1$. Suppose that there exists an orthogonal set of polynomials $\{\varphi_n(\xi)\}_{n=0}^{\infty}$ such that $\mathbb{E}[\varphi_n(\xi)\varphi_m(\xi)] = \delta_{mn} n!$. For instance, if $\{\xi_k\}_{k=1}^{\infty}$ are standard Gaussian variables, then $\{\varphi_n\}_{n=0}^{\infty}$ is the set of probabilists' Hermite polynomials. Orthogonal polynomials for a wide class of random distributions are listed in [33]. In particular, it is clear that $\varphi_0(\xi) = 1$ and $\varphi_1(\xi) = \xi$.

Then we construct the polynomial chaos basis functions under the notation of multi-indices. Denote by $\mathcal{J}$ the set of multi-indices $\alpha = (\alpha_k)_{k=1}^{\infty}$ of finite length $|\alpha| = \sum_{k=1}^{\infty} \alpha_k$:

$$\mathcal{J} = \left\{\alpha = (\alpha_k)_{k=1}^{\infty} : \alpha_k \geq 0,\ |\alpha| < \infty\right\}.$$

Let ε_0 be the multi-index whose entries are all zero, and ε_k be the multi-index such that its k-th entry is 1 and all other entries are zero. We also define polynomials and factorials of multi-indices:

Table 1 Selected notation used throughout this article

ξ_k	i.i.d. random variables $k = 1, \dots, \infty$
Ξ	The sequence of these i.i.d. random variables $\{\xi_k\}_{k=1}^{\infty}$ for the SODE and SPDE
$\mathcal{J}$	The set of multi-indices $\alpha = (\alpha_k)_{k=1}^{\infty}$ of finite length
ε_0	The multi-index whose entries are all zero
ε_k	The multi-index such that its k-th entry is 1 and all other entries are zero
Φ_α	The multi-variate basis functions $\Phi_\alpha := \prod_{k=1}^{\infty} \varphi_{\alpha_k}(\xi_k)$
d	The dimension of the spatial domain $D \subset \mathbb{R}^d$
M_f	The degrees of the spatial discretization freedom
K	The number of random variables is no more than K
M	Polynomial order is no more than M
$\mathcal{J}_{M,K}$	The truncated multi-index set
$\mathcal{N}$	The size of $\mathcal{J}_{M,K}$
N	The number of multi-indices selected by RBM
$u_{M,K}^{\mathcal{N}}$	The "truth approximation"
$u_{M,K}^{N}$	The RBM solution
$e_N(\boldsymbol{\mu})$	Reduced basis solution error
$\Delta_N(\boldsymbol{\mu})$	Error estimate (upper bound) for $\|e_N(\boldsymbol{\mu})\|$
$c_j(\alpha)$	The coefficient of the RBM solutions
$\mathcal{N}_c$	The number of collocation points, degrees of freedom in the PDE "truth" solver
$\mathcal{N}_x$	The number of collocation points for each dimension
n	Time steps
ε_{tol}	The tolerance error

$$\xi^\alpha := \prod_{k=1}^{\infty} \xi_k^{\alpha_k}, \quad \alpha! := \prod_{k=1}^{\infty} \alpha_k!.$$

The generalized polynomial chaos (gPC) basis functions [34] are taken to be the tensor products of $\{\varphi_n\}_{n=0}^{\infty}$:

$$\Phi_\alpha := \prod_{k=1}^{\infty} \varphi_{\alpha_k}(\xi_k), \text{ for each } \alpha \in \mathcal{J}. \tag{2.1}$$

Then $\Phi_{\varepsilon_0} = 1, \Phi_{\varepsilon_k} = \xi_k$, and $\mathbb{E}[\Phi_\alpha \Phi_\beta] = \alpha! \delta_{\alpha\beta}$. It is demonstrated in [25] that $\{\Phi_\alpha, \alpha \in \mathcal{J}\}$ is indeed a complete Cameron-Martin type basis [7].

Proposition 2.1 *We assume that the moment generating function* $\mathbb{E}[\exp(\theta\xi_k)]$ *exists for all* θ *in some neighborhood of 0. Then* $\{\Phi_\alpha, \alpha \in \mathcal{J}\}$ *forms a complete orthogonal basis of* $L^2(\Omega, \sigma(\Xi), \mathbb{P})$*, and for any* $\eta \in L^2(\Omega, \sigma(\Xi), \mathbb{P})$*, we have its stochastic polynomial chaos expansion*

$$\eta = \sum_{\alpha \in \mathcal{J}} \eta_\alpha \Phi_\alpha, \quad \eta_\alpha = \frac{\mathbb{E}[\eta \Phi_\alpha]}{\alpha!}.$$

The Wick product [31] serves as a convolution type binary operator on polynomial chaos basis functions:

$$\Phi_\alpha \diamond \Phi_\beta = \Phi_{\alpha+\beta},$$

so that for two random variables u and v,

$$u \diamond v = \sum_{\alpha\in\mathcal{J}}\sum_{\beta\in\mathcal{J}} u_\alpha v_\beta \Phi_{\alpha+\beta} \text{ where } u = \sum_{\alpha\in\mathcal{J}} u_\alpha \Phi_\alpha \text{ and } v = \sum_{\beta\in\mathcal{J}} v_\beta \Phi_\beta.$$

Now we take the time variable into account. Let $[0, T]$ be some time interval and $\{m_k(t)\}_{k=1}^{\infty}$ be a complete orthonormal basis of $L^2([0, T])$. We define the following driving noise $\dot{\mathfrak{N}}(t)$:

$$\dot{\mathfrak{N}}(t) = \sum_{k=1}^{\infty} m_k(t)\xi_k = \sum_{k=1}^{\infty} m_k(t)\Phi_{\varepsilon_k}, \tag{2.2}$$

and the stochastic process

$$\mathfrak{N}(t) = \int_0^t \dot{\mathfrak{N}}(s)\, ds = \sum_{k=1}^{\infty}\left(\int_0^t m_k(s)\, ds\right)\xi_k. \tag{2.3}$$

We take the linear parabolic SPDE as our model problem. The general form is

$$\begin{aligned} &\partial_t u(t, x) = \mathcal{L}u(t, x) + \mathcal{M}u(t, x) \diamond \dot{\mathfrak{N}}(t), \quad (t, x) \in (0, T] \times D,\\ &u(0, x) = u_0(x), \quad x \in D. \end{aligned} \tag{2.4}$$

Here $D \subset \mathbb{R}^d$ is some spatial domain, and

$$\mathcal{L}u(t, x) = \sum_{i=1}^{d}\sum_{j=1}^{d} a_{ij}(x)\partial_i\partial_j u(t, x) + \sum_{i=1}^{d} b_i(x)\partial_i u(t, x) + c(x)u(t, x), \tag{2.5}$$

$$\mathcal{M}u(t, x) = \sum_{i=1}^{d} \alpha_i(x)\partial_i u(t, x) + \beta(x)u(t, x), \tag{2.6}$$

where ∂_i is the i-th spatial partial derivative. We expand $u(t, x)$ into $\{\Phi_\alpha, \alpha \in \mathcal{J}\}$:

$$u(t, x) = \sum_{\alpha\in\mathcal{J}} u_\alpha(t, x)\Phi_\alpha.$$

By the definition of Wick product,

$$\mathcal{M}u(t, x) \diamond \dot{\mathfrak{N}}(t) = \sum_{\alpha\in\mathcal{J}}\sum_{k=1}^{\infty} \mathcal{M}u_\alpha(t, x)m_k(t)\Phi_{\alpha+\varepsilon_k}.$$

We come up with the deterministic propagator system by comparing the expansion coefficients on both sides of (2.4):

$$\begin{aligned} &\partial_t u_\alpha(t, x) = \mathcal{L}u_\alpha(t, x) + \sum_{\varepsilon_k\le\alpha} \mathcal{M}u_{\alpha-\varepsilon_k}(t, x)m_k(t), \quad (t, x) \in (0, T] \times D,\\ &u_\alpha(0, x) = u_0(x)\mathbb{1}_{\{\alpha=\varepsilon_0\}}, \quad x \in D. \end{aligned} \tag{2.7}$$

It is a system of linear parabolic deterministic PDEs, with a lower-triangular and sparse structure, i.e. a multi-index with order n only talks to itself and multi-indices with order $n - 1$. As a result, we can solve the system sequentially, and coefficients with the same order can be updated in parallel. Additionally, the system does not depend on the type of randomness involved, which implies the computational overhead from changes of noise is almost negligible. That is, the propagator is solved once for all types of randomness.

Remark 2.1 The i.i.d. assumption of the sequence of random variables can be relaxed. In [25] the authors only assumed that $\{\xi_k\}_{k=1}^{\infty}$ contains uncorrelated random variables with zero mean and unit variance. Here we restrict ourselves to i.i.d. sequences for technical simplicity.

Remark 2.2 In the special case of i.i.d standard Gaussian variables, $\dot{\mathfrak{N}}(t)$ is the Gaussian white noise $\dot{W}(t)$, and $\mathfrak{N}(t)$ is the Wiener process $W(t)$. Moreover, (2.4) is equivalent to the Itô type SPDE

$$du(t,x) = \mathcal{L}u(t,x)dt + \mathcal{M}u(t,x)dW(t). \tag{2.8}$$

More details can be found in [12].

2.2 Reduced Basis Methods: A Brief Overview

In this section, we present a brief overview of the main ingredients of the reduced basis method while referring to the recent surveys and monographs [16,18,27] for more details. Let $\boldsymbol{\mu}$ be a p-dimensional parameter, and X be a Hilbert space of functions on D. Given any parameter value $\boldsymbol{\mu}$ in a prescribed parameter domain, the goal is to compute $u(\boldsymbol{\mu}) \in X$ satisfying a PPDE written in a weak form

$$a(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = f(v; \boldsymbol{\mu}), \quad v \in X. \tag{2.9}$$

We denote by $(\cdot,\cdot)_X$ the inner product associated with the space X, whose induced norm is $||\cdot||_X = \sqrt{(\cdot,\cdot)_X}$. As is common in the RBM literature [29], we assume that $a(\cdot,\cdot;\boldsymbol{\mu})$ and $f(\cdot;\boldsymbol{\mu})$ are "affine"[1] with respect to the parameter $\boldsymbol{\mu}$. There are strategies for situations when the affine assumption is not satisfied, e.g., empirical interpolation [1]. We assume there exists a spatial discretization having $N_h \gg 1$ degrees of freedom to compute an approximate solution of (2.9), $u^{N_h}(\boldsymbol{\mu}) \in X^{N_h}$ with $\dim(X^{N_h}) = N_h$ called truth solution, within an acceptable accuracy for every $\boldsymbol{\mu}$. RBM attempts to, after a once-for-all preparation stage, provide a surrogate to u^{N_h} with comparable accuracy, but with orders-of-magnitude less computational cost than the truth solver. The essential idea is to compress the collection of discrete solutions $u^{N_h}(\boldsymbol{\mu})$ for enough μ into a low-dimensional space, and then to efficiently compute a projected approximation for any $\boldsymbol{\mu}$.

Indeed, an RBM algorithm approximates the solution space by an N-dimensional subspace of X^{N_h}, denoted by $X_N^{N_h}$, with $N \ll N_h$. $X_N^{N_h}$ is formed through a greedy algorithm in a hierarchical manner as the span of the so-called "snapshots", by hierarchically constructing a sample set $S^N = \{\boldsymbol{\mu}^1, \dots, \boldsymbol{\mu}^N\}$ from the training set discretizing the parameter domain and obtaining the truth solution for S^N. The surrogate RB solution $u_N^{N_h}(\boldsymbol{\mu})$ is computed as an energy projection into $X_N^{N_h}$. In addition to the size of the system decreasing from N_h to N, further saving is realized through pre-computation and storing of the parameter-independent components of the RB "stiffness" matrix which is decomposed via the affine assumption.

[1] There exist $\boldsymbol{\mu}$-dependent coefficient functions $\Theta_a^q : \mathcal{D} \to \mathbb{R}$ for $q = 1, \dots Q_a$, and $\Theta_f^q : \mathcal{D} \to \mathbb{R}$ for $q = 1, \dots, Q_f$, and corresponding continuous $\boldsymbol{\mu}$-independent bilinear forms $a^q(\cdot,\cdot) : X \times X \to \mathbb{R}$ and linear forms $f^q(\cdot) : X \to \mathbb{R}$, respectively, such that $a(w, v; \boldsymbol{\mu}) = \sum_{q=1}^{Q_a} \Theta_a^q(\boldsymbol{\mu}) a^q(w, v)$, and $f(w; \boldsymbol{\mu}) = \sum_{q=1}^{Q_f} \Theta_f^q(\boldsymbol{\mu}) f^q(w)$.

Moreover, these components can be computed by adding one more row and one more column each time a new sample location $\boldsymbol{\mu}^i$ is selected and the new snapshot resolved, thanks to the nested structure of $X_N^{N_h}$.

Selecting snapshots through the *a posteriori* error estimate: The key question left is how to select the representative parameters $\boldsymbol{\mu}^1, \ldots, \boldsymbol{\mu}^N$ for the critically important sample set S^N. RBM adopts a greedy scheme to iteratively construct S^N leaning on an efficiently-computable error estimate that quantifies the discrepancy between the dimension-*i* RBM surrogate solution $u_i^{N_h}(\boldsymbol{\mu})$ and the truth solution $u^{N_h}(\boldsymbol{\mu})$, $\Delta_i(\boldsymbol{\mu}) \geq \left\| u_i^{N_h}(\boldsymbol{\mu}) - u^{N_h}(\boldsymbol{\mu}) \right\|_{X^{N_h}}$. Assuming the existence of this error estimate, the greedy procedure for constructing S^N is summarized in Algorithm 1.

Algorithm 1 Greedy algorithm for constructing S^N and $X_N^{N_h}$.

```
1: Input: training set $\Gamma_{\text{train}}$, an accuracy tolerance $\varepsilon_{\text{tol}}$, maximum RB dimension $N_{\max}$.
2: Randomly select the first sample $\boldsymbol{\mu}^1 \in \Gamma_{\text{train}}$, and set $i = 1$ and $\varepsilon = 2\varepsilon_{\text{tol}}$.
3: Obtain truth solution $u^{N_h}(\boldsymbol{\mu}^1)$, and set $X_1^{N_h} = \text{span}\left\{u^{\mathcal{N}}(\boldsymbol{\mu}^1)\right\}$.
4: while ($\varepsilon > \varepsilon_{\text{tol}}$ and $i < N_{\max}$) do
5:     for each $\boldsymbol{\mu} \in \Gamma_{\text{train}}$ do
6:         Obtain RBM solution $u_i^{N_h}(\boldsymbol{\mu}) \in X_i^{N_h}$ and error estimate $\Delta_i(\boldsymbol{\mu})$
7:     end for
8:     $\boldsymbol{\mu}^{i+1} = \underset{\boldsymbol{\mu} \in \Gamma_{\text{train}}}{\text{argmax}}\, \Delta_i(\boldsymbol{\mu})$, $\varepsilon = \Delta_i(\boldsymbol{\mu}^{i+1})$.
9:     Augment the sample set $S^{i+1} = S^i \bigcup \{\boldsymbol{\mu}^{i+1}\}$ and the RB space $X_{i+1}^{N_h} = X_i^{N_h} \oplus \{u^{N_h}(\boldsymbol{\mu}^{i+1})\}$.
10:    Set $i \leftarrow i + 1$.
11: end while
```

Dealing with time: There are two standard approaches to deal with time within reduced basis methods [14]. The first (and more standard one) is based on a time-stepping scheme in the offline phase. The reduced basis is built by the so called POD-Greedy method [4,15,17]. It combines the standard greedy algorithm for exploring the parameter dependence as in Algorithm 1, and a proper orthogonal decomposition (POD) in time to determine the time steps that contains the most information of the temporal trajectory for the chosen parameter that can not be well approximated by the current RB space. During the online phase, the algorithm amounts to a time-stepping system that is reduced in size but with the same time step as the "truth" solver.

The second is based on the space-time variational formulation of the original problem [30,35] which considers time as an additional variable. This results in a Petrov-Galerkin problem in $d+1$ dimensions with d being the spatial dimension of the problem. The reduced basis is then formed following the standard greedy approach as outlined in this section.

3 COFRB Algorithms

Numerical discretization of (2.7), which provides the basis for the solution of the SPDE (2.4) via the spectral expansion, usually follows the method of lines technique, in which we start with standard spatial discretization schemes, transforming the propagator system into a larger system of ODEs. Suppose that the spatial discretization has M_f degrees of freedom, and $\tilde{A}$ and $\tilde{B}$ are $M_f \times M_f$ discretized version of $\mathcal{L}$ and $\mathcal{M}$. The ODE system is

$$\mathbf{u}'_\alpha(t) = \tilde{A}\mathbf{u}_\alpha(t) + \sum_{\varepsilon_k \le \alpha} m_k(t)\tilde{B}\mathbf{u}_{\alpha-\varepsilon_k}(t), \quad t \in (0, T] \tag{3.1a}$$

$$\mathbf{u}_\alpha(0) = \mathbf{u}_0 \mathbb{1}_{\{\alpha=\varepsilon_0\}} \tag{3.1b}$$

where $\mathbf{u}_\alpha(t) \in \mathbb{R}^{M_f}$ is the vector $u_\alpha(x, t)$ evaluated at those degrees of freedom. Then suitable ODE solvers such as Runge-Kutta methods can be directly adopted .

Since there are infinitely many i.i.d. random variables in Ξ, it is impossible to handle the infinite system (2.7) or (3.1) as is. A finite truncation is always necessary. Toward that end, for positive integers $K, M \ge 0$, we define the truncated multi-index set

$$\mathcal{J}_{M,K} := \{\alpha \in \mathcal{J} : |\alpha| \le M,\ d(\alpha) \le K\}, \quad \mathcal{N} = \#(\mathcal{J}_{M,K}) = \binom{M+K}{M}. \tag{3.2}$$

That is, $\mathcal{J}_{M,K}$ contains multi-indices whose corresponding polynomial order is no more than M, and number of random variables no more than K. The size of $\mathcal{J}_{M,K}$ grows rapidly with respect to both M and K. Given this truncation, the truth solution $\mathbf{u}^{N_h}_{M,K}$ approximately solving the SPDE (2.4) is

$$\mathbf{u}^{N_h}_{M,K}(t) := \sum_{\alpha \in \mathcal{J}_{M,K}} \mathbf{u}_\alpha(t)\Phi_\alpha, \tag{3.3}$$

with $\mathbf{u}_\alpha(t)$ solving (3.1) for all $\alpha \in \mathcal{J}_{M,K}$. This procedure gives highly accurate approximations. However, it is prohibitively expensive to solve when $\mathcal{N}$ is large as the ODE system (3.1) has to be resolved $\mathcal{N}$ times. This challenge can be mitigated by the COFRB method, described below for ODE and PDE separately.

3.1 SODE Problem

In this subsection, we consider the following simple linear SODE problem.

$$u'(t) = u(t) + 1 + u(t) \diamond \dot{\mathfrak{N}}(t), \quad t \in [0, T] \tag{3.4a}$$

$$u(0) = 1. \tag{3.4b}$$

The corresponding propagator system amounts to

$$\frac{d}{dt}u_\alpha(t) = u_\alpha(t) + \mathbb{1}_{\{\alpha=\varepsilon_0\}} + \sum_{\varepsilon_k \le \alpha} u_{\alpha-\varepsilon_k}(t)m_k(t). \tag{3.5}$$

For simplicity, we use the forward Euler and Crank–Nicolson time discretization schemes to describe our space-time treatment which is the foundation of the resulting COFRB method. Toward that end, we partition the time domain $(0, T]$ uniformly by a grid $0 = t_0 < t_1 < \ldots < t_n = T$ with time step size $\Delta t = t_j - t_{j-1}$. The forward Euler and Crank–Nicolson schemes read

$$u^j_{\varepsilon_0} = u^{j-1}_{\varepsilon_0} + \Delta t(u^{j-1}_{\varepsilon_0} + 1), \quad j = 1, \ldots, n \tag{3.6a}$$

$$u^j_\alpha = u^{j-1}_\alpha + \Delta t\left(u^{j-1}_\alpha + \sum_{\varepsilon_k \le \alpha} u^{j-1}_{\alpha-\varepsilon_k} m^{j-1}_k\right) \text{ for } \alpha \ne \varepsilon_0, \quad j = 1, \ldots, n \tag{3.6b}$$

$$u_{\varepsilon_0}^j = u_{\varepsilon_0}^{j-1} + \frac{1}{2}\Delta t\left(u_{\varepsilon_0}^{j-1}+1\right) + \frac{1}{2}\Delta t\left(u_{\varepsilon_0}^{j}+1\right), \quad j=1,\ldots,n \tag{3.7a}$$

$$u_\alpha^j = u_\alpha^{j-1} + \frac{1}{2}\Delta t\left(u_\alpha^{j-1} + \sum_{\varepsilon_k\le\alpha} u_{\alpha-\varepsilon_k}^{j-1} m_k^{j-1}\right) + \frac{1}{2}\Delta t\left(u_\alpha^{j} + \sum_{\varepsilon_k\le\alpha} u_{\alpha-\varepsilon_k}^{j} m_k^{j}\right),$$
$$\text{for } \alpha \ne \varepsilon_0,\ j=1,\ldots,n. \tag{3.7b}$$

Here u_α^j is an approximation of $u_\alpha(t)$ at time $t=t_j$. Our space-time treatment is to rewrite the scheme as the following system with $\vec{U}_\alpha$ encoding the full time evolution except the initial condition.

$$A\vec{U}_\alpha = \vec{f}(u_\alpha^0, u_{\alpha-\varepsilon_k}) \quad \vec{U}_\alpha = \left(u_\alpha^1, u_\alpha^2, \ldots, u_\alpha^n\right)^T. \tag{3.8}$$

We therefore have a parametric problem in (3.8) with α being the parameter. The challenge is that $\vec{U}_\alpha$ is coupled with $\vec{U}_{\alpha-\varepsilon_k}$ through the right hand side. This distinction from the usual RBM setting (2.9) means that we can not inquire (3.8) for a single α. Instead, it must be done sequentially starting from the lowest index ε_0. As a consequence, unfortunately the usual greedy Algorithm 1, thus the very existence of the offline procedure, does not apply. This motivates our design of a *"offline-free"* algorithm. To present it, we slightly abuse the notation to denote the right hand side simply by $\vec{f_\alpha}$ hiding its dependence on $u_{\alpha-\varepsilon_k}$ in the remainder of the paper. The following definition of A and $\vec{f_\alpha}$ for forward Euler and Crank–Nicolson methods finishes our description of the "truth" approximation.

$$A = \begin{pmatrix} 1 & & & \\ -(1+\Delta t) & 1 & & \\ & \ddots & \ddots & \\ & & -(1+\Delta t) & 1 \end{pmatrix}, \ \vec{f_\alpha} = \begin{pmatrix} u_\alpha^0 + \Delta t u_\alpha^0 + \Delta t \sum_{\varepsilon_k\le\alpha} u_{\alpha-\varepsilon_k}^0 m_k^0 \\ \Delta t \sum_{\varepsilon_k\le\alpha} u_{\alpha-\varepsilon_k}^1 m_k^1 \\ \vdots \\ \Delta t \sum_{\varepsilon_k\le\alpha} u_{\alpha-\varepsilon_k}^{n-1} m_k^{n-1} \end{pmatrix} \tag{3.9}$$

$$A = \begin{pmatrix} 1-\frac{1}{2}\Delta t & & & \\ -(1+\frac{1}{2}\Delta t) & 1-\frac{1}{2}\Delta t & & \\ & \ddots & \ddots & \\ & & -(1+\frac{1}{2}\Delta t) & 1-\frac{1}{2}\Delta t \end{pmatrix}; \tag{3.10a}$$

$$\vec{f_\alpha} = \begin{pmatrix} u_\alpha^0 + \frac{1}{2}\Delta t u_\alpha^0 + \frac{1}{2}\Delta t \sum_{\varepsilon_k\le\alpha} u_{\alpha-\varepsilon_k}^0 m_k^0 + \frac{1}{2}\Delta t \sum_{\varepsilon_k\le\alpha} u_{\alpha-\varepsilon_k}^1 m_k^1 \\ \frac{1}{2}\Delta t \sum_{\varepsilon_k\le\alpha} u_{\alpha-\varepsilon_k}^1 m_k^1 + \frac{1}{2}\Delta t \sum_{\varepsilon_k\le\alpha} u_{\alpha-\varepsilon_k}^2 m_k^2 \\ \vdots \\ \frac{1}{2}\Delta t \sum_{\varepsilon_k\le\alpha} u_{\alpha-\varepsilon_k}^{n-1} m_k^{n-1} + \frac{1}{2}\Delta t \sum_{\varepsilon_k\le\alpha} u_{\alpha-\varepsilon_k}^{n} m_k^{n} \end{pmatrix}. \tag{3.10b}$$

Proceeding similarly to the classical RBM, given that we already have a pre-selected multi-indices set $S^i = \{\alpha_k\}_{k=1}^i$ and the corresponding "truth approximations" written in a matrix form $\vec{U} = \left(\vec{U}_{\alpha_1}, \ldots, \vec{U}_{\alpha_i}\right)$, the goal is to find, for each α, the RB coefficients $\vec{c}_\alpha = (c_1(\alpha), c_2(\alpha), \ldots, c_i(\alpha))^T$ such that the RB solution $\vec{U}_\alpha^i = \vec{U}\vec{c}_\alpha$ satisfies the equation $A\vec{U}_\alpha^i = \vec{f_\alpha}$ in certain sense. In this paper, we solve the least squares problem as in [8]

$$\vec{U}^T A^T A \vec{U} \vec{c}_\alpha = \vec{U}^T A^T \vec{f_\alpha} \tag{3.11}$$

minimizing $\|A\vec{U}_\alpha^N - \vec{f}_\alpha\|_{l^2}$ which is a Petrov-Galerkin formulation. We note that we can also formulate the reduced problem as a Galerkin projection,

$$\vec{U}^T A\vec{U}\vec{c}_\alpha = \vec{U}^T \vec{f}_\alpha, \tag{3.12}$$

which is less stable according to our numerical result (not reported in this paper). The remaining key question of how to determine the reduced multi-indices set S^i is answered by a "keep-or-toss" approach guided by a quantity $\Delta_i(\alpha)$ bounding from the above the error between the reduced solution $\vec{U}_\alpha^i$ and truth approximation $\vec{U}_\alpha$ for any multi-index α. This upper bound resulting from a residual-based *a posteriori* error estimate,

$$\Delta_i(\alpha) = \frac{\|A\vec{U}_\alpha^i - \vec{f}_\alpha\|_{l^2}}{\sqrt{\beta_{LB}}} \tag{3.13}$$

is standard for RBM [8,29]. Here β_{LB} is the lower bound for the smallest eigenvalue of $A^T A$. The detailed algorithm is provided in Algorithm 2. We note that this "keep-or-toss" approach was initially explored in a different setting in [13], and then extended to a more general multi-layer enhanced greedy algorithm for the classical RBM setting in [21].

Algorithm 2 Certified Offline-Free Reduce Basis Method for ODE (COFRB_ODE)

```
1: Set error tolerance $\varepsilon_{\text{tol}}$ and maximum number of selected multi-indices $N_{\max}$.
2: Solve $\alpha = \varepsilon_0$ to obtain $\vec{U}_{\varepsilon_0}$.
3: Normalize $\vec{U}_\varepsilon$ to form basis matrix $\vec{U} = \left(\vec{U}_{\varepsilon_0}\right)$, $i = 1$.
4: for $\alpha \in \mathcal{J}_{M,K}$ sequentially do
5:     Solve $\vec{U}^T A^T A\vec{U}\vec{c}_\alpha = \vec{U}^T A^T \vec{f}_\alpha$ to obtain $\vec{U}_\alpha^i = \sum_{j=1}^i c_j(\alpha)\vec{U}_{\alpha_j}$.
6:     Calculate $\Delta_i(\alpha)$.
7:     if $\Delta_i(\alpha) > \varepsilon_{\text{tol}}$ & $i < N_{\max}$ then
8:         Solve $A\vec{U}_\alpha = \vec{f}_\alpha$ to obtain $\vec{U}_\alpha$.
9:         Update the basis matrix $\vec{U} = \left(\vec{U}_{\alpha_1}, \vec{U}_{\alpha_2}, \ldots, \vec{U}_{\alpha_i}, \vec{U}_\alpha\right)$.
10:        Apply a modified Gram–Schmidt transformation on $\vec{U}$ and also set the $\vec{c}_\alpha$.
11:        $i \leftarrow i + 1$.
12:    else
13:        $\vec{U}_\alpha = \vec{U}_\alpha^i$.
14:    end if
15: end for
```

Remark 3.1 (Generality of time discretizations) This paper presents the method for two kinds of time discretization schemes. In fact, any ODE solver that can be written in the form of (3.8), such as Runge-Kutta, linear Adams multistep methods et al, can be handled.

Remark 3.2 (Computational details) To ensure that the reduced system is well-conditioned, we apply the modified Gram–Schmidt (MGS) transformation on the basis matrix $\vec{U}$. Moreover, special care must be taken to ensure efficiency and accuracy of the COFRB method when it comes to calculating the RB matrices and vectors ($\vec{U}^T A^T A\vec{U}$ and $\vec{U}^T A^T \vec{f}_\alpha$), and the residual norm (as an example, a straightforward implementation may result in a negative norm). For these reasons, we provide the details of implementing steps 5, 6 and 10 of Algorithm 2 in "Appendix A".

3.2 SPDE Problem

We devote this section to the stochastic PDE case using linear parabolic PDEs as an example. For the truth approximation, we adopt a Fourier collocation approach with $\mathcal{N}_c$ collocation points for the spatial discretization [6]. These $\mathcal{N}_c$ collocation points $C^{\mathcal{N}_c} = \{x_j\}_{j=1}^{\mathcal{N}_c}$ are taken as a tensor product of $\mathcal{N}_x$ collocation points for each dimension. Obviously, for $D \subset \mathbb{R}^d$ we have $\mathcal{N}_c = \mathcal{N}_x^d$. We denote this spatial discretization degrees of freedom by M_f and present two ways to apply COFRB to this problem.

3.2.1 Method with No Spatial Compression Embedded

The first, a direct application of the Algorithm COFRB_ODE presented in the last section, becomes obvious once we write the "truth" solver in the form of (3.8). It achieves the RB reduction of the parameter-induced manifold by simply replacing numbers u_α^j for ODE by vectors $\mathbf{u}_\alpha^j$ for PDE which contains the spatial variation of the solution at the j^{th} time step. Toward that end, we also consider forward Euler and Crank–Nicolson time discretization methods for (3.1) which read

$$\text{FE}: \mathbf{u}_\alpha^j = \mathbf{u}_\alpha^{j-1} + \Delta t \left(\tilde{A}\mathbf{u}_\alpha^{j-1} + \sum_{\varepsilon_k \le \alpha} m_k^{j-1} \tilde{B}\mathbf{u}_{\alpha-\varepsilon_k}^{j-1} \right), \quad j = 1, \ldots, n \tag{3.14}$$

$$\text{CN}: \mathbf{u}_\alpha^j = \mathbf{u}_\alpha^{j-1} + \frac{\Delta t}{2} \left(\tilde{A}\mathbf{u}_\alpha^{j-1} + \sum_{\varepsilon_k \le \alpha} m_k^{j-1} \tilde{B}\mathbf{u}_{\alpha-\varepsilon_k}^{j-1} \right) + \frac{\Delta t}{2} \left(\tilde{A}\mathbf{u}_\alpha^{j} + \sum_{\varepsilon_k \le \alpha} m_k^{j} \tilde{B}\mathbf{u}_{\alpha-\varepsilon_k}^{j} \right), \quad j = 1, \ldots, n. \tag{3.15}$$

Here $\mathbf{u}_\alpha^j$ is now a vector containing the spatial variations, $\mathbf{u}_\alpha^j = [u_\alpha^{(1)}(t_j), \ldots, u_\alpha^{(\mathcal{N}_c)}(t_j)]^T = [u_\alpha(t_j, x_1), \ldots, u_\alpha(t_j, x_{\mathcal{N}_c})]^T$. Recall that $\tilde{A}$ is the spatial differential operator discretized by the spectral collocation method. We can now rewrite them in a matrix form:

$$A\vec{U}_\alpha = \vec{f}_\alpha, \quad \vec{U}_\alpha = \left((\mathbf{u}_\alpha^1)^T, (\mathbf{u}_\alpha^2)^T, \ldots, (\mathbf{u}_\alpha^n)^T \right)^T, \tag{3.16}$$

with A written block-wise for the FE and CN schemes as follows

$$A = \begin{pmatrix} I & & & \\ -(I + \Delta t\tilde{A}) & I & & \\ & \ddots & \ddots & \\ & & -(I + \Delta t\tilde{A}) & I \end{pmatrix}; \; A = \begin{pmatrix} I - \frac{\Delta t}{2}\tilde{A} & & & \\ -(I + \frac{\Delta t}{2}\tilde{A}) & I - \frac{\Delta t}{2}\tilde{A} & & \\ & \ddots & \ddots & \\ & & -(I + \frac{\Delta t}{2}\tilde{A}) & I - \frac{\Delta t}{2}\tilde{A} \end{pmatrix}. \tag{3.17}$$

Here $I \in \mathbb{R}^{\mathcal{N}_c \times \mathcal{N}_c}$ is the identity matrix. To facilitate presentation of the detailed efficient implementation of the algorithm (see "Appendix A"), we write $\vec{f}_\alpha$ for the FE solver in the following form

$$\vec{f}_\alpha = \begin{pmatrix} \mathbf{u}_\alpha^0 + \Delta t\tilde{A}\mathbf{u}_\alpha^0 + \Delta t \sum_{\varepsilon_k \le \alpha} m_k^0 \tilde{B}\mathbf{u}_{\alpha-\varepsilon_k}^0 \\ \Delta t \sum_{\varepsilon_k \le \alpha} m_k^1 \tilde{B}\mathbf{u}_{\alpha-\varepsilon_k}^1 \\ \vdots \\ \Delta t \sum_{\varepsilon_k \le \alpha} m_k^{n-1} \tilde{B}\mathbf{u}_{\alpha-\varepsilon_k}^{n-1} \end{pmatrix} \tag{3.18}$$

$$
\begin{aligned}
&= \begin{pmatrix} \mathbf{u}_\alpha^0 + \Delta t \tilde{A}\mathbf{u}_\alpha^0 + \Delta t \sum_{\varepsilon_k \le \alpha} m_k^0 \tilde{B}\mathbf{u}_{\alpha-\varepsilon_k}^0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \Delta t \sum_{\varepsilon_k \le \alpha} \begin{pmatrix} 0 & & & \\ & m_k^1 \tilde{B} & & \\ & & \ddots & \\ & & & m_k^{n-1}\tilde{B} \end{pmatrix} \begin{pmatrix} 0 \\ \mathbf{u}_{\alpha-\varepsilon_k}^1 \\ \vdots \\ \mathbf{u}_{\alpha-\varepsilon_k}^{n-1} \end{pmatrix} \\
&= \vec{f}_\alpha^0 + \Delta t \sum_{\varepsilon_k \le \alpha} M_k^0 \vec{U}^0 \vec{c}_{\alpha-\varepsilon_k}
\end{aligned} \tag{3.19}
$$

where $M_k^0 = \operatorname{diag}\{0, m_k^1 \tilde{B}, \ldots, m_k^{n-1}\tilde{B}\}$ and $\vec{U}^0$ is the full spatial-temporal representation of the RB basis matrix with the initial and final time step truncated.

$$
\vec{U}^0 = \begin{pmatrix} 0 & 0 & \ldots & 0 \\ \mathbf{u}_{\alpha_1}^1 & \mathbf{u}_{\alpha_2}^1 & \ldots & \mathbf{u}_{\alpha_i}^1 \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{u}_{\alpha_1}^{n-1} & \mathbf{u}_{\alpha_2}^{n-1} & \ldots & \mathbf{u}_{\alpha_i}^{n-1} \end{pmatrix}. \tag{3.20}
$$

This compact form can be achieved for CN scheme as well:

$$
\vec{f}_\alpha = \begin{pmatrix} \mathbf{u}_\alpha^0 + \Delta t \tilde{A}\mathbf{u}_\alpha^0 + \frac{\Delta t}{2} \sum_{\varepsilon_k \le \alpha} m_k^0 \tilde{B}\mathbf{u}_{\alpha-\varepsilon_k}^0 + \frac{\Delta t}{2} \sum_{\varepsilon_k \le \alpha} m_k^1 \tilde{B}\mathbf{u}_{\alpha-\varepsilon_k}^1 \\ \frac{\Delta t}{2} \sum_{\varepsilon_k \le \alpha} \left(m_k^1 \tilde{B}\mathbf{u}_{\alpha-\varepsilon_k}^1 + m_k^2 \tilde{B}\mathbf{u}_{\alpha-\varepsilon_k}^2 \right) \\ \vdots \\ \frac{\Delta t}{2} \sum_{\varepsilon_k \le \alpha} \left(m_k^{n-1} \tilde{B}\mathbf{u}_{\alpha-\varepsilon_k}^{n-1} + m_k^n \tilde{B}\mathbf{u}_{\alpha-\varepsilon_k}^n \right) \end{pmatrix} \tag{3.21}
$$

$$
\begin{aligned}
&= \begin{pmatrix} \mathbf{u}_\alpha^0 + \frac{\Delta t}{2}\mathbf{u}_\alpha^0 + \frac{\Delta t}{2} \sum_{\varepsilon_k \le \alpha} m_k^0 \tilde{B}\mathbf{u}_{\alpha-\varepsilon_k}^0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \frac{\Delta t}{2} \sum_{\varepsilon_k \le \alpha} \begin{pmatrix} 0 & & & \\ & m_k^1 \tilde{B} & & \\ & & \ddots & \\ & & & m_k^{n-1}\tilde{B} \end{pmatrix} \begin{pmatrix} 0 \\ \mathbf{u}_{\alpha-\varepsilon_k}^1 \\ \vdots \\ \mathbf{u}_{\alpha-\varepsilon_k}^{n-1} \end{pmatrix} \\
&\quad + \frac{\Delta t}{2} \sum_{\varepsilon_k \le \alpha} \begin{pmatrix} m_k^1 \tilde{B} & & & \\ & m_k^2 \tilde{B} & & \\ & & \ddots & \\ & & & m_k^n \tilde{B} \end{pmatrix} \begin{pmatrix} \mathbf{u}_{\alpha-\varepsilon_k}^1 \\ \mathbf{u}_{\alpha-\varepsilon_k}^2 \\ \vdots \\ \mathbf{u}_{\alpha-\varepsilon_k}^n \end{pmatrix} \\
&= \vec{f}_\alpha^0 + \frac{\Delta t}{2} \sum_{\varepsilon_k \le \alpha} \left(M_k^0 \vec{U}^0 + M_k \vec{U} \right) \vec{c}_{\alpha-\varepsilon_k}
\end{aligned} \tag{3.22}
$$

where $M_k = \operatorname{diag}\{m_k^1 \tilde{B}, m_k^2 \tilde{B}, \ldots, m_k^n \tilde{B}\}$ and $\vec{U}$ is the full spatial-temporal representation of the RB basis matrix with the initial time step truncated.

$$
\vec{U} = \begin{pmatrix} \mathbf{u}_{\alpha_1}^1 & \mathbf{u}_{\alpha_2}^1 & \ldots & \mathbf{u}_{\alpha_i}^1 \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{u}_{\alpha_1}^{n-1} & \mathbf{u}_{\alpha_2}^{n-1} & \ldots & \mathbf{u}_{\alpha_i}^{n-1} \\ \mathbf{u}_{\alpha_1}^n & \mathbf{u}_{\alpha_2}^n & \ldots & \mathbf{u}_{\alpha_i}^n \end{pmatrix}. \tag{3.23}
$$

3.2.2 Method with Spatial Compression Embedded

The consequence of ignoring the spatial compressibility is two-fold. First is the large memory the algorithm is consuming. For example, the "truth approximation" from (3.16), $\vec{U}_\alpha$, is in

$\mathbb{R}^{n\mathcal{N}_x^d}$. So are the number of rows for $\vec{U}^0$ and $\vec{U}$ and the resulting matrices $\vec{U}$, $A\vec{U}$, $A^TA\vec{U}$ etc. This proves challenging especially for the high spatial dimension ($d \geq 2$) case. The second negative consequence is the high operation counts thanks to the multiplication of matrices of this size.

To mitigate these difficulties, we design the following algorithm to "compress" the basis every time one is added to the reduced space while maintaining its accuracy. To achieve that, we rewrite the vector $\vec{U}_\alpha$ as a matrix $\widehat{\vec{U}_\alpha}$ to which we apply singular value decomposition (SVD).

$$\widehat{\vec{U}_\alpha} = \left(\mathbf{u}_\alpha^1, \mathbf{u}_\alpha^2, \ldots, \mathbf{u}_\alpha^n\right) = \eta\Sigma\zeta^T = \sum_{j=1}^{\min(\mathcal{N}_c,n)} \lambda_j\eta_j\zeta_j^T \tag{3.24}$$

where η is an $\mathcal{N}_c$-by-$\mathcal{N}_c$ orthonormal matrix, and ζ is an n-by-n orthonormal matrix, and Σ is an $\mathcal{N}_c$-by-n matrix whose entries are zero except for its $\min(\mathcal{N}_c, n)$ diagonal elements. They are denoted by λ_j that is decreasing with j. η_j and ζ_j are the columns of matrices η and ζ. Our algorithm truncates this SVD by storing only the first $m(\ll \min(\mathcal{N}_c, n))$ vectors η_j, ζ_j and singular values of λ_j to approximate $\widehat{\vec{U}_\alpha}$ as follows.

$$\widehat{\vec{U}_\alpha} \approx \sum_{j=1}^{m}\lambda_j\eta_j\zeta_j^T = \begin{pmatrix} \eta_1^1 & \eta_2^1 & \ldots & \eta_m^1 \\ \eta_1^2 & \eta_2^2 & \ldots & \eta_m^2 \\ \vdots & \vdots & \vdots & \vdots \\ \eta_1^{\mathcal{N}_c} & \eta_2^{\mathcal{N}_c} & \ldots & \eta_m^{\mathcal{N}_c} \end{pmatrix} \begin{pmatrix} \lambda_1\zeta_1^1 & \lambda_1\zeta_1^2 & \ldots & \lambda_1\zeta_1^n \\ \lambda_2\zeta_2^1 & \lambda_2\zeta_2^2 & \ldots & \lambda_2\zeta_2^n \\ \vdots & \vdots & \vdots & \vdots \\ \lambda_m\zeta_m^1 & \lambda_m\zeta_m^2 & \ldots & \lambda_m\zeta_m^n \end{pmatrix} \tag{3.25}$$

$$= V_\alpha^m \left(w_\alpha^1 \; w_\alpha^2 \; \ldots \; w_\alpha^n\right) \tag{3.26}$$

where $V_\alpha^m \in \mathbb{R}^{\mathcal{N}_c\times m}$, $w_\alpha^j \in \mathbb{R}^m$, $j = 1, \ldots, n$. We can then rewrite the truth approximation back as a vector form

$$\vec{U}_\alpha \approx \begin{pmatrix} V_\alpha^m & & \\ & \ddots & \\ & & V_\alpha^m \end{pmatrix} \begin{pmatrix} w_\alpha^1 \\ \vdots \\ w_\alpha^n \end{pmatrix} = \vec{V}_\alpha^m \vec{w}_\alpha. \tag{3.27}$$

Now we only store $m\mathcal{N}_c + mn$ elements instead of $n\mathcal{N}_c$ for each truth approximation, a $\frac{m}{n} + \frac{m}{\mathcal{N}_c}$ reduction. The basis matrix $\vec{U}$ can in turn be approximated as

$$\vec{U} \approx \begin{pmatrix} V_{\alpha_1}^m & & & V_{\alpha_2}^m & & & V_{\alpha_i}^m & & \\ & \ddots & & & \ddots & & & \ddots & \\ & & V_{\alpha_1}^m & & & V_{\alpha_2}^m & & & V_{\alpha_i}^m \end{pmatrix} \begin{pmatrix} w_{\alpha_1}^1 & 0 & \ldots & 0 \\ \vdots & \vdots & & \vdots \\ w_{\alpha_1}^n & 0 & \ldots & 0 \\ 0 & w_{\alpha_2}^1 & \ldots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & w_{\alpha_2}^n & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & w_{\alpha_i}^1 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \ldots & w_{\alpha_i}^n \end{pmatrix}$$

$$= \left(\vec{V}_{\alpha_1}^m \; \vec{V}_{\alpha_2}^m \; \ldots \; \vec{V}_{\alpha_i}^m\right)\left(\vec{W}_{\alpha_1} \; \vec{W}_{\alpha_2} \; \ldots \; \vec{W}_{\alpha_i}\right) = \vec{V}\vec{W}. \tag{3.28}$$

The other basis matrix $\vec{U}^0$ is approximated in the same fashion, simply with $\vec{W}$ replaced by $\vec{W}^0$ that contains $w^j_{\alpha_i}$ for $j = 0, \ldots, n-1$ instead of $j = 1, \ldots, n$ with $w^0_{\alpha_i} = 0$,

$$\vec{U}^0 = \vec{V}\vec{W}^0.$$

Then the reduced solver (3.11) becomes

$$\vec{W}^T\vec{V}^T A^T A\vec{V}\vec{W}\vec{c}_\alpha = \vec{W}^T\vec{V}^T A^T \vec{f}_\alpha \tag{3.29}$$

$$= \begin{cases} \vec{W}^T\vec{V}^T A^T \left(\vec{f}^0_\alpha + \Delta t \sum_{\varepsilon_k \le \alpha} M^0_k \vec{V}\vec{W}^0\vec{c}_{\alpha-\varepsilon_k}\right) & \text{for FE} \\ \vec{W}^T\vec{V}^T A^T \left(\vec{f}^0_\alpha + \frac{\Delta t}{2} \sum_{\varepsilon_k \le \alpha} (M^0_k \vec{V}\vec{W}^0 + M_k\vec{V}\vec{W})\vec{c}_{\alpha-\varepsilon_k}\right) & \text{for CN.} \end{cases} \tag{3.30}$$

The detailed algorithm, called COFRB_PDE, is provided in Algorithm 3 with implementation details postponed to “Appendix B”.

Algorithm 3 Certified Offline-Free Reduced Basis Method for PDE (COFRB_PDE)

```
1: Set tolerance error $\varepsilon_{\text{tol}}$ and maximum number of selected multi-indices $N_{\max}$.
2: Solve $\alpha = \varepsilon_0$ to obtain $\vec{U}_{\varepsilon_0}$, set $\alpha_1 = \varepsilon_0$.
3: Do SVD on $\widehat{\vec{U}_{\alpha_1}}$ to obtain $V^m_{\alpha_1}$ and $\vec{w}_{\alpha_1}$.
4: Normalize $\vec{w}_{\alpha_1}$ and set $i = 1$.
5: for $\alpha \in \mathcal{J}_{M,K}$ sequentially do
6:     Solve $\vec{W}^T\vec{V}^T A^T A\vec{V}\vec{W}\vec{c}_\alpha = \vec{W}^T\vec{V}^T A^T \vec{f}_\alpha$ to obtain $\vec{U}^i_\alpha = \sum_{j=1}^i c_j(\alpha)\vec{V}^m_{\alpha_j}\vec{w}_{\alpha_j}$.
7:     Calculate $\Delta_i(\alpha)$.
8:     if $\Delta_i(\alpha) > \varepsilon_{\text{tol}}$ & $i < N_{\max}$ then
9:         Solve $A\vec{U}_\alpha = \vec{f}_\alpha$ to obtain $\vec{U}_\alpha$.
10:        Do SVD on $\vec{U}_\alpha$ to obtain $V^m_\alpha$ and $\vec{W}_\alpha$.
11:        Update the matrix $\vec{V} = \left(\vec{V}^m_{\alpha_1}, \vec{V}^m_{\alpha_2}, \ldots, \vec{V}^m_{\alpha_i}, \vec{V}^m_\alpha\right)$ and $\vec{W} = \begin{pmatrix} \vec{W} & 0 \\ 0 & \vec{w}_\alpha \end{pmatrix}$.
12:        Normalize $\vec{w}_\alpha$ and set $\vec{c}_\alpha = (0, \ldots, 0, \|\vec{w}_\alpha\|)^T$.
13:        $i \leftarrow i + 1$.
14:    else
15:        $\vec{U}_\alpha = \vec{U}^i_\alpha$, $\vec{c}_\alpha = (c_1(\alpha), c_2(\alpha), \ldots, c_i(\alpha))$.
16:    end if
17: end for
```

Remark 3.3 Note that the columns of the matrices $\vec{W}$ and $\vec{W}^0$ are orthogonal. So we only need to normalize them. The absence of MGS for the basis matrix $\vec{U}$ improves efficiency of the algorithm. Currently, we use the same m across the RB bases. This may lead to some redundancy in $\vec{U}$ resulting in a possibly non-invertible RB matrix $A_{RB} = \vec{W}^T\vec{V}^T A^T A\vec{V}\vec{W}$. In this case, we adopt the Moore-Penrose pseudo-inversion [2], $(\vec{W}^T\vec{V}^T A^T A\vec{V}\vec{W})^\dagger$, to solve the least square problem which amounts to identifying the RB coefficients $\vec{c}_\alpha$ for (3.29) with minimum norm. This redundancy can be removed, thus the Moore-Penrose pseudo-inversion avoided, by using an adaptively chosen m which gets smaller as we build up the RB space.

4 Complexity Analysis of the COFRB Algorithms

As is well-known [28], the tremendous speedup of the reduced basis method originates from the decomposition of the computation into an offline stage and an online stage. The essence is that, whenever a new basis is added to the reduced space, components of the reduced solver and its error estimation that are dependent on this basis should be precomputed and stored. As a consequence, the reduces solver and its error estimation are independent of the degrees of freedom of the truth solver. Keeping this in mind, our COFRB algorithm can be implemented in a similar fashion. The lack of clear offline-online decomposition is overcome by that, in the sequential procedure of the problem resolution, there are overwhelmingly more multi-indices whose solutions are provided by the reduced, as opposed to full, solver. The amount of savings from these reduced solves being independent of the size of the full problem more than compensates for the additional computation, after each full solve, to prepare for the reduced solve. Thus, the total computational complexity is less than the original scheme. In this section, we provide a detailed computational complexity count taking forward Euler time discretization as an example. It makes evident the theoretical basis of the practical numerical speedup we observe in the numerical results section below.

4.1 Computational Complexity for COFRB_ODE

We first consider Algorithm 2 for the SODE problem. The complexity of the original scheme, that is obtaining truth approximations $\vec{U}_\alpha$ from (3.8) for all $\alpha \in \mathcal{J}_{M,K}$, is of the order

$$\mathcal{C}^{\mathrm{O}}_{\mathrm{Full}} = n\mathcal{N}K.$$

Recall $\mathcal{N}$ is the cardinality of $\mathcal{J}_{M,K}$. To compute the complexity for Algorithm 2, we note that for each $\alpha \in \mathcal{J}_{M,K}$, we need to form $\vec{f}_{RB} = \vec{U}^T A^T \vec{f}_\alpha$, solve $\vec{c}_\alpha$, and calculate $\Delta_i(\alpha)$ from (6.11). The operation count of this procedure is, at most, of the order

$$\mathcal{N}\left(\overbrace{N^2K + N^3}^{\text{form } \vec{f}_{RB} \text{ and solve } \vec{c}_\alpha} + \underbrace{\overbrace{NK^2}^{(6.11)}}_{\text{calculate } \Delta_i(\alpha)}\right)$$

This cost is, as expected, independent of the number of time step (in this case the size of the truth approximation) n. During this procedure, if the solution corresponding to the current multi-index is determined to be a worthy addition to the RB space, we encounter the following additional cost which is aggregated for all the N chosen multi-indices.

$$\sum_{i=1}^{N}\left(\underbrace{\overbrace{niK}^{(6.4) \text{ and } (6.5)}}_{\text{update } \vec{U}^T A^T A\vec{U} \text{ and } \vec{U}^T A^T M_k^0 U^0} + \overbrace{ni}^{\text{MGS}} + \underbrace{\overbrace{niK^2}^{(6.10)}}_{\text{QR prep}}\right)$$

Therefore, the total complexity of Algorithm 2 for SODE problem is of the order

$$\mathcal{C}^{\mathrm{O}}_{\mathrm{RB}} = nN^2K^2 + N(N^2 + NK + K^2)\mathcal{N}.$$

Remark 4.1 The amount of speedup, assuming $N \geq K$, is in the order of

$$\frac{\mathcal{C}^{\mathrm{O}}_{\mathrm{RB}}}{\mathcal{C}^{\mathrm{O}}_{\mathrm{Full}}} = \frac{N^2K}{\mathcal{N}} + \frac{N^3}{nK}.$$

4.2 Computational Complexity for COFRB_PDE

One-dimensional case: The cost of the original scheme is of the order $n\mathcal{N}_x^2\mathcal{N}K$. For Algorithm 2, a similar analysis as above shows that the total cost is of the order

$$n\mathcal{N}_x NK(NK+\mathcal{N}_x)+N(N^2+NK+K^2)\mathcal{N}.$$

For Algorithm 3, due to the spatial reduction, the size of the space-time snapshots decreases from $n\mathcal{N}_x$ to $m\mathcal{N}_x+mn$. For each multi-index, we need to form $\vec{W}^T\vec{V}^TA^TA\vec{V}\vec{W}$ and $\vec{W}^T\vec{V}^TA^TM_k^0\vec{V}\vec{W}^0$ (of size at most N, the number of reduced basis), calculate the $(\vec{W}^T\vec{V}^TA^TA\vec{V}\vec{W})^\dagger$, and evaluate $\Delta_i(\alpha)$ through forming and decomposing $\mathcal{B}^T\mathcal{B}$ (see "Appendix B"). All these operations add up to be, at most

$$\mathcal{N}\left(\overbrace{(N\mathcal{N}_x m(\mathcal{N}_x+m)+Nnm^2)(K+1)}^{\text{form } \vec{W}^T\vec{V}^TA^TA\vec{V}\vec{W} \text{ and } \vec{W}^T\vec{V}^TA^TM_k^0\vec{V}\vec{W}^0}+\overbrace{N^3}^{(\vec{W}^T\vec{V}^TA^TA\vec{V}\vec{W})^\dagger}+\overbrace{NK^2nm^2+(NK)^3}^{\text{form } \mathcal{B}^T\mathcal{B}}\right)$$
$$\text{and orthor. decomp.}$$

For the N chosen multi-indices, we have the following additional cost

$$N\left(\overbrace{n\mathcal{N}_x^2K}^{\text{obtain } \vec{U}_\alpha}+\overbrace{\min(n,\mathcal{N}_x)^3}^{\text{SVD on } \widehat{\vec{U}_\alpha}}\right).$$

Thus, the total cost is of the order

$$\mathcal{N}\big(N\mathcal{N}_x m(\mathcal{N}_x+m)K+nNK^2m^2+N^3K^3\big)+n\mathcal{N}_x^2NK+N\min(n,\mathcal{N}_x)^3.$$

Two-dimensional case: In the 2D case, the differentiation matrices $\tilde{A}$ and $\tilde{B}$ are sparse, thus the complexity of matrix vector multiplication is of the order $\mathcal{N}_x^3$. So the complexity of obtaining $\vec{U}_\alpha$ from the original scheme is of the order $n\mathcal{N}_x^3\mathcal{N}K$.

For Algorithm 2, the complexity is of order

$$n\mathcal{N}_x^2NK(NK+\mathcal{N}_x)+N(N^2+NK+K^2)\mathcal{N}.$$

For Algorithm 3,

$$\mathcal{N}\big(N\mathcal{N}_x^2 m(\mathcal{N}_x+m)K+nNK^2m^2+N^3K^3\big)+n\mathcal{N}_x^3NK+N\min(n,\mathcal{N}_x^2)^3.$$

Summary: We list here the complexities of the original scheme, direct extension of COFRB_ODE, and then COFRB_PDE for the d-dimensional SPDE problems.

$$\begin{aligned}
\mathcal{C}_{\text{Full}}^{\text{P}} &= n\mathcal{N}_x^{d+1}\mathcal{N}K,\\
\mathcal{C}_{\text{RB}}^{\text{P},1} &= n\mathcal{N}_x^d NK(NK+\mathcal{N}_x)+N(N^2+NK+K^2)\mathcal{N},\\
\mathcal{C}_{\text{RB}}^{\text{P},2} &= \mathcal{N}\Big(N\mathcal{N}_x^d m(\mathcal{N}_x+m)K+nNK^2m^2+N^3K^3\Big)+n\mathcal{N}_x^{d+1}NK+N\min(n,\mathcal{N}_x^d)^3.
\end{aligned}$$

After simple algebraic simplifications, the amount of speedup of the COFRB_PDE algorithm is

$$\frac{\mathcal{C}_{\text{RB}}^{\text{P},2}}{\mathcal{C}_{\text{Full}}^{\text{P}}}=\frac{N}{\mathcal{N}}+\frac{Nm}{n}+\frac{NK}{\mathcal{N}_x^{d+1}}\left(m^2+\frac{N^2K}{n}\right)+\frac{N}{\mathcal{N}K}\min\left(\frac{n^2}{\mathcal{N}_x^{d+1}},\frac{\mathcal{N}_x^{2d-1}}{n}\right).$$

5 Numerical Results

In this section, we apply our algorithms to one SODE problem and two SPDE problems, and compare the costs of the original scheme and the COFRB methods. In all the experiments, we set the maximum number of reduced multi-indices $N_{\max}$ to be 200. For the COFRB_PDE algorithm, we set the number of columns of V_α^m, m to be 6 in all experiments. Exact solutions are obtained through solving the corresponding moment equations of (3.1).

Example 5.1 (Linear SODE). Suppose that $u = u(t)$ satisfies the following linear SODE

$$\begin{aligned} u'(t) &= u(t) + 1 + u(t) \diamond \dot{\mathfrak{N}}(t), \quad t \in [0, T] \\ u(0) &= 1. \end{aligned} \tag{5.1}$$

We evolve the propagator ODE system to the final time $T = 1$ with time step size $\Delta t = 10^{-4}$ i.e. time steps $n = 10{,}000$. In order to examine the error of reduced basis solution in comparison to the truth approximation, we define the following square error

$$e_2^{RBM} = \mathbb{E}[|u_{M,K}^N(1) - u_{M,K}^{\mathcal{N}}(1)|^2]. \tag{5.2}$$

As a reference, we also compute the error of the truth approximation in comparison to the exact solution

$$e_2^{ORI} = \mathbb{E}[|u_{M,K}^{\mathcal{N}}(1) - u(1)|^2]. \tag{5.3}$$

We test both the forward Euler and Crank–Nicolson time discretization schemes for $(M, K) = (8, 8)$ and $(9, 9)$ and plot the histories of convergence, number of chosen multi-indices, and computation time, all with respect to ε_{tol} in Fig. 1. It is clear that the method is highly effective in capturing the "important" multi-indices. With 35 (out of a total of more than 12,000) of them, the COFRB solution reaches an accuracy of more than 8 digits. In

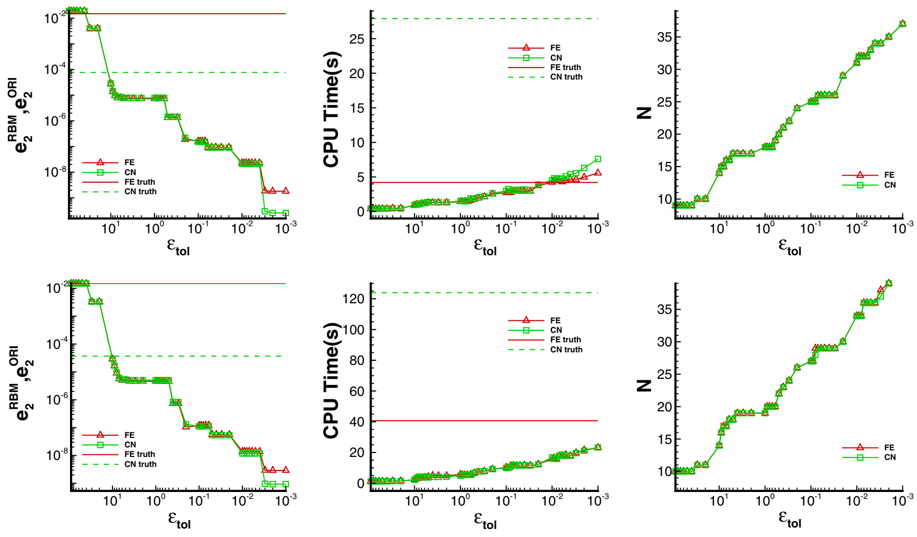

Fig. 1 SODE problem for $(M, K) = (8, 8)$ (top) and $(M, K) = (9, 9)$ (bottom). From left to right are histories of convergence of e_2^{RBM} and e_2^{ORI}, CPU time of Algorithm 2 as a function of ε_{tol}, and number of chosen multi-indices (out of 12,870 or 48,620) as a function of ε_{tol}

Table 2 SODE problem: error and computing costs for Algorithm COFRB_ODE

Method	(M, K)	ε_{tol}	N	$\mathcal{N}$	e_2^{RBM}	e_2^{ORI}	CPU time	
							COFRB_ODE	Full
FE	(8, 8)	30.00E+00	10	12870	4.00E−03	1.50E−02	1.12E−01	1
	(9, 9)	30.00E+00	11	48620	3.30E−03	1.50E−02	3.43E−02	1
CN	(8, 8)	10.00E+00	14	12870	2.86E−05	7.71E−05	3.41E−01	1
	(9, 9)	10.00E+00	14	48620	2.98E−05	3.67E−05	1.85E−02	1

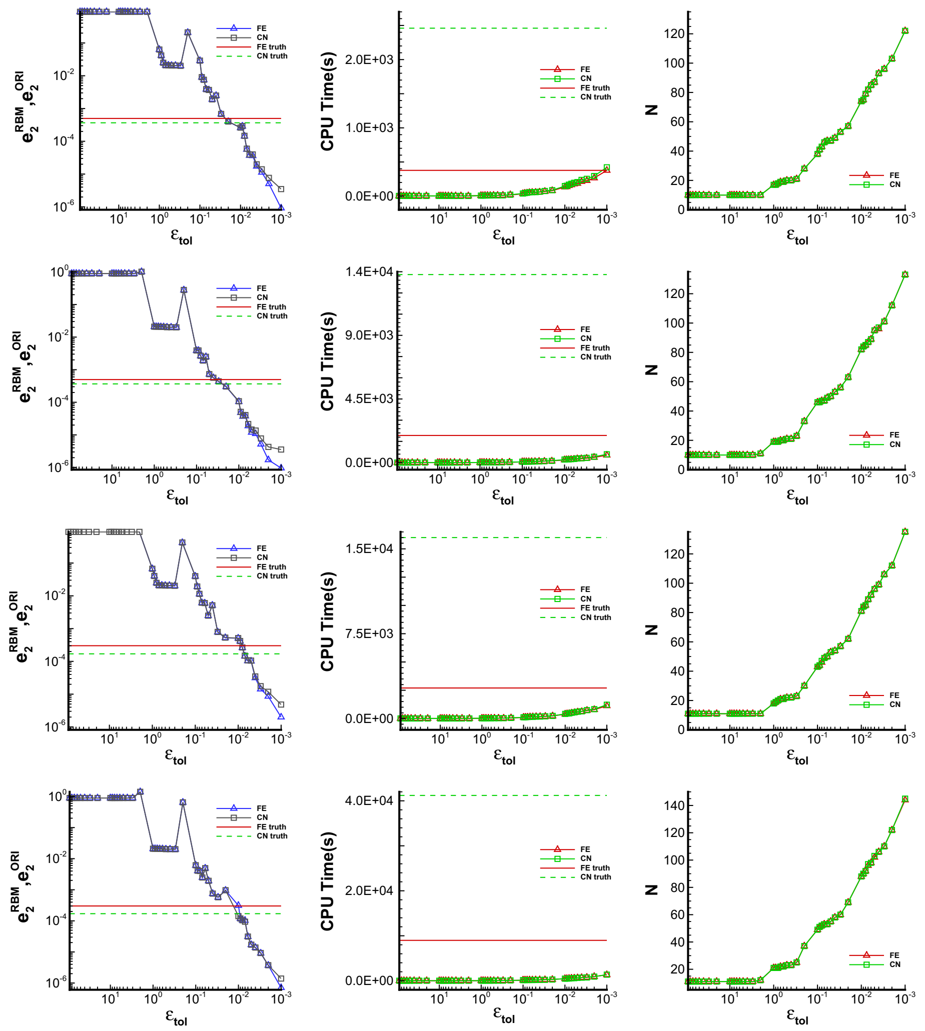

Fig. 2 1D SPDE problem for $(M, K, \mathcal{N}_c) = (9, 9, 32)$ (Row 1), $(9, 9, 64)$ (Row 2), $(10, 10, 32)$ (Row 3), and $(10, 10, 64)$ (Row 4). From left to right are histories of convergence of e_2^{RBM} and e_2^{ORI}, CPU time of Algorithm 3 as a function of ε_{tol}, and number of chosen multi-indices (out of 48,620) as a function of ε_{tol}

Table 2, we observe that when e_2^{RBM} and e_2^{ORI} reach the same level, our algorithm achieves savings of two orders of magnitude.

Example 5.2 (Linear 1D parabolic PDE). We solve the linear parabolic PDE in Sect. 2.1. Consider the one-dimensional space region $D = [0, 2\pi]$ with periodic boundary condition. The initial data is $u_0(x) = \cos x$. Differential operators $\mathcal{L}$ and $\mathcal{M}$ are set to be

$$\mathcal{L}u = 0.145\partial_x^2 u + 0.1 \sin x \partial_x u, \quad \mathcal{M}u = 0.5\partial_x u. \tag{5.4}$$

Fourier collocation method with $\mathcal{N}_c$ collocation points is used for spatial discretization. Let $\{x_i\}_{i=1}^{\mathcal{N}_c}$ be the set of equidistant collocation points such that $x_j = \frac{2\pi(j-1)}{\mathcal{N}_c}$. We test our

Table 3 1D SPDE problem: error and computing costs of our Algorithms COFRB_ODE and COFRB_PDE

$\mathcal{N}_c$	(M, K)	ε_{tol}	$N/\mathcal{N}$ COFRB_ODE	COFRB_PDE	e_2^{RBM} COFRB_ODE	COFRB_PDE	e_2^{ORI}	CPU Time COFRB_ODE	COFRB_PDE	Full
Forward Euler time discretization										
32	(9, 9)	1.00E−02	74/48620	74/48620	1.52E−04	2.57E−04	4.98E−04	2.82E−01	1.40E−01	1
	(10, 10)	1.00E−02	81/184756	81/184756	4.16E−04	5.07E−04	3.01E−04	2.14E−01	1.21E−01	1
64	(9, 9)	1.00E−02	82/48620	82/48620	1.98E−04	1.04E−04	4.98E−04	4.08E−01	8.75E−02	1
	(10, 10)	1.00E−02	88/184756	88/184756	5.20E−04	3.10E−04	3.01E−04	1.21E−01	5.10E−02	1
Crank–Nicolson time discretization										
32	(9, 9)	1.00E−02	74/48620	74/48620	1.53E−04	2.61E−04	3.67E−04	1.56E−01	6.20E−02	1
	(10, 10)	1.00E−02	81/184756	81/184756	4.18E−04	5.14E−04	1.70E−04	5.10E−02	2.55E−02	1
64	(9, 9)	1.00E−02	82/48620	82/48620	1.43E−04	1.07E−04	3.67E−04	6.13E−02	1.64E−02	1
	(10, 10)	1.00E−02	88/184756	88/184756	5.00E−04	1.42E−04	1.70E−04	3.00E−02	1.13E−02	1

Table 4 1D SPDE problem: error and computing costs of our Algorithm COFRB_PDE for varying polynomial degree M

$\mathcal{N}_c$	(M, K)	ε_{tol}	$N/\mathcal{N}$	e_2^{RBM}	e_2^{ORI}	CPU Time COFRB_PDE	Full
Forward Euler time discretization							
32	(6, 9)	9.84E−02	38/5005	3.62E−03	4.48E−03	8.23E−01	1
	(7, 9)	7.18E−02	46/11440	1.62E−03	1.97E−03	4.98E−01	1
	(8, 9)	3.43E−02	51/24310	4.96E−04	9.41E−04	2.92E−01	1
	(9, 9)	2.02E−02	57/48620	3.96E−04	4.98E−04	1.64E−01	1
	(10, 9)	7.06E−03	82/92378	2.37E−04	3.04E−04	1.76E−01	1
	(11, 9)	6.36E−03	84/167960	2.01E−04	2.33E−04	1.32E−01	1
Crank–Nicolson time discretization							
32	(6, 9)	1.02E−01	38/5005	3.62E−03	4.38E−03	1.75E−01	1
	(7, 9)	7.50E−02	45/11440	1.62E−03	1.85E−03	1.07E−01	1
	(8, 9)	3.58E−02	50/24310	4.96E−04	8.15E−04	6.25E−02	1
	(9, 9)	1.54E−02	62/48620	3.24E−04	3.67E−04	5.52E−02	1
	(10, 9)	6.65E−03	83/92378	1.20E−04	1.70E−04	6.84E−02	1
	(11, 9)	4.36E−03	89/167960	7.37E−05	8.30E−05	1.42E−02	1

algorithms for collocation points $\mathcal{N}_c = 32, 64$ with (M, K) pairs being (9, 9) and (10, 10) respectively. Both forward Euler and Crank–Nicolson methods are implemented up to $T = 5$ with time step size $\Delta t = 10^{-3}$ i.e. time steps $n = 5000$. The results, shown in Fig. 2, indicates that the COFRB_PDE method is effective for PDE. We only need to resolve *about* 0.2% of all the multi-indices through the high-fidelity simulation. Everybody else can be supplied by the reduced solver without degrading the accuracy. We compute $e_2^{RBM} = \mathbb{E}[\|u_{M,K}^{N}(5,\cdot) - u_{M,K}^{\mathcal{N}}(5,\cdot)\|_{l^2}^2]$ as proxy of error, where the discrete L^2 norm for v is defined by

$$\|v\|_{l^2}^2 := \frac{2\pi}{\mathcal{N}_c}\sum_{j=1}^{\mathcal{N}_c}(v(x_j))^2$$

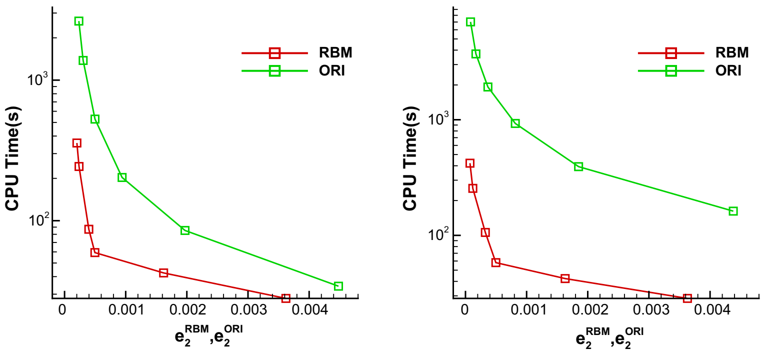

Fig. 3 CPU times (or COFRB_PDE and the original algorithm) versus errors for the 1D SPDE problem with $(M, K, \mathcal{N}_c) = (6, 9, 32), (7, 9, 32), \ldots, (11, 9, 32)$. Left: Forward Euler time discretization; Right: Crank–Nicolson time discretization

and for comparison, we also compute the error $e_2^{ORI} = \mathbb{E}[\|u_{N,K}^{ORI}(5,\cdot) - u(5,\cdot)\|_{l^2}^2]$ where u is the exact solution of the SPDE. In Table 3, we find that when the errors e_2^{RBM} and e_2^{ORI} reach the same order of magnitude, our algorithm is much faster than the original algorithm. This result also stands for the Crank–Nicolson method. We normalize the time with respect to that for one truth solve. We see that our algorithm achieves savings of two orders of magnitude. And we also compare Algorithm 2 with 3, to verify that Algorithm 3 not only saves more storage but is also more efficient than the Algorithm 2. In addition, we demonstrate the scalability of our Algorithm COFRB_PDE. Toward that end, we fix the number of random variables $K = 9$ while increasing polynomial degree M. The results are

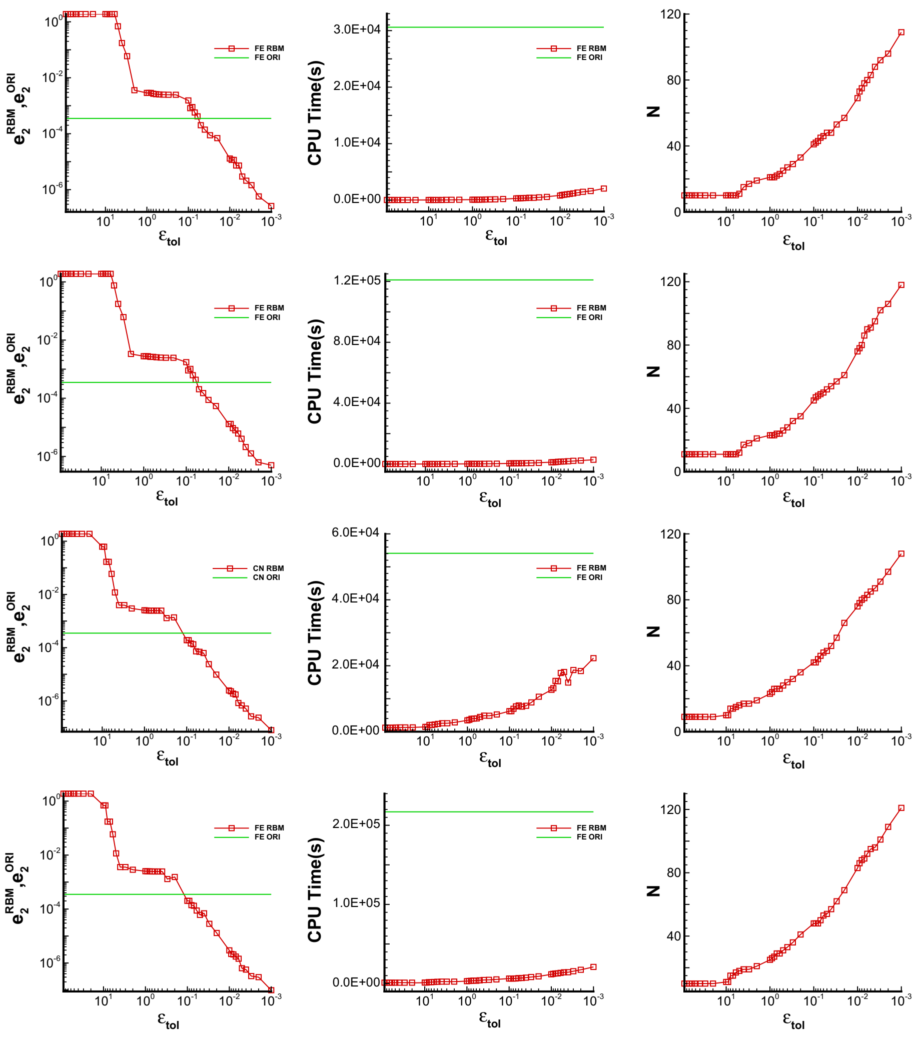

Fig. 4 COFRB for 2D SPDE problem based on Forward Euler method with $(M, K, \mathcal{N}_c) = (9, 9, 32 \times 32)$ (Row 1), $(10, 10, 32 \times 32)$ (Row 2), $(8, 8, 64 \times 64)$ (Row 3), and $(9, 9, 64 \times 64)$ (Row 4). From left to right are histories of convergence of e_2^{RBM} and e_2^{ORI}, CPU time of Algorithm 3 as a function of ε_{tol}, and number of chosen multi-indices as a function of ε_{tol}

listed in the Table 4. We observe that, to reach the same level of accuracy as the original algorithm, the number of snapshots N does increase as expected since the original algorithm is getting more accurate. However, this increase in N is quite gradual. As a consequence, our algorithm is actually getting slightly more efficient overall in comparison to the original algorithm which becomes more costly faster as M increases. We visualize these trends in Fig. 3.

Example 5.3 (Linear 2D parabolic PDE) We consider two-dimensional linear transport type SPDE. Consider the following distribution-free equation, equipped with periodic boundary condition.

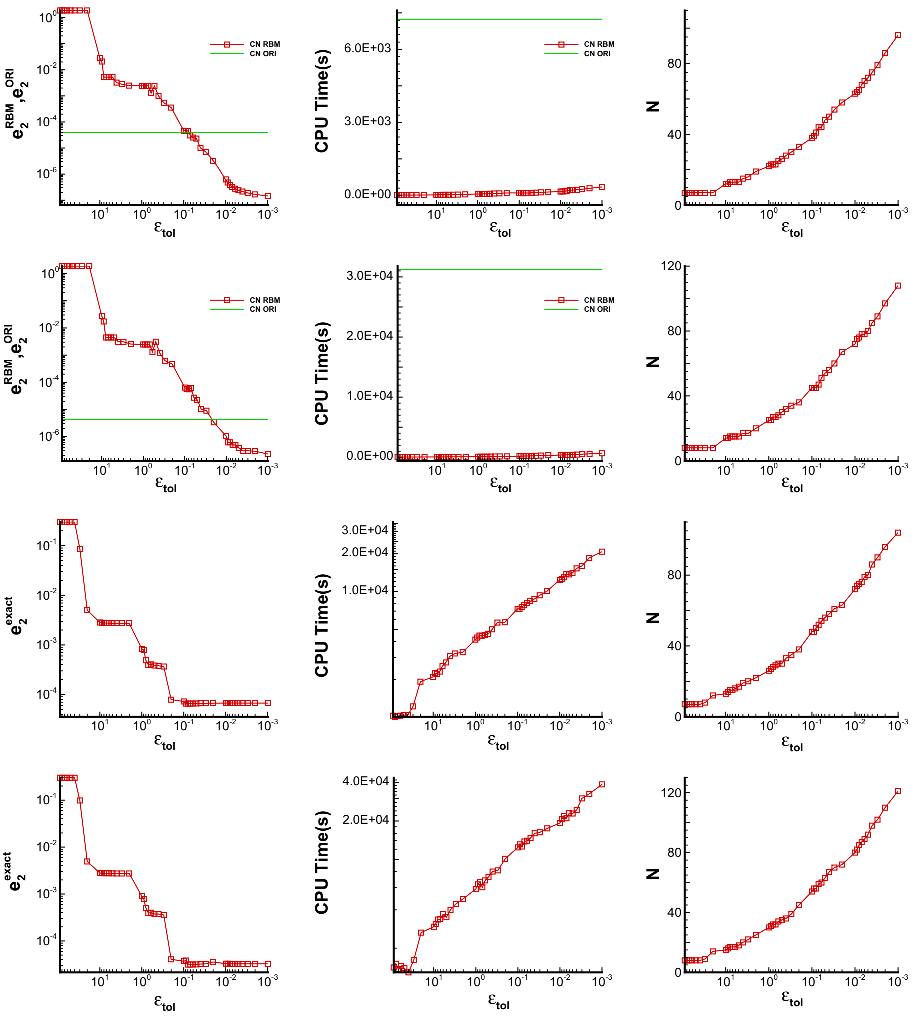

Fig. 5 COFRB for 2D SPDE problem based on Crank Nicolson method with $(M, K, \mathcal{N}_c) = (6, 6, 32 \times 32)$ (Row 1), $(7, 7, 32 \times 32)$ (Row 2), $(6, 6, 64 \times 64)$ (Row 3), and $(7, 7, 64 \times 64)$ (Row 4). From left to right are histories of convergence of e_2^{RBM} and e_2^{ORI}, CPU time of Algorithm 3 as a function of ε_{tol}, and number of chosen multi-indices as a function of ε_{tol}

$$\partial_t u(t,x,y) = \left(\frac{1}{2}\partial_x^2 + \cos(x)\partial_y\right) u + \partial_x u \diamond \dot{\mathfrak{N}}(t), \quad (t,x,y) \in [0,T] \times [0,2\pi]^2 \quad (5.5)$$

$$u(0,x,y) = \sin(2x)\sin(y), \quad (x,y) \in [0,2\pi]^2. \quad (5.6)$$

Fourier collocation method is used for spatial discretization with $\mathcal{N}_x = 32, 64$ collocation points in each dimension. Equidistant collocation points are denoted by $x_j = y_j = \frac{2\pi(j-1)}{\mathcal{N}_x}$. The propagator system is then evolved to $T = 0.2$. We test Algorithm 3 for both forward Euler and Crank Nicolson methods. For forward Euler, we set time steps $n = 5000$, $(M,K) = (9,9), (10,10)$ for $\mathcal{N}_x = 32$ and $(M,K) = (8,8), (9,9)$ for $\mathcal{N}_x = 64$. For Crank–Nicolson, n is set to be 400, (M,K) to be $(6,6), (7,7)$ for $\mathcal{N}_x = 32$ and $\mathcal{N}_x = 64$. We also compute mean square truncation error, $e_2^{RBM} = \mathbb{E}[\|u_{M,K}^{N}(0.2,\cdot,\cdot) - u_{M,K}^{\mathcal{N}}(0.2,\cdot,\cdot)\|_{l^2}^2]$, $e_2^{ORI} = \mathbb{E}[\|u_{M,K}^{\mathcal{N}}(0.2,\cdot,\cdot) - u(0.2,\cdot,\cdot)\|_{l^2}^2]$ where the discrete L^2 norm is $\|v\|_{l^2}^2 := \frac{4\pi^2}{\mathcal{N}_x^2}\sum_{i=1}^{\mathcal{N}_x}\sum_{j=1}^{\mathcal{N}_x}(v(x_i,y_j))^2$. We plot the histories of convergence, computation time, and the number of chosen multi-indices as functions of ε_{tol} in Figs. 4 (forward Euler) and 5 (Crank Nicolson). And we also plot the variance of exact solution and the RBM solutions in Figs. 6 and 7. It is clear the COFRB method captures the variances well.

Table 5, where we list the computational time, shows that the COFRB_PDE algorithm achieves savings of two orders of magnitude when the RBM solution reaches the same accuracy level as truth approximation. We note that Algorithm 2 becomes infeasible for the two-dimensional problem due to the memory constraint. For Crank–Nicolson method with

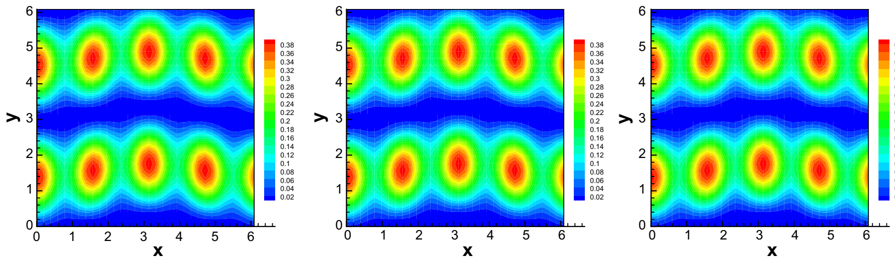

Fig. 6 Variance of 2D SPDE problem with $\mathcal{N}_c = 32 \times 32$: from left to right are exact solution, RBM solutions with Algorithm 3 based on Forward Euler method with $(M,K) = (9,9)$ and Crank Nicolson method with $(M,K) = (6,6)$

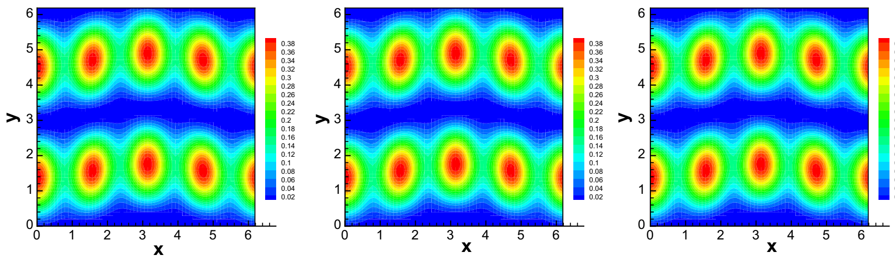

Fig. 7 Variance of 2D SPDE problem with $\mathcal{N}_c = 64 \times 64$: from left to right are exact solution, RBM solutions with Algorithm 3 based on Forward Euler method with $(M,K) = (8,8)$ and Crank Nicolson method with $(M,K) = (6,6)$

Table 5 2D SPDE problem: error and computing costs of Algorithm COFRB_PDE

Method	$\mathcal{N}_x \times \mathcal{N}_x$	(M, K)	ε_{tol}	N	$\mathcal{N}$	e_2^{RBM}	e_2^{ORI}	CPU Time	
								COFRB_PDE	Full
FE	32×32	(9, 9)	5.00E−02	48	48620	2.01E−04	3.51E−04	1.37E−02	1
		(10, 10)	3.00E−02	52	184756	2.04E−04	3.49E−04	4.60E−03	1
	64×64	(8, 8)	1.00E−01	42	12870	1.91E−04	3.50E−04	1.15E−01	1
		(9, 9)	1.00E−01	48	48620	2.01E−04	3.50E−04	2.92E−02	1
CN	32×32	(6, 6)	1.00E−01	38	924	4.62E−05	3.89E−05	1.49E−02	1
		(7, 7)	2.00E−02	67	3432	3.35E−06	4.29E−06	1.03E−02	1
	64×64	(6, 6)	1.00E−01	48	924	7.24E−05		7301.45	NAN
		(7, 7)	1.00E−01	54	3432	3.69E−05		12399.77	NAN

$\mathcal{N}_c = 64 \times 64$, the full problem is out of reach. Therefore, we compute the error between the RBM solutions and the exact solutions, e_2^{exact}, and show them in the Table 5.

$$e_2^{\text{exact}} = \mathbb{E}\left[\|u_{M,K}^N(0.2,\cdot,\cdot) - u(0.2,\cdot,\cdot)\|_{l^2}^2\right]. \tag{5.7}$$

The fact that the reduced solver COFRB_PDE reaches five digits of accuracy when the full solver or the COFRB_ODE is out of reach underscores the power of the COFRB_PDE method.

6 Concluding Remarks

In this paper, we develop new reduced basis methods for SODEs and SPDEs, called COFRB_ODE and COFRB_PDE respectively. The main features include a new space-time-like treatment of time in the numerical schemes for ODEs and PDEs, an accurate yet efficient compression technique for the spatial component of the space-time RBM bases, a non-conventional "parameterization" of a non-parametric problem, and finally a RBM that is free of any dedicated offline procedure yet still efficient. The numerical experiments corroborate the effectiveness and robustness of our algorithms. Future work includes extension of the current approaches to nonlinear and purely transport problems, and convergence analysis of the error with respect to the number of chosen multi-indices.

Appendix A: Implementation of Algorithm 2

In this appendix, we present some details for efficient implementation of Algorithm 2 whose steps 5, 6, and 10 are key.

A.1 Step 5

For the forward Euler time discretization, we rewrite the right hand side of (3.8) in the matrix form

$$\begin{aligned}\vec{f}_\alpha &= \begin{pmatrix} u_\alpha^0 + \Delta t u_\alpha^0 + \Delta t \sum_{\varepsilon_k \le \alpha} m_k^0 u_{\alpha-\varepsilon_k}^0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \Delta t \sum_{\varepsilon_k \le \alpha} \begin{pmatrix} 0 & & & \\ & m_k^1 & & \\ & & \ddots & \\ & & & m_k^{n-1} \end{pmatrix} \begin{pmatrix} 0 \\ u_{\alpha-\varepsilon_k}^1 \\ \vdots \\ u_{\alpha-\varepsilon_k}^{n-1} \end{pmatrix} \\ &= \vec{f}_\alpha^0 + \Delta t \sum_{\varepsilon_k \le \alpha} M_k^0 \vec{U}^0 \vec{c}_{\alpha-\varepsilon_k} \end{aligned} \tag{6.1}$$

where $M_k^0 = \text{diag}\{0, m_k^1, \ldots, m_k^{n-1}\}$ and

$$\vec{U}^0 = \begin{pmatrix} 0 & 0 & \ldots & 0 \\ u_{\alpha_1}^1 & u_{\alpha_2}^1 & \ldots & u_{\alpha_i}^1 \\ \vdots & \vdots & \vdots & \vdots \\ u_{\alpha_1}^{n-1} & u_{\alpha_2}^{n-1} & \ldots & u_{\alpha_i}^{n-1} \end{pmatrix} = \left(\vec{U}_{\alpha_1}^0 \; \vec{U}_{\alpha_2}^0 \ldots \vec{U}_{\alpha_i}^0 \right). \tag{6.2}$$

For the Crank–Nicolson method, we can also rewrite (3.10) similarly

$$\vec{f}_\alpha = \begin{pmatrix} u_\alpha^0 + \frac{1}{2}\Delta t u_\alpha^0 + \frac{1}{2}\Delta t \sum_{\varepsilon_k \le \alpha} m_k^0 u_{\alpha-\varepsilon_k}^0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \frac{1}{2}\Delta t \sum_{\varepsilon_k \le \alpha} \begin{pmatrix} 0 & & & \\ & m_k^1 & & \\ & & \ddots & \\ & & & m_k^{n-1} \end{pmatrix} \begin{pmatrix} 0 \\ u_{\alpha-\varepsilon_k}^1 \\ \vdots \\ u_{\alpha-\varepsilon_k}^{n-1} \end{pmatrix}$$

$$+ \frac{1}{2}\Delta t \sum_{\varepsilon_k \le \alpha} \begin{pmatrix} m_k^1 & & & \\ & m_k^2 & & \\ & & \ddots & \\ & & & m_k^n \end{pmatrix} \begin{pmatrix} u_{\alpha-\varepsilon_k}^1 \\ u_{\alpha-\varepsilon_k}^2 \\ \vdots \\ u_{\alpha-\varepsilon_k}^n \end{pmatrix}$$

$$= \vec{f}_\alpha^0 + \frac{1}{2}\Delta t \sum_{\varepsilon_k \le \alpha} \left(M_k^0 \vec{U}^0 + M_k \vec{U} \right) \vec{c}_{\alpha-\varepsilon_k} \tag{6.3}$$

where $M_k = \mathrm{diag}\{m_k^1, m_k^2, \ldots, m_k^n\}$.

Due to the hierarchical nature of the RB space, the RB stiffness matrix $A_{RB} = \vec{U}^T A^T A \vec{U}$ can be formed by appending a row and a column each time a new reduced basis $\vec{U}_{\alpha_{i+1}}$ is identified. That is, we exploit the following identity.

$$\begin{pmatrix} \vec{U}^T \\ (\vec{U}_{\alpha_{i+1}})^T \end{pmatrix} A^T A \left(\vec{U}, \vec{U}_{\alpha_{i+1}} \right) = \begin{pmatrix} \vec{U}^T A^T A \vec{U} & \vec{U}^T A^T A \vec{U}_{\alpha_{i+1}} \\ (\vec{U}_{\alpha_{i+1}})^T A^T A \vec{U} & (\vec{U}_{\alpha_{i+1}})^T A^T A \vec{U}_{\alpha_{i+1}} \end{pmatrix}. \tag{6.4}$$

For the RB right hand side $\vec{f}_{RB} = \vec{U}^T A^T \vec{f}_\alpha$, recognizing that

$$\vec{f}_{RB} = \vec{U}^T A^T \vec{f}_\alpha = \begin{cases} \vec{U}^T A^T \vec{f}_\alpha^0 + \Delta t \sum_{\varepsilon_k \le \alpha} \vec{U}^T A^T M_k^0 \vec{U}^0 \vec{c}_{\alpha-\varepsilon_k} & \text{for FE} \\ \vec{U}^T A^T \vec{f}_\alpha^0 + \frac{\Delta t}{2} \sum_{\varepsilon_k \le \alpha} \vec{U}^T A^T (M_k^0 \vec{U}^0 + M_k \vec{U}) \vec{c}_{\alpha-\varepsilon_k} & \text{for CN} \end{cases} \tag{6.5}$$

we can also exploit the hierarchical nature to gradually build up $\vec{U}^T A^T$, $\vec{U}^T A^T M_k^0 \vec{U}^0$ and $\vec{U}^T A^T M_k \vec{U}$.

A.2 Step 6

Efficient and accurate evaluation of the error estimator is critical for the correct identification of the key multi-indices and thus the convergence of the COFRB algorithms. The classical approach of computing the square norm of $A\vec{U}\vec{c}_\alpha - \vec{f}_\alpha$ and then expanding it to enable an offline-online decomposition leads to numerical instability [11]. In our setting, it will result in this norm being negative. To detail the numerically stable method, we follow [11]. Noting that we can assume $\vec{f}_\alpha^0 = 0$ since $u_{\alpha-\varepsilon_k}^0 = 0$ when $|\alpha| \ge 2$ due to the initial condition being deterministic and all α with $|\alpha| = 1$ are usually chosen meaning the residual will be zero, we can rewrite the residual $A\vec{U}\vec{c}_\alpha - \vec{f}_\alpha$ as

$$A\vec{U}\vec{c}_\alpha - \vec{f}_\alpha = \mathcal{B}\tilde{C}_\alpha \tag{6.6}$$

where

$$\mathcal{B} = \Big(A\vec{U}_{\alpha_1}, M_1^0 \vec{U}_{\alpha_1}^0, M_2^0 \vec{U}_{\alpha_1}^0, \ldots, M_K^0 \vec{U}_{\alpha_1}^0, A\vec{U}_{\alpha_2}, M_1^0 \vec{U}_{\alpha_2}^0, \ldots, M_K^0 \vec{U}_{\alpha_2}^0, \ldots, A\vec{U}_{\alpha_i}, M_1^0 \vec{U}_{\alpha_i}^0, \ldots, M_K^0 \vec{U}_{\alpha_i}^0 \Big) \tag{6.7}$$

$$\begin{aligned}\tilde{C}_\alpha = \big(&c_1(\alpha), -\Delta t\delta_{\alpha-\varepsilon_1} c_1(\alpha-\varepsilon_1), -\Delta t\delta_{\alpha-\varepsilon_2} c_1(\alpha-\varepsilon_2), \ldots, -\Delta t\delta_{\alpha-\varepsilon_K} c_1(\alpha-\varepsilon_K),\\ &c_2(\alpha), -\Delta t\delta_{\alpha-\varepsilon_1} c_2(\alpha-\varepsilon_1), -\Delta t\delta_{\alpha-\varepsilon_2} c_2(\alpha-\varepsilon_2), \ldots, -\Delta t\delta_{\alpha-\varepsilon_K} c_2(\alpha-\varepsilon_K), \ldots,\\ &c_i(\alpha), -\Delta t\delta_{\alpha-\varepsilon_1} c_i(\alpha-\varepsilon_1), -\Delta t\delta_{\alpha-\varepsilon_2} c_i(\alpha-\varepsilon_2) \ldots, -\Delta t\delta_{\alpha-\varepsilon_K} c_i(\alpha-\varepsilon_K)\big)^T\end{aligned} \tag{6.8}$$

and $\delta_{\alpha-\varepsilon_k}$ is defined as follows

$$\delta_{\alpha-\varepsilon_k} = \begin{cases} 1, & \text{if } \varepsilon_k \le \alpha \text{ is true} \\ 0, & \text{otherwise.} \end{cases} \tag{6.9}$$

We adopt the rank-revealing QR factorization through modified Gram–Schmidt for matrix $\mathcal{B}$.

$$\mathcal{B} = \mathcal{Q}\mathcal{R}, \quad \mathcal{Q} \in \mathbb{R}^{n\times \text{rank}(\mathcal{B})}, \quad \mathcal{R} \in \mathbb{R}^{\text{rank}(\mathcal{B})\times M(K+1)}. \tag{6.10}$$

where $\text{rank}(\mathcal{B}) \le M(K+1)$ is the rank of matrix $\mathcal{B}$. Then

$$\|A\vec{U}\vec{c}_\alpha - \vec{f}_\alpha\|^2 = \tilde{C}_\alpha^T \mathcal{R}^T \mathcal{R} \tilde{C}_\alpha. \tag{6.11}$$

The cost of computing this term is independent of n if we pre-compute matrix $\mathcal{R}$. Notice that the matrices $\mathcal{Q}$ and $\mathcal{R}$ must also be gradually expanded similar to the way RB stiffness matrix is handled recognizing that QR factorization can be expanded in a hierarchical fashion as the data matrix is expanded.

A.3 Step 10

When a new multi-index α_{i+1} is deemed a candidate for addition to the RB space i.e. $\Delta_i(\alpha_{i+1}) > \varepsilon_{\text{tol}}$, we take $v_0 = \vec{U}_{\alpha_{i+1}}$. Then the modified Gram–Schmidt follows

$$\tilde{v}_j = v_{j-1} - < v_{j-1}, \vec{U}_{\alpha_i} > \vec{U}_{\alpha_i}, \tag{6.12}$$

$$v_j = \frac{\tilde{v}_j}{\|\tilde{v}_j\|}, \quad j = 1, 2, \ldots, i \tag{6.13}$$

where $< \cdot, \cdot >$ denotes the inner product. If $\|\tilde{v}_j\| = 0$ for some $j \le i$, we discard this candidate and set the (RB) coefficients for this multi-index as

$$\vec{c}_{\alpha_{i+1}} = \left(< v_0, \vec{U}_{\alpha_1} >, < v_1, \vec{U}_{\alpha_2} > \|\tilde{v}_1\|, \ldots, < v_{j-1}, \vec{U}_{\alpha_j} > \Pi_{l=1}^{j-1} \|\tilde{v}_l\|, 0, \ldots, 0\right).$$

Otherwise, we update $\vec{U}_{\alpha_{i+1}} = v_i$ and augment the basis matrix, and set the (RB) coefficients

$$\vec{c}_{\alpha_{i+1}} = \left(< v_0, \vec{U}_{\alpha_1} >, < v_1, \vec{U}_{\alpha_2} > \|\tilde{v}_1\|, \ldots, < v_{i-1}, \vec{U}_{\alpha_i} > \Pi_{j=1}^{i-1} \|\tilde{v}_j\|, \Pi_{j=1}^{i} \|\tilde{v}_j\|\right).$$

Appendix B: Implementation of Algorithm 3

In this appendix, we present some details for the efficient implementation of Algorithm 3. The RB stiffness matrix and vector A_{RB} and $\vec{f}_{RB} = \vec{W}^T \vec{V}^T A^T \vec{f}_\alpha$ are expanded in the same fashion as in Appendix A. For calculating $\Delta_i(\alpha)$, slight changes are necessary. Under the same assumption $\vec{f}_\alpha^0 = 0$, we have

$$\|A\vec{V}\vec{W}\vec{c}_\alpha - \vec{f}_\alpha\|^2 = \|\mathcal{B}\tilde{C}_\alpha\|^2 \tag{6.14}$$

$$= \tilde{C}_\alpha^T \mathcal{B}^T \mathcal{B}\tilde{C}_\alpha, \tag{6.15}$$

where

$$\mathcal{B} = \Big(A\vec{V}_{\alpha_1}^m \vec{W}_{\alpha_1}, M_1^0\vec{V}_{\alpha_1}^m \vec{W}_{\alpha_1}^0, M_2^0\vec{V}_{\alpha_1}^m \vec{W}_{\alpha_1}^0, \ldots, M_K^0\vec{V}_{\alpha_1}^m \vec{W}_{\alpha_1}^0, A\vec{V}_{\alpha_2}^m \vec{W}_{\alpha_2}, M_1^0\vec{V}_{\alpha_2}^m \vec{W}_{\alpha_2}^0, \ldots,$$
$$M_K^0\vec{V}_{\alpha_2}^m \vec{W}_{\alpha_2}^0, \ldots, A\vec{V}_{\alpha_i}^m \vec{W}_{\alpha_i}, M_1^0\vec{V}_{\alpha_i}^m \vec{W}_{\alpha_i}^0, \ldots, M_K^0\vec{V}_{\alpha_i}^m \vec{W}_{\alpha_i}^0\Big). \tag{6.16}$$

We first use hierarchical expansion to form the matrix $\mathcal{B}^T\mathcal{B}$ which is of size $i(K+1)$-by-$i(K+1)$. Since it is symmetrical and positive definite, there exists an orthogonal decomposition. Hence,

$$\tilde{C}_\alpha^T \mathcal{B}^T \mathcal{B}\tilde{C}_\alpha = \tilde{C}_\alpha^T P^T \Lambda P \tilde{C}_\alpha = \omega_\alpha^T \Lambda \omega_\alpha \tag{6.17}$$

where P is an $i(K+1)$-by-$i(K+1)$ orthogonal matrix, Λ is a diagonal matrix whose diagonal elements are square of singular values of matrix $\mathcal{B}$ denoted by s_j and $\omega_\alpha = P\tilde{C}_\alpha$. Thus the residue

$$\|A\vec{U}\vec{c}_\alpha - \vec{f}_\alpha\|^2 = \sum_{j=1}^{i(K+1)} s_j(\omega_\alpha)_j^2. \tag{6.18}$$

In this way, we only need to store the $i(K+1)$-by-$i(K+1)$ matrices $\mathcal{B}^T\mathcal{B}$, P and $i(K+1)$ diagonal elements of Λ.

References

1. Barrault, M., Nguyen, N.C., Maday, Y., Patera, A.T.: An “empirical interpolation” method: application to efficient reduced-basis discretization of partial differential equations. C. R. Acad. Sci. Paris **339**, 667–672 (2004)
2. Ben-Israel, A., Greville, T.N.: Generalized Inverses: Theory and Applications, vol. 15. Springer Science & Business Media, New York (2003)
3. Berkooz, P.H.G., Lumley, J.: The proper orthogonal decomposition in the analysis of turbulent flows. Ann. Rev. Fluid Mech. **25**(1), 539–575 (1993)
4. Bernard, Haasdonk: Convergence rates of the pod-greedy method. ESAIM: Math. Model. Numer. Anal. **47**(3), 859–873 (2013)
5. Binev, P., Cohen, A., Dahmen, W., Devore, R., Petrova, G., Wojtaszczyk, P.: Convergence rates for greedy algorithms in reduced basis methods. SIAM J. Math. Anal. **43**(3), 1457–1472 (2011)
6. Boyd, J. P.: Chebyshev and Fourier spectral methods, Courier Corporation, (2001)
7. Cameron, R.H., Martin, W.T.: The orthogonal development of non-linear functionals in series of Fourier-Hermite functionals. Ann. Math. **48**(2), 385–392 (1947)
8. Chen, Y., Gottlieb, S.: Reduced collocation methods: reduced basis methods in the collocation framework. J. Sci. Comput. **55**(3), 718–737 (2013)
9. Chen, Y., Hesthaven, J.S., Maday, Y., Rodríguez, J.: Certified reduced basis methods and output bounds for the harmonic maxwell’s equations. SIAM J. Sci. Comput. **32**(2), 970–996 (2010)
10. Chen, T., Rozovskii, B., Shu, C.-W.: Numerical solutions of stochastic pdes driven by arbitrary type of noise. Stoch. Partial Diff. Equ. Anal. Comput. **7**(1), 1–39 (2019)
11. Chen, Y., Jiang, J., Narayan, A.: A robust error estimator and a residual-free error indicator for reduced basis methods. Comput. Math. Appl. **77**, 1963–1979 (2019)
12. Di Nunno, G., Øksendal, B.K., Proske, F.: Malliavin Calculus for Lévy Processes with Applications to Finance, vol. 2. Springer, New York (2009)
13. Elman, H., Liao, Q.: Reduced basis collocation methods for partial differential equations with random coefficients. SIAM/ASA J. Uncertain. Quantif. **1**(1), 192–217 (2013)
14. Glas, S., Mayerhofer, A., Urban, K.: Two Ways to Treat Time in Reduced Basis Methods, pp. 1–16. Springer International Publishing, Cham (2017)

15. Grepl, M.A., Patera, A.T.: A posteriori error bounds for reduced-basis approximations of parametrized parabolic partial differential equations. ESAIM: Math. Modell. Numer. Anal. **39**(1), 157–181 (2005)
16. Haasdonk, B.: Chapter 2: Reduced Basis Methods for Parametrized PDEsNA Tutorial Introduction for Stationary and Instationary Problems, pp. 65–136
17. Haasdonk, B., Ohlberger, M.: Reduced basis method for finite volume approximations of parametrized linear evolution equations. ESAIM: Math. Model. Numer. Anal. **42**(2), 277–302 (2008)
18. Hesthaven, J., Rozza, G., Stamm, B.: Certified Reduced Basis Methods for Parametrized Problems. Springer Briefs in Mathematics. Springer (2015)
19. Holden, H., Øksendal, B., Ubøe, J., Zhang, T.: Stochastic Partial Differential Equations. Springer, New York (1996)
20. Hou, T.Y., Luo, W., Rozovskii, B., Zhou, H.-M.: Wiener chaos expansions and numerical solutions of randomly forced equations of fluid mechanics. J. Comput. Phys. **216**(2), 687–706 (2006)
21. Jiang, J., Chen, Y., Narayan, A.: Offline-Enhanced reduced basis method through adaptive construction of the surrogate training set. J. Sci. Comput. **73**(2), 853–875 (2017)
22. Lototsky, S., Rozovskii, B.: Stochastic differential equations: a wiener chaos approach. In From Stochastic Calculus to Mathematical Finance, pp. 433–506. Springer, Berlin (2006)
23. Lototsky, S., Mikulevicius, R., Rozovskii, B.L.: Nonlinear filtering revisited: a spectral approach. SIAM J. Control Optim. **35**(2), 435–461 (1997)
24. Mikulevicius, R., Rozovskii, B.: On unbiased stochastic Navier–Stokes equations. Probab. Theory Relat. Fields **154**(3–4), 787–834 (2012)
25. Mikulevicius, R., Rozovskii, B.: On distribution free Skorokhod–Malliavin calculus. Stoch. Partial Diff. Equ. Anal. Comput. **4**(2), 319–360 (2016)
26. Prud'homme, C., Rovas, D., Veroy, K., Maday, Y., Patera, A.T., Turinici, G.: Reliable real-time solution of parametrized partial differential equations: reduced-basis output bound methods. J. Fluids Eng. **124**(1), 70–80 (2002)
27. Quarteroni, A., Manzoni, A., Negri, F.: Reduced Basis Methods for Partial Differential Equations. UNITEXT, vol. 92. Springer International Publishing, Cham (2016)
28. Rozza, G., Huynh, D.B.P., Patera, A.T.: Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. Arch. Comput. Methods Eng. **15**(3), 1 (2007)
29. Rozza, G., Huynh, D., Patera, A.: Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations: application to transport and continuum mechanics. Arch. Comput. Methods Eng. **15**(3), 229–275 (2008)
30. Urban, K., Patera, A.T.: An improved error bound for reduced basis approximation of linear parabolic problems. Math. Comput. **83**(288), 1599–1615 (2014)
31. Wick, G.-C.: The evaluation of the collision matrix. Phys. Rev. **80**(2), 268 (1950)
32. Willcox, K., Peraire, J.: Balanced model reduction via the proper orthogonal decomposition. AIAA J. **40**(11), 2323–2330 (2002)
33. Xiu, D.: Numerical Methods for Stochastic Computations: A Spectral Method Approach. Princeton University Press, Princeton (2010)
34. Xiu, D., Karniadakis, G.E.: The Wiener–Askey polynomial chaos for stochastic differential equations. SIAM J. Sci. Comput. **24**(2), 619–644 (2002)
35. Yano, M., Patera, A.T., Urban, K.: A space-time hp-interpolation-based certified reduced basis method for burgers' equation. Math. Models Methods Appl. Sci. **24**(09), 1903–1935 (2014)
36. Zhang, Z., Rozovskii, B., Tretyakov, M.V., Karniadakis, G.E.: A multistage wiener chaos expansion method for stochastic advection–diffusion–reaction equations. SIAM J. Sci. Comput. **34**(2), A914–A936 (2012)
37. Zhang, Z., Tretyakov, M.V., Rozovskii, B., Karniadakis, G.E.: 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)