Rank adaptive tensor recovery based model reduction for partial differential equations with high-dimensional random inputs

Kejun Tang[a], Qifeng Liao[b,*]

[a]*School of Information Science and Technology, ShanghaiTech University, Shanghai, China*
[b]*School of Information Science and Technology, ShanghaiTech University, Shanghai, China*

Abstract

This work proposes a systematic model reduction approach based on rank adaptive tensor recovery for partial differential equation (PDE) models with high-dimensional random parameters. Since the standard outputs of interest of these models are discrete solutions on given physical grids which are high-dimensional, we use kernel principal component analysis to construct stochastic collocation approximations in reduced dimensional spaces of the outputs. To address the issue of high-dimensional random inputs, we develop a new efficient rank adaptive tensor recovery approach to compute the collocation coefficients. Novel efficient initialization strategies for non-convex optimization problems involved in tensor recovery are also developed in this work. We present a general mathematical framework of our overall model reduction approach, analyze its stability, and demonstrate its efficiency with numerical experiments.

Keywords: tensor recovery; model reduction; PDEs; uncertainty quantification

1. Introduction

During the last few decades there has been a rapid development in surrogate and reduced order modelling for PDE systems with random inputs. The PDE systems are fundamental mathematical models describing complex physical and engineering problems, which can involve multiple disciplines, a large number of input parameters, and multiple sources of uncertainty. A main challenge of surrogate modelling for these PDE models is the so-called curse of dimensionality. First, due to the high complexity of practical problems, the random input parameters are typically high-dimensional. Second, the standard output of these PDE models is the spatial fields (e.g., temperature, pressure and velocity), and their fine resolution representation requires a large number of degrees of freedom, which make the output high-dimensional.

A type of widely used surrogate modelling approach for these PDE models is the stochastic spectral methods [1, 2, 3, 4, 5], while the high dimensionality of the random inputs causes difficulties in applying them. To alleviate the difficulty, modifications of these methods have been actively introduced by exploiting certain properties of the underlying problem. For example, sparse (generalized) polynomial chaos (gPC) expansions [6, 7, 8, 9, 10, 11] are developed through using the sparsity in spectral approximations. Moreover, the stochastic collocation method [3, 12, 4] is reformulated as a tensor style quadrature problem in [13], which shows that the corresponding collocation

coefficients can be efficiently computed through tensor recovery techniques. On the other hand, the high dimensionality in outputs poses challenges in both surrogate modelling and data storage. The surrogates proposed to resolve high-dimensional inputs as discussed above are typically restricted to problems with a single output. Naively extending them to high-dimensional outputs (building independent surrogates for multiple outputs) is computationally infeasible. For making progress, dimension reduction methods for the outputs gain a lot of interests. For example, principal component analysis (PCA) and kernel component analysis (kPCA) methods are successfully established for Gaussian process surrogates [14, 15]. Especially, since kPCA captures highly nonlinear low-rank structures in the output space, it can provide dramatically tight representation of the outputs [16, 15].

In this work, we focus on tensor recovery based stochastic collocation. As discussed in [13], gPC coefficients in stochastic collocation can be computed through inner products of weight tensors and data tensors (see section 2.2 for details), where the weight tensors are given but the data tensors are expensive to obtain. Instead of directly evaluating the expensive data tensor, tensor recovery here is to use a small number of entries of the data tensor to recover the whole tensor [17, 18, 19]. A popular recovery strategy is developed in [17] based on canonical polyadic (CP) decomposition. In this recovery approach, the CP rank of the underlying tensor needs to be known a priori, which limits its application to our PDE models where the corresponding CP ranks are not given. For this purpose, we develop a novel rank adaptive tensor recovery (RATR) approach, which do not require any prior information for the tensor ranks. Moreover, as this kind of tensor recovery procedure requires solving a non-convex optimization problem [13], initialization strategies for this kind of optimization problem are crucial for successful recovery. In our RATR approach, new efficient initializaiton strategies are proposed based on a hierarchical rank-one updating procedure, and their stability is theoretically proven in this work.

The aim of this paper is to develop a systematic model reduction framework to curb this challenging high-dimensional input-output problem, and our overall procedure is as follows. First, kPCA is conducted for the outputs, which gives their reduced-dimensional representations. After that, for each kPCA mode, RATR based stochastic collocation is proposed to construct sparse gPC expansions for each kPCA mode. The inverse mapping method introduced in [16, 15] is finally adopted to construct an overall estimates of the outputs in the high-dimensional space. To summarize, the main contributions of this work are three-fold: first, stochastic collocation methods are reformulated with manifold learning for high-dimensional outputs; second, a novel rank adaptive tensor recovery (RATR) approach is proposed to recover tensors without knowing their ranks a priori; third, new efficient initialization strategies for RATR are proposed and their stability is analyzed.

The rest of the paper is organized as follows. In the next section, we formulate stochastic collocation methods for stochastic PDEs based on manifold learning. Details of tensors and standard tensor recovery approaches are introduced in section 3. Our main algorithms and analysis for RATR and the overall RATR-collocation surrogate are presented in section 4. In section 5, we demonstrate the efficiency of our RATR-collocation approach for stochastic diffusion and incompressible flow problems. Finally section 6 concludes the paper.

2. Problem setting and stochastic collocation based on manifold learning

Let D denote a spatial domain (in $\mathbb{R}^2$ or $\mathbb{R}^3$) which is bounded, connected and with a polygonal boundary ∂D, and x denote a spatial variable. Let $\boldsymbol{\xi}$ be a vector which collects a finite number of random variables. The dimension of $\boldsymbol{\xi}$ is denoted by d, i.e., we write $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_d]^T$. The probability density function of $\boldsymbol{\xi}$ is denoted by $\pi(\boldsymbol{\xi})$. In this paper, we restrict our attention to the situation that $\boldsymbol{\xi}$ has a bounded and connected support. Without loss of generality, we next assume the support of $\boldsymbol{\xi}$ to be I^d where $I := [-1, 1]$, since any bounded connected domain in $\mathbb{R}^d$ can be mapped to I^d. The physics of problems considered in this paper are governed by a PDE over the spatial domain D and boundary conditions on the boundary ∂D. This PDE problem is stated as: find $u(x, \boldsymbol{\xi}) : D \times I^d \to \mathbb{R}$, such that

$$\mathcal{L}(x, \boldsymbol{\xi}; u(x, \boldsymbol{\xi})) = f(x) \qquad \forall (x, \boldsymbol{\xi}) \in D \times I^d, \tag{1}$$

$$\mathfrak{b}(x, \boldsymbol{\xi}; u(x, \boldsymbol{\xi})) = g(x) \qquad \forall (x, \boldsymbol{\xi}) \in \partial D \times I^d, \tag{2}$$

where $\mathcal{L}$ is a partial differential operator and $\mathfrak{b}$ is a boundary operator, both of which can have random coefficients. f is the source function and g specifies the boundary conditions. We also define output quantities of interest. For each realization of $\boldsymbol{\xi}$, if the deterministic version of (1)–(2) is solved using a high-fidelity numerical scheme (simulator), for example finite element and difference methods, a natural definition of the output is the discrete solution. A high-fidelity discrete solution is also called a *snapshot* and can be represented as $\boldsymbol{y} = [u(x^{(1)}, \boldsymbol{\xi}), \ldots, u(x^{(N_h)}, \boldsymbol{\xi})]^T \in \mathbb{R}^{N_h}$, where $u(x^{(i)}, \boldsymbol{\xi}), i = 1, \ldots, N_h$ denotes the value of $u(x, \boldsymbol{\xi})$ at a specified location on a spatial grid and N_h refers to the spatial degrees of freedom. The manifold consisting of all snapshots is denoted by $\mathcal{M} \subset \mathbb{R}^{N_h}$, and it is assumed to be smooth. A PDE simulator can be viewed as a mapping $\chi : I^d \to \mathcal{M}$, where I^d and $\mathcal{M} \in \mathbb{R}^{N_h}$ are the input space and the output manifold respectively, and we denote it as $\chi(\boldsymbol{\xi}) = \boldsymbol{y} = [u(x^{(1)}, \boldsymbol{\xi}), \ldots, u(x^{(N_h)}, \boldsymbol{\xi})]^T \in \mathcal{M}$ for an arbitrary realization of the input $\boldsymbol{\xi} \in I^d$.

The goal of this study is to build surrogates for conducting uncertainty qualification (UQ) of the output $\boldsymbol{y}$, given limited training data points $\boldsymbol{y}^{(j)} = \chi(\boldsymbol{\xi}^{(j)})$ for $j = 1 : N_t$ where N_t is the size of a training data set. We focus on the challenging situations that the input and the output are both high-dimensional. To make progress, we reformulate the stochastic collocation surrogates [4, 3] based on manifold learning and tensor recovery quadrature. Manifold learning gives a reduced dimension representation for the output space through kernel principal component analysis (kPCA) and inverse mappings [20, 21, 15], and tenor recovery provides estimates of collocation coefficients associated with high-dimensional random parameters through exploiting low rank structures in these coefficients [13]. The rest of this section is to discuss the manifold learning based collocation and the setting of tensor formulation, while detailed tensor recovery methods and our new rank adaptive schemes are presented in the next two sections.

2.1. Kernel principal component analysis (kPCA)

To simplify the presentation, the given training data are denoted by $\boldsymbol{y}^{(j)} = \chi(\boldsymbol{\xi}^{(j)})$ for $j = 1, \ldots, N_t$. Following [20, 22, 15], the kernel principal component analysis (kPCA) proceeds through two steps: mapping the training data to a higher-dimensional feature space, and performing linear principal component analysis (PCA) in the feature space.

Denoting the feature space by $\mathcal{F}$, we define a mapping $\Gamma : \mathcal{M} \to \mathcal{F}$, which maps each training data point $\boldsymbol{y}^{(j)} \in \mathcal{M}$ to $\Gamma(\boldsymbol{y}^{(j)}) \in \mathcal{F}$ for $j = 1, \ldots, N_t$. A covariance matrix of the mapped data is defined as

$$C_{\mathcal{F}} := \frac{1}{N_t} \sum_{j=1}^{N_t} \widetilde{\Gamma}\left(\boldsymbol{y}^{(j)}\right) \widetilde{\Gamma}\left(\boldsymbol{y}^{(j)}\right)^T, \tag{3}$$

where $\widetilde{\Gamma}(\boldsymbol{y}^{(j)}) = \Gamma(\boldsymbol{y}^{(j)}) - \bar{\Gamma}$ and $\bar{\Gamma} = (1/N_t) \sum_{j=1}^{N_t} \Gamma(\boldsymbol{y}^{(j)})$. Eigenvectors of $C_{\mathcal{F}}$ can give a new basis to represent the mapped data, and the eigenvectors associated with dominate eigenvalues can provide an effective reduced dimensional representation for them.

However, the mapping Γ in practice is typically defined implicitly through kernel functions, and the eigenvectors of $C_{\mathcal{F}}$ are always replaced by eigenvectors of some centred kernel matrices. A kernel function in this setting is a mapping from $\mathbb{R}^{N_h} \times \mathbb{R}^{N_h}$ to $\mathbb{R}$, which is denoted by $\boldsymbol{k}(\cdot, \cdot)$. The kernel matrix associated with $\boldsymbol{k}(\cdot, \cdot)$ is denoted by $\boldsymbol{K} \in \mathbb{R}^{N_h \times N_h}$, of which each entry is defined as

$$\boldsymbol{K}_{ij} = \boldsymbol{k}(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)}) \quad \text{for} \quad i, j = 1, 2, \ldots, N_t.$$

A standard choice of the kernel function is the Gaussian kernel

$$\boldsymbol{k}(\boldsymbol{y}^{(i)}, \boldsymbol{y}^{(j)}) = \exp\left(-\frac{\left\|\boldsymbol{y}^{(i)} - \boldsymbol{y}^{(j)}\right\|_2^2}{2\sigma_g^2}\right),$$

where σ_g is the bandwidth parameter. The centred kernel matrix is defined as

$$\widetilde{\boldsymbol{K}} := \left(\boldsymbol{K} - \boldsymbol{1}_{\frac{1}{N_t}} \boldsymbol{K} - \boldsymbol{K} \boldsymbol{1}_{\frac{1}{N_t}} + \boldsymbol{1}_{\frac{1}{N_t}} \boldsymbol{K} \boldsymbol{1}_{\frac{1}{N_t}}\right),$$

where $\boldsymbol{1}_{\frac{1}{N_t}}$ denotes the matrix with all entries equaling to $1/N_t$.

Let $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{N_t}] \in \mathbb{R}^{N_t \times N_t}$ collect the eigenvectors of $\widetilde{\boldsymbol{K}}$ associated with eigenvalues $\lambda_1 > \lambda_2 \ldots > \lambda_{N_t}$. Basis functions of the mapped data in $\mathcal{F}$ are defined as $\omega_e := \sum_{j=1}^{N_t} \widetilde{\alpha}_{je} \widetilde{\Gamma}(\boldsymbol{y}^{(j)})$, where $\widetilde{\alpha}_{je} = \alpha_{je}/\sqrt{\lambda_e}$ and α_{je} is the j-th component of $\boldsymbol{\alpha}_e$, for $e = 1, \ldots, N_t$. A mapped training point $\widetilde{\Gamma}(\boldsymbol{y}^{(j)})$ for $j = 1, \ldots, N_t$ can be represented as $\widetilde{\Gamma}(\boldsymbol{y}^{(j)}) = \sum_{e=1}^{N_t} \tilde{\gamma}_e(\boldsymbol{y}^{(j)}) \omega_e$, where each coefficient $\tilde{\gamma}_e(\boldsymbol{y}^{(j)})$ is computed through

$$\tilde{\gamma}_e\left(\boldsymbol{y}^{(j)}\right) = \sum_{i=1}^{N_t} \widetilde{\alpha}_{ie} \widetilde{\boldsymbol{K}}_{ij}. \tag{4}$$

To result in dimension reduction, the first N_r dominant eigenvectors of $\widetilde{\boldsymbol{K}}$ are selected as the the principal components, with the criterion $(\sum_{e=1}^{N_r} \lambda_e)/(\sum_{e=1}^{N_t} \lambda_e) > tol_{\text{PCA}}$ where tol_{PCA} is a given tolerance. The basis functions associate with the principal components are then ω_e, $e = 1, \ldots, N_r$, and each mapped training data point can be approximated as $\widetilde{\Gamma}(\boldsymbol{y}^{(j)}) \approx \sum_{e=1}^{N_r} \tilde{\gamma}_e(\boldsymbol{y}^{(j)}) \omega_e$.

It can be seen that the overall procedure of kPCA defines a mapping from the output manifold $\mathcal{M}$ to the reduced feature space $\text{span}\{\omega_1, \ldots, \omega_{N_r}\}$. We denote this mapping as $\kappa(\boldsymbol{y}) = \sum_{e=1}^{N_r} \tilde{\gamma}_e(\boldsymbol{y}) \omega_e$, where each coefficient $\tilde{\gamma}_e(\boldsymbol{y})$ is obtained from (4). The basis $\{\omega_e\}_{e=1}^{N_r}$ discussed above depends on the mapping $\widetilde{\Gamma}$ which are defined implicitly. Collecting these coefficients, a reduced output vector is denoted by $\tilde{\boldsymbol{\gamma}}(\boldsymbol{y}) := [\tilde{\gamma}_1(\boldsymbol{y}), \ldots, \tilde{\gamma}_{N_r}(\boldsymbol{y})]^T \in \mathbb{R}^{N_r}$ for any $\boldsymbol{y} \in \mathcal{M}$. The manifold consisting of all reduced output vectors is denoted by $\mathcal{M}_r$. We next denote $\boldsymbol{\gamma}(\boldsymbol{\xi}) := \tilde{\boldsymbol{\gamma}}(\boldsymbol{y}) \in \mathcal{M}_r \subset \mathbb{R}^{N_r}$. In summary, each training data point $\boldsymbol{y}^{(j)} = \chi(\boldsymbol{\xi}^{(j)}) \in \mathcal{M}$ is mapped to $\boldsymbol{\gamma}(\boldsymbol{\xi}^{(j)}) := [\gamma_1(\boldsymbol{\xi}^{(j)}), \ldots, \gamma_{N_r}(\boldsymbol{\xi}^{(j)})]^T \in \mathcal{M}_r$ for $j = 1, \ldots, N_t$. We next construct stochastic collocation surrogates for each component of $\boldsymbol{\gamma}(\boldsymbol{\xi})$.

4

2.2. Stochastic collocation

For each $\gamma_e(\boldsymbol{\xi})$, $e = 1, \ldots, N_r$, a truncated generalized polynomial chaos (gPC) approximation [1, 2] can be written as

$$\gamma_e(\boldsymbol{\xi}) \approx \gamma_e^{\mathrm{gPC}}(\boldsymbol{\xi}) := \sum_{\|\boldsymbol{i}\|_1=0}^{p} c_{e\boldsymbol{i}} \Phi_{\boldsymbol{i}}(\boldsymbol{\xi}), \tag{5}$$

where $\boldsymbol{i} = [i_1, i_2, \ldots, i_d]^T \in \mathbb{N}^d$ is a multi-index, $\|\boldsymbol{i}\|_1 = i_1 + i_2 + \cdots + i_d$, and p is a given oder for truncation. Denoting the set of the multi-indices by $\Upsilon := \{\boldsymbol{i} | \boldsymbol{i} \in \mathbb{N}^d \text{ and } \|\boldsymbol{i}\|_1 = 0, \ldots, p\}$, which implies that the number of basis function is $|\Upsilon| = (p + d)!/(p!d!)$. The basis functions $\{\Phi_{\boldsymbol{i}}(\boldsymbol{\xi}) | \boldsymbol{i} \in \Upsilon\}$ are orthogonal polynomials with respect to the density function $\pi(\boldsymbol{\xi})$

$$\langle \Phi_{\boldsymbol{i}}(\boldsymbol{\xi}), \Phi_{\boldsymbol{i}'}(\boldsymbol{\xi}) \rangle = \int_{I^d} \Phi_{\boldsymbol{i}}(\boldsymbol{\xi}) \Phi_{\boldsymbol{i}'}(\boldsymbol{\xi}) \pi(\boldsymbol{\xi}) d\boldsymbol{\xi} = \delta_{\boldsymbol{i},\boldsymbol{i}'},$$

where δ denotes the Kronecker delta function, i.e., $\delta_{\boldsymbol{i},\boldsymbol{i}'} = 1$ if $\boldsymbol{i}$ is the same as $\boldsymbol{i}'$ and $\delta_{\boldsymbol{i},\boldsymbol{i}'} = 0$ otherwise. Each basis function $\Phi_{\boldsymbol{i}}(\boldsymbol{\xi})$ can be expressed as the product of a set of univariate orthogonal polynomials, $\Phi_{\boldsymbol{i}}(\boldsymbol{\xi}) = \prod_{k=1}^{d} \phi_{i_k}(\xi_k)$, with each univariate orthogonal polynomial defined through a three term recurrence [23],

$$\phi_{j+1}(\xi) = (\xi - \zeta_j)\phi_j(\xi) - \tau_j \phi_{j-1}(\xi), \quad j = 1, 2, \ldots, p - 1,$$

$$\phi_0(\xi) = 0, \quad \phi_1(\xi) = 1,$$

where $\zeta_j = \int_{-1}^{1} \xi \pi_\xi(\xi) \phi_j^2(\xi) d\xi / \int_{-1}^{1} \pi_\xi(\xi) \phi_j^2(\xi) d\xi$, $\tau_j = \int_{-1}^{1} \pi_\xi(\xi) \phi_j^2(\xi) d\xi / \int_{-1}^{1} \pi_\xi(\xi) \phi_{j-1}^2(\xi) d\xi$, and $\pi_\xi(\xi)$ is the marginal density function of ξ (ξ denotes a component of $\boldsymbol{\xi}$).

According to orthogonality of the gPC basis functions, the coefficients in (5) can be computed through

$$c_{e\boldsymbol{i}} = \int_{I^d} \gamma_e(\boldsymbol{\xi}) \Phi_{\boldsymbol{i}}(\boldsymbol{\xi}) \pi(\boldsymbol{\xi}) d\boldsymbol{\xi}. \tag{6}$$

This integral can be computed through quadrature rules, and following [13] we focus on the tensor style quadrature. Let $\{\xi^{(j)}, w^{(j)}\}_{j=1}^{n}$ denote n quadrature nodes and weights on the interval $[-1, 1]$. The quadrature form of (6) is

$$c_{e\boldsymbol{i}} = \sum_{1 \le j_1, \ldots, j_d \le n} \gamma_e(\boldsymbol{\xi}_{j_1 \ldots j_d}) \Phi_{\boldsymbol{i}}(\boldsymbol{\xi}_{j_1 \ldots j_d}) w_{j_1 \ldots j_d}, \tag{7}$$

where

$$\boldsymbol{\xi}_{j_1 \ldots j_d} = \left[\xi_1^{(j_1)}, \xi_2^{(j_2)}, \ldots, \xi_d^{(j_d)} \right]^T, \tag{8}$$

$$w_{j_1 \ldots j_d} = w^{(j_1)} w^{(j_2)} \cdots w^{(j_d)}, \tag{9}$$

for $1 \le j_1, \ldots, j_d \le n$ are the nodes and the weights spanned by the tensor product of the one-dimensional quadrature rule.

Following [13], the quadrature form (7) can be formulated as a tensor inner product as follows. For each $e = 1, \ldots, N_r$, the values $\gamma_e(\boldsymbol{\xi}_{j_1 \ldots j_d})$ for $1 \le j_1, \ldots, j_d \le n$ form a d-th order *data tensor* $\mathbf{X}_e \in \mathbb{R}^{n \times \cdots \times n}$, of which each entry is

$$\mathbf{X}_e(j_1, \ldots, j_d) = \gamma_e(\boldsymbol{\xi}_{j_1 \ldots j_d}). \tag{10}$$

For each multi index $\boldsymbol{i}$ with $\|\boldsymbol{i}\|_1 \leq p$, the values $\Phi_{\boldsymbol{i}}(\boldsymbol{\xi}_{j_1 \dots j_d}) w_{j_1 \dots j_d}$ for $1 \leq j_1, \dots, j_d \leq n$ form a d-th order *weight tensor* $\mathbf{W}_{\boldsymbol{i}} \in \mathbb{R}^{n \times \dots \times n}$ with

$$\mathbf{W}_{\boldsymbol{i}}(j_1, \dots, j_d) = \Phi_{\boldsymbol{i}}(\boldsymbol{\xi}_{j_1 \dots j_d}) w_{j_1 \dots j_d}. \tag{11}$$

Defining

$$\hat{\boldsymbol{w}}_k^{(i_k)} = \left[\phi_{i_k}\left(\xi^{(1)}\right) w^{(1)}, \phi_{i_k}\left(\xi^{(2)}\right) w^{(2)}, \dots, \phi_{i_k}\left(\xi^{(n)}\right) w^{(n)} \right]^T \in \mathbb{R}^n, \quad \text{for} \quad k = 1, \dots, d, \tag{12}$$

each entry of $\mathbf{W}_{\boldsymbol{i}}$ can be written as

$$\mathbf{W}_{\boldsymbol{i}}(j_1, j_2, \dots, j_d) = \hat{\boldsymbol{w}}_1^{(i_1)}(j_1) \hat{\boldsymbol{w}}_2^{(i_2)}(j_2) \cdots \hat{\boldsymbol{w}}_d^{(i_d)}(j_d) \quad \text{for all} \quad 1 \leq j_k \leq n, \ k = 1, \dots, d,$$

and $\mathbf{W}_{\boldsymbol{i}}$ can be expressed as

$$\mathbf{W}_{\boldsymbol{i}} = \hat{\boldsymbol{w}}_1^{(i_1)} \circ \hat{\boldsymbol{w}}_2^{(i_2)} \circ \cdots \circ \hat{\boldsymbol{w}}_d^{(i_d)} \quad \text{with} \quad \boldsymbol{i} = [i_1, \dots, i_d]^T, \tag{13}$$

where "$\circ$" is the vector outer product. With the notation above, the coefficient $c_{e\boldsymbol{i}}$ in (7) can be rewritten as the tensor inner product

$$c_{e\boldsymbol{i}} = \langle \mathbf{X}_e, \mathbf{W}_{\boldsymbol{i}} \rangle, \tag{14}$$

where the tensor inner product [24, 25] is defined as,

$$\langle \mathbf{X}_e, \mathbf{W}_{\boldsymbol{i}} \rangle = \sum_{j_1}^n \sum_{j_2}^n \cdots \sum_{j_d}^n \mathbf{X}_e(j_1, j_2, \dots, j_d) \mathbf{W}_{\boldsymbol{i}}(j_1, j_2, \dots, j_d). \tag{15}$$

The tensor norm induced by this inner product is denoted by $\|\cdot\| = \langle \cdot, \cdot \rangle^{1/2}$. Details of tensor decomposition and recovery are discussed in section 3 and section 4.

2.3. Inverse mapping

After the gPC approximation (5) for each γ_e, $e = 1, \dots, N_r$, is constructed through the above collocation procedure, the reduced output $\gamma(\boldsymbol{\xi}) = \tilde{\boldsymbol{\gamma}}(\boldsymbol{y}) = \tilde{\boldsymbol{\gamma}}(\chi(\boldsymbol{\xi})) \in \mathcal{M}_r$ for an arbitrary realization of $\boldsymbol{\xi}$ can be cheaply estimated through this gPC surrogate. However, our goal is to quantify the uncertainties in the output $\boldsymbol{y} = \chi(\boldsymbol{\xi}) \in \mathcal{M}$, which requires an inverse mapping κ^{-1} from the reduced output manifold $\mathcal{M}_r$ to the original output manifold $\mathcal{M}$. Following [21, 15], an inverse mapping can be obtained through an interpolation of neighbouring points in the training data set $\{\boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(N_t)}\}$. That is, the Euclid distance between an arbitrary output $\boldsymbol{y} \in \mathcal{M}$ and each training point $\boldsymbol{y}^{(j)}$ (for $j = 1, \dots, N_t$) is first computed through

$$d_j = \sqrt{-2\sigma_g^2 \log\left(1 - 0.5\hat{d}_j^2\right)},$$

where $\hat{d}_j = 1 + \tilde{\boldsymbol{\gamma}}(\boldsymbol{y})^T K \tilde{\boldsymbol{\gamma}}(\boldsymbol{y}) - 2\tilde{\boldsymbol{\gamma}}(\boldsymbol{y})^T \mathbf{k}_{\boldsymbol{y}^{(j)}}$ are computed through the kernel function, $\mathbf{k}_{\boldsymbol{y}^{(j)}} = [k(\boldsymbol{y}^{(j)}, \boldsymbol{y}^{(1)}), \dots, k(\boldsymbol{y}^{(j)}, \boldsymbol{y}^{(N_t)})]^T$ and $k(\cdot, \cdot)$ is the given kernel function. The distances $\{d_1, \dots, d_{N_t}\}$ are sorted next. Given a positive integer N_n, the indices with the smallest N_n distances are collected in a set $\mathcal{J} \subset \{1, \dots, N_t\}$, i.e., $d_j \leq d_i$ for any $j, i = 1, \dots, N_t$ with $j \in \mathcal{J}$ but $i \notin \mathcal{J}$. After that, $\boldsymbol{y}$ can be approximated as

$$\boldsymbol{y} \approx \sum_{j \in \mathcal{J}} \frac{d_j^{-1}}{\sum_{j \in \mathcal{J}} d_j^{-1}} \boldsymbol{y}^{(j)}. \tag{16}$$

6

3. Tensor recovery based quadrature

It is clear that the main computational cost of the above collocation procedure based on manifold learning comes from generating the gPC expansion (5) for each kPCA mode $e = 1, \ldots, N_r$, where evaluating each collocation coefficient requires computing a tensor inner product (14). When the input parameter $\boldsymbol{\xi}$ is high-dimensional (d is large), each data tensor $\mathbf{X}_e \in \mathbb{R}^{n \times \cdots \times n}$ is large (with n^d entries). Evaluating each entry of $\mathbf{X}_e$ requires computing a snapshot (see (10)), and it is therefore expensive to form these data tensors through computing snapshots for all entries. As an alternative, tensor recovery methods provide efficient estimates of tensors using a small number of exact entries. For forward UQ problems with a single output, when tensor ranks are given, a tensor recovery based collocation approach is developed in [13], which can be applied to construct the gPC approximation for each kPCA component (5). We here review this tensor recovery based collocation approach and provide new detailed computational cost assessments. Since computation procedures for generating the gPC surrogates for each $\gamma_e(\boldsymbol{\xi})$, $e = 1, \ldots, N_r$, are identical, we generically denote the data tensor $\mathbf{X}_e$ defined in (10) as $\mathbf{X}_{\text{exact}}$ in this section (i.e., the subscript e is temporally ignored).

3.1. Canonical polyadic (CP) decomposition

Following the presentation in [25], the CP decomposition is reviewed as follows. For a d-th order tensor $\mathbf{X} \in \mathbb{R}^{n \times \cdots \times n}$, its CP decomposition is expressed as

$$\mathbf{X} = \sum_{r=1}^{R} \boldsymbol{v}_1^{(r)} \circ \boldsymbol{v}_2^{(r)} \circ \cdots \circ \boldsymbol{v}_d^{(r)} \tag{17}$$

where $\boldsymbol{v}_k^{(r)} \in \mathbb{R}^n$ for $k = 1, \ldots, d$, R is the CP rank of $\mathbf{X}$, and "$\circ$" is the vector outer product. The CP rank is defined as

$$R := \text{rank}(\mathbf{X}) := \min \left\{ R' \ \middle| \ \mathbf{X} = \sum_{r=1}^{R'} \boldsymbol{v}_1^{(r)} \circ \boldsymbol{v}_2^{(r)} \circ \cdots \circ \boldsymbol{v}_d^{(r)} \right\}.$$

Figure 1 shows a third-order tensor with its CP decomposition. For each $r = 1, \ldots, R$, $\boldsymbol{v}_1^{(r)} \circ \boldsymbol{v}_2^{(r)} \circ \cdots \circ \boldsymbol{v}_d^{(r)}$ in (17) is
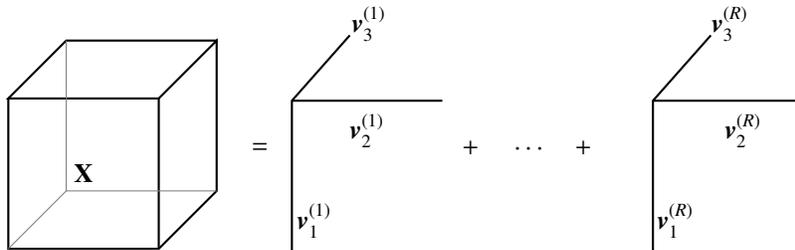
Figure 1: CP decomposition of a third-order tensor with rank R.

called a rank-one component. For each $k = 1, \ldots, d$, the matrix $A_k = [\boldsymbol{v}_k^{(1)}, \boldsymbol{v}_k^{(2)}, \ldots, \boldsymbol{v}_k^{(R)}] \in \mathbb{R}^{n \times R}$ is called the kth-order factor matrix. With these factor matrices, the CP decomposition (17) can be rewritten as

$$\mathbf{X} = [[A_1, A_2, \ldots, A_d]]. \tag{18}$$

From (13), it is clear that each weight tensor is a rank-one tensor, and we here generically denote it as $\mathbf{W} := w_1 \circ w_2 \circ \cdots \circ w_d \in \mathbb{R}^{n \times \cdots \times n}$. Following [25], the inner product of $\mathbf{X}$, $\mathbf{W}$ (for computing (14)) can be efficiently computed as

$$
\begin{aligned}
\langle \mathbf{X}, \mathbf{W} \rangle &= \langle [[A_1, A_2, \ldots, A_d]], w_1 \circ w_2 \circ \cdots \circ w_d \rangle \\
&= \sum_{r=1}^{R} \langle v_1^{(r)} \circ v_2^{(r)} \circ \cdots \circ v_d^{(r)}, w_1 \circ w_2 \circ \cdots \circ w_d \rangle \\
&= \sum_{r=1}^{R} \sum_{j_1, j_2, \ldots, j_d} v_1^{(r)}(j_1) v_2^{(r)}(j_2) \cdots v_d^{(r)}(j_d) w_1(j_1) w_2(j_2) \cdots w_d(j_d) \\
&= \sum_{r=1}^{R} \left(\left(\sum_{j_1} v_1^{(r)}(j_1) w_1(j_1) \right) \left(\sum_{j_2} v_2^{(r)}(j_2) w_2(j_2) \right) \cdots \left(\sum_{j_d} v_d^{(r)}(j_d) w_d(j_d) \right) \right) \\
&= \sum_{r=1}^{R} \left(\prod_{k=1}^{d} \langle v_k^{(r)}, w_k \rangle \right).
\end{aligned}
\tag{19}
$$

The cost for computing the tensor inner product $\langle \mathbf{X}, \mathbf{W} \rangle$ through (19) is $O(dnR)$, while the cost is $O(n^d)$ if the inner product is computed directly through its definition (15).

3.2. Missing data tensor recovery

Denoting the full index set for a d-th order tensor in $\mathbb{R}^{n \times \cdots \times n}$ as $\Theta_{\text{full}} := \{ [j_1, j_2, \ldots, j_d]^T \in \mathbb{N}^d \mid j_k = 1, \ldots, n \text{ for } k = 1, \ldots, d \}$, $\Theta \subset \Theta_{\text{full}}$ denotes an observation index set, which consists of $|\Theta| \ll d^n$ indices uniformly sampled from Θ_{full}. To numbering the elements in Θ, the following sort operator is introduced.

Definition 1 (Sort operator). *For a given finite set $\Theta \subset \mathbb{N}^d$, we first sort its elements in alphabetical order: for any two different indices $\hat{j} = [\hat{j}_1, \ldots, \hat{j}_d]^T$ and $\tilde{j} = [\tilde{j}_1, \ldots, \tilde{j}_d]^T$ belonging in Θ, $\hat{j}$ is ordered before $\tilde{j}$ if for the smallest number k such that $\hat{j}_k \neq \tilde{j}_k$, we have $\hat{j}_k < \tilde{j}_k$. Then for any $j \in \Theta$, $s(\Theta, j) \in \{1, \ldots, |\Theta|\}$ is defined to be the position of j in the sorted array.*

A projection operator that takes tensor values over the observed indices are denoted by $\mathbb{P}_\Theta$: for any d-th order tensor $\mathbf{X} \in \mathbb{R}^{n \times \cdots \times n}$, $\mathbb{P}_\Theta(\mathbf{X}) := [p_1, \ldots, p_{|\Theta|}]^T \in \mathbb{R}^{|\Theta|}$ where $p_{s(\Theta, j)} = \mathbf{X}(j_1, \ldots, j_d)$ for all $j \in \Theta$. Tensor recovery here is to find an approximation of the data tensor $\mathbf{X}_{\text{exact}}$ based on the entries over the observation indices, i.e., $\mathbb{P}_\Theta(\mathbf{X}_{\text{exact}})$. Since it is assumed that $|\Theta| \ll d^n$, the cost for generating $\mathbb{P}_\Theta(\mathbf{X}_{\text{exact}})$ is small compared with the cost for generating the whole $\mathbf{X}_{\text{exact}}$. When the CP rank of $\mathbf{X}_{\text{exact}}$ (denoted by R) is given, Acar et al. [17] formulate the tensor recovery problem as the following optimization problem

$$
\begin{aligned}
\min_{\mathbf{X}} \quad & \frac{1}{2} \|\mathbb{P}_\Theta(\mathbf{X}) - \mathbb{P}_\Theta(\mathbf{X}_{\text{exact}})\|_2^2 \\
\text{s.t.} \quad & \mathbf{X} = \sum_{r=1}^{R} v_1^{(r)} \circ v_2^{(r)} \circ \cdots \circ v_d^{(r)},
\end{aligned}
\tag{20}
$$

It is clear that evaluating $\mathbb{P}_\Theta(\mathbf{X})$ requires $O(|\Theta| dR)$ flops.

To take the sparsity of gPC coefficients (5) into account (Cf. [6, 8] for sparse gPC approximations), a l_1 regularized version of (20) is formulated as follows [13]:

$$
\min_{A_1, A_2, \ldots, A_d} J(A_1, A_2, \ldots, A_d) = \frac{1}{2} \|\mathbb{P}_\Theta([[A_1, A_2, \ldots, A_d]]) - \mathbb{P}_\Theta(\mathbf{X}_{\text{exact}})\|_2^2 + \beta \sum_{\|i\|_1=0}^{p} |\langle [[A_1, A_2, \ldots, A_d]], \mathbf{W}_i \rangle|, \tag{21}
$$

8

where β is a regularization parameter, p is a given gPC order, and A_k for $k = 1, \ldots, d$ are CP factor matrices of $\mathbf{X}$ (see (18)). To solve (21), the alternative minimization iterative method can be applied [13]. Letting $A_k^{(q)}$ for $k = 1, \ldots, d$ be the CP factor matrices at q-th iteration step ($q \geq 0$ is an integer), each CP factor matrix $A_k^{(q+1)}$ at iteration step $q + 1$ is obtained through

$$A_k^{(q+1)} = \arg \min_{A_k} J(A_1^{(q+1)}, \ldots, A_{k-1}^{(q+1)}, A_k, A_{k+1}^{(q)}, \ldots, A_d^{(q)}), \tag{22}$$

which leads to a generalized lasso problem and is discussed next.

3.3. A generalized lasso problem

Let vec(A) denote the vector form of a given matrix A (as implemented in the MATLAB function **reshape**). Following the procedures discussed in [13], (22) can be written as a generalized lasso problem

$$\text{vec}\left(A_k^{(q+1)}\right) = \arg \min_s \frac{1}{2} \|Bs - b\|_2^2 + \beta \|Fs\|_1, \tag{23}$$

where $B := \mathcal{B}_{\Theta,k}([[A_1^{(q+1)}, \ldots, A_{k-1}^{(q+1)}, A_k, A_{k+1}^{(q)}, \ldots, A_d^{(q)}]])$, $F := \mathcal{F}_{\Upsilon,k}([[A_1^{(q+1)}, \ldots, A_{k-1}^{(q+1)}, A_k, A_{k+1}^{(q)}, \ldots, A_d^{(q)}]])$ with $\mathcal{B}_{\Theta,k}(\cdot)$ and $\mathcal{F}_{\Upsilon,k}(\cdot)$ defined as follows (in Definition 2 and Definition 3 respectively), and $b = \mathbb{P}_{\Theta}(\mathbf{X}_{\text{exact}}) \in \mathbb{R}^{|\Theta|}$.

Definition 2. *For a given observation index set $\Theta \subset \mathbb{N}^d$ and $k \in \{1, \ldots, d\}$, the operator $\mathcal{B}_{\Theta,k}$ defines a mapping: for a d-th order tensor $[[A_1, A_2, \ldots, A_d]] \in \mathbb{R}^{n \times \cdots \times n}$ with rank R, entries of $B := \mathcal{B}_{\Theta,k}([[A_1, A_2, \ldots, A_d]]) \in \mathbb{R}^{|\Theta| \times nR}$ are zero except*

$$B(s(\Theta, j), (r-1)n + j_k) = \prod_{k'=1}^{k-1} A_{k'}(j_{k'}, r) \prod_{k'=k+1}^{d} A_{k'}(j_{k'}, r), \text{ for all } j = [j_1, \ldots, j_d] \in \Theta \text{ and } r = 1, \ldots, R, \tag{24}$$

where $s(\Theta, \cdot)$ is the sort operator defined in Definition 1.

Proposition 1. *Let Θ be an observation index set, and $[[A_1, A_2, \ldots, A_d]] \in \mathbb{R}^{n \times \cdots \times n}$ be a d-th order tensor with rank R. Letting $A_{i\delta} := [A_i, \delta a] \in \mathbb{R}^{n \times (R+1)}$ for $i = 1, 2, \ldots, d$, for any $k \in \{1, \ldots, d\}$, we have that*

$$\mathcal{B}_{\Theta,k}([[A_{1\delta}, A_{2\delta}, \ldots, A_{d\delta}]]) = [\mathcal{B}_{\Theta,k}([[A_1, A_2, \ldots, A_d]]), \mathcal{B}_{\Theta,k}([[\delta a, \delta a, \ldots, \delta a]])]. \tag{25}$$

Proof. This proposition is straightforward since the matrix B is constructed by (24). $\qquad\square$

Definition 3. *Denoting the set of the multi-indices in (5) as $\Upsilon := \{i | i \in \mathbb{N}^d \text{ and } |i| = 0, \ldots, p\}$, where p is a given gPC order. The operator $\mathcal{F}_{\Upsilon,k}$ defines a mapping: for a d-th order tensor $[[A_1, A_2, \ldots, A_d]] \in \mathbb{R}^{n \times \cdots \times n}$ with rank R and $k \in \{1, \ldots, d\}$, entries of $F := \mathcal{F}_{\Upsilon,k}([[A_1, A_2, \ldots, A_d]]) \in \mathbb{R}^{|\Upsilon| \times nR}$ are specified as*

$$F(s(\Upsilon, i), (r-1)n + m_k) = \hat{w}_k^{(i_k)}(m_k) \prod_{k'=1}^{k-1} \left\langle A_{k'}(:, r), \hat{w}_{k'}^{(i_{k'})} \right\rangle \prod_{k'=k+1}^{d} \left\langle A_{k'}(:, r), \hat{w}_{k'}^{(i_{k'})} \right\rangle, \tag{26}$$

for all $i = [i_1, i_2, \ldots, i_d]^T \in \Upsilon$, $m_k = 1, \ldots, n$ and $r = 1, \ldots, R$, where $s(\Upsilon, \cdot)$ is the sort operator defined in Definition 1 and each $\hat{w}_k^{(i_k)}$ is defined in (12).

9

The operation numbers to construct $\boldsymbol{B}$ and $\boldsymbol{F}$ are $O(|\Theta|Rd)$ and $O(|\Upsilon|Rn^2d)$ respectively. The generalized lasso problem (23) can be solved by alternating direction method of multipliers (ADMM) [26] as follows. First, (23) is rewritten as

$$
\min_{s,t} \quad \frac{1}{2}\|\boldsymbol{B}\boldsymbol{s} - \boldsymbol{b}\|_2^2 + \beta\|\boldsymbol{t}\|_1
$$
$$
\text{s.t.} \quad \boldsymbol{F}\boldsymbol{s} = \boldsymbol{t}.
$$
(27)

The augmented Lagrangian function of (27) is given by

$$
L(\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{z}, \varrho) = \frac{1}{2}\|\boldsymbol{B}\boldsymbol{s} - \boldsymbol{b}\|_2^2 + \beta\|\boldsymbol{t}\|_1 + \langle \boldsymbol{z}, \boldsymbol{F}\boldsymbol{s} - \boldsymbol{t}\rangle + \frac{\varrho}{2}\|\boldsymbol{F}\boldsymbol{s} - \boldsymbol{t}\|_2^2 ,
$$

where $\boldsymbol{z} \in \mathbb{R}^{|\Upsilon|}$ is the Lagrange multiplier, and $\varrho > 0$ is an augmented Lagrange multiplier. Following [27], the optimization problem (27) can be solved as

$$
\boldsymbol{s}^{(i+1)} = \arg\min_{s} L(\boldsymbol{s}, \boldsymbol{t}^{(i)}, \boldsymbol{z}^{(i)}, \varrho^{(i)})
$$
$$
\boldsymbol{t}^{(i+1)} = \arg\min_{t} L(\boldsymbol{s}^{(i+1)}, \boldsymbol{t}, \boldsymbol{z}^{(i)}, \varrho^{(i)})
$$
$$
\boldsymbol{z}^{(i+1)} = \boldsymbol{z}^{(i)} + \varrho^{(i)}(\boldsymbol{F}\boldsymbol{s}^{(i+1)} - \boldsymbol{t}^{(i+1)}).
$$

Following [26], details of the ADMM algorithm for (27) are summarized Algorithm 1, and the soft-thresholding operator $\mathbb{S}_{\frac{\beta}{\varrho}}$ on line 4 of Algorithm 1 is defined as

$$
\mathbb{S}_{\frac{\beta}{\varrho}}(h) = \begin{cases} h - \frac{\beta}{\varrho} & \text{when} \quad h \geq \frac{\beta}{\varrho}, \\ 0 & \text{when} \quad |h| < \frac{\beta}{\varrho}, \\ h + \frac{\beta}{\varrho} & \text{when} \quad h \leq -\frac{\beta}{\varrho}. \end{cases}
$$

The cost of using Algorithm 1 to solve (27) is analyzed as follows:

- updating $\boldsymbol{s}^{(k+1)}$ line 3 of Algorithm 1 requires a matrix inversion and matrix-vector products, of which the total cost is $O(n^3R^3 + n^2R^2(|\Theta| + |\Upsilon|))$.

- updating the soft-thresholding operator and $\boldsymbol{z}^{(k+1)}$ requires $O(|\Upsilon|nR)$ operations.

Therefore, the total cost of Algorithm 1 is $C_{Alg1} := O(n^3R^3 + n^2R^2(|\Theta| + |\Upsilon|)) + O(|\Upsilon|Rn^2d + |\Theta|Rd)$.

The stopping criterion for the overall optimization problem (21) is specified through three parts in [13]: the relative changes of factor matrices, objective function values, and gPC coefficients. The relative change of factor matrices between iteration step q and $q + 1$ is defined as $\epsilon_{\text{factor}} := (\sum_{k=1}^{d}\|A_k^{(q+1)} - A_k^{(q)}\|_F^2)^{1/2}/(\sum_{k=1}^{d}\|A_k^{(q)}\|_F^2)^{1/2}$ where $\|\cdot\|_F$ denotes the matrix Frobenius norm. The complexity of computing ϵ_{factor} is $O(dnR)$. Similarly, the relative changes of objective function values and gPC coefficients are defined as $\epsilon_J := |J^{(q+1)} - J^{(q)}|/|J^{(q)}|$ and $\epsilon_c := \|c^{(q+1)} - c^{(q)}\|_1/\|c^{(q)}\|_1$ respectively, where $\|\cdot\|_1$ denotes the vector l_1 norm and $c^{(q)}$ collects the collocation coefficients (14) obtained with $[[A_1^{(q)}, \ldots, A_d^{(q)}]]$. Since evaluating the objective function of (21) includes computing the projection $\mathbb{P}_\Theta$ and the tensor inner products, the cost of computing ϵ_J and ϵ_c are $O(|\Theta|dR + |\Upsilon|ndR)$. For a given tolerance δ, the optimization iteration for (21) terminates if $\epsilon_{\text{factor}} < \delta$, $\epsilon_J < \delta$ and $\epsilon_c < \delta$. The details for solving (21) is summarized Algorithm 2, which is proposed in [13]. The total cost of Algorithm 2 is $C_{Alg2} := dC_{Alg1} + O((n + |\Theta| + |\Upsilon|n)dR)$.

Algorithm 1 ADMM for generalized lasso [26]

Input: $B, F, b, \beta, \nu \geq 1$ (for augment Lagrange multiplier)

1: Initialize $\varrho^{(0)}, s^{(0)}, t^{(0)} = Fs^{(0)}$, and $z^{(0)}, i = 0$

2: **while** not converged **do**

3:      $s^{(i+1)} = (B^T B + \varrho F^T F)^\dagger (B^T b + \varrho F^T t^{(i)} - F^T z^{(i)})$

4:      $t^{(i+1)} = \mathbb{S}_{\frac{\beta}{\varrho}}(Fs^{(i+1)} + \frac{1}{\varrho} z^{(i)})$ (element-wise)

5:      $z^{(i+1)} = z^{(i)} + \varrho(Fs^{(i+1)} - t^{(i+1)})$

6:      $\varrho^{(i+1)} = \varrho^{(i)} \nu$

7:      $i = i + 1$

8: **end while**

Output: $s^* = s^{(i)}$ and A (the matrix form of s^*).

Algorithm 2 Fixed-rank tensor recovery [13]

Input: CP rank R, initial rank R factor matrices $A_k^{(0)}$ for $k = 1, \ldots, d$, Θ, Υ and $\mathbb{P}_\Theta(\mathbf{X}_{\text{exact}})$.

1: Let $q = 0$ and $b = \mathbb{P}_\Theta(\mathbf{X}_{\text{exact}})$.

2: Initialize $\epsilon_{\text{factor}} \geq \delta, \epsilon_J \geq \delta, \epsilon_c \geq \delta$.

3: **while** $\epsilon_{\text{factor}} \geq \delta, \epsilon_J \geq \delta$ or $\epsilon_c \geq \delta$ **do**

4:      $q = q + 1$.

5:      **for** $k = 1 : d$ **do**

6:          $B := \mathcal{B}_{\Theta,k}([[A_1^{(q+1)}, \ldots, A_{k-1}^{(q+1)}, A_k, A_{k+1}^{(q)}, \ldots, A_d^{(q)}]])$.

7:          $F := \mathcal{F}_{\Upsilon,k}([[A_1^{(q+1)}, \ldots, A_{k-1}^{(q+1)}, A_k, A_{k+1}^{(q)}, \ldots, A_d^{(q)}]])$.

8:          Obtain factor matrix $A_k^{(q+1)}$ by Algorithm 1.

9:      **end for**

10:     Compute $\epsilon_{\text{factor}}, \epsilon_J, \epsilon_c$.

11: **end while**

Output: CP factor matices $A_k = A_k^{(q)}$ for $k = 1, 2, \cdots, d$ and the recovered tensor $\mathbf{X} = [[A_1, \ldots, A_d]]$.

---

4. Rank adaptive tensor recovery for stochastic collocation

Our goal is to perform uncertainty propagation from a high-dimensional random input vector $\boldsymbol{\xi}$ to the snapshot $\boldsymbol{y} = [u(x^{(1)}, \boldsymbol{\xi}), \ldots, u(x^{(N_h)}, \boldsymbol{\xi})]^T \in \mathbb{R}^{N_h}$ which is also high-dimensional. For this purpose, we develop a novel rank adaptive tensor recovery collocation (RATR-collocation) approach in this section. We first present our new general rank adaptive tensor recovery (RATR) algorithm for a general tensor, and then analyze its stability. After that, we present our main algorithm for this high-dimensional forward UQ problem.

4.1. Rank adaptive tensor recovery (RATR)

As discussed in section 3, the standard tensor recovery quadrature requires a given rank of the data tensor, which causes difficulties for problems where the tensor rank is not given a priori. Especially in our setting, tensor recovery quadratures are applied to compute the gPC coefficients for each kPCA mode (5), where the ranks of data tensors (10) are not given and the data tensor ranks associated with different kPCA modes can be different. To address this issue, we develop a new rank adaptive tensor recovery (RATR) approach.

Our idea is that, starting with setting the CP rank $R = 1$, we gradually increase the CP rank until the recovered tensor $\mathbf{X}$ approximates the exact tensor $\mathbf{X}_{\text{exact}}$ well. To measure the quality of the recovered tensor, the following quantity of error is introduced

$$\varepsilon_{\Theta'}(\mathbf{X}) = \frac{\|\mathbb{P}_{\Theta'}(\mathbf{X}) - \mathbb{P}_{\Theta'}(\mathbf{X}_{\text{exact}})\|_2}{\|\mathbb{P}_{\Theta'}(\mathbf{X}_{\text{exact}})\|_2}, \tag{28}$$

where Θ is the observation index set, Θ' is a validation index set (see [28] for validation) randomly sampled from $\Theta_{\text{full}} := \{[j_1, j_2, \ldots, j_d]^T \in \mathbb{N}^d \mid j_k = 1, \ldots, n \text{ for } k = 1, \ldots, d\}$, such that $\Theta' \cap \Theta = \emptyset$ and $|\Theta'| < |\Theta| \ll d^n$. Evaluating the relative error $\varepsilon_{\Theta'}(\mathbf{X})$ requires $O((|\Theta| + |\Theta'|)dR)$ flops, which is discussed in section 3.2.

Since the optimization problem (21) is non-convex, initial factor matrices in Algorithm 2 need to be properly chosen. We provide a detailed analysis of the initialization strategy in section 4.2. Here, supposing the tensor $\mathbf{X}^{(R)} = [[A_1, \ldots, A_d]]$ is obtained, where $\{A_1, \ldots, A_d\}$ is the solution of (21) with rank $R \geq 1$, we consider one higher rank, i.e., $R + 1$. While an analogous approach for tensor completion using tensor train decomposition can be found in [29], we here focus on CP decomposition and give the following scheme of rank-one update. The initial factor matrices for rank $R + 1$ is set to the rank-one updates of the factor matrices of $\mathbf{X}^{(R)}$, i.e.,

$$A_k^{(0)} = [A_k, \delta\boldsymbol{a}], \quad \text{for } k = 1, \ldots, d, \tag{29}$$

where $\delta\boldsymbol{a} \in \mathbb{R}^n$ is a random perturbation vector and $A_1, \ldots, A_d$ are the factor matrices of $\mathbf{X}^{(R)}$. With these new initial factor matrices, the recovered tensor $\mathbf{X}^{(R+1)}$ for rank $R + 1$ are obtained using Algorithm 2. To assess the progress obtained through this update of the CP rank, the difference between the recovery errors of $\mathbf{X}^{(R+1)}$ and $\mathbf{X}^{(R)}$ are assessed through $\Delta\varepsilon_{\Theta'} := \varepsilon_{\Theta'}(\mathbf{X}^{(R)}) - \varepsilon_{\Theta'}(\mathbf{X}^{(R+1)})$, where $\varepsilon_{\Theta'}(\cdot)$ is computed through (28). After that, we update the CP rank $R := R + 1$, and the above procedure is repeated until $\Delta\varepsilon_{\Theta'} < 0$, i.e., $\varepsilon_{\Theta'}(\mathbf{X}^{(R+1)}) > \varepsilon_{\Theta'}(\mathbf{X}^{(R)})$.

Details of our RATR method are presented in Algorithm 3. The initial rank-one matrices $A_1^{(0)}, \ldots, A_d^{(0)}$ in the input are discussed in the next section. The other inputs are the observation index set Θ, the validation index set Θ', and

entries of the data tensor (see (10)) on these index sets ($\mathbb{P}_\Theta(\mathbf{X}_{\text{exact}})$ and $\mathbb{P}_{\Theta'}(\mathbf{X}_{\text{exact}})$). To start the **While** loop of this algorithm, $\Delta\varepsilon_{\Theta'}$ is initially set to an arbitrary number that is larger than 0 on line 1. The output of this algorithm gives an estimation of the data tensor and its estimated rank. The cost of this algorithm is $C_{Alg2} + O(|\Theta'|dR)$.

Algorithm 3 Rank adaptive tensor recovery (RATR)

Input: $\Theta, \Theta', \Upsilon, \mathbb{P}_\Theta(\mathbf{X}_{\text{exact}}), \mathbb{P}_{\Theta'}(\mathbf{X}_{\text{exact}})$, and initial rank-one matrices $A_k^{(0)}$ for $k = 1, \cdots, d$.

1: Initialize the CP rank $R := 1$ and set $\Delta\varepsilon_{\Theta'} > 0$.

2: Run Algorithm 2 to obtain $\mathbf{X}^{(1)}$.

3: Compute $\varepsilon_{\Theta'}\left(\mathbf{X}^{(1)}\right)$ by (28).

4: **while** $\Delta\varepsilon_{\Theta'} \geq 0$ **do**

5:      Initialize $A_k^{(0)} := [A_k, \delta a]$ for $k = 1, \ldots, d$, where each A_k is a factor matrix of $\mathbf{X}^{(R)}$.

6:      Update the CP rank $R := R + 1$.

7:      Run Algorithm 2 to obtain $\mathbf{X}^{(R)}$.

8:      Compute $\varepsilon_{\Theta'}\left(\mathbf{X}^{(R)}\right)$ by (28).

9:      Compute the relative change in errors $\Delta\varepsilon_{\Theta'} := \varepsilon_{\Theta'}\left(\mathbf{X}^{(R-1)}\right) - \varepsilon_{\Theta'}\left(\mathbf{X}^{(R)}\right)$.

10: **end while**

11: Let $\mathbf{X} := \mathbf{X}^{(R-1)}$.

12: Let $R := R - 1$.

Output: the CP rank R and the recovered tensor $\mathbf{X}$.

4.2. Numerical stability analysis for RATR

While the tensor recovery problem (21) is a non-convex optimization problem, the initial guesses for the factor matrices need to be chosen properly. As discussed in section 3, (21) is solved using the alternative minimization iterative method, where the generalized lasso problem (23) needs to be solved at each iteration step. As studied in [30], (23) becomes ill-defined if B is ill-conditioned. Therefore, a necessary condition for the initial factor matrices in (21) is that the resulting matrix B (see Definition 2) needs to be well-conditioned. In this section, we first show that if the initial factor matrices are sampled through some given distributions, the condition number of B is bounded with high probability for the case of rank $R = 1$. Next, we focus on the rank-one update procedure in our RATR approach (on line 5 of Algorithm 3), and show that the condition number of B in (23) associated this update procedure is bounded under certain conditions. We begin our analysis with introducing the following definitions.

Definition 4 (Uniform observation index set). *An observation index set Θ is uniform if and only if $|\{\boldsymbol{j}|\boldsymbol{j} = [j_1, \ldots, j_d] \in \Theta \text{ and } j_k = i\}| = |\Theta|/n$ for each $i = 1, \ldots, n$ and $k = 1, \ldots, d$, where $|\cdot|$ denotes the size of a set.*

Definition 5 (*d*-th order ratio with *m* degrees of freedom). *Let $\psi_1, \ldots, \psi_{d-1}$ form a random sample from a given distribution $\mathbb{P}$ and for a given positive integer d, let $\Psi := \prod_{j=1}^{d-1} \psi_j$. For a given positive integer m, let $\Psi_1, \ldots, \Psi_m$ form a random sample from the sampling distribution of Ψ, and let $\Xi := \sum_{k=1}^{m} \Psi_k^2$. The d-th order ratio with m degrees of*

freedom of $\mathbb{P}$ is

$$\mu = \frac{\sqrt{\mathrm{Var}(\Xi)}}{\mathbb{E}(\Xi)}, \tag{30}$$

where $\mathbb{E}(\Xi)$ and $\mathrm{Var}(\Xi)$ are the expectation and the variance of Ξ respectively.

Note that μ in (30) is typically called the coefficient of variation of Ξ [31, p. 845-846].

Theorem 1. *Let $[[A_1, \ldots, A_d]] \in \mathbb{R}^{n \times \cdots \times n}$ be a d-th order rank-one tensor and Θ be an observation index set. Suppose that for $i = 1, \ldots, d$, all entries of A_i form a random sample from distribution $\mathbb{P}$. Assume that the observation index set Θ is uniform. For a given constant $c_1 > 1$, if the d-th order ratio with $|\Theta|/n$ degrees of freedom of $\mathbb{P}$ satisfies $\mu < 1$, $Q := (c_1 - 1)/(c_1\mu + \mu) > 1$, then the condition number of $B = \mathcal{B}_{\Theta, k}([[A_1, A_2, \ldots, A_d]])$ for $k = 1, 2, \ldots, d$ (see Definition 2) satisfies $\mathrm{cond}^2(B) \le c_1$ with probability at least $1 - 1/Q^2$.*

Proof. Since the entries of B are zero except $B(s(\Theta, j), j_k)$ for $j = [j_1, \ldots, j_d]^T \in \Theta$ (see Definition 2) where $s(\Theta, \cdot)$ is the sort operator, each row of B can have at most one nonzero entry. Therefore, $B^T B$ is a diagonal matrix.

Since all entries of A_i form a random sample from distribution $\mathbb{P}$ for $i = 1, \ldots, d$, based on Definition 2, for $j = [j_1, \ldots, j_d]^T \in \Theta$, $k = 1, \ldots, d$ and $j_k = 1, \ldots, n$, we can express $B(s(\Theta, j), j_k) := \Psi := \prod_{j=1}^{d-1} \psi_j$, where $\psi_1, \ldots, \psi_{d-1}$ form a random sample from $\mathbb{P}$. We next denote $\Psi_{s(\Theta, j), j_k} := B(s(\Theta, j), j_k)$, where $\Psi_{s(\Theta, j), j_k}$ form a random sample from the sampling distribution of Ψ for $j = [j_1, \ldots, j_d]^T \in \Theta$, $k = 1, \ldots, d$ and $j_k = 1, \ldots, n$. Therefore, with the assumption that Θ is uniform, we have $B^T B = \mathrm{diag}(\Xi_1, \ldots, \Xi_n)$ with $\Xi_i := \sum_{j=1}^{|\Theta|/n} (\Psi_j)^2$ for $i = 1, \ldots, n$, where Ψ_j for $j = 1, \ldots, \Theta|/n$ form a random sample from the sampling distribution of Ψ.

According to the Chebyshev inequality,

$$\mathrm{Prob}\left(|\Xi - \mathbb{E}(\Xi)| > Q\sqrt{\mathrm{Var}(\Xi)}\right) \le \frac{\mathrm{Var}(\Xi)}{Q^2 \mathrm{Var}(\Xi)},$$

which is equivalent to

$$\mathrm{Prob}\left(\mathbb{E}(\Xi) - Q\sqrt{\mathrm{Var}(\Xi)} \le \Xi \le \mathbb{E}(\Xi) + Q\sqrt{\mathrm{Var}(\Xi)}\right) \ge 1 - \frac{1}{Q^2}. \tag{31}$$

Using $\mathrm{diag}(\Xi_1, \ldots, \Xi_n) = B^T B$ gives

$$\mathrm{cond}^2(B) = \frac{\max\{\Xi_1, \ldots, \Xi_n\}}{\min\{\Xi_1, \ldots, \Xi_n\}} \le \frac{\mathbb{E}(\Xi) + Q\sqrt{\mathrm{Var}(\Xi)}}{\mathbb{E}(\Xi) - Q\sqrt{\mathrm{Var}(\Xi)}} = \frac{1 + Q\mu}{1 - Q\mu}. \tag{32}$$

Noting that $Q := (c_1 - 1)/(c_1\mu + \mu) > 1$ which gives $c_1 = (1 + Q\mu)/(1 - Q\mu)$, combining (31) and (32) establishes

$$\mathrm{cond}^2(B) \le c_1$$

with probability at least $1 - 1/Q^2$. $\qquad\qquad\square$

The conditions in Theorem 1 require that $\mu < 1$ (μ is defined in Definition 5) and imply $Q\mu < 1$, such that $\mathrm{cond}(B)$ is bounded above with probability at least $1 - 1/Q^2$. To achieve a high probability for a bounded $\mathrm{cond}(B)$, Q should be large and μ should then be small. So, the initial factor matrices (inputs of Algorithm 2) should be generated using realizations of a distribution $\mathbb{P}$ of which μ is small. As an example, we show the estimated μ (the d-th order ratio with

Table 1: Examples of the d-th order ratio with $|m|$ degrees of freedom for several standard distributions with $d = 48$ and $m = 33$ and the corresponding values of cond(B).

Distribution $\mathbb{P}$	Estimated μ	Average cond(B)
U(1, 2)	0.0394	1.9031
N(9, 0.1)	0.0242	1.1491
U(1, 3)	0.1051	3.7348
N(9, 0.5)	0.1390	1.1830
U(0, 1)	231.4262	2.27×10^3
N(0, 1)	163.5804	1.89×10^4

$|m|$ degrees of freedom) for several standard distributions in Table 1, where we set $d = 48$ and $m = 33$. Here, U(a_1, a_2) refers to a uniform distribution on the interval $[a_1, a_2]$, and N(a_1, a_2) refers to a normal distribution with mean a_1 and standard deviation a_2. To compute the estimated μ in Table 1, $\mathbb{E}(\Xi)$ and Var(Ξ) in (30) are computed using the sample mean and the sample variance of 10^5 samples of Ξ (note that the relationship between Ξ and $\mathbb{P}$ is stated in Definition 5). In the procedure of generating each sample of Ξ, the corresponding B is formulated and its condition number cond(B) is stored (see Definition 5 and Theorem 1 for the relationship between Ξ and B). The 10^5 samples of Ξ are associated with 10^5 samples of cond(B), and Table 1 also shows the average of these samples of cond(B) associated with each distribution. As shown in Table 1, the distributions listed above the dash line have $\mu < 1$, and they therefore can be used to generate initial factor matrices, while U(0, 1) and N(0, 1) should not be used.

Next, our analysis proceeds through induction. That is, supposing for a rank R tensor $\mathbf{X}^{(R)} = [[A_1, \ldots, A_d]]$, its corresponding B (see Definition 2) is well-conditioned, we show that the matrix B associated with $\mathbf{X}^{(R+1)} = [[A_1^{(0)}, \ldots, A_d^{(0)}]]$ is also well-conditioned, where $A_k^{(0)} = [A_k, \delta a]$ for $k = 1, \ldots, d$ are the rank-one updates of the factor matrices and $\delta a \in \mathbb{R}^n$ is a perturbation vector. Before introducing our main theorem (Theorem 2), the following lemma is given.

Lemma 1. *Given two matrices $X_1 \in \mathbb{R}^{n_1 \times n_2}$ and $X_2 \in \mathbb{R}^{n_1 \times n_3}$ with full column ranks where $n_1 > n_2 \geq n_3$, let their singular value decompositions be $X_1 = U_1 \Sigma_1 V_1^T$ and $X_2 = U_2 \Sigma_2 V_2^T$, where $U_1 \in \mathbb{R}^{n_1 \times n_1}$, $\Sigma_1 \in \mathbb{R}^{n_1 \times n_2}$, $V_1 \in \mathbb{R}^{n_2 \times n_2}$, $U_2 \in \mathbb{R}^{n_1 \times n_1}$, $\Sigma_2 \in \mathbb{R}^{n_1 \times n_3}$ and $V_2 \in \mathbb{R}^{n_3 \times n_3}$, and let $u_1^{(i)} \in \mathbb{R}^{n_1}$ for $i = 1, \ldots, n_1$ and $u_2^{(j)} \in \mathbb{R}^{n_1}$ for $j = 1, \ldots, n_1$ denote the left singular vectors of X_1 and X_2 respectively. Assume the following two conditions hold: first*

$$\Sigma_2 \Sigma_2^T = \mathrm{diag}(\underbrace{\lambda, \ldots, \lambda}_{n_3}, \underbrace{0, \cdots, 0}_{n_1 - n_3}), \tag{33}$$

where λ is a positive constant; second

$$\langle u_2^{(j)}, u_1^{(i)} \rangle = 0, \text{ for } j = n_3 + 1, \ldots, n_1 \text{ and } i = 1, \ldots, n_1. \tag{34}$$

Then $X_1 X_1^T$ and $X_2 X_2^T$ commute, i.e., $X_1 X_1^T X_2 X_2^T = X_2 X_2^T X_1 X_1^T$.

Proof. Let $P = U_2^T U_1 \in \mathbb{R}^{n_1 \times n_1}$, $X = P \Sigma_1 \Sigma_1^T P^T \in \mathbb{R}^{n_1 \times n_1}$ and $Z = X \Sigma_2 \Sigma_2^T \in \mathbb{R}^{n_1 \times n_1}$. Using (33), $Z(i, j) = 0$ for $j = n_3 + 1, \ldots, n_1$ and $i = 1, \ldots, n_1$, while $Z(i, j) = \lambda X(i, j)$ for $i = 1, \ldots, n_1$ and $j = 1, \ldots, n_3$. Denoting the singular

values of $X_1 \in \mathbb{R}^{n_1 \times n_2}$ by $\sigma_1, \ldots, \sigma_{n_2}$ (note that $n_1 > n_2$) gives

$$X = \sum_{i=1}^{n_2} \sigma_i^2 p_i p_i^T, \tag{35}$$

where p_i for $i = 1, \ldots, n_1$ are columns of P. Since $p_i = U_2^T u_1^{(i)}$ for $i = 1, \ldots, n_1$, each element of p_i is $p_i(j) = \langle u_2^{(j)}, u_1^{(i)} \rangle$. Using (34) gives $p_i(j) = 0$ for $j = n_3+1, \ldots, n_1$ and $i = 1, \ldots, n_1$. Therefore, (35) gives $X(i, j) = 0$ for $j = n_3+1, \ldots, n_1$ and $i = n_3 + 1, \ldots, n_1$. In summary, each entry of Z is

$$Z(i, j) = \begin{cases} \lambda X(i, j) & \text{for } i \le n_3, j \le n_3, \\ 0 & \text{otherwise.} \end{cases} \tag{36}$$

Similarly, each entry of $Z' := \Sigma_2 \Sigma_2^T X \in \mathbb{R}^{n_1 \times n_1}$ is

$$Z'(i, j) = \begin{cases} \lambda X(i, j) & \text{for } i \le n_3, j \le n_3, \\ 0 & \text{otherwise.} \end{cases} \tag{37}$$

Combing (36)–(37) gives $Z = Z'$, and thus $X\Sigma_2\Sigma_2^T = \Sigma_2\Sigma_2^T X$, which leads to

$$U_2^T U_1 \Sigma_1 \Sigma_1^T U_1^T U_2 \Sigma_2 \Sigma_2^T = \Sigma_2 \Sigma_2^T U_2^T U_1 \Sigma_1 \Sigma_1^T U_1^T U_2.$$

Left multiplying both sides of the above equation by U_2 and right multiplying them by U_2^T give

$$X_1 X_1^T X_2 X_2^T = X_2 X_2^T X_1 X_1^T.$$

$\square$

Theorem 2. *Let Θ be an observation index set, and $[[A_1 \ldots, A_d]] \in \mathbb{R}^{n \times n \times \cdots \times n}$ be a d-th order tensor with rank R. Suppose that $A_{i_\delta} = [A_i, \delta a]$ with $\delta a = [1, \ldots, 1]^T \in \mathbb{R}^n$ is a rank-one update of A_i for $i = 1, \ldots, d$. Let $B := \mathcal{B}_{\Theta,k}([[A_1 \ldots, A_d]])$, $\delta B := \mathcal{B}_{\Theta,k}([[\delta a, \delta a, \ldots, \delta a]])$, and $B_\delta = \mathcal{B}_{\Theta,k}([[A_{1\delta}, \ldots, A_{d\delta}]])$, and let $u_{\delta B}^{(j)}$ and $u_B^{(l)}$ are the j-th and the l-th left singular vectors of δB and B respectively for $j, l = 1, \ldots, |\Theta|$. If the following three conditions hold:*

1) *the observation index set Θ is uniform as defined in Definition 4,*
2) *there exist positive constants c_2 and c_3 which are independent of B, such that $\text{cond}^2(B) = s_{\max}^2(B)/s_{\min}^2(B) \le c_2$ and $|\Theta|/(n s_{\min}^2(B)) \le c_3$, where $s_{\max}$ and $s_{\min}$ are the largest and the smallest singular values of B respectively,*
3) *$\langle u_{\delta B}^{(j)}, u_B^{(l)} \rangle = 0$, for $j = n + 1, \ldots, |\Theta|$ and $l = 1, \ldots, |\Theta|$,*

then the condition number of B_δ satisfies that

$$\text{cond}^2(B_\delta) \le c_2 + c_3. \tag{38}$$

Proof. By Proposition 1, we have $B_\delta = [B, \delta B]$. Note that the largest singular value of B_δ satisfies that $s_{\max}^2(B_\delta) = \lambda_{\max}(BB^T + \delta B \delta B^T)$. Using the min-max theorem,

$$s_{\max}^2(B_\delta) = \max_{\|x\|_2=1} x^T (BB^T + \delta B \delta B^T)x$$

$$\le \max_{\|x\|_2=1} x^T (BB^T)x + \max_{\|x\|_2=1} x^T (\delta B \delta B^T)x$$

$$= s_{\max}^2(B) + s_{\max}^2(\delta B)$$

16

Next we consider the smallest singular value of $\boldsymbol{B}_\delta$ under the above conditions. Let $\boldsymbol{B} = \boldsymbol{U}_1\boldsymbol{\Sigma}_1\boldsymbol{V}_1^T$ and $\delta\boldsymbol{B} = \boldsymbol{U}_2\boldsymbol{\Sigma}_2\boldsymbol{V}_2^T$ be the singular value decompositions of $\boldsymbol{B}$ and $\delta\boldsymbol{B}$, respectively.

By condition 1) and noting that $\delta\boldsymbol{a} = [1, \ldots, 1]^T$, we have $\boldsymbol{\Sigma}_2\boldsymbol{\Sigma}_2^T = \mathrm{diag}(\underbrace{|\Theta|/n, \ldots, |\Theta|/n}_{n}, \underbrace{0, \cdots, 0}_{|\Theta|-n})$. Since $\boldsymbol{\Sigma}_2\boldsymbol{\Sigma}_2^T = \mathrm{diag}(\underbrace{|\Theta|/n, \ldots, |\Theta|/n}_{n}, \underbrace{0, \cdots, 0}_{|\Theta|-n})$ and $\langle \boldsymbol{u}_{\delta B}^{(j)}, \boldsymbol{u}_{B}^{(l)} \rangle = 0$, $j = n+1, \ldots, |\Theta|$, and $l = 1, \ldots, |\Theta|$. By Lemma 1, $\boldsymbol{B}\boldsymbol{B}^T$ and $\delta\boldsymbol{B}\delta\boldsymbol{B}^T$ commute. Therefore, $\boldsymbol{B}\boldsymbol{B}^T$ and $\delta\boldsymbol{B}\delta\boldsymbol{B}^T$ can be simultaneously diagonalizable, i.e., there exists an orthonormal matrix $\boldsymbol{Q}$ such that

$$\boldsymbol{B}\boldsymbol{B}^T = \boldsymbol{Q}\boldsymbol{\Lambda}_1\boldsymbol{Q}^T, \ \boldsymbol{\Lambda}_1 = \mathrm{diag}(s_{\max}^2(\boldsymbol{B}), \ldots, s_{\min}^2(\boldsymbol{B}), \underbrace{0, \cdots, 0}_{|\Theta|-nR}),$$

$$\delta\boldsymbol{B}\delta\boldsymbol{B}^T = \boldsymbol{Q}\boldsymbol{\Lambda}_2\boldsymbol{Q}^T, \ \boldsymbol{\Lambda}_2 = \mathrm{diag}(s_{\max}^2(\delta\boldsymbol{B}), \ldots, s_{\min}^2(\delta\boldsymbol{B}), \underbrace{0, \cdots, 0}_{|\Theta|-n}).$$

Then it follows that $\boldsymbol{B}\boldsymbol{B}^T + \delta\boldsymbol{B}\delta\boldsymbol{B}^T = \boldsymbol{Q}(\boldsymbol{\Lambda}_1 + \boldsymbol{\Lambda}_2)\boldsymbol{Q}^T$, and $s_{\min}^2(\boldsymbol{B}_\delta) = s_{\min}^2(\boldsymbol{B})$, which gives that

$$\begin{aligned}
\mathrm{cond}^2(\boldsymbol{B}_\delta) &= \frac{s_{\max}^2(\boldsymbol{B}_\delta)}{s_{\min}^2(\boldsymbol{B}_\delta)} \\
&\leq \frac{s_{\max}^2(\boldsymbol{B}) + s_{\max}^2(\delta\boldsymbol{B})}{s_{\min}^2(\boldsymbol{B})} = \mathrm{cond}^2(\boldsymbol{B}) + |\Theta|/(n s_{\min}^2(\boldsymbol{B})) \leq c_2 + c_3.
\end{aligned}$$

$\square$

In summary, in our RATR algorithm (Algorithm 3), the fixed-rank tensor recovery algorithm (Algorithm 2) is invoked. The stability of Algorithm 2 is dependent on the observation index set Θ and the initial factor matrices. From our above analysis, if the observation index set Θ is uniform, and the initial rank-one factor matrices are sampled from the distributions given in Table 1 with $\mu < 1$, the first tensor recovery step in RATR (on line 2 of Algorithm 3) is stable with high probability. In the rank adaptive procedure, our analysis shows that the initial factor matrices specified on line 5 of Algorithm 3 can lead to stable tensor recovery on line 7 of Algorithm 3, if each $\boldsymbol{B}$ (see Definition 1) associated with the data tensor obtained in the previous iteration step is well-conditioned. While the overall tensor recovery problem (21) is solved using the alternative minimization iterative method, our analysis is restricted to the first iteration step. To analyze the stability for the generalized lasso problem (23) for arbitrary iterations steps during the alternative minimization procedure remains an open problem. Nevertheless, our analysis here gives a systematic guidance to initialize the factor matrices for Algorithm 3 (also for Algorithm 2), and our numerical results in section 5 show that our RATR approach is stable and efficient.

4.3. RATR-collocation algorithm

Our goal is to efficiently conduct uncertainty propagation from the random input $\boldsymbol{\xi} \in I^d$ to the discrete solution (which is high-dimensional) $\boldsymbol{y} = \chi(\boldsymbol{\xi}) = [u(x^{(1)}, \boldsymbol{\xi}), \ldots, u(x^{(N_h)}, \boldsymbol{\xi})]^T \in \mathcal{M}$ of (1)–(2). The overall procedure of RATR-collocation approach is presented as the following three steps: generating data, processing data to construct RATR-collocation model, and conducting predictions using the RATR-collocation model.

For generating data, a tensor style quadrature rule [32] is first specified with n quadrature nodes in each dimension. The full index set is then defined as $\Theta_{\mathrm{full}} := \{[j_1, j_2, \ldots, j_d]^T \in \mathbb{N}^d \mid j_k = 1, \ldots, n \text{ for } k = 1, \ldots, d\}$ and quadrature

nodes are denoted as $\{\boldsymbol{\xi}_{j_1\ldots j_d}$, for $\boldsymbol{j} = [j_1, j_2, \ldots, j_d]^T \in \Theta_{\text{full}}\}$. A observation index set Θ, and a validation index set Θ' are randomly selected from Θ_{full}, such that $\Theta' \cap \Theta = \emptyset$ and $|\Theta'| < |\Theta| \ll n^d$. After that, snapshots $\chi(\boldsymbol{\xi}_{j_1\ldots j_d})$ for $\boldsymbol{j} = [j_1, j_2, \ldots, j_d]^T \in \Theta \cup \Theta'$ are computed through solving deterministic versions of (1)–(2) with high-fidelity numerical schemes. At the end of this step, the snapshots are stored in a data matrix $\boldsymbol{Y} = [\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \cdots, \boldsymbol{y}^{(N_t)}]$, where $\boldsymbol{y}^{(s(\Theta \cup \Theta', \boldsymbol{j}))} := \chi(\boldsymbol{\xi}_{j_1\ldots j_d})$ for $\boldsymbol{j} = [j_1, j_2, \ldots, j_d]^T \in \Theta \cup \Theta'$ and $s(\cdot, \cdot)$ is the sort operator defined in Definition 1.

To process the data, kPCA (see section 2.1) is first applied to result in a reduced-dimensional representation of $\boldsymbol{Y}$—each $\boldsymbol{y}^{(l)} = \chi(\boldsymbol{\xi}^{(l)}) \in \mathcal{M}$ is mapped to $\boldsymbol{\gamma}(\boldsymbol{\xi}^{(l)}) = [\gamma_1(\boldsymbol{\xi}^{(l)}), \ldots, \gamma_{N_r}(\boldsymbol{\xi}^{(l)})]^T \in \mathcal{M}_r$ for $l = 1, \ldots, N_t$. After that, for each kPCA mode $e = 1, \ldots, N_r$, an estimated data tensor (10) is generated through our RATR approach presented in section 4.1. That is, through setting the observed data $\mathbb{P}_\Theta(\mathbf{X}_{\text{exact}}) := \boldsymbol{p}$ with $\boldsymbol{p} = [p_1, \ldots, p_{|\Theta|}]^T$, where $p_{s(\Theta, \boldsymbol{j})} = \gamma_e\left(\boldsymbol{\xi}^{(s(\Theta, \boldsymbol{j}))}\right)$ for $\boldsymbol{j} \in \Theta$, and the validation data $\mathbb{P}_{\Theta'}(\mathbf{X}_{\text{exact}}) := \boldsymbol{p}$ with $\boldsymbol{p} = [p_1, \ldots, p_{|\Theta'|}]^T$, where $p_{s(\Theta', \boldsymbol{j})} = \gamma_e\left(\boldsymbol{\xi}^{(s(\Theta', \boldsymbol{j}))}\right)$ for $\boldsymbol{j} \in \Theta'$, Algorithm 3 gives an approximation of $\mathbf{X}_e$, which is denoted by $\tilde{\mathbf{X}}_e$. With this estimated data tensor, each gPC approximation (see (5)) $\gamma_e(\boldsymbol{\xi}) \approx \gamma_e^{\text{gPC}} := \sum_{|\boldsymbol{i}|=0}^p c_{e\boldsymbol{i}}\Phi_{\boldsymbol{i}}(\boldsymbol{\xi})$ for $e = 1, \ldots, N_r$ is obtained with coefficients computed through $c_{e\boldsymbol{i}} := \langle \tilde{\mathbf{X}}_e, \mathbf{W}_{\boldsymbol{i}} \rangle$, where $\mathbf{W}_{\boldsymbol{i}}$ is defined in (13). In the following, we call these gPC approximations $\{\gamma_e^{\text{gPC}}(\boldsymbol{\xi}) := \sum_{|\boldsymbol{i}|=0}^p c_{e\boldsymbol{i}}\Phi_{\boldsymbol{i}}(\boldsymbol{\xi})\}_{e=1}^{N_r}$ the RATR-collocation model.

The above two steps for generating data and constructing the RATR-collocation model are summarized in Algorithm 4. For conducting a prediction of the snapshot for an arbitrary realization of $\boldsymbol{\xi}$, we first use RATR-collocation model to compute the output $[\gamma_1^{\text{gPC}}(\boldsymbol{\xi}), \ldots, \gamma_{N_r}^{\text{gPC}}(\boldsymbol{\xi})]^T$ in the reduced-dimensional manifold $\mathcal{M}_r$. With the reduced output $[\gamma_1^{\text{gPC}}(\boldsymbol{\xi}), \ldots, \gamma_{N_r}^{\text{gPC}}(\boldsymbol{\xi})]^T$ and the data matrix $\boldsymbol{Y}$ (generated in Algorithm 4), an estimation of the snapshot is obtained through the inverse mapping (see section 2.3 and [15]), which is denoted as $\boldsymbol{y}_{\text{RATR}} := \chi_{\text{RATR}}(\boldsymbol{\xi}) \in \mathbb{R}^{N_h}$.

Algorithm 4 RATR-colocation in the reduced-dimensional manifold $\mathcal{M}_r$

Input: a full index set Θ_{full}, quadrature nodes $\left\{\boldsymbol{\xi}_{j_1\ldots j_d}, \text{ for } \boldsymbol{j} = [j_1, j_2, \ldots, j_d]^T \in \Theta_{\text{full}}\right\}$, an observation index set Θ, a validation index set Θ', and a gPC order p.

1: Generate a data matrix $\boldsymbol{Y} = \left[\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \cdots, \boldsymbol{y}^{(N_t)}\right]$, where $\boldsymbol{y}^{(s(\Theta \cup \Theta', \boldsymbol{j}))} := \chi\left(\boldsymbol{\xi}_{j_1\ldots j_d}\right)$ for $\boldsymbol{j} \in \Theta \cup \Theta'$ are obtained through high-fidelity simulations for deterministic versions of (1)–(2) and $s(\cdot, \cdot)$ is defined in Definition 1.

2: Perform kPCA for $\boldsymbol{Y}$ to obtain $\boldsymbol{\gamma}\left(\boldsymbol{\xi}^{(l)}\right) = \left[\gamma_1\left(\boldsymbol{\xi}^{(l)}\right), \ldots, \gamma_{N_r}\left(\boldsymbol{\xi}^{(l)}\right)\right]^T$ for $l = 1, \ldots, N_t$.

3: **for** $e = 1 : N_r$ **do**

4:    Define $\mathbb{P}_\Theta(\mathbf{X}_{\text{exact}}) := \boldsymbol{p}$ with $\boldsymbol{p} = [p_1, \ldots, p_{|\Theta|}]^T$, where $p_{s(\Theta, \boldsymbol{j})} = \gamma_e\left(\boldsymbol{\xi}^{(s(\Theta, \boldsymbol{j}))}\right)$ for $\boldsymbol{j} \in \Theta$.

5:    Define $\mathbb{P}_{\Theta'}(\mathbf{X}_{\text{exact}}) := \boldsymbol{p}$ with $\boldsymbol{p} = [p_1, \ldots, p_{|\Theta'|}]^T$, where $p_{s(\Theta', \boldsymbol{j})} = \gamma_e\left(\boldsymbol{\xi}^{(s(\Theta', \boldsymbol{j}))}\right)$ for $\boldsymbol{j} \in \Theta'$.

6:    Generate an estimated data tensor $\mathbf{X}$ using Algorithm 3, and define $\tilde{\mathbf{X}}_e := \mathbf{X}$.

7:    Generate the gPC approximation $\gamma_e(\boldsymbol{\xi}) \approx \gamma_e^{\text{gPC}} := \sum_{\|\boldsymbol{i}\|_1=0}^p c_{e\boldsymbol{i}}\Phi_{\boldsymbol{i}}(\boldsymbol{\xi})$ with $c_{e\boldsymbol{i}} := \left\langle \tilde{\mathbf{X}}_e, \mathbf{W}_{\boldsymbol{i}} \right\rangle$ for $\boldsymbol{i} \in \Upsilon := \{\boldsymbol{i} | \boldsymbol{i} \in \mathbb{N}^d \text{ and } \|\boldsymbol{i}\|_1 = 0, \ldots, p\}$, where $\mathbf{W}_{\boldsymbol{i}}$ is defined in (13).

8: **end for**

Output: gPC approximations $\gamma_1^{\text{gPC}}(\boldsymbol{\xi}), \ldots, \gamma_{N_r}^{\text{gPC}}(\boldsymbol{\xi})$ and the data matrix $\boldsymbol{Y} = \left[\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \cdots, \boldsymbol{y}^{(N_t)}\right]$.

---

5. Numerical study

In this section, we first consider diffusion problems in section 5.1 and section 5.2, and consider a Stokes problem in section 5.3. The governing equations of the diffusion problems are

$$-\nabla \cdot [a(x, \boldsymbol{\xi}) \nabla u(x, \boldsymbol{\xi})] = 1 \qquad \text{in} \qquad D \times I^d, \tag{39}$$

$$u(x, \boldsymbol{\xi}) = 0 \qquad \text{on} \qquad \partial D_D \times I^d, \tag{40}$$

$$\frac{\partial u(x, \boldsymbol{\xi})}{\partial n} = 0 \qquad \text{on} \qquad \partial D_N \times I^d, \tag{41}$$

where $\partial u/\partial n$ is the outward normal derivative of u on the boundaries, $\partial D_D \cap \partial D_N = \emptyset$ and $\partial D = \partial D_D \cup \partial D_N$. In the following numerical studies, the spatial domain is taken to be $D = (0, 1) \times (0, 1)$. The condition (40) is applied on the left ($x = 0$) and right ($x = 1$) boundaries, and (41) is applied on the top and bottom boundaries. Defining $H^1(D) := \{u : D \to \mathbb{R}, \int_D u^2 \, dD < \infty, \int_D (\partial u/\partial x_l)^2 \, dD < \infty, l = 1, 2\}$ and $H_0^1(D) := \{v \in H^1(D) \,|\, v = 0 \text{ on } \partial D_D\}$, the weak form of (39)–(41) is to find $u(x, \xi) \in H_0^1(D)$ such that $(a\nabla u, \nabla v) = (1, v)$ for all $v \in H_0^1(D)$. We discretize in space using a bilinear finite element approximation [33], with a uniform 65×65 grid ($N_h = 4225$).

The diffusion coefficient $a(x, \boldsymbol{\xi})$ in our numerical studies is assumed to be a random field with mean function $a_0(x)$, standard deviation σ and covariance function $Cov(x, y)$,

$$Cov(x, y) = \sigma^2 \exp\left(-\frac{|x_1 - y_1|}{l_c} - \frac{|x_2 - y_2|}{l_c}\right), \tag{42}$$

where $x = [x_1, x_2]^T, y = [y_1, y_2]^T \in \mathbb{R}^2$ and l_c is the correlation length. This random field can be approximated by a truncated Karhunen–Loève (KL) expansion [1]

$$a(x, \boldsymbol{\xi}) \approx a_0(x) + \sum_{i=1}^{d} \sqrt{\lambda_i} a_i(x) \xi_i, \tag{43}$$

where $a_i(x)$ and λ_i are eigenfunctions and eigenvalues of (42), d is the number of KL modes retained, and $\{\xi_i\}_{i=1}^d$ are uncorrelated random variables. We set the random variables $\{\xi_i\}_{i=1}^d$ to be independent uniform distributions with range $I = [-1, 1]$, and set $a_0(x) = 1$ and $\sigma^2 = 0.25$. For test problem 1 (in section 5.1), we set $l_c = 0.8$ and $d = 48$, such that at least 95% of the total variance is captured, i.e., $(\sum_{i=1}^d \lambda_i)/(|D|\sigma^2) > 0.95$, where $|D|$ is the area of D. For test problem 2 (in section 5.2), we set $l_c = 1/16$ and again set $d = 48$.

For all test problems, we set the gPC order $p = 2$ (see section 2.2) and take $n = 3$ Gaussian quadrature points for each dimension, while Θ_{full} is constructed by the tensor product of these three points ($|\Theta_{\text{full}}| = 3^{48}$). As in the input of Algorithm 4, an observation index set Θ and a validation index Θ' are required. We test three cases of Θ uniformly sampled from Θ_{full} with sizes $|\Theta| = 100$, 300 and 600 respectively, and generate Θ' using 20 samples uniformly sampled from Θ_{full}, such that $\Theta \cap \Theta' = \emptyset$. Note that the number of high-fidelity simulations (the finite element methods here) in our RATR is $|\Theta| + |\Theta'|$, while that in standard tensor grid collocation [12] is $|\Theta_{\text{full}}| = 3^{48}$ and that in sparse grid collocation [3, 4] is still around 4705 (for a comparable grid level). So, the cost of RATR-collocation is much smaller than the costs of both tensor and sparse grid collocation methods for high-dimensional problems.

For the diffusion test problems. The regularization parameter β in (21) is set to 0.01, the tolerance in Algorithm 2 is set to $\delta = 10^{-5}$, and the initial rank-one matrices for Algorithm 3 are generated with samples of $U(1, 2)$ which

is an optimal initializaiton strategy as discussed in section 4.2. For kPCA as reviewed in section 2.1, we set the criterion for selecting principal components to $tol_{\mathrm{PCA}} = 90\%$, and set the bandwidth to $\sigma_g = 5$ for the diffusion test problems. For a given realization of $\boldsymbol{\xi}$, $\boldsymbol{y} := \chi(\boldsymbol{\xi})$ denotes the finite element solution, and $\boldsymbol{y}_{\mathrm{RATR}} := \chi_{\mathrm{RATR}}(\boldsymbol{\xi})$ refers to a RATR-collocation approximation solution (see section 4.3). A relative error is then defend as

$$\text{Relative error} = \frac{\left\|\boldsymbol{y} - \boldsymbol{y}_{\mathrm{RATR}}\right\|_2}{\|\boldsymbol{y}\|_2}. \tag{44}$$

5.1. Test problem 1: diffusion problem with $l_c = 0.8$ and $d = 48$

For each case of the observation index set Θ (with $N_t := |\Theta| = 100, 300$ and 600 respectively), we first generate the corresponding data matrix Y and apply kPCA for dimension reduction. For the given tolerance $tol_{\mathrm{PCA}} = 90\%$, the number of kPCA modes retained is $N_r = 4$ for the three cases here (see section 2.1 for the definitions of N_r and tol_{PCA}). For each kPCA mode, our RATR algorithm gives an estimation $\tilde{\mathbf{X}}_e$ of the data tensor $\mathbf{X}_e$ for $e = 1, \ldots, N_r$ (see line 6 of Algorithm 4), where $\mathbf{X}_e$ is defined in (10). Tabel 2 shows the estimated CP ranks of $\mathbf{X}_e$ generated through Algorithm 3. It is clear that, these estimated ranks of each $\mathbf{X}_e$ are similar for the three cases of Θ, and they are very small—the maximum estimated CP rank for this test problem is four.

Table 2: Estimated CP ranks of each data tensor $\mathbf{X}_e$ for $e = 1, \ldots, 4$, test problem 1.

rank \ e				
	1	2	3	4
$\|\Theta\|$				
100	4	2	3	1
300	2	1	1	3
600	4	1	1	2

To assess the efficiency of our RATR procedure, we compare Algorithm 3 with the standard fixed-rank tensor recovery approach (Algorithm 2) to recover $\mathbf{X}_1$ with $|\Theta| = 600$ for this test problem. As discussed above, the initial rank-one factor matrices for RATR are generated thorough the distribution $U(1, 2)$. For Algorithm 2, for each given rank $R = 1, \ldots, 4$, two distributions are tested for generating the initial matrices: $U(1, 2)$ and $N(0, 1)$. Note that, as discussed in section 4.2, $U(1, 2)$ is an optimal choice and $N(0, 1)$ is a non-optimal choice for the situation that the CP rank is one. In the following, the fixed-rank tensor recovery approach (Algorithm 2) with initial factor matrices generated through the optimal choice $U(1, 2)$ is denoted by FRTR-O, and that with initial factor matrices generated through the non-optimal choice $N(0, 1)$ is denoted by FRTR-N. Figure 2(a) shows the validation errors (28) of the recovered tensor generated by RATR, FRTR-O and FRTR-N respectively, where it is clear that for each rank $R = 1, \ldots, 4$, our RATR has the smallest validation error. As discussed in section 3.2, the overall tensor recovery problem (21) is solved through the alternative minimization iterative method (see (22)). Looking more closely, the validation errors at each iteration step of the alternative minimization iterative method for $R = 1, 2, 4$ are shown in Figure 2(b), Figure 2(c) and Figure 2(d) respectively (since the results of $R = 3$ and $R = 4$ similar, we only show the results of

$R = 4$). For $R = 1$ (Figure 2(b)), there is no rank adaptive procedure preformed in RATR, and the validation errors of RATR and FRTR-O are the same, while it is clear that they are much smaller than the errors of FRTR-N. Moreover, the validation error of FRTR-N can even become larger as the iteration step increases for $R = 1$, which is consistent with Theorem 1. For $R = 2, 4$ (Figure 2(c) and Figure 2(d)), it can be seen that RATR has the smallest validation errors at each iteration step, which shows that our rank-one updating procedure (on line 5 of Algorithm 3) gives efficient initial factor matrices for the generalized lasso problem (23).

(a) Validation errors w.r.t CP ranks

(b) Validation errors at $R = 1$

(c) Validation errors at $R = 2$

(d) Validation errors at $R = 4$
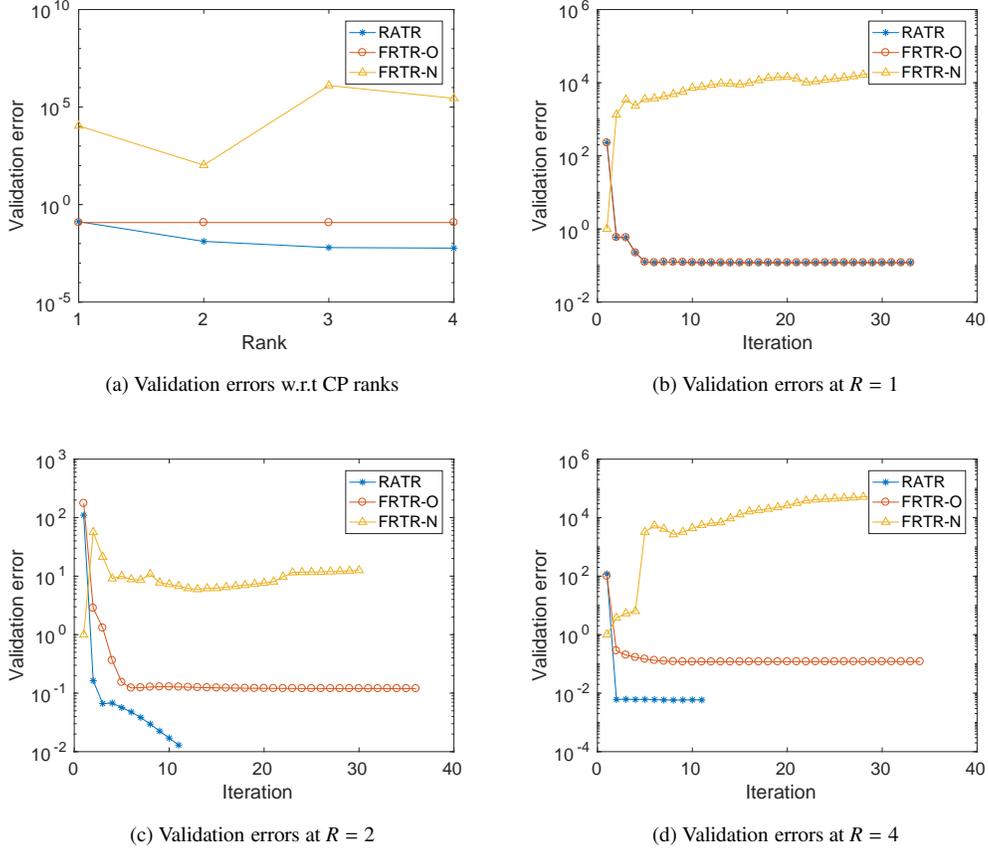
Figure 2: Validation errors of rank adaptive tensor recovery (RATR), fixed-rank tensor recovery with initial factor matrices generated through U(1, 2) (FRTR-O), and fixed-rank tensor recovery with initial factor matrices generated through N(0, 1) (FRTR-N), test problem 1.

While the sparsity of the gPC coefficients is taken into account in the tensor recovery problem (21), we show the absolute value of each the gPC coefficient c_{ei} (see (6)) for each kPCA mode $e = 1, \ldots, 4$ and each gPC multi-index $i \in \Upsilon$ (see section 2.2) in Figure 3. In Figure 3, the gPC multi-index set is labeled as $\Upsilon = \{i^{(1)}, \ldots, i^{|\Upsilon|}\}$, where the indices are sorted by the sort operator $s(\Upsilon, \cdot)$ (see section 3.2, Definition 1). From Figure 3, it is clear that the gPC coefficients are sparse—absolute values of most coefficients are smaller than 10^{-4}, which is consist with the results in [6].

Figure 4 shows the finite element solution y and the RATR-collocation approximation y_{RATR} responding to a given realization of ξ, where it can be seen that they are visually indistinguishable. Finally, we generate 500 samples of ξ, and compute the relative error (44) for the three cases ($|\Theta| = 100, 300$ and 600 respectively). Figure 5 shows Tukey

(a) 1st kPCA mode

(b) 2nd kPCA mode

(c) 3rd kPCA mode
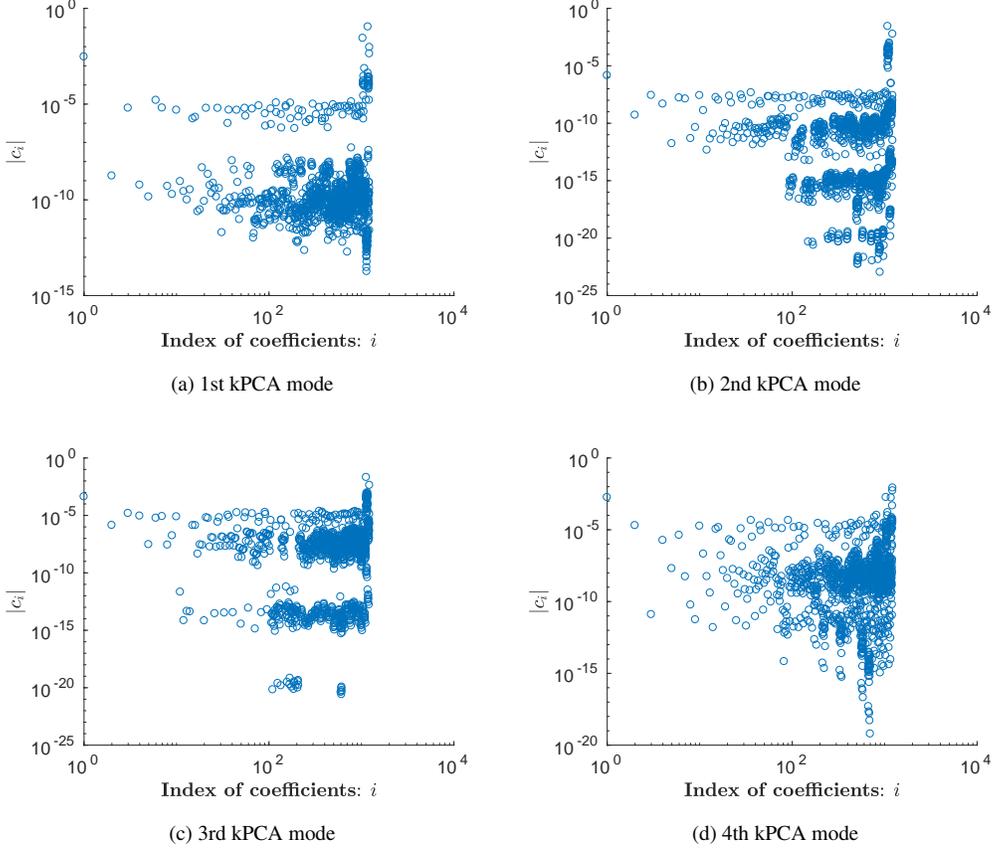
(d) 4th kPCA mode

Figure 3: Sparsity of gPC coefficients for each kPCA mode, test problem 1.

box plots of these errors. Here, the central line in each box is the median, the lower and the upper edges are the first and the third quartiles respectively, and the red crosses are the outliers where the relative errors are large. From Figure 5, it is clear that as the size of the observation index set ($|\Theta|$) increases, values of the median, the first and the third quartiles of the errors decrease.

5.2. Test problem 2: diffusion problem with $l_c = 1/16$ and $d = 48$

For this test problem, the correlation length is very small, and the diffusion problem becomes highly non-smooth. Following the discussion procedure in test problem 1, we first generate the corresponding data matrix Y for the three cases of Θ ($|\Theta| = 100, 300$ and 600) and apply kPCA on it. For $tol_{PCA} = 90\%$, the number of kPCA modes retained is $N_r = 7$ for this test problem. Tabel 3 shows the estimated CP ranks of $\mathbf{X}_e$ generated through Algorithm 3 for each $e = 1, \ldots, 7$, where it is clear that the estimated ranks are small (the maximum of the estimated ranks is seven). Figure 6 shows validation errors of our RATR, FRTR-O and FRTR-N (Algorithm 2 with initial factor matrices generated through U(1, 2) and N(0, 1) respectively) for recovering $\mathbf{X}_1$ (see (10)) with $|\Theta| = 600$. From Figure 6(a), it can be seen that our RATR has the smallest validation error for each rank $R = 2, \ldots, 7$. It is also clear that, as the ranks increase, the error of RATR decreases, while the errors of FRTR-O and FRTR-N do not decrease. The other pictures in Figure 6 show the validation errors at each iteration step of the alternative minimization iterative procedure for $R = 1, 2, 3, 4, 7$

22
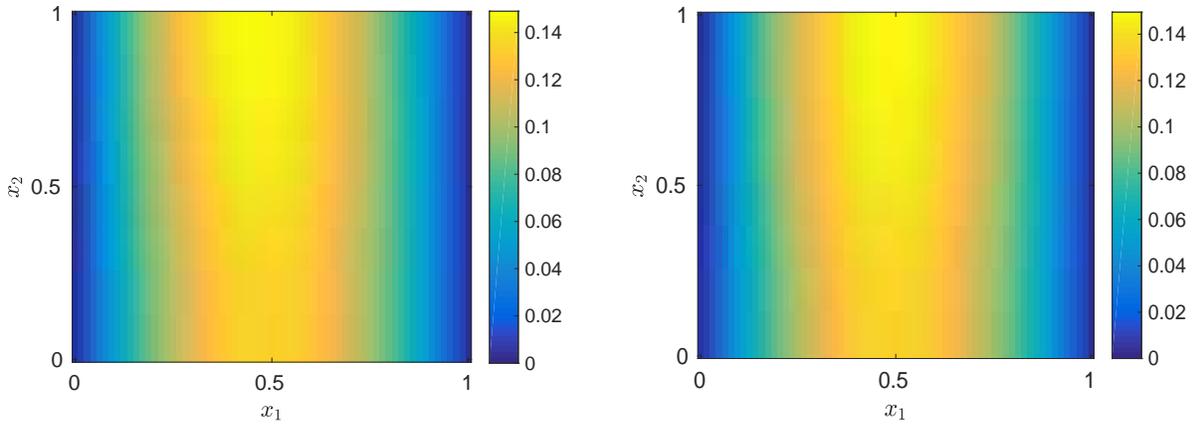
(a) Finite element solution

(b) RATR-collocation approximation

Figure 4: The finite element solution and the RATR-collocation approximation (with $|\Theta| = 600$) responding to a given realization of $\boldsymbol{\xi}$, test problem 1.
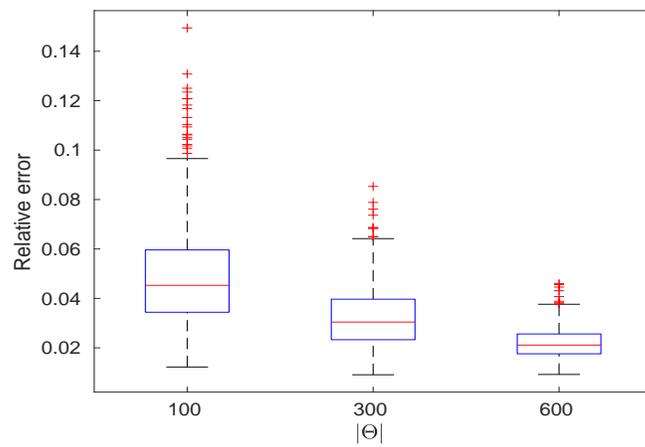
Figure 5: Relative errors of RATR-collocation approximation for 500 test samples, test problem 1.

(since the errors for $R = 5, 6$ are similar to the errors for $R = 7$, they are not shown here). For the case $R = 1$ (Figure 6(b)), while RATR is the same as FRTR-O, the error of RATR is larger than the error of FRTR-N, but the errors of RATR and FRTR-N are both very large (larger than one), which implies that this rank (R=1) is too small to accurately recover the data tensor. As the rank increases, for $R = 2, 3, 4, 7$, the validation errors of RATR are clearly smaller than the errors of FRTR-O and FRTR-N, which is consist with the results in test problem 1. Figure 7 shows the absolute values of the gPC coefficient c_{ei} (see (6)) for $i \in \Upsilon$ and $e = 1, \ldots, 4$ (the first four kPCA modes). It is clear that absolute values of most gPC coefficients are very small. Therefore, the gPC expansions for these four kPCA modes are sparse. For the other kPCA modes ($e = 5, 6, 7$), since the situation is similar to that of the first four kPCA modes, their corresponding gPC coefficients are not shown here, while these gPC expansions are also sparse. Finally, Figure 8 shows Tukey box plots of the relative errors (44) for 500 test samples for this test problem, where the central line in each box is the median, the lower and the upper edges are the first and the third quartiles respectively, and the red crosses are the outliers. From Figure 8, it is clear that, as the size of the observation index set ($|\Theta|$) increases, values of the median, the first and the third quartiles of the errors decrease, which are all consistent with the results in test problem 1.

Table 3: Estimated CP ranks of each data tensor $\mathbf{X}_e$ for $e = 1, \ldots, 7$, test problem 2.

rank \ e $|\Theta|$	1	2	3	4	5	6	7
100	7	1	3	2	1	2	1
300	6	3	1	1	7	1	1
600	7	1	4	3	7	1	3

5.3. Test problem 3: the Stokes equations

The governing equations for this test problem are

$$\nabla \cdot [a(x, \xi)\nabla u(x, \xi)] + \nabla p(x, \xi) = 0 \quad \text{in} \quad D \times I^d, \tag{45}$$

$$\nabla \cdot u(x, \xi) = 0 \quad \text{in} \quad D \times I^d, \tag{46}$$

$$u(x, \xi) = g(x) \quad \text{on} \quad \partial D \times I^d, \tag{47}$$

where $D \subset \mathbb{R}^2$, and $u(x, \xi) = [u_1(x, \xi), u_2(x, \xi)]^T$ and $p(x, \xi)$ are the flow velocity and the scalar pressure respectively. We consider the problem with uncertain viscosity $a(x, \xi)$, which is assumed to be a random field with mean function $a_0(x) = 1$, variance $\sigma^2 = 0.25$, and covariance function (42). The correlation length is set to $l_c = 0.8$, and we take $d = 48$ to capture 95% of the total variance as in test problem 1. We here consider the driven cavity flow problem posed on $D = (0, 1) \times (0, 1)$. For boundary conditions, the velocity profile $u = [1, 0]^T$ is imposed on the top boundary ($x_2 = 1$ where $x = [x_1, x_2]^T$), and $u = [0, 0]^T$ is imposed on all other boundaries. We discretize in space using the inf-sup stable $\mathbf{Q}_2 - \mathbf{P}_{-1}$ mixed finite element method (biquadratic velocity–linear discontinuous pressure [33]) as

(a) Validation errors w.r.t CP ranks   (b) Validation errors at $R = 1$ (c) Validation errors at $R = 2$

(d) Validation errors at $R = 3$ (e) Validation errors at $R = 4$ (f) Validation errors at $R = 7$
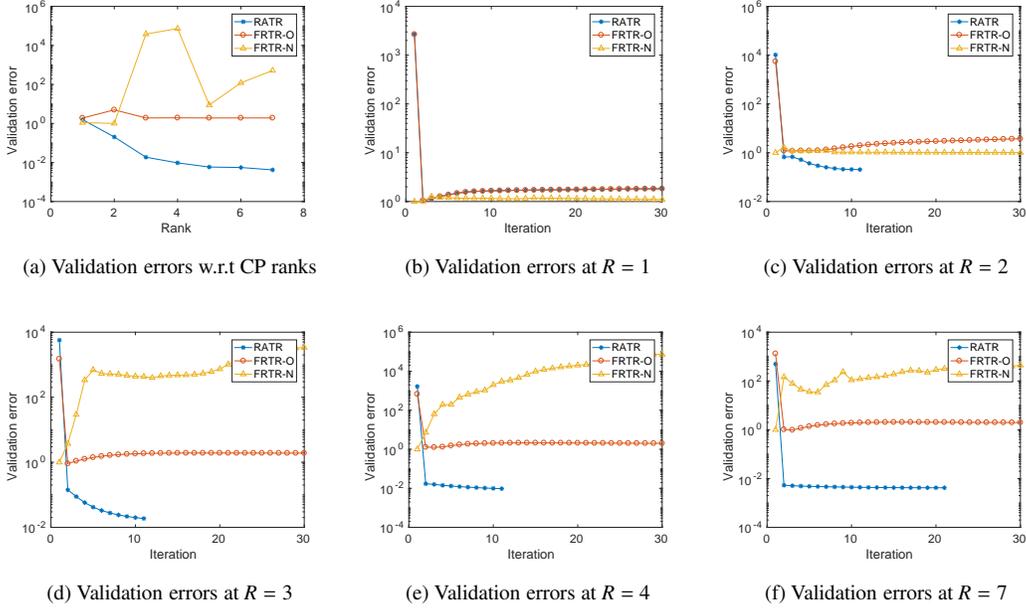
Figure 6: Validation errors of rank adaptive tensor recovery (RATR), fixed-rank tensor recovery with initial factor matrices generated through U(1, 2) (FRTR-O), and fixed-rank tensor recovery with initial factor matrices generated through N(0, 1) (FRTR-N), test problem 2.

implemented in IFISS [34] with a uniform 33×33 grid, which yields the velocity degrees of freedom $N_{h,u} = 2178$ and the pressure degrees of freedom $N_{h,p} = 768$. The output y here is defined to be a vector collecting both discrete velocity and pressure solutions, and the overall dimension of y is then $N_h = N_{h,u} + N_{h,p} = 2946$. For this test problem, the regularization parameter β in (21) is set to 0.1, and the tolerance in Algorithm 2 is set to $\delta = 10^{-5}$. The bandwidth σ_g of kPCA for dimension reduction is set to 10, and we again set tol_{PCA} to 90%.

We first generate the corresponding data matrix Y for the three cases of Θ ($|\Theta| = 100, 300$ and 600) and apply kPCA on it. For $tol_{\text{PCA}} = 90\%$, our results show that the number of kPCA modes retained is $N_r = 9$ for the case $|\Theta| = 100$, while $N_r = 10$ for the cases $|\Theta| = 300$ and $|\Theta| = 600$, which implies that the sample size 100 may not be large enough for an accurate dimension reduction. Tabel 4 shows the estimated CP ranks of $\mathbf{X}_e$ generated through Algorithm 3 for each kPCA mode $e = 1, \ldots, N_r$. Again, it is clear that, the estimated ranks are small—the maximum estimated rank is only seven. This shows that the rank-one update produced in Algorithm 3 is performed seven times at most, and it is therefore not costly.

Figure 9 shows the validation errors of our RATR (Algorithm 3 with initial factor matrices generated through U(1, 2)), FRTR-O and FRTR-N (Algorithm 2 with initial factor matrices generated through U(1, 2) and N(0, 1) respectively) for recovering $\mathbf{X}_1$ (see (10)) with $|\Theta| = 600$. From Figure 9(a), it can be seen that our RATR has the smallest validation error for each rank. It is also clear that, as the rank increases from one to three, the errors of RATR reduces significantly, while the iterations from rank three to seven are caused by our stopping criterion on line 9 of Algorithm 3. The other pictures in Figure 9 show the validation errors at each iteration step of the alternative minimization iterative procedure for $R = 1, 2, 3, 6, 7$ (while the errors for $R = 4, 5$ are similar to the errors for $R = 6$, they are not shown here). Similarly to test problem 1, for the case $R = 1$, RATR is the same as FRTR-O, and their errors are smaller

(a) 1st kPCA mode

(b) 2nd kPCA mode

(c) 3rd kPCA mode
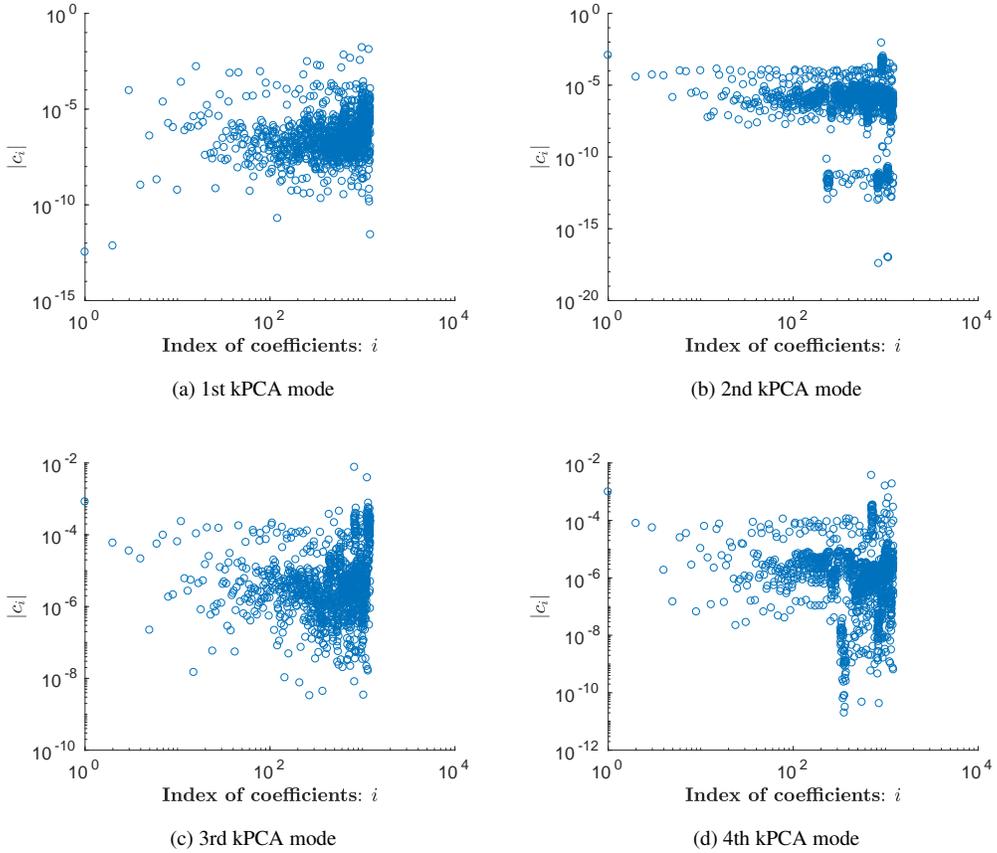
(d) 4th kPCA mode

Figure 7: Sparsity of gPC coefficients for each kPCA mode, test problem 2.
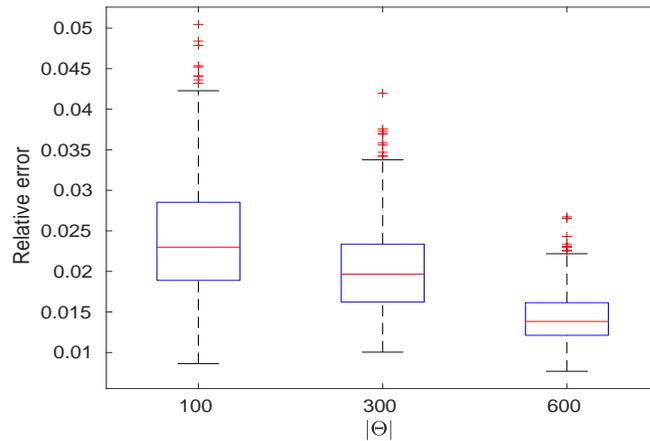
Figure 8: Relative errors of RATR-collocation approximation for 500 test samples, test problem 2.

than the error of FRTR-N. For $R = 2, 3, 6, 7$, the validation error of RATR is again clearly smaller than the errors of FRTR-O and FRTR-N, which shows that our rank-one update procedure is efficient for this Stokes problem.

Figure 10 shows the absolute values of the gPC coefficients of the first four kPCA modes for this test problem. It

is clear that absolute values of most gPC coefficients are small, and the gPC expansions for these four kPCA modes are therefore sparse. For the other kPCA modes ($e = 5, 6, 7$), while the situation is similar (the corresponding gPC expansions are also clearly sparse), they are not shown here. Figure 11 shows the flow streamlines and the pressure fields generated by the mixed finite element method and RATR-collocation (see section 4.3) responding to a given realization of $\boldsymbol{\xi}$. It can be seen that there is no visual difference between the results obtained through finite elements and RATR-collocation. Finally, we generate 500 samples of $\boldsymbol{\xi}$ and the compute the relative errors (44). Figure 8 shows Tukey box plots of these errors, where the central line in each box is the median, the lower and the upper edges are the first and the third quartiles respectively. It is clear that, as the size of the observation index set ($|\Theta|$) increases, values of the median, the first and the third quartiles of the errors decrease, which are all consistent with the results of the diffusion test problems.

Table 4: Estimated CP ranks of each data tensor $\mathbf{X}_e$ for $e = 1, \ldots, 10$, test problem 3.

rank \ e	1	2	3	4	5	6	7	8	9	10
$|\Theta|$										
100	3	5	7	1	2	2	4	1	3	–
300	7	3	2	1	1	5	5	2	2	2
600	7	7	7	4	2	2	2	1	4	7

(a) Validation errors w.r.t CP ranks

(b) Validation errors at $R = 1$

(c) Validation errors at $R = 2$

(d) Validation errors at $R = 3$

(e) Validation errors at $R = 6$

(f) Validation errors at $R = 7$
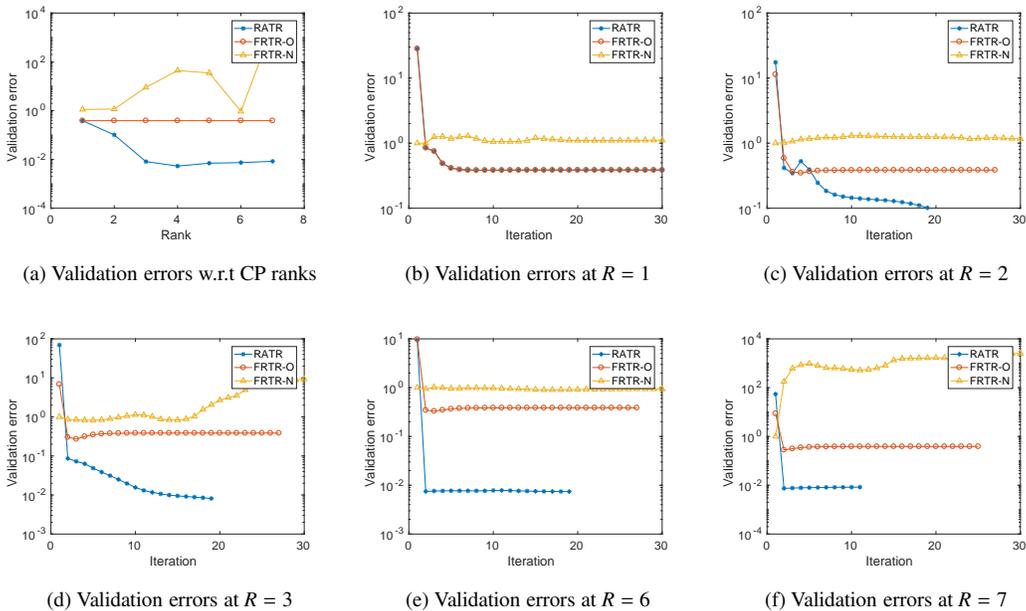
Figure 9: Validation errors of rank adaptive tensor recovery (RATR), fixed-rank tensor recovery with initial factor matrices generated through U(1, 2) (FRTR-O), and fixed-rank tensor recovery with initial factor matrices generated through N(0, 1) (FRTR-N), test problem 3.

(a) 1st kPCA mode

(b) 2nd kPCA mode

(c) 3rd kPCA mode
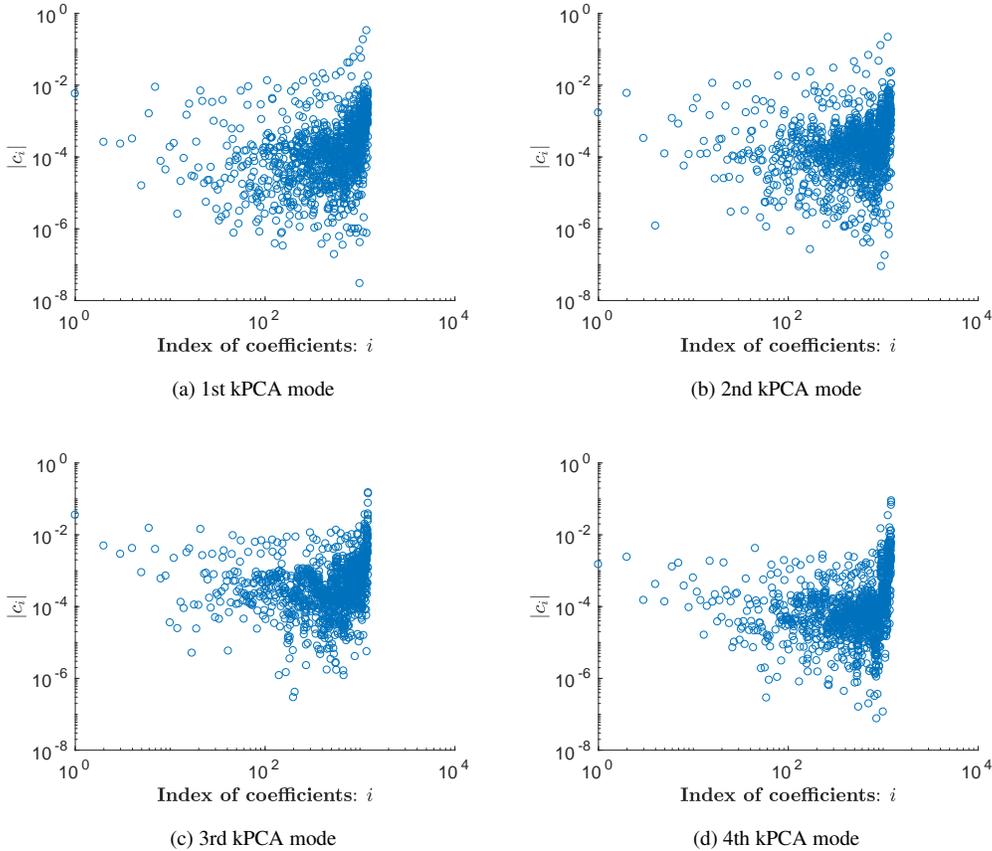
(d) 4th kPCA mode

Figure 10: Sparsity of gPC coefficients for each kPCA mode, test problem 3.

6. Conclusions

Exploiting potential low-dimensional structures is a fundamental concept of efficient surrogate and reduced order modelling for high-dimensional UQ problems. With a focus on the tensor recovery based stochastic collocation, our main conclusion is that our rank adaptive tensor recovery collocation (RATR-collocation) approach can efficiently exploit low-dimensional structures in this challenging problem in two aspects: first, we reformulate stochastic colocation based on manifold learning, where nonlinear low-dimensional structures in the snapshots are captured through kPCA; second, our novel RATR algorithm automatically explores the low-rank structures in the data tensors for computing the collocation coefficients without requiring a given tensor rank. Moreover, another main contribution of this work is the analysis of RATR, where the stability of our initialization strategies and the rank-one update procedure for the non-convex optimization problems involved is proven theoretically, such that a systematic guidance to initialize the the factor matrices is provided to result in efficient and stable recovery results. As the performance of RATR algorithm depends on the CP rank of the data tensor (although it does not need to be explicitly given), our RATR-collocation is efficient when the CP rank is small, while it may not be efficient for high-rank problems. A possible solution for efficiently recovering high-rank tensors is to conduct adaptivity with respect to physical properties of the underlying PDE models, e.g., domain decomposition methods. Designing and analyzing such strategies will be the focus of our
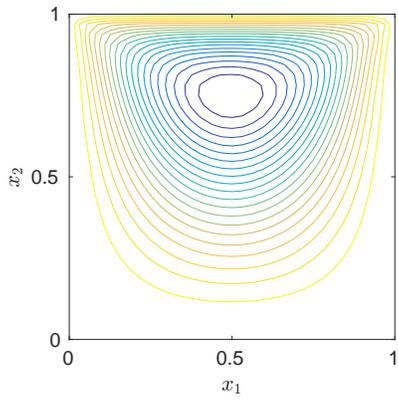
(a) Streamline: finite element solution
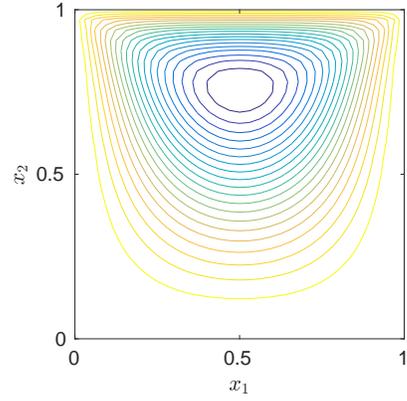
(b) Streamline: RATR-collocation approximation
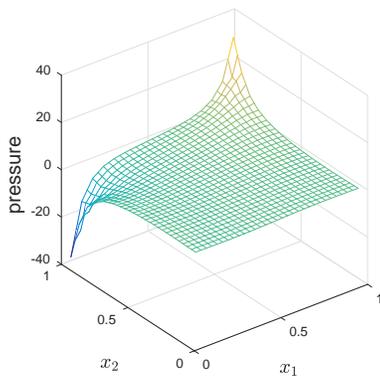
(c) Pressure: finite element solution
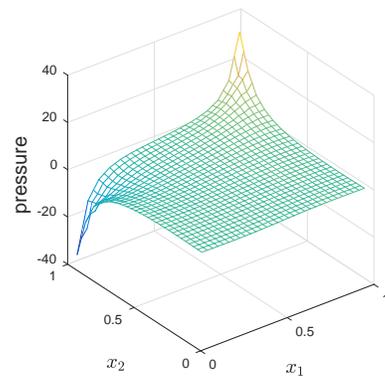
(d) Pressure: RATR-collocation approximation

Figure 11: The finite element solution and the RATR-collocation approximation (with $|\Theta| = 600$) responding to a given realization of ξ, test problem 3.
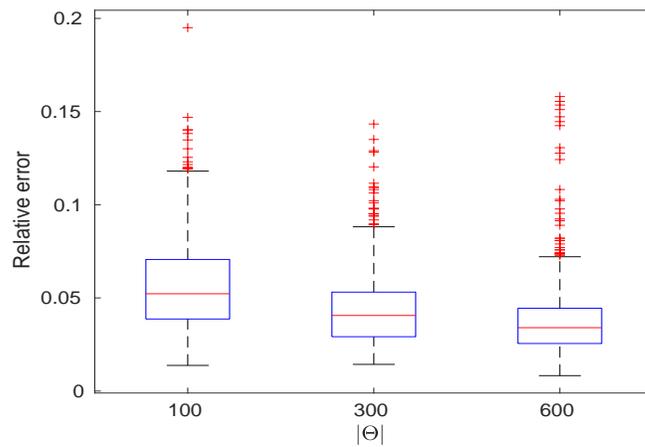
Figure 12: Relative errors of RATR-collocation approximation for 500 test samples, test problem 3.

future work.

References

References

[1] R. G. Ghanem, P. D. Spanos, Stochastic Finite Elements: A Spectral Approach, Courier Corporation, 2003.

[2] D. Xiu, G. E. Karniadakis, The wiener–askey polynomial chaos for stochastic differential equations, SIAM journal on scientific computing 24 (2) (2002) 619–644.

[3] D. Xiu, J. S. Hesthaven, High-order collocation methods for differential equations with random inputs, SIAM Journal on Scientific Computing 27 (3) (2005) 1118–1139.

[4] X. Ma, N. Zabaras, An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations, Journal of Computational Physics 228 (8) (2009) 3084–3113.

[5] C. Powell, H. Elman, Block-diagonal preconditioning for spectral stochastic finite-element systems, IMA Journal of Numerical Analysis 29 (2009) 350–375.

[6] A. Doostan, H. Owhadi, A non-adapted sparse approximation of pdes with stochastic inputs, Journal of Computational Physics 230 (8) (2011) 3015–3034.

[7] J. Peng, J. Hampton, A. Doostan, A weighted l_1-minimization approach for sparse polynomial chaos expansions, Journal of Computational Physics 267 (2014) 92–111.

[8] L. Yan, L. Guo, D. Xiu, Stochastic collocation algorithms using l_1-minimization, International Journal for Uncertainty Quantification 2 (3).

[9] J. D. Jakeman, A. Narayan, T. Zhou, A generalized sampling and preconditioning scheme for sparse approximation of polynomial chaos expansions, SIAM Journal on Scientific Computing 39 (3) (2017) A1114–A1144.

[10] H. Lei, X. Yang, B. Zheng, G. Lin, N. A. Baker, Constructing surrogate models of complex systems with enhanced sparsity: quantifying the influence of conformational uncertainty in biomolecular solvation, Multiscale Modeling & Simulation 13 (4) (2015) 1327–1353.

[11] L. Guo, A. Narayan, T. Zhou, A gradient enhanced l_1-minimization for sparse approximation of polynomial chaos expansions, Journal of Computational Physics 367 (2018) 49–64.

[12] I. Babuška, F. Nobile, R. Tempone, A stochastic collocation method for elliptic partial differential equations with random input data, SIAM Journal on Numerical Analysis 45 (3) (2007) 1005–1034.

[13] Z. Zhang, T.-W. Weng, L. Daniel, Big-data tensor recovery for high-dimensional uncertainty quantification of process variations, IEEE Transactions on Components, Packaging and Manufacturing Technology 7 (5) (2017) 687–697.

[14] S. Conti, A. OHagan, Bayesian emulation of complex multi-output and dynamic computer models, Journal of statistical planning and inference 140 (3) (2010) 640–651.

[15] W. Xing, V. Triantafyllidis, A. Shah, P. Nair, N. Zabaras, Manifold learning for the emulation of spatial fields from computational models, Journal of Computational Physics 326 (2016) 666–690.

[16] X. Ma, N. Zabaras, Kernel principal component analysis for stochastic input model generation, Journal of Computational Physics 230 (19) (2011) 7311–7331.

[17] E. Acar, D. M. Dunlavy, T. G. Kolda, M. Mrup, Scalable tensor factorizations for incomplete data , Chemometrics & Intelligent Laboratory Systems 106 (1) (2010) 41–56.

[18] S. Gandy, B. Recht, I. Yamada, Tensor completion and low-n-rank tensor recovery via convex optimization, Inverse Problems 27 (2) (2011) 025010.

[19] J. Liu, P. Musialski, P. Wonka, J. Ye, Tensor completion for estimating missing values in visual data., in: IEEE International Conference on Computer Vision, 2013, pp. 2114–2121.

[20] R. Vidal, Y. Ma, S. S. Sastry, Generalized principal component analysis, Vol. 5, Springer, 2016.

[21] J.-Y. Kwok, I.-H. Tsang, The pre-image problem in kernel methods, IEEE transactions on neural networks 15 (6) (2004) 1517–1525.

[22] B. Schölkopf, A. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural computation 10 (5) (1998) 1299–1319.

[23] W. Gautschi, On generating orthogonal polynomials, SIAM Journal on Scientific and Statistical Computing 3 (3) (1982) 289–317.

[24] L. D. Lathauwer, B. D. Moor, J. Vandewalle, A multilinear singular value decomposition, SIAM Journal on Matrix Analysis & Applications 21 (4) (2000) 1253–1278.

[25] T. G. Kolda, B. W. Bader, Tensor decompositions and applications, SIAM Review 51 (3) (2009) 455–500.

[26] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al., Distributed optimization and statistical learning via the alternating direction method of multipliers, Foundations and Trends® in Machine learning 3 (1) (2011) 1–122.

[27] D. P. Bertsekas, Nonlinear programming, Athena scientific Belmont, 1999.

[28] S. Arlot, A. Celisse, et al., A survey of cross-validation procedures for model selection, Statistics surveys 4 (2010) 40–79.

[29] M. Steinlechner, Riemannian optimization for high-dimensional tensor completion, SIAM Journal on Scientific Computing 38 (5) (2016) 461–484.

[30] W. Deng, W. Yin, On the global and linear convergence of the generalized alternating direction method of multipliers, Journal of Scientific Computing 66 (3) (2016) 889–916.

[31] M. H. DeGroot, M. J. Schervish, Probability and statistics, Pearson Education, 2012.

[32] J.-P. Ryckaert, G. Ciccotti, H. Berendsen, Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes, Journal of Computational Physics 23 (3) (1977) 327–341.

[33] H. Elman, D. Silvester, A. Wathen, Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics, Oxford University Press (UK), 2014.

[34] H. Elman, A. Ramage, D. Silvester, IFISS: A computational laboratory for investigating incompressible flow problems, SIAM Review 56 (2014) 261–273.