Algebraic PGD for tensor separation and compression: an algorithmic approach

Pedro Díez, Sergio Zlotnik, Alberto García-González, and Antonio Huerta

Laboratori de Càlcul Numèric,
E.T.S. de Ingenieros de Caminos, Canales y Puertos,
Universitat Politècnica de Catalunya – (UPC BarcelonaTech),
Jordi Girona 1, E-08034 Barcelona, Spain

Abstract

The Proper Generalized Decomposition (PGD) is devised as a computational method to solve high-dimensional boundary value problems (where many dimensions are associated with the space of parameters defining the problem). The PGD philosophy consists in providing a separated representation of the multidimensional solution using a greedy approach combined with an alternated directions scheme to obtain the successive rank-one terms. This paper presents an algorithmic approach to high-dimensional tensor separation based on solving the Least Squares approximation in a separable format of multidimensional tensor using PGD. This strategy is usually embedded in a standard PGD code in order to compress the solution (reduce the number of terms and optimize the available storage capacity) but it stands also as an alternative and highly competitive method for tensor separation.

Key words: tensor separation, algebraic PGD, least-squares approximation

1. Introduction

1.1. *Framework and motivation*

Data is often collected in terms of multidimensional arrays. The number of dimensions is denoted by d and the object containing the information is a tensor $\boldsymbol{F}$ of order d. A multi-index notation identifies each entry of the array as corresponding to specific values of d parameters. The size of the tensor in each

Email addresses: pedro.diez@upc.edu (Pedro Díez), sergio.zlotnik@upc.edu (Sergio Zlotnik), berto.garcia@upc.edu (Alberto García-González), antonio.huerta@upc.edu (Antonio Huerta).

dimension is denoted by $\mathtt{n}_i$, for $i = 1, 2, \ldots, d$ and $\boldsymbol{F}$ is characterized by the generic term $[\boldsymbol{F}]_{j_1 j_2 \ldots j_d}$ for $j_i = 1, 2, \ldots, \mathtt{n}_i$.

Tensor separation is a generalization of matrix diagonalization and allows representing the data collected in the tensor in a compact form. The separated form is expressed as the sum of M terms, each of them consisting in the tensorial product of d vectors of dimensions $\mathtt{n}_i$, $i = 1, 2, \ldots, d$. Note that only in terms of storage, the number of entries of $\boldsymbol{F}$ is $\prod_{i=1}^{d} \mathtt{n}_i$ and in the separable version it is described with $M \sum_{i=1}^{d} \mathtt{n}_i$ scalar quantities. For *small* values of M, this represents a huge reduction of the information to be stored.

In the following, the methodology to perform this separation is presented first in the simple case of $d = 2$, where it can be done optimally. Then, the generalization to higher dimensions, in which there is no optimal strategy, is devised using the PGD philosophy.

1.2. *SVD and matrix separation*

For $d = 2$, $\boldsymbol{F}$ is a matrix and the Singular Value Decomposition (SVD) (and matrix diagonalization as a particular case for squared matrices) is the standard and optimal tool to obtain a reduced representation of a 2D array (a matrix or second order tensor). The outcome of the SVD allows representing the matrix as a sum of tensor products (rank-one matrices) of the left and right eigenvectors, each weighted by the corresponding eigenvalue.

Namely, the SVD of $\boldsymbol{F} \in \mathbb{R}^{n_1 \times n_2}$ consists in finding square unit matrices $\boldsymbol{U} \in \mathbb{R}^{n_1 \times n_1}$ and $\boldsymbol{V} \in \mathbb{R}^{n_2 \times n_2}$ such that

$$\boldsymbol{F} = \boldsymbol{U} \Sigma \boldsymbol{V}^{\mathsf{T}} \tag{1}$$

being Σ a diagonal matrix in $\mathbb{R}^{n_1 \times n_2}$ containing the singular values of $\boldsymbol{F}$. That is, $\boldsymbol{U}$ and $\boldsymbol{V}$ are such that $\boldsymbol{U}^{\mathsf{T}}\boldsymbol{U} = \mathbb{I}_{n_1}$ and $\boldsymbol{V}^{\mathsf{T}}\boldsymbol{V} = \mathbb{I}_{n_2}$, and Σ has the format

$$\Sigma = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_{n_2} \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \end{bmatrix} \text{ for } n_1 > n_2 \text{ or } \Sigma = \begin{bmatrix} \sigma_1 & & & 0 \ldots 0 \\ & \sigma_2 & & 0 \ldots 0 \\ & & \ddots & 0 \ldots 0 \\ & & \sigma_{n_1} & 0 \ldots 0 \end{bmatrix} \text{ for } n_1 < n_2.$$

It is assumed that the singular values are sorted in decreasing order, that is

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(n_1,n_2)} \geq 0.$$

The column vectors of matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are denoted by $\boldsymbol{u}_j$ and $\boldsymbol{v}_k$, for $j = 1, 2, \ldots, n_1$ and $j = 1, 2, \ldots, n_1$, that is such that

$$\boldsymbol{U} = [\boldsymbol{u}_1\, \boldsymbol{u}_2 \ldots \boldsymbol{u}_{n_1}] \text{ and } \boldsymbol{V} = [\boldsymbol{v}_1\, \boldsymbol{v}_2 \ldots \boldsymbol{v}_{n_1}]$$

Thus, the SVD in (1) is rewritten as

$$\boldsymbol{F} = \sum_{j=1}^{\min(n_1,n_2)} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^{\mathsf{T}} = \sum_{j=1}^{\min(n_1,n_2)} \sigma_j \boldsymbol{u}_j \otimes \boldsymbol{v}_j. \tag{2}$$

Note that the two notations in the equation above are equivalent because each term $\boldsymbol{u}_j \boldsymbol{v}_j^{\mathsf{T}}$ is a rank-one $n_1 \times n_2$ matrix that can also be denoted as $\boldsymbol{u}_j \otimes \boldsymbol{v}_j$ using the tensor (or external) product. The latter

notation is preferred in the following because it is easily extended to problems of dimension higher than two.

The representation displayed in (2) is particularly attractive to obtain a Least-Squares (LS) low rank approximation of $\boldsymbol{F}$. The Frobenius matrix norm reads

$$\|\boldsymbol{F}\|^2 = \boldsymbol{F} : \boldsymbol{F} = [\boldsymbol{F}]_{j_1 j_2} [\boldsymbol{F}]_{j_1 j_2} \tag{3}$$

Using this *Euclidean-type* norm, the truncation of (2) to M terms (being $M < \min(n_1, n_2)$) results in the best LS rank-M approximation, that is

$$\sum_{j=1}^{M} \sigma_j \boldsymbol{u}_j \otimes \boldsymbol{v}_j = \arg \min_{\boldsymbol{A} \in \mathbb{R}^{n_1 \times n_2} \text{ of rank} M} \|\boldsymbol{F} - \boldsymbol{A}\| \tag{4}$$

This LS optimality is associated with the orthogonality of the left and right eigenvectors.

In particular, the best rank-one approximation of $\boldsymbol{F}$ is $\sigma_1 \boldsymbol{u}_1 \otimes \boldsymbol{v}_1$. This is easily shown by noting that for any rank-one matrix $\boldsymbol{w}^1 \otimes \boldsymbol{w}^2$ where $\boldsymbol{w}^1 = \sum_{k=1}^{n_1} \alpha_k \boldsymbol{u}_k$ and $\boldsymbol{w}^2 = \sum_{\ell=1}^{n_2} \beta_\ell \boldsymbol{v}_\ell$, LS discrepancy with $\boldsymbol{F}$ reads

$$\|\boldsymbol{F} - \boldsymbol{w}^1 \otimes \boldsymbol{w}^2\|^2 = \|\Sigma - \boldsymbol{\alpha} \otimes \boldsymbol{\beta}\|^2 \tag{5}$$

being $\boldsymbol{\alpha} \in \mathbb{R}^{n_1}$ and $\boldsymbol{\beta} \in \mathbb{R}^{n_2}$ the vectors representing $\boldsymbol{w}^1$ and $\boldsymbol{w}^2$ in the bases of left and right eigenvectors ($\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{n_1}\}$ and $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{n_2}\}$, respectively). It is clear from the right-hand-side of (5) that an optimal choice for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ is $\alpha_k = \beta_k = \delta_{k1}\sqrt{\sigma_k}$.

Remark 1 *The equality* (5) *is a direct consequence of observing that*

$$\|\boldsymbol{F} - \boldsymbol{w}^1 \otimes \boldsymbol{w}^2\|^2 = \|\boldsymbol{F}\|^2 + \|\boldsymbol{w}^1 \otimes \boldsymbol{w}^2\|^2 - 2\boldsymbol{F} : \boldsymbol{w}^1 \otimes \boldsymbol{w}^2,$$

$$\|\boldsymbol{F}\|^2 = \sum_{k=1}^{\min(n_1,n_2)} \sigma_k^2 = \|\Sigma\|^2 \ , \qquad \|\boldsymbol{w}^1 \otimes \boldsymbol{w}^2\|^2 = \sum_{k=1}^{n_1}\sum_{\ell=1}^{n_2} \alpha_k^2 \beta_\ell^2 = \|\boldsymbol{\alpha} \otimes \boldsymbol{\beta}\|^2$$

and

$$\boldsymbol{F} : \boldsymbol{w}^1 \otimes \boldsymbol{w}^2 = \sum_{k=1}^{n_1}\sum_{\ell=1}^{n_2} \alpha_k \beta_\ell \boldsymbol{F} : (\boldsymbol{u}_k \otimes \boldsymbol{v}_\ell) = \sum_{k=1}^{\min(n_1,n_2)} \alpha_k \beta_k \sigma_k = \Sigma : \boldsymbol{\alpha} \otimes \boldsymbol{\beta}$$

These properties are due to the orthonormality of the bases $\{\boldsymbol{u}_j\}_{j=1,\ldots,n_1}$ *and* $\{\boldsymbol{v}_j\}_{j=1,\ldots,n_2}$ *that has as a direct consequence*

$$\boldsymbol{F} : (\boldsymbol{u}_k \otimes \boldsymbol{v}_\ell) = \left[\sum_j \sigma_j \boldsymbol{u}_j \otimes \boldsymbol{v}_j\right] : (\boldsymbol{u}_k \otimes \boldsymbol{v}_\ell) = \sum_j \sigma_j (\boldsymbol{u}_j \cdot \boldsymbol{u}_k)(\boldsymbol{v}_j \cdot \boldsymbol{v}_\ell) = \delta_{k\ell}\sigma_k$$

The generalization of these tools to high order tensor formats is the object of extensive research activities and different strategies are labeled as High-Order SVD (HOSVD). Nevertheless, for a number of dimensions larger than two the orthogonality among the eigenvectors and therefore the optimality of the separation is no longer guaranteed.

1.3. *Problem statement: High-order separation*

For $d > 2$, tensor $\boldsymbol{F} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is expressed in a separable format if for some integer value M, and for $i = 1, 2, \ldots, d$ and $m = 1, 2, \ldots, M$, there exist a set of vectors $\tilde{\boldsymbol{f}}_i^m \in \mathbb{R}^{n_i}$ such that

$$\boldsymbol{F} = \sum_{m=1}^{M} \tilde{\boldsymbol{f}}_1^m \otimes \tilde{\boldsymbol{f}}_2^m \otimes \cdots \otimes \tilde{\boldsymbol{f}}_d^m \tag{6}$$

An alternative version of this format uses unit vectors $\boldsymbol{f}_i^m$ and amplitudes σ_m, for $i = 1, \ldots, d$ and $m = 1, \ldots, M$

$$\boldsymbol{F} = \sum_{m=1}^{M} \sigma_m \boldsymbol{f}_1^m \otimes \boldsymbol{f}_2^m \otimes \cdots \otimes \boldsymbol{f}_d^m \tag{7}$$

Note that (6) and (7) are readily shown to be equivalent by taking

$$\boldsymbol{f}_i^m = \frac{1}{\|\tilde{\boldsymbol{f}}_i^m\|} \tilde{\boldsymbol{f}}_i^m \text{ for } i = 1, \ldots, d \text{ and } \sigma_m = \prod_{i=1}^{d} \|\boldsymbol{f}_i^m\| \text{ for } m = 1, \ldots, M$$

1.4. *Standard approaches*

The problem of finding a separable expression with the form of (6) or (7) that approximates a tensor $\boldsymbol{F}$ has a wide range of applications in many engineering and scientific fields. The pioneering works appeared in the field of psicometrics, see for example the work of Tucker [1] and Harshman [2], and in the field of chemometrics [3,4]. Later, the interest of tensor decompositions reached many different scientific communities such as neuroscience [5,6], computer vision [7,8], signal processing [9,10] and data mining [11]. Numerical analysis is no exception and examples of applications on tensor decompositions include [12,13]. This list is by no means exhaustive, for a comprehensive review on tensor decomposition methods see [14].

2. PGD strategy and rank-one algorithm

The PGD strategy aims at obtaining an approximation of some tensor $\boldsymbol{F}$ in the separable form given by (7). The idea is extensively described in [15–18] and references therein and consists in combining a *greedy* algorithm (that is, computing first for $m = 1$ and compute sequentially for any m when the previous $m - 1$ terms are already obtained) with an *alternated directions* scheme to solve the series of rank-one problems corresponding to term m.

2.1. *Greedy scheme and rank-one terms*

The first step in the greedy algorithm is to find a rank-one approximation of $\boldsymbol{F}$, namely find d unit vectors $\boldsymbol{f}_1^1, \ldots, \boldsymbol{f}_d^1$ and the corresponding amplitude σ_1 such that

$$\boldsymbol{F} \approx \sigma_1 \boldsymbol{f}_1^1 \otimes \boldsymbol{f}_2^1 \otimes \cdots \otimes \boldsymbol{f}_d^1$$

or, equivalently,

$$\boldsymbol{F} \approx \tilde{\boldsymbol{f}}_1^1 \otimes \tilde{\boldsymbol{f}}_2^1 \otimes \cdots \otimes \tilde{\boldsymbol{f}}_d^1. \tag{8}$$

Also at this stage, the LS criterion is used to qualify the best rank-one approximation, following the ideas draft in the appendix of [18] describing the PGD compression strategy. That is, the d vectors $\tilde{\boldsymbol{f}}_1^1, \ldots, \tilde{\boldsymbol{f}}_d^1$ are sought such that they minimize the scalar functional $\mathcal{J}(\cdot)$ defined by

$$\mathcal{J}(\tilde{\boldsymbol{f}}_1^1, \ldots, \tilde{\boldsymbol{f}}_d^1) = \|\boldsymbol{F} - \tilde{\boldsymbol{f}}_1^1 \otimes \tilde{\boldsymbol{f}}_2^1 \otimes \cdots \otimes \tilde{\boldsymbol{f}}_d^1\|. \tag{9}$$

Note that in this context, analogously to (3) the Frobenius-type norm for tensors reads

$$\|\boldsymbol{F}\|^2 = [\boldsymbol{F}]_{j_1 j_2 \ldots j_d} [\boldsymbol{F}]_{j_1 j_2 \ldots j_d} = \sum_{j_1=1}^{n_1} \sum_{j_2=1}^{n_2} \cdots \sum_{j_d=1}^{n_d} [\boldsymbol{F}]^2_{j_1 j_2 \ldots j_d}\,. \tag{10}$$

That is,

$$\left(\tilde{\boldsymbol{f}}_1^1, \ldots, \tilde{\boldsymbol{f}}_d^1\right) = \arg \min_{\mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_d}} \mathcal{J}(\tilde{\boldsymbol{f}}_1^1, \ldots, \tilde{\boldsymbol{f}}_d^1) \tag{11}$$

Functional $\mathcal{J}(\cdot)$ is nonlinear and therefore also the problem given in (11) is nonlinear and requires devising an iterative solver. Note also that the set of possible solutions (constituted by all the rank-one tensors) is not provided with the structure of a linear vectorial space. Obviously, the sum of two rank-one tensors is, in general, a tensor of rank two.

The algorithm proposed to solve problem (11) is detailed in section 2.2. Once the solution of (11) is available, this is taken as the first term of the PGD approximation, that is

$$\boldsymbol{F}_{\text{PGD}}^1 = \tilde{\boldsymbol{f}}_1^1 \otimes \tilde{\boldsymbol{f}}_2^1 \otimes \cdots \otimes \tilde{\boldsymbol{f}}_d^1. \tag{12}$$

From this point on, the algorithm is recursive and obtains the best approximation of the remainder part of $\boldsymbol{F}$. Namely, assuming that $\boldsymbol{F}_{\text{PGD}}^{M-1}$ is available

$$\boldsymbol{F}_{\text{PGD}}^{M-1} = \sum_{m=1}^{M-1} \tilde{\boldsymbol{f}}_1^m \otimes \tilde{\boldsymbol{f}}_2^m \otimes \cdots \otimes \tilde{\boldsymbol{f}}_d^m,$$

the next term is obtained as

$$\boldsymbol{F}_{\text{PGD}}^{M} = \boldsymbol{F}_{\text{PGD}}^{M-1} + \tilde{\boldsymbol{f}}_1^M \otimes \tilde{\boldsymbol{f}}_2^M \otimes \cdots \otimes \tilde{\boldsymbol{f}}_d^M,$$

and

$$\left(\tilde{\boldsymbol{f}}_1^M, \ldots, \tilde{\boldsymbol{f}}_d^M\right) = \arg \min_{\mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_d}} \|\boldsymbol{F} - \boldsymbol{F}_{\text{PGD}}^{M-1} - \tilde{\boldsymbol{f}}_1^M \otimes \tilde{\boldsymbol{f}}_2^M \otimes \cdots \otimes \tilde{\boldsymbol{f}}_d^M\|. \tag{13}$$

Note that (11) and (13) have exactly the same structure, they are both rank-one least squares approximation problems and therefore the same iterative algorithm devised for (11) is going to be used for (13).

The stopping criteria used to decide wether the number of PGD terms, M is sufficiently large are mainly based on characterizing the relative importance of the last term added to the sum. An alternative approach is computing the residual $\|\boldsymbol{F} - \boldsymbol{F}_{\text{PGD}}^M\|$, but this is often discarded because it requires reconstructing $\boldsymbol{F}_{\text{PGD}}^M$ as a multidimensional tensor and this has a large computational cost.

Recall that using normalized vectors, the expression for the PGD solution reads

$$\boldsymbol{F}_{\text{PGD}}^M = \sum_{m=1}^{M} \sigma_m \boldsymbol{f}_1^m \otimes \boldsymbol{f}_2^m \otimes \cdots \otimes \boldsymbol{f}_d^m. \tag{14}$$

Typically, M is considered to be large enough if, for some tolerance $\eta^\star$, the following inequality holds

$$\sigma_M < \eta^\star \sigma_1.$$

That is, the greedy algorithm is stopped when the amplitude σ_M of the last term is significantly lower than the amplitude of the first one.

2.2. *Alternated directions scheme: iterating in sectional problems*

This section is devoted to describe the iterative algorithm devised to solve the rank-one problem (11) (or (13)). Thus, the goal is to compute $\left(\tilde{\boldsymbol{f}}_1^1, \ldots, \tilde{\boldsymbol{f}}_d^1\right)$ minimizing the functional $\mathcal{J}(\tilde{\boldsymbol{f}}_1^1, \ldots, \tilde{\boldsymbol{f}}_d^1)$ given by (9). The idea is to follow an alternated directions strategy, consisting in computing each of the *sectional* unknowns, say $\tilde{\boldsymbol{f}}_\gamma^1$ for $\gamma = 1, 2, \ldots, d$, assuming that all the others ($\tilde{\boldsymbol{f}}_j^1$ for $j \neq \gamma$) are known. This is done for $\gamma = 1, 2, \ldots, d$ and then iterated until convergence is reached.

The functional $\mathcal{J}(\cdot)$ is rewritten as:

$$\mathcal{J}(\tilde{\boldsymbol{f}}_1^1,\ldots,\tilde{\boldsymbol{f}}_d^1) = \|\boldsymbol{F}\|^2 + \prod_{j=1}^{d} \tilde{\boldsymbol{f}}_j^{1\,\mathsf{T}}\tilde{\boldsymbol{f}}_j^1 - 2\boldsymbol{F} : \tilde{\boldsymbol{f}}_1^1 \otimes \cdots \otimes \boldsymbol{f}_d^1 \tag{15}$$

where the symbol : must be understood here as total tensor contraction (summing up in all the indices) as corresponds to the expression of the norm described in (10).

Thus, in order to compute an approximation to $\tilde{\boldsymbol{f}}_\gamma^1$, it is assumed that the other modes, $\tilde{\boldsymbol{f}}_j^1$ for $j \neq \gamma$ are known and the functional $\mathcal{J}(\cdot)$ is to be minimized with respect to $\tilde{\boldsymbol{f}}_\gamma^1$. Namely

$$\mathcal{J}(\tilde{\boldsymbol{f}}_\gamma^1) = \|\boldsymbol{F}\|^2 + \alpha\tilde{\boldsymbol{f}}_\gamma^{1\,\mathsf{T}}\tilde{\boldsymbol{f}}_\gamma^1 - 2\tilde{\boldsymbol{f}}_\gamma^{1\,\mathsf{T}}\boldsymbol{g} \tag{16}$$

where the computable quantities α and $\boldsymbol{g}$ are

$$\alpha := \left(\prod_{j\neq\gamma}^{d} \tilde{\boldsymbol{f}}_j^{1\,\mathsf{T}}\tilde{\boldsymbol{f}}_j^1\right) \tag{17}$$

and

$$\boldsymbol{g} := \boldsymbol{F} \vdots \bigotimes_{j\neq\gamma}^{d} \tilde{\boldsymbol{f}}_j^1, \tag{18}$$

where symbol $\vdots$ indicates tensor contraction of all possible indices. In this case, provided that $\boldsymbol{F}$ is a tensor of d dimensions and $\bigotimes_{j\neq\gamma}^{d}\tilde{\boldsymbol{f}}_j^1$ is a tensor of $d-1$ dimensions, this means summing up in all indices i_j for $j = 1,\ldots,d$ with $j \neq \gamma$. That is, (18) is equivalent to

$$[\boldsymbol{g}]_{i_\gamma} := \sum_{i_1=1}^{n_1}\cdots\sum_{i_{\gamma-1}=1}^{n_{\gamma-1}}\sum_{i_{\gamma+1}=1}^{n_{\gamma+1}}\cdots\sum_{i_d=1}^{d}[\boldsymbol{F}]_{i_1\ldots i_{\gamma-1}i_\gamma i_{\gamma+1}\ldots i_d}\prod_{j\neq\gamma}^{d}\left[\tilde{\boldsymbol{f}}_j^1\right]_{i_j}. \tag{19}$$

Thus, the operation represented in (18) (in compact form) and (19) (with the complete index notation) consist in contracting all the dimensions of tensor $\boldsymbol{F}$ but one (the dimension γ, for $\gamma = 1,2,\ldots,d$) with the tensorial product of all the vectors $\tilde{\boldsymbol{f}}_j^1$, for $j \neq \gamma$.

The vector $\tilde{\boldsymbol{f}}_\gamma^1$ minimizing $\mathcal{J}(\tilde{\boldsymbol{f}}_\gamma^1)$ in (16) is precisely

$$\tilde{\boldsymbol{f}}_\gamma^1 = \frac{1}{\alpha}\boldsymbol{g} \tag{20}$$

This has to be done for all the dimensions, that is for $\gamma = 1,2,\ldots,d$ and iterated until the consecutive approximations of $\tilde{\boldsymbol{f}}_j^1$ reach stationarity. That is to say, assume each loop on γ is computing $\tilde{\boldsymbol{f}}_j^{\mathtt{new}}$ as an approximation to $\tilde{\boldsymbol{f}}_j^1$ starting from initial guesses $\tilde{\boldsymbol{f}}_j^{\mathtt{old}}$. Convergence is reached once, for some tolerance η, $\|\boldsymbol{f}_j^{\mathtt{new}} - \boldsymbol{f}_j^{\mathtt{old}}\| < \eta$ for all $j = 1,2,\ldots,d$. Note that the stationarity condition is expressed in terms of the normalized vectors $\boldsymbol{f}_j$ to avoid the possibility that arbitrary constants with unit product may affect the d terms. Note that the sectional errors $\|\boldsymbol{f}_j^{\mathtt{new}} - \boldsymbol{f}_j^{\mathtt{old}}\|$ are already relative because all the vectors are normalized. Moreover, in the practical implementation the global error taken into the account is the product of all the sectional norms that stands for the Frobenius norm of the multidimensional error tensor, namely

$$\varepsilon = \|\boldsymbol{F}_{\mathtt{PGD}}^{\mathtt{new}} - \boldsymbol{F}_{\mathtt{PGD}}^{\mathtt{old}}\| = \left[\prod_{j=1}^{d}(\boldsymbol{f}_j^{\mathtt{new}} - \boldsymbol{f}_j^{\mathtt{old}})^{\mathsf{T}}(\boldsymbol{f}_j^{\mathtt{new}} - \boldsymbol{f}_j^{\mathtt{old}})\right]^{1/2} \tag{21}$$

The strategy presented above for a rank-one approximation of some tensor $\boldsymbol{F}$ is summaryzed in Algorithm 1.

Data: Tensor of order d to be approximated: $\boldsymbol{F}$ with general term $[\boldsymbol{F}]_{i_1 i_2 \ldots i_d}$, (for $i_\gamma = 1, \ldots, n_\gamma$ and $\gamma = 1, \ldots, d$)
Result: Rank-one approximation: $\boldsymbol{F}_{\text{PGD}} = \sigma \boldsymbol{f}_1 \otimes \boldsymbol{f}_2 \otimes \cdots \otimes \boldsymbol{f}_d$
Initialize: assign values to $\boldsymbol{f}_i^{\text{old}}$, for $i = 1, 2, \ldots, d$; select a tolerance η

Compute $\sigma^{\text{old}} = \prod_{j=1}^{d} \left[\boldsymbol{f}_j^{\text{old}\,\mathsf{T}} \boldsymbol{f}_j^{\text{old}}\right]^{1/2}$;
Normalize: $\boldsymbol{f}_j^{\text{old}} \leftarrow \left[\boldsymbol{f}_j^{\text{old}\,\mathsf{T}} \boldsymbol{f}_j^{\text{old}}\right]^{-1/2} \boldsymbol{f}_j^{\text{old}}$, for $j = 1, 2, \ldots, d$;

while $\varepsilon > \eta$ *or* $\varepsilon_\sigma > \eta$ **do**
  for $\gamma = 1 \ldots d$ **do**
    Compute $\alpha = \prod_{j \neq \gamma} \boldsymbol{f}_j^{\text{old}\,\mathsf{T}} \boldsymbol{f}_j^{\text{old}}$
    Compute $\boldsymbol{g}$ such that
    $$[\boldsymbol{g}]_{i_\gamma} = \sum_{i_1=1}^{n_1} \cdots \sum_{i_{\gamma-1}=1}^{n_{\gamma-1}} \sum_{i_{\gamma+1}=1}^{n_{\gamma+1}} \cdots \sum_{i_d=1}^{d} [\boldsymbol{F}]_{i_1 \ldots i_{\gamma-1} i_\gamma i_{\gamma+1} \ldots i_d} \prod_{j \neq \gamma}^{d} \left[\tilde{\boldsymbol{f}}_j^1\right]_{i_j}$$
    Compute $\boldsymbol{f}_\gamma^{\text{new}} = \frac{1}{\alpha} \boldsymbol{g}$
  end
  Compute $\sigma^{\text{new}} = \prod_{j=1}^{d} \left[\boldsymbol{f}_j^{\text{new}\,\mathsf{T}} \boldsymbol{f}_j^{\text{new}}\right]^{1/2}$;
  Normalize: $\boldsymbol{f}_j^{\text{new}} \leftarrow \left[\boldsymbol{f}_j^{\text{new}\,\mathsf{T}} \boldsymbol{f}_j^{\text{new}}\right]^{-1/2} \boldsymbol{f}_j^{\text{new}}$, for $j = 1, 2, \ldots, d$;
  Compute modal error $\varepsilon = \left[\prod_{j=1}^{d} (\boldsymbol{f}_j^{\text{new}} - \boldsymbol{f}_j^{\text{old}})^{\mathsf{T}} (\boldsymbol{f}_j^{\text{new}} - \boldsymbol{f}_j^{\text{old}})\right]^{1/2}$;
  Compute amplitude error $\varepsilon_\sigma = \left|\sigma^{\text{new}} - \sigma^{\text{old}}\right| / |\sigma^{\text{new}}|$;
  Update $\boldsymbol{f}_j^{\text{old}} \leftarrow \boldsymbol{f}_j^{\text{new}}$, for $j = 1, 2, \ldots, d$; $\sigma^{\text{old}} \leftarrow \sigma^{\text{new}}$;
end
Store $\boldsymbol{f}_j \leftarrow \boldsymbol{f}_j^{\text{new}}$, for $j = 1, 2, \ldots, d$; $\sigma \leftarrow \sigma^{\text{new}}$;

Algorithm 1. Algebraic rank-one approximation for non-separable tensor $\boldsymbol{F}$.

2.3. *Accounting for a separable input*

Algorithm 1 is easily adapted to the case in which the input tensor is already provided in a separated format. That is, instead of having $\boldsymbol{F}$, we have $\boldsymbol{\Phi}$ such that

$$\boldsymbol{\Phi} = \sum_{\ell=1}^{L} \boldsymbol{\phi}_1^\ell \otimes \boldsymbol{\phi}_2^\ell \otimes \cdots \otimes \boldsymbol{\phi}_d^\ell \tag{22}$$

The LS approximation of $\boldsymbol{\Phi}$ in the form of another separated tensor $\boldsymbol{F}_{\text{PGD}}$ is meaningful because it may significantly reduce the number of terms required to represent the same tensorial magnitude. That is, one would expect having $M \ll L$ and achiving a similar accuracy. Actually, this strategy is often used along the PGD computations because, when solving parametric boundary value problems, the PGD terms may exhibit redundancies (linear functional dependencies) from a LS viewpoint, see [18].

In practice, replacing the full tensor $\boldsymbol{F}$ by the separated tensor $\boldsymbol{\Phi}$ results only in a difference in the computation of the auxiliary vector $\boldsymbol{g}$ in (18) or (19). Actually, in this case, (18) is readily replaced by

$$\boldsymbol{g} = \boldsymbol{\Phi} \vdots \bigotimes_{j \neq \gamma}^{d} \tilde{\boldsymbol{f}}_j^1 = \sum_{\ell=1}^{L} \left[\prod_{j \neq \gamma}^{d} \boldsymbol{\phi}_j^{\ell\,\mathsf{T}} \tilde{\boldsymbol{f}}_j^1\right] \boldsymbol{\phi}_\gamma^\ell. \tag{23}$$

Thus, adapting Algorithm 1 to the case in which the input tensor is already expressed in separable format requires only replacing the line in which vector $\boldsymbol{g}$ is computed, using (23) instead of (18). This is

in practice the only difference between the algorithm providing rank-one tensor separation and rank-one tensor compression.

3. Complete LS PGD algorithm

The strategies described in the previous sections allow obtaining a LS PGD approximation $\boldsymbol{F}_{\text{PGD}}^{M}$ according to (14) of a d-dimensional tensor $\boldsymbol{F} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$. The same strategy works also for an already separated input tensor $\boldsymbol{\Phi}$ as given in (22). Both the separation of $\boldsymbol{F}$ and the compression of $\boldsymbol{\Phi}$ are performed by algorithms having the same structure and the only difference is in the computation of vector $\boldsymbol{g}$ which is performed using (18) for the separation and (23) for the compression.

The present section is devoted to summarize the ideas of sections 2.1 and 2.2 in a compact algorithmic form.

As already indicated above, the main idea is to use the rank-one approximation Algorithm 1 to compute the successive terms that conform $\boldsymbol{F}_{\text{PGD}}$, following the greedy approach described in section 2.1. The core of the algorithm is the rank-one updating from $\boldsymbol{F}_{\text{PGD}}^{m-1}$ to $\boldsymbol{F}_{\text{PGD}}^{m}$. This is essentially following the same rationale as in Algorithm 1 but replacing $\boldsymbol{F}$ by $\boldsymbol{F} - \boldsymbol{F}_{\text{PGD}}^{m-1}$ as the tensor to be rank-one approximated.

Thus, the rank-one approximation inside the greedy loop aims at approximating $\boldsymbol{f}_1^m, \boldsymbol{f}_2^m, \ldots, \boldsymbol{f}_d^m$ and σ_m that provides the best rank-one update of $\boldsymbol{F}_{\text{PGD}}^{m-1}$. Here again, this results in a different expression to compute vector $\boldsymbol{g}$, alternative to (18), namely

$$\boldsymbol{g} = (\boldsymbol{F} - \boldsymbol{F}_{\text{PGD}}^{m-1}) \vdots \bigotimes_{j \neq \gamma}^{d} \boldsymbol{f}_j^m = \boldsymbol{F} \vdots \bigotimes_{j \neq \gamma}^{d} \boldsymbol{f}_j^m - \sum_{\tilde{m}=1}^{m-1} \sigma_{\tilde{m}} \left[\prod_{j \neq \gamma}^{d} \boldsymbol{f}_j^{\tilde{m}\,\mathsf{T}} \boldsymbol{f}_j^m \right] \boldsymbol{f}_\gamma^{\tilde{m}}. \tag{24}$$

Where the separated structure of $\boldsymbol{F}_{\text{PGD}}$ is used in the last term, following exactly the same as in (23) with $\boldsymbol{\Phi}$.

Thus, the difference between the rank-one approximation described in Algorithm 1 and the complete PGD approximation summarized in Algorithm 2 lies in the addition of an outer loop on the number of terms (looping for m) and in the computation of $\boldsymbol{g}$ that is performed according to (24).

3.1. *Algorithmic details: separated and complex inputs*

Analogously as it discussed in Section 2.3, Algorithm 2 is easily adapted to accept a separable input $\boldsymbol{\Phi}$ instead of a full multimensional tensor $\boldsymbol{F}$. Here again, the only difference between separation (of $\boldsymbol{F}$) and compression (of $\boldsymbol{\Phi}$) lies in the expression to compute vector $\boldsymbol{g}$ that for compression reads

$$\boldsymbol{g} = (\boldsymbol{\Phi} - \boldsymbol{F}_{\text{PGD}}^{m-1}) \vdots \bigotimes_{j \neq \gamma}^{d} \boldsymbol{f}_j^m = \sum_{\ell=1}^{L} \left[\prod_{j \neq \gamma}^{d} \boldsymbol{\phi}_j^{\ell\,\mathsf{T}} \boldsymbol{f}_j^m \right] \boldsymbol{\phi}_\gamma^\ell - \sum_{\tilde{m}=1}^{m-1} \sigma_{\tilde{m}} \left[\prod_{j \neq \gamma}^{d} \boldsymbol{f}_j^{\tilde{m}\,\mathsf{T}} \boldsymbol{f}_j^m \right] \boldsymbol{f}_\gamma^{\tilde{m}}. \tag{25}$$

In the case the input data is complex, that is either $\boldsymbol{F}$ or $\boldsymbol{\Phi}$ lie in $\mathbb{C}^{n_1 \times \cdots \times n_d}$ instead of $\mathbb{R}^{n_1 \times \cdots \times n_d}$, the algorithm has to be slightly modified. In practice, all the modifications derive from the fact that the Frobenius-type norm for complex tensors differs from (10) in the sense that the first argument in the product has to be conjugated and therefore it reads

$$\|\boldsymbol{F}\|^2 = \bar{\boldsymbol{F}} \vdots \boldsymbol{F} = \left[\bar{\boldsymbol{F}}\right]_{j_1 j_2 \ldots j_d} [\boldsymbol{F}]_{j_1 j_2 \ldots j_d} = \sum_{j_1=1}^{n_1} \sum_{j_2=1}^{n_2} \cdots \sum_{j_d=1}^{n_d} \left| [\boldsymbol{F}]_{j_1 j_2 \ldots j_d} \right|^2, \tag{26}$$

Data: Tensor of order d to be approximated: $\boldsymbol{F}$ with general term $[\boldsymbol{F}]_{i_1 i_2 \ldots i_d}$,
(for $i_\gamma = 1, \ldots, n_\gamma$ and $\gamma = 1, \ldots, d$)
Tolerances $0 < \eta \ll 1$ for alternated directions and $0 < \eta^\star \ll 1$ for greedy algorithm

Result: PGD approximation: $\boldsymbol{F}_{\texttt{PGD}} = \displaystyle\sum_{m=1}^{M} \sigma_m \boldsymbol{f}_1^m \otimes \boldsymbol{f}_2^m \otimes \cdots \otimes \boldsymbol{f}_d^m$

Initialize counter of PGD terms $m = 1$ and starting value of amplitude $\sigma_1 = 1$
while $\sigma_m > \eta^\star \sigma_1$ **do**
  Initialize: assign values to $\boldsymbol{f}_j^{\texttt{old}}$, for $j = 1, 2, \ldots, d$;
  Compute $\sigma^{\texttt{old}} = \displaystyle\prod_{j=1}^{d} \left[\boldsymbol{f}_j^{\texttt{old}\,\mathsf{T}} \boldsymbol{f}_j^{\texttt{old}}\right]^{1/2}$;
  Normalize: $\boldsymbol{f}_j^{\texttt{old}} \leftarrow \left[\boldsymbol{f}_j^{\texttt{old}\,\mathsf{T}} \boldsymbol{f}_j^{\texttt{old}}\right]^{-1/2} \boldsymbol{f}_j^{\texttt{old}}$, for $j = 1, 2, \ldots, d$;

  while $\varepsilon > \eta$ *or* $\varepsilon_\sigma > \eta$ **do**
    for $\gamma = 1 \ldots d$ **do**
      Compute $\alpha = \displaystyle\prod_{j \neq \gamma} \boldsymbol{f}_j^{\texttt{old}\,\mathsf{T}} \boldsymbol{f}_j^{\texttt{old}}$
      Compute $\boldsymbol{g}$ such that
      $$[\boldsymbol{g}]_{i_\gamma} = \sum_{i_1=1}^{n_1} \cdots \sum_{i_{\gamma-1}=1}^{n_{\gamma-1}} \sum_{i_{\gamma+1}=1}^{n_{\gamma+1}} \cdots \sum_{i_d=1}^{d} [\boldsymbol{F}]_{i_1 \ldots i_{\gamma-1} i_\gamma i_{\gamma+1} \ldots i_d} \prod_{j \neq \gamma}^{d} \left[\boldsymbol{f}_j^{\texttt{old}}\right]_{i_j} - \sum_{\tilde{m}=1}^{m-1} \sigma_{\tilde{m}} \left(\prod_{j \neq \gamma} \boldsymbol{f}_j^{\tilde{m}\,\mathsf{T}} \boldsymbol{f}_j^{\texttt{old}} \right) \left[\boldsymbol{f}_\gamma^{\tilde{m}}\right]_{i_\gamma} , \text{ for } i_\gamma = 1, 2, \ldots, n_\gamma \text{ ;}$$
      Compute $\boldsymbol{f}_\gamma^{\texttt{new}} = \dfrac{1}{\alpha} \boldsymbol{g}$
    end
    Compute $\sigma^{\texttt{new}} = \displaystyle\prod_{j=1}^{d} \left[\boldsymbol{f}_j^{\texttt{new}\,\mathsf{T}} \boldsymbol{f}_j^{\texttt{new}}\right]^{1/2}$;
    Normalize: $\boldsymbol{f}_j^{\texttt{new}} \leftarrow \left[\boldsymbol{f}_j^{\texttt{new}\,\mathsf{T}} \boldsymbol{f}_j^{\texttt{new}}\right]^{-1/2} \boldsymbol{f}_j^{\texttt{new}}$, for $j = 1, 2, \ldots, d$;
    Compute modal error $\varepsilon = \left[\displaystyle\prod_{j=1}^{d} (\boldsymbol{f}_j^{\texttt{new}} - \boldsymbol{f}_j^{\texttt{old}})^{\mathsf{T}} (\boldsymbol{f}_j^{\texttt{new}} - \boldsymbol{f}_j^{\texttt{old}}) \right]^{1/2}$;
    Compute amplitude error $\varepsilon_\sigma = \left| \sigma^{\texttt{new}} - \sigma^{\texttt{old}} \right| / \left| \sigma^{\texttt{new}} \right|$;
    Update $\boldsymbol{f}_j^{\texttt{old}} \leftarrow \boldsymbol{f}_j^{\texttt{new}}$, for $j = 1, 2, \ldots, d$; $\sigma^{\texttt{old}} \leftarrow \sigma^{\texttt{new}}$;
  end
  Store: $\sigma_m \leftarrow \sigma$; $\boldsymbol{f}_j^m \leftarrow \boldsymbol{f}_j^{\texttt{new}}$, for $i = 1, 2, \ldots, d$;
  Update $m \leftarrow m + 1$;
end
Store: $M \leftarrow m$;

Algorithm 2. Algebraic PGD approximation for multidimensional tensor $\boldsymbol{F}$

where the bar $\bar{(\cdot)}$ stands for the conjugate quantity. This affects the computation of the norms of the sectional vectors such that the scalar products require not only to transpose one of the vectors but also to conjugate it. Moreover, in the tensorial multicontraction in, for instance, (25) what is located left to symbol $\vdots$ must be conjugated. This results in replacing (25) by

$$\boldsymbol{g} = \overline{(\boldsymbol{\Phi} - \boldsymbol{F}_{\texttt{PGD}}^{m-1})} \vdots \bigotimes_{j \neq \gamma}^{d} \boldsymbol{f}_j^m = \sum_{\ell=1}^{L} \left[\prod_{j \neq \gamma}^{d} \bar{\boldsymbol{\phi}}_j^{\ell\,\mathsf{T}} \boldsymbol{f}_j^m \right] \boldsymbol{\phi}_\gamma^\ell - \sum_{\tilde{m}=1}^{m-1} \sigma_{\tilde{m}} \left[\prod_{j \neq \gamma}^{d} \bar{\boldsymbol{f}}_j^{\tilde{m}\,\mathsf{T}} \boldsymbol{f}_j^m \right] \bar{\boldsymbol{f}}_\gamma^{\tilde{m}} . \tag{27}$$

3.2. *Disambiguating normalization*

In the stopping criterion of the alternated directions iterations, the comparison of the successive approximation to the sectional modes in Algorithms 1 and 2 is carried out in terms of the normalized vectors,

see (21). Thus, in the algorithms, the normalization is readily indicated as $\boldsymbol{f}_j^{\mathtt{new}} \leftarrow \left[\boldsymbol{f}_j^{\mathtt{new}\,\mathsf{T}} \boldsymbol{f}_j^{\mathtt{new}}\right]^{-1/2} \boldsymbol{f}_j^{\mathtt{new}}$, for $j = 1, 2, \ldots, d$. However, this operation must be a little bit more sophisticated. In the real framework, the normalization must also disambiguate the sign. In other words, dividing by the norm there could be ± 1 factors that may affect all the sectional modes (with an even number of -1 that globally cancel out). In this situation, the numerical stopping criterion is requiring more iterations although, in practice, the stationarity in the alternated directions iterations has already been reached. The solution to this problem is simple but it cannot be overlooked. One possibility is to disambiguate the sign by enforcing that the largest (in absolute value) component of $\boldsymbol{f}_j^{\mathtt{new}}$ is positive.

If dealing with complex variables, the standard normalization would be $\boldsymbol{f}_j^{\mathtt{new}} \leftarrow \left[\bar{\boldsymbol{f}}_j^{\mathtt{new}\,\mathsf{T}} \boldsymbol{f}_j^{\mathtt{new}}\right]^{-1/2} \boldsymbol{f}_j^{\mathtt{new}}$, for $j = 1, 2, \ldots, d$. In this case, the ambiguity is not only up to a sign but up to any complex coefficient with unit module. The different modes $\boldsymbol{f}_j^{\mathtt{new}}$ may be affected by coefficients z_j, $j = 1, 2, \ldots, d$, with $|z_j| = 1$ and such that $\prod_{j=1}^{d} z_j = 1$ to cancel out globally. In order to disambiguate these factors, the normalization is carried out such that that the largest (in module) component of $\boldsymbol{f}_j^{\mathtt{new}}$ is real and positive.

4. Numerical examples

The methodologies presented above for PGD tensor separation and tensor compression are tested in this section for different numerical examples.

4.1. *Example 1: separation of a 2D manufactured tensor*

The first example uses a tensor $\boldsymbol{F}$ synthetically manufactured from a reduced number (six) of separated modes. Thus, the separation algorithm, whose input is the full reconstructed tensor is challenged to produce a number of modes similar to the number used to manufacture the input. The bidimensional ($d = 2$) tensor $\boldsymbol{F}$ is generated from a separated expression,

$$\boldsymbol{\Phi}(x, y) = \sum_{\ell=1}^{6} \boldsymbol{\phi}_x^{\ell} \otimes \boldsymbol{\phi}_y^{\ell} \tag{28}$$

where the modal vectors $\boldsymbol{\phi}_x^{\ell}$ and $\boldsymbol{\phi}_y^{\ell}$, for $\ell = 1, \ldots, 6$, are defined as the images of vector $\boldsymbol{\xi} = [0\ \frac{1}{99}\ \frac{2}{99}\ \ldots\ \frac{98}{99}\ 1]^{\mathsf{T}} \in \mathbb{R}^{100}$ by different continuous functions taking values in $[0, 1]$, namely $\boldsymbol{\phi}_x^{\ell} = \phi_x^{\ell}(\boldsymbol{\xi})$ and $\boldsymbol{\phi}_y^{\ell} = \phi_y^{\ell}(\boldsymbol{\xi})$, where

$$\phi_x^1(x) = \sin(\pi x) \qquad \phi_y^1(y) = \sin(\pi y) \tag{29a}$$

$$\phi_x^2(x) = \exp\left(\frac{-(x-0.5)^2}{0.01}\right) \qquad \phi_y^2(y) = -\exp\left(\frac{-(y-0.5)^2}{0.01}\right) \tag{29b}$$

$$\phi_x^3(x) = \exp\left(\frac{-(x-0.2)^2}{0.01}\right) \qquad \phi_y^3(y) = \exp\left(\frac{-(y-0.2)^2}{0.02}\right) \tag{29c}$$

$$\phi_x^4(x) = \exp\left(\frac{-(x-0.6)^2}{0.005}\right) \qquad \phi_y^4(y) = \exp\left(\frac{-(y-0.9)^2}{0.005}\right) \tag{29d}$$

$$\phi_x^5(x) = \exp\left(\frac{-(x-0.1)^2}{0.01}\right) \qquad \phi_y^5(y) = \exp\left(\frac{-(y-0.75)^2}{0.005}\right) \tag{29e}$$

$$\phi_x^6(x) = \exp\left(\frac{-(x-0.8)^2}{0.02}\right) \qquad \phi_y^6(y) = -\exp\left(\frac{-(y-0.2)^2}{0.002}\right). \tag{29f}$$

Figure 1 shows the 1D modal functions ϕ_x^ℓ and ϕ_y^ℓ. The reconstructed 2D tensor $\boldsymbol{F}$ is represented in Figure 4a.

The PGD separation described in Algorithm 2 is applied to $\boldsymbol{F}$, and provides

$$\boldsymbol{F}_{\text{PGD}}^M = \sum_{m=1}^{M} \sigma_m \boldsymbol{f}_x^m \otimes \boldsymbol{f}_y^m. \tag{30}$$

The functions corresponding to vectors $\boldsymbol{f}_x^m$ and $\boldsymbol{f}_y^m$ defining the first six modes (depicted in Figure 1) account for the exact separated representation of $\boldsymbol{F}$. The algorithm in two-dimensions provides an optimal separated form coinciding to the results provided by a singular value decomposition method (see Figure 2a). The modal error evolution is computed as the Frobenius norm of the difference of the tensors $\boldsymbol{F}$ and $\boldsymbol{F}_{\text{PGD}}^M$. The exact solution is reached with 6 modes up to machine precision as can be seen in Figure 2b.

4.2. *Example 2: separation of a higher dimension tensor*

Ir order to analyze the behaviour of the algorithm in higher dimensions, the previous case has been extended to a seven-dimensional synthetic problem. A set of functions similar to (29) is used to build the higher-order tensor,

$$\phi_k^1(x_k) = \sin(\pi x_k)$$
$$\phi_k^j(x_k) = a_{jk} \exp\left(\frac{-(x_k - b_{jk})^2}{c_{jk}}\right).$$

for $j = 2, \ldots, 6$ and $k = 1, \ldots 7$, in the domain $[0,1]^7$. The discretization of each dimension is done with 10 points, that is $\boldsymbol{\xi} = [0\ \frac{1}{9}\ \frac{2}{9}\ \ldots\ \frac{8}{9}\ 1]^\intercal \in \mathbb{R}^{10}$, providing a full tensor with 10^7 real entries. Coefficients a, b and c are described in Appendix A.

The results of the seven dimensional case show that the obtained separation does not recover the original 6 modes used to build the tensor. It actually requires 150 terms to obtain separated representation with a relative error of 10^{-3} (Figure 3). This is standard in higher dimensional problems, because none of the available separation procedures is able to provide optimal solutions for $d > 2$ [14]. Note however that the reduction in terms of storage is significant: the 10^7 real entries of the full tensor are reduced in the separated form to $150 \times 7 \times 10 \approx 10^4$. That is, the relative error of 10^{-3} is the (small) price to pay for saving 3 orders of magnitude in the storage.

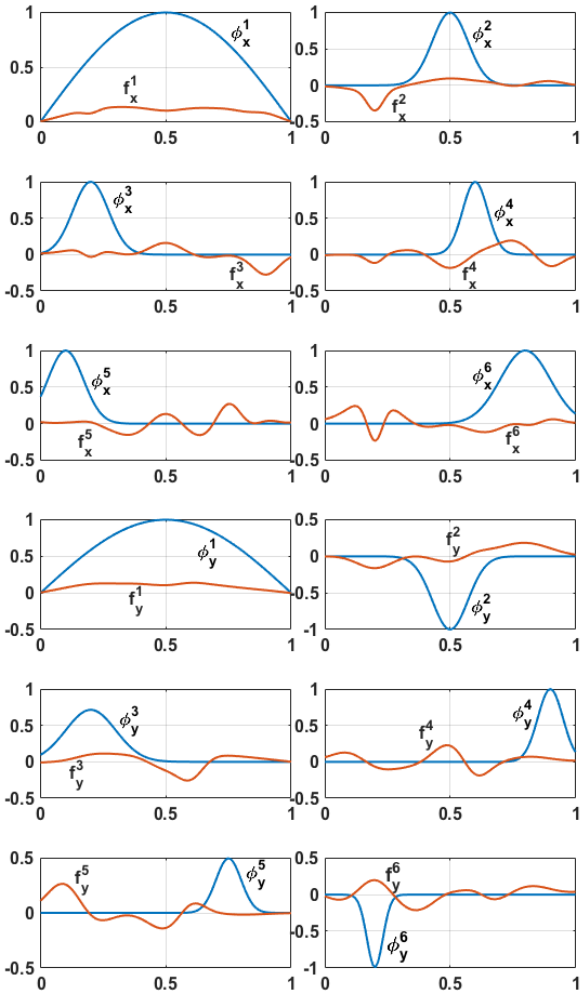

Fig. 1. Example 1: Input modal functions ϕ_x^ℓ and ϕ_y^ℓ, for $\ell = 1, \ldots, 6$ (in blue) used to manufacture $\boldsymbol{F}$ and modes $\boldsymbol{f}_x^j$ and $\boldsymbol{f}_y^j$, for $\ell = 1, \ldots, M = 6$ (in red) of the separated representation of $\boldsymbol{F}_{\text{PGD}}$.

4.3. *Example 3: denoising data*

An additional application of the proposed PGD algorithm is to filter noise in a data set. In this context, this example retakes the previous 2D case of Section 4.1 perturbing the input by adding a random noise with an amplitude of 2% of the maximum tensor value, see Figure 4b. Thus, the 6 modes describing the tensor presented in Example 1 are duly identified and separated from the noise that is actually modelled (or represented) by the remaining modes. Figure 5 shows the modal error evolution with respect to the

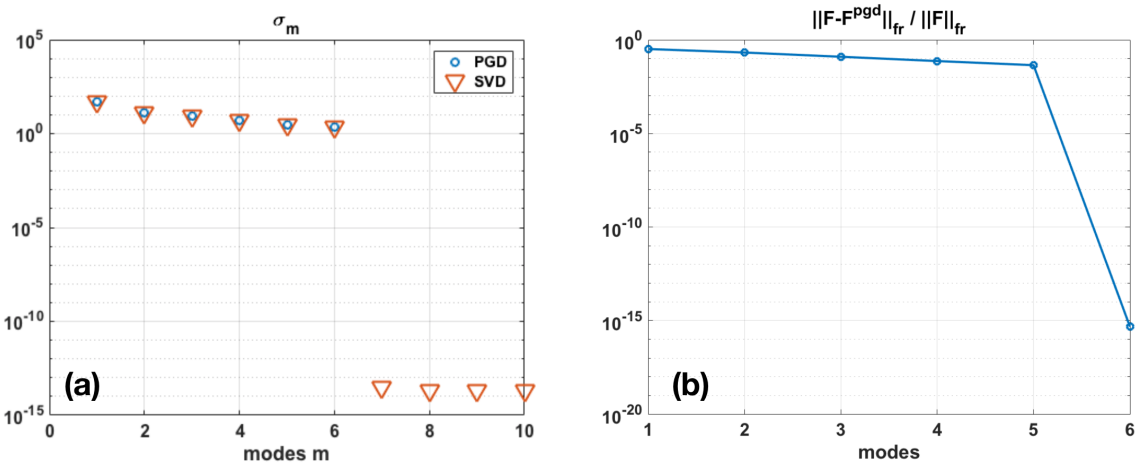

Fig. 2. Example 1: [Panel a] Modal amplitudes (σ_m) of the first 6 terms of the PGD separated tensor and comparison with the modal amplitudes provided the Singular Value Decomposition method. [Panel b] Relative error of the PGD separated tensor in Frobenius norm as a function of number of terms.

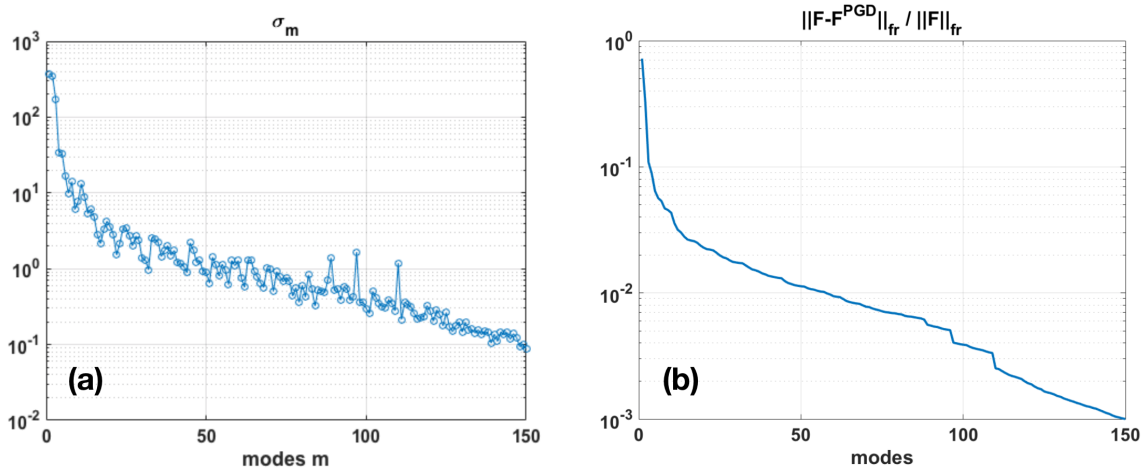

Fig. 3. Example 2: [Panel a] Modal amplitudes (σ_m) of the first 150 terms of the PGD separated tensor. [Panel b] Relative error of the PGD separated tensor in Frobenius norm as a function of number of terms.

original and noised tensors. As it can be seen, the first 6 modes provide an accurate approximation of the original tensor, while from mode 7 on, the algorithm is adding to the separated representation the information added by the noise. It can be observed in Figure 5 that the discrepancies $\|\boldsymbol{F}_{\texttt{noised}} - \boldsymbol{F}_{\texttt{PGD}}\|$ and $\|\boldsymbol{F} - \boldsymbol{F}_{\texttt{PGD}}\|$ differ at the sixth mode and the following ones. This is associated with the fact that $\boldsymbol{F}_{\texttt{PGD}}$ is computed as an approximation of the tensor perturbed by the noise, $\boldsymbol{F}_{\texttt{noised}}$, and not from the original one, $\boldsymbol{F}$. This is essentially visible for the 6th mode because it is where relative amplitude of the mode (the accuracy) meets the magnitude of the noise.

That is the reason why the error computed against the original tensor increases after mode 6. Furthermore, in case of the error computed against the noised tensor, it decreases as the solution approaches to the noised configuration. Note that from mode 6 on, such error decreases linearly with a very low convergence rate as expected in the reproduction of a function with structure (noise).

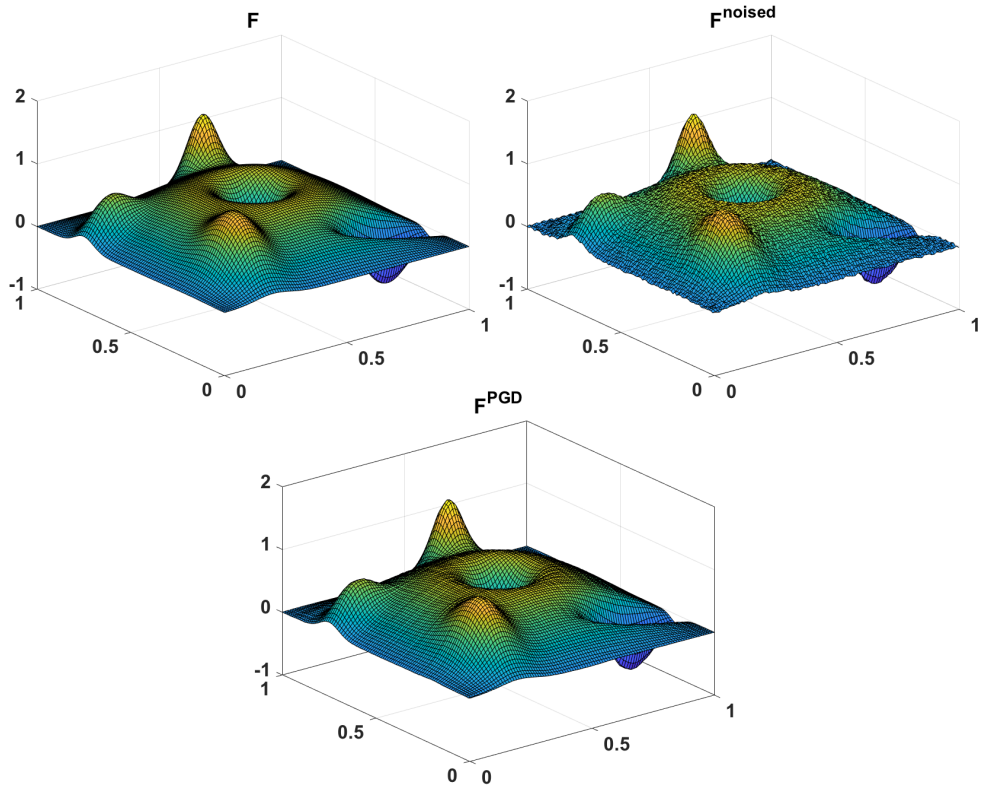

Fig. 4. Examples 1 and 3: [Top left panel] Representation of the tensor $\boldsymbol{F}$ used in Example 1. [Top right panel] Noised version of $\boldsymbol{F}$. [Bottom panel] Denoised tensor via the PGD algorithm.

4.4. *Example 4: compression of higher-order complex separated tensor*

The PGD compression strategy presented above is here applied to a complex tensor of large size and dimension $d = 4$. The tensor, $\boldsymbol{\Phi}$, is already available in a separated format as shown in (22) and contains the data to be compressed. It is the PGD solution of a parameterized sea wave propagation problem in harbor taken from [18], see Figure 6 for an illustration. The four dimensions correspond to the space distribution of the (complex) wave height and the three parameters, which are: 1) the incoming wave direction, 2) the wave frequency and 3) the reflectivity of the coastline. The original dataset has $L = 1500$ modes using a discretization of space, frequency, angle and reflectivity of 15757, 100, 50 and 10 degrees of freedom respectively. Note that the problem is stated in frequency domain and all dimensions are stored as complex numbers. The full tensor of this solution, therefore, occupies 12GB of memory (corresponding to $15757 \times 100 \times 50 \times 10 = 787.85 \times 10^6$ complex entries) The storage of the separated solution with 1500 modes (corresponding to $1500 \times (15757 + 100 + 50 + 10) = 23.8755 \times 10^6$ complex entries) requires 364MB. The goal of the compression is to approximate $\boldsymbol{\Phi}$ by $\boldsymbol{F}_{\text{PGD}}$ with $M \ll L$ and to significantly reduce the storage requirements while maintaining the accuracy of the represented quantities. Here the PGD compression is carried out to obtain 200 modes (instead of using a tolerance $\eta^\star$ to stop the

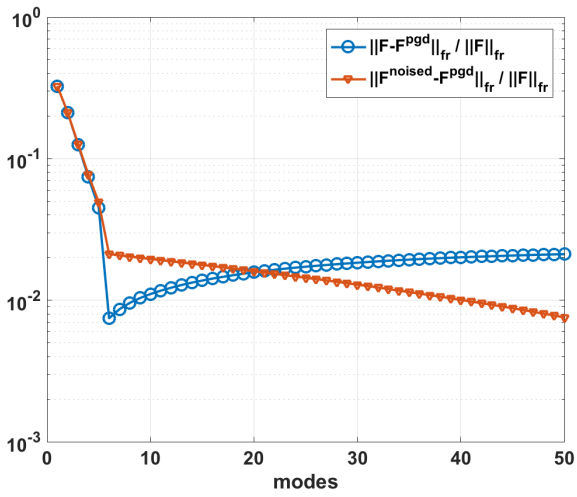

Fig. 5. Example 3: Evolution with the number of modes of the relative error of the result of the PDG algorithm, $\boldsymbol{F}_{\texttt{PGD}}$, computed against i) the original tensor $\boldsymbol{F}$ and ii) the noised version $\boldsymbol{F}_{\texttt{noised}}$ (used as input for PGD).

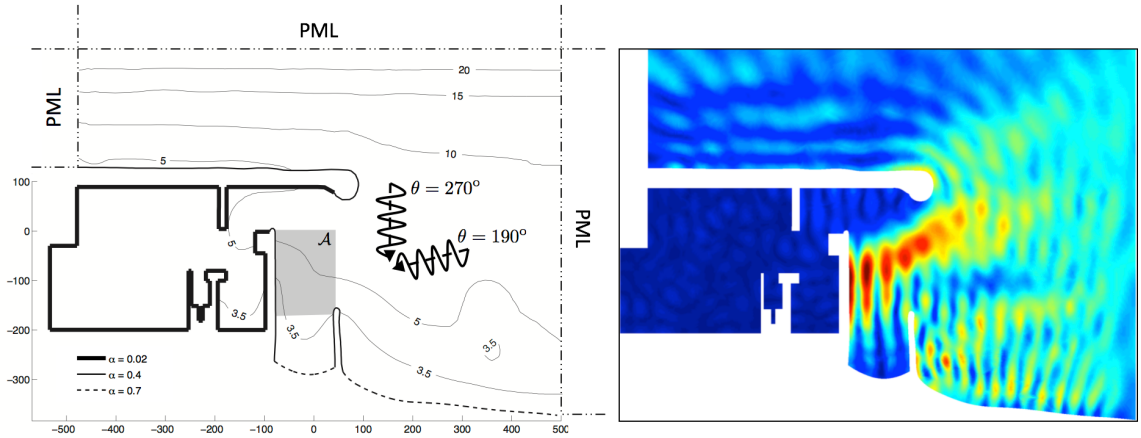

Fig. 6. Example 4: [Left panel] domain, boundary conditions and parameters of the parameterized wave problem (from [18]). [Right panel] illustration of one spatial solution for some set of parameters.

process). Note that with 200 modes, the storage requirement reduce to 48.5MB. The objective is to check wether the PGD compression produces a better description of the data when the number of modes is limited by the storage capacity. The results are shown in Figure 7. In the left, the evolution of the modal amplitudes is represented and, as expected, they globally show a decreasing trend until they get stabilized. More interestingly, in the right curve, the actual error $\|\boldsymbol{\Phi} - \boldsymbol{F}^m_{\texttt{PGD}}\|$ is represented and shows a monotonic decreasing behavior. Moreover when compared with the error associated with the truncation of $\boldsymbol{\Phi}$ to the first 200 terms, namely $\|\boldsymbol{\Phi} - \sum_{\ell=1}^{200} \boldsymbol{\phi}_1^\ell \otimes \boldsymbol{\phi}_2^\ell \otimes \boldsymbol{\phi}_3^\ell \otimes \boldsymbol{\phi}_4^\ell\|$. This value is represented in the right panel of Figure 7 as a red cross. It can be noted that, with respect to the first 200 terms of the original separation of $\boldsymbol{\Phi}$, the error associated with $\boldsymbol{F}^{200}_{\texttt{PGD}}$ is one order of magnitude lower.

This example is therefore demonstrating that the PGD compression is able to shorten the separable expressions and improve the accuracy of the overall representation at a limited storage capacity.

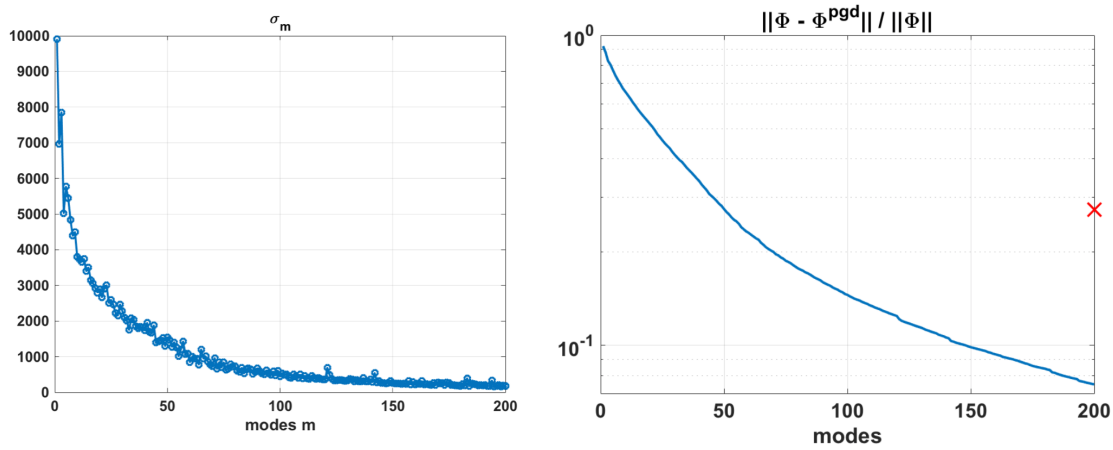

Fig. 7. Example 4: [Left panel] evolution of modal amplitudes (σ_m) along the first 200 terms of PGD compression. [Right panel] evolution of the relative error of the PGD compression (with respect to the full $\mathbf{\Phi}$ tensor). The red cross marks the value of the norm of the difference between $\mathbf{\Phi}$ and the tensor reconstructed with the first original 200 modes of $\mathbf{\Phi}$.

5. Concluding remarks

The PGD Least-Squares approximation is presented here as a computational tool to perform high-dimensional tensor separation at an affordable computational cost and with a limited coding complexity. The algorithms are discussed in detail, with special emphasis on the stopping criteria for both the greedy strategy and the alternated directions iteration scheme.

Moreover, the same idea is also used to compress separated approximations with a large number of terms and to reduce the storage requirements while keeping the accuracy of the separated representation.

References

[1] L. R. Tucker, Some mathematical notes on three-mode factor analysis, Psychometrika 31 (1966) 279–311.

[2] R. A. Harshman, Foundations of the parafac procedure: Models and conditions for an explanatory multi-modal factor analysis, UCLA Working Papers in Phonetics 16 (1970) 1–84.

[3] C. J. Appellof, E. R. Davidson, Strategies for analyzing data from video fluorometric monitoring of liquid chromatographic effluents, Anal. Chem. 53 (1981) 2053–2056.

[4] R. Bro, PARAFAC. Tutorial and applications, Chemometrics and Intelligent Laboratory Systems 38 (1997) 149–171.

[5] C. Beckmann, S. Smith, Tensorial extensions of independent component analysis for multisubject FMRI analysis, NeuroImage 25 (2005) 294–311.

[6] M. De Vos, A. Vergult, L. De Lathauwer, W. De Clercq, S. Van Huffel, P. Dupont, A. Palmini, W. Van Paesschen, Canonical decomposition of ictal scalp eeg reliably detects the seizure onset zone, NeuroImage 37 (2007) 844–854.

[7] H. Wang, N. Ahuja, Facial expression decomposition, ICCV 2003: Proceedings of the 9th IEEE International Conference on Computer Vision 2 (2003) 958–965.

[8] A. Shashua, T. Hazan, Non-negative tensor factorization with applications to statistics and computer vision, ICML 2005: Proceedings of the 22nd International Conference on Machine Learning (2005) 792–799.

[9] B. Chen, A. Petropolu, L. De Lathauwer, Blind identification of convolutive mimo systems with 3 sources and 2 sensors, EURASIP J. Appl. Signal Process. 5 (2002) 487–496.

[10] L. De Lathauwer, J. Vandewalle, Dimensionality reduction in higher-order signal processing and rank-(r1,r2,...,rn) reduction in multilinear algebra, Linear Algebra Appl. 391 (2004) 31–55.

[11] N. Liu, B. Zhang, J. Yan, Z. Chen, W. Liu, F. Bai, L. Chien, Text representation: From vector to tensor, Proceedings of the 5th IEEE International Conference on Data Mining (2005) 725–728.

[12] I. Ibraghimov, Application of the three-way decomposition for matrix compression, Numer. Linear Algebra Appl. 9 (2002) 551–565.

[13] W. Hackbusch, B. N. Khoromskij, Tensor-product approximation to operators and functions in high dimensions, J. Complexity 23 (2007) 697–714.

[14] T. Kolda, B. Bader, Tensor decompositions and applications, SIAM Rev. 51 (2009) 455–500.

[15] F. Chinesta, E. Cueto, A. Huerta, PGD for solving multidimensional and parametric models, in: Separated representations and PGD-based model reduction, Vol. 554 of CISM Courses and Lectures, Springer, Vienna, 2014, pp. 27–89.

[16] F. Chinesta, R. Keunings, A. Leygue, The proper generalized decomposition for advanced numerical simulations. A primer, Springer Briefs in Applied Sciences and Technology, Springer, Cham, 2014.

[17] S. Zlotnik, P. Díez, D. Modesto, A. Huerta, Proper generalized decomposition of a geometrically parametrized heat problem with geophysical applications, Int. J. Numer. Methods Eng. 103 (10) (2015) 737–758.

[18] D. Modesto, S. Zlotnik, A. Huerta, Proper Generalized Decomposition for parameterized helmholtz problems in heterogeneous and unbounded domains: application to harbor agitation, Comput. Methods Appl. Mech. Eng. 295 (2015) 127–149.

Appendices

A. Appendix

The functions used in the construction of the seven-dimensional tensor used in Section 4.2 are described next. The matrix $\mathbf{A}$ built with the coefficients a_{jk} reads,

$$\mathbf{A} = \begin{bmatrix} 1 & -1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Similarly, the coefficients b_{jk} and c_{jk} are,

$$\mathbf{B} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.6 & 0.9 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.1 & 0.75 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.8 & 0.2 & 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0.01 & 0.02 & 0.01 & 0.02 & 0.02 & 0.02 & 0.02 \\ 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0.01 & 0.02 & 0.01 & 0.02 & 0.02 & 0.02 & 0.02 \end{bmatrix}$$