Aguado, J.V., Borzacchiello, D., Kollepara, K.S., Chinesta, F. and Huerta, A., "Tensor representation of non-linear models using cross approximations," to appear in *Journal of Scientific Computing* (2019).

Tensor representation of non-linear models using cross approximations

José V. Aguado, Domenico Borzacchiello, Kiran S. Kollepara
Institut de Calcul Intensif (ICI), École Centrale de Nantes, Nantes, France

Francisco Chinesta
ESI Group International Chair, Laboratoire PIMM,
École Nationale des Arts et Métiers (ENSAM) ParisTech, Paris, France.

Antonio Huerta
Laboratori de Càlcul Numèric (LaCàN), ETS de Ingenieros de Caminos, Canales y Puertos,
Universitat Politècnica de Catalunya, Barcelona, Spain

January 6, 2019

Abstract

Tensor representations allow compact storage and efficient manipulation of multi-dimensional data. Based on these, tensor methods build low-rank subspaces for the solution of multi-dimensional and multi-parametric models. However, tensor methods cannot always be implemented efficiently, specially when dealing with non-linear models. In this paper, we discuss the importance of achieving a tensor representation of the model itself for the efficiency of tensor-based algorithms. We investigate the adequacy of interpolation rather than projection-based approaches as a means to enforce such tensor representation, and propose the use of cross approximations for models in moderate dimension. Finally, linearization of tensor problems is analyzed and several strategies for the tensor subspace construction are proposed.

Keywords: Non-linear Modeling; Reduced Order Modeling; Low-rank Tensor Approximation; Proper Generalized Decomposition; Cross Approximations; Parametrized PDE; Multi-dimensional Problems

1 Introduction

Tensor methods can be regarded as a specific class of reduced-order methods (ROM), designed for the solution of multi-dimensional and multi-parametric problems. ROM have

proved in all its variants to be a key tool in simulation-based engineering sciences, providing a scientific and mathematical basis for fast simulation (sometimes even in real-time) of engineered systems. They have opened the path towards a better integration of numerical simulation in dynamic data-driven application systems [1], rapid numerical prototyping [2] and dedicated simulation applications [3], among many other examples.

Reduced-order methods are designed to provide fast yet accurate approximations of the solution of a numerical model. The concept of *reducibility* is therefore central. It states that the solution of many models can be well approximated using few degrees of freedom, provided that a suitable representation basis is chosen. The two principal families of ROM differ essentially on how they build such appropriate representation basis. One one hand, *a-posteriori* ROM extract a low-dimensional subspace from a training simulation dataset [4–8]. On the other hand, tensor methods (also referred to as *a-priori* ROM) compute low-rank tensor subspaces by setting up an optimization problem [9–11]. Both families of methods require a special treatment of non-linear problems in order to preserve their efficiency.

Literature on non-linear a-posteriori ROM is abundant. Most of the approaches propose building a secondary low-dimensional subspace, other than the solution's subspace, in order to approximate the non-linear term. Then, the Empirical Interpolation Method (EIM) [12, 13], as well as its discrete counterpart (DEIM) [14], can be used to give interpolation properties to the non-linear term's subspace. As opposed to projection, interpolation allows approximating the non-linear term by evaluating only few points, which can be done inexpensively during the on-line stage. A number of variants have been proposed, such as the unassembled DEIM [15], which is specifically tailored for finite element computations, and the localized DEIM [16], that proposes to switch between *local* subspaces in order to capture different regimes of the system, using clustering and supervised learning techniques. We shall give in Section 3.2 further details on the interpolation approach as a means to introduce cross approximations, which can be seen as a sort of extension of EIM to a tensor framework.

Interpolation-based techniques were described in [17] as approximate-then-project approaches. Alternatively, project-then-approximate (also, hyper reduction) approaches propose roughly to approximate the projection of the non-linear term directly [18–20]. This ultimately leads to the definition of empirical quadrature rules [21], which seem to provide a more consistent basis to address non-linear reduced order modeling.

As opposed to a-posteriori ROM, literature on tensor-based ROM for non-linear problems is much scarcer and lacks, in general, of a systematic approach. For instance, only polynomial non-linearities can be efficiently addressed with the Asymptotical Numerical Method (ANM) [22–24]. The LATIN method has been successfully applied to compute space-time (or pseudo-time) separated representations involving elasto-plasticity, contact and multi-scale problems [25–27].

The objective of this paper is three-fold: in first place, we discuss the importance of

tensorization for the efficiency of tensor-based ROM. By *tensorization* we mean the ability to enforce a tensor representation of the model itself[1], which reveals crucial for the efficiency of tensor-based algorithms. Secondly, we investigate the adequacy of interpolation rather than projection-based approaches as a means to enforce tensorization, and propose the use of cross approximations [28,29] for models in moderate dimension. And finally, linearization of tensor problems is analyzed and several strategies for the tensor subspace construction are proposed. Specifically, the canonical tensor format is used here as a reference, in spite of some well-known limitations that shall be discussed in detail in Section 4.3. As a practical consequence of these, the use of canonical format should be restricted to second or third order tensors. However, in this paper we shall put emphasis on concepts rather than on technical details, and therefore the validity of the proposed approach should not be compromised by the choice of tensor format. Other tensor formats are likely to perform better in higher dimensional problems and might motivate future research.

The rest of the paper is organized as follows. In Section 2, we review the basic formulation of tensor methods and the importance of tensorization for their efficient implementation. To this end, tensor methods are formulated as an optimization problem. In Section 3, we consider a generic non-linear problem and show how tensorization is in general lost. We also analyze both projection and interpolation approaches as a means to recover tensorization. In Section 4, we present cross approximations as a means to enforce tensorization of non-linear models. Strategies for coupling a given linearization scheme with the incremental subspace construction are analyzed in Section 5. Finally, the performance of the proposed method is assessed in Section 6 and Section 7 by means of several numerical examples.

2 Basic formulation of tensor methods

Tensor methods build a low-rank tensor subspace for the solution of multi-dimensional and multi-parametric problems. To this end, the weak form of the model at hand is regarded as an optimization problem where the set of admissible solutions is constrained to a low-rank tensor subspace. The efficiency of tensor methods relies on the tensorization of the model at hand, as it allows splitting a multi-dimensional problem into a series of lower-dimensional ones. When dealing with non-linear models, tensorization is in general lost, thus compromising the overall efficiency of tensor methods.

In next lines, tensor-based reduced-order methods (ROM) are introduced from a-posteriori ones by drawing a parallel between both approaches.

[1] We shall use the term *tensorization* throughout the entire paper for the sake of conciseness.

2.1 From vector to tensor subspaces

To the purposes of this section, it is useful to consider a parametrized problem as a model case. Let $\boldsymbol{u}(\boldsymbol{\mu}) \in \mathbb{R}^{N_0}$ be its solution on a given representation basis of dimension N_0, for a given D-tuple of parameters $\boldsymbol{\mu} \in \mathcal{M}$. *A-posteriori* ROM split the computational cost into"offline-online" stages. In the offline stage, a subspace of lower dimension, $M \ll N_0$, is extracted from the statistical analysis of a training simulation dataset. Then, admissible solutions are constrained to this low-dimensional subspace:

$$\forall \boldsymbol{\mu} \in \mathcal{M}: \quad \boldsymbol{u}(\boldsymbol{\mu}) \approx \boldsymbol{u}_M(\boldsymbol{\mu}) := \mathbf{W}_0\, \boldsymbol{\alpha}(\boldsymbol{\mu}), \tag{1}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^M$ are the representation coefficients of the solution on the subspace spanned by the columns of $\mathbf{W}_0 \in \mathbb{R}^{N_0 \times M}$.

For an efficient offline-online splitting of the computational cost, restricting the solution to a subspace of dimension M should yield a reduced system equations whose complexity does not depend on N_0 anymore. However, this is not always the case, specially when dealing with non-linear models. In such a case, it is well-known that the non-linear terms of the equation need, in principle, to be fully evaluated and then projected at each iteration of the linearization scheme [13, 14]. A number of techniques, already reviewed in Section 1, have been proposed in order to address this situation.

In contrast with a-posteriori ROM, tensor methods are designed to compute the solution of parametrized problem as an explicit function of the parameters. Therefore, tensor methods do not produce a reduced system of equations but the parametric solution itself. In this regard, parameters are treated exactly the same as if they were coordinates, and consequently, the computational domain becomes of higher dimension. In practice, this implies that the solution is no longer represented by a one-dimensional array, as in Eq. (1), but by a multi-dimensional array (or tensor) instead.

Remark 1 (Notation for tensors)**.** *Although sometimes it is convenient to use multi-indices for tensors, here we prefer linear indexing in order to take advantage of standard matrix algebra operations, such as the Kronecker and Khatri-Rao products that will be used below.*

Let $\boldsymbol{u} \in \mathbb{R}^N$, $N := N_0 N_1 \cdots N_D$, be the *full tensor representation* of the parametric solution after discretizing the parametric domain $\mathcal{M}$ with a $N_1 \times \cdots \times N_D$ grid. Clearly, full representations are in general precluded even in moderate dimension. Therefore, tensor subspaces must be introduced to represent parametric solutions efficiently. For the sake of concretion, we shall only discuss here the canonical tensor subspace, which lies at the basis of the Proper Generalized Decomposition (PGD) method [9, 26, 30]. It is defined as follows:

$$T_M := \operatorname{span}\{\boldsymbol{w}_0^m \otimes \boldsymbol{w}_1^m \otimes \cdots \otimes \boldsymbol{w}_D^m\}_{m=1}^M, \tag{2}$$

where $\boldsymbol{w}_d^m \in \mathbb{R}^{N_d}$ stands for the m-th factor, or mode, in direction d, with $0 \leq d \leq D$. Note that T_M shall be understood throughout this paper as a non-linear structure, where

$\boldsymbol{w}_d$ can be chosen freely, unless indicated otherwise. A low-rank approximation $\boldsymbol{u}_M \in T_M$ writes as:

$$\boldsymbol{u} \approx \boldsymbol{u}_M := \sum_{m=1}^{M} \alpha_m \, \boldsymbol{w}_0^m \otimes \boldsymbol{w}_1^m \otimes \cdots \otimes \boldsymbol{w}_D^m,$$

commonly referred to as separated tensor representation. Using the Khatri-Rao product (i.e. column-wise Kronecker product), it can also be written as:

$$\boldsymbol{u}_M \equiv (\mathbf{W}_0 \odot \mathbf{W}_1 \odot \cdots \odot \mathbf{W}_D)\, \boldsymbol{\alpha}, \tag{3}$$

where columns of matrix $\mathbf{W}_d \in \mathbb{R}^{N_d \times M}$ span a subspace of dimension M, and its m-th column is the mode $\boldsymbol{w}_d^m$. In Eq. (3), "$\odot$" stands for the Khatri-Rao product. The array $\boldsymbol{\alpha} \in \mathbb{R}^M$ contains the representation coefficients of the parametric solution on the canonical tensor subspace of rank M. Notation based on the Khatri-Rao product will be preferred in this paper by its compactness.

Remark 2 (Tensor reconstruction and decomposition)**.** *Given a low-rank representation, Eq. (3), the reconstruction operation allows obtaining the full tensor representation, noted as* $\boldsymbol{u}_M \leftarrow \texttt{recons}(\boldsymbol{\alpha}, \mathbf{W}_0, \ldots, \mathbf{W}_D)$*. Conversely, decomposition is defined as* $\{\boldsymbol{\alpha}, \mathbf{W}_0, \ldots, \mathbf{W}_D\} \leftarrow \texttt{decomp}(\boldsymbol{u}_M)$*. While reconstruction is trivial, decomposition requires careful study [31], as it might not even be well-posed in some cases.*

2.2 Optimization problem for tensor subspace construction

Consider a multi-dimensional, linear, steady-state model. After discretization, it can be written as follows:

$$\mathbf{A}\boldsymbol{u} + \boldsymbol{f} = \mathbf{0}, \tag{4}$$

where $\boldsymbol{u} \in \mathbb{R}^N$ is a full tensor representation of some parametric solution, as it has been previously defined in Section 2.1. On the other hand, $\mathbf{A} \in \mathbb{R}^{N \times N}$ represents a linear operator and $\boldsymbol{f} \in \mathbb{R}^N$ is the independent term. Tensor methods are designed to build a tensor subspace by turning Eq. (4) into an optimization problem. No sampling of the parameter domain is in principle required.

In particular, PGD uses a greedy strategy to build a tensor subspace progressively by computing rank-one corrections, i.e. by building a series of nested subspaces:

$$T_1 \subset T_2 \subset \cdots \subset T_M \quad \text{where} \quad T_M := T_{M-1} + T_1. \tag{5}$$

In practice, the actual rank is driven by some error estimate [32, 33] able to determine when the solution subspace is accurate enough. In general, the greedy rank-one correction strategy will not yield a best rank-$(M+1)$ approximation, even if $\boldsymbol{u}_M$ is a best rank-M

approximation. Assuming that a rank-M tensor approximation of the solution is known, $\boldsymbol{u}_M \in T_M$, we seek a rank-one correction $\delta\boldsymbol{u} \in T_1$ such that:

$$\begin{gathered}\boldsymbol{u}_{M+1} := \boldsymbol{u}_M + \delta\boldsymbol{u} \quad \text{where} \\ \boldsymbol{u}_M = \bigodot_{d=0}^{D} \mathbf{W}_d\,\boldsymbol{\alpha} \;\text{ with }\; \boldsymbol{\alpha} \in \mathbb{R}^M,\; \mathbf{W}_d \in \mathbb{R}^{N_d \times M} \;\text{ and }\; \delta\boldsymbol{u} = \bigotimes_{d=0}^{D} \boldsymbol{w}_d.\end{gathered} \tag{6}$$

Assuming that the operator $\mathbf{A}$ is symmetric positive definite (specific formulations of PGD exist for non-symmetric problems, see [10, 34–36]), Eq. (4) can be regarded as an optimization problem where the set of admissible solutions is constrained to T_1 [30]:

$$\delta\boldsymbol{u} := \arg\min_{\boldsymbol{v}\in T_1} \frac{1}{2}\langle \mathbf{A}\boldsymbol{v}, \boldsymbol{v}\rangle + \langle \mathbf{A}\boldsymbol{u}_M, \boldsymbol{v}\rangle + \langle \boldsymbol{f}, \boldsymbol{v}\rangle, \tag{7}$$

where $\langle \bullet, \bullet \rangle$ stands for the scalar product in $\mathbb{R}^N$. Eq. (7) constitutes a non-linear optimization problem due to the tensor multiplicative structure of the subspace, see Eq. (2). For the efficient solution of Eq. (7), tensorization of $\mathbf{A}$ and $\boldsymbol{f}$ is essential.

2.3 Tensorization for the efficient optimization

Let us suppose that the following tensor representations are known:

$$\begin{aligned}\mathbf{A} &= \sum_{r=1}^{R} \bigotimes_{d=0}^{D} \mathbf{A}_d^r \qquad \text{with } \mathbf{A}_d^r \in \mathbb{R}^{N_d \times N_d} \quad \text{and} \\ \boldsymbol{f} &= \bigodot_{d=0}^{D} \mathbf{V}_d\,\boldsymbol{\gamma} \qquad \text{with } \boldsymbol{\gamma} \in \mathbb{R}^S,\;\; \mathbf{V}_d \in \mathbb{R}^{N_d \times S}.\end{aligned} \tag{8}$$

By inserting Eq. (8) into Eq. (7), and after some tedious but conceptually simple manipulations, we see that the scalar product in $\mathbb{R}^N$ can in fact be computed as the product of lower-dimensional scalar products. For instance, the first term in Eq. (7) writes:

$$\langle \mathbf{A}\boldsymbol{v}, \boldsymbol{v}\rangle \equiv \sum_{r=1}^{R} \prod_{d=0}^{D} \langle \mathbf{A}_d^r \boldsymbol{v}_d, \boldsymbol{v}_d\rangle_d, \tag{9}$$

where $\langle \bullet, \bullet \rangle_d$ stands for the scalar product in $\mathbb{R}^{N_d}$. Eq. (9) defines the separation property of the scalar product. This property suggests applying an alternating directions algorithm in order to optimize each direction $\boldsymbol{w}_d$ [37]. This can be achieved by simply projecting the functional in Eq. (7) onto each direction. Thus, let $\boldsymbol{w}_*$ the current direction to be optimized, implying that $\boldsymbol{w}_d$, $\forall d \neq *$, are frozen at their most current update. The restriction of Eq. (7) onto the current direction yields:

$$\boldsymbol{w}_* := \arg\min_{\boldsymbol{v}_* \in \mathbb{R}^{N_*}} \sum_{r=1}^{R} \left\{ \frac{1}{2}\langle \beta_*^r \mathbf{A}_*^r \boldsymbol{v}_*, \boldsymbol{v}_*\rangle_* + \langle \mathbf{A}_*^r \mathbf{W}_* \tilde{\boldsymbol{\alpha}}_*^r, \boldsymbol{v}_*\rangle_* \right\} + \langle \mathbf{V}_* \tilde{\boldsymbol{\gamma}}_*, \boldsymbol{v}_*\rangle_*, \tag{10}$$

where $\beta_*^r \in \mathbb{R}$, $\tilde{\boldsymbol{\alpha}}_*^r \in \mathbb{R}^M$ and $\tilde{\boldsymbol{\gamma}}_* \in \mathbb{R}^S$ are coefficients carrying the result of the scalar products in all directions except the current one. See Annex A for the definition of these coefficients as well as for a detailed solution of Eq. (10).

In summary, the correction $\delta\boldsymbol{u}$ can be computed by an alternating optimization of each separated factor, using Eq. (10), until stagnation [32, 38]. In this way, the algorithm splits a multi-dimensional problem into a series of low-dimensional ones. This has been possible thanks to the tensorization of the problem, introduced in Eq. (8). In many cases, tensorization of linear problems can be achieved quite easily. However, when dealing with non-linear models, tensorization is in general lost, thus compromising the overall efficiency of tensor methods.

3 Non-linear problem formulation

In this section we present a generic non-linear problem and discuss in detail the adequacy of interpolation rather than projection approaches to enforce tensorization. To this end, we simply transform the independent term in Eq. (4) into a non-linear function. This yields the following multi-dimensional, non-linear, steady-state model:

$$\mathbf{A}\boldsymbol{u} + \boldsymbol{f}(\boldsymbol{u}) = \mathbf{0}, \tag{11}$$

where $\boldsymbol{f}(\boldsymbol{u})$ is a non-linear function evaluated component-wise, that is $\boldsymbol{f}(\boldsymbol{u}) := [f(u_1) \cdots f(u_N)] \in \mathbb{R}^N$.

3.1 Tensorization is lost in non-linear problems

Let us consider, without loss of generality, a fixed-point linearization. Assuming that a rank-M tensor approximation of the solution at iteration ℓ is known, $\boldsymbol{u}_M^\ell \in T_M$, the next iterate can be computed from:

$$\mathbf{A}\boldsymbol{u}^{\ell+1} + \boldsymbol{f}(\boldsymbol{u}_M^\ell) = \mathbf{0}. \tag{12}$$

We seek to compute $\boldsymbol{u}_\star^{\ell+1} \in T_\star$, a good approximation of $\boldsymbol{u}^{\ell+1}$, where the rank will be driven by some error estimate [32,33]. As explained in Section 2.2, $T_\star$ can be built progressively by computing rank-one corrections. Assuming that $\boldsymbol{u}_{M'}^{\ell+1} \in T_{M'}$ has already been computed, a rank-one correction $\delta\boldsymbol{u} \in T_1$ can be computed from:

$$\delta\boldsymbol{u} := \arg\min_{\boldsymbol{v}\in T_1} \frac{1}{2}\langle \mathbf{A}\boldsymbol{v}, \boldsymbol{v}\rangle + \langle \mathbf{A}\boldsymbol{u}_{M'}^{\ell+1}, \boldsymbol{v}\rangle + \langle \boldsymbol{f}(\boldsymbol{u}_M^\ell), \boldsymbol{v}\rangle. \tag{13}$$

Even though $\boldsymbol{u}_M^\ell$ is a tensor form, no tensorized form of $\boldsymbol{f}(\boldsymbol{u}_M^\ell)$ is in general known. As a consequence, the non-linear term cannot be evaluated efficiently. Recall that it is essential for the efficiency of tensor-based algorithms to take advantage of the scalar

product separation property. If it holds, higher-dimensional scalar products in Eq. (13) could be computed as a series of lower-dimensional ones. Hence we require methods to compute a rank-S approximation of the non-linear term:

$$\boldsymbol{f}(\boldsymbol{u}_M^\ell) \approx \boldsymbol{f}_S \in T_S \quad \Leftrightarrow \quad \boldsymbol{f}_S = \bigodot_{d=0}^{D} \mathbf{V}_d \boldsymbol{\gamma} \text{ with } \boldsymbol{\gamma} \in \mathbb{R}^S, \ \mathbf{V}_d \in \mathbb{R}^{N_d \times S}, \tag{14}$$

where neither $\mathbf{V}_d$ nor $\boldsymbol{\gamma}$ are a priori known, i.e. both have to be computed for a given $\boldsymbol{u}_M^\ell$. The rank of the non-linear term approximation must not be confused with the rank of the independent term in Eq. (8), although we use the same letter: "S". These have nothing to do with each other. Note that if Eq. (14) was available, the problem would be completely equivalent to the one already studied in Section 2. Indeed, Eq. (14) could be inserted into Eq. (13), yielding an appropriately tensorized problem, which can be solved applying an alternating directions algorithm, as explained in Section 2.3. Therefore, the correction $\delta\boldsymbol{u}$ could be computed by an alternating optimization of each separated factor, provided that we dispose of Eq. (14). Unfortunately, it is by no means trivial to achieve a tensor representation such as Eq. (14).

Tensorization could be, in principle, recovered by means of a three-step procedure, involving *reconstruction-evaluation-decomposition* (RED) operations:

1. Reconstruction of the current iterate, yielding a full representation, as defined in Remark 2: $\boldsymbol{u}_M^\ell \leftarrow \texttt{recons}(\boldsymbol{\alpha}, \mathbf{W}_0, \dots, \mathbf{W}_D)$.
2. Evaluation of the non-linear term: $\boldsymbol{f}(\boldsymbol{u}_M^\ell) \leftarrow \texttt{nlfun}(\boldsymbol{u}_M^\ell)$.
3. Decomposition of the non-linear term into a rank-S tensor representation: $\{\boldsymbol{\gamma}, \mathbf{V}_0, \dots, \mathbf{V}_D\} \leftarrow \texttt{decomp}(\boldsymbol{f}(\boldsymbol{u}_M^\ell))$.

However, RED steps compromise the overall efficiency of tensor-based algorithms, as the manipulation of full representations is required. The choice of the decomposition method is crucial here. Some of the classical techniques that have been used within the PGD framework are the Singular Value Decomposition (SVD) [37, 39] and its variant for higher dimensions, the high-order SVD [40]. PGD itself has also been used as a decomposition technique [41]. These approaches are expensive from both memory and computational standpoints, as they require operating on the full tensor representation. In this paper we shall use interpolation-based algorithms that are able to adaptively determine the entries of the tensor that have to be read in order to compute a low-rank approximation. Therefore, the full representation of the tensor is not required, but only its evaluation at specific points.

3.2 Projection versus interpolation approaches

Coming back to Eq. (14), we seek an approximation $\boldsymbol{f}_S \in T_S$ of $\boldsymbol{f}(\boldsymbol{u}_M^\ell)$. Let us assume that we have somehow managed to compute factor matrices $\mathbf{V}_d$, i.e. T_S can be now un-

derstood as a linear subspace, and we only need to compute the representation coefficients $\boldsymbol{\gamma}$. Therefore, we dispose of $\mathbf{V} := \bigodot_{d=0}^{D} \mathbf{V}_d \in \mathbb{R}^{N \times S}$. Even in such a favorable scenario, computing $\boldsymbol{\gamma}$ using projection techniques is expensive. The projection problem is defined as follows:

$$\boldsymbol{\gamma} := \arg\min_{\boldsymbol{\omega} \in \mathbb{R}^S} \langle \mathbf{V}\,\boldsymbol{\omega}, \mathbf{V}\,\boldsymbol{\omega} \rangle - \langle \boldsymbol{f}(\boldsymbol{u}_M^\ell), \mathbf{V}\,\boldsymbol{\omega} \rangle, \tag{15}$$

which yields the following normal equations:

$$\bigcirc_{d=0}^{D} \mathbf{V}_d^T \mathbf{V}_d\, \boldsymbol{\gamma} = \mathbf{V}^T \boldsymbol{f}(\boldsymbol{u}_M^\ell), \tag{16}$$

where $\bigcirc_{d=0}^{D}$ is performs the Hadamard (i.e. component-wise) product. While $\mathbf{V}^T\mathbf{V}$ can be efficiently computed, thanks to the tensor structure of the basis, as indicated in the left-hand side of Eq. (16), the right-hand side requires a full evaluation of the non-linear term. It is worth to recall that if matrices $\mathbf{V}_d$ were orthonormal, the left-hand side computation could be encompassed. Therefore, we see that in spite of the technicality of the tensor framework, when it comes to the evaluation of the non-linear term, the same kind of issues are encountered in both a-posteriori ROM and tensor-based framework.

Inspired by EIM, interpolation-based approaches could be envisaged. In its discrete form, EIM returns a set of interpolation indices, $\boldsymbol{\varrho} \in \mathbb{N}^S$, from the basis $\mathbf{V}$. This would allow defining the following interpolation problem:

$$\mathbf{P}^T\mathbf{V}\boldsymbol{\gamma} = \mathbf{P}^T\boldsymbol{f}(\boldsymbol{u}_M^\ell) \quad \rightarrow \quad \boldsymbol{f}(\boldsymbol{u}_M^\ell) \approx \mathbf{V}(\mathbf{P}^T\mathbf{V})^{-1}\mathbf{P}^T\boldsymbol{f}(\boldsymbol{u}_M^\ell), \tag{17}$$

where $\mathbf{P} \in \mathbb{R}^{N \times S}$ is an extractor matrix full of zeros, with the exception of entries $P_{\varrho_i, i} = 1$. Indices $\boldsymbol{\varrho}$ are such that $\mathbf{P}^T\mathbf{V} \in \mathbb{R}^{S \times S}$ is invertible. As opposed to projection, interpolation allows approximating the non-linear term by evaluating only few points, which can be done inexpensively.

However, a detailed examination of the algorithm yielding the interpolation indices reveals practical issues for its implementation in a tensor framework. Without going into details, the algorithm selects the interpolation indices at the absolute maximum value of some approximation residual, see [12, 14]. When working with tensors, the `max` function must be used carefully, in order to avoid full tensor complexity.

Additionally, recall that a suitable tensor approximation basis has been assumed to be available, which is often not the case. An alternative approach able to deliver both a tensor interpolation basis and the interpolation indices is required. Full inspection of the non-linear term is precluded.

4 Enforcing tensorization with cross approximations

In this section, cross approximations are presented as a means to tensorize the non-linear term. Cross approximations were developed first for matrix approximations [28] and then

extended to the tensor framework [29]. When working with canonical tensor subspaces, as it is the case of this paper, cross approximations are subsidiary of the properties of this tensor subset: linear complexity in the representation but ill-posedness of the best approximation problem [31,42]. Limitations derived from this fact, as well as perspectives for future research based on other tensor formats, are discussed in Section 4.3. Although cross approximations were developed independently from EIM, the exposition is made in such a way that resemblances between both methods are highlighted.

Some relevant features of the cross approximations method are:

- It builds a canonical tensor subspace with interpolation properties. The interpolation indices are provided too. Both are computed simultaneously using a fiber search algorithm.
- The tensor subspace is constructed progressively by adding rank-one corrections.
- Like the EIM, the interpolation indices are placed where the residual of the approximation is maximum. Furthermore, they are nested.
- The full tensor representation is not required, but only its evaluation at specific points.

4.1 Non-linear term approximation via interpolation

Cross approximations build both a rank-$\star$ subspace, $T_\star$, and a set of interpolation indices $\mathcal{I}_\star := \{\varrho_s\}_{s=1}^{\star}$, such that $\boldsymbol{f}_\star \in T_\star$ interpolates $\boldsymbol{f}(\boldsymbol{u}_M^\ell)$ at $\mathcal{I}_\star$. Here, "$\star$" is the rank of the approximation, to be determined adaptively by the cross approximations algorithm. Note that $\mathcal{I}_\star$ contains linear indices, i.e. $1 \leq \varrho_s \leq N$. However, for notational convenience, in this section we will also use multi-indices, i.e. $\mathcal{H}_\star := \{\boldsymbol{\rho}_s\}_{s=1}^{\star}$, with $\boldsymbol{\rho}_s \in \mathbb{N}^{D+1}$ and $1 \leq \rho_s^d \leq N_d$.

As stated in the beginning of this section, cross approximations build $T_\star$ progressively by adding rank-one corrections. Therefore, let us assume that T_S has already been computed, and by consequence, matrices $\mathbf{V}_d \in \mathbb{R}^{N_d \times S}$ and interpolation coefficients $\mathcal{H}_S$ (also, $\mathcal{I}_S$) are available. A rank-S approximation, $\boldsymbol{f}_S := (\bigodot_{d=0}^{D} \mathbf{V}_d)\,\boldsymbol{\gamma}$, can be readily computed by solving the following interpolation problem:

$$\bigodot_{d=0}^{D} \mathbf{P}_d^T \mathbf{V}_d \,\boldsymbol{\gamma} = \mathbf{P}^T \boldsymbol{f}(\boldsymbol{u}_M^\ell), \tag{18}$$

where $\mathbf{P}_d \in \mathbb{R}^{N_d \times S}$ are extractor matrices full of zeros, with the exception of entries (ρ_s^d, s), which are populated with ones. On the other hand, $\mathbf{P} = \bigodot_{d=0}^{D} \mathbf{P}_d \in \mathbb{R}^{N \times S}$.

Remark 3 (Non-linear term evaluation)**.** *From the non-linear term definition in Section 3, we have that the right-hand side of Eq. (18) can be written as follows:*

$$\mathbf{P}^T \boldsymbol{f}(\boldsymbol{u}_M^\ell) \equiv \boldsymbol{f}(\mathbf{P}^T \boldsymbol{u}_M^\ell) \equiv \boldsymbol{f}\left(\left\{\bigodot_{d=0}^{D} \mathbf{P}_d^T \mathbf{W}_d\right\} \boldsymbol{\alpha}\right).$$

It is therefore clear that $\boldsymbol{u}_M^\ell$ *has to be evaluated at only* S *indices.*

4.2 Simultaneous construction of both interpolation basis and indices

From the knowledge of $\boldsymbol{f}_S$, we seek a rank-one correction $\delta\boldsymbol{f} \in T_1$ as well as its corresponding interpolation index, ϱ, such that:

$$\begin{aligned} \boldsymbol{f}_{S+1} &:= \boldsymbol{f}_S + \delta\boldsymbol{f} \\ \mathcal{I}_{S+1} &:= \mathcal{I}_S \oplus \varrho, \end{aligned} \tag{19}$$

where $\delta\boldsymbol{f} = \bigotimes_{d=0}^{D} \boldsymbol{v}_d$. Recall that ϱ can be readily converted into a multi-index $\boldsymbol{\rho}$ such that $\mathcal{H}_{S+1} := \mathcal{H}_S \oplus \boldsymbol{\rho}$. Like EIM, cross approximations seek to place the interpolation index ϱ where the residual of the current rank-S approximation reaches its maximum value:

$$\varrho := \arg\max |\boldsymbol{r}| \quad \text{with} \quad \boldsymbol{r} := \boldsymbol{f}(\boldsymbol{u}_M^\ell) - \boldsymbol{f}_S. \tag{20}$$

Since searching for the maximum of the approximation's residual is computationally expensive, a fiber search algorithm (presented hereafter) is preferred instead. Starting with a random choice for the multi-index $\boldsymbol{\rho}$, we loop over its components seeking for the maximum of the residual restricted to the current component. The location where the maximum of the residual is found allows updating the current multi-index component. Having completed this iteration over all components, we obtain a new multi-index $\boldsymbol{\rho}$. Fig. 1 illustrates this process in a three-dimensional tensor. It shows the multi-index $\boldsymbol{\rho}$ optimized over a *cross*, see Fig. 1d, and its associated *fibers*, depicted in Fig. 1a, Fig. 1b and Fig. 1c for each one of the three directions.

Let ρ_* be the current direction, implying that ρ_d, $\forall d \neq *$ are frozen at their most current update. The restriction of Eq. (20) onto the current fiber yields:

$$\rho_* := \arg\max |\boldsymbol{r}_*| \quad \text{with} \quad \boldsymbol{r}_* := \boldsymbol{f}(\mathbf{W}_* \, \tilde{\boldsymbol{\alpha}}_*) - \mathbf{V}_* \, \tilde{\boldsymbol{\gamma}}_*. \tag{21}$$

See Annex B for the definition of $\tilde{\boldsymbol{\alpha}}_* \in \mathbb{R}^M$ and $\tilde{\boldsymbol{\gamma}}_* \in \mathbb{R}^S$. By iterating along all directions, Eq. (21) allows updating each component of the multi-index $\boldsymbol{\rho}$, until stagnation. The number of evaluations to be performed at each alternated fiber search is relatively small, as the size of the residual's restriction on a fiber is $\boldsymbol{r}_* \in \mathbb{R}^{N_*}$.

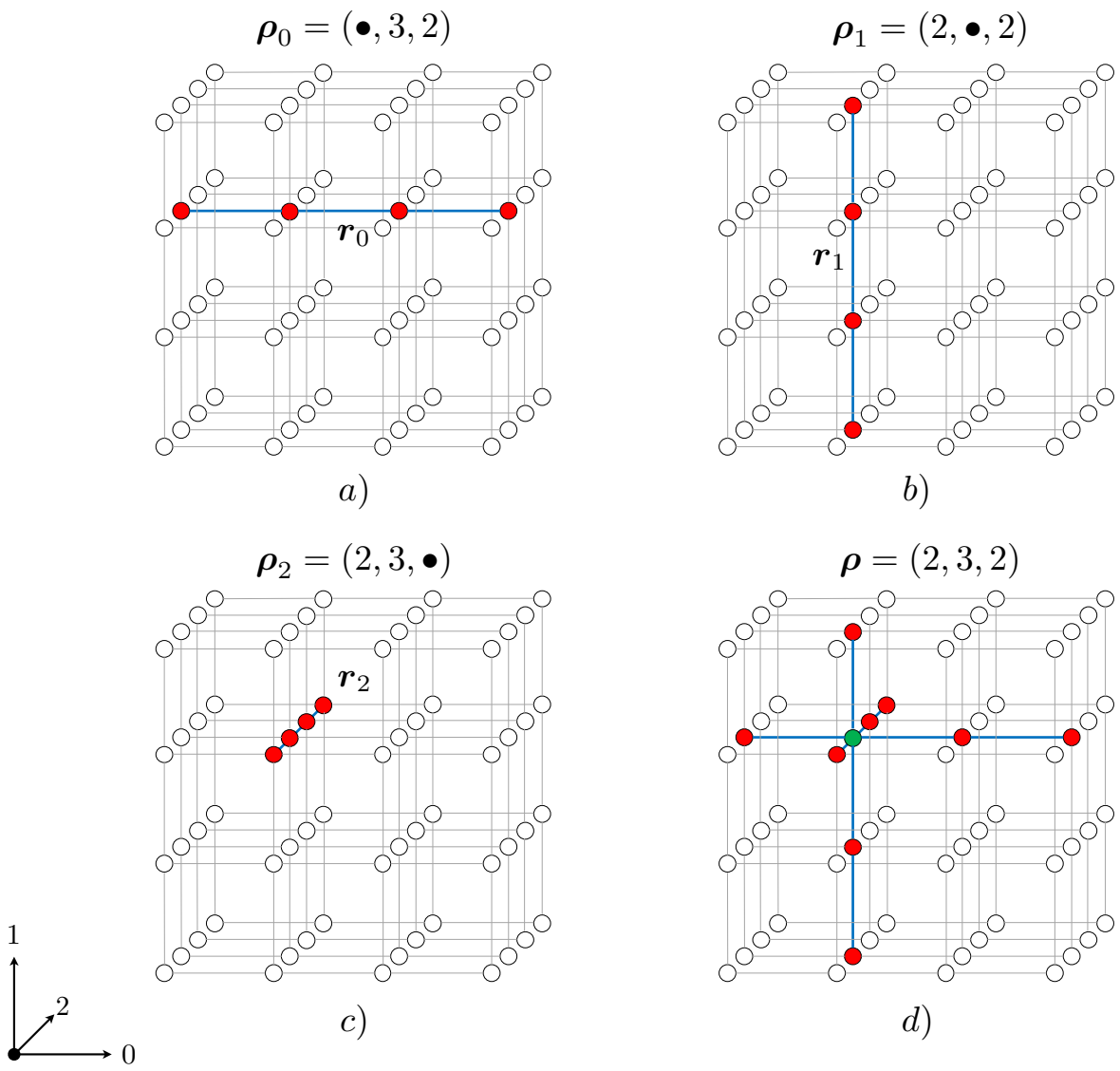


Figure 1: Illustration of tensor fibers and crosses definition, for a given multi-index $\boldsymbol{\rho}$. The residual restriction onto its corresponding fibers, $\boldsymbol{r}_*$, are also indicated.

The just described cross approximation algorithm will not succeed, in general dimension, to find the maximum of the residual. Therefore, $\boldsymbol{\rho} \equiv \varrho \neq \arg\max |\boldsymbol{r}|$. Further details are given in Section 4.3.

The interpolation basis is defined from the fiber cross associated to the interpolation index: we set $\boldsymbol{v}_d := \boldsymbol{r}_d$, with $0 \leq d \leq D$. Usually, the modes are normalized such that the ρ_d-th entry of $\boldsymbol{v}_d$ is equal to one. However, no distinct notation for normalized modes is used for the sake of simplicity. With the interpolation basis and the interpolation index at hand, the definition of the rank-one correction $\delta\boldsymbol{f}$ is completed. Coefficients $\boldsymbol{\gamma} \in \mathbb{R}^{S+1}$ can be readily computed by solving the interpolation problem defined in Eq. (18).

The final rank of the approximation is determined by looking at the maximum of the residual, which should in principle be located at the interpolation index. For some prescribed tolerance, the stop criterion is as follows:

$$\left| f\left(\left\{ \bigodot_{d=0}^{D} \mathbf{P}_d^T \mathbf{W}_d \right\} \boldsymbol{\alpha} \right) - \left\{ \bigodot_{d=0}^{D} \mathbf{P}_d^T \mathbf{V}_d \right\} \boldsymbol{\gamma} \right| < \texttt{tol}, \tag{22}$$

where $\mathbf{P}_d \in \mathbb{R}^{N_d \times 1}$ is an extractor matrices full of zeros, except the entry ρ_d which is equal to one. Due to some issues discussed below, the criterion given in Eq. (22) may not behave consistently. In practice, we prefer to evaluate the residual at a *test set* of indices, and not only at the interpolation index, in order to ensure the approximation is below the specified tolerance.

4.3 Discussion and alternative tensor formats

In this section, limitations inherited from the canonical tensor format shall be discussed, and perspectives on the use of other tensor formats as a basis for future research shall be given.

Compared to other formats, the canonical format suffers from ill-posedness of the best approximation problem in general dimension [31, 42]. As a consequence, it is not possible to recover exactly a low-rank tensor from samples [43]. In two-dimensional problems, it can be proven that $\delta\boldsymbol{f}$ interpolates $\boldsymbol{f}(\boldsymbol{u}_M^\ell)$ at the cross defined by the new interpolation index $\boldsymbol{\rho}$ [29, 43]. Equivalently, the residual vanishes at the cross. From a practical point of view, this property implies that the interpolation matrix defined in the left-hand side of Eq. (18) is lower triangular. However, in dimension higher than two, the interpolation property on the crosses is lost, i.e. the residual may not vanish on three or higher dimensional crosses. As a consequence, cross approximations as formulated in Section 4.2 are likely to yield tensor decompositions of an unnecessarily high rank, see [29, 43] for details. A PGD projection step can be used in order to control the rank of the approximation, as discussed in Section 4.4. Some refinements related to the fiber search algorithm are also provided in the previous references. In our experience, cross approximations as formulated in Section 4.2 deliver reasonably good results in moderate dimension (two or three order tensors).

Potentially, this would allow addressing problems defined in up to 6-D or 9-D domains, assuming that each tensor direction corresponds to the discretization of a 3-D domain.

In order to overcome limitations of the canonical format, cross approximations were extended to other tensor formats, such as Tucker [44], Tensor Train [45] and more general tree-based Hierarchical Tucker tensors [46]. We refer the interested reader to [47–50]. Contrary to the canonical format, the aforementioned formats allow recovering exactly a low-rank tensor from samples in general dimension. These could be used as a means to enforce tensorization, as suggested in this paper, and coupled with efficient solvers for high-dimensional systems [51, 52], other than the PGD used in this paper. These are valuable references that might motivate future research and extension of the ideas discussed in this paper.

4.4 Compressed rank via PGD projection

Cross approximations as formulated in 4.2 may deliver tensor decompositions of an unnecessarily high rank. Therefore, in this section we propose to use a PGD projection step in order to compute a rank as small as possible. Note that, in general, the compressed rank shall not be the exact rank, since PGD relies on the canonical tensor format, see Section 4.3 for details. The projection problem is formulated as in Eq. (15), but letting the approximation basis as an unknown to be optimized with the PGD method. Another difference is that in this section we do not seek to project $\boldsymbol{f}(\boldsymbol{u}_M^\ell)$, but $\boldsymbol{f}_S \in T_S$ instead, which has tensor structure. This yields a conveniently tensorized PGD projection problem, as it shall be shown below.

Therefore, we seek to build a lower-rank subspace, $T_\star$, with $\star \leq S$, such that $\boldsymbol{f}_\star \in T_\star$ is equal to $\boldsymbol{f}_S$, up to some desired tolerance. Let us assume that $\boldsymbol{f}_Q$ has already been computed and we seek a rank-one correction $\delta\boldsymbol{f} \in T_1$ such that

$$\begin{gathered} \boldsymbol{f}_{Q+1} := \boldsymbol{f}_Q + \delta\boldsymbol{f} \quad \text{where} \\ \boldsymbol{f}_Q = \bigodot_{d=0}^{D} \mathbf{V}'\boldsymbol{\gamma}' \;\text{ with }\; \boldsymbol{\gamma}' \in \mathbb{R}^Q,\; \mathbf{V}'_d \in \mathbb{R}^{N_d \times Q} \;\text{ and }\; \delta\boldsymbol{f} = \bigotimes_{d=0}^{D} \boldsymbol{v}'_d. \end{gathered} \tag{23}$$

The projection problem is formulated as follows:

$$\delta\boldsymbol{f} := \arg\min_{\boldsymbol{z}\in T_1} \langle \boldsymbol{z}, \boldsymbol{z}\rangle + \langle \boldsymbol{f}_Q - \boldsymbol{f}_S, \boldsymbol{z}\rangle, \tag{24}$$

Since all quantities in Eq. (25) are tensorized, an alternating directions algorithm can be applied straightforwardly in order optimize each $\boldsymbol{v}'_d$. Assuming that the current direction being optimized is $\boldsymbol{v}'_*$:

$$\boldsymbol{v}'_* := \arg\min_{\boldsymbol{z}_*\in\mathbb{R}^{N_*}} \langle a_*\boldsymbol{z}_*, \boldsymbol{z}_*\rangle_* + \langle \mathbf{V}'_*\tilde{\boldsymbol{\gamma}}'_* - \mathbf{V}_*\tilde{\boldsymbol{\gamma}}_*, \boldsymbol{z}_*\rangle_*, \tag{25}$$

where coefficients $a_* \in \mathbb{R}$, $\tilde{\boldsymbol{\gamma}}'_* \in \mathbb{R}^Q$ and $\tilde{\boldsymbol{\gamma}}_* \in \mathbb{R}^S$ can be deduced applying the same methodology already shown in Appendix A, and therefore, their definition is not given explicitly for the sake of brevity.

5 Linearization and incremental subspace construction

Both the solution and the non-linear subspaces are built incrementally. This yields several strategies for coupling a given linearization scheme with the incremental subspace construction. In next lines, we choose for the sake of clarity a fixed-point linearization and analyze several strategies.

Consider the non-linear problem already introduced in Eq. (11). The following strategies for coupling the linearization scheme with the incremental subspace construction are possible.

5.1 The *outer linearization* strategy

The first and most intuitive option consists in first linearizing the problem at hand, then applying a tensor method to solve the resulting linearized problem. Therefore, we have essentially two nested loops: one for the linearization scheme (the outer loop) and another one for the PGD rank-one corrections (the inner loop). The outer loop iterations are indexed by ℓ.

For the sake of notational clarity, we denote in this section $\boldsymbol{u}^\ell := \boldsymbol{u}_{M_\ell} \in T_{M_\ell}$, that is, a rank-$M_\ell$ approximation of the solution at iteration ℓ. Likewise the non-linear term is denoted as $\boldsymbol{f}^\ell := \boldsymbol{f}_{S_\ell} \in T_{S_\ell}$. Therefore, the previous iterate is $\boldsymbol{u}^{\ell-1}$. After tensorization of the non-linear term evaluated at the previous iteration, $\boldsymbol{f}^{\ell-1}$, the next iterate $\boldsymbol{u}^\ell$ can be computed from the solution of:

$$\mathbf{A}\boldsymbol{u}^\ell + \boldsymbol{f}^{\ell-1} = \mathbf{0}. \tag{26}$$

A tensor method can be used to solve Eq. (26). Observe that here the tensor method is regarded merely as a linear solver. Algorithm 1 summarizes the just described strategy.

Observe that the outer linearization strategy as described in Algorithm 1 builds a tensor subspace T_{M_ℓ} from scratch. A possible variant is to reuse the subspace of the previous iterate by setting: $T_{M_\ell} := T_{M_{\ell-1}} + T_{\tilde{M}}$, where $T_{\tilde{M}}$ is a rank-$\tilde{M}$ correction. As we converge, we expect $\boldsymbol{u}^\ell \approx \boldsymbol{u}^{\ell-1}$, and then we can hope $\tilde{M} \ll M_\ell$. Consequently, much less terms per iteration would be need to be computed. This variant of the outer linearization scheme may tend to cumulate terms unnecessarily during intermediate iterations. A compressed rank can be obtained via PGD projection, as explained in Section 4.4.

Algorithm 1 The outer linearization strategy

Set $\ell = 0$ and `flag` = true
Solution initialization: $\boldsymbol{u}^{\ell}$
while ($\ell \leq \ell_{\texttt{max}}$ & `flag`) **do** ▷ Outer loop
    Set $\ell = \ell + 1$
    Tensorize non-linear term: $\boldsymbol{f}^{\ell-1} \approx \boldsymbol{f}(\boldsymbol{u}^{\ell-1})$ ▷ Cross Approx.
    New iterate: $\boldsymbol{u}^{\ell} \leftarrow$ `PGD` on Eq. (26) ▷ Inner loop
    if Convergence **then**
        Set `flag` = false
return $\boldsymbol{u}^{\ell} \in T_{M_\ell}$

5.2 The *inner linearization* strategy

In this strategy, the order of the outer and inner loops defined in Section 5.1 is interchanged: the linearization loop becomes the inner loop while the PGD rank-one corrections loop becomes the outer loop. Therefore, suppose that we have already built $\boldsymbol{u}_M \in T_M$ and we want to compute a rank-one correction:

$$\boldsymbol{u}_{M+1} := \boldsymbol{u}_M + \delta\boldsymbol{u}. \tag{27}$$

Introducing Eq. (27) into Eq. (11), we have

$$\mathbf{A}\delta\boldsymbol{u} + \mathbf{A}\boldsymbol{u}_M + \boldsymbol{f}(\boldsymbol{u}_M + \delta\boldsymbol{u}) = \mathbf{0}, \tag{28}$$

that can be linearized around the correction. We denote by $\delta\boldsymbol{u}^{\ell}$ the ℓ-th iterate of the correction $\delta\boldsymbol{u}$. The fixed-point linearization yields the following approximation: $\boldsymbol{f}(\boldsymbol{u}_M + \delta\boldsymbol{u}) \approx \boldsymbol{f}(\boldsymbol{u}_M + \delta\boldsymbol{u}^{\ell-1})$, that is, the non-linear term is evaluated at the previous iterate of the rank-one correction. Next, the non-linear term is tensorized: $\boldsymbol{f}^{\ell-1} := \boldsymbol{f}_{S_{\ell-1}} \approx \boldsymbol{f}(\boldsymbol{u}_M + \delta\boldsymbol{u}^{\ell-1})$. The current iterate of the rank-one correction can be computed from the solution of:

$$\mathbf{A}\delta\boldsymbol{u}^{\ell} + \mathbf{A}\boldsymbol{u}_M + \boldsymbol{f}^{\ell-1} = \mathbf{0}. \tag{29}$$

Eq. (29) can be solved with an alternating direction algorithm. The just described strategy is summarized in Algorithm 2.

Observe that the inner linearization strategy as described in Algorithm 2 builds a subspace T_M from rank-one corrections, each one of them being iterated until convergence of the linearization. In doing so, the non-linear term needs to be tensorized several times (until convergence of the linearization scheme) for each rank-one correction. A possible variant is to reuse the non-linear term subspace from one iteration to another by setting: $T_{S_\ell} := T_{S_{\ell-1}} + T_{\tilde{S}}$. As rank-one correction converges, we expect $\delta\boldsymbol{u}^{\ell} \approx \delta\boldsymbol{u}^{\ell-1}$ and hope that $\tilde{S} \ll S_\ell$.

In the following lines, we briefly summarize some of the convergence properties observed from the numerical experiments carried out in Section 6. See Section 6.2 for a detailed

Algorithm 2 The inner linearization strategy

Set $M = 0$ and `flagPgd` = true
Solution initialization: $\boldsymbol{u}_M \equiv \boldsymbol{0}$
while ($M \leq M_{\texttt{max}}$ & `flagPgd`) **do** ▷ Outer loop
    Set $M = M + 1$
    Set $\ell = 0$ and `flagNl` = true
    Initialize $\delta \boldsymbol{u}^{\ell}$
    while ($\ell \leq \ell_{\texttt{max}}$ & `flagNl`) **do** ▷ Inner loop
        Set $\ell = \ell + 1$
        Compute $\boldsymbol{f}^{\ell-1} \approx \boldsymbol{f}(\boldsymbol{u}_M + \delta \boldsymbol{u}^{\ell-1})$ ▷ Using cross approximations
        $\delta \boldsymbol{u}^{\ell} \leftarrow$ `ADA` on Eq. (29) ▷ Alternating Dir. Algorithm
        if Convergence **then**
            Set `flagNl` = false
    $\boldsymbol{u}_M \leftarrow \boldsymbol{u}_M + \delta \boldsymbol{u}^{\ell}$
    if Convergence **then**
        Set `flagPgd` = false
return $\boldsymbol{u}_M \in T_M$

comparison between outer and inner schemes. As a matter of fact, both schemes showed quite different behavior. While the outer scheme converged monotonically at a constant rate, the inner scheme behaved less smoothly, and in some cases, the correction did not even converge. However, in spite of this lack of robustness, the global convergence of the inner strategy seemed to be faster. Specifically, the inner scheme required a smaller number of calls to the alternating directions algorithm. The number of calls to the fiber search algorithm was also counted for both outer and inner schemes. Results showed that the outer scheme required significantly less calls. As the computational cost tends to be dominated by the alternating directions algorithm rather than by the fiber search algorithm, the inner strategy seemed to be globally faster.

6 A non-linear 2-D steady state problem

In this section we provide a first numerical example in order to illustrate and assess the performance of the algorithms discussed in previous sections.

Let $u := u(x, y)$ be the solution the following non-linear 2-D steady state problem, defined in a unit square domain:

$$\begin{cases} -\nabla^2 u + \sigma \sin(2\pi u) = s & \text{in} \quad \Omega := [0,1]^2 \\ u = 0 & \text{on} \quad \partial\Omega, \end{cases} \tag{30}$$

where the source term is defined as $s := s(x, y) = 100 \sin(2\pi x) \sin(2\pi y)$. On the other hand, σ stands for the magnitude of the non-linear term. We shall consider four different

values: $\sigma \in \{1, 5, 10, 15\}$ as a means to analyze the convergence of different linearization strategies. In what follows, we associate x and y with subscripts 0 and 1, respectively.

A rank-M approximation $\boldsymbol{u} \approx \boldsymbol{u}_M := (\mathbf{W}_0 \odot \mathbf{W}_1)\,\boldsymbol{\alpha}$ is sought, where $\boldsymbol{u}$ stands for the discretized version of u. Both directions x and y are discretized with $N_0 = N_1 = 100$ nodes, i.e. 99 two-nodes linear finite elements per direction. This yields an equivalent 2-D finite element mesh with 99×99 bi-linear, four-nodes quadrangular elements (only used for visualization purposes).

6.1 Illustrating cross approximations

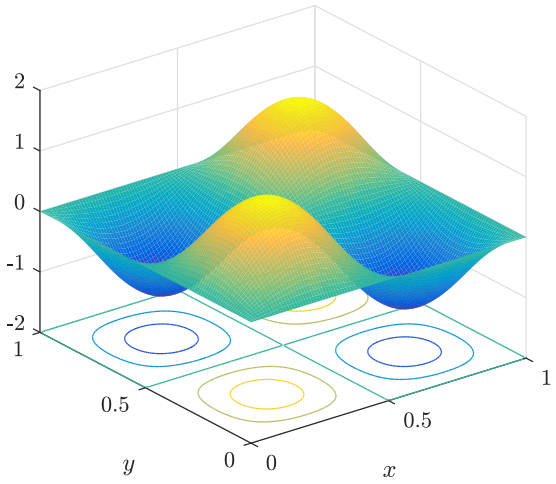
To illustrate cross approximations, we consider $\sigma = 10$ and an outer linearization strategy, see Section 5.1. Convergence is measured with the relative norm-2 of the residual, with respect to the first iterate. Stopping tolerance is set to 10^{-6}. In these conditions, the solution is obtained after $\ell = 11$ non-linear iterations, and the final rank of the solution is $M = 19$. Fig. 2b and Fig. 2c show, respectively, the first three modes (unit norm) of the solution in 0-direction $\{\boldsymbol{w}_0^m\}_{m=1}^3$ and 1-direction $\{\boldsymbol{w}_1^m\}_{m=1}^3$. Fig. 2a shows the 2-D reconstruction of the solution.

We illustrate now the tensorization using cross approximations of the non-linear term in the last iteration before convergence, $\ell = 11$. We denote by $\boldsymbol{f}_S := (\mathbf{V}_0 \odot \mathbf{V}_1)\,\boldsymbol{\gamma}$ an approximation of the non-linear term in Eq. (30). The convergence tolerance of cross approximations is set to 10^{-9}, yielding an approximation rank $S = 14$. Fig. 3c and Fig. 3d show, respectively, the first three modes (unit norm) in 0-direction $\{\boldsymbol{v}_0^s\}_{s=1}^3$ and 1-direction $\{\boldsymbol{v}_1^s\}_{s=1}^3$. The interpolation indices associated to all 14 modes are depicted in Fig. 3b. On the other hand, Fig. 3a shows the reconstruction of the non-linear term's approximation.

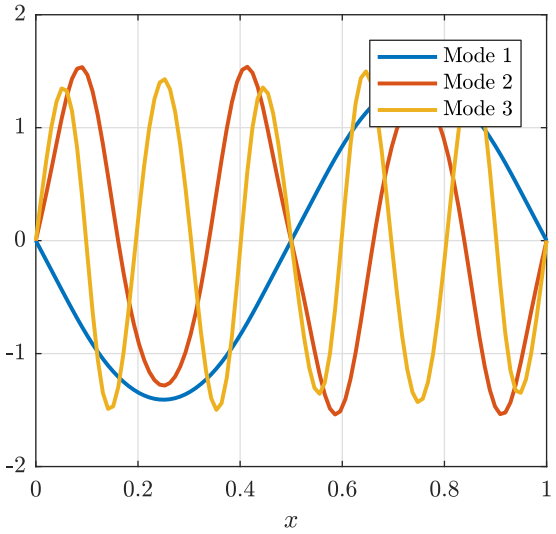
We shall now illustrate the incremental construction of the non-linear term subspace. Fig. 4a shows the 2-D reconstruction of the initial residual, that is, when $S = 0$. The maximum of the residual is sought using a fiber search algorithm which is initialized with a random index. After 4 iterations, the algorithm converges to the point $\boldsymbol{\rho}_0 = (0.54, 0.19)$, depicted in red in Fig. 4a, as well as its associated cross. Fig. 4b and Fig. 4c show the residual after one term and two terms have been computed, respectively, as well as their corresponding interpolation indices. Finally, Fig. 4d shows that the final residual is in the order of 10^{-9}.

6.2 Comparison of linearization strategies

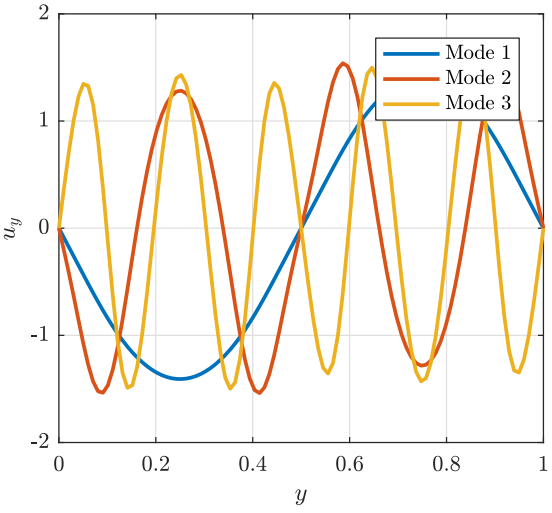
The outer and inner linearization strategies defined in Section 5 are discussed here. To this end, we introduce a quantity measuring the cumulated number of mode computations, or equivalently, the total number of calls to the alternating directions algorithm (ADA)



(a) Reconstructed 2-D view of the solution: $\boldsymbol{u}_M^\ell$

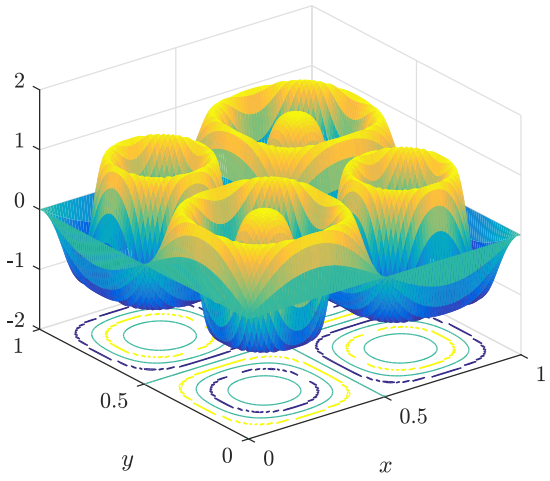


(b) First 3 x-dir. modes: $\{\boldsymbol{w}_0^m\}_{m=1}^3$

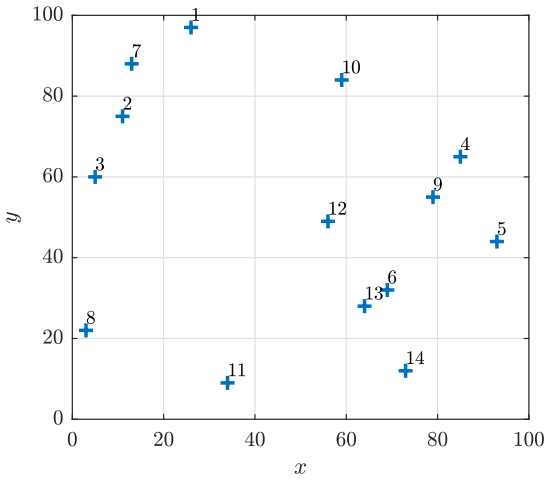


(c) First 3 y-dir. modes: $\{\boldsymbol{w}_1^m\}_{m=1}^3$

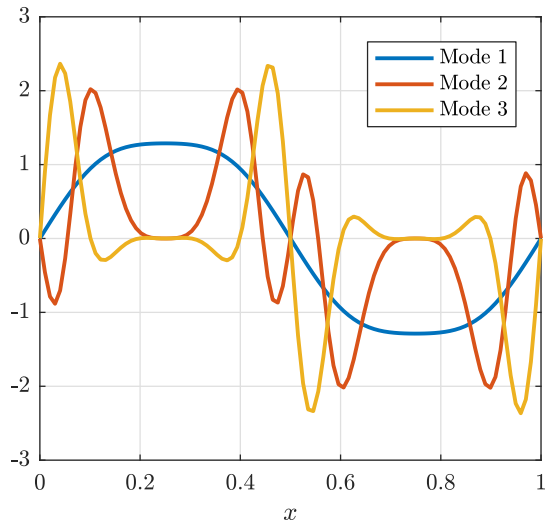
Figure 2: PGD solution ($\sigma = 10$) at last iteration ($\ell = 11$), using an outer linearization strategy. Solution's rank is $M = 19$.



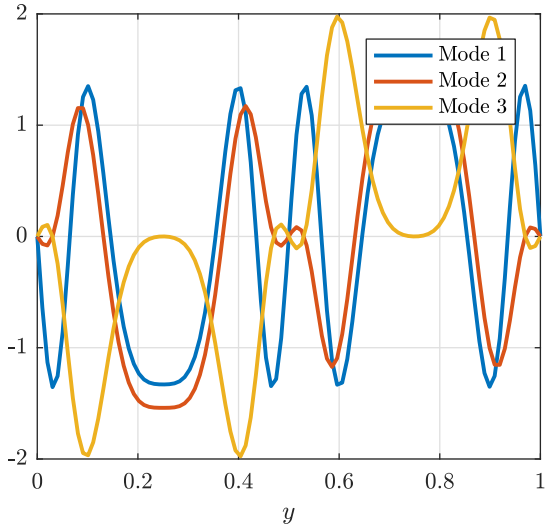
(a) Reconstructed 2-D view of the non-linear term: $\boldsymbol{f}_S \approx \boldsymbol{f}(\boldsymbol{u}_M^{\ell-1})$



(b) Set of interpolation indices: $\mathcal{H}_S$

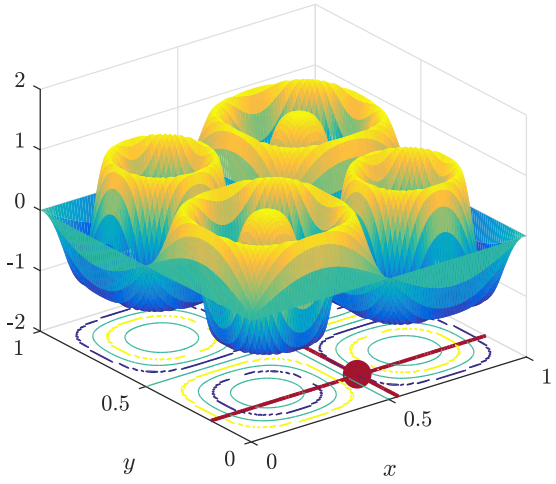


(c) First 3 x-dir. modes: $\{\boldsymbol{v}_0^s\}_{s=1}^3$

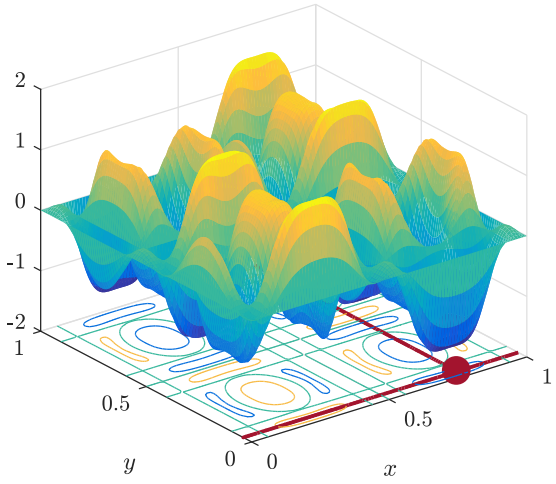


(d) First 3 y-dir. modes: $\{\boldsymbol{v}_1^s\}_{s=1}^3$

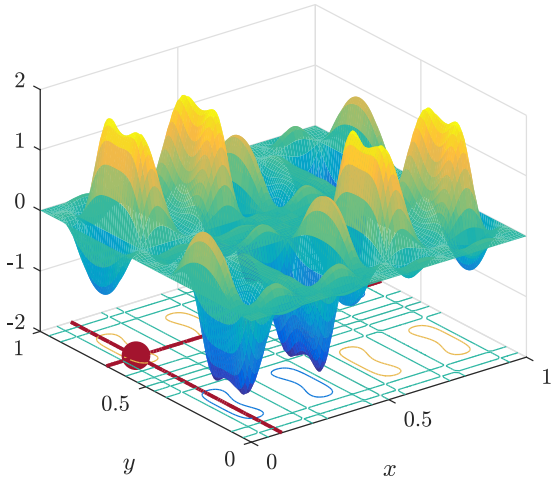
Figure 3: Cross Approximation of the non-linear term ($\sigma = 10$) at last iteration ($\ell = 11$), using an outer linearization strategy. Solution's rank is $M = 19$. Non-linear term's rank is $S = 14$.



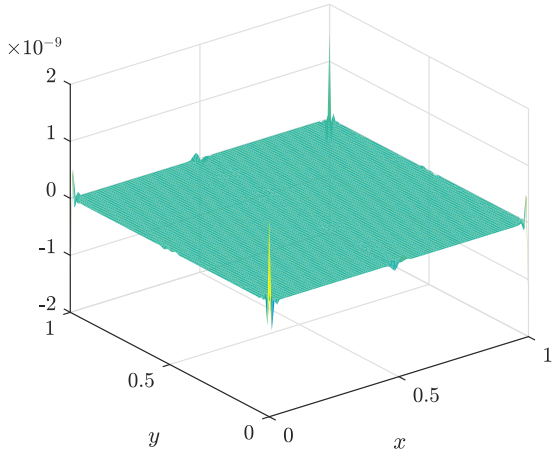
(a) Initial residual $\boldsymbol{r}_0$ and interpolation index $\boldsymbol{\rho}_0 = (0.54, 0.19)$



(b) Residual after 1 mode $\boldsymbol{r}_1$ and interpolation index $\boldsymbol{\rho}_1 = (0.78, 0.04)$

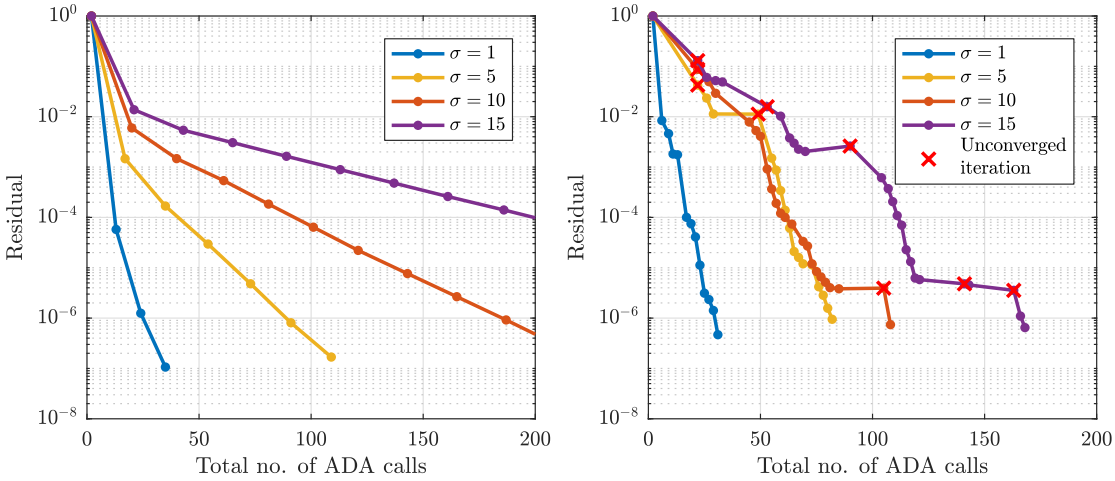


(c) Residual after 2 modes $\boldsymbol{r}_2$ and interpolation index $\boldsymbol{\rho}_2 = (0.11, 0.69)$



(d) Last residual, after 14 terms: $\boldsymbol{r}_{14}$

Figure 4: Reconstructed 2-D view of the residual of the non-linear term cross approximation. Crosses and interpolation points depicted in red.



(a) Outer linearization

(b) Inner linearization

Figure 5: Residual reduction of both outer and inner linearization strategies, varying the magnitude of the non-linear term with the coefficient σ. In Fig. 5a, each dot stands for a non-linear iteration, while in Fig. 5b each dot stands for a mode.

within PGD. This quantity is defined, in each case, as follows:

$$C_{\texttt{ADA}}^{\texttt{outer}} = \sum_{\ell=1}^{\ell^\star} M_\ell \quad \text{and} \quad C_{\texttt{ADA}}^{\texttt{inner}} = \sum_{M=1}^{M^\star} \ell_M^\star, \tag{31}$$

where $\ell^\star$ is the number of non-linear iterations until convergence and M_ℓ is the number of modes computed at each iteration. On the other hand, $M^\star$ is the number of modes at convergence and $\ell_M^\star$ is the number of non-linear iterations for the convergence of each mode.

Fig. 5 depicts the total number of ADA calls against the relative residual reduction. Four different values of the non-linear term's magnitude are used. As expected, the greater is the magnitude of the non-linear term, the more ADA calls are required to attain the same error level. However, convergence plots for outer and inner strategies are quite different, see Fig. 5a and Fig. 5b, respectively. While the outer strategy behaves regularly and converges at a constant rate, the inner strategy is less predictable. In some cases, the computed correction does not converge (it stagnates but no divergence is observed), as indicated in red in Fig. 5b. A possible explanation may be that two kinds of non-linearities are addressed simultaneously inside the inner loop, see Algorithm 2: the computation of the multiplicative factors of the correction and the non-linear term itself. Despite its lack of robustness, the global convergence of the inner strategy is faster, as less ADA calls are required to attain the predefined error level.

In order to evaluate the cost of cross approximation for both outer and inner approaches, the total number of calls to the fiber search algorithm (FSA) is evaluated. This quantity

is defined, in each case, as follows:

$$C_{\texttt{FSA}}^{\texttt{outer}} = \sum_{\ell=1}^{\ell^\star} S_\ell \quad \text{and} \quad C_{\texttt{FSA}}^{\texttt{inner}} = \sum_{M=1}^{M^\star} \sum_{\ell_M=1}^{\ell_M^\star} S_{\ell_M}, \tag{32}$$

where S_ℓ is the number of modes of the non-linear term at each non-linear iteration ℓ. On the other hand, S_{ℓ_M} is the number of modes of the non-linear term at iteration ℓ of mode M. Therefore, the inner scheme is likely to require a higher cost in terms of FSA calls. Fig. 6 compares the number of ADA versus FSA calls for both inner and outer schemes, and several values of the non-linear term's magnitude.

7 Parametrized fully-developed flow

To further assess the performance of the cross approximations algorithm, in this section we consider a problem of practical interest. The problem at hand consist in the fully developed inertialess flow of two stratified non-Newtonian fluids in a channel with square cross section. The flow field is described in terms of the streamwise velocity component $w(x, y)$. The governing equation is formulated as follows:

$$\boldsymbol{\nabla} \cdot (\eta \boldsymbol{\nabla} w) = -\frac{\partial p}{\partial z}, \tag{33}$$

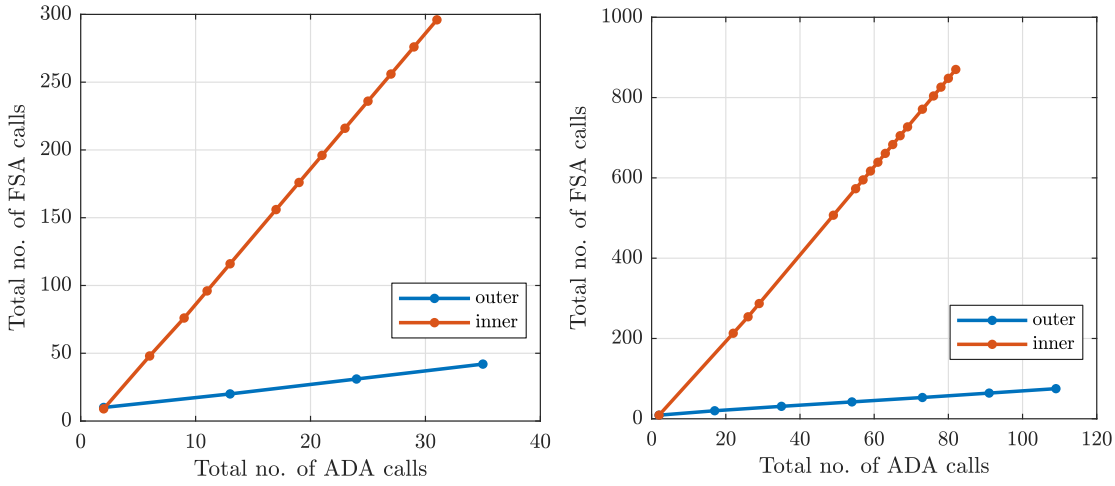
in which $\frac{\partial p}{\partial z}$ is the pressure gradient in the streamwise direction, which is arbitrarily set to the unit value, while $\boldsymbol{\nabla} := \boldsymbol{i}\partial/\partial x + \boldsymbol{j}\partial/\partial y$ is the gradient operator in the cross section plane. The non-linear viscosity is modeled according to the Carreau-Yasuda law:

$$\eta(\dot{\gamma}) = \eta_\infty + (\eta_0 - \eta_\infty)(1 + (\tau\dot{\gamma})^a)^{\frac{n-1}{a}}. \tag{34}$$

In the latter expression the viscosity is a function of the shear rate:

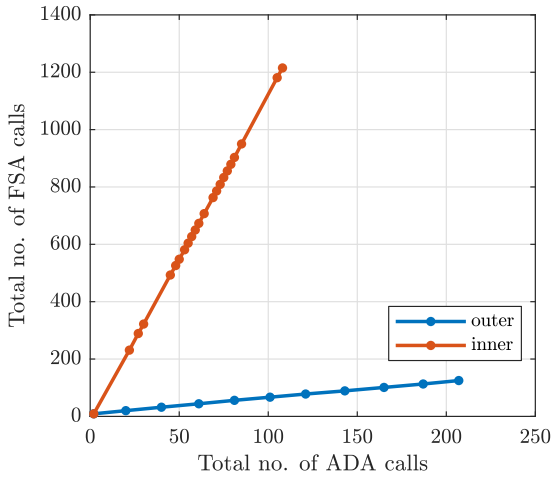
$$\dot{\gamma} = \sqrt{\left(\frac{\partial w}{\partial x}\right)^2 + \left(\frac{\partial w}{\partial y}\right)^2}. \tag{35}$$

The material parameters η_∞, η_0, τ, denote respectively the infinite and zero shear viscosities and the characteristic relaxation time of the fluid, while the phenomenological constants n and a are adjusted to reproduce shear thinning behavior. In what follows, we shall assume $n = 0.2$ and $a = 2$. In practical applications stratified flow of shear thinning fluids can be found in simulation of multi-component coextrusion flows of molten polymers [53, 54]. In this section we assume a bicomponent three-layer flow with uniformly distributed thickness of the fluid layers as depicted in Fig. 7. In practice, for fixed layer thicknesses and pressure gradient, the difference in viscosity between the two fluids determines how the flow rate is distributed in the cross section. Indeed, the layer with smallest viscosity is characterized

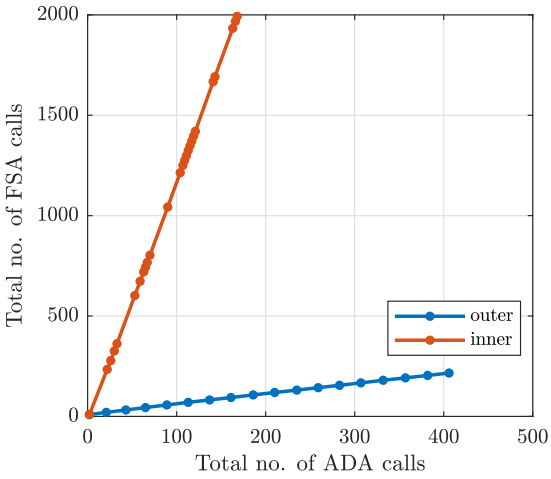


(a) $\sigma = 1$

(b) $\sigma = 5$



(c) $\sigma = 10$



(d) $\sigma = 15$

Figure 6: Assessment of the algorithmic performances of both outer and inner strategies: ADA versus FSA calls, varying the magnitude of the non-linear term with the coefficient σ.

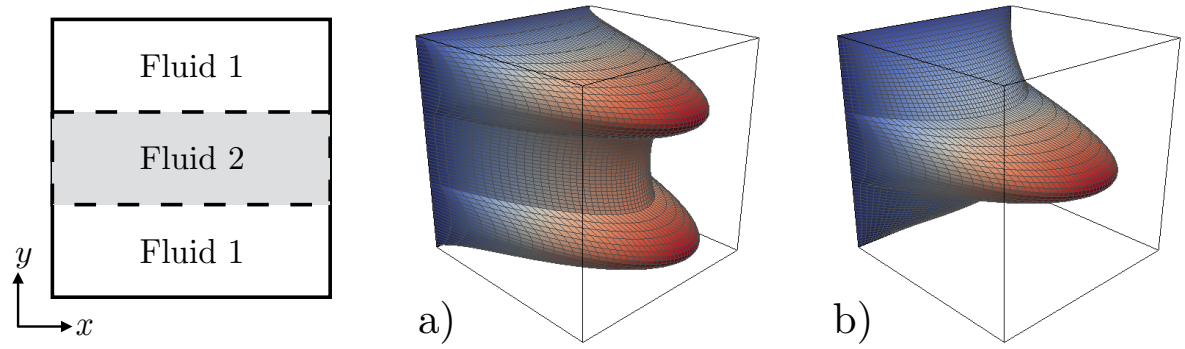


Figure 7: Schematics of bicomponent three-layer flow of non-Newtonian fluids in a square cross section channel. Velocity profiles for the high viscosity core configuration (a) and low viscosity core configuration (b).

by the highest flow rate. The differences in the velocity profile between low viscosity and high viscosity core configuration are also shown in Fig. 7.

For the numerical solution, the equation of flow is considered in its dimensionless form. Therefore the following parameters are defined:

$$\begin{aligned}
\mu_1 &= \frac{\eta_{02}}{\eta_{01}} \in [0.1, 10] \quad \text{Zero shear viscosity ratio} \\
\mu_2 &= \frac{\eta_{\infty 1}}{\eta_{01}} \in [0.1, 1] \quad \text{Infinite to zero shear viscosity ratio (fluid 1)} \\
\mu_3 &= \frac{\eta_{\infty 2}}{\eta_{02}} \in [0.1, 1] \quad \text{Infinite to zero shear viscosity ratio (fluid 2)} \\
\mu_4 &= \frac{\tau_2}{\tau_1} \in [0.1, 10] \quad \text{Characteristic relaxation time ratio}
\end{aligned}$$

When written in dimensionless form for two fluids, the expression for the viscosity given in Eq. (34) reads as:

$$\begin{aligned}
\text{Fluid 1:} \quad \eta_1(\dot{\gamma}) &= \frac{2}{1+\mu_1}\left(\mu_2 + (1-\mu_2)(1+\dot{\gamma}^a)^{\frac{n-1}{a}}\right), \\
\text{Fluid 2:} \quad \eta_2(\dot{\gamma}) &= \frac{2\mu_1}{1+\mu_1}\left(\mu_3 + (1-\mu_3)(1+(\mu_4\dot{\gamma})^a)^{\frac{n-1}{a}}\right).
\end{aligned} \tag{36}$$

Both space and parametric dimensions are discretized using standard first order Finite Element approximation. Each dimension is discretized with 64 equally spaced nodes.

The problem is then linearized using Picard iterations. At iteration ℓ, the viscosity is estimated using the cross approximation scheme based on the velocity field at iteration $\ell - 1$. The iteration loop is started using the Newtonian fluid flow as initial solution guess

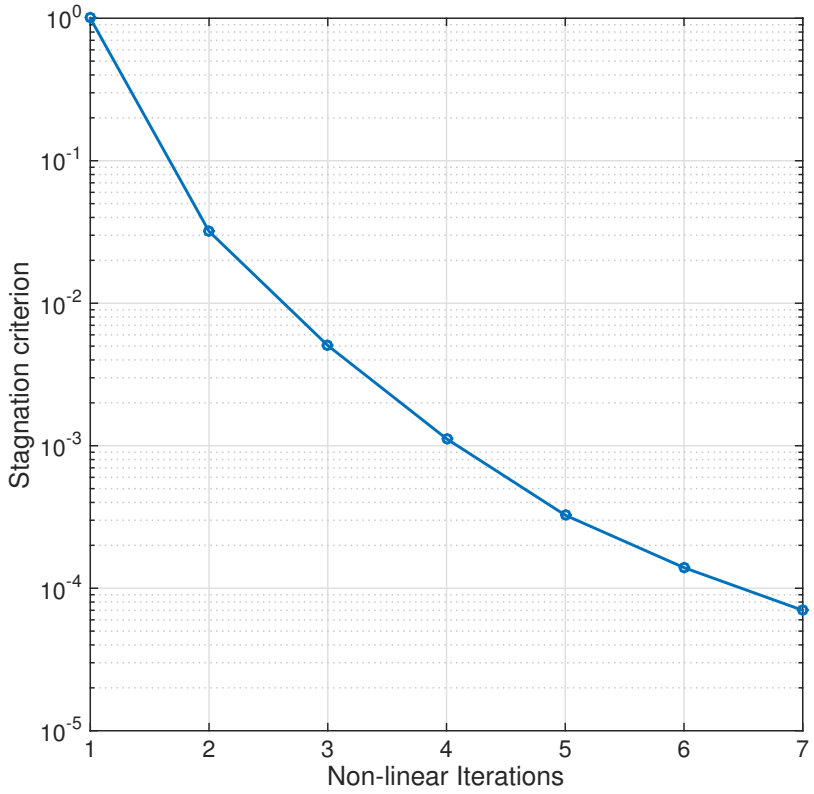


Figure 8: Convergence of the non-linear problem measured in terms of the relative difference between two consecutive iterations (stagnation criterion).

and is stopped when the relative variation in the solution between two successive iterations is less than 10^{-4}. The convergence diagram of this iterative procedure is presented in Fig. 8.

At each non-linear iteration, PGD sub-iterations are used to construct a separated variables format of the solution (outer linearization scheme, see Section 5.1). In particular the following tensor format is enforced:

$$w(x, y, \mu_1, \mu_2, \mu_3, \mu_4) = \sum_{m=1}^{M} W_0^m(x, y)\, W_1^m(\mu_1, \mu_2)\, W_2^m(\mu_3, \mu_4). \tag{37}$$

The complexity of the original six-dimensional problem is broken into a succession of three two-dimensional problems. The final solution consists of $M = 28$ terms. In terms of memory, the reduced solution requires 2.63 MByte, whereas the full explicit storage of its equivalent 6D solution would take 512 GByte in double-precision format (recall there are 64 nodes per dimension). This represents a reduction factor in the order of 10^5, i.e. five orders of magnitude. Average CPU time over 5 runs was 724.71 sec (around 12

minutes) on a 64 bit machine with the following specifications: 2.9 GHz Intel Core i5 with 16 GBytes 1867MHz DDR3, running under MacOS X 10.12.6. Total RAM consumption during execution never went over 410 MBytes, which could eventually further reduced in an optimized implementation. CPU time of a brute-force approach was not measured for practical reasons, but it can be estimated instead. Assuming an optimistic runtime per simulation of 1 msec, it would take about 5 hours to complete in a single-thread implementation. Runtime for a compression step to reduce the 512 GByte is excluded from this estimation.

Finally, the PGD solution is compared to the FEM at randomly drawn points in the parametric space. Results shown in Fig. 9 refer to the velocity profile in the vertical centerline of the cross section and are in good agreement with the direct FEM reference solutions. Indeed, the computed relative error in the maximum norm is in all cases less than 10^{-3}.

8 Conclusions

In this paper, we have shown the importance of tensorization for the efficiency of tensor-based algorithms. The main ideas have been presented in the Proper Generalized Decomposition framework, but they could also be applied to other tensor representations. We have discussed the adequacy of interpolation rather than projection-based approaches as a means to approximate non-linear equations as well as to enforce tensorization. The use of cross approximations reveals simple and efficient for an interpolation-based approximation of models in moderate dimension. Finally, the performance of several schemes for the linearization of tensor problems has been assessed.

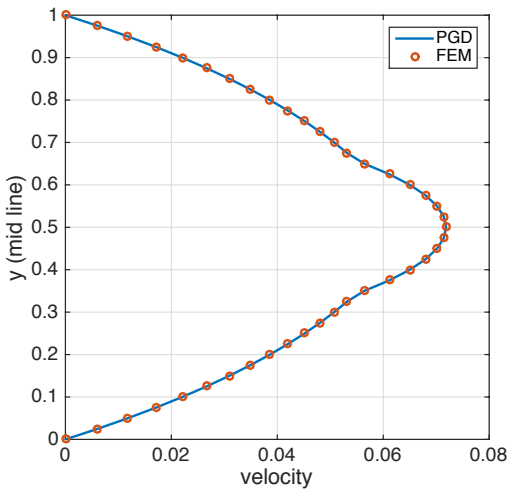
A Coefficients definition for alternating directions optimization

Coefficients $\beta_*^r \in \mathbb{R}$, $\tilde{\boldsymbol{\alpha}}_*^r \in \mathbb{R}^M$ and $\tilde{\boldsymbol{\gamma}}_* \in \mathbb{R}^S$ in Eq. (10) are defined as follows:

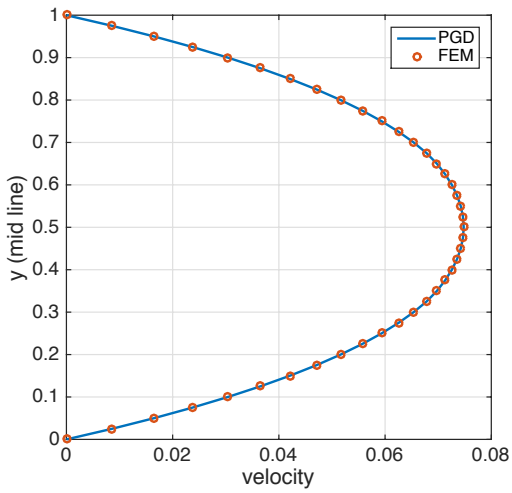
$$\beta_*^r := \prod_{d=0, d\neq *}^{D} \langle \mathbf{A}_*^r \boldsymbol{v}_*, \boldsymbol{v}_* \rangle_* \,,$$

$$\tilde{\boldsymbol{\alpha}}_*^r := \boldsymbol{\alpha} \circ \left(\bigodot_{d=0, d\neq *}^{D} \langle \mathbf{A}_*^r \mathbf{W}_*, \boldsymbol{v}_* \rangle_* \right)^T \quad \text{and}$$

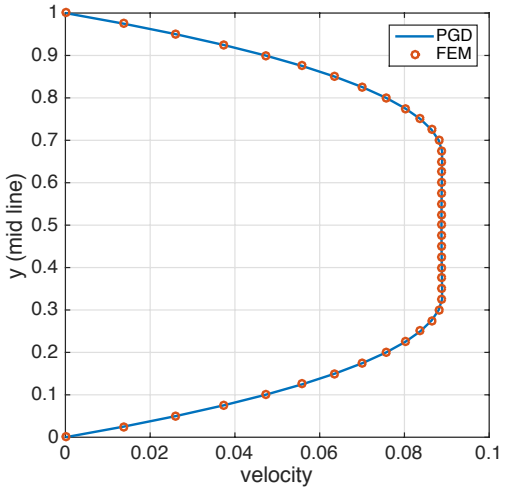
$$\tilde{\boldsymbol{\gamma}}_* := \boldsymbol{\gamma} \circ \left(\bigodot_{d=0, d\neq *}^{D} \langle \mathbf{V}_*, \boldsymbol{v}_* \rangle_* \right)^T ,$$



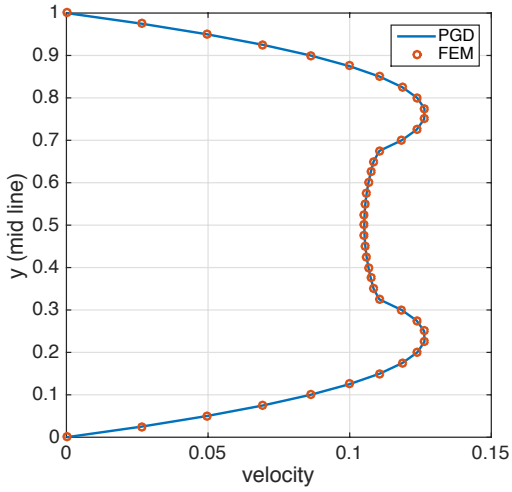
a) $\boldsymbol{\mu}_1 = (0.35, 0.93, 0.98, 0.66)$



b) $\boldsymbol{\mu}_2 = (1.09, 0.35, 0.84, 0.60)$



c) $\boldsymbol{\mu}_3 = (3.07, 0.75, 0.33, 0.51)$



d) $\boldsymbol{\mu}_4 = (8.76, 0.80, 0.57, 0.15)$

Figure 9: Comparison of the PGD solution against the direct FEM solution at the vertical centerline of the cross section, for four randomly drawn parameter combinations.

where "$\bullet^T$" denotes the transpose and "$\circ$" stands for the Hadamard (component-wise) product. Recall that $\boldsymbol{\alpha}$ contains the representation coefficients of the rank-M approximation of the solution, while $\boldsymbol{\gamma}$ are the representation coefficients of the rank-S representation of the non-linear term. See Eq. (6) and Eq. (8), respectively.

Using the coefficients defined above, we can define the following quantities:

$$\tilde{\mathbf{A}}_* := \sum_{r=1}^{R} \beta_*^r \mathbf{A}_*^r \quad \text{and} \quad \tilde{\boldsymbol{b}}_* := -\sum_{r=1}^{R} \mathbf{A}_*^r \mathbf{W}_* \tilde{\boldsymbol{\alpha}}_*^r - \mathbf{V}_* \tilde{\boldsymbol{\gamma}}_*,$$

that can be introduced in Eq. (10), leading to the following minimization problem:

$$\boldsymbol{w}_* = \arg \min_{\boldsymbol{v}_* \in \mathbb{R}^{N_*}} \frac{1}{2} \langle \tilde{\mathbf{A}}_* \boldsymbol{v}_*, \boldsymbol{v}_* \rangle_* + \langle \tilde{\boldsymbol{b}}_*, \boldsymbol{v}_* \rangle_*, \tag{38}$$

whose solution is $\boldsymbol{w}_* = \tilde{\mathbf{A}}_*^{-1} \tilde{\boldsymbol{b}}_*$.

B Fiber search algorithm in detail

Let $\mathbf{P} := \bigodot_{d=0}^{D} \mathbf{P}_d \in \mathbb{R}^{N \times 1}$ be an extractor matrix, with $\mathbf{P}_* := \mathbf{I} \in \mathbb{R}^{N_* \times N_*}$. Additionally, $\mathbf{P}_d \in \mathbb{R}^{N_d \times 1}$, for $d \neq *$, are extractor matrices full of zeros, except the entry ρ_d (see Section 4.1), which is equal to one.

The restriction of the approximation residual onto direction "$*$" can be written as follows:

$$\boldsymbol{r}_* := \mathbf{P}^T (\boldsymbol{f}(\boldsymbol{u}_M^\ell) - \boldsymbol{f}_S) \equiv \boldsymbol{f}(\mathbf{P}^T \boldsymbol{u}_M^\ell) - \mathbf{P}^T \boldsymbol{f}_S. \tag{39}$$

Recalling the definition of both $\boldsymbol{u}_M^\ell$ and $\boldsymbol{f}_S$, we arrive to:

$$\boldsymbol{r}_* = \boldsymbol{f}(\mathbf{W}_* \, \tilde{\boldsymbol{\alpha}}_*) - \mathbf{V}_* \, \tilde{\boldsymbol{\gamma}}_*, \tag{40}$$

where the coefficients $\tilde{\boldsymbol{\alpha}}_* \in \mathbb{R}^M$ and $\tilde{\boldsymbol{\gamma}}_* \in \mathbb{R}^S$ are defined as follows:

$$\tilde{\boldsymbol{\alpha}}_* := \boldsymbol{\alpha} \circ \left(\bigodot_{d=0, d \neq *}^{D} \mathbf{P}_d^T \mathbf{W}_d \right)^T, \quad \text{and}$$

$$\tilde{\boldsymbol{\gamma}}_* := \boldsymbol{\gamma} \circ \left(\bigodot_{d=0, d \neq *}^{D} \mathbf{P}_d^T \mathbf{V}_d \right)^T.$$

Here "$\circ$" stands for the Hadamard (component-wise) product.

References

[1] Ch. Ghnatios, F. Masson, A. Huerta, A. Leygue, E. Cueto, and F. Chinesta. Proper Generalized Decomposition based dynamic data-driven control of thermal processes. *Comput. Meth. Appl. Mech. Engrg.*, 213-216:29–41, 2012.

[2] D. Borzacchiello, J.V. Aguado, and F. Chinesta. Reduced Order Modelling for efficient optimisation of a hot-wall Chemical Vapour Deposition reactor. *Int. J. Numer. Method H.*, 27(4), 2017, DOI: 10.1108/HFF-04-2016-0153.

[3] J.V. Aguado, D. Borzacchiello, Ch. Ghnatios, F. Lebel, R. Upadhyay, C. Binetruy, and F. Chinesta. A Simulation App based on reduced order modeling for manufacturing optimization of composite outlet guide vanes. *Adv. Model. Simul. Eng. Sci.*, 4(1):1–26, 2017.

[4] C. Prud'homme, D.V. Rovas, K. Veroy, L. Machiels, Y. Maday, A.T. Patera, and G. Turinici. Reliable real-time solution of parametrized partial differential equations: Reduced-Basis Output Bound Methods. *J. Fluids Eng.*, 124(1):70–80, 2001.

[5] K. Willcox and J. Peraire. Balanced model reduction via the Proper Orthogonal Decomposition. *AIAA Journal*, 40(11):2323–2330, 2002.

[6] G. Beylkin and M. Mohlenkamp. Algorithms for numerical analysis in high dimensions. *SIAM J. Sci. Comput.*, 26(6):2133–2159, 2005.

[7] G. Rozza, D.B.P. Huynh, and A.T. Patera. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations – application to transport and continuum mechanics. *Arch. Comput. Methods Eng.*, 15(3):229–275, 2008.

[8] A. Quarteroni, A. Manzoni, and F. Negri. *Reduced Basis Methods for Partial Differential Equations: An Introduction.* Modeling and Simulation in Science, Engineering and Technology. Springer, Basel, 1st edition, 2015.

[9] A. Ammar, B. Mokdad, F. Chinesta, and R. Keunings. A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids. Part II: transient simulation using space-time separated representations. *J. Non-Newtonian Fluid Mech.*, 144(2-3):98–121, 2007.

[10] A. Nouy. A priori model reduction through Proper Generalized Decomposition for solving time-dependent partial differential equations. *Comput. Meth. Appl. Mech. Engrg.*, 199:1603–1626, 2010.

[11] L. Grasedyck, D. Kressner, and C. Tobler. A literature survey of low-rank tensor approximation techniques. *arXiv:1302.7121*, 2013.

[12] M. Barrault, Y. Maday, N.C. Nguyen, and A.T. Patera. An "empirical interpolation method": application to efficient reduced-basis discretization of partial differential equations. *C.R. Acad. Sci. I-Math.*, 339(9):667–672, 2004.

[13] Y. Maday, N.C. Nguyen, A.T. Patera, and S.H. Pau. A general multipurpose interpolation procedure: the Magic Points. *CPAA*, 8(1):383–404, 2009.

[14] S. Chaturantabut and D.C. Sorensen. Nonlinear model order reduction via Discrete Empirical Interpolation. *SIAM J. Sci. Comput.*, 32(5):2737–2764, 2010.

[15] P. Tiso and D.J. Rixen. *Discrete Empirical Interpolation Method for Finite Element Structural Dynamics*, pages 203–212. Springer New York, New York, NY, 2013.

[16] B. Peherstorfer, D. Butnaru, K. Willcox, and H.J. Bungartz. Localized discrete empirical interpolation method. *SIAM J. Sci. Comput.*, 36(1):168–192, 2014.

[17] T. Chapman, P. Avery, P. Collins, and C. Farhat. Accelerated mesh sampling for the hyper reduction of nonlinear computational models. *Int. J. Numer. Meth. Engng.*, 109(12):1623–1654, 2017.

[18] D. Ryckelynck. A priori hypereduction method : an adaptive approach. *J. Comput. Phys.*, 202(1):346–366, 2005.

[19] C. Farhat, P. Avery, T. Chapman, and J. Cortial. Dimensional reduction of nonlinear finite element dynamic models with finite rotations and energy-based mesh sampling and weighting for computational efficiency. *Int. J. Numer. Meth. Engng.*, 98(9):625–662, 2014.

[20] C. Farhat, T. Chapman, and P. Avery. Structure-preserving, stability, and accuracy properties of the energy-conserving sampling and weighting method for the hyper reduction of nonlinear finite element dynamic models. *Int. J. Numer. Meth. Engng.*, 102:1077–1110, 2015.

[21] J. Hernández, M.A. Caicedo, and A. Ferrer. Dimensional hyper-reduction of nonlinear finite element models via empirical cubature. *Comput. Meth. Appl. Mech. Engrg.*, 313:687–722, 2017.

[22] S. Niroomandi, I. Alfaro, E. Cueto, and F. Chinesta. Model order reduction for hyperelastic materials. *Int. J. Numer. Meth. Engng.*, 81(9):1180–1206, 2010.

[23] A. Leygue, F. Chinesta, M. Beringhier, T.L. Nguyen, J.C. Grandidier, F. Pesavento, and B. Schrefler. Towards a framework for non-linear thermal models in shell domains. *Int. J. Numer. Method H.*, 23(1):55–73, 2013.

[24] S. Niroomandi, D. González, I. Alfaro, F. Bordeu, A. Leygue, E. Cueto, and F. Chinesta. Realtime simulation of biological soft tissues: a PGD approach. *Int. J. Numer. Methods Biomed. Eng.*, 29(5):586–600, 2013.

[25] P. Ladevèze, J.-C. Passieux, and D. Neron. The LATIN multiscale computational method and the Proper Generalized Decomposition. *Comput. Meth. Appl. Mech. Engrg.*, 199(21-22):1287–1296, 2010.

[26] D. Néron and P. Ladevèze. Proper generalized decomposition for multiscale and multiphysics problems. *Arch. Comput. Methods Eng.*, 17(4):351–372, 2010.

[27] M. Capaldo, P.-A. Guidault, D. Néron, and P. Ladevèze. The Reference Point Method, a hyperreduction technique: Application to PGD-based nonlinear model reduction. *Comput. Meth. Appl. Mech. Engrg.*, 322:483–514, 2017.

[28] E. Tyrtyshnikov. Incomplete Cross Approximation in the Mosaic-Skeleton Method. *Computing*, 64(4):367–380, 2000.

[29] M. Espig, L. Grasedyck, and W. Hackbusch. Black box low tensor-rank approximation using fiber-crosses. *Constr. Approx.*, 30:557–597, 2009.

[30] A. Falcó and A. Nouy. A Proper Generalized Decomposition for the solution of elliptic problems in abstract form by using a functional Eckart-Young approach. *J. Math. Anal. Appl.*, 376:469–480, 2011.

[31] T.G. Kolda and B.W. Bader. Tensor decompositions and applications. *SIAM Rev.*, 51(3):455–500, 2009.

[32] A. Ammar, F. Chinesta, P. Díez, and A. Huerta. An error estimator for separated representations of highly multidimensional models. *Comput. Meth. Appl. Mech. Engrg.*, 199(25-28):1872–1880, 2010.

[33] P. Ladevèze and L. Chamoin. On the verification of model reduction methods based on the proper generalized decomposition. *Comput. Meth. Appl. Mech. Engrg.*, 200(23):2032–2047, 2011.

[34] L. Boucinha, A. Ammar, A. Gravouil, and A. Nouy. Ideal minimal residual-based Proper Generalized Decomposition for non-symmetric multi-field models. Application to transient elastodynamics in space-time domain. *Comput. Meth. Appl. Mech. Engrg.*, 273:56–76, 2014.

[35] A. Barbarulo, P. Ladevèze, H. Riou, and L. Kovalevsky. Proper generalized decomposition applied to linear acoustic: a new tool for broad band calculation. *J. Sound Vibr.*, 333(11):2422–2431, 2014.

[36] M. Billaud-Friess, A. Nouy, and O. Zahm. A tensor approximation method based on ideal minimal residual formulations for the solution of high-dimensional problems. *ESAIM Math. Model. Num.*, 48(6):1777–1806, 2014.

[37] F. Chinesta, P. Ladevèze, and E. Cueto. A short review on model order reduction based on Proper Generalized Decomposition. *Arch. Comput. Methods Eng.*, 18(4):395–404, 2011.

[38] A. Uschmajew. Local convergence of the alternating least squares algorithm for canonical tensor approximation. *SIAM J. Matrix Anal. Appl.*, 33(2):639–652, 2012.

[39] A. Ammar, A. Zghal, F. Morel, and F. Chinesta. On the space-time separated representation of integral linear viscoelastic models. *C.R. Mécanique*, 343(4):247–263, 2015.

[40] L. De Lathauwer, B. De Moor, and J. Vanderwalle. A multilinear Singular Value Decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278, 2000.

[41] D. Modesto, S. Zlotnik, and A. Huerta. Proper Generalized Decomposition for parameterized Helmholtz problems in heterogeneous and unbounded domains: application to harbor agitation. *Comput. Methods Appl. Mech. Eng.*, 2015.

[42] J. Hastad. Tensor rank is NP-complete. *J. Algorithms*, 11:644–654, 1990.

[43] Wolfang Hackbusch. *Tensor spaces and numerical tensor calculus*. Springer Series in Computational Mathematics. Springer-Verlag, Berlin-Heidelberg, 1st edition, 2012.

[44] L.R. Tucker. The extension of factor analysis to three-dimensional matrices. In Rinehardt Holt and New York Winston, editors, *Contributions to Mathematical Psychology*, pages 110–127, 1964.

[45] I.V. Oseledets. Tensor-train decomposition. *SIAM J. Sci. Comput.*, 33(5):2295–2317, 2011.

[46] W. Hackbusch and S. Kühn. A new scheme for the tensor representation. *J. Fourier Anal. Appl.*, 15(5):706–722, 2009.

[47] I.V. Oseledets, D.V. Savostianov, and E.E. Tyrtyshnikov. Tucker dimensionality reduction of three-dimensional arrays in linear time. *SIAM J. Matrix Anal. Appl.*, 30(3):939–956, 2008.

[48] I.V. Oseledets and E.E. Tyrtyshnikov. TT-cross approximation for multidimensional arrays. *Linear Algebra Appl.*, 432(1):70–88, 2010.

[49] M. Bebendorf. Adaptive cross approximation of multivariate functions. *Constr. Approx.*, 34:149–179, 2011.

[50] J. Ballani, L. Grasedyck, and M. Kluge. Black box approximation of tensors in hierarchical tucker format. *Linear Algebra Appl.*, 438(2):639–657, 2013.

[51] I.V. Oseledets and S.V. Dolgov. Solution of linear systems and matrix inversion in the TT-Format. *SIAM J. Sci. Comput.*, 34(5):A2718–A2739, 2012.

[52] S.V. Dolgov and D.V. Savostyanov. Alternating minimal energy methods for linear systems in higher dimensions. *SIAM J. Sci. Comput.*, 36(5):A2248–A2271, 2013.

[53] D. Borzacchiello, E. Leriche, B. Blottière, and J. Guillet. On the mechanism of viscoelastic encapsulation of fluid layers in polymer coextrusion. *J. Rheol.*, 58(2):493–512, 2014.

[54] D. Borzacchiello, E. Leriche, B. Blottière, and J. Guillet. Three-dimensional finite volume computation of viscoelastic fluid encapsulation by phase-field modeling. *J. Non-Newtonian Fluid Mech.*, 200:52–64, 2013.