Tensor-train approximation of parametric constitutive equations in elasto-viscoplasticity

Clément Olivier, David Ryckelynck, Julien Cortial

Tensor-train approximation of parametric constitutive equations in elasto-viscoplasticity

Clément Olivier[a,b], David Ryckelynck[b], Julien Cortial[a,*]

[a]*Safran Tech (Safran Group), Rue des Jeunes Bois, Châteaufort, CS 80112, 78772 Magny-Les-Hameaux, France*
[b]*MINES ParisTech, PSL - Research University, Centre des matériaux, CNRS UMR 7633, 10 rue Desbruères, 91003 Evry, France*

Abstract

This work presents a novel approach to construct surrogate models of parametric Differential Algebraic Equations based on a tensor representation of the solutions. The procedure consists in building simultaneously, for every output of the reference model, an approximation given in tensor-train format. A parsimonious exploration of the parameter space coupled with a compact data representation allows to alleviate the curse of dimensionality. The approach is thus appropriate when many parameters with large domains of variation are involved. The numerical results obtained for a nonlinear elasto-viscoplastic constitutive law show that the constructed surrogate model is sufficiently accurate to enable parametric studies such as the calibration of material coefficients.

Keywords: parameter-dependent model, surrogate modeling, tensor-train decomposition, Gappy POD, heterogeneous data, elasto-viscoplasticity

1. Introduction

Predictive numerical simulations in solid mechanics require complex material laws that involve systems of highly nonlinear Differential Algebraic Equations (DAEs). These models are essential in challenging industrial applications, for instance to study the effects of the extreme thermo-mechanical loadings that turbine blades may sustain in helicopter engines [1, 2]. These DAE systems are referred to as constitutive laws in the material science community. They express, for a specific material, the relationship between the mechanical quantities such as the strain, the stress and miscellaneous internal variables, and stand as the closure relations of the physical equations of mechanics. Complex constitutive equations are often tuned through a set of parameters called material coefficients.

An appropriate calibration of these coefficients is necessary to ensure that the numerical model mimics the actual physical behavior. Numerical parametric studies, consisting in analyzing the influence of the parameter values on the solutions, are typically used to perform the identification. However, when the number of parameters increases and unless the computational effort required for a single numerical simulation is negligible, the exploration of the parameter domain turns into a tedious task and exhaustive analyses become unfeasible. Moreover, defining an unambiguous criterion measuring the fidelity of the model to experimental data is a challenge for models with complex behaviors.

A common technique is to rely on a surrogate model (or metamodel) that maps the input parameters to the outputs of interest of the physical model. Once built, the surrogate model can be exploited very efficiently during a so-called *online phase* since its evaluation is very cheap compared to the original model. The real-time response provided by the metamodel for any parameter value helps carrying out parametric studies. In material science, the robustness of the calibration process could be dramatically improved if this type of methodology was followed.

The idea of representing the set of all possible parameter-dependent solutions of ODEs and PDEs as a multiway tensor was introduced with the Proper Generalized Decomposition (PGD) [3–5]. In this representation, each dimension corresponds to a spatial/temporal coordinate or a parameter coefficient. The resulting tensor is never assembled

*Corresponding author
Email addresses: clement.olivier@safrangroup.com (Clément Olivier), david.ryckelynck@mines-paristech.fr (David Ryckelynck), julien.cortial@safrangroup.com (Julien Cortial)

explicitly but instead remains an abstract object for which a low-rank approximation based on a Canonical Polyadic decomposition [6] is computed. The PGD method further alleviates the curse of dimensionality by introducing a multidimensional weak formulation over the entire parameter space, and the solutions are sought in a particular form where all variables are separated. When differential operators admit a tensor decomposition, the PGD method is very efficient because the multiple integrals involved in the multidimensional weak form of the equations are simplified as a sum of products of simple integrals.

Unfortunately realistic constitutive equations as well as less sophisticated elasto-viscoplastic models admit no tensor decomposition with respect to the material coefficients and the time variables. An extension of the PGD to highly nonlinear laws is therefore non-trivial. However, many other tensor decomposition approaches have been successfully proposed to approximate functions or solutions of differential equations defined over high dimensional spaces. We refer the reader to [7–9] for detailed reviews on tensor decomposition techniques and their applications. Among the existing formats – CP decomposition [6, 10, 11], Tucker decomposition [8, 12], Hierarchical Tucker decomposition [8, 13] – this work investigates the tensor-train (TT) decomposition [14, 15]. The TT-cross algorithm, introduced in [14] and further developed in [16, 17], is a practical procedure to build an approximation of a given tensor under the tensor-train format.

In the present work, an alternative procedure inspired by the TT-cross algorithm is developed. As its original counterpart, it does not break down when only a parsimonious exploration of the physics-based tensors is affordable, in particular when their entries are solutions of a DAE system involving a high-dimensional parameter space. All outputs of interest of the physical model are grouped into a set of physics-based tensor representations according to their physical units or specific end user requirements. Then the proposed non-intrusive method enables to build simultaneously TT approximations for all these objects. The original aspects of this work are:

- The introduction of an abstract TT approximation algorithm framework that encompasses TT construction procedures such as the TT-SVD and the left-to-right sweep of the TT-cross and enables the use of alternative low-rank approximations instead of the Singular Value Decomposition (SVD) or the Pseudo-Skeleton Decomposition (PSD);

- The development of a variant of the abstract TT algorithm based on a Gappy POD formulation that accommodates the heterogeneous, real-valued outputs that are ubiquitous in mechanics of materials. The resulting procedure – termed *Multiple TT algorithm* in the sequel – constructs an arbitrary number of tensor representations based on a shared sampling set of the parameter domain;

- The application of the proposed tensor-decomposition-based methodology to the numerical solution of a highly nonlinear elasto-viscoplastic constitutive law.

The text is organized as follows. Section 2 introduces the notations required for the subsequent developments. Section 3 presents the tensor-train format and a multilinear interpolation method is detailed in Section 4. Section 5 details an abstract TT algorithm that generates an approximate tensor-train decomposition of a reference tensor. The key step of the algorithm, namely the low-rank matrix approximation, is discussed in Section 6. Section 7 introduces a general formulation for a class of DAE systems covered by the proposed approach. The Multiple TT algorithm detailed in Section 8 generalizes the previously introduced TT algorithm for heterogeneous, multiple outputs. Section 9 describes the nonlinear elasto-viscoplastic law used in the numerical application and presents the associated results. Finally, some perspectives are discussed in Section 10.

2. Notations

A tensor of order d $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ (denoted with bold calligraphic letter) refers to a multidimensional array (also called multiway array).

The entry of $\boldsymbol{\mathcal{A}}$ identified by the indices $(i_1, \ldots i_d) \in I_1 \times \cdots \times I_d$ is denoted by:

$$\boldsymbol{\mathcal{A}}(i_1, \ldots, i_d) \in \mathbb{R}$$

where $I_k = [1 : n_k]$ is the set of natural numbers from 1 to n_k (inclusive) for $k = 1, \ldots, d$.

The *bracket notation* allows to define a tensor as the collection of its elements:

$$\mathcal{A} = [\mathcal{A}(i_1, \ldots, i_d),\ \forall\ (i_1, \ldots, i_d) \in I_1 \times \cdots \times I_d]$$

When the respective ranges of definition of the indices are obvious, the notation can be shortened as follows:

$$\mathcal{A} = [\mathcal{A}(i_1, \ldots, i_d)]$$

The *shape* of a tensor is the tuple constituted by the size of all dimensions (in the above example $(n_1, \ldots, n_d)$). Given a tensor, the *reshape* operation returns a tensor of different shape with similar elements, referred to as a *reshaped tensor*. More specifically, the process of reshaping consists in re-indexing the reference tensor. From a given tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ the association of consecutive dimensions enables to define reshaped tensors. For instance, the association of the q consecutive dimensions $p, (p+1), \ldots, (p+q-1)$ yields a tensor of order $(d-q+1)$:

$$\mathcal{B} \in \mathbb{R}^{n_1 \times \cdots \times n_{p-1} \times \bar{n}_p \times n_{p+q} \times \cdots \times n_d} \quad \text{where} \quad \bar{n}_p = \prod_{k=p}^{p+q-1} n_k$$

and the relation between the elements of both tensors are given by:

$$\mathcal{A}(i_1, \ldots, i_d) = \mathcal{B}\left(i_1, \ldots, i_{p-1}, \bar{\imath}_p, i_{p+q}, \ldots, i_d\right) \quad \text{where} \quad \bar{\imath}_p = i_p + \sum_{k=p+1}^{p+q-1} (i_k - 1) \prod_{l=p}^{k-1} n_l \tag{1}$$

When using the bracket notation, the association of dimensions is denoted by adding parentheses around the indices thus creating a *multi-index* [18]. The equality (1) can be written as:

$$\mathcal{B} = \left[\mathcal{A}\left(i_1, \ldots, i_{p-1}, \left(i_p, \ldots, i_{(p+q-1)}\right), i_{(p+q)}, \ldots, i_d\right)\right]$$

Two particular types of reshaping will be used in the rest of the article.

First, the *matricization* consists in re-indexing the elements in order to interpret a tensor as a matrix (a 2^{nd}-order tensor). The q^{th} matricization of $\mathcal{A}$ denoted by $\langle \mathcal{A} \rangle_q$ consists in associating the dimensions of $\mathcal{A}$ into two groups, the q leading dimensions and the $(d-q)$ trailing dimensions, such that the newly defined multi-indices enumerate respectively the rows and columns of the matrix $\langle \mathcal{A} \rangle_q$. $\langle \mathcal{A} \rangle_q$ is given with the bracket notation as:

$$\langle \mathcal{A} \rangle_q = \left[\mathcal{A}\left((i_1, \ldots, i_q), (i_{q+1}, \ldots, i_d)\right)\right] \in \mathbb{R}^{(n_1 \ldots n_q) \times (n_{q+1} \ldots n_d)}$$

Second, for a given tensor $\mathcal{A}$, the *squeeze* operation returns a tensor $\mathcal{A}^\star$ for which all dimensions of size 1 have been removed. For instance, $\mathcal{A} \in \mathbb{R}^{10 \times 10 \times 1}$ is transformed into $\mathcal{A}^\star \in \mathbb{R}^{10 \times 10}$ such that:

$$\mathcal{A}^\star(i, j) = \mathcal{A}(i, j, 1) \quad \forall (i, j) \in [1:10]^2$$

The Frobenius norm is denoted by $\|.\|$ without the usual subscript $\mathcal{F}$. For $\mathcal{A} \in \mathbb{R}^{n_1 \times \ldots n_d}$, it reads:

$$\|\mathcal{A}\| = \sqrt{\sum_{i_1, \ldots, i_d \in I_1 \times \cdots \times I_d} \mathcal{A}(i_1, \ldots, i_d)^2}$$

The Frobenius norm of a tensor is invariant under re-indexing. In particular, all matricizations of a given tensor share the same Frobenius norm.

The usual 2-norm of a matrix $A \in \mathbb{R}^{n \times m}$ is denoted by $\|A\|_2$.

Define the tensor dot product $\bullet$ such that for $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_p \times r}$ and $\mathcal{B} \in \mathbb{R}^{r \times m_1 \times \cdots \times m_q}$ the entries of the tensor $\mathcal{C} = \mathcal{A} \bullet \mathcal{B} \in \mathbb{R}^{n_1 \times \cdots \times n_p \times m_1 \times \cdots \times m_q}$ are given by:

$$\mathcal{C}(i_1, \ldots, i_p, j_1, \ldots, j_q) = \sum_{\alpha=1}^{r} \mathcal{A}(i_1, \ldots, i_p, \alpha) \mathcal{B}(\alpha, j_1, \ldots, j_p)$$

Considering the matrix $A \in \mathbb{R}^{n\times m}$, define the *column selection matrix* $Q \in \mathbb{R}^{m\times s}$ associated with the set of column indices $\mathcal{J} = \left\{\mathcal{J}^{(1)}, \ldots, \mathcal{J}^{(s)}\right\} \subset [1:m]$ of A:

$$Q(i,j) = \delta_{i,\mathcal{J}^{(j)}} \quad \forall (i,j) \in [1:m] \times [1:s] \tag{2}$$

where δ denotes the Kronecker delta.

The *colon notation* [19, Sections 1.1.8 and 1.2.9] is used to denote the extraction of a submatrix. The matrix product of A by Q yields a submatrix constituted by the columns $\mathcal{J}$ of A such as:

$$AQ = A(:, \mathcal{J})$$

Similarly, define the *row selection matrix* $P \in \mathbb{R}^{n\times s}$ associated with the set of row indices $\mathcal{I} = \left\{\mathcal{I}^{(1)}, \ldots, \mathcal{I}^{(s)}\right\} \subset [1:n]$ of A:

$$P(i,j) = \delta_{i,\mathcal{I}^{(j)}} \quad \forall (i,j) \in [1:n] \times [1:s] \tag{3}$$

Extracting the rows $\mathcal{I}$ of A consists in applying the transpose of P to the matrix A such that:

$$P^T A = A(\mathcal{I}, :)$$

Remark 1. *All column and row selection matrices have orthogonal columns.*

3. Tensor-train format

The storage complexity for all elements of a tensor $\mathcal{A} \in \mathbb{R}^{n_1\times\cdots\times n_d}$ is $O(n^d)$ where $n = \max(n_1, \ldots, n_d)$. The exponential dependence on the order d prohibits to store simultaneously and explicitly all elements of the tensor even for a "small" d. However, in the case of low-rank tensors, some representations based on tensor decompositions enable their actual storage.

A tensor $\mathcal{T} \in \mathbb{R}^{n_1\times\cdots\times n_d}$ is said to be in tensor-train format (TT format) if its elements are given by the following matrix products:

$$\mathcal{T}(i_1, \ldots, i_d) = G_1(i_1) \ldots G_d(i_d) \in \mathbb{R} \tag{4}$$

where for $k = 1, \ldots, d$, the so-called *tensor carriages* (or *core tensors*) are such that:

$$G_k(i_k) \in \mathbb{R}^{r_{k-1}\times r_k} \quad \forall i_k \in I_k$$

In the original definition of the tensor-train format [14], the leading and trailing factors (corresponding to $G_1(i_1)$ and $G_d(i_d)$ for any choice of i_1 and i_d) are respectively row and column vectors. The convention $r_0 = r_d = 1$ is adopted so that row matrices $G_1(i_1)$ and column matrices $G_d(i_d)$ can be interpreted as vectors depending on the context.

The tensor carriages G_k can be further interpreted are 3^{rd}-order tensors $\mathcal{G}_k \in \mathbb{R}^{r_{k-1}\times n_k\times r_k}$ with:

$$G_k(i_k)(p,q) = \mathcal{G}_k(p, i_k, q) \quad \forall (i_k, p, q) \in I_k \times [1:r_{k-1}] \times [1:r_k]$$

The TT decomposition can be equivalently written:

$$\mathcal{T}(i_1, \ldots, i_d) = \sum_{\alpha_1=1}^{r_1} \cdots \sum_{\alpha_{d-1}=1}^{r_{d-1}} \mathcal{G}_1^\star(i_1, \alpha_1)\mathcal{G}_2(\alpha_1, i_2, \alpha_2) \ldots \mathcal{G}_{d-1}(\alpha_{d-2}, i_{d-1}, \alpha_{d-1})\mathcal{G}_d^\star(\alpha_{d-1}, i_d) \tag{5}$$

where $\mathcal{G}_1^\star$ and $\mathcal{G}_d^\star$ are the squeezed versions of $\mathcal{G}_1$ and $\mathcal{G}_d$.

Using the tensor dot product notation, Equation (5) can be compactly rewritten:

$$\mathcal{T} = \mathcal{G}_1^\star \bullet \mathcal{G}_2 \bullet \cdots \bullet \mathcal{G}_{d-1} \bullet \mathcal{G}_d^\star \tag{6}$$

The TT format allows significant gains in terms of memory storage and therefore is well-suited to high order tensors. The storage complexity is $O(n\bar{r}^2 d)$ where $\bar{r} = \max(r_1, \ldots, r_{d-1})$ and depends linearly on the order d of the tensor. In many applications of practical interest the small TT-ranks r_k enable to alleviate the curse of dimensionality [14].

The sequential computational complexity of the evaluation of a single element of a tensor in TT format is $O\left(d\bar{r}^2\right)$. Assuming that $\bar{r}$ is small enough, the low computational cost allows a real-time evaluation of the underlying tensor. Therefore, in terms of online exploitation, this representation conforms with the expected requirements of the surrogate model. Figure 1 illustrates the sequence of matrix multiplications required to compute one element of the tensor train.

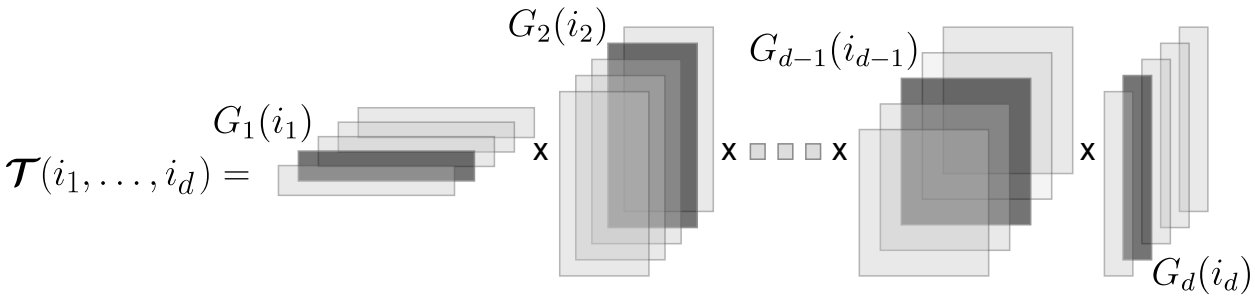


Figure 1: Illustration of the evaluation of one element of the tensor train. The entry $\mathcal{T}(i_1, \ldots, i_d) \in \mathbb{R}$ is obtained by multiplying the set of matrices $G_1(i_1), G_2(i_2) \ldots, G_d(i_d)$ identified by a darker shade.

4. Piecewise multilinear interpolation using the TT format

Implementations of basic operations such as addition and multiplication have been addressed [15] for tensor-train representations. More complex operations such as convolutions, scalar products, norms, contractions, Hadamard product and high-dimensional integration of functions, have also been studied [14, 15]. The case of piecewise multilinear interpolation is addressed here.

Consider a multidimensional function defined over a discretized domain of dimension d. The 'naive' piecewise multilinear interpolation of the function theoretically requires the evaluation of 2^d points. However, the multilinear interpolation is a tensor product of one-dimensional piecewise linear interpolations. Considering a tensor-train representation of the function and exploiting its particular structure, an efficient formula to compute the piecewise multilinear interpolation can be derived.

Consider a function:

$$\begin{array}{rccc} F^\Delta & : & \mathcal{D}^\Delta = \mathcal{D}_1^\Delta \times \cdots \times \mathcal{D}_d^\Delta & \rightarrow & \mathbb{R} \\ & & (\mu_1, \ldots, \mu_d) & \mapsto & F^\Delta(\mu_1, \ldots, \mu_d) \end{array} \tag{7}$$

where the definition domain $\mathcal{D}^\Delta$ is the Cartesian product of the sets $\mathcal{D}_k^\Delta$ representing discretized intervals of $\mathbb{R}$:

$$\mathcal{D}_k^\Delta = \left\{ \mu_k^{(i_k)} \in \mathbb{R} \mid \forall i_k \in I_k \right\} \quad \text{with} \quad I_k = [1 : n_k] \tag{8}$$

with

$$\mu_k^{(1)} < \mu_k^{(2)} < \cdots < \mu_k^{(n_k)} \tag{9}$$

The function F^Δ is naturally associated with the tensor $\mathcal{T} \in \mathbb{R}^{n_1 \times \ldots n_d}$ such that:

$$\mathcal{T}(i_1, \ldots, i_d) = F^\Delta\left(\mu_1^{(i_1)}, \ldots, \mu_d^{(i_d)}\right) \quad \forall \quad (i_1, \ldots, i_d) \in I_1 \times \cdots \times I_d \tag{10}$$

Assume that the latter admits a tensor-train decomposition such that:

$$\mathcal{T}(i_1, \ldots, i_d) = G_1(i_1) \ldots G_d(i_d) \tag{11}$$

with

$$G_k(i_k) \in \mathbb{R}^{r_{k-1} \times r_k} \quad \forall i_k \in I_k \tag{12}$$

Define the continuous domain $\mathcal{D} = \mathcal{D}_1 \times \cdots \times \mathcal{D}_d$ such that $\mathcal{D}_k = \left[\mu_k^{(1)}, \mu_k^{(n_k)}\right]$ are intervals of $\mathbb{R}$. Define over $\mathcal{D}$ a function F which corresponds to F^Δ on $\mathcal{D}^\Delta$ and is a piecewise multilinear interpolation over $\mathcal{D}$.

Proposition 1. *The evaluation of F at the point $(\mu_1, \ldots, \mu_d) \in \mathcal{D}$ reads:*

$$F(\mu_1, \ldots, \mu_d) = L_1(\mu_1) \ldots L_d(\mu_d) \tag{13}$$

with

$$L_k(\mu_k) = \frac{1}{\hat{\mu}_k^{+1} - \hat{\mu}_k^{-1}} \left[\left(\hat{\mu}_k^{+1} - \mu_k\right) G_k\left(i_k^{-1}\right) - \left(\hat{\mu}_k^{-1} - \mu_k\right) G_k\left(i_k^{+1}\right) \right]$$

where $\hat{\mu}_k^{+1}$ and $\hat{\mu}_k^{-1}$ are respectively the upper and lower closest elements to μ_k that belong to $\mathcal{D}_k^\Delta$, and i_k^{-1} and i_k^{+1} are defined such that:

$$\hat{\mu}_k^{+1} = \mu_k^{(i_k^{+1})} \quad \textit{and} \quad \hat{\mu}_k^{-1} = \mu_k^{(i_k^{-1})} \tag{14}$$

The proof is given in Appendix B.

L_k can be interpreted as a matrix-valued function defined over $\mathcal{D}_k$ that coincide with G_k on $\mathcal{D}_k^\Delta$, that is:

$$L_k\left(\mu_k^{(i_k)}\right) = G_k(i_k) \quad \text{for} \quad i_k \in I_k$$

and is a piecewise multilinear interpolation over $\mathcal{D}_k$.

The computational complexity of the evaluation of $L_k(\mu_k)$ (piecewise linear interpolation between two matrices) is $O(\bar{r}^2)$ with $\bar{r} = \max(r_0, \ldots, r_d)$. The evaluation of $F(\mu_1, \ldots, \mu_d)$ involves d interpolations of matrices $(G_1, \ldots, G_d)$. Hence, the computational complexity for interpolating all the matrices G_k is $O\left(d\bar{r}^2\right)$. Since the computational complexity to compute an element of a tensor in TT format is $O\left(d\bar{r}^2\right)$, the computational complexity for evaluating F at one point is $O\left(2d\bar{r}^2\right)$. There is only a factor 2 between the complexities of the evaluation of the piecewise multilinear interpolation and the evaluation of a tensor-train decomposition. When the function F^Δ is a restriction of a reference continuous function, the expression (13) enables to estimate efficiently an approximation of this reference function.

Remark 2. *The computational cost of finding the indices i_k^{+1} and i_k^{-1} according to Equation* (14) *may be not negligible. In particular, if the grid D_k^Δ is not regular (that is, with non constant steps) the cost is $O(\log n_k)$ (search in an ordered list). For a regular grid, the cost is independent from n_k.*

5. Generic algorithm for approximate tensor-train decomposition

5.1. Core tensors definition

Given a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, Algorithm 1 generates a set of matrices $\{H_1, \ldots, H_d\}$ that enables to define a tensor-train decomposition:

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{H}}_1^\star \bullet \boldsymbol{\mathcal{H}}_2 \bullet \cdots \bullet \boldsymbol{\mathcal{H}}_{d-1} \bullet \boldsymbol{\mathcal{H}}_d^\star \in \mathbb{R}^{n_1 \times \cdots \times n_d} \tag{15}$$

where

$$\boldsymbol{\mathcal{H}}_k \in \mathbb{R}^{s_{k-1} \times n_k \times s_k} \text{ for } k = 1, \ldots, d$$

such that :

$$\langle \boldsymbol{\mathcal{H}}_k \rangle_2 = H_k \in \mathbb{R}^{(s_{k-1} n_k) \times s_k} \tag{16}$$

$\boldsymbol{\mathcal{T}}$ is an approximation of the tensor $\boldsymbol{\mathcal{A}}$ up to an error whose expression is detailed in Section 5.2.

Remark 3. *The outputs H_k of the algorithm are not necessarily given as explicit matrices but instead may be returned as matrix decompositions. The latter format may be preferable when applying the refactoring procedure presented in Section 5.4.*

Remark 4. *For $k = d - 1$, the matricization step* (20) *returns $A_d \in \mathbb{R}^{(s_{d-1} n_d) \times 1}$, hence the convention $s_d = 1$.*

Algorithm 1: Generic TT approximation algorithm

Input: A tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ whose elements are given by a blackbox procedure.
Output: A set of matrices $\{H_1, \ldots, H_d\}$

Initial matricization:
Define the matrix $A_1 \in \mathbb{R}^{(s_0 n_1) \times (n_2 \ldots n_d)}$ with $s_0 = 1$, as the first matricization of the tensor $\mathcal{A}$:

$$A_1 = \langle \mathcal{A} \rangle_1 \tag{17}$$

for $k = 1, \ldots, d-1$ **do**

Low-rank approximation:
Build a rank-r_k approximation T_k of the matrix $A_k \in \mathbb{R}^{(s_{k-1} n_k) \times (n_{k+1} \ldots n_d)}$ given in the form:

$$T_k = H_k P_k^T A_k \quad \text{with} \quad H_k, P_k \in \mathbb{R}^{(s_{k-1} n_k) \times s_k} \quad \text{and} \quad s_k \geq r_k \tag{18}$$

Tensorization:
Define the tensor $\mathcal{A}^{(k+1)} \in \mathbb{R}^{s_k \times n_{k+1} \times \cdots \times n_d}$ such that:

$$\left\langle \mathcal{A}^{(k+1)} \right\rangle_1 = P_k^T A_k \in \mathbb{R}^{s_k \times (n_{k+1} \ldots n_d)} \tag{19}$$

Matricization:
Define the matrix $A_{k+1} \in \mathbb{R}^{(s_k n_{k+1}) \times (n_{k+2} \ldots n_d)}$ as the second matricization of the tensor $\mathcal{A}^{(k+1)}$:

$$A_{k+1} = \left\langle \mathcal{A}^{(k+1)} \right\rangle_2 \tag{20}$$

Finalization:
Define the matrix $H_d \in \mathbb{R}^{(s_{d-1} n_d) \times s_d}$ with $s_d = 1$ such that:

$$H_d = A_d \tag{21}$$

5.2. Approximation error

For $k = 1, \ldots, d-1$, define the error matrices $E_k \in \mathbb{R}^{(s_{k-1}n_k)\times(n_{k+1}\ldots n_d)}$ as:

$$E_k = A_k - T_k \tag{22}$$

and the corresponding error tensors:

$$\boldsymbol{\mathcal{E}}_k \in \mathbb{R}^{s_{k-1}\times n_k\times\cdots\times n_d}$$

such that

$$\langle \boldsymbol{\mathcal{E}}_k \rangle_2 = E_k \in \mathbb{R}^{(s_{k-1}n_k)\times(n_{k+1}\ldots n_d)} \tag{23}$$

Proposition 2. *The approximation error between a given tensor $\boldsymbol{\mathcal{A}}$ and its tensor-train approximation $\boldsymbol{\mathcal{T}}$ obtained by Algorithm 1 is:*

$$\begin{aligned} \boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{T}} = \quad & \boldsymbol{\mathcal{H}}_1^\star \bullet \boldsymbol{\mathcal{H}}_2 \bullet \cdots \bullet \boldsymbol{\mathcal{H}}_{d-2} \bullet \boldsymbol{\mathcal{E}}_{d-1} \\ & + \boldsymbol{\mathcal{H}}_1^\star \bullet \boldsymbol{\mathcal{H}}_2 \bullet \cdots \bullet \boldsymbol{\mathcal{H}}_{d-3} \bullet \boldsymbol{\mathcal{E}}_{d-2} \\ & + \ldots \\ & + \boldsymbol{\mathcal{H}}_1^\star \bullet \boldsymbol{\mathcal{E}}_2 \\ & + \boldsymbol{\mathcal{E}}_1^\star \end{aligned} \tag{24}$$

See Appendix C for proof.

A corollary (Proposition 3) of Proposition 2 exhibits an upper bound for the norm of the approximation error as a function of the norm of the output matrices H_k and the error matrices E_k.

Proposition 3. *With the hypotheses of Proposition 2:*

$$\|\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{T}}\|^2 \leq \sum_{k=1}^{d-1} \prod_{k'=1}^{k-1} \|H_{k'}\|_2^2 \, \|E_k\|^2 \tag{25}$$

See Appendix D for proof.

5.3. Low-rank approximation step

The low-rank approximation step (18) of the algorithm is expressed as an abstract matrix factorization, namely:

$$T_k = H_k P_k^T A_k$$

to emphasize that several concrete alternatives are possible. For the sake of simplicity, the rest of the article will be restricted to the cases where $H_k \in \mathbb{R}^{(s_{k-1}n_k)\times s_k}$ has rank r_k with $s_k \leq r_k$ and $P_k \in \mathbb{R}^{(s_{k-1}n_k)\times s_k}$ has orthogonal columns.

Equations (19) and (20) taken together indicate that the matrix A_{k+1}, that will be exploited at the next iteration, is a reshaped version of the matrix $P_k^T A_k$. Depending on the adopted low-rank approximation, the matrix P_k has a different structure and thus leads to different definition of the matrices A_{k+1}. That choice also determines which elements of the reference tensor must computed at each iteration. A key objective is thus to chose a low-rank approximation that retains sufficient accuracy and only requires a parsimonious exploration of the entries of $\boldsymbol{\mathcal{A}}$ to make Algorithm 1 computationally affordable. A detailed discussion of some candidates and their respective properties is postponed until Section 6.

5.4. Refactoring procedure

Depending on the selected low-rank approximation (18) in Algorithm 1, the TT decomposition (15) may be sub-optimal. More specifically, the tensor $\mathcal{A}$ may admit an alternate TT decomposition whose tensor carriages involve fewer elements. The refactoring procedure introduced below shows how to obtain a more compact tensor-train decomposition. The *refactored decomposition* is equivalent to the reference decomposition (15) in the sense that their evaluations coincide (at least in exact arithmetic). The new decomposition relies only on the outputs H_k of Algorithm 1 (See Remark 3) and therefore requires no further computation of any elements of the reference tensor $\mathcal{A}$.

The low-rank approximation (18) features a matrix $H_k \in \mathbb{R}^{(s_{k-1}n_k)\times s_k}$ of rank r_k. Hence, there exists $K_k \in \mathbb{R}^{(s_{k-1}n_k)\times r_k}$ and $L_k \in \mathbb{R}^{r_k \times s_k}$ such that:

$$H_k = K_k L_k \quad \text{with} \quad r_k \leq s_k \tag{26}$$

The matrices K_k and L_k can be further interpreted as the tensors $\mathcal{K}_k \in \mathbb{R}^{s_{k-1}\times n_k \times r_k}$ and $\mathcal{L}_k \in \mathbb{R}^{r_k \times s_k}$ such that:

$$K_k = \langle \mathcal{K}_k \rangle_2 \quad \text{and} \quad L_k = \mathcal{L}_k$$

Hence, for $k = 1, \ldots, d-1$:

$$\mathcal{H}_k = \mathcal{K}_k \bullet \mathcal{L}_k \tag{27}$$

Casting (27) into (15) reads:

$$\mathcal{T} = \mathcal{K}_1^\star \bullet \mathcal{L}_1 \bullet \mathcal{K}_2 \bullet \mathcal{L}_2 \bullet \cdots \bullet \mathcal{K}_{d-1} \bullet \mathcal{L}_{d-1} \bullet \mathcal{H}_d^\star \tag{28}$$

For $k = 1, \ldots, d$, define $\mathcal{G}_k \in \mathbb{R}^{r_{k-1}\times n_k \times r_k}$ such that:

$$\mathcal{G}_1 = \mathcal{K}_1 \tag{29}$$
$$\mathcal{G}_k = \mathcal{L}_{k-1} \bullet \mathcal{K}_k \text{ for } k = 2, \ldots, d-1 \tag{30}$$
$$\mathcal{G}_d = \mathcal{L}_{d-1} \bullet \mathcal{H}_d \tag{31}$$

Equations (28), (29), (30) and (31) enable to define the refactored TT decomposition:

$$\mathcal{T} = \mathcal{G}_1^\star \bullet \mathcal{G}_2 \bullet \cdots \bullet \mathcal{G}_{d-1} \bullet \mathcal{G}_d^\star \tag{32}$$

6. Selection of a concrete low-rank matrix approximation

The operation left to be detailed in Algorithm 1 is the construction of the low-rank approximation (18).

6.1. Singular Value Decomposition

In [14], the Singular Value Decomposition (SVD) is first considered and leads to the algorithm called TT-SVD. With usual notations, for a matrix $A \in \mathbb{R}^{n\times m}$, the low-rank approximation T_{svd} given by the truncated SVD reads:

$$A = \underbrace{V_{svd}\Sigma_{svd}W_{svd}^T}_{=T_{svd}} + E_{svd} \quad \text{where} \quad \|E_{svd}\| \leq \epsilon_{svd}\,\|A\| \tag{33}$$

where ϵ_{svd} is the truncation tolerance.

The columns of the matrix $V_{svd} \in \mathbb{R}^{n\times r}$ constitute the POD reduced basis (where POD stands for Proper Orthogonal Decomposition) and correspond to the r most significant left singular vectors of the SVD [19, Section 5.5.4] of A.

Algorithm 1 is specialized into the TT-SVD with the following matrix substitutions:

$$T_k = T_{svd} \quad \text{and} \quad H_k = P_k = V_{svd}$$

The TT-SVD enables to assert that any tensor can be represented exactly by a tensor-train decomposition. Moreover, the reference [14, Theorem 2.2] provides an upper bound for the approximation error given predefined TT-ranks.

The computation of $P_k^T A_k$ (19) requires the availability of the entire matrix A_k. Since A_1 is the first matricization of $\mathcal{A}$ (Equation (17)), all elements of the full tensor have to be explicitly computed at the first iteration. Furthermore, at all subsequent iterations, the whole matrices A_k have to be assembled and stored. The resulting computational burden makes the TT-SVD intractable in high parametric dimension.

The remedy shared by the alternative low-rank approximations presented in Section 6.2 to 6.4 consists in using only a parsimonious exploration of the reference tensor to keep the computational effort reasonable.

6.2. Snapshot POD

A first approach consists in avoiding building the POD basis from the full matrix A_k. An approximate POD reduced basis of rank r_k can be obtained via the Snapshot POD method [20]. The method consists in applying the truncated SVD on the submatrix $\tilde{A} = A(:, \mathcal{J}_{pod})$ constituted by a selection of columns $\mathcal{J}_{pod}$ of A. Hence the accuracy of the resulting POD reduced basis relies on the quality of the sampling procedure that generally introduces a sampling error. The discussion of the selection of $\mathcal{J}_{pod}$ is postponed until Section 8.

Using this approach leads to an alternate algorithm, referred to as the *TT-POD*, with the following matrix substitutions:

$$T_k = T_{pod}, \quad H_k = V_{pod} V_{pod}^T, \quad \text{and} \quad P_k = \mathbb{I}$$

In this case, the recursive definition of A_k indicates that it corresponds to the k^{th} matricization of the tensor $\mathcal{A}$. Using the Snapshot POD to build the basis matrix V_k is not sufficient to relieve the curse of dimensionality. Indeed, the number of rows of the matrices A_k grows with k and becomes too large to consider computing a single column. Hence, the TT-POD remains practically unfeasible for large tensors.

6.3. Pseudo-Skeleton Decomposition

A more practical approach to effectively construct an approximate TT decomposition, called the TT-cross method, is proposed in [14]. The TT-cross consists in dropping the concept of a POD basis and using the Pseudo-Skeleton Decomposition (PSD) introduced in [21] as low-rank approximation. Unlike the TT-SVD and the TT-POD, the TT-cross enables to build an approximation based on a sparse exploration of the reference tensor. The *TT-PSD* presented hereafter, is a specialization of Algorithm 1 where the low-rank approximation is the PSD. It is similar to the first left-to-right sweep of the TT-cross.

The Pseudo-Skeleton Decomposition can be used to approximate any matrix $A \in \mathbb{R}^{n \times m}$ and is written as:

$$A = \underbrace{A Q_{psd} \left[P_{psd}^T A Q_{psd} \right]^{-1} P_{psd}^T A}_{= T_{psd}} + E_{psd} \tag{34}$$

where $P_{psd} \in \mathbb{R}^{n \times s}$ and $Q_{psd} \in \mathbb{R}^{m \times s}$ are respectively row and column selection matrices associated with the sets $\mathcal{I}_{psd}$ and $\mathcal{J}_{psd}$. The definition is valid only when the matrix $P_{psd}^T A Q_{psd}$ is non-singular. In particular, the number s of rows and columns has to be identical. The approximation decomposition (34) can be written with the colon notation as:

$$T_{psd} = A(:, \mathcal{J}_{psd}) \left[A(\mathcal{I}_{psd}, \mathcal{J}_{psd}) \right]^{-1} A(\mathcal{I}_{psd}, :)$$

This approximation (34) features an interpolation property at the selected rows and columns:

$$T(\mathcal{I}_{psd}, :) = A(\mathcal{I}_{psd}, :) \quad \text{and} \quad T(:, \mathcal{J}_{psd}) = A(:, \mathcal{J}_{psd}) \tag{35}$$

The Pseudo-Skeleton Decomposition is a matrix factorization similar to the decomposition used in the Adaptive Cross Approximation (ACA) [22] and the CUR decomposition [23, 24]. Additionally, these references provide algorithms to effectively build the factorization. That decomposition has also been used in the context of model order reduction, for instance in the Empirical Interpolation Method (EIM) proposed in [25, 26].

Algorithm 1 specializes into the *TT-PSD* with the following matrix substitutions:

$$T_k = T_{psd}, \quad H_k = AQ_{psd}\left[P_{psd}^T A Q_{psd}\right]^{-1} \quad \text{and} \quad P_k = P_{psd}$$

P_k is a row selection matrix associated with the set $\mathcal{I}_k$ of s_k rows of A_k. Hence, the matrix A_{k+1}, used at the next iteration, consists of a reshaped version of the submatrix $A_k(\mathcal{I}_k, :)$. By induction, it can be shown that the matrices A_k are constituted by sets of multi-indices of the reference tensor $\mathcal{A}$. Each iteration amounts then to the computation a low-rank approximation of the k^{th} matricization of a subtensor of $\mathcal{A}$. The number of rows of A_k is then limited and enables in practice to construct the low-rank approximation when $s_k \ll s_{k-1}n_k$. The provided sparse exploration makes the TT-PSD affordable to compute.

6.4. Gappy POD

A last, novel approach consists in using the Gappy POD introduced in [27] as the low-rank approximation in (18). This strategy aims at combining beneficial features of the Snapshot POD and the PSD. Indeed, the Gappy POD a) relies on a POD basis that remains computationally affordable and b) requires only a limited number of rows of the matrix to be approximated. Both properties are key ingredients for an efficient, parsimonious exploration of the reference tensor during Algorithm 1.

6.4.1. Formulation

The Gappy POD approximation T_{gap} of a matrix $A \in \mathbb{R}^{n\times m}$ is given by:

$$A = \underbrace{V_{pod}\left[P_{gap}^T V_{pod}\right]^\dagger P_{gap}^T A}_{= T_{gap}} + E_{gap} \tag{36}$$

where † denotes the Moore-Penrose pseudo-inverse [19, Section 5.5.2]. $V_{pod} \in \mathbb{R}^{n\times r}$ is a POD basis matrix of rank r and $P_{gap} \in \mathbb{R}^{n\times s}$ is a row selection matrix associated with a set of s rows $\mathcal{I}_{gap} \subset [1:n]$.

The matrix $P_{gap}^T V_{pod}$ must have linearly independent columns to ensure that the approximation is meaningful. Since V_{pod} is a rank-r POD basis, there exists a set of s rows such that this property holds as long as $s \geq r$. In the numerical results presented hereafter the rows are selected using the Q-DEIM algorithm [28] that was shown to be a superior alternative to the better-known DEIM procedure [29, Algorithm 1]. For the sake of completeness, the Q-DEIM algorithm is detailed in Appendix E.

Remark 5. *When $P_{gap}^T V_{pod}$ has linearly independent columns, its Moore-Penrose pseudoinverse can be computed as:*

$$\left[P_{gap}^T V_{pod}\right]^\dagger = \left[V_{pod}^T P_{gap} P_{gap}^T V_{pod}\right]^{-1} V_{pod}^T P_{gap}$$

It follows that $\left[P_{gap}^T V_{pod}\right]^\dagger$ is a left inverse of $P_{gap}^T V_{pod}$ since:

$$\left[P_{gap}^T V_{pod}\right]^\dagger P_{gap}^T V_{pod} = \mathbb{I}_r$$

Unlike the PSD, the Gappy POD enables to select a number of rows that exceeds the rank of the low-rank approximation. In this case, the interpolation property does not hold as in the PSD case (35).

Algorithm 1 is specialized into the proposed *TT-Gappy* with the following matrix substitutions:

$$T_k = T_{gap}, \quad H_k = V_{pod}\left[P_{gap}^T V_{pod}\right]^\dagger \quad \text{and} \quad P_k = P_{gap} \tag{37}$$

Similarly to the PSD case, the row selection matrix P_{gap} enables a sparse exploration of the reference tensor $\mathcal{A}$ in Algorithm 1.

6.4.2. Error bound for the TT decomposition

To quantify the theoretical accumulation of errors introduced at each iteration, Proposition 4 gives an upper bound for the approximation error associated with a tensor-train decomposition built by the TT-Gappy.

Proposition 4. *Consider $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ and its tensor-train approximation $\mathcal{T}$ constructed by the TT-Gappy. Assuming that for all $k \in [1 : d-1]$*

$$\exists \nu_k, \quad \left\| \left(\mathbb{I} - V_k V_k^T \right) A_k \right\| \leq \nu_k \left\| A_k \right\| \tag{38}$$

the following inequality holds:

$$\left\| \mathcal{A} - \mathcal{T} \right\|^2 \leq \sum_{k=1}^{d-1} \frac{\nu_k^2}{\left[\sigma_{min}\left(P_1^T V_1 \right) \sigma_{min}\left(P_2^T V_2 \right) \ldots \sigma_{min}\left(P_k^T V_k \right) \right]^2} \left\| \mathcal{A} \right\|^2 \tag{39}$$

where σ_{min} refers to the smallest singular value of its matrix argument.

The proof is given in Appendix F.

Remark 6. *The property* (38) *is always true for the choice $\nu_k = 1$. Moreover, if V_k is computed as a truncated POD basis of rank r_k of the full matrix A_k then:*

$$A_k = V_k \Sigma_k W_k^T + E_{svd,k} \quad with \quad \left\| E_{svd,k} \right\| \leq \epsilon_k \left\| A_k \right\| \tag{40}$$

and the property (38) *and therefore the bound* (39) *hold for $\nu_k = \epsilon_k$.*

Remark 6 suggests that the TT-Gappy approximation error (24) can be controlled by the truncation tolerances ϵ_k set by the user. However, the bound (39) tends to be very loose and the hypothesis (38) may be difficult to verify when the basis V_k stems from a column sampling of the matrix A_k. Hence, the convergence should be assessed empirically in practical cases.

6.5. Relevance of the refactoring procedure

Table 1 summarizes the actual definitions of the matrices K_k, L_k and P_k involved in Algorithm 1 and the refactoring procedure 5.4 for the four types of low-rank approximation introduced earlier.

Low-rank approximation	K_k	L_k	P_k
SVD	V_{svd}	$\mathbb{I}_r$	V_{svd}
Snapshot POD	V_{pod}	V_{pod}^T	$\mathbb{I}_r$
PSD	$AQ_{psd}\left[P_{psd}^T A Q_{psd} \right]^{-1}$	$\mathbb{I}_r$	P_{psd}
Gappy POD	$V_{pod}\left[V_{pod}^T P_{gap} P_{gap}^T V_{pod} \right]^{-1}$	$V_{pod}^T P_{gap}$	P_{gap}

Table 1: Concrete definitions of the matrices appearing in expressions (18) and (26). $\mathbb{I}_r \in \mathbb{R}^{r \times r}$ stands for the identity matrix.

The usefulness of the refactoring procedure varies according to the selected type of low-rank matrix approximation. In the case of the SVD and the PSD, the tensors $\mathcal{H}_k$ (16) are already given in a compact format. The refactoring procedure does not provide a more compressed decomposition. Even if it has been shown in Section 6.2 that the Snapshot POD has no practical value, the refactoring procedure provides a large compression that in theory allows to store the compressed tensor train unlike the initial representation. Finally, the size of the tensors $\mathcal{H}_k$ constructed following the Gappy POD approach depends on the number of sampled rows and the latter may be larger than the rank of the approximations. The refactoring procedure allows in this case to redefine the tensor carriages such that their respective sizes scale with the TT-ranks of the decomposition.

7. Problem formulation

7.1. Differential algebraic equations

Consider a system of parameterized Differential Algebraic Equations (DAEs):

$$\boldsymbol{R}\left(\boldsymbol{Y}^1, \ldots, \boldsymbol{Y}^N, \dot{\boldsymbol{Y}}^1, \ldots, \dot{\boldsymbol{Y}}^N, \boldsymbol{\mu}, t \right) = 0 \tag{41}$$

where

- $\boldsymbol{R}$ is a nonlinear operator;
- The parameters are denoted by $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_{d-1}) \in \mathcal{D} = \mathcal{D}_1 \times \cdots \times \mathcal{D}_{d-1}$ with $d > 0$;
- The time is $t \in [0, T]$ where $T > 0$;
- The variables $\boldsymbol{Y}^\chi$ (for $\chi = 1, \ldots, N$) are referred to as the outputs of the DAEs.

The parameter domain $\mathcal{D}$ is defined as a Cartesian product of $d-1$ intervals $\mathcal{D}_k \subset \mathbb{R}$. Such DAEs are typically introduced in mechanics of materials to setup constitutive equations [30, 31]. In these applications, the outputs $\boldsymbol{Y}^\chi$ can stand for internal variables, stress or strain.

Assuming that the hypotheses of the implicit function theorem are fulfilled, each output $\boldsymbol{Y}^\chi$ can be interpreted as a time-dependent tensor-valued function with m^χ components:

$$\begin{array}{rcccl} \boldsymbol{Y}^\chi & : & \mathcal{D}_1 \times \cdots \times \mathcal{D}_{d-1} \times [0,T] & \to & \mathbb{R}^{m^\chi} \\ & & (\mu_1, \ldots, \mu_{d-1}, t) & \mapsto & \boldsymbol{Y}^\chi(t, \boldsymbol{\mu}) = \begin{pmatrix} Y_1^\chi(t, \boldsymbol{\mu}) \\ \vdots \\ Y_{m^\chi}^\chi(t, \boldsymbol{\mu}) \end{pmatrix} \end{array}$$

with $Y_j^\chi(t, \boldsymbol{\mu}) \in \mathbb{R}$ being the j^{th} component of the output χ, at time t, for the parameter set value $\boldsymbol{\mu}$.

The outputs are described as *heterogeneous* in the sense that they may have different physical units and/or different order of magnitudes. As a consequence, they might not be compared to each others in a meaningful way. This observation justifies the construction of a distinct surrogate model for each output of interest.

Remark 7. *The solution of the DAE system for a given set of parameter values $\boldsymbol{\mu}$ provides all outputs $\boldsymbol{Y}^\chi$ at all times $t \leq t_f$. Indeed:*

- *Unlike the parameters, time is a causal coordinate. The value of $\boldsymbol{Y}^\chi(t_f, \boldsymbol{\mu})$ at time t_f depends on the solution at the previous time $t \leq t_f$.*
- *The outputs are correlated which means that they stem from the same DAE system.*

7.2. Tensor representation of outputs

Discretizations of the parameter space and the time interval are introduced to define the tensor representations for the various outputs $\boldsymbol{Y}^\chi$. The time integration scheme used to solve the DAEs relies on a time sampling $\mathcal{T}^\Delta$ of the interval $[0, T]$:

$$\mathcal{T}^\Delta = \left\{ t^{(i_t)} \in [0,T] \mid \forall i_t \in I_t \right\} \quad \text{with} \quad t^{(1)} < \cdots < t^{(n_t)} \tag{42}$$

where $I_t = [1 : n_t]$ and n_t is the number of time steps.

The discretization of the parameter domain $\mathcal{D}$ is given by $\mathcal{D}^\Delta = \mathcal{D}_1^\Delta \times \cdots \times \mathcal{D}_{d-1}^\Delta$ and is similar to the one introduced in Section 4. For $\chi = 1, \ldots, N$, define the component index range $I_{comp}^\chi = [1 : m^\chi]$. m^χ may differ from one output to another.

The introduced discretizations leads to associate every output $\boldsymbol{Y}^\chi$ with a *physics-based* (or *reference*) tensor $\boldsymbol{\mathcal{A}}^\chi$. The latter aggregates the values of the output χ for all time steps and all possible parameter values in the discretized parameter domain such that:

$$\forall \left(i_1, \ldots, i_{d-1}, i_t, i_{comp}\right) \in I_1 \times \ldots I_{d-1} \times I_t \times I_{comp}^\chi \quad \boldsymbol{\mathcal{A}}^\chi \left(i_1, \ldots, i_{d-1}, \left(i_t, i_{comp}\right)\right) = Y_{i_{comp}}^\chi \left(t^{(i_t)}; \mu_1^{(i_1)}, \ldots, \mu_{d-1}^{(i_{d-1})}\right) \tag{43}$$

Without restricting the generality of the proposed approach, the last two indices are paired in a single multi-index $\left(i_t, i_{comp}\right)$. Consequently $\boldsymbol{\mathcal{A}}^\chi$ is a tensor of order d such that:

$$\boldsymbol{\mathcal{A}}^\chi \in \mathbb{R}^{n_1 \times \cdots \times n_{d-1} \times n_d^\chi} \quad \text{where } n_d^\chi = n_t m^\chi \tag{44}$$

Accessing the elements of the physics-based tensors requires to solve the DAEs for the right choice of parameter values. Depending on the considered physical model, the access can be expensive preventing a real-time exploration to the elements.

Remark 8. *The correlation between the outputs mentioned in Remark 7 is expressed as follows: For all* $\left(\chi, i_t, i_{comp}\right) \in [1:N] \times I_t \times I_{comp}^{\chi}$ *and a single multi-index* $(i_1, \ldots, i_{d-1}) \in I_1 \times \cdots \times I_{d-1}$*, the elements*

$$\mathcal{A}^{\chi}\left(i_1, \ldots, i_{d-1}, \left(i_t, i_{comp}\right)\right) \in \mathbb{R}$$

are associated with the same DAE solution. As a consequence, these elements are always computed simultaneously. This consideration will be factored in when building the surrogate model.

7.3. Approximation errors

A *time discretization error* (in addition to the modelisation error) is introduced when solving the DAE system with a time integration algorithm on the interval $[0, T]$. This source of error is not addressed here. In fact, the solutions produced by the numerical model are considered as "the truth" in the sense that the quality of the surrogate model is assessed with respect to them.

Since the physics-based tensors correspond to the solutions of the DAEs for parameter values on the discretized grid, a parametric *interpolation error* is also introduced when evaluating the surrogate model for parameter values that do not belong to the discretized grid. However, the piecewise multilinear interpolation method can be efficiently transposed to functions represented by a tensor-train decomposition (See Section 4). The interpolation error can thus be reduced by refining the discretized grid.

8. Multiple TT algorithm

The objective of the proposed approach is to build for each physics-based tensor $\mathcal{A}^{\chi}$ (44) an approximate tensor $\mathcal{T}^{\chi}$ given in TT format. Algorithm 2 provides the set of matrices $\{H_1^{\chi}, \ldots, H_d^{\chi}\}$ that enable to define the tensor-train decompositions as in Section 5.1. The refactoring procedure (Section 5.4) can be transposed straightforwardly for any tensor given in TT format.

Remark 8 asserts that solving the DAEs for one set of parameter values provides multiple elements of all physics-based tensors $\mathcal{A}^{\chi}$. The idea of the approach, embodied in Algorithm 2, is to apply simultaneously the TT-Gappy algorithm on the physics-based tensors to construct tensor-train approximations based on shared simulation results.

The method provided by Algorithm 2 is non-intrusive. Indeed, the implementation details of the numerical solution of the DAEs are completely irrelevant.

8.1. Snapshot column sampling

At each iteration $k = 1, \ldots, d-1$, the Snapshot POD method, used to build the POD reduced basis (46), requires to sample a set:

$$\mathcal{J}_k^{\chi} = \left\{\mathcal{J}_k^{\chi,(1)}, \ldots, \mathcal{J}_k^{\chi,(\tilde{n}_k^{\chi})}\right\}$$

of $\tilde{n}_k^{\chi}$ columns of $A_k^{\chi} \in \mathbb{R}^{(s_{k-1}n_k)\times\left(n_{k+1}\ldots n_{d-1}n_d^{\chi}\right)}$ for every χ .

Recall that A_k^{χ} is a submatrix of the k^{th} matricization of the tensor $\mathcal{A}^{\chi}$. According to Remark 8, the entire range of indices of the last dimension for $\mathcal{A}^{\chi}$ correspond to the same DAE solution and are therefore obtained simultaneously. The column sampling amounts, consequently, to a parsimonious selection of $\tilde{n}_k$ points in the partial discretized parameter domain $\mathcal{D}_{k+1}^{\Delta} \times \cdots \times \mathcal{D}_{d-1}^{\Delta}$ and an exhaustive sampling of the last dimension for each tensor $\mathcal{A}^{\chi}$. The considered submatrices $\tilde{A}_k^{\chi} = A_k^{\chi}\left(:, \mathcal{J}_k^{\chi}\right)$ are then constituted of $\tilde{n}_k^{\chi} = \tilde{n}_k n_d^{\chi}$ columns (See Figure 2).

In the present work, the sampling of the partial discretized domain $\mathcal{D}_{k+1}^{\Delta} \times \cdots \times \mathcal{D}_{d-1}^{\Delta}$ is done by a design of experiment based on a low-discrepancy Halton sequence [32]. The estimation of the quality of the selection is carried out a posteriori thanks to a measurement of the error between the surrogate and the physical model (See Section 9.4).

Algorithm 2: Multiple TT decomposition

Input: Tensors $\mathcal{A}^{\chi} \in \mathbb{R}^{n_1 \times \cdots \times n_{d-1} \times n_d^{\chi}}$ for $\chi = 1, \ldots, N$ associated with a DAE system (Equation (41))
Output: Sets of matrices $\left\{H_1^{\chi}, \ldots, H_d^{\chi}\right\}$ for $\chi = 1, \ldots, N$.
Initialization:
For each χ, define the matrix $A_1^{\chi} \in \mathbb{R}^{(s_0 n_1) \times \left(n_2 \ldots n_{d-1} n_d^{\chi}\right)}$ with $s_0 = 1$, as the first matricization of the tensor $\mathcal{A}^{\chi}$:

$$A_1^{\chi} = \langle \mathcal{A}^{\chi} \rangle_1 \tag{45}$$

for $k = 1, \ldots, d-1$ **do**

Snapshot POD:
Define consistent sets of sampling columns $\mathcal{J}_k^{\chi}$ (See Section 8.1). And compute from the DAEs the submatrices:

$$\tilde{A}_k^{\chi} = A_k^{\chi}\left(:, \mathcal{J}_k^{\chi}\right) \quad \text{for} \quad \chi = 1, \ldots, N$$

Apply the truncated SVD (33) on each $\tilde{A}_k^{\chi}$ with the truncation tolerance ϵ_k to get the rank-r_k^{χ} matrices:

$$V_k^{\chi} \in \mathbb{R}^{(s_{k-1} n_k) \times r_k^{\chi}} \quad \text{for} \quad \chi = 1, \ldots, N \tag{46}$$

Row Sampling:
From each χ, select a set of rows $\mathcal{I}_k^{\chi}$ applying the Q-DEIM algorithm [28] (See Algorithm 3 in Appendix E) to the basis V_k^{χ}. Define the union of all selected rows and the corresponding row selection matrix:

$$\mathcal{I}_k = \bigcup_{\chi=1}^{N} \mathcal{I}_k^{\chi} \tag{47}$$

and

$$P_k = \mathbb{I}_{s_{k-1} n_k}(:, \mathcal{I}_k) \in \mathbb{R}^{(s_{k-1} n_k) \times s_k} \quad \text{where} \quad s_k = \text{Card}(\mathcal{I}_k) \tag{48}$$

Output definitions:
Compute the matrices $H_k^{\chi} \in \mathbb{R}^{(s_{k-1} n_k) \times s_k}$ such that:

$$H_k^{\chi} = V_k^{\chi} \left[(P_k)^T V_k^{\chi}\right]^{\dagger}$$

Tensorization:
Define the tensors $\mathcal{A}^{\chi,(k+1)} \in \mathbb{R}^{s_k \times n_{k+1} \times \cdots \times n_{d-1} \times n_d^{\chi}}$ such that:

$$\left\langle \mathcal{A}^{\chi,(k+1)} \right\rangle_1 = P_k^T A_k^{\chi} \in \mathbb{R}^{s_k \times \left(n_{k+1} \ldots n_{d-1} n_d^{\chi}\right)} \tag{49}$$

Matricization:
Define the matrix $A_{k+1}^{\chi} \in \mathbb{R}^{(s_k n_{k+1}) \times \left(n_{k+2} \ldots n_{d-1} n_d^{\chi}\right)}$ as the second matricization of the tensor $\mathcal{A}^{\chi,(k+1)}$:

$$A_{k+1}^{\chi} = \left\langle \mathcal{A}^{\chi,(k+1)} \right\rangle_2 \tag{50}$$

Finalization:
For each $\chi = 1, \ldots, N$, define the matrix $H_d^{\chi} \in \mathbb{R}^{\left(s_{d-1} n_d^{\chi}\right) \times s_d}$ with $s_d = 1$ such that:

$$H_d^{\chi} = A_d^{\chi} \tag{51}$$

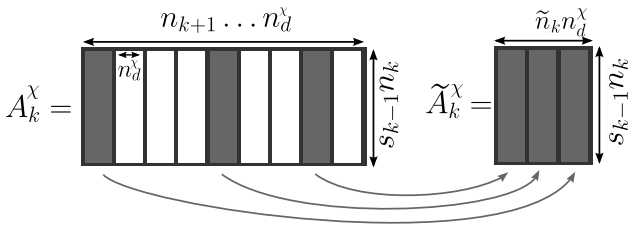


Figure 2: Definition of the submatrix $\tilde{A}_k^\chi$ used to construct the POD reduced basis. In the illustration, the Snapshot POD sample size is $\tilde{n}_k = 3$

8.2. Row sampling

In the row sampling step, specific sets of rows $\mathcal{I}_k^\chi$ are first determined independently for each output χ but a common, aggregated set $\mathcal{I}_k$ (47) is then used to sample the entries of all outputs via P_k (48). Indeed, according to Remark 8, computing the elements of all submatrices $A_k^\chi(\mathcal{I}_k, :)$ requires no more calls to the DAE system solver than evaluating the entries of all (smaller) submatrices $A_k^\chi\left(\mathcal{I}_k^\chi, :\right)$. Furthermore, the Gappy POD naturally accommodates a number of rows larger than the rank r_k^χ for each approximation of A_k^χ, and considering a larger sample size for each individual χ is expected to provide an approximation with a better precision.

Remark 9. *The idea of aggregating sample points generated for various outputs of a simulation has already been used in the context of projection-based model order reduction, for instance in the hyper-reduction approach [33] or in the GNAT method [34].*

8.3. Tensorization and matricization

The tensorization and matricization steps define the matrices to be approximated at the next iteration. The recursive definition of the matrix A_k^χ implies that the latter is equal to the k^{th} matricization of a subtensor extracted from $\mathcal{A}^\chi$. Equivalently, the matrix A_k^χ corresponds to a submatrix of the k^{th} matricization of $\mathcal{A}^\chi$, as illustrated in Figure 3.

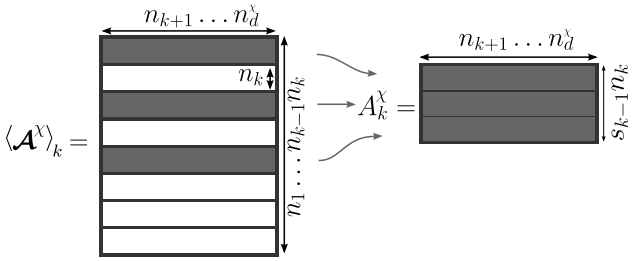


Figure 3: Definition of A_k^χ based on $\mathcal{A}^\chi$. In the illustration, the number of rows selected at the previous iteration $k-1$ is $s_{k-1} = 3$.

9. Application to elasto-viscoplastic constitutive equations

9.1. Physical model

The application case consists of a nonlinear constitutive law in elasto-viscoplasticity [30, 31] linking the following time-dependent mechanical variables:

- The strain tensor: $\underset{\sim}{\boldsymbol{\varepsilon}} = \underset{\sim}{\boldsymbol{\varepsilon}}_e + \underset{\sim}{\boldsymbol{\varepsilon}}_{vp}$ [Dimensionless] (sum of an elastic part and a viscoplastic part);
- The stress tensor: $\underset{\sim}{\boldsymbol{\sigma}}$ [MPa];
- An internal hardening variable: $\underset{\sim}{\boldsymbol{X}}$ [MPa];
- The cumulative viscoplastic deformation: p [Dimensionless].

where $\underset{\sim}{\boldsymbol{\varepsilon}}$, $\underset{\sim}{\boldsymbol{\varepsilon}}_e$, $\underset{\sim}{\boldsymbol{\varepsilon}}_{vp}$, $\underset{\sim}{\boldsymbol{\sigma}}$ and $\underset{\sim}{\boldsymbol{X}}$ are second order tensors in $\mathbb{R}^{3\times3}$.

The hypotheses of the infinitesimal strain theory are assumed to hold.

The model involves eight material coefficients: E, ν, n, K, R_0, Q, b and C. The Young and Poisson coefficients are set to $E = 200\,000$ MPa and $\nu = 0.3$. Table 2 presents the range of variation of the other material coefficients considered as inputs parameters of the model.

	n	K [MPa.s^{-n}]	R_0 [MPa]	Q [MPa]	b	C [MPa]
Lower bound	2	100	1	1	0.02	150
Upper bound	12	10 000	200	2 000	2 000	150 000

Table 2: Parameter range of variations considered in the model. When applicable, the unit is indicated between brackets.

9.1.1. System of equations

The elastic behavior is governed by:

$$\underset{\sim}{\boldsymbol{\sigma}} = \frac{E}{1+\nu}\left(\underset{\sim}{\boldsymbol{\varepsilon}}_e + \frac{\nu}{1-2\nu} Tr\left(\underset{\sim}{\boldsymbol{\varepsilon}}_e\right)\underset{\sim}{\mathbb{I}}\right) \tag{52}$$

The viscoplastic behavior is described by the Norton flow rule (53) formulated with the von Mises criterion (56). The yield function and the normal to the yield function are given by (54) and (55). (57) gives the definition of the deviatoric stress tensor involved in (56).

$$\frac{d}{dt}\underset{\sim}{\boldsymbol{\varepsilon}}_{vp} = \underset{\sim}{\boldsymbol{N}}\left\langle\frac{f}{K}\right\rangle_+^n \tag{53}$$

$$f = J\left(\underset{\sim}{\boldsymbol{\sigma}}^D - \underset{\sim}{\boldsymbol{X}}\right) - R \tag{54}$$

$$\underset{\sim}{\boldsymbol{N}} = \frac{3}{2}\frac{\underset{\sim}{\boldsymbol{\sigma}}^D - \underset{\sim}{\boldsymbol{X}}}{J\left(\underset{\sim}{\boldsymbol{\sigma}}^D - \underset{\sim}{\boldsymbol{X}}\right)} \tag{55}$$

$$J\left(\underset{\sim}{\boldsymbol{\sigma}}^D - \underset{\sim}{\boldsymbol{X}}\right) = \sqrt{\frac{3}{2}\left(\underset{\sim}{\boldsymbol{\sigma}}^D - \underset{\sim}{\boldsymbol{X}}\right):\left(\underset{\sim}{\boldsymbol{\sigma}}^D - \underset{\sim}{\boldsymbol{X}}\right)} \tag{56}$$

$$\underset{\sim}{\boldsymbol{\sigma}}^D = \underset{\sim}{\boldsymbol{\sigma}} - \frac{1}{3}Tr\left(\underset{\sim}{\boldsymbol{\sigma}}\right)\underset{\sim}{\mathbb{I}} \tag{57}$$

where $\langle . \rangle_+$ denotes the positive part function.

The operator **:** denotes the contracted product defined as:

$$\underset{\sim}{\boldsymbol{Z}}_1 : \underset{\sim}{\boldsymbol{Z}}_2 = \sum_{i=1}^{3}\sum_{j=1}^{3} Z_1^{ij} Z_2^{ij} \quad \text{for} \quad \underset{\sim}{\boldsymbol{Z}}_1, \underset{\sim}{\boldsymbol{Z}}_2 \in \mathbb{R}^{3\times3}$$

The nonlinear isotropic hardening is modeled by (58) where (59) gives the viscoplastic cumulative rate.

$$R = R_0 + Q\left(1 - e^{-bp}\right) \tag{58}$$

$$\frac{dp}{dt} = \sqrt{\frac{2}{3}\frac{d}{dt}\underset{\sim}{\boldsymbol{\varepsilon}}_{vp} : \frac{d}{dt}\underset{\sim}{\boldsymbol{\varepsilon}}_{vp}} \tag{59}$$

Finally the linear kinematic hardening is given by:

$$\underset{\sim}{\boldsymbol{X}} = \frac{2}{3}C\underset{\sim}{\boldsymbol{\varepsilon}}_{vp} \tag{60}$$

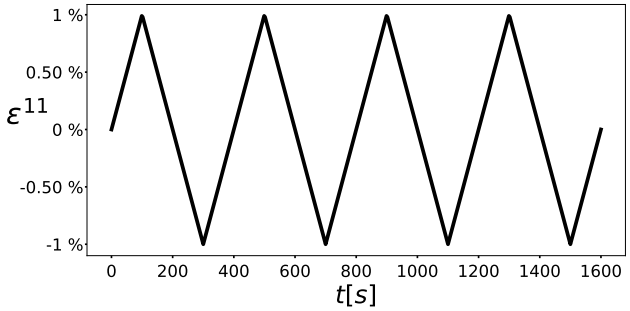


Figure 4: The applied strain component $\varepsilon^{11}(t)$ consists of a triangular pattern of period 400s with a peak-to-peak amplitude of 2% centered in 0.

9.1.2. Applied deformation and initial conditions

The case of a uniaxial cyclic tensile testing driven by deformation is considered. The loading is applied by imposing $\varepsilon^{11}(t)$ with the pattern shown in Figure 4 and $\sigma^{12}(t) = \sigma^{13}(t) = \sigma^{23}(t) = \sigma^{22}(t) = \sigma^{33}(t) = 0$.

The initial conditions for the internal variables are:

$$p(t = 0) = 0 \quad \text{and} \quad \underset{\sim}{\boldsymbol{X}}(t = 0) = \underset{\sim}{\mathbf{0}}$$

The model is highly nonlinear. First the isotropic hardening law introduces an exponential nonlinearity. The most significant nonlinearity arises from the Norton law (53) featuring the positive part function. Capturing the resulting threshold effect is particularly challenging for surrogate models.

9.2. Tensor abstraction

Algorithm 2 is applied to build tensor-train representations for the following mechanical variables involved in the model:

$$\underset{\sim}{\boldsymbol{\varepsilon}}, \quad \underset{\sim}{\boldsymbol{\varepsilon}}_{vp}, \quad \underset{\sim}{\boldsymbol{\sigma}} \text{ [MPa]}, \quad p$$

Surrogate modeling aims at representing the relations between the parameters (inputs of the model) and the time-dependent mechanical variables (outputs of the model):

$$(n, K, R_0, Q, b, C) \mapsto \left(\underset{\sim}{\boldsymbol{\varepsilon}}(t), \underset{\sim}{\boldsymbol{\varepsilon}}_{vp}(t), \underset{\sim}{\boldsymbol{\sigma}}(t), p(t)\right)$$

For each parameter, the interval of definition is discretized as in (8) by a regular grid with 30 points:

$$n_1 = n_2 = n_3 = n_4 = n_5 = n_6 = 30$$

The time interval used for the numerical solution is discretized as in (42) by a regular grid with $n_t = 537$ points.

To be consistent with the notations of the framework (41), define:

$$(\mu_1, \mu_2, \ldots, \mu_6) = (n, K, R_0, Q, b, C)$$

Following Voigt's notations, the outputs are grouped as follows:

$$\begin{aligned}
\boldsymbol{Y}^1 &= \left(\varepsilon^{11}, \varepsilon^{22}, \varepsilon^{33}, \varepsilon^{12}, \varepsilon^{13}, \varepsilon^{23}\right)^T \\
\boldsymbol{Y}^3 &= \left(\varepsilon^{11}_{vp}, \varepsilon^{22}_{vp}, \varepsilon^{33}_{vp}, \varepsilon^{12}_{vp}, \varepsilon^{13}_{vp}, \varepsilon^{23}_{vp}\right)^T \\
\boldsymbol{Y}^2 &= \left(\sigma^{11}, \sigma^{22}, \sigma^{33}, \sigma^{12}, \sigma^{13}, \sigma^{23}\right)^T \\
\boldsymbol{Y}^4 &= (p)
\end{aligned}$$

The discretizations allow to define 4 physics-based tensors:

$$\mathcal{A}^{\chi} \in \mathbb{R}^{n_1 \times \cdots \times n_6 \times n_7^{\chi}} \quad \text{for} \quad \chi = 1, 2, 3, 4 \tag{61}$$

where the time index and component index have been paired in the last multi-index as described in Equation (44) such that:

$$n_7^1 = n_7^2 = n_7^3 = 6n_t \quad \text{and} \quad n_7^4 = n_t$$

The Snapshot POD sample sizes (defined in Section 8.1) are:

$$\tilde{n}_1 = \tilde{n}_2 = \tilde{n}_3 = \tilde{n}_4 = \tilde{n}_5 = 100 \quad \text{and} \quad \tilde{n}_6 = 30$$

In the following applications, a common value denoted by ϵ_{svd} is used for all truncation tolerances ϵ_k associated with the POD basis at iterations $k \in [1 : d-1]$.

9.3. Performance indicators

The truncation tolerance is chosen here to be $\epsilon_{svd} = 10^{-3}$. The construction of the tensor-train decompositions requires to solve the system of DAEs $\sum_{k=1}^{d-1} s_k n_k \tilde{n}_k$ times with as many sets of parameter values. In the proposed numerical example, it amounts to 514 050 solutions. 15 hours are necessary on a 16-core workstation to carry out the computations. 98% of the effort is devoted to the solution of the physical model and the remaining 2% to the decomposition operations.

For a single simulation on a personal laptop computer, the solution of the physical model takes 0.7 s, whereas the surrogate model is evaluated in only 1 ms, corresponding to a speed-up of 700.

Storing the Multiple TT approximations requires 2 709 405 double-precision floating-point values. For comparison purposes, storing a single solution (constituted of the multiple time-dependent outputs) of the DAE system involves 10 203 values. Therefore, the storage of the tensor-train decompositions is commensurate with the storage of 265 solutions while it can express the approximation of 30^6 solutions.

For $\chi = 1, \ldots, 4$, the rank r_k^{χ} is bounded from above by the theoretical maximum rank $r_{max,k}^{\chi}$ of the matrix A_k^{χ}. More specifically, $r_{max,k}^{\chi}$ corresponds to the case where A_k^{χ} has full rank and is the k^{th} matricizations of the tensors $\mathcal{A}^{\chi}$. Given the choice of truncation tolerance $\epsilon_{svd} = 10^{-3}$, the TT-ranks listed in Table 3 show that the resulting tensor trains are small enough to be stored. Table 4 emphasizes that in practice $r_k^{\chi} \ll r_{max,k}^{\chi}$ except for $k = 1$ where $r_{max,k}^{\chi}$ is already "small".

	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$
r_k^1	7	9	10	24	27	30
r_k^2	13	23	29	123	143	134
r_k^3	11	17	20	67	90	100
r_k^4	9	12	14	24	20	21
$r_{max,k}^1 = r_{max,k}^2 = r_{max,k}^3$	30	30^2	30^3	30^4	$6 \times 30n_t$	$6 \times n_t$
$r_{max,k}^4$	30	30^2	30^3	$30^2 n_t$	$30n_t$	n_t

Table 3: TT-ranks of the outputs of interest and theoretical maximum ranks.

9.4. Approximation error

The accuracy of the surrogate model is estimated a posteriori by measuring the discrepancy between its own outputs and the outputs of the original physical model. The estimation is conducted by comparing solutions associated with 20 000 samples of parameter set values randomly selected according to a uniform law on each discretized parameter intervals. Hence, no interpolation error is introduced in the comparison. The difference between the surrogate and the physical models is measured based on the following norms:

	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$
$r^1_{max,k}/r^1_k$	4.3	1.0×10^2	2.7×10^3	3.4×10^4	3.6×10^3	1.1×10^2
$r^2_{max,k}/r^2_k$	2.3	3.9×10^1	9.3×10^2	6.6×10^3	6.8×10^2	2.4×10^1
$r^3_{max,k}/r^3_k$	2.7	5.3×10^1	1.4×10^3	1.2×10^4	1.1×10^3	3.2×10^1
$r^4_{max,k}/r^4_k$	3.3	7.5×10^1	1.9×10^3	2.0×10^4	8.1×10^2	2.6×10^1

Table 4: Ratio between the theoretical maximum ranks and the TT-ranks of the outputs of interest.

$$\|x\|_T^2 = \int_0^T x^2 dt \quad \text{et} \quad \left\|\underset{\sim}{\boldsymbol{x}}\right\|_T^2 = \int_0^T \underset{\sim}{\boldsymbol{x}} : \underset{\sim}{\boldsymbol{x}}\, dt$$

where x and $\underset{\sim}{\boldsymbol{x}}$ are respectively scalar and tensor time-dependent function.

For the mechanical variable $\square$ (where $\square$ can stand for any one of $\underset{\sim}{\boldsymbol{\varepsilon}}$, $\underset{\sim}{\boldsymbol{\varepsilon}}_{vp}$, $\underset{\sim}{\boldsymbol{\sigma}}$ and p), $\square^{PM}$ and $\square^{TT}$ denote the output corresponding respectively to the solution of the DAEs and the surrogate model. A relative error is associated with each mechanical variable, namely:

- Total strain tensor: $e_\varepsilon = \dfrac{\left\|\underset{\sim}{\boldsymbol{\varepsilon}}^{PM} - \underset{\sim}{\boldsymbol{\varepsilon}}^{TT}\right\|_T}{\left\|\underset{\sim}{\boldsymbol{\varepsilon}}^{PM}\right\|_T}$;
- Viscoplastic strain tensor: $e_{\varepsilon_{vp}} = \dfrac{\left\|\underset{\sim}{\boldsymbol{\varepsilon}}^{PM}_{vp} - \underset{\sim}{\boldsymbol{\varepsilon}}^{TT}_{vp}\right\|_T}{\left\|\underset{\sim}{\boldsymbol{\varepsilon}}^{PM}\right\|_T}$;
- Stress tensor: $e_\sigma = \dfrac{\left\|\underset{\sim}{\boldsymbol{\sigma}}^{PM} - \underset{\sim}{\boldsymbol{\sigma}}^{TT}\right\|_T}{\left\|\underset{\sim}{\boldsymbol{\sigma}}^{PM}\right\|_T}$;
- Cumulative viscoplastic deformation: $e_p = \dfrac{\left\|p^{PM} - p^{TT}\right\|_T}{\left\|\underset{\sim}{\boldsymbol{\varepsilon}}^{PM}\right\|_T}$.

Depending on the parameter values, the viscoplastic part of the behavior may or may not be negligible as measured by the magnitudes of $\|p\|$ and $\left\|\underset{\sim}{\boldsymbol{\varepsilon}}_{vp}\right\|$ relative to $\left\|\underset{\sim}{\boldsymbol{\varepsilon}}\right\|$. Hence, in the proposed application, the focus is on comparing the norm of the approximation error for $\underset{\sim}{\boldsymbol{\varepsilon}}$, $\underset{\sim}{\boldsymbol{\varepsilon}}_{vp}$ and p with respect to the norm of $\underset{\sim}{\boldsymbol{\varepsilon}}$.

The histograms featured on Figures 5a, 5b, 5c and 5d present, for each mechanical variables, the empirical distribution of the relative error for all simulation results. The surrogate model given by the tensor-train decompositions features a level of error that is sufficiently low to carry out parametric studies such as calibration of constitutive laws where errors lower than 2% are typically tolerable.

9.5. Convergence with respect to the truncation tolerance

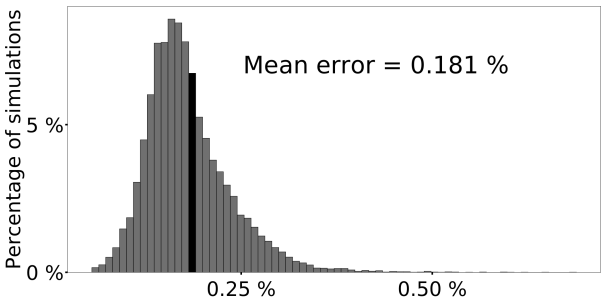
A first surrogate model is constructed from the physical model with the prescribed truncation tolerance $\epsilon_{svd} = 10^{-3}$. Then, this first surrogate model is used as an input for Algorithm 2. Running the algorithm several times with different truncation tolerances:

$$\epsilon_{svd} \in \left\{1\times10^{-3};\ 2\times10^{-3};\ 4.6\times10^{-3}; 1\times10^{-2};\ 2\times10^{-2};\ 4.6\times10^{-2};\ 1\times10^{-1}\right\}$$

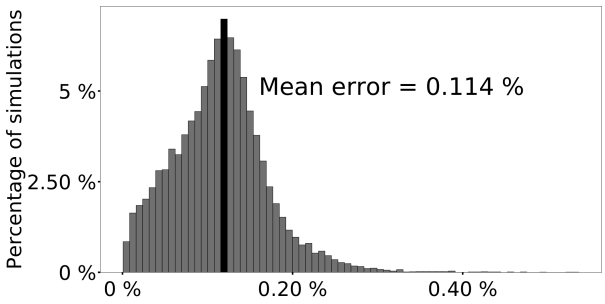
generates as many new surrogate models.

Figures 7a, 7b, 7c and 7d present the evolution of the relative error distribution (for the different mechanical variables) with respect to the truncation tolerance based on a random sample of 20 000 parameter set values chosen as in Section 9.4. Figure 6 details the graphical notations. The results empirically show for each mechanical output, the relative error decreases together with ϵ_{svd}. It is consistent with the expected behavior of the algorithm.

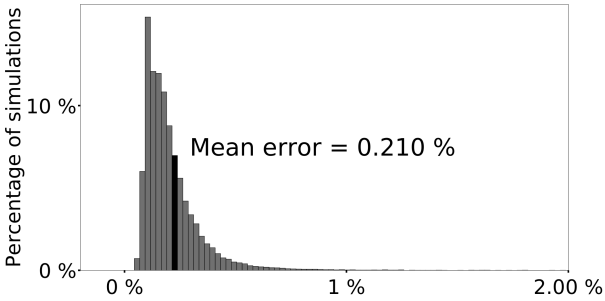
Plots in Figure 8a and 8b show the dependence of the number of stored elements and the number of calls to the physical model on ϵ_{svd}.



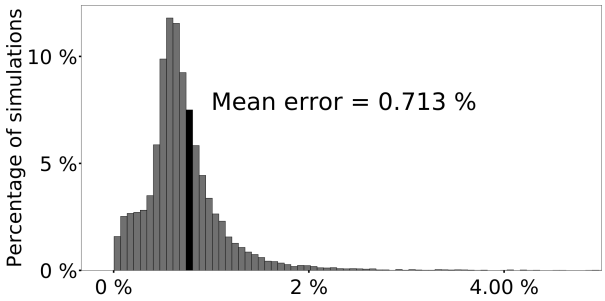
(a) Empirical distribution for e_ε. The size of the histogram bucket is 0.009%.



(b) Empirical distribution for $e_{\varepsilon_{vp}}$. The size of the histogram bucket is 0.008%.



(c) Empirical distribution for e_σ. The size of the histogram bucket is 0.024%.



(d) Empirical distribution for e_p. The size of the histogram bucket is 0.066%.

Figure 5: Empirical distribution of the errors for every mechanical variables.

9.6. Online coherence estimators

Based on the physical model, the surrogate model gives an approximation of each output of interest. However, the approximate outputs may be inconsistent with the physics in the sense that they may lead to non-zero residuals when introduced into (the discrete version of) the system (41).

A *coherence estimator* is an indicator that measures how closely the physical equations are verified by the outputs of the surrogate model. It is reasonable to expect the accuracy of the metamodel to be correlated with the coherence estimator.

Using Equation (52) let:

$$\underset{\sim}{\boldsymbol{\sigma}}^{eq,TT} = \frac{E}{1+\nu}\left(\underset{\sim}{\boldsymbol{\varepsilon}}_e^{TT} + \frac{\nu}{1-2\nu} Tr\left(\underset{\sim}{\boldsymbol{\varepsilon}}_e^{TT}\right)\underset{\approx}{\mathbb{I}}\right)$$

and define the associated coherence estimator as follows:

$$\eta_\sigma = \frac{\left\|\underset{\sim}{\boldsymbol{\sigma}}^{TT} - \underset{\sim}{\boldsymbol{\sigma}}^{eq,TT}\right\|_T}{\left\|\underset{\sim}{\boldsymbol{\sigma}}^{TT}\right\|_T} \tag{62}$$

Figure 9 displays the relation between the relative error for $\underset{\sim}{\boldsymbol{\sigma}}$ and the effectivity of the estimator η_σ / e_σ for 20 000 simulation results drawn randomly. The error increases with the final cumulative deformation, that is when the material exhibits a more intense viscoplastic behavior.

Furthermore, the plot shows a correlation between the coherence estimator and the relative error. In particular, the effectivity tends to be larger than 1 which indicates that the coherence estimator behaves like an upper bound of the relative error. Excluding a few outliers, the coherence estimator does not overestimate the relative error by more than a factor 7.

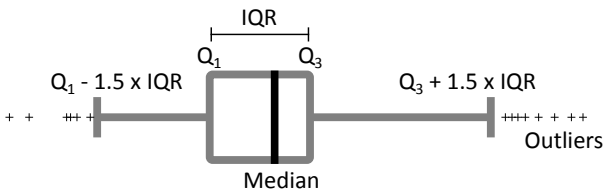
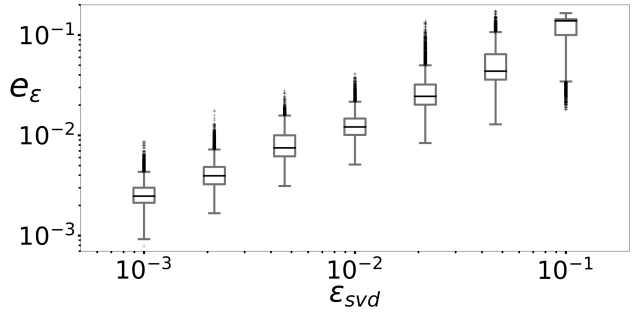
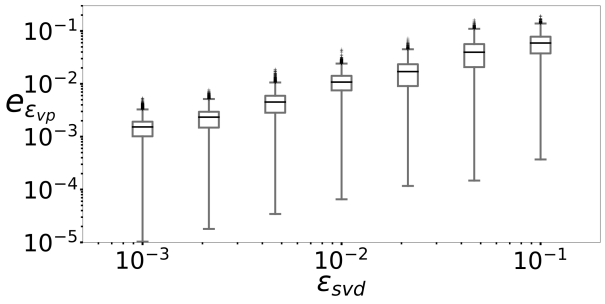


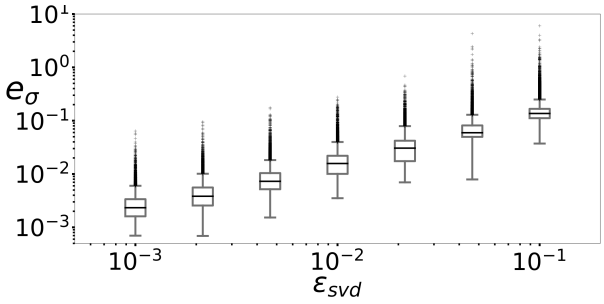
Figure 6: The left and right sides are the first and third quartiles (respectively Q_1 and Q_3). The line inside the box represents the median. The reach of the whiskers past the first and third quartiles is 1.5 times the interquartile range (IQR). The crosses represent the outliers lying beyond the whiskers.



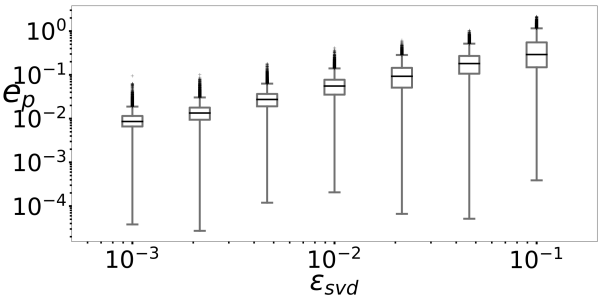
(a) Empirical distribution for e_ε



(b) Empirical distribution for $e_{\varepsilon_{vp}}$



(c) Empirical distribution for e_σ



(d) Empirical distribution for e_p

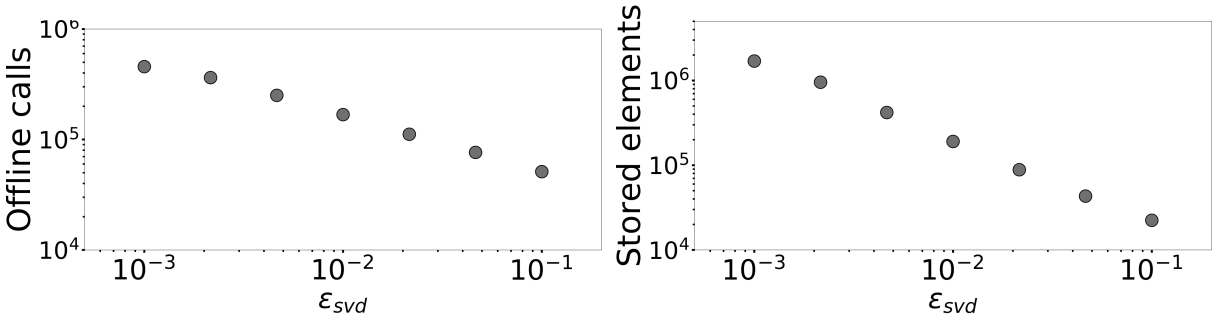
Figure 7: Empirical distribution of the relative approximation error for every mechanical variables.

Finally, the effectivity of the coherence estimator empirically converges to 1 (that is, the estimator becomes sharper) as the magnitude of the relative error increases.

This coherence estimator is very cheap to compute and only relies on outputs of the surrogate model. The results suggest that the coherence estimator could be used as an online error indicator that increases the reliability of the surrogate model at the current point when exploring in real-time the parameter domain.

10. Conclusions and perspectives

The present work assesses the performance of tensor-train representations for the approximation of numerical solutions of nonlinear DAE systems. The proposed method enables to incorporate a large number of simulation results ($\simeq$ 500 000 scalar values) to produce a metamodel that is accurate over the entire parameter domain. More specifically, numerical results show that the Multiple TT decomposition gives promising results when used as a surrogate model for an elasto-viscoplastic constitutive law. For this particular application, the surrogate model exhibits a satisfying accuracy given the moderate computational effort spent for its construction and the data storage requirements. Moreover, the observed behavior of the proposed empirical coherence estimator indicates that the latter could be exploited



(a) Dependence of the number of calls to physical model on ϵ_{svd}

(b) Dependence of the number of stored elements on ϵ_{svd}

Figure 8: Dependence of computational cost and memory storage indicators on ϵ_{svd}

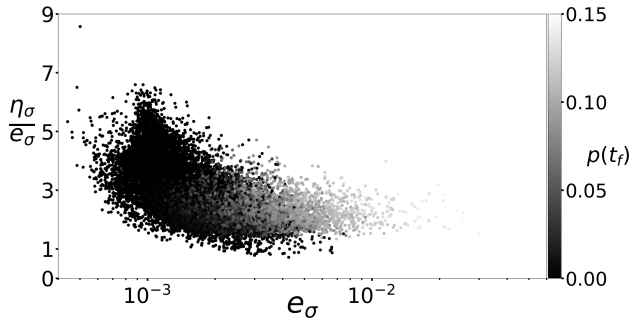


Figure 9: Effectivity of the coherence estimator η_σ (62) associated with σ. The color scale indicates the final cumulative deformation.

to assess the approximation error in real time.

The application to more complex material constitutive laws involving a larger number of parameters is under way. In addition, ongoing work concerns the actual calibration of constitutive laws with the help of the Multiple TT approach. Surrogate models have the potential to transform the way of carrying out parametric studies in material science. In particular, they may enable the design of efficient methods for sensibility estimation or uncertainty quantification of numerical models. Future work will investigate the combination of the proposed method with "usual" model order reduction techniques such as hyper-reduction [33] in order to take into account the space dimension.

Appendix A. Preliminaries

Lemma 1. *Let* $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_p \times r}$ *and* $\mathcal{B} \in \mathbb{R}^{r \times m_1 \times \cdots \times m_q}$*, then:*

$$\|\mathcal{A} \bullet \mathcal{B}\|^2 \leq \left\|\langle \mathcal{A} \rangle_p\right\|_2^2 \|\mathcal{B}\|^2$$

Proof. From the definition of matricization (See Section 2) it comes:

$$\langle \mathcal{A} \bullet \mathcal{B} \rangle_p = \langle \mathcal{A} \rangle_p \langle \mathcal{B} \rangle_1$$

Hence

$$\begin{aligned}\|\mathcal{A} \bullet \mathcal{B}\|^2 &= \left\|\langle \mathcal{A} \bullet \mathcal{B} \rangle_p\right\|^2 \\ &= \left\|\langle \mathcal{A} \rangle_p \langle \mathcal{B} \rangle_1\right\|^2 \\ &\leq \left\|\langle \mathcal{A} \rangle_p\right\|_2^2 \|\langle \mathcal{B} \rangle_1\|^2 \\ &\leq \left\|\langle \mathcal{A} \rangle_p\right\|_2^2 \|\mathcal{B}\|^2\end{aligned}$$

□

Appendix B. Proof of Proposition 1

Proof. The generalization in arbitrary dimension d of the classical linear, bilinear and trilinear interpolation yields:

$$F(\mu_1, \ldots, \mu_d) = \sum_{\eta_1, \ldots, \eta_d = \pm 1} \left[F^\Delta\left(\hat{\mu}_1^{-\eta_1}, \ldots, \hat{\mu}_d^{-\eta_d}\right) \prod_{k=1}^{d} w_{\eta_k}(\mu_k) \right] \tag{B.1}$$

where $F^\Delta\left(\hat{\mu}_1^{-\eta_1}, \ldots, \hat{\mu}_d^{-\eta_d}\right) \in \mathbb{R}$ and

$$\begin{array}{rccl} w_{\eta_k} & : & [\hat{\mu}_k^{-1}, \hat{\mu}_k^{+1}] & \to & \mathbb{R} \\ & & \mu_k & \mapsto & \dfrac{\eta_k\left(\hat{\mu}_k^{\eta_k} - \mu_k\right)}{\left(\hat{\mu}_k^{+1} - \hat{\mu}_k^{-1}\right)} \end{array}$$

is an affine function of μ_k with $\eta_k = \pm 1$.

It can be shown that the function F defined at Equation (B.1) is piecewise multilinear (that is, piecewise affine with respect to each of the variables μ_k) and coincides with F^Δ (defined by Equation (7)) on $\mathcal{D}^\Delta$.

According to the definition of μ_l^{-1} and μ_l^{+1} (14), (B.1) can be rewritten:

$$F(\mu_1, \ldots, \mu_d) = \sum_{\eta_1, \ldots, \eta_d = \pm 1} F^\Delta\left(\mu_1^{\left(i_1^{-\eta_1}\right)}, \ldots, \mu_d^{\left(i_d^{-\eta_d}\right)}\right) \prod_{k=1}^{d} w_{\eta_k}(\mu_k) \tag{B.2}$$

Based on the definition of $\mathcal{T}$ (Equation (10)), (B.2) becomes:

$$F(\mu_1, \ldots, \mu_d) = \sum_{\eta_1, \ldots, \eta_d = \pm 1} \mathcal{T}\left(i_1^{-\eta_1}, \ldots, i_d^{-\eta_d}\right) \prod_{k=1}^{d} w_{\eta_k}(\mu_k)$$

Since $\mathcal{T}$ is given in TT format (Equation (11)):

$$\begin{aligned} F(\mu_1, \ldots, \mu_d) &= \sum_{\eta_1, \ldots, \eta_d = \pm 1} \prod_{k=1}^{d} G_k\left(i_k^{-\eta_k}\right) \prod_{k=1}^{d} w_{\eta_k}(\mu_k) \\ &= \sum_{\eta_1, \ldots, \eta_d = \pm 1} \prod_{k=1}^{d} w_{\eta_k}(\mu_k) G_k\left(i_k^{-\eta_k}\right) \\ &= \prod_{k=1}^{d} \sum_{\eta_k = \pm 1} w_{\eta_k}(\mu_k) G_k\left(i_k^{-\eta_k}\right) \\ &= \prod_{k=1}^{d} L_k(\mu_k) \end{aligned}$$

□

Appendix C. Proof of Proposition 2

Proof. Following the notation of Algorithm 1, define $\mathcal{A}^{(1)} \in \mathbb{R}^{s_0 \times n_1 \times \cdots \times n_d}$ with $s_0 = 1$ such that:

$$\left(\mathcal{A}^{(1)}\right)^{\star} = \mathcal{A} \tag{C.1}$$

.

Equation (17) can be written:

$$A_1 = \left\langle \mathcal{A}^{(1)} \right\rangle_2 \tag{C.2}$$

Hence, Equations (C.2), (18), (19), (20) and the definition of the error (22) yield:

$$\left\langle \mathcal{A}^{(k)} \right\rangle_2 = H_k \left\langle \mathcal{A}^{(k+1)} \right\rangle_1 + E_k \text{ for } k = 1, \ldots, d-1 \tag{C.3}$$

With the indicial notation (C.3) reads:

$$\begin{aligned} \left\langle \mathcal{A}^{(k)} \right\rangle_2 ((\alpha_{k-1}, i_k), (i_{k+1}, \ldots, i_d)) = \sum_{\alpha_k=1}^{s_k} H_k ((\alpha_{k-1}, i_k), \alpha_k) \left\langle \mathcal{A}^{(k+1)} \right\rangle_1 (\alpha_k, (i_{k+1}, \ldots, i_d)) \\ + E_k ((\alpha_{k-1}, i_k), (i_{k+1}, \ldots, i_d)) \end{aligned} \tag{C.4}$$

Based on the definitions (16) and (23), Equation (C.4) can be written:

$$\mathcal{A}^{(k)}(\alpha_{k-1}, i_k, \ldots, i_d) = \sum_{\alpha_k=1}^{s_k} \mathcal{H}_k(\alpha_{k-1}, i_k, \alpha_k) \mathcal{A}^{(k+1)}(\alpha_k, i_{k+1}, \ldots, i_d) + \mathcal{E}_k(\alpha_{k-1}, i_k, \ldots, i_d) \tag{C.5}$$

and can be shortened with the tensor dot product as follows:

$$\mathcal{A}^{(k)} = \mathcal{H}_k \bullet \mathcal{A}^{(k+1)} + \mathcal{E}_k \quad \text{for } k = 1, \ldots, d-1 \tag{C.6}$$

Define:

$$\mathcal{H}_{(k)} = \mathcal{H}_1 \bullet \cdots \bullet \mathcal{H}_k \text{ for } k = 1, \ldots, d-1 \quad \text{and} \quad \mathcal{H}_{(0)} = \mathbb{I} \tag{C.7}$$

Given $k \in [1 : d-1]$, let $\mathcal{P}_k$ denote the property

$$\mathcal{A}^{(1)} = \mathcal{H}_{(k)} \bullet \mathcal{A}^{(k+1)} + \sum_{k'=0}^{k-1} \mathcal{H}_{(k')} \bullet \mathcal{E}_{k'+1} \tag{C.8}$$

$\mathcal{P}_k$ is proved hereafter by induction on the index k.
Base case Equation (C.6) for $k = 1$ implies:

$$\begin{aligned} \mathcal{A}^{(1)} &= \mathcal{H}_1 \bullet \mathcal{A}^{(2)} + \mathcal{E}_1 \\ &= \mathcal{H}_{(1)} \bullet \mathcal{A}^{(2)} + \mathcal{H}_{(0)} \bullet \mathcal{E}_1 \quad \text{according to (C.7)} \end{aligned} \tag{C.9}$$

Hence $\mathcal{P}_1$ holds.
Inductive step Let $k \leq d-2$ such that $\mathcal{P}_k$ is true. Then:

$$\begin{aligned} \mathcal{A}^{(1)} &= \mathcal{H}_{(k)} \bullet \mathcal{A}^{(k+1)} + \sum_{k'=0}^{k-1} \mathcal{H}_{(k')} \bullet \mathcal{E}_{k'+1} \\ &= \mathcal{H}_{(k)} \bullet \left[\mathcal{H}_{k+1} \bullet \mathcal{A}^{(k+2)} + \mathcal{E}_{k+1} \right] + \sum_{k'=0}^{k-1} \mathcal{H}_{(k')} \bullet \mathcal{E}_{k'+1} \qquad \text{according to (C.6) since } k+1 \leq d-1 \\ &= \mathcal{H}_{(k)} \bullet \mathcal{H}_{k+1} \bullet \mathcal{A}^{(k+2)} + \mathcal{H}_{(k)} \bullet \mathcal{E}_{k+1} + \sum_{k'=0}^{k-1} \mathcal{H}_{(k')} \bullet \mathcal{E}_{k'+1} \\ &= \mathcal{H}_{(k+1)} \bullet \mathcal{A}^{(k+2)} + \sum_{k'=0}^{k} \mathcal{H}_{(k')} \bullet \mathcal{E}_{k'+1} \end{aligned}$$

Hence $\mathcal{P}_{k+1}$ holds and by induction, the property (C.8) is true for all $k \in [1 : d-1]$.
Equation (C.8) for $k = d-1$ implies:

$$\mathcal{A}^{(1)} = \mathcal{H}_{(d-1)} \bullet \mathcal{A}^{(d)} + \sum_{k'=0}^{d-2} \mathcal{H}_{(k')} \bullet \mathcal{E}_{k'+1} \tag{C.10}$$

Equation (19) defines $\mathcal{A}^{(d)}$ such that Equations (16) and (21) yield:

$$\mathcal{A}^{(d)} = \mathcal{H}_d^{\star} \tag{C.11}$$

Equations (C.10) and (C.11) give:

$$\begin{aligned} \mathcal{A}^{(1)} = \quad & \mathcal{H}_{(d-1)} \bullet \mathcal{H}_d^{\star} \\ & + \mathcal{H}_{(d-2)} \bullet \mathcal{E}_{d-1} \\ & + \dots \\ & + \mathcal{H}_{(0)} \bullet \mathcal{E}_1 \end{aligned} \tag{C.12}$$

Given the definition of $\mathcal{T}$ (15):

$$\mathcal{T} = \left(\mathcal{H}_{(d-1)} \bullet \mathcal{H}_d^{\star}\right)^{\star} \tag{C.13}$$

Finally, (C.1), (C.12) and (C.13) yield:

$$\mathcal{A} - \mathcal{T} = \sum_{k=1}^{d-1} [\mathcal{H}_{(k-1)} \bullet \mathcal{E}_k]^{\star} \tag{C.14}$$

which ends the proof of Proposition 2. □

Appendix D. Proof of Proposition 3

Proof. The triangle inequality for the Frobenius norm applied to (C.14) and the invariance of the Frobenius norm under reshaping yield:

$$\|\mathcal{A} - \mathcal{T}\|^2 \leq \sum_{k=1}^{d-1} \left\|\mathcal{H}_{(k-1)} \bullet \mathcal{E}_k\right\|^2 \tag{D.1}$$

For $k = 1, \dots, d-1$, applying recursively Lemma 1 to $\left\|\mathcal{H}_{(k-1)} \bullet \mathcal{E}_k\right\|^2$ yields:

$$\left\|\mathcal{H}_{(k-1)} \bullet \mathcal{E}_k\right\|^2 \leq \|\langle \mathcal{H}_1 \rangle_2\|_2^2 \, \|\langle \mathcal{H}_2 \rangle_2\|_2^2 \dots \|\langle \mathcal{H}_{k-1} \rangle_2\|_2^2 \, \|E_k\|^2$$

Hence, according to the definition of $\mathcal{H}_k$ (16):

$$\left\|\mathcal{H}_{(k-1)} \bullet \mathcal{E}_k\right\|^2 \leq \|H_1\|_2^2 \, \|H_1\|_2^2 \dots \|H_{k-1}\|_2^2 \, \|E_k\|^2$$

Finally (D.1) becomes:

$$\|\mathcal{A} - \mathcal{T}\|^2 \leq \sum_{k=1}^{d-1} \|H_1\|_2^2 \dots \|H_{k-1}\|_2^2 \, \|E_k\|^2$$

which ends the proof of Proposition 3. □

Appendix E. Q-DEIM Algorithm

Algorithm 3: Q-DEIM [28]

Input: A matrix $U \in \mathbb{R}^{n\times r}$ with $r \leq n$ with orthogonal columns.
Output: A set $\mathcal{I} = \left\{\mathcal{I}^{(1)}, \ldots, \mathcal{I}^{(r)}\right\} \subset [1:n]$ of r row indices.
Define $A = U^T$ and compute the QR decomposition of A with the Businger-Golub column pivoting strategy [19, Algorithm 5.4.1]. It yields:

$$A\Pi = QR \quad \text{with} \quad \Pi \in \mathbb{R}^{n\times n},\ Q \in \mathbb{R}^{r\times r} \text{ and } R \in \mathbb{R}^{r\times n}$$

where Π is a permutation matrix, Q is orthogonal and R is an upper trapezoidal matrix.
Define the list of columns $\mathcal{I}_\Pi$ such that:

$$A\Pi = A(:, \mathcal{I}_\Pi)$$

Define $\mathcal{I}$ as the set of r first indices of the list $\mathcal{I}_\Pi$:

$$\mathcal{I} = \mathcal{I}_\Pi(:r)$$

Remark 10. *Algorithm 5.4.1 of [19] is meant to be applied to matrices with more rows than columns. Nevertheless, the strategy of column pivoting is the same as in the Q-DEIM algorithm.*

Appendix F. Proof of Proposition 4

Proof. Recall the Gappy POD approximation of a matrix $A \in \mathbb{R}^{n\times m}$ (See Equation (36)):

$$A = \mathbb{P}A + E$$

where $\mathbb{P} \in \mathbb{R}^{n\times n}$ is the gappy projection:

$$\begin{aligned}\mathbb{P} &= HP^T \\ &= V\left[P^TV\right]^\dagger P^T\end{aligned}$$

and $V \in \mathbb{R}^{n\times r}$ has orthogonal columns.

From the definition of H:

$$\begin{aligned}\|H\|_2 &= \left\|V\left[P^TV\right]^\dagger\right\|_2 \\ &= \left\|\left[P^TV\right]^\dagger\right\|_2 && \text{since } V \text{ has orthogonal columns} \\ &= \frac{1}{\sigma_{min}\left(P^TV\right)} && \text{since } P^TV \text{ has full column rank} \end{aligned} \tag{F.1}$$

where σ_{min} refers to the smallest singular value.

Following the notations of Algorithm 1, Equation (F.1) yields for $k = 1, \ldots, d-1$:

$$\|H_k\|_2 = \frac{1}{\sigma_{min}\left(P_k^TV_k\right)}$$

Define the projection matrix $\mathbb{P}^r = VV^T \in \mathbb{R}^{n\times n}$ then:

$$\mathbb{P}\mathbb{P}^r = \mathbb{P}^r \tag{F.2}$$

since:

$$\begin{aligned}
\mathbb{P}\mathbb{P}^r &= HP^TVV^T \\
&= V\left[P^TV\right]^\dagger P^TVV^T \\
&= VV^T && \text{according to Remark 5} \\
&= \mathbb{P}^r
\end{aligned}$$

Therefore

$$\begin{aligned}
E &= A - \mathbb{P}A \\
&= (A - \mathbb{P}^rA) - (\mathbb{P}A - \mathbb{P}^rA) \\
&= (A - \mathbb{P}^rA) - (\mathbb{P}A - \mathbb{P}\mathbb{P}^rA) \\
&= (\mathbb{I} - \mathbb{P})(A - \mathbb{P}^rA) \\
&= (\mathbb{I} - \mathbb{P})(\mathbb{I} - \mathbb{P}^r)A
\end{aligned}$$

where $\mathbb{I} \in \mathbb{R}^{n\times n}$ denotes the identity matrix.

Moreover:

$$\begin{aligned}
\|\mathbb{I} - \mathbb{P}\|_2 &= \|\mathbb{P}\|_2 && \text{assuming that } \mathbb{P} \text{ is neither 0 nor } \mathbb{I} \text{ (See [35])} \\
&= \left\|HP^T\right\|_2 \\
&\leq \|H\|_2 && \text{since } P^T \text{ has orthogonal rows (See Remark 1)} \\
&\leq \frac{1}{\sigma_{min}\left(P^TV\right)} && \text{according to (F.1)}
\end{aligned} \tag{F.3}$$

Then

$$\begin{aligned}
\|E\| &= \|(\mathbb{I} - \mathbb{P})(\mathbb{I} - \mathbb{P}^r)A\| \\
&\leq \|\mathbb{I} - \mathbb{P}\|_2 \|(\mathbb{I} - \mathbb{P}^r)A\| \\
&\leq \frac{1}{\sigma_{min}\left(P^TV\right)} \|(\mathbb{I} - \mathbb{P}^r)A\| && \text{according to (F.3)}
\end{aligned} \tag{F.4}$$

Following the notations of Algorithm 1 and using the assumption (38), it comes:

$$\|E_k\| \leq \frac{\left\|(\mathbb{I} - V_kV_k^T)A_k\right\|}{\sigma_{min}\left(P_k^TV_k\right)} \leq \frac{\nu_k}{\sigma_{min}\left(P_k^TV_k\right)}\|A_k\|$$

and based on the definition of A_k:

$$\|A_k\| = \left\|\mathcal{A}^{(k)}\right\| \leq \|\mathcal{A}\|$$

therefore

$$\|E_k\| \leq \frac{\nu_k}{\sigma_{min}\left(P_k^TV_k\right)}\|\mathcal{A}\|$$

Finally, according to (25):

$$\|\mathcal{A} - \mathcal{T}\|^2 \leq \sum_{k=1}^{d-1} \frac{\nu_k^2}{\left[\sigma_{min}\left(P_1^TV_1\right)\sigma_{min}\left(P_2^TV_2\right)\ldots\sigma_{min}\left(P_k^TV_k\right)\right]^2}\|\mathcal{A}\|^2 \tag{F.5}$$

□

Acknowledgements

Funding: This work was supported by the Association Nationale de la Recherche et de la Technologie (ANRT) [grant number CIFRE 2014/0923].

The authors gratefully acknowledge fruitful discussions with Safran Helicopter Engines (Safran Group).

References

[1] J. Ghighi, J. Cormier, E. Ostoja-Kuczynski, J. Mendez, G. Cailletaud, F. Azzouz, A microstructure sensitive approach for the prediction of the creep behaviour and life under complex loading paths, Technische Mechanik 32 (2-5) (2012) 205–220.

[2] J.-B. Le Graverend, J. Cormier, F. Gallerneau, P. Villechaise, S. Kruch, J. Mendez, A microstructure-sensitive constitutive modeling of the inelastic behavior of single crystal nickel-based superalloys at very high temperature, International Journal of Plasticity 59 (2014) 55–83.

[3] P. Ladevèze, A. Nouy, On a multiscale computational strategy with time and space homogenization for structural mechanics, Computer Methods in Applied Mechanics and Engineering 192 (28–30) (2003) 3061 – 3087, Multiscale Computational Mechanics for Materials and Structures.

[4] A. Ammar, B. Mokdad, F. Chinesta, R. Keunings, A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids, Journal of Non-Newtonian Fluid Mechanics 139 (3) (2006) 153 – 176.

[5] A. Nouy, A priori model reduction through proper generalized decomposition for solving time-dependent partial differential equations, Computer Methods in Applied Mechanics and Engineering 199 (23–24) (2010) 1603 – 1626.

[6] F. L. Hitchcock, The expression of a tensor or a polyadic as a sum of products, Journal of Mathematics and Physics 6 (1-4) (1927) 164–189.

[7] B. N. Khoromskij, Tensors-structured numerical methods in scientific computing: Survey on recent advances, Chemometrics and Intelligent Laboratory Systems 110 (1) (2012) 1–19.

[8] L. Grasedyck, D. Kressner, C. Tobler, A literature survey of low-rank tensor approximation techniques, GAMM-Mitteilungen 36 (1) (2013) 53–78.

[9] D. Bigoni, A. P. Engsig-Karup, Y. M. Marzouk, Spectral tensor-train decomposition, SIAM Journal on Scientific Computing 38 (4) (2016) A2405–A2439.

[10] 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.

[11] H. A. Kiers, Towards a standardized notation and terminology in multiway analysis, Journal of chemometrics 14 (3) (2000) 105–122.

[12] L. R. Tucker, The extension of factor analysis to three-dimensional matrices, Contributions to mathematical psychology 110119.

[13] W. Hackbusch, S. Kühn, A new scheme for the tensor representation, Journal of Fourier analysis and applications 15 (5) (2009) 706–722.

[14] I. Oseledets, E. Tyrtyshnikov, TT-cross approximation for multidimensional arrays, Linear Algebra and its Applications 432 (1) (2010) 70–88.

[15] I. V. Oseledets, Tensor-train decomposition, SIAM Journal on Scientific Computing 33 (5) (2011) 2295–2317.

[16] D. Savostyanov, I. Oseledets, Fast adaptive interpolation of multi-dimensional arrays in tensor train format, in: Multidimensional (nD) Systems (nDs), 2011 7th International Workshop on, 2011, pp. 1–8.

[17] D. V. Savostyanov, Quasioptimality of maximum-volume cross interpolation of tensors, Linear Algebra and its Applications 458 (2014) 217–244.

[18] W. Hackbusch, Tensor spaces and numerical tensor calculus, Vol. 42, Springer Science & Business Media, 2012.

[19] G. H. Golub, C. F. Van Loan, Matrix computations, 4th Edition, The Johns Hopkins University Press, 2013.

[20] L. Sirovich, Turbulence and the dynamics of coherent structures. part 1: Coherent structures, Quarterly of Applied Mathematics 45 (3) (1987) 561–571.

[21] E. Tyrtyshnikov, S. Goreinov, N. Zamarashkin, Pseudo-skeleton approximations, Tech. rep., Institute of Numerical Mathematics of the Russian Academy of Sci., Leninski Prosp. 32-A Moscow 117334, Russia (1993).

[22] M. Bebendorf, Approximation of boundary element matrices, Numerische Mathematik 86 (4) (2000) 565–589.

[23] M. W. Berry, S. A. Pulatova, G. Stewart, Algorithm 844: Computing sparse reduced-rank approximations to sparse matrices, ACM Transactions on Mathematical Software (TOMS) 31 (2) (2005) 252–269.

[24] G. Stewart, Four algorithms for the the efficient computation of truncated pivoted QR approximations to a sparse matrix, Numerische Mathematik 83 (2) (1999) 313–323.

[25] Y. Maday, N. C. Nguyen, A. T. Patera, S. H. Pau, A general multipurpose interpolation procedure: the magic points, Communications on Pure and Applied Analysis 8 (1) (2009) 383–404.

[26] M. Barrault, Y. Maday, N. C. Nguyen, A. T. Patera, An empirical interpolation method: application to efficient reduced-basis discretization of partial differential equations, Comptes-Rendus Mathématiques 339 (9) (2004) 667–672.

[27] R. Everson, L. Sirovich, Karhunen-Loève procedure for gappy data, Journal of the Optical Society of America A 12 (1995) 1657–1664.

[28] Z. Drmac, S. Gugercin, A new selection operator for the discrete empirical interpolation method—improved a priori error bound and extensions, SIAM Journal on Scientific Computing 38 (2) (2016) A631–A648.

[29] S. Chaturantabut, D. C. Sorensen, Nonlinear model reduction via discrete empirical interpolation, SIAM Journal on Scientific Computing 32 (5) (2010) 2737–2764.

[30] J. Lemaitre, J.-L. Chaboche, Mechanics of solid materials, Cambridge University Press, 1994.

[31] J. Besson, G. Cailletaud, J.-L. Chaboche, S. Forest, Non-linear mechanics of materials, 1st Edition, Vol. 167, Springer Netherlands, 2010.

[32] J. H. Halton, On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals, Numer. Math. 2 (1) (1960) 84–90.

[33] D. Ryckelynck, F. Vincent, S. Cantournet, Multidimensional a priori hyper-reduction of mechanical models involving internal variables, Computer Methods in Applied Mechanics and Engineering 225–228 (2012) 28–43.

[34] K. Carlberg, C. Farhat, J. Cortial, D. Amsallem, The GNAT method for nonlinear model reduction: Effective implementation and application to computational fluid dynamics and turbulent flows, Journal of Computational Physics 242 (2013) 623–647.
[35] D. B. Szyld, The many proofs of an identity on the norm of oblique projections, Numerical Algorithms 42 (3-4) (2006) 309–323.