Proper Generalised Decomposition for Tensor Separation and Compression

Rafel Perelló i Ribas

December 2019

Abstract

Proper Generalized Decomposition (PGD) is a method used to solve in an efficient manner high-dimensional parametric Partial Differential Equations (PDEs). This computational tool provides an explicit solution in terms of the parameters of the PDE. In this paper we present a method to obtain separated representations of such solutions based on Least Squares projection.

1 Introduction

Proper Generalised Decomposition is used as a method to reduce the Degrees of Freedom (DoF) of high dimensional problems. Suppose that the spatial domain of interest is $\Omega \subset \mathbb{R}^{n_{sd}}$ and the parametric domain of interest is $\mathcal{P} = I_1 \times I_2 \times ... \times I_{n_p} \subset \mathbb{R}^{n_p}$. The domain of the parametric PDE is $\mathcal{D} = \Omega \times \mathcal{P}$.

In order to solve the problem in the classical way a discretisation of $n_{sd} + n_p$ dimensions is required. Once the spatial domain and every parametric dimension is discretized, the number of total DoF is $n_{DoF} = n_{DoF(\Omega)} \cdot \prod_{i=1}^{n_p} n_{DoF(I_i)}$. This results in a huge number of DoF, too large for computational purposes. For this reason, the separable approximation is used.

Suppose that the solution is required to be in the following space $u \in \mathcal{V}(\mathcal{D}) = \mathcal{V}_\Omega(\Omega) \otimes \mathcal{V}_{I_1}(I_1) \otimes \mathcal{V}_{I_2}(I_2) \otimes ... \otimes \mathcal{V}_{I_{n_p}}(I_{n_p})$ where $\mathcal{V}_i$ denotes an arbitrary Hilbert space. We restrict ourselves to the study of the discretised problem so we shall assume that all Hilbert spaces are finite dimensional. Now a subset of $\mathcal{V}$ is defined as the set of rank-one functions:

$$\mathcal{V}^{R1}(\mathcal{D}) = \left\{u(x, \boldsymbol{\mu}) = u_\Omega(x) \cdot u_{I_1}(\mu_1) \cdot u_{I_2}(\mu_2) \cdot ... \cdot u_{I_{n_p}}(\mu_{n_p}) : u_\Omega \in \mathcal{V}(\Omega), u_{I_i} \in \mathcal{V}_{I_i}(I_i)\right\}, \quad (1)$$

where $\boldsymbol{\mu} = (\mu_1, \mu_2, ..., \mu_{n_p})$.

Note that $\mathcal{V}^{R1}$ is not a vector space. The PGD approximation of the solution is a sum of functions in $\mathcal{V}^{R1}$:

$$u^{PGD} \in \mathcal{V}_M^{PGD}(\mathcal{D}) = \left\{u = \sum_{i=1}^{M} u_i^{R1} \in \mathcal{V}(\mathcal{D}) : u_i^{R1} \in \mathcal{V}^{R1}(\mathcal{D})\right\}, \quad (2)$$

where $M \in \mathbb{Z}^+$ denotes the number of modes. The set $\mathcal{V}_M^{PGD}$ is able to approximate the exact solution up to the desired accuracy for large enough M. This is due to the property that states that any function in $\mathcal{V}_\Omega(\Omega) \otimes \mathcal{V}_{I_1}(I_1) \otimes \mathcal{V}_{I_2}(I_2) \otimes ... \otimes \mathcal{V}_{I_{n_p}}(I_{n_p})$ can be written as the limit of a convergent sequence $(u_m)_{m \geq 1}$ where $u_m \in \mathcal{V}_m^{PGD}(\mathcal{D})$ [1].

With this approach the number of DoF of the discretisation of $\mathcal{V}^{R1}$ is $n_{DoF}^{R1} = n_{DoF(\Omega)} + \sum_{i=1}^{n_p} n_{DoF(I_i)}$. The number of DoF in the PGD solution is $n_{DoF}^{PGD} = M \cdot n_{DoF}^{R1}$ which is orders of magnitude lower than n_{DoF}.

We have presented the capabilities of the PGD approach. We shall explain the methodology used to obtain separated solutions from the function $u \in \mathcal{V}$. Two algorithms are presented depending on how the function is given. If the function is discretised in the classical way, i.e. $u \in \mathbb{R}^{DoF(\Omega)} \otimes \mathbb{R}^{DoF(I_1)} \otimes \mathbb{R}^{DoF(I_2)} \otimes ... \otimes \mathbb{R}^{DoF(I_{n_p})}$ the method is called Tensor Separation. Alternatively, if the provided solution is already in separated format, i.e. $u \in \mathcal{V}_M^{PGD}$ for some $M \in \mathbb{Z}^+$ the method is called Tensor Compression. This latter requires much less computational effort as the number of DoF is much lower. In this case what is expected is to reduce the number of modes used for the representation of the solution.

Both methods are *a posteriori* reduction techniques as they require first to solve the parametric PDE. If the classical FEM is used to solve the parametric problem, the solution obtained is not in separable format and the Tensor Separation method can be used to obtain a separated approximation. Alternatively, if some *a priori* method is used to solve the PDE, then the solution is in separable format. However, these methods are far from being optimal in the sense that the solutions they provide can be represented with much less modes with a relatively low loss of accuracy. In this case, the Tensor Compression can be used to reduce the number of modes used to represent the solution.

Although *a priori* PGD algorithms are much more computationally efficient to solve parametric PDEs they are out of the scope of this paper. The interested reader may consult [2, 4].

2 Methodology

In this section is presented how to obtain PGD approximations of an already computed function $u \in \mathcal{V}$. This methodology is the used for Tensor Separation and Tensor Compression (remind that $\mathcal{V}_M^{PGD} \subset \mathcal{V}$). However, they differ in the computational implementation.

Let first define the Least Squares projection from $\mathcal{V}$ to $\mathcal{V}^{R1}$ as:

$$\begin{aligned} \mathbb{P} : \mathcal{V} &\to \mathcal{V}^{R1} \\ u &\to \underset{u^{R1} \in \mathcal{V}^{R1}}{\arg\min} \left\| u - u^{R1} \right\| \end{aligned} \tag{3}$$

The first mode approximation is defined as $u_1 = \mathbb{P}(u)$. As the approximation is not, in general, exact the residual of the first mode is defined as $r_1 = u - u_1$. To improve the approximation the second separated mode is computed as $u_2 = \mathbb{P}(r_1)$ and the residual is $r_2 = r_1 - u_2$. Note that $u_2 \neq 0$ in general as $\mathcal{V}^{R1}$ is not a vector space. By induction, the separation is written as:

$$\begin{aligned} r_0 &= u, \\ u_i &= \mathbb{P}(r_{i-1}), \\ r_i &= r_{i-1} - u_i. \end{aligned} \tag{4}$$

The approximated solution is then $u^{PGD} = \sum_i u_i$. Modes are added until some convergence requirements are met.

For convenience, in the PGD representation, all vectors are normalized in their corresponding dimension and each mode is scaled up by an scalar σ^m representing the norm of the m-th mode:

$$u_M^{PGD} \in \{u(x, \boldsymbol{\mu}) = \sum_{m=1}^{M} \sigma^m u_\Omega^m(x) \prod_{i=1}^{n_p} u_i^m(\mu_i) : \|u_i^m\|_{\mathcal{V}_i} = 1, u_i^m \in \mathcal{V}_i^{R1}(\mathcal{D})\}. \tag{5}$$

2.1 Tensor Separation

Now the implementation of an algorithm that computes the PGD tensor separation of a given full tensor is presented. For a more extended explanation of the procedure, the reader can remit to [3], for instance. In both methods of Tensor Separation and Compression all dimensions are treated equally (no distinction is made between spatial and parametric dimensions).

Let $\boldsymbol{F}$ be the tensor of order d representing u. The tensor is separated as:

$$\boldsymbol{F}_{PGD} = \sum_{m=1}^{M} \sigma_m \bigotimes_{j=1}^{d} \boldsymbol{f}_j^m, \tag{6}$$

where

$$\sigma_m = \prod_{j=1}^{d} \left\| \tilde{\boldsymbol{f}}_j^m \right\|, \tag{7}$$

$$\boldsymbol{f}_j^m = \frac{1}{\left\| \tilde{\boldsymbol{f}}_j^m \right\|} \tilde{\boldsymbol{f}}_j^m . \tag{8}$$

The PGD representation of $\boldsymbol{F}$ is obtained using (4). To compute each projection $\mathbb{P}(\boldsymbol{r}_{m-1})$ a nonlinear minimisation problem is computed. To find such projection, a minimisation problem is stated as the minimisation of the nonlinear functional $\mathcal{J}(\cdot)$ defined as:

$$\mathcal{J}(\tilde{\boldsymbol{f}}_1^m, ..., \tilde{\boldsymbol{f}}_d^m) = \left\| \left(\boldsymbol{F} - \sum_{\tilde{M}=1}^{m-1} \bigotimes_{j=1}^{d} \tilde{\boldsymbol{f}}_j^{\tilde{M}} \right) - \bigotimes_{j=1}^{d} \tilde{\boldsymbol{f}}_j^m \right\|^2 . \tag{9}$$

The method used to minimise $\mathcal{J}$ is an alternated directions scheme. This consists in minimise $\mathcal{J}$ as a function of only one dimensional vector and then alternate successively the dimension in which $\mathcal{J}$ is minimised until convergence is achieved. After performing some algebraic manipulations it can be shown that $\mathcal{J}(\tilde{\boldsymbol{f}}_\gamma^m)$ can be expressed as

$$\mathcal{J}(\tilde{\boldsymbol{f}}_\gamma^m) = \|\boldsymbol{F}\|^2 + \alpha(\tilde{\boldsymbol{f}}_\gamma^m, \tilde{\boldsymbol{f}}_\gamma^m)_\gamma - 2(\tilde{\boldsymbol{f}}_\gamma^m, \boldsymbol{g})_\gamma, \tag{10}$$

where

$$\alpha := \prod_{j \neq \gamma}^{d} (\tilde{\boldsymbol{f}}_j^m, \tilde{\boldsymbol{f}}_j^m)_j, \tag{11}$$

$$\boldsymbol{g} := \boldsymbol{F} ... \bigotimes_{j \neq \gamma}^{d} \tilde{\boldsymbol{f}}_j^m - \tilde{\boldsymbol{g}}, \tag{12}$$

$$\tilde{\boldsymbol{g}} = \sum_{\tilde{M}=1}^{m-1} \sigma_{\tilde{M}} \Big[\prod_{j \neq \gamma}^{d} (\boldsymbol{f}_j^{\tilde{M}}, \boldsymbol{f}_j^m)_j \Big] \boldsymbol{f}_\gamma^{\tilde{M}}. \tag{13}$$

Here $(\cdot, \cdot)_j$ denotes the inner product in the j dimension and the symbol ... indicates tensor contraction in all possible indices with respect to the defined inner product.

The one-dimensional minimisation problem is solved as

$$\tilde{\boldsymbol{f}}_\gamma^m = \frac{1}{\alpha} \boldsymbol{g} \tag{14}$$

The alternate direction scheme is repeated until convergence is achieved. Then the directional vectors are normalized as stated in (8). With this a new mode has been added to the separated solution. New modes are be added until some stopping criteria are met. One very efficient stop criterion is the relation of the norm of the last computed mode over the norm of the first mode $\frac{\sigma^M}{\sigma^1}$. When this quantity is less than a small tolerance it is assumed than the residual of the PGD approximation is small enough and convergence is achieved.

The pseudocode summarizing the method can be found in Appendix A.

2.2 Tensor Compression

Tensor Compression algorithm differs only in the type of the input data. In this case, the input data is also a tensor of the same dimensions but written in separated form:

$$\mathbf{\Phi} = \sum_{l=1}^{L} \sigma^l \cdot \boldsymbol{\phi}_1^l \otimes \boldsymbol{\phi}_2^l \otimes ... \otimes \boldsymbol{\phi}_d^l \tag{15}$$

The only changes in the algorithm are in the computation of $\boldsymbol{g}$. More precisely, in the contraction $\mathbf{\Phi}... \bigotimes_{j\neq\gamma}^{d} \tilde{\boldsymbol{f}}_j^m$. In this case

$$\mathbf{\Phi}... \bigotimes_{j\neq\gamma}^{d} \tilde{\boldsymbol{f}}_j^m = \sum_{l=1}^{L} \sigma^l \left[\prod_{j\neq\gamma}^{d} (\boldsymbol{\phi}_j^l, \boldsymbol{f}_j^m)_j \right] \boldsymbol{\phi}_\gamma^l \tag{16}$$

The algorithm to compute the compression is the same than the used for Tensor Separation.

3 Numerical example

To illustrate the methods presented an example is given. It is taken from [3]. It consists in separating a 7-dimensional tensor. The tensor is expressed as the sum of 6 rank-one modes. However, as the method is not optimal for dimensions higher than 2 it is expected that several modes are needed to recover accurately the solution. Each dimension is discretised with 10 equally spaced points along the interval $(0, 1)$. That means that the tensor has 10^7 real coefficients. The functions defining the tensor are:

$$\begin{aligned} \phi_k^j(x_k) =& a_{jk} \exp\left(\frac{-(x_k - b_{jk})^2}{c_{jk}}\right), \\ \phi_k^6(x_k) =& \sin(\pi x_k), \end{aligned}$$

for $j = 1, ..., 5$ and $k = 1, ..., 7$. Here j denotes the mode number and k the dimension number. Coefficients a, b and c are given in Appendix B.

The results of the separation of the tensor show that to obtain an approximation with accuracy of the order of 10^{-5}, 100 modes are needed (Fig. 1). This results in $100 \times 7 \times 10 = 7 \cdot 10^4$ coefficients used to store the solution.

For the sake of completeness, the obtained separated tensor has been compressed using the Tensor Compression algorithm. However, as expected, it does not provide any significant improvement of the solution as the methodology used for Tensor Compression is the same than the used in Tensor Separation.

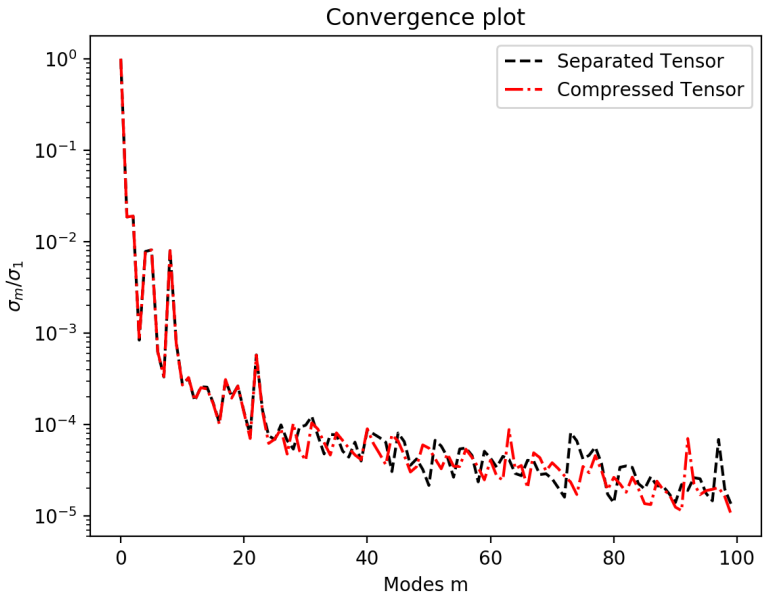

Figure 1: Normalized modal amplitudes ($\sigma_m \backslash \sigma_1$)

4 Conclusions

The PGD method based on Least Squares projection is a powerful computational tool to perform high dimensional tensor separation. It may be used to reduce the storage needed to obtain a representation of a high dimensional tensor enabling to perform complex computations with low computational resources or even to use such representations in real time applications.

Although it solves the problem of representing the solution of high dimensional parametric PDEs it does not give a solution on how to compute this solutions with low computational effort. The interested reader may consult the references provided about this subject.

5 Appendix A

Here it is presented the pseudocode summarising the algorithm of Tensor Separation and Compression.

Data: Tensor of order d to be approximated: $\mathbf{\Phi}$
Result: PGD approximation: $\boldsymbol{F}_{PGD} = \sum_{m=1}^{M} \sigma_m \boldsymbol{f}_1^m \otimes \boldsymbol{f}_2^m \otimes ... \otimes \boldsymbol{f}_d^m$
Initialize:mode counter $m = 1$, iteration counter $iMode = 1$

while $iMode < maxModes$ *and* $\sigma_m > tolModes \cdot \sigma_1$ **do**
  Initialize: Assign values to $\boldsymbol{f}_j^{old}$, for $j = 1, 2, ..., d$ such that $\left\| \boldsymbol{f}_j^{old} \right\| = 1$, $\sigma^{old} = 1$, iteration counter $itr = 1$

  while $itr < maxIter$ *and* $(error_\sigma > tolIter\_s$ *or* $error_f > tolIter\_f)$ **do**
    $\sigma^{new} = \sigma^{old}$
    $\boldsymbol{f}_j^{new} = \boldsymbol{f}_j^{old}$, for $j = 1, 2, ..., d$

    for $i_D = 1, 2, ..., d$ **do**
      $\alpha = \prod_{\substack{j=1 \\ j \neq i_D}}^{d} \left\| f_j^{new} \right\|^2$
      $\boldsymbol{g} := \mathbf{\Phi} ... \bigotimes_{j \neq i_D}^{d} \boldsymbol{f}_j^{new} - \tilde{\boldsymbol{g}}$
      $\boldsymbol{f}_{i_D}^{new} = \frac{1}{\alpha} \boldsymbol{g}$
    end

    $\sigma^{new} = \prod_{j=1}^{d} \left\| \boldsymbol{f}_j^{new} \right\|$
    $\boldsymbol{f}_j^{new} = \frac{\boldsymbol{f}_j^{new}}{\left\| \boldsymbol{f}_j^{new} \right\|}$, for $j = 1, 2, ..., d$
    $error_\sigma = \frac{|\sigma^{new} - \sigma^{old}|}{|\sigma^{new}|}$
    $error_f = \prod_{j=1}^{d} \left\| \boldsymbol{f}_j^{new} - \boldsymbol{f}_j^{m^{old}} \right\|^2$

    $\sigma^{old} = \sigma^{new}$
    $\boldsymbol{f}_j^{old} = \boldsymbol{f}_j^{new}$, for $j = 1, 2, ..., d$
  end

  Store the values σ^{new} and f_j^{new}
end

Algorithm 1: Tensor separation pseudocode

6 Appendix B

The tensor used for the numerical example is constructed using the coefficients a_{jk}, b_{jk} and c_{jk} from the matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ shown below:

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

$$\boldsymbol{B} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.6 & 0.9 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.1 & 0.75 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.8 & 0.2 & 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \end{bmatrix}$$

$$\boldsymbol{C} = \begin{bmatrix} 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0.01 & 0.02 & 0.01 & 0.02 & 0.02 & 0.02 & 0.02 \\ 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0.01 & 0.02 & 0.01 & 0.02 & 0.02 & 0.02 & 0.02 \end{bmatrix}$$

References

[1] R.A. Ryan, *Introduction to Tensor Products of Banach Spaces*, Springer (2002)

[2] A. Nouy, *A priori model reduction through Proper Generalized Decomposition for solving time-dependent partial differential equations*, Comput. Methids Appl. Mech. Engrg (2010) vol.199, pp. 1603-1626.

[3] P. Díez et al., *Algebraic PGD for tensor separation and compression: An algorithmic approach*, C. R. Mécanique (2018) vol.346, pp. 501-514.

[4] A. Sibileau, et al., *Explicit parametric solutions of lattice structures with Proper Generalized Decomposition (PGD): Applications to the design of 3D-printed architectured materials*, Computational Mechanics (2018) vol.62, pp. 871-891.