A new method for reconstruction of cross-sections using Tucker decomposition

Thi Hieu Luu, Yvon Maday, Matthieu Guillo, Pierre Guérin

A new method for reconstruction of cross-sections using Tucker decomposition

Thi Hieu LUU[a,b,*], Yvon MADAY[b,c], Matthieu GUILLO[a], Pierre GUÉRIN[a]

[a] *Département SINETICS, EDF Lab Paris-Saclay*
7, Boulevard Gaspard Monge, 91120 Palaiseau, France
[b] *Sorbonne Universités, UPMC Univ Paris 06, UMR 7598,*
Laboratoire Jacques-Louis Lions, F-75005, Paris, France
[c] *Institut Universitaire de France and*
Division of Applied Mathematics, Brown University, Providence, RI, USA

Abstract

The full representation of a d-variate function requires exponentially storage size as a function of dimension d and high computational cost. In order to reduce these complexities, function approximation methods (called *reconstruction* in our context) are proposed, such as: interpolation, approximation, etc. The traditional interpolation model like the multilinear one, has this dimensionality problem. To deal with this problem, we propose a new model based on the Tucker format - a low-rank tensor approximation method, called here *the Tucker decomposition*. The Tucker decomposition is built as a tensor product of one-dimensional spaces where their one-variate basis functions are constructed by an extension of the Karhunen-Loève decomposition into high-dimensional space. Using this technique, we can acquire, direction by direction, the most important information of the function and convert it into a small number of basis functions. Hence, the approximation for a given function needs less data than that of the multilinear model. Results of a test case on the neutron cross-section reconstruction demonstrate that the Tucker decomposition achieves a better accuracy while using less data than the multilinear interpolation.

Keywords: function approximation (reconstruction), low-rank tensor approximation, Tucker decomposition, Karhunen-Loève decomposition, cross-sections, neutronics

1. Introduction

The concept of "function approximation" is widely used in many branches of applied mathematics and computer science to apprehend quantities of interest,

*Corresponding author
Email addresses: luu@ljll.math.upmc.fr (Thi Hieu LUU), maday@ann.jussieu.fr (Yvon MADAY), matthieu.guillo@edf.fr (Matthieu GUILLO), pierre.guerin@edf.fr (Pierre GUÉRIN)

or a function depends on input parameters, such as spatial position, time evolution or any other type of input quantity. The a priori knowledge of the quantity of interest may either be "explicit", as coming out from measurements, or "implicit" as solution of a modeling equation. There is a third way that is relevant to this paper. Cross-sections that feed the neutron flux solver cannot be directly measured in an environment such as a nuclear reactor core. They neither can be computed using a single equation since there is an interaction between many physics (neutronics, hydraulic, thermic, ...). The way to solve this situation is through a simplified "experimental simulation" and we will call later on this process a "calculation scheme". Since cross-sections depend continuously on many parameters, a parametrized calculation scheme is also very useful to emulate many different "experiments" with different configurations. Therefore, one calculation point corresponds to one experiment for a given configuration. That explains why, although cross-sections are computed and not measured, we will consider them as "explicit" data acquired through "experimental simulation".

Multipurpose/universal approaches such as global or piecewise polynomial approximations are general approximation methods that are valid for a large variety of functions: depending on the hypothesis/knowledge we have on the function such as regularity in terms of existence of a certain number of derivatives, we may prefer to use global polynomial versus piecewise ones. This step of the approximation requires some know-how that is now rather well understood. Once the approximation basis set is chosen, the coefficients or components of the function we are interested in and we want to approximate in this basis set are determined in order to fit with the "input" that are explicitly available and have been acquired by some series of measurements. The stability of the mapping from the input to the coefficients or components that is an important feature for the quality of the approximation also depends on the chosen basis set. Once this is done, a second step in the approximation is the reconstruction (or evaluation) of the function we are interested in, on other points than those that have been used to construct the approximation.

All this framework involves four different concepts: i) data acquisition, ii) storage of these data, iii) reconstruction of the function we are interested in, iv) further evaluation, that all have their particular complexity and cost, that, of course depend on the number of the inputs that are used to define the function of interest. These complexity and cost suffer from what is known as the curse of dimension that leads to an exponential explosion of the complexity and cost with respect to the number of inputs.

In order to face this particular problem, different ad'hoc strategies have been proposed and leave away the notion of linear approximation in multipurpose/universal representation spaces for preferring nonlinear, adapted representations. This enters in the concept of model reduction approaches.

In this paper, we are interested in the research coming from the particular application, which is the *reconstruction of cross-sections in neutronics*.

In neutronics, cross-sections are used to represent the probability of interaction between an incident neutron and a given target nuclei ([1]). These cross-sections are inputs for the Boltzmann equation ([2]) that describes the neutron

population in the core. They depend on various parameters which characterize the local material behavior. Among the parameters stand for instance: *i) burnup, ii) fuel temperature, iii) moderator density, iv) boron concentration v) xenon level, ...* These are 5 parameters that we are interested in in this paper but there may be many more, leading to larger values of d. They are denoted by $\mathbf{x} = (x_1, \ldots, x_d)$, where here $d = 5$ and they vary in a space called *parameter-phase space*; hence cross-sections are multivariate functions of $\mathbf{x}$, $\mathbf{x} \in$ *parameter-phase space*.

There are different cross-section kinds that represent different aspects of the physics involved (fission, absorption, scattering ...). These different kinds of *reaction* are indexed by *" r"*.

The cross-sections also depend on the energy of the incident neutron, this energy is discretized through "groups" and we designate by the exponent *"g"* the incident discretized energy group. *Microscopic cross-sections* (σ) depend also on the target nuclei (or isotope), designated by *"i"*. Therefore, $\{\sigma^g_{r,i}\}$ stands for these microscopic cross-sections.

Macroscopic cross-sections (Σ) that feed the neutron flux solver are related to above quoted microscopic ones using a formula such as:

$$\Sigma^g_r = \sum_{i=1}^{I} c_i \sigma^g_{r,i} \tag{1}$$

where c_i is the concentration of isotope i.

For EDF's applications, $I \sim 50$ isotopes, $g = 2$ energy groups and $r \sim 10$ reactions, we obtain already one thousand types of microscopic cross-sections, i.e one thousand multivariate functions to approximate.

In the current core simulations, the core is described as a full three-dimensional object $\subset \mathbb{R}^3$. At each different position P in the core, we have specific thermo-hydraulic-state conditions, leading to a corresponding value of $\mathbf{x} = (x_1, \ldots, x_d)$ in the parameter-phase space. It means that the cross-sections $\{\sigma^g_{r,i}\}$, $\{\Sigma^g_r\}$ that are functions of $\mathbf{x}$ thus depend (implicitely) on the position P in the core since $\mathbf{x}$ is a function of P: $P \in \mathbb{R}^3 \mapsto \mathbf{x}(P)$.

In practice, the core is discretized into cells. Therefore, $\mathbf{x}$ depends now on the position k of a cell, $\mathbf{x} = \mathbf{x}(\text{cell}_k)$. Hence, cross-sections need to be determined cell per cell (see figure 1).

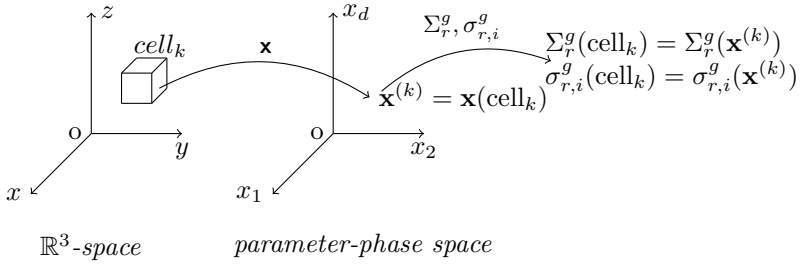


Figure 1: Dependence of cross-sections on parameters.

In the core simulation, we need accurate values for the various cross-sections at all cells (about $200,000$ cells for industrial cases and 10 times more for "reference" cases). This leads to a number of cross-section values needed for the simulation which scales in billions.

It may be time to indicate how, for a given value of $\mathbf{x} = (x_1, \ldots, x_d)$ each cross-section is computed. These are homogenized quantities representing average behaviors computed on a small homogenization cell, representing locally the core, with the full complexity of the physics (note that currently the full solution of such a model in a full three dimensional complexity of a core is far out of reach: this explains why a two stage approach has to be done).

In our applications at EDF-R&D, these cross-sections are extracted from the lattice code named APOLLO2 [3] that is developed at CEA. In this step, for a given point $\mathbf{x}$ on which cross-sections depend, *an APOLLO2 calculation is performed to provide all microscopic cross-sections* $\{\sigma^g_{r,i}\}$ *(as functions of* $\boldsymbol{x}$*) at this point.* The macroscopic cross-sections $\{\Sigma^g_r\}$ are then derived via the relation (1).

The main goal of APOLLO2 calculations is to compute the *neutron flux* ϕ. The calculation schemes used in APOLLO2 code are very complex because the resolution for many neutron equations are required. Such a calculation is expensive in time and for a given $\mathbf{x}$, it is referred to as *calculation point*. From this flux, all required cross-sections (see figure 2) can be computed very quickly .

$\underline{1^{st} \text{ step}}$:

$$\mathbf{x} = (x_1, \ldots, x_d) \xrightarrow[\text{very slow}]{APOLLO2} \phi(\mathbf{x}) \xrightarrow[\text{fast}]{} \left\{ \begin{array}{l} \Sigma^g_r(\mathbf{x}) \\ \sigma^g_{r,i}(\mathbf{x}) \end{array} \right\}_{r,i,g} \leftarrow \text{thousands of microscopic sections}$$

Figure 2: Diagram of APOLLO2 calculations used in the first step.

The determination on the fly of all billion values of cross-sections by APOLLO2 is impossible because too complex and time consuming.

Currently, in most of the core codes, like in the COCAGNE code that is developed at EDF-R&D in our team, out of few calculations performed in a first stage (with APOLLO2) and stored in files, called *neutron libraries*, approximations of all the required cross-sections $\Sigma^g_r(\mathbf{x}(cell))$ for each cell are obtained through multilinear interpolation. Note that a constraint in neutronics is that cross-sections need to be evaluated with a high accuracy (the absolute error is often measured in "pcm" (per cent mille) where 1 pcm = 10^{-5}). After this reconstruction performed by the core code COCAGNE, the flux solver of the core code computes the neutron flux $\phi(cell)$ in the next stage. This whole stage is illustrated in figure 3.

Let us explain in more details the multilinear interpolation that is currently implemented in COCAGNE to reconstruct each cross-section. A high-dimensional tensorized grid for $\mathbf{x}$ is defined, where each axis (or phase direction)

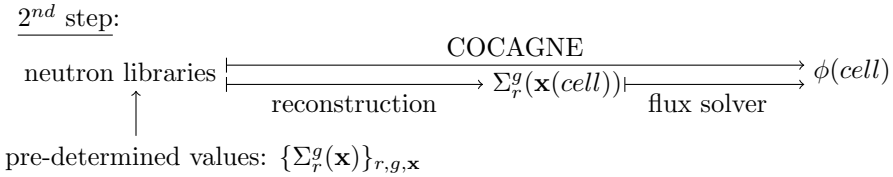


Figure 3: Diagram of cross-section reconstruction with the COCAGNE code in the second step.

is a set of parameter values. Such a grid is referred to as *multilinear grid*. The values of each set $\{\Sigma_r^g(\mathbf{x})\}_{r,g}$ and $\{\sigma_{r,i}^g(\mathbf{x})\}_{r,i,g}$ are required for each node $\mathbf{x}$ of this grid. These are obtained in the first step above mentioned as outputs of the APOLLO2 code performed on every node. The values obtained are then stored in the neutron libraries. Then, a multilinear interpolation is used in the core code to evaluate cross-sections at any point inside this grid, that involves a simple linear combination of the cross-section values at the 2^d closest nodes.

Currently, cross-sections computed by APOLLO2 depend on $d = 5$ parameters. If the multilinear grid contains n points per axis, then we need to perform n^5 calculations with APOLLO2. With thousands of microscopic and around ten macroscopic cross-sections considered in the simulation, the pre-calculated process is heavy in computation time and in storage. This leads to difficulties for the following situations:

- 5 parameters are currently used in the regular situation but incident situations depend on more parameters.
- We would like to extend the calculation domain with parameter values $\mathbf{x}$ far away from the current ones.

The first situation increases the number of dimensions, whereas the last one increases the number of values per axis: we go from n^d to $m^{d'}$ with $m > n$ and $d' > d$. The time necessary to generate the neutron libraries as well as the size of data stored increase exponentially.

To deal with this problem, there exist already some works which try to improve the cross-section reconstruction model. For instance, global polynomial interpolation on sparse grids [4], [5] or spline [6]. These have some limits due to lack of regularity of the cross-section in some directions.

In this paper, we introduce a new, non-linear and ad'hoc model based on the Tucker format [7], [8], viewed as a low-rank tensor approximation technique. Using this model, each cross-section is approached by a limited linear combination of tensor products of one-dimensional functions, referred to as *tensor directional basis functions*. The tensor directional basis functions are constructed by an extension of the Karhunen - Loève decomposition to high-dimensional spaces.

The tensor directional basis functions are not *a priori* chosen but tuned to each cross-section we consider. In particular, the number of retained tensor directional basis functions (for a given accuracy) depends on the cross-section

and also on the phase direction. Thanks to the Karhunen - Loève decomposition technique, the most important information of each cross-section is extracted, axis by axis, and represented into few tensor directional basis functions.

The coefficients in the combination of the tensor products are determined by a system of linear equations traducing interpolation equalities at some points. Note that the points are the same, whatever the cross-section, this allows us to limit the number of APOLLO2 calculations performed on these points at the stage of determining the coefficients. In order to determine the points on which this system depends, we rely on the idea presented in the empirical interpolation method [9].

With these techniques, we can reconstruct the cross-sections with a high accuracy while reducing significantly the calculation points and the storage in the neutron library.

In the light to what was presented at the beginning of the introduction about the four concepts for approximation, our approach allows us to minimize the number of data acquisition that each involves a use of the code APOLLO2. Since this data is an important part of the storage, the storage size is therefore significantly reduced. The other part of our approach is the definition of the tensor directional basis functions. The interpolation process through a tensor shape approach of each cross-section allows us to minimize the complexity of the function reconstruction with further evaluation in various different points.

The paper is organized in the following manner:

Section 2 describes the theoretical background of the Karhunen - Loève decomposition on which our approach is based in order to construct the tensor directional basis functions.

In section 3, we present our proposed methodology for the Tucker decomposition in a general case.

Section 4 is reserved for practical applications to our problem: the reconstruction of cross-sections in neutronics. The implementation procedure as well as the cost of Tucker model will be detailed.

In section 5, we show the numerical results of a test case in order to compare the Tucker model with the multilinear model using the following criteria: the number of calculation points, the storage in neutron libraries and the accuracy.

Section 6 is reserved for conclusion and discussion.

2. Theoretical background

2.1. Problem statement

From a mathematical point of view, our problem: the reconstruction of cross-sections, stands as the approximation of about one thousand multivariate functions $\{f_k(\mathbf{x})\}_k$, defined on the domain $\mathbf{\Omega}$. Here, $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbf{\Omega} \subset \mathbb{R}^d$, d is the number of parameters and $\mathbf{\Omega} = \Omega_1 \times \ldots \times \Omega_d$ ($\Omega_{i,1\leq i\leq d} \subset \mathbb{R}$).

The objective is to acquire information, store the data, and propose a reconstruction of each f_k with a [high accuracy]/[complexity] ratio. Remember that we expect an absolute accuracy to be of the order of 10^{-5} (or *pcm*).

Our approach is based on a low-rank tensor technique in order to represent each function f_k. Low-rank tensor representations (or formats) are widely used to treat the large-scale problems and there are many ways to express them, depending on the specific domain of application. We refer to [8], [10] and [11] for a general description of different formats. The Karhunen-Loève decomposition will be at the basis of our approach so we recall in the following subsection some elements of context.

2.1.1. Karhunen-Loève decomposition

The Karhunen - Loève decomposition that was introduced in statistics for continuous random processes [12], [13] is also known in various communities, as e.g. Proper Orthogonal Decomposition (POD) [14], Principal Component Analysis (PCA) [15], [16], Empirical Orthogonal Functions (EOFs) [17]. This decomposition is available for two-dimensional function spaces but has no direct extension to high-dimensional ones. The interest of this decomposition is that it provides an optimal r-rank approximation for two-variate functions, with respect to the L^2-norm.

Let $f = f(x, y)$ be a two-variate function in $L^2(\Omega_x \times \Omega_y)$. Through the Karhunen - Loève decomposition, f can be expressed as follows:

$$f(x,y) = \sum_{n=1}^{+\infty} \sqrt{\lambda_n}\varphi_n(x)\psi_n(y),\ \forall (x,y) \in \Omega_x \times \Omega_y \tag{2}$$

where $\{\varphi_n(x)\}_{n=1}^{+\infty}$ (resp. $\{\psi_n(y)\}_{n=1}^{+\infty}$) is a L^2-orthonormal basis of $L^2(\Omega_x)$ (resp. L^2-orthonormal basis of $L^2(\Omega_y)$). Furthermore, each $(\lambda_n, \varphi_n(x))$ (resp. $(\lambda_n, \psi_n(y))$) is a couple of eigenvalue-eigenfunction of a Hilbert-Schmidt operator $K_f^{(x)}$ (resp. Hilbert-Schmidt operator $K_f^{(y)}$) with the following definitions:

$$\begin{array}{rcccl} K_f^{(x)} & : & L^2(\Omega_x) & \longrightarrow & L^2(\Omega_x) \\ & & u & \longmapsto & K_f^{(x)}u(x) = \int_{\Omega_x}\int_{\Omega_y} f(x,y)f(x',y)dyu(x')dx' \\ & & & & \\ (resp.\ K_f^{(y)} & : & L^2(\Omega_y) & \longrightarrow & L^2(\Omega_y) \\ & & u & \longmapsto & K_f^{(y)}u(y) = \int_{\Omega_y}\int_{\Omega_x} f(x,y)f(x,y')dxu(y')dy' \quad) \end{array} \tag{3}$$

The $\{\lambda_n\}_{n=1}^{+\infty}$ are all positive and sorted in decreasing order: $\lambda_1 \geq \lambda_2 \geq \ldots > 0$.

For a given number $r \in \mathbb{N}^*$, if we search an approximation of f by r separate variable functions:

$$f(x,y) \approx f_r(x,y) = \sum_{n=1}^{r} a_n u_n(x) v_n(y),\ \forall (x,y) \in \Omega_x \times \Omega_y \tag{4}$$

then the Karhunen-Loève decomposition provides the best approximation (see

[18]) in the sense of minimizing the root mean squared error (RMSE):

$$e^{RMSE} = \sqrt{||f(x,y) - f_r(x,y)||^2_{L^2(\Omega)}} = \sqrt{\int_{\Omega_x}\int_{\Omega_y}(f(x,y) - f_r(x,y))^2 dxdy} \tag{5}$$

Here, the best approximation f_r^{best} is the truncation by the first r terms of the Karhunen-Loève decomposition (2):

$$f_r^{best} = \sum_{n=1}^{r} \sqrt{\lambda_n}\varphi_n(x)\psi_n(y) \tag{6}$$

In this case, we have:

$$e^{RMSE} = \sqrt{||f(x,y) - f_r^{best}(x,y)||^2_{L^2(\Omega)}} = \sqrt{\sum_{n=r+1}^{\infty} \lambda_n} \tag{7}$$

The Karhunen-Loève decomposition for two-dimensional function spaces shows that the most important information to represent a two-variate function can be found in the first eigenfunctions (associated with the first largest eigenvalues). This property is exploited by the Higher-Order Singular Value Decomposition (HOSVD) technique [19], [8], which has objectives:

1. Exploiting the best r-rank approximation for truncation.
2. Using $\{\varphi_n\}_n$ (or $\{\psi_n\}_n$) as tensor directional basis functions.

2.1.2. Implementation of the Karhunen-Loève decomposition

The tensor directional basis functions $\{\varphi_i(x)\}_i$ (resp. $\{\psi_i(y)\}_i$) of $L^2(\Omega_x)$ (resp. $L^2(\Omega_y)$) are thus eigenfunctions of the Hilbert-Schmidt operator $K_f^{(x)}$ (resp. $K_f^{(y)}$). The corresponding eigenproblem is the following integral problem:

$$\begin{aligned} &\text{Finding } (\lambda, \varphi) \in \mathbb{R} \times L^2(\Omega_x) \text{ such that:} \\ &K_f^{(x)}\varphi = \lambda\varphi \\ \Leftrightarrow \quad &\boxed{\int_{\Omega_x}\int_{\Omega_y} f(x,y)f(x',y)\varphi(x')dx'dy = \lambda\varphi(x),\ \forall x \in \Omega_x} \end{aligned} \tag{8}$$

The problem (8) is known as a Fredholm equation of the second type ([20]) which does not in general have an analytical solution. Hence, this problem is numerically solved by discretizing the domain $\Omega_x \times \Omega_y$ of f with a tensorized grid. The values of f on the discretized points (x_p, y_q) ($\{x_p\}_p$ being a suitable grid on Ω_x and $\{y_q\}_q$ a suitable grid on Ω_y), are denoted by $f_{pq} = f(x_p, y_q)$ and we set $\Omega_{N_x} = \{x_p | x_p \in \Omega_x,\ p = 1, 2, ..., N_x\}$, and $\Omega_{N_y} = \{y_q | y_q \in \Omega_y,\ q = 1, 2, ..., N_y\}$. The points x_p, y_q and (x_p, y_q) are *quadrature points*.

The discretization of the problem (8) leads to the discrete integral equation:

$$\sum_{p=0}^{N_x}\sum_{q=0}^{N_y} f(x_k, y_q)f(x_p, y_q)\Delta_{pq}\varphi(x_p) = \lambda\varphi(x_k),\ \forall x_k \in \Omega_{N_x} \tag{9}$$

where Δ_{pq} are *quadrature weights* at (x_p, y_q), depending on selected numerical integration method. In our work, $\Delta_{pq} = \delta_p \delta_q$ with δ_p (resp.δ_q) are one dimensional *quadrature weights* at x_p (resp. at y_q).

With the previous notations, the integral eigenproblem becomes a discrete matricial problem that reads

$$\sum_{p=0}^{N_x} \sum_{q=0}^{N_y} f_{kq} f_{pq} \Delta_{pq} (\overrightarrow{\varphi})_p = \lambda (\overrightarrow{\varphi})_k, \ \overrightarrow{\varphi} = (\varphi(x_p))_{x_p \in \Omega_{N_x}} \in \mathbb{R}^{N_x} \tag{10}$$

Eigenfunctions $\varphi \in L^2(\Omega_x)$ in (8) are discretely represented by eigenvectors $\overrightarrow{\varphi} \in \mathbb{R}^{N_x}$ in (10).

If we define a matrix $A = (f_{pq}) \in M_{\mathbb{R}}(N_x, N_y)$ and $\tilde{A} = (f_{pq}\Delta_{pq}) \in M_{\mathbb{R}}(N_x, N_y)$, then the eigenvalue problem (10) will be written in a matrix formula as follows:

$$A\tilde{A}^T \overrightarrow{\varphi} = \lambda \overrightarrow{\varphi} \tag{11}$$

Since the matrix $A\tilde{A}^T \in M_{\mathbb{R}}(N_x, N_x)$, we obtain in general N_x eigenvalues for the problem (11) and N_x corresponding eigenvectors: $\overrightarrow{\varphi}_1, \ldots, \overrightarrow{\varphi}_i, \ldots, \overrightarrow{\varphi}_{N_x}$ where $size(\overrightarrow{\varphi}_i) = N_x$.

3. Methodology for the Tucker decomposition based on an extension of the Karhunen-Loève decomposition

3.1. Extension of the Karhunen-Loève decomposition into high-dimensional space for the construction of one-dimensional tensor directional basis functions

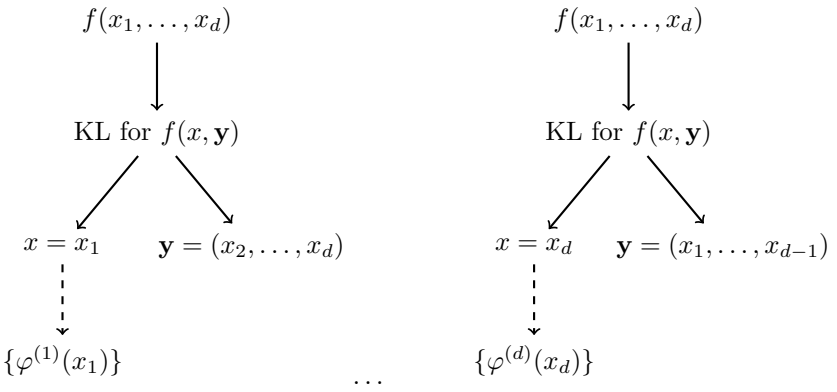
For a two-variate function, Karhunen-Loève decomposition leads, after truncation, to approximations of the behavior of the function in each variable with few ad'hoc one dimensional basis functions. As we mentioned earlier, there is no direct extension of the Karhunen-Loève decomposition for high-dimensional spaces.

Following the formalism of the Tucker decomposition and of α-rank decompositions, we propose to consider a multidimensional function as a two-variate function of x and $\mathbf{y}$ and apply the Karhunen-Loève decomposition on this expression of the function: the first direction being one of the original variable present in the multidimensional setting, i.e. $x = x_j$ for $j = 1, \ldots, d$, the other being the remaining variables all condensed into one $\mathbf{y} := (x_1, \ldots, x_{j-1}, x_{j+1}, x_d)$. This approach, known as matricization, performed independently in each direction, allows us to propose appropriate tensor directional basis functions.

The Karhunen-Loève decomposition is iteratively used for each j and leads to d integral problems, each one is written as follows:

$$\int_{\Omega_x} \int_{\mathbf{\Omega_y}} f(x, \mathbf{y}) f(x', \mathbf{y}) \varphi(x') dx' d\mathbf{y} = \lambda \varphi(x), \ \forall x \in \Omega_x \tag{12}$$

Solving these integral problems allows us to determine tensor directional basis functions $\{\varphi^{(j)}(x_j)\}$ for each direction j $(1 \leq j \leq d)$. We only keep from this decomposition, the family of eigenvectors in the x_j variable. The extension process is illustrated in figure (4).



(a) Construction of tensor directional basis functions for the direction $x = x_1$.

(b) Construction of tensor directional basis functions for the direction $x = x_d$.

Figure 4: Construction of tensor directional basis functions for all directions using an extension of the Karhunen-Loève decomposition (KL).

3.2. Numerical integration method used for integral equation

In order to solve numerically the integral equations (8) (for two-dimensional spaces) or (12) (for high dimensional spaces), as explained in subsection 2.1.2, we need a method to approximate the integrals. Even if the approach is based on a two-variate presentation of the multidimensional function, the function in the variable $\mathbf{y}$ is in dimension $d-1$. In this context we want to propose a quadrature rule adapted to our quest, which is to capture the behavior in each variable while get some understanding of the global function.

In practice, for a d-variate function $f = f(x_1, \ldots, x_d)$, we shall need d different quadrature rules based on d grids, each one used to construct tensor directional basis functions for only one direction j. Therefore, we propose adaptive discretization in order to capture well information of the concerned direction but using as less discretized points as possible. Concretely, the concerned direction j is finely discretized while coarsely for the others (say by 2^{d-1} points). Such a discretization is referred to as a *x_j-grid*, $1 \leq j \leq d$. Because we have a total of d grids in a d-dimensional space, on each axis (or direction), we realize d different discretizations but only one is fine. These discretizations are shown in figure 5.

In our case, we chose the Clenshaw-Curtis quadrature associated with *Clenshaw-Curtis points* [21], [22], [23] because the Clenshaw-Curtis points are nested from N points to $2N$ points. If we discretize the interval $[-1, 1]$ into $N+1$ Clenshaw-Curtis points then each one is defined by the following formula:

$$x_p = cos(\frac{p\pi}{N}),\ 0 \leq p \leq N \tag{13}$$

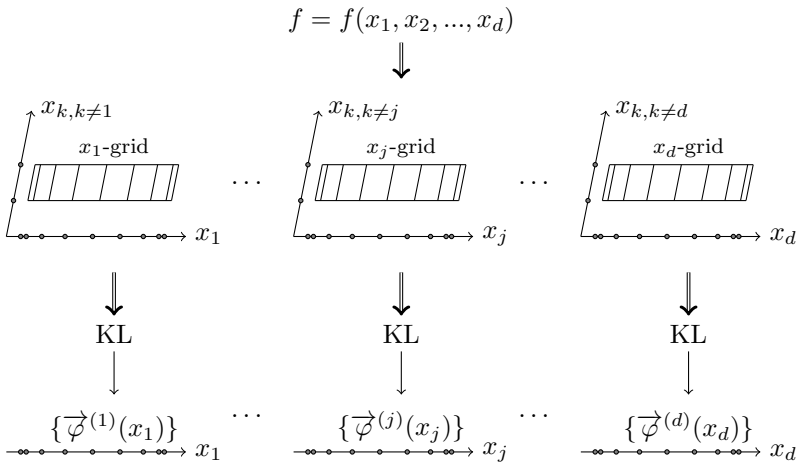


Figure 5: Adaptive discretizations (x_j*-grids*) used for an extension of the Karhunen-Loève decomposition (*KL*) to construct tensor directional basis functions, direction per direction.

3.3. Determination of tensor directional basis functions in the Tucker decomposition

After solving numerically the integral equations (12), we obtain eigenvectors $\{\overrightarrow{\varphi}^{(j)}(x_j)\}$ but not yet eigenfunctions $\{\varphi^{(j)}(x_j)\}$ for each direction j, $1 \leq j \leq d$. However, our goal is to propose an approximation of our multivariate function written as a low rank tensor, more precisely a tensor written as a Tucker decomposition:

$$f(x_1, \ldots, x_d) \approx \tilde{f}(x_1, \ldots, x_d) = \sum_{i_1=1}^{r_1} \cdots \sum_{i_d=1}^{r_d} \mathbf{a}_{i_1 \ldots i_d} \prod_{j=1}^{d} \varphi_{i_j}^{(j)}(x_j) \qquad (14)$$

that demands eigenfunctions on its left hand side. Then we need a method to reconstruct eigenfunctions from eigenvectors, direction by direction.

Such a reconstruction method applied to a direction j, should be coherent with the integration method used for this direction. In our current work, the integration method is based on a polynomial approach (with the quadrature rules). Hence, we choose the Lagrange interpolation for the reconstruction of eigenfunctions. It means that, for a direction j, the eigenfunctions $\varphi^{(j)}$ are represented as Lagrange polynomials via the quadrature points of this direction and via the eigenvector values at these points. The quadrature points mentioned here are the points in the fine discretization of the concerned direction, i.e., in our test, the Clenshaw-Curtis points (see figure 6).

3.4. Criterion for the selection of tensor directional basis functions in the Tucker decomposition

We need to select from integral problems (12) the most important eigenfunctions. We propose a criterion which takes only eigenfunctions associated with

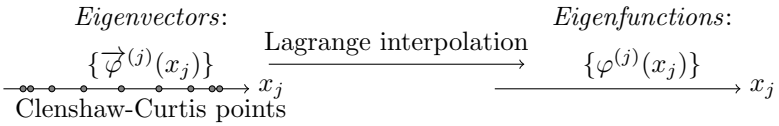


Figure 6: Current method for the reconstruction of eigenfunctions from eigenvectors.

the eigenvalues λ_i, where λ_i satisfies $\frac{\lambda_i}{\lambda_1} > \epsilon$, with $\lambda_1 \geq \lambda_2 \geq \ldots > 0$. Here, ϵ is a free parameter that can be chosen by the user. In our test, we take $\epsilon = 10^{-10}$.

3.5. Determination of coefficients in the Tucker decomposition

In the application of the Tucker decomposition (14) for $f = f(x_1, \ldots, x_d)$, we have to determine $R = \prod_{j=1}^{d} r_j$ coefficients $\mathbf{a}_i$, where $\mathbf{a}_i := a_{i_1 \ldots i_d}$ and r_j is the number of tensor directional basis functions in the direction j: $r_j = \#\{\varphi_{i_j}^{(j)}\}$. We could find the best $\mathbf{a}_i$ through projection, i.e. $\mathbf{a}_i = a_{i_1 \ldots i_d} =< f, \prod_{j=1}^{d} \varphi_{i_j}^{(j)} >$. But this method requires a high computational cost. Another approach that uses the explicit knowledge we have is based on interpolation at R well selected points $\{\mathbf{x}_t\}_{t=1}^{R}$. If these points are given with their coordinates $\mathbf{x}_t = (x_{t_1}, \ldots, x_{t_j}, \ldots, x_{t_d})$, then the R coefficients $\{\mathbf{a}_1, \ldots, \mathbf{a}_i, \ldots, \mathbf{a}_R\}$ are the solution of the following system:

$$\begin{cases} \sum_{i=1}^{R} \mathbf{a}_i \prod_{j=1}^{d} \varphi_{i_j}^{(j)}(x_{1_j}) = f(\mathbf{x}_1) \\ \ldots \\ \sum_{i=1}^{R} \mathbf{a}_i \prod_{j=1}^{d} \varphi_{i_j}^{(j)}(x_{t_j}) = f(\mathbf{x}_t) \\ \ldots \\ \sum_{i=1}^{R} \mathbf{a}_i \prod_{j=1}^{d} \varphi_{i_j}^{(j)}(x_{R_j}) = f(\mathbf{x}_R) \end{cases} \tag{15}$$

In our context, $\{\mathbf{x}_t\}_{t=1}^{R}$ (on the right hand side of (15)) will be referred to as *evaluated points of* f. The coordinates $\{x_{t_j}\}_{t=1}^{R}$ (on the left hand side of (15)) for a given direction j will be referred to as *evaluated points of tensor directional basis functions* $\{\varphi_{i_j}^{(j)}\}_{i_j=1}^{r_j}$ *in the direction* j.

4. Particular application for the reconstruction of cross-sections

4.1. Subdivision in the discretization of integral equations

In order to have a better knowledge of the functions we want to approximate, we have studied the variation of the cross-sections per parameter by varying the values of one parameter while the others are fixed at *nominal values* (the values on which the reactor operates normally). We found that the curve as a function of burnup varies a lot (especially for low burnup values) while the others are more close to linear. We show here an example in the case of the cross-section $\nu\Sigma_f^2$ (see figure (7)).

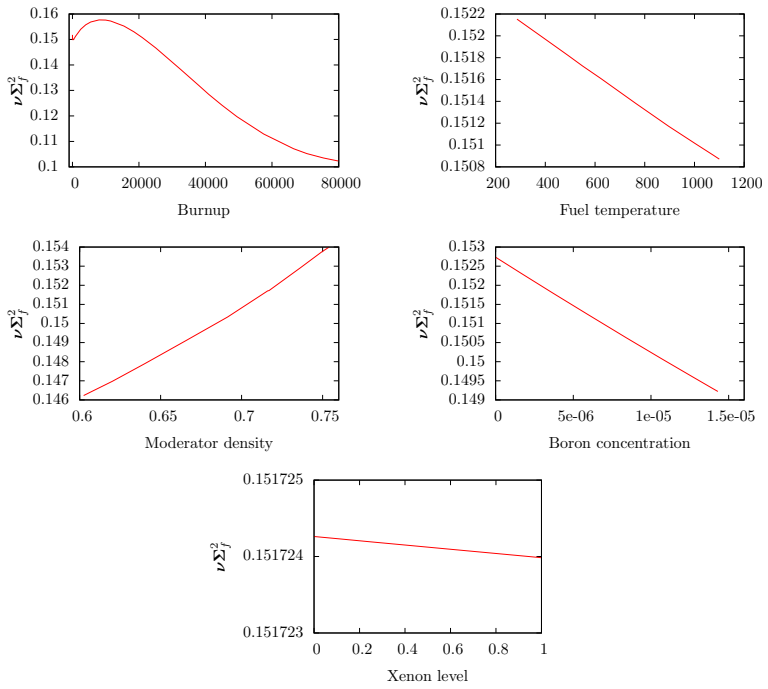


Figure 7: Cross-section $\nu\Sigma_f^2$ as function of each parameter.

We thus have used far more points on the burnup axis than on any other axes (25 points for burnup versus 5 for others). In order to capture more information of the burnup axis and avoid a high degree of polynomial approximations (used in the integration method), we subdivided this axis into three sub-intervals: $[0, 150]$, $[150, 10000]$ and $[10000, 80000]$ (see figure 8). Each sub-interval has 9 Clenshaw-Curtis points (there are two common points: 150 and 10000 for 3 intervals).

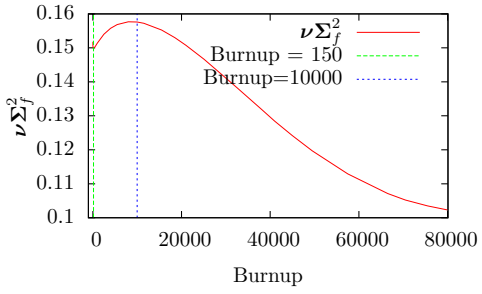


Figure 8: Subdivision of the burnup axis.

4.2. Selection of the evaluated points by using recursively a EIM

4.2.1. Evaluated points on the right hand side of the system (15)

On the right hand side of (15), we need the values of each cross-section f at R evaluated points $\{\mathbf{x}_t\}_{t=1}^R$. Already, the APOLLO2 calculations (performed in order to determine the cross-section values on x_j-grid, $1 \leq j \leq d$) can be re-used for interpolating f at those points. As explained in section 1, APOLLO2 calculations are expensive in computation time, we therefore want to reduce as much as possible these calculations at this stage.

In our case, the number of retained tensor directional basis functions in the Tucker decomposition (14) depends on the cross-section Σ_k with $1 \leq k \leq K$. Thus the selected points for interpolation in (15) depend on the considered cross-section.

Since an APOLLO2 calculation provides values of all cross-sections at a given point, we can determine a set of points, denoted by $\{\mathbf{x}_t(\bigcup_k \Sigma_k)\}_{t\geq 1}$, such that this set is usable for evaluated points of every cross-section, i.e.: $\{\mathbf{x}_t(\Sigma_k)\}_{t=1}^{R(\Sigma_k)}$ of each Σ_k can be extracted from $\{\mathbf{x}_t(\bigcup_k \Sigma_k)\}_{t\geq 1}$. Such a set $\{\mathbf{x}_t(\bigcup_k \Sigma_k)\}_{t\geq 1}$ is optimal for the number of APOLLO2 calculations if the cross-sections can use a maximal number of common evaluated points, i.e. $\#(\cap_{\Sigma_k}\{\mathbf{x}_t(\Sigma_k)\}_{t=1}^{R(\Sigma_k)})$ should be maximal.

4.2.2. Proposed constitution of evaluated points

For each cross-section, we need a total of $R = \prod_{j=1}^d r_j$ evaluated points $\{\mathbf{x}_t\}_{t=1}^R$ on the right hand side of (15), where r_j is the number of tensor directional basis functions in the direction j. It is natural to use a set of tensorized points. Therefore, we can choose for each direction j ($1 \leq j \leq d$) a set of only r_j evaluated points $\{x_{t_j}^{(j)}\}_{t_j=1}^{r_j}$ ($1 \leq j \leq d$) to constitute the $\prod_{j=1}^d r_j$ points of $\{\mathbf{x}_t\}_{t=1}^R$:

$$\{\mathbf{x}_t\}_{t=1}^R = \times_{j=1}^d \{x_{t_j}^{(j)}\}_{t_j=1}^{r_j} \tag{16}$$

Using this construction, the set $\{x_{t_j}\}_{t=1}^R$ becomes $\{x_{t_j}^{(j)}\}_{t_j=1}^{r_j}$ with $r_j < R$.

4.2.3. Problem of the evaluated points choice

For sake of convenience, because this allows us to use on the left hand side of the system (15) the values that were used to build the tensor directional basis functions $\{\varphi_{i_j}^{(j)}\}_{i_j=1}^{r_j}$, we propose to choose the evaluated points $\{x_{t_j}^{(j)}\}_{t_j=1}^{r_j}$ among the fine discretization $\{x_1^{(j)}, \ldots, x_{N_j}^{(j)}\}$ of the direction j. This choice is nontrivial because $r_j < N_j$, leads to a total of $C_{N_j}^{r_j}$ choices for $\{x_{t_j}^{(j)}\}_{t_j=1}^{r_j}$ and $\prod_{j=1}^d C_{N_j}^{r_j}$ choices to constitute $\{\mathbf{x}_t\}_{t=1}^R$ by (16). To illustrate this, our cross-sections depend on 5 parameters and if we choose $r_j = 4$, $N_j = 10$ ($1 \leq j \leq 5$), thus we already have $C_{10}^4 = 210$ choices for evaluated points of one direction j and 210^5 choices for evaluated points $\mathbf{x} = (x_1, \ldots, x_d)$.

When many cross-sections are studied at the same time, we have to determine a choice such that:

- The cross-sections use as many common evaluated points $\mathbf{x} = (x_1, \ldots, x_d)$ as possible to reduce the number of APOLLO2 calculations. This is equivalent to find as many as possible the common evaluated points $\{x_j\}$ for each direction j if we use the constitution proposed by (16).
- The accuracy of the Tucker decomposition for every cross-section must be ensured. (Because each choice leads to a change of coefficient values $\mathbf{a}$ on which the accuracy of the Tucker decomposition depends).

Remark: The choice of evaluated points $\{\mathbf{x}_t\}_{t=1}^{R}$ to solve the system (15) does not change the storage cost of the Tucker decomposition. This changes only the values of the coefficients $\mathbf{a}$ determined by (15) which are used in the Tucker decomposition (14). Since the accuracy of this decomposition is very sensitive to these coefficients, we must determine accurate coefficients by choosing the pertinent evaluated points $\{\mathbf{x}_t\}_{t=1}^{R}$ and $\{x_{t_j}^{(j)}\}_{t_j=1}^{r_j}$.

4.2.4. Proposed choice based recursively on the empirical interpolation method (EIM)

We propose to use the empirical interpolation method (EIM), described in[24], to deal with our problem of constructing evaluated points $\{x_{t_j}^{(j)}\}_{t_j=1}^{r_j}$, $1 \leq j \leq d$, and $\{\mathbf{x}_t\}_{t=1}^{R}$. This method is based on a greedy algorithm to construct interpolation points. Let us present it in a general framework: Let ω be a domain of $\mathbb{R}^n$: $\omega \subset \mathbb{R}^n$ ($1 \leq n \in \mathbb{N}$ in our applications, n will be 1 or d). $\mathcal{G}$ is a group of K functions defined on ω: $\mathcal{G} = \{\varphi : \omega \to \mathbb{R} |\, \varphi \in L^\infty(\omega)\}$ and $\#\mathcal{G} = K$. The main idea of the EIM is to determine simultaneously P interpolation points $\mathcal{X} = \{x_1, \ldots, x_P\} \subset \omega$ and the associated interpolation operators $\mathcal{I}_P$, such that for any function $\varphi \in \mathcal{G}$, $\mathcal{I}_P(\varphi)$ is an interpolation of φ with the interpolation error satisfying: $||\varphi - \mathcal{I}_P(\varphi)||_{L^\infty(\omega)} \to 0$ rapidly as $P \to \infty$ ([24]).

The EIM can be described as follows:

- For $p = 1$

$$u_1 = argmax_{\varphi \in \mathcal{G}} ||\varphi||_{L^\infty(\omega)}$$
$$x_1 = argmax_{x \in \omega} |u_1(x)|$$

(i.e. $|u_1(x_1)| = \max_{x \in \omega, \varphi \in \mathcal{G}} |\varphi(x)|$)

  - Set $\mathcal{X}_1 = \{x_1\}$
  - Establish the interpolation operator $\mathcal{I}_1$:

$$\forall \varphi \in \mathcal{G} : \mathcal{I}_1[\varphi] = \frac{\varphi(x_1)}{u_1(x_1)} u_1(.)$$

- For $p > 1$

$$u_{p+1} = argmax_{\varphi \in \mathcal{G}} ||\varphi - \mathcal{I}_p[\varphi]||_{L^\infty(\omega)}$$
$$x_{p+1} = argmax_{x \in \omega} |u_{p+1}(x) - \mathcal{I}_p[u_{p+1}](x)|$$

(i.e. $|u_{p+1}(x_{p+1})| = \max_{x \in \omega, \varphi \in \mathcal{G}} |\varphi(x) - \mathcal{I}_p[\varphi(x)]|$)

– Set $\mathcal{X}_{p+1} = \mathcal{X}_p \bigcup \{x_{p+1}\}$

– Build the interpolation operator $\mathcal{I}_{p+1}$:

$\forall \varphi \in \mathcal{G} : \mathcal{I}_{p+1}[\varphi] =$ Lagrange interpolation via the $p+1$ points of $\mathcal{X}_{p+1}$

- Continue until the required value, i.e. $p = P$.

The EIM is well suited to the problem of finding the evaluated points $\{x_{t_j}^{(j)}\}_{t_j=1}^{r_j}$ in each direction j. Indeed, this algorithm allows us to determine a set of points for a group of functions, where the cardinal of these points is a given number and on which, each function in the group is well represented, well evaluated.

In our case, assume that all studied cross-sections are $\{\Sigma_k\}_{1\leq k\leq K}$ (Σ_k depends on d parameters). For a cross-section Σ_k and for a concerned direction j, we need to determine $r_j(\Sigma_k)$ evaluated points for the tensor directional basis functions $\{\varphi_{i_j}^{(j)}(\Sigma_k)\}_{i_j=1}^{r_j(\Sigma_k)}$ of this cross-section. Therefore, the *EIM applied to the cross-section Σ_k and the direction j for finding evaluated points* $\mathcal{X}(\Sigma_k) = \{x_{i_j}^{(j)}(\Sigma_k)\}_{i_j=1}^{r_j(\Sigma_k)}$ is performed with:

- $\omega = \{\text{all evaluated points } x_j \text{ of the direction } j\} = \{x_1^{(j)}, \ldots, x_{N_j}^{(j)}\} \subset \mathbb{R}$
- $P = r_j(\Sigma_k) < N_j^{(j)}$
- $\mathcal{G} = \{\varphi_{i_j}^{(j)}(\Sigma_k)\}_{i_j=1}^{r_j(\Sigma_k)}$

The evaluated points $\{\mathbf{x}_t(\Sigma_k)\}_{t=1}^{R(\Sigma_k)}$ are then constituted as a tensor product of evaluated points $\{x_{i_j}^{(j)}(\Sigma_k)\}_{i_j=1}^{r_j(\Sigma_k)}$ using (16).

Such a determination is well-suited for each cross-section but is not optimal for the number of APOLLO2 calculations. In fact, the APOLLO2 calculations are performed on every evaluated point $\mathbf{x}$ (used on the right hand side of (15)). Since we have a total of $\{\Sigma_k\}_{1\leq k\leq K}$ cross-sections, so the total evaluated points used by all cross-sections is $\bigcup_{\Sigma_k, 1\leq k\leq K}\{\mathbf{x}_t(\Sigma_k)\}_{t=1}^{R(\Sigma_k)}$, with $\{\mathbf{x}_t(\Sigma_k)\}_{t=1}^{R(\Sigma_k)} \neq \{\mathbf{x}_t(\Sigma_l)\}_{t=1}^{R(\Sigma_l)}$ if $k \neq l$, in general. This leads to a significant number of APOLLO2 calculations for $\bigcup_{\Sigma_k, 1\leq k\leq K}\{\mathbf{x}_t(\Sigma_k)\}_{t=1}^{R(\Sigma_k)}$. As mentioned in section 4.2.3, a solution is finding as many as possible the common evaluated points between the sets $\{x_{i_j}^{(j)}(\Sigma_k)\}_{i_j=1}^{r_j(\Sigma_k)}$, for each direction j.

We denote by $\{x_j(\bigcup_k \Sigma_k)\}$ the set of common evaluated points for the direction j. The cardinal of this set must be as small as possible to have the maximal common evaluated points between the cross-sections. Since $\{x_j(\bigcup_k \Sigma_k)\}$ are used for tensor directional basis functions in $\bigcup_{\Sigma_k, 1\leq k\leq K}\{\varphi_{i_j}^{(j)}(\Sigma_k)\}_{i_j=1}^{r_j(\Sigma_k)}$, we propose to take a set $\{x_j(\bigcup_k \Sigma_k)\}$ such that $\#(\{x_j(\bigcup_k \Sigma_k)\}) = P_j$ where P_j is equal to the highest number $r_j(\Sigma_k)$, i.e. :

$$P_j := \#\{x_j(\bigcup_k \Sigma_k)\} = r_j^{max} = \max_{\Sigma_k, 1\leq k\leq K} r_j(\Sigma_k)$$

This means that all points of $\{x_j(\bigcup_k \Sigma_k)\}$ are also the evaluated points of any cross-section. After determining $\{x_j(\bigcup_k \Sigma_k)\}$, all cross-sections Σ_k can select their evaluated points $\{x_{i_j}^{(j)}(\Sigma_k)\}_{i_j=1}^{r_j(\Sigma_k)}$ with $\#\{x_{i_j}^{(j)}(\Sigma_k)\}_{i_j=1}^{r_j(\Sigma_k)} = r_j(\Sigma_k)$ from r_j^{max} points of $\{x_j(\bigcup_k \Sigma_k)\}$. *This selection can use again the EIM.*

In our case, we propose to use a discrete version of the EIM where $\{\varphi_{i_j}^{(j)}(\Sigma_k)\}_{i_j=1}^{r_j(\Sigma_k)}$ are replaced by $\{\overrightarrow{\varphi}_{i_j}^{(j)}(\Sigma_k)\}_{i_j=1}^{r_j(\Sigma_k)}$.

Assuming that all tensor directional basis functions $\{\varphi_{i_j}^{(j)}(\Sigma_k)\}_{i_j=1}^{r_j(\Sigma_k)}$ are determined in the direction j, *our EIM version applied to a direction j for finding* $\mathcal{X}(\bigcup_k \Sigma_k) = \{x_j(\bigcup_k \Sigma_k)\}$ is described as follows:

- Collect all tensor directional basis functions of the direction j into a group $\mathcal{G}_j$:
$$\mathcal{G}_j = \bigcup_{\Sigma_k,\, 1\leq k\leq K} \{\varphi_{i_j}^{(j)}(\Sigma_k)\}_{i_j=1}^{r_j(\Sigma_k)}$$
- Find the max number of evaluated points $\{x_{i_j}^{(j)}(\Sigma_k)\}_{i_j=1}^{r_j(\Sigma_k)}$ over all cross-sections:
$$r_j^{max} = \max_{\Sigma_k} r_j(\Sigma_k) \tag{17}$$
- Apply the EIM (see above) for each direction j with:
$\mathcal{G} \equiv \mathcal{G}_j$, $P \equiv P_j = r_j^{max}$ and $\Omega \equiv \{N_j$ points in the fine discretization for $j\}$

With this strategy, the number of APOLLO2 calculations used for evaluated points $\mathbf{x}$ (on the left hand side of (15)) of all studied cross-sections ($\{\Sigma_k\}_{1\leq k\leq K}$) are determined by:

$$\prod_{j=1}^{d} r_j^{max}, \text{ where } r_j^{max} \text{ is defined in (17)} \tag{18}$$

After this stage, performed offline, for each direction j, we have all evaluated points $\mathcal{X}_{r_j^{max}} = \{x_j(\bigcup_k \Sigma_k)\}$ on which each cross-section $\Sigma_{k,k\geq 1}$ can select its proper evaluated points for finding $\mathcal{X}(\Sigma_k) = \{x_{i_j}^{(j)}(\Sigma_k)\}_{i_j=1}^{r_j(\Sigma_k)}$ again from the EIM:

- $\mathcal{G} \equiv \{\varphi_1^{(j)}(\Sigma_k), \ldots, \varphi_{r_j}^{(j)}(\Sigma_k)\}$
- $P \equiv r_j(\Sigma_k)$
- $\Omega \equiv \mathcal{X}_{r_j^{max}} = \{x_j(\bigcup_k \Sigma_k)\}$

4.3. Summary of the implementation procedure

We have introduced in the previous sections our proper Tucker decomposition and the procedures to determine the components of this decomposition. Let us now summarize the principal steps when applied to a multivariate function $f = f(x_1, \ldots, x_d)$, where $x_j \in \Omega_j \subset \mathbb{R}$, $j = 1, \ldots, d$:

1. *Find the one-dimensional tensor directional basis functions for all directions j, $1 \leq j \leq d$:*
   - Domain discretization leads to d grids (x_j-grids):
     - Fine discretization of Ω_j into N_j points.
     - Coarse discretization for the other directions with $N_{k(j)}$ points, $k \neq j$.
   - Use of the extension of the Karhunen-Loève decomposition to construct tensor directional basis functions, direction per direction:
     - Establish the integral problems in high dimensional space like equation (12).
     - Solve numerically the integral problems by solving the eigenvalue problems similar to the equation (10).
     - Use the criterion described in section 3.4 to select the r_j dominant eigenvectors $\varphi_{i_j}^{(j)}$, leads to $i_j = 1, \ldots, r_j$ in the concerned direction j.
     - Construct the tensor directional basis functions for the direction j by interpolating the selected eigenvectors in a polynomial basis.
2. *Determination of the coefficients $\boldsymbol{a}$ in the Tucker decomposition*:

$$f(x_1, \ldots, x_d) \approx \tilde{f} = \sum_{i_1=1}^{r_1} \cdots \sum_{i_d=1}^{r_d} \mathbf{a}_{i_1 \ldots i_d} \prod_{j=1}^{d} \varphi_{i_j}^{(j)}(x_j) \tag{19}$$

   - For a direction j: choose r_j points $\{x_{i_j}^j\}_{i_j=1}^{r_j}$ among r_j^{max} evaluated points of $\mathcal{X}_{r_j^{max}}$ with the EIM (section 4.2.4).
   - Constitute $r_1 \ldots r_d = \prod_{j=1}^{d} r_j$ evaluated points $\mathbf{x} = (x_1, \ldots, x_d)$ by a tensor product of $\{x_{i_j}^{(j)}\}_{i_j=1}^{r_j}$ (by (16)).
   - Find the coefficients $\mathbf{a}$ by solving the linear system (15).
3. *Evaluation of f at any point $\boldsymbol{x} = (x_1, \ldots, x_d)$:*
   - Evaluate all $\varphi_{i_j}^{(j)}$ at x_j with $i = 1, \ldots, r_j$, $j = 1, \ldots, d$.
   - Evaluate $f(\mathbf{x})$ using the Tucker decomposition (19).

4.4. Cost of the multilinear interpolation and the Tucker decomposition

4.4.1. Multilinear interpolation process and the number of APOLLO2 calculations, the storage size of cross-sections

The multilinear interpolation process can be illustrated by figure 9.

We see that the APOLLO2 calculations are performed on all nodes $\mathbf{x}$ of the multilinear grid and then the cross-section values on these nodes are directly stored in the neutron library.

$$\{\text{node}_{\mathbf{x}}\} \in \begin{matrix}\text{multilinear}\\ \text{grid}\end{matrix} \xrightarrow[\text{APOLLO2}]{} \{\Sigma_r^g(node_{\mathbf{x}})\} \quad \xrightarrow[\text{Store}]{} \text{Neutron library}$$

Figure 9: Multilinear interpolation process.

If the multilinear grid has N_j points on the axis j, $(1 \leq j \leq d)$, then the number of APOLLO2 calculations is equal to:

$$\prod_{j=1}^{d} N_j \tag{20}$$

and the *storage is the same* for all cross-sections:

$$\sum_{j=1}^{d} N_j + \prod_{j=1}^{d} N_j \tag{21}$$

4.4.2. Tucker decomposition process and the number of APOLLO2 calculations, the storage of cross-sections

Tucker decomposition process

The Tucker decomposition process can be illustrated by figure 10.

$$\{\text{node}_{\mathbf{x}}\} \in \left\{\begin{array}{l} x_1\text{-grid} \\ \dots \\ x_d\text{-grid} \end{array}\right\} \xrightarrow{\text{APOLLO2}} \{\Sigma_r^g(node_{\mathbf{x}})\} \xrightarrow{\text{KL}} \begin{array}{l} \{\overrightarrow{\varphi}^{(1)}(\Sigma_r^g)\} \\ \dots \\ \{\overrightarrow{\varphi}^{(d)}(\Sigma_r^g)\} \end{array} \xrightarrow[(15)]{\text{System}} \{\mathbf{a}(\Sigma_r^g)\}$$

$$\downarrow \text{Store}$$

$$\text{Neutron library}$$

Figure 10: Tucker decomposition process (here KL: Karhunen-Loève decomposition).

We see that the APOLLO2 calculations are performed on the nodes $\mathbf{x}$ of the x_j-grids, but we do not store the cross-section values on these nodes like in the case of the multilinear model. Instead of that, we store the values of the eigenvectors $\{\overrightarrow{\varphi}^{(j)}(\Sigma_r^g)\}_{j=1}^{d}$ and the coefficients $\mathbf{a}$.

Number of APOLLO2 calculations in the Tucker decomposition

In the Tucker model, the total number of APOLLO2 calculations used *for all cross-sections* $\{\Sigma_r^g\}_{r,g}$ is calculated as follows:

$$\sum_{j=1}^{d}(N_j \prod_{k(j)=1,k(j)\neq j}^{d} N_{k(j)}) + \prod_{j=1}^{d} r_j^{max} \tag{22}$$

Where:

- N_j: the number of points in the fine discretization of the direction j.

- $N_j \prod_{k(j)=1,k(j)\neq j}^{d} N_{k(j)}$: number of nodes in the x_j-grid on which APOLLO2 calculations are performed (ref. figure 5).
- r_j^{max}: defined by (17)
- $\prod_{j=1}^{d} r_j^{max}$: the number of evaluated points $\{\mathbf{x}_t\}_{t=1}^{R}$ used by all cross-sections. Here, the APOLLO2 calculations are used on these points to compute cross-section values on the right hand side of each system like (15), to get the coefficients $\mathbf{a}$.

In our test, we chose $N_{k(j)} = 2$ so the first part of (22) is equal to $\sum_{j=1}^{d} N_j 2^{d-1}$.

Storage in the Tucker decomposition

The storage size *for one cross-section* $\Sigma_{k,k\geq 1}$ is the number of floating points stored. It depends on cross-sections and is given by:

$$\sum_{j=1}^{d} N_j + \sum_{j=1}^{d} r_j(\Sigma_r^g) * N_j + \prod_{j=1}^{d} r_j(\Sigma_r^g) \tag{23}$$

Where:

- N_j: the number of points in the fine discretization of the direction j.
- $r_j(\Sigma_r^g) * N_j$: the values of $r_j(\Sigma_r^g)$ eigenvectors $\overrightarrow{\varphi}^{(j)}$ for the direction j. Here, N_j is also the size of $\overrightarrow{\varphi}^{(j)}$.
- $\prod_{j=1}^{d} r_j(\Sigma_r^g)$: the number of coefficients $\mathbf{a}$ for the cross-section Σ_r^g.

5. Numerical results

5.1. Description of the test case

We present here a numerical test case using the Tucker decomposition and the multilinear model to reconstruct the cross-sections. Some cross-sections are considered in our work like: the `macro totale` - Σ_t^g, the `macro absorption` - Σ_a^g, the `macro fission` - Σ_f^g and the `macro nu*fission` - $\nu\Sigma_f^g$. A particular case, the `macro scattering` depends on 3 indexes: g (departure energy group) , g' (arrival energy group) and o (anisotropy order), denoted by $\Sigma_{so}^{g\to g'}$. In our test case, $o \in \{0,1\}$, $g \in \{1,2\}$, $g' \in \{1,2\}$ and the cross-sections depend on 5 parameters: *burnup, fuel temperature, moderator density, boron concentration* and *xenon level*. These parameters vary in the intervals given in table 1, around their nominal values.

5.2. Discretization of the parameter space

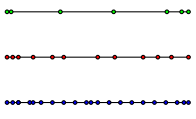
In order to compare the Tucker decomposition with the multilinear interpolation method already implemented in the COCAGNE code, we need a reference sample of points included in a grid named *reference grid*. In this grid, we tried to select the points such that they are different from the multilinear grid and

Parameter name	min value	max value
Burnup, MWd/t	0.0	80000.0
Fuel temperature,$\mathring{C}$	286.0	1100.0
Moderator density, g/cm^3	0.602	0.753
Boron concentration, ppm	0.0	1800.0
Xenon level, %	0.0	1.0

Table 1: Parameters and their intervals.

from the x_j-grid, $1 \leq j \leq d$. The reason is that if a point belongs to the multilinear grid or the tensorized grid (16) then cross-section values are exact at this point (no interpolation) with the corresponding model. In our test case, the *multilinear grid* has a large number of points in the burnup direction (33 points) while the others have 2 or 3. The Tucker model contains 5 x_j-grids corresponding to 5 directions. Each one has 5 Clenshaw-Curtis points in the studied direction and 2 points in all other directions except for the burnup-grid with 25 points in the burnup direction (see the description in section 4.1).

These discretizations are illustrated in a one-dimensional space in figure 11 and are described in table 2.



x_j-grids $\longrightarrow$ Tucker decomposition

multilinear grid $\longrightarrow$ multilinear interpolation

reference grid $\longrightarrow$ to calculate evaluated error of each method

Figure 11: Illustration of the different grids used in the test case.

Direction	*reference grid*	*multilinear grid*	*x_j-grids*
burnup	36	33	25 ($25 * 2^4 = 400$ points)
Fuel temperature	4	3	5 ($5 * 2^4 = 80$ points)
Moderator density	4	3	5 ($5 * 2^4 = 80$ points)
Boron concentration	4	3	5 ($5 * 2^4 = 80$ points)
Xenon level	3	2	5 ($5 * 2^4 = 80$ points)
Total	6912 points $= 36 * 4^3 * 3$	1782 points $= 33 * 3^3 * 2$	720 points $= 25 * 2^4 + 4 * (5 * 2^4)$

Table 2: The discretization of the *reference grid*, of the *multilinear grid* and of the *x_j-grids*.

5.3. Approximation errors for cross-sections

We compare the accuracy of the multilinear interpolation and the Tucker decomposition on each node $\mathbf{x}_i$ of the *reference grid*. The approximation error on the *reference grid* is defined either by the infinity norm (inf) of relative

errors or by the root mean square of absolute errors ($RMSE$) :

$$e^{(inf)} = \max_{\mathbf{x}_i \in reference\ grid} | \underbrace{\frac{\tilde{f}(\mathbf{x}_i) - f(\mathbf{x}_i)}{\max|\{f(\mathbf{x}_i)\}|} * 10^5}_{\text{relative error}} | \tag{24}$$

or

$$e^{(RMSE)} = \frac{\sqrt{\sum_{i=1}^{N}[f(\mathbf{x}_i) - \tilde{f}(\mathbf{x}_i)]^2}}{\sum_{i=1}^{N}\sqrt{f^2(\mathbf{x}_i)}} * 10^5, \text{ with } N = \#(reference\ grid) \tag{25}$$

where $f(\mathbf{x}_i)$ and $\tilde{f}(\mathbf{x}_i)$ are respectively exact values (calculated by APOLLO2) and approximated values (evaluated by either method) of a cross-section, performed at $\mathbf{x}_i \in$ *reference grid*. The 10^5 factor is here to have units in *pcm*.

We denote e_{Tucker} and $e_{multilinear}$ respectively the error by the Tucker approximation and by the multilinear interpolation.

5.4. Approximation errors for reactivity

Cross-sections are merely inputs for the flux solver. We are more interested in k_{eff} or reactivity for instance, which are outputs of the flux solver. In some simplified cases, we do not need to solve a neutron equation, the following analytic formula is applied (see page 1172, 1173, 1221 of the book [1]):

$$\text{reactivity} = 1 - \frac{1}{k_{eff}}, \text{with } k_{eff} = k_\infty = \frac{\nu\Sigma_f^1 * (\Sigma_t^2 - \Sigma_{s0}^{2\to2}) + \nu\Sigma_f^2 * \Sigma_{s0}^{1\to2}}{(\Sigma_t^1 - \Sigma_{s0}^{1\to1}) * (\Sigma_t^2 - \Sigma_{s0}^{2\to2}) - \Sigma_{s0}^{1\to2} * \Sigma_{s0}^{2\to1}}$$

In order to measure the reactivity error, we use $\mathbf{e}^{(inf)}_{\text{reactivity}}$ with the following definition:

$$\mathbf{e}^{(inf)}_{\text{reactivity}} = \max_{\mathbf{x}_i \in reference\ grid} |(\frac{1}{k_\infty(\mathbf{x}_i)} - \frac{1}{\tilde{k}_\infty(\mathbf{x}_i)}) * 10^5|$$

5.5. Results

In this section, results will be shown for the two cases: the Tucker decomposition using and not using the EIM. We will compare these results with the multilinear model using the following criteria: the number of calculation points, the storage and the accuracy.

We present in table 3, the results issued from the Karhunen-Loève decomposition for some macroscopic cross-sections. This allows us to compute the number of APOLLO2 calculations and the storage required by this method (see section 4.4.2).

Cross section	**Number of tensor directional basis functions r_j for direction j**					**Number of coefficients a** ($\prod_{j=1}^{d} r_j$)
	r_{burnup}	$r_{Fuel\,temperature}$	$r_{Moderator\,density}$	r_{Boron}	r_{Xenon}	
Σ_t^1	4	2	2	2	2	64
Σ_t^2	3	2	3	2	2	72
Σ_a^1	5	3	2	3	2	180
Σ_a^2	5	2	3	3	2	180
$\nu\Sigma_f^1$	7	2	2	3	2	168
$\nu\Sigma_f^2$	5	2	3	2	2	120
$\Sigma_{s0}^{1\to1}$	4	2	2	2	2	64
$\Sigma_{s0}^{1\to2}$	4	3	3	3	2	216
$\Sigma_{s0}^{2\to1}$	6	2	3	3	2	216
$\Sigma_{s0}^{2\to2}$	3	2	3	2	2	72
Σ_f^1	7	2	2	2	2	112
Σ_f^2	5	2	3	2	2	120
$\mathbf{r_j^{max}}$	***7***	***3***	***3***	***3***	***2***	
$\Longrightarrow$ **Number of evaluated points** $\{\mathbf{x}_t\}_{t=1}^{R}$ (used on all right hand sides of the systems (15), determined by (18)): 7*3*3*3*2 = **378**						

Table 3: Number of: tensor directional basis functions, coefficients **a** and evaluated points **x** in the Tucker decomposition for some considered cross-sections.

5.5.1. Comparison of the number of calculation points

The number of calculation points for the Tucker model (see (22)) includes the calculations on all nodes of x_j-grids (in this case, we have 5 grids for 5 parameters with 720 nodes in total (table 2)) and the calculations performed on the evaluated points **x** on the right hand side of (15) (in this case, we have 378 evaluated points (see table 3)). The number of calculation points for the multilinear model (see (20)) is equal to the number of nodes in the multilinear grid. Hence, we obtain the results presented in table 4. These results show that, by using the Tucker decomposition, the number of calculation points has been reduced by 38% compared with the multilinear model.

Number of calculation points	
Tucker decomposition	Multilinear Interpolation
$\underbrace{720}_{\textit{for the 5 } x_j\textit{-grids}}$ + $\underbrace{378}_{\text{for evaluated points } \mathbf{x}}$ =**1098**	**1782**

Table 4: Comparison of the number of calculation points between the Tucker decomposition and the multilinear interpolation.

5.5.2. Comparison of the storage

The storage of the Tucker decomposition for one cross-section includes the axial discretization values, the values of selected eigenvectors and the coefficients **a** (see (23)). *The storage of the multilinear model* includes the axial discretization values, the cross-section values on the nodes of the multilinear grid (21). The number of axial discretization values (25+4*5 = 45 for the Tucker model and 33+3+3+3+2 = 44 for the multilinear model) is the same for all cross-sections and becomes small when all cross-sections are considered. Therefore, these numbers can be neglected in the comparison of the storage. We obtained

the results presented in table 5 for some cross-sections. These results show that, by using the Tucker decomposition, the storage size has been reduced by a factor from 4 to 9 (depending on the cross-section) compared to the multilinear model.

Cross section	**Storage (number of floats)**	
	Tucker	Multilinear
Σ_t^1	244	1782
Σ_t^2	192	1782
Σ_a^1	355	1782
Σ_a^2	355	1782
$\nu\Sigma_f^1$	388	1782
$\nu\Sigma_f^2$	290	1782
$\Sigma_{s0}^{1\to1}$	244	1782
$\Sigma_{s0}^{1\to2}$	371	1782
$\Sigma_{s0}^{2\to1}$	416	1782
$\Sigma_{s0}^{2\to2}$	192	1782
Σ_f^1	327	1782
Σ_f^2	290	1782

Table 5: Comparison of the storages between the Tucker decomposition and the multilinear interpolation.

5.5.3. Comparison of accuracy

When the tensor directional basis functions are determined, the accuracy of the Tucker decomposition depends only on values of the coefficients $\mathbf{a}$. We recall that the storage size (e.g. in table 5) does not depend on the coefficient values. We present here the results of the Tucker decomposition with and without using the EIM (it does not change the storage size). We recall that without the EIM, we select randomly the evaluated points. The accuracies of the Tucker decomposition are compared to the multilinear interpolation over the 6912 points of the reference grid.

The comparisons are presented in table 6. In this table, the Tucker decomposition without using the EIM has already a better accuracy for the e^{RMSE} errors for all reconstructed cross-sections (see in column 6), compared to the multilinear interpolation (see in column 5). But this is worse than the multilinear one for some cross-sections ($\nu\Sigma_f^1$, Σ_f^1) with e^{inf} errors (presented by framed errors) and for the reactivity with e^{RMSE}. With the EIM, all errors are better than those of the multilinear interpolation.

Figure 12 presents the relative errors for the cross-section $\nu\Sigma_f^1$. In this case, the accuracy of the Tucker decomposition using the EIM is the best, compared to the multilinear model and the Tucker decomposition without using the EIM.

Figure 13 presents the relative errors for the cross-section $\Sigma_{s0}^{2\to1}$. In this case, the accuracy of the Tucker decomposition is also better than that of the multilinear interpolation. The distribution of errors in the case using the EIM seems to be the same as the case without using the EIM.

Figure 14 presents the approximation errors of the reactivity. The accuracy of the Tucker decomposition with the EIM is the same order of multilinear

interpolation accuracy.

These results show that the Tucker model is more accurate than the multilinear model in general while reducing significantly the computational cost (number of APOLLO2 calculations and storage in neutron libraries).

Cross Section	$\mathbf{e}^{(\mathbf{inf})}_{\mathbf{multilinear}}$	$\mathbf{e}^{(\mathbf{inf})}_{\mathbf{Tucker}}$ (without EIM)	$\mathbf{e}^{(\mathbf{inf})}_{\mathbf{Tucker}}$ (with EIM)	$\mathbf{e}^{(\mathbf{RMSE})}_{\mathbf{multilinear}}$	$\mathbf{e}^{(\mathbf{RMSE})}_{\mathbf{Tucker}}$ (without EIM)	$\mathbf{e}^{(\mathbf{RMSE})}_{\mathbf{Tucker}}$ (with EIM)
Σ_t^1	47.24	28.15	26.58	12.26	6.05	5.94
Σ_t^2	118.11	40.63	34.67	59.45	11.16	7.96
Σ_a^1	195.76	25.86	28.10	76.31	8.47	9.61
Σ_a^2	417.01	49.64	49.52	89.91	20.03	21.01
$\nu\Sigma_f^1$	116.65	130.66	42.58	44.04	21.09	12.79
$\nu\Sigma_f^2$	289.75	72.80	70.72	81.32	27.68	29.06
$\Sigma_{s0}^{1\to1}$	40.86	25.42	23.88	10.63	5.58	5.44
$\Sigma_{s0}^{1\to2}$	194.94	24.07	19.28	43.67	6.43	6.34
$\Sigma_{s0}^{2\to1}$	794.39	97.17	96.73	328.21	63.61	60.78
$\Sigma_{s0}^{2\to2}$	126.08	10.61	18.92	62.60	3.19	4.19
Σ_f^1	122.62	153.09	58.79	44.53	22.51	13.86
Σ_f^2	297.06	68.16	65.81	81.07	25.86	27.12
Reactivity	**467.40**	**342.82**	**334.04**	**85.92**	**88.19**	**72.26**

Table 6: Comparison of approximation errors (pcm) on 6912 points of the reference grid for the Tucker decomposition and for the multilinear interpolation.

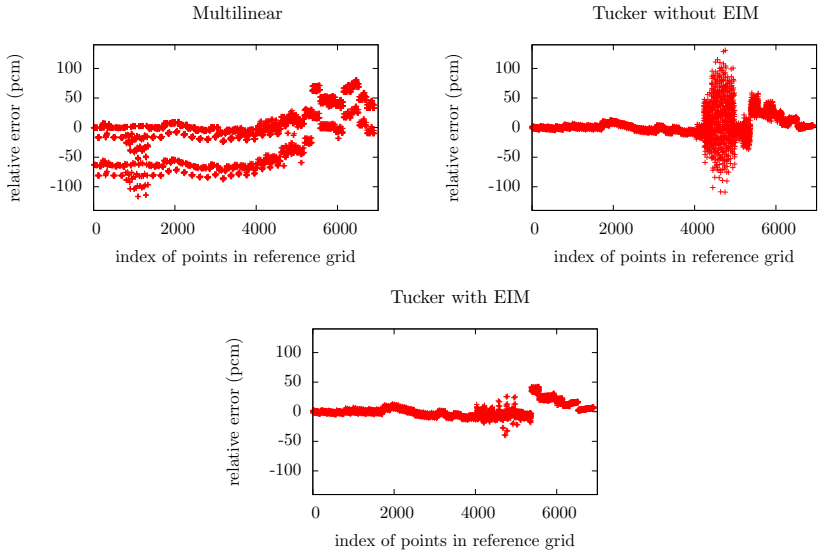


Figure 12: Comparison of relative errors (pcm) for the cross-section $\nu\Sigma_f^1$.

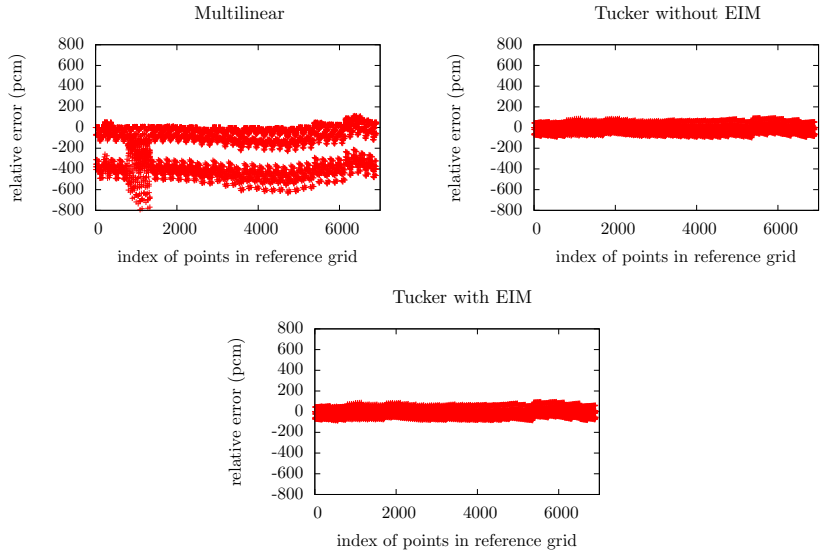


Figure 13: Comparison of relative errors (pcm) for the cross-section $\Sigma_{s0}^{2\to1}$.

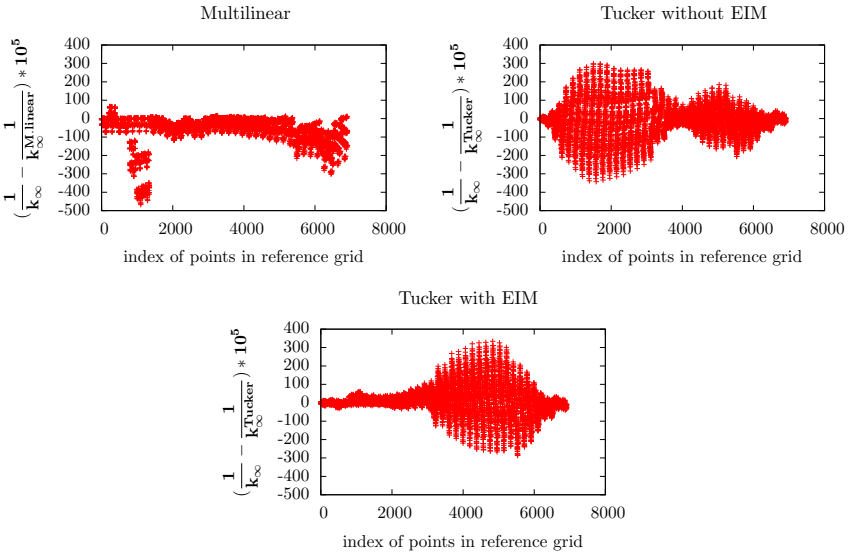


Figure 14: Comparison of approximation errors (in pcm) for the reactivity.

6. Conclusion and discussion

We have presented the Tucker decomposition applied to the reconstruction of neutron cross-sections. We showed that this method has better accuracy for cross-sections while using a smaller number of calculation points and a smaller storage size than those of the multilinear interpolation. Since cross-sections are merely inputs for the flux solver, the results for the reactivity are especially interesting. In the simplified infinite medium hypothesis, we showed that the reactivity accuracy with the Tucker model is of the same order of multilinear interpolation one.

However, the Tucker decomposition has some limitations compared to the multilinear model. It could be slower than the multilinear interpolation in the reconstruction step. At this stage, the Tucker model uses the Lagrange interpolation to evaluate the tensor directional basis functions (expensive in time), while the multilinear model uses the linear interpolation. Moreover, the multilinear interpolation ensures the positivity of cross-sections contrary to the Tucker model. The multilinear interpolation also keeps the linearity relation, e.g. $\Sigma_t^1 = \Sigma_a^1 + \Sigma_{s0}^{1\to 2} + \Sigma_{s0}^{1\to 1}$ is correct at every point, while it is not true for the Tucker decomposition (unless we build Σ_t^1 like this expression).

Our future work will focus on the resolution of these Tucker limitations. Furthermore, we plan to study a criteria which allows us to eliminate the less important coefficients $\mathbf{a}$ in the representation of the Tucker decomposition. This should lead to a similar accuracy while reducing storage in the neutron libraries. We will also apply the Tucker model to more complex cases, e.g. extension of the calculation domain. The accuracy will be verified for real values of k_{eff} which are performed by the flux solver instead of analytic formula which is only valid for simplified cases.

References

[1] S. Marguet, La physique des réacteurs nucléaires, 2ème édition, Tec& Doc, Lavoisier, 2013.

[2] S. Harris, An introduction to the theory of the Boltzmann equation, Courier Corporation, 2004.

[3] R. Sanchez, I. Zmijarevic, M. Coste-Delclaux, E. Masiello, S. Santandrea, E. Martinolli, L. Villate, N. Schwartz, N. Guler, APOLLO2 year 2010, Nuclear Engineering and Technology 42 (2010) 474 – 499.

[4] D. Botes, P. M. Bokov, Hierarchical, multilinear representation of few-group cross sections on sparse grids, in: International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2011), Rio de Janeiro, Brazil, 2011.

[5] B. Danniëll, M. B. Pavel, Polynomial interpolation of few-group neutron cross sections on sparse grids, Annals of Nuclear Energy 64 (February 2014) 156–168.

[6] G. Hobson, H.-W. Bolloni, K.-A. Breith, R. van Geemert, H. Haase, B. Hartmann, ARTEMISTM Core Simulator: Latest developments, in: International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2013 (SNA + MC 2013), 2013.

[7] T. H. Luu, Y. Maday, M. Guillo, P. Guérin, Reconstruction of neutronic macroscopic cross-sections using Tucker decomposition, in: International Conference on Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method, Nashville, Tennessee, USA, 2015.

[8] W. Hackbusch, Tensor spaces and numerical tensor calculus, Springer, 2012.

[9] Y. Maday, O. Mula, G. Turinici, et al., A priori convergence of the generalized empirical interpolation method., in: 10th international conference on Sampling Theory and Applications (SampTA 2013), 2013, pp. 168–171.

[10] A. Nouy, Low-rank tensor methods for model order reduction (2016) 1–26.

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

[12] K. Karhunen, Zur Spektraltheorie stochastischer Prozesse, Annales Academiae scientiarum Fennicae. Series A. 1, Mathematica-physica, 1946.

[13] M. Loève, Probability Theory: Foundations, Random Sequences, University series in higher mathematics, Van Nostrand, 1955.

[14] G. Berkooz, P. Holmes, J. L. Lumley, The proper orthogonal decomposition in the analysis of turbulent flows, Annual review of fluid mechanics 25 (1) (1993) 539–575.

[15] K. Pearson, On lines and planes of closest fit to systems of points in space, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 2 (11) (1901) 559–572.

[16] H. Hotelling, Analysis of a complex of statistical variables into principal components, Journal of Educational Psychology 24(6) (1933) 417–441 and 498–520.

[17] S. W. Lyons, Empirical orthogonal function analysis of Hawaiian rainfall, Journal of Applied Meteorology 21 (11) (1982) 1713–1729.

[18] J. Šimša, The best L2-approximation by finite sums of functions with separable variables, Aequationes mathematicae 43(2-3) (1992) 248–263.

[19] L. De Lathauwer, B. De Moor, J. Vandewalle, A multilinear singular value decomposition, SIAM journal on Matrix Analysis and Applications 21(4) (2000) 1253–1278.

[20] K. E. Atkinson, The numerical solution of integral equations of the second kind, Vol. 4, Cambridge university press, 1997.

[21] C. W. Clenshaw, A. R. Curtis, A method for numerical integration on an automatic computer, Numerische Mathematik 2(1) (1960) 197–205.

[22] J. P. Boyd, Chebyshev and Fourier spectral methods, Courier Dover Publications, 2001.

[23] L. N. Trefethen, Is Gauss quadrature better than Clenshaw-Curtis?, SIAM review 50 (1) (2008) 67–87.

[24] Y. Maday, N. C. Nguyen, A. T. Patera, G. S. Pau, A general, multipurpose interpolation procedure: the magic points.