Proceedings of the Seventeenth International Conference on
Civil, Structural and Environmental Engineering Computing
Edited by: P. Iványi, J. Kruis and B.H.V. Topping
Civil-Comp Conferences, Volume 6, Paper 13.4
Civil-Comp Press, Edinburgh, United Kingdom, 2023
doi: 10.4203/ccc.6.13.4
©Civil-Comp Ltd, Edinburgh, UK, 2023

Analytical Hierarchical Tucker Representation using Binary Trees

Z. Qiu[1,2], F. Magoulès[2,3], D. Peláez[1]

[1] **Institut des Sciences Moléculaires d'Orsay, Université Paris-Saclay, Orsay, Île-de-France, France**
[2] **MICS, CentraleSupélec, Paris-Saclay University, Gif-sur-Yvette, France**
[3] **Faculty of Engineering and Information Technology, University of Pécs, Pécs, Hungary**

Abstract

In this contribution we show that it is possible to achieve an analytical binary tree representation for a tensor stemming from an underlying scalar field. As initial data-structure we use a binary tree. This is obtained by a hierarchical Tucker (HT) decomposition of a reference tensor. To achieve this, tensor matricizations are followed by their truncated singular value decompositions. Then we fit the left singular vectors at each node using a set of auxiliary basis functions. These are system-dependent orthogonal polynomials. We call this finite basis representation (FBR). The resulting HT-FBR expression can be reconstructed to grids of any density, within the same domain of definition, while keeping the error reasonably/physically small. This paves the way to the direct optimisation of these compact analytical binary tree structures.

Keywords: tensor decomposition, low-rank approximations, finite basis representation, singular value decomposition, analytical data-structure, binary tree

1 Introduction

Tensors, *aka* multidimensional arrays of data, are ubiquitous in the fields of mathematics, engineering, as well as natural sciences, in those problems represented on grids. Owing to the so-called curse of dimensionality [2], tensor decomposition methods have become essential for the efficient treatment of high-dimensional data structures. This is of particular relevance in Deep Learning approaches [1]. In the following paragraphs, we will succinctly describe some of the terms and concepts necessary for our discussion.

Let A be a scalar field (a multivariate function of the coordinates) mapped onto a discrete d-dimensional space:

$$A(x_1, x_2, \dots, x_d) \to \mathbb{R}^d \tag{1}$$

The tensor resulting from its mapping onto a grid reads:

$$A \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_\mu \times \ldots \times I_d} \tag{2}$$

where I_μ refers to the number of grid points associated to mode μ. Unless otherwise stated, the scalar fields we refer to will correspond to smooth physical quantities (continuous and differentiable functions). In other words, when mapped onto a tensor two any of its elements will fulfil:

$$|A_{i_1, i_2, \ldots, i_\mu, \ldots i_d} - A_{i_1, i_2, \ldots, (i+1)_\mu, \ldots i_d}| < \epsilon \tag{3}$$

with ϵ small. Grid representations suffer from the so-called *curse of dimensionality* which refers to the exponential growth in data-points and the concomitant number of operations upon increase of the number of dimensions of the system. Whereas most common applications in engineering or applied mathematics will generally make use of tensors with d = 3 or 4 (*e.g.* heatmaps, video or image compression), in natural sciences, on the other hand, d can be of any dimension. This is epitomized by the field of quantum mechanics in which the (wave) function defining an N-body system ($N \gg 4$) will usually depend on the coordinates of every particle. As a consequence, accurate low-rank tensor approximations become mandatory. In fact, major developments in tensor representations have arisen from this field. [4, 7, 9, 10]

Specifically in this work we will focus on the so-called sum-of-products (SOP), separable form. These are compact, low-rank representations which are, in turn, very appropriate for the solution of multidimensional integrals. [7] Indeed, a multidimensional integral can be turned into a SOP of one-dimensional integrals provided that every quantity in the integrand is of SOP form. Arguably, the most widely employed SOP *ansätze* are Tucker 4:

$$A_{i_1, \ldots, i_d} \approx A^{\text{Tucker}}_{i_1, \ldots, i_d} = \sum_{j_1}^{m_1} \cdots \sum_{j_d}^{m_d} C_{j_1, \ldots, j_d} \prod_{\mu=1}^{d} U_{i_\mu j_\mu} \tag{4}$$

and the canonical polyadic (CP) form 5:

$$A_{i_1,\ldots,i_d} \approx A^{\text{CP}}_{i_1,\ldots,i_d} = \sum_{r=1}^{R} \lambda_r \prod_{\mu=1}^{d} U_{i_\mu r} \tag{5}$$

In the case of Tucker any tensor element is approximated by an expansion in terms of a set of, exponentially growing, *core* tensor coefficients $C_{j_1,\ldots,j_d}$ and a set of *factor matrices* (U). In contrast, for CP, a much smaller number of terms is needed since no exponential scaling is present. Though formally similar, the main difference between them is that, contrary to Tucker, the CP factor matrices are not orthogonal, though the column vectors are possibly normalized. This extra degree of flexibility leads to very compact CP expansions when compared to Tucker ones. The traditional algorithms to obtain such expressions are algebraic, in other words, grid-based and, consequently severely limited by dimensionality.

Recently some of us have proposed turning such grid-based tensor decompositions into analytical expressions though the use of a set of auxiliary basis functions which, hereafter, we will refer to as Finite Basis Representation (FBR). We have done this for the Tucker form (SOP-FBR) [6] as well as for CP schemes (CP-FBR) [5]. In the former case, Tucker, the core tensor grows exponentially thus limiting its scope of application. In the latter, despite its linear scaling, the computation of the basis is problematic. [5,9] Since HTD provides an intermediate path, polynomial scaling (*vide infra*), in this work, we propose to apply our FBR approach to a hierarchical Tucker decomposition (HT) and show that the analogous HT-FBR not only possible but also advantageous with respect the other two methods. In short, the factor matrices in Eq. 4 can be expressed in SOP-FBR form:

$$\sigma : \mathbb{R}^{n_t} \mapsto \mathbb{R}^{k_t+1} \tag{6}$$

$$P : \mathbb{R}^{n_t \times r_t} \mapsto \mathbb{R}^{(k_t+1)\times r_t}, P(U_t) = \sum_{r}^{r_t} \sigma_r((U_t)_r) \tag{7}$$

with k_t being the degree of expansion polynomial σ. The latter will provide a means of interpolating each of the columns of U_t. Its choice will be problem dependent. [6]. Usually storage of a polynomial coefficient is far less than an entire U_t since $n_t > k_t + 1$ in most of the case. As a result, Tucker format in the terms of FBR is therefore:

$$A^{*}_{i_1,\ldots,i_d} = \sum_{j_1} \cdots \sum_{j_d} C_{j_1,\cdots,j_d} \prod_{t}^{d} (\sum_{r}^{r_t} \sigma_r((U_t)_r)) \tag{8}$$

2 Theory

In what follows, we briefly describe the algorithm we have employed for our initial Hierarchical Tucker decompositions (HTD) [3]. Then, we show how using a set of auxiliary basis (finite basis representation, FBR), one can turn the HTD structure into an analytical HT-FBR function. Finally, we apply it to a model 6D tensor (stemming from a scalar field) to illustrate its properties.

2.1 Hierarchical Tucker decomposition of tensors

Let $A \in \mathbb{R}^I$ be an d dimension tensor, note index set in each dimension $\mu \in \mathbb{Z}, \mu < d$ as I_μ such that:

$$I := I_1 \times ... \times I_\mu \times ... \times I_d \tag{9}$$
$$I_\mu := (1, ..., n_\mu), \mu \in (1, ..., d) \tag{10}$$
$$I^\mu := I_1 \times ..., I_{(\mu-1)} \times I_{(\mu+1)}, ..., I_d \tag{11}$$

The orthogonal Tucker format of tensor A:

$$A = (U_1, ..., U_d) \circ C \tag{12}$$

in which $U_\mu \in \mathbb{R}^{n_\mu \times r_\mu}$ so called mode frame, r_k is k^{th} Tucker rank of A and entire Tucker rank pair forms a tuple $r_n \in \mathbb{Z}^d = (r_1, ..., r_\mu, ..., r_d)$. Notation $\circ$ represents the multiliniear multiplication. A mode-μ multiplication product $U_\mu \circ_\mu A$'s elements can be written as

$$U_\mu \circ_\mu A_{(i_1,...,i_{\mu-1},i,i_{\mu+1},...,i_d)} = \sum_{k=1}^{I_\mu} A_{(i1,...,i_{\mu-1},k,i_{\mu+1},...,i_d)} U(i,k) \tag{13}$$

U_μ is the mode frame of representation as 12. Actually, 12 with 13 lead to SOP form of Tucker in 4

We here use the notation $A^t := M_t(A)$ matricization of A as in [3] and

$$I_t := \times_{\mu \in t} I_\mu \tag{14}$$
$$I_{t'} := \times_{\mu \in t'} I_\mu \tag{15}$$

where t is a mode cluster and $t' := \{1, ..., d\} \diagdown t$ the complementary cluster, then

$$M_t : \mathbb{R}^I \to \mathbb{R}^{I_t \times I_{t'}}, \tag{16}$$
$$(M_t(A))_{(i_\mu)_{\mu \in t}, (i_\mu)_{\mu \in t'}} := A_{i_1,...,i_d} \tag{17}$$

Let T be a binary tree for dimension d_t such that any node $t_n, n < d_t$ belonging to T has and only has at most two child nodes, depth of $T = \lceil log_2(d_t) \rceil$. We note a node t_n belongs to leaf node set $\mathfrak{L}$ if it has no associated child node while root node is the only node in the binary tree has no parent node. In between interior nodes are node with one and only one parent node and no less than one child node, let $\mathfrak{I}$ be the entire set of interior nodes. A full binary tree is a binary tree with all nodes belongs to the tree has either no or two child nodes. Let full binary tree associate a cluster of mode to form a dimension tree T_I, we propose a $d_t = 6(d = 4)$ example in Figure 1, each node is distributed a mode cluster t, i.e. $t = \{0, 1, 2, 3\}$ in the root node. Mode clusters are deployed from root to leaves recursively.

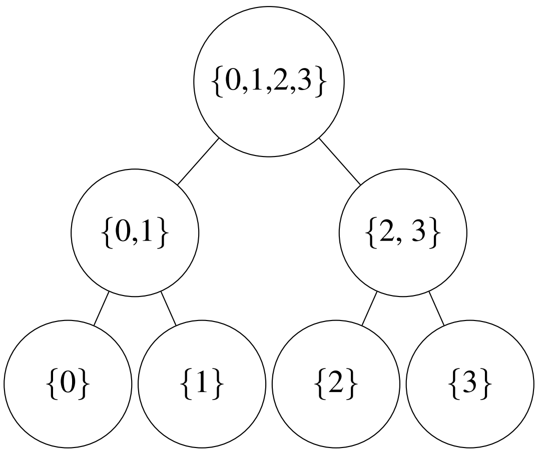

Figure 1: Example of a 4D cluster tree, each node is so-called a mode cluster.

For each node we settle A^t and compute its SVD decomposition,

$$A^t = U_t \Sigma_t V_t^T \tag{18}$$

U_t and V_t are the set of singular vectors of A^t, Σ_t is a diagonal matrix where diagonal elements are singular values in descending order. Then we choose first r_t columns as mode frame of node. Each mode frame U_t belongs to $\mathfrak{I}$ can be represented by mode frames U_{tl} and U_{tr} of its child nodes by a transfer tensor B_t:

$$(B_t)_{i,j,v} := (U_t)_i \cdot (U_{tl})_j \otimes (U_{tr})_v \tag{19}$$

The cluster $((U_t)_{t\in\mathfrak{L}}, (B_t)_{t\in\mathfrak{I}})$ forms a hierarchical decomposition of tensor A. The truncation of HTD, in SOP form, is defined by

$$\pi_{(t\in T_I^l)} = A \cdot U_t \tag{20}$$

$$A^H = \prod_{t\in T_I^1} \cdots \prod_{t\in T_I^d} \pi_t A \tag{21}$$

$\pi_{(t\in T_I^l)}$ is the projection of A on the U_t at level l of T_I. Total storage complexity less than $(d-1)r_{max}^3 + r_{max}\sum_{\mu=1}^d n_\mu$, $r_{max} = max(r_1, ..., r_\mu, ..., r_d)$ [3]. As comparison, Tucker storage grows exponentially with the total rank $\prod_{\mu=1}^d r_\mu + \sum_{\mu=1}^d n_\mu \cdot r_\mu$ while CP grows linearly with r, that is, $r + r\sum_{\mu=1}^d n_\mu = r(1 + \sum_{\mu=1}^d n_\mu)$.

3 Hierarchical Tucker using Finite Basis Representation (HT-FBR)

In a nutshell, we first define the hierarchical Tucker format (HT) of a tensor, we introduce the FBR to interpolate the left singular vectors of each mode. This typically would reduce the storage complexity. So far, according to the methods introduced before, we propose a new algorithm, which combines HTD and FBR, interpolates the leaf nodes or all nodes of HTD, and further reduces the storage complexity while ensuring high precision.

$\mathfrak{L}$ is set of leaf nodes belongs to a frame tree T_I, $U_{t\in\mathfrak{L}}$ are mode frames associated with $\mathfrak{L}$. We therefore interpolate $U_{t\in\mathfrak{L}}$ with same $B_{t\in\mathfrak{I}}$ as those in 19 since B_t represents the linear combination of the Kronecker product, thus combination still holds. We have a HTD-reconstructed tensor (A^H):

$$A^H = \prod_{t\in T_I^1} \cdots \prod_{t\in T_I^d} P(\pi_{t\in\mathfrak{L}})A \tag{22}$$

We simply note $U_t^c \in \mathbb{R}^{k_t\times r_t} = \sum_r^{r_d} \sigma_{r_k}(U_t)_r$ The frame tree associated with $((U_t^c)_{t\in\mathcal{L}}, (B_t)_{t\in\mathcal{L}^c})$ is called HT-FBR of tensor A. Storage complexity in HTD2.1 becomes less than $(d-1)r_{max}^3 + r_{max}\sum_{\mu=1}^d k_t$

Since the fact that the basis functions of polynomial interpolation are a set of continuous functions, it is an obvious idea to expand grid of sample tensor to a finer one as named in [5], so called primitive grid for the former and coarse grid for the later.

$$\sigma^\dagger : \mathbb{R}_\Omega^{n_t} \mapsto \mathbb{R}_\Omega^{r_N}, \Omega \in [a \in \mathbb{R}, b \in \mathbb{R}], a < b \tag{23}$$

$$P^\dagger : \mathbb{R}_\Omega^{n_t\times r_t} \mapsto \mathbb{R}_\Omega^{r_N\times r_t} \tag{24}$$

$$A^\dagger = \prod_{t\in T_I^1} \cdots \prod_{t\in T_I^d} P_\Omega^\dagger(\pi_{t\in\mathfrak{L}})A \tag{25}$$

$\sigma^\dagger$ is a polynomial interpolation function and Ω is a target domain of $\mathbb{R}$ range from a to b that $a < b$. It expand current grid order n_t to r_N normally higher than n_t. The initial tensor A then is projected to a higher order set of π_t which leads to avoid calculate a exact higher order one.

4 Results and discussion

In the following we will present a discussion on the performance of our approach by comparing to other available methods. Python and Numpy have been used throughout this work. Concerning CP and HOOI (Tucker flavour) tensor decomposition methods, we have used the Tensorly library. The HTD and HTD-FBR are our own implemementations developed with Python version 3.10, Numpy version is 1.23.5 . In the experiment we use Numpy polynomial package to interpolate modes, mainly used Hermite and Chebyshev polynomial. All the figures have been realised with Matplotlib 3.7.1.

4.1 Proof of concept: HT-FBR for a 6D scalar field

We now discuss a physically meaningful and more general example. We first test the performance of HT-FBR with different degree of polynomial basis for the reconstruction of a model 6D scalar field like defined in [8]:

$$A_{scal}(R_1, R_2, R_3, \theta_1, \theta_2, \tau) = \\ S_0 + e^{-D(R_1,R_2,R_3,\theta_1,\theta_2)} \times \sum_{ijklmn} c_{ijklm} Q_1^i Q_2^j Q_3^k Q_4^l Q_5^m \cos(nQ_6) \tag{26}$$

where $D = \sum_{i=1}^{3} d_i(R_i - R_i^{ref})^2 + \sum_{j=1}^{2} d_{j+3}(\theta_j - \theta_j^{ref})^2$ and $Q_{1/2/3} = 1 - e^{-0.7(R_{1/2/3} - R_{1/2/3}^{ref})}$, $Q_{4/5} = \theta_{1/2} - \theta_{1/2}^{ref}$, $Q_6 = \tau - \tau^{ref}$ with a set of known parameters R_1^{ref}, R_2^{ref}, R_3^{ref}, θ_1^{ref}, θ_2^{ref}, τ^{ref}.

Our reference tensor (A_{scal}), to be HT-decomposed, is defined in Table 1: its dimensions, aka sampled points per interval, ($\{n_t\}$), the dimension of the reconstructed tensors, interpolated values, ($\{R_{N_i}\}$, with $i = s, l$) as well as the domain of definition (Ω) which is constant. The latter corresponds to the range of definition of the physical coordinates. Our experiment will consist in using the HT-FBR of A_{scal} to expand its representation on the initial (and coarser) grid to a finer one ($A^\dagger$). We have considered two *finer* cases: (i) a small (but fine) grid ($A_s^\dagger$); and (ii) a large (even finer) one ($A_l^\dagger$).

Definition of the 6D surface A_{scal} mapping				
dim (node)	n_t	R_{N_s}	R_{N_l}	Ω
0	5	7	10	[2.10, 3.25]
1	5	8	10	[1.30, 2.45]
2	5	7	10	[1.90, 2.60]
3	5	7	10	[-0.65, 0.25]
4	5	7	10	[-0.65, -0.10]
5	12	15	24	[0, π]

Table 1: Interval with sample points number n_t in each dimension and corresponding interpolated points number R_{N_s} and R_{N_l} as defined in Eq. (23)

Table 2 presents the RMSE analysis of the polynomial reconstructed (U_t) : leaf nodes and interior nodes in binary tree after reconstruction with different node rank and different k_t compare with the U_t of the node (from HTD) See Fig. 1 for the node notation. The second column shows the node rank (r_k) for corresponding node. The third column presents either the degree of the polynomial (k_t). Note that as discussed in the Theory section, we have stored the HTD transfer tensors. It is hence possible to converge the interpolation at each node by choosing an appropriate k_t and node rank. As a result of this simple step, we have obtained an analytical expression which can be evaluated to reconstruct tensors of *any* data-point density. As discussed below, the error remains virtually constant to any degree of interpolation (see Table 3). This has also been observed in our other SOP-FBR and CP-FBR methods [5, 6]. This reflects the smooth nature of the underlying scalar field. More relevantly, the number of hyperparameters of our HT-FBR model is, obviously, independent of the size of the reconstructed tensor so that one can get a reasonable (constant) sum-of-product low-rank binary tree approximation to tensors of any size.

Quality of the node frame polynomial reconstruction (A_{scal} HTD)			
node	r_k	k_t	reconstruction RMSE
$\{0\}$	5	5	1.786e-13
$\{5\}$	12	5	0.204
$\{5\}$	12	10	0.083
$\{5\}$	12	15	2.447e-12
$\{1, 2\}$	(10, 5)	5 (max leaf)	1.524e-16
$\{1, 2\}$	(25, 5)	5 (max leaf)	1.833e-16
$\{4, 5\}$	(10, 12)	5 (max leaf)	0.085
$\{4, 5\}$	(10, 12)	15 (max leaf)	9.034e-13
$\{4, 5\}$	(60, 12)	10 (max leaf)	0.037
$\{4, 5\}$	(60, 12)	15 (max leaf)	1.125e-12
$\{0, 1, 2\}$	(100, 10, 10)	None	0.065
$\{3, 4 ,5\}$	(100, 10, 10)	None	0.045
$\{3, 4 ,5\}$	(300, 60, 12)	None	9.911e-17
$\{3, 4 ,5\}$	(300, 60, 12)	15 (max leaf)	1.178e-15

Table 2: Influence of the node rank and polynomial degree on the RMSE of the reconstructed node frames.

In 2, we display the change in RMSE with the growing total rank. For full rank, HTD and HT-FBR both converge to a minimum faster as the node rank increases, and CP overfits when the node rank beyond 600. The CPU time necessary for HTD and HT-FBR in the high-dimensional case increases proportionally to the node rank. This is due to the fact that the algorithms are neither parallelized nor memory optimized yet. All four methods converge to an accuracy lower than 1e-9 which is more than sufficient for practical physical applications. Both HTD and HOOI (Tucker) reach 1e-15 and HT-FBR converges to 1e-13. On the other hand, CP (ALS) reaches a minimum of 1e-9. A comprehensive mode-analysis is presented in Fig.3.

Tensor decomposition methods on 6D A_{scal} (5, 5, 5, 5, 5, 12)			
Method	RMSE	rank (or node rank)	time
HOOI	2.47835e-15	(6, 6, 6, 6, 6, 6) total: 46656	3.6781s
CP	3.40570e-9	593	6.8984s
HTD	9.02599e-15	(300, 60, 12) total: 547	13.1288
HT-FBR (Hermite) ($k_t = 10$)	7.64059e-3	(100, 60, 12) Total: 357	6.5228
HT-FBR (Hermite) ($k_t = 10$)	5.50068e-7	(300, 60, 12) Total: 547	13.4994
HT-FBR (Hermite) ($k_t = 15$)	7.08053e-13	(300, 60, 12) Total: 547	13.1376

Table 3: Comparison of the quality and CPU times for common tensor-decomposition methods: (i) Tucker (HOOI); (ii) CP; (iii) HTD; and (iv) HT-FBR using Hermite polynomials. The CP rank corresponds to the lowest RMSE value.

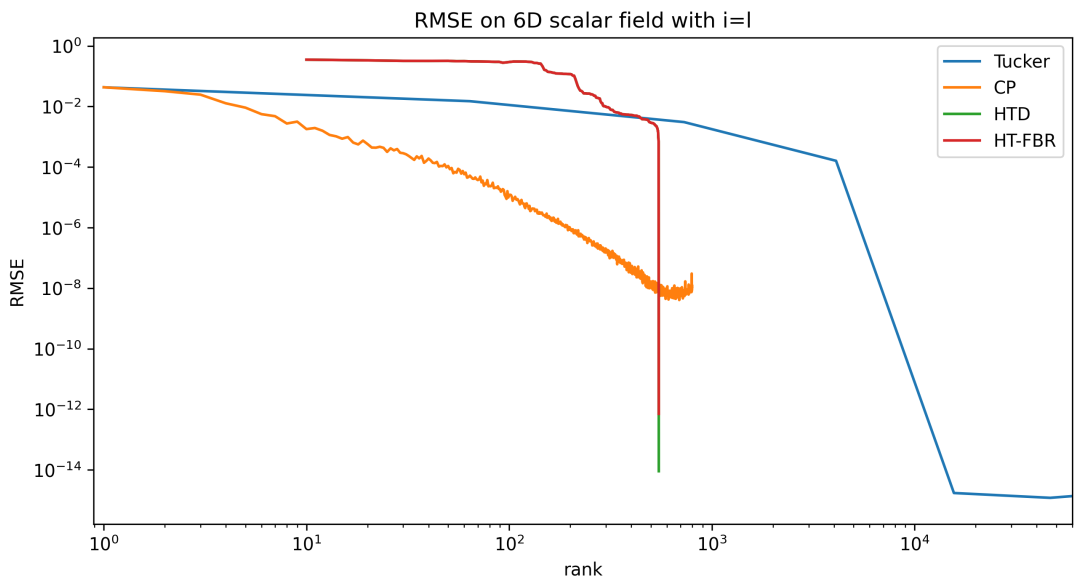

Figure 2: Comparison of the decrease (logarithmic scale) of the RMSE with the total rank for: (1) HOOI/Tucker (blue/middle curve); (2) CP (orange/lowest curve); (3) HTD and (4) HT-FBR (upper curves, superimposed) using our 6D scalar field as benchmark. We use same rank for all dimensions of HOOI, in the figure plots the multiplication of them while CP rank and HTD/HT-FBR using their exact total rank.

RMSE of recontructed tensor DIMS expansion with different polynomial degree			
grid i	k_t of dim (node) 0-4	k_t of dim (node) 5	RMSE
s	4	10	1.0222e-4
s (HTD)	None	None	3.2368e-2
l	4	8	1.3655e-4
l	4	15	1.1688e-4

Table 4: Comparison of the RMSE with full rank for HT-FBR reconstructions of the $A_s^\dagger$ and $A_l^\dagger$ tensors using the HT-FBR parameters from the HTD of the original A_{scal}. It should be highlighted that the number of elements are: $3.75{\cdot}10^4$, $>2.88{\cdot}10^5$ and $2.4{\cdot}10^6$ for A_{scal}, $A_s^\dagger$ and $A_l^\dagger$, respectively. Note that we have used the same polynomial degree for node $\{0\}$ to node $\{4\}$ since they have the same size of U_t.

As final result, in Table 4 we present a comparison of the quality (RMSE) of different HT-FBR interpolations. They all stem from the HTD of A_{scal} using full rank (300, 60, 12). The corresponding HT-FBR has been used for the reconstruction of fine grid tensors $A_s^\dagger$ and $A_l^\dagger$. As it can be observed, the quality of our HT-FBR expansion is virtually independent of the size of the targeted grid. Errors in both grids amount to 1e-4, a value that compares well with previous methods (see Table IV in Ref. [7]).

5 Conclusions

We present a simple method to obtain the analytical binary tree representation of a d-dimensional tensor. Our method, HT-FBR is the analytical counterpart of HTD and, as such , we consider binary trees of R nodes with node rank r_k. Our approach relies on a set of auxiliary functions (typically orthogonal polynomials) which serve to fit the r_k left singular vectors of every child in the tree. The resulting tree data structure is analytical and depends on $(R-1) * r_k * n$ parameters. Note that the root node does not have an associated node rank. HT-FBR is able to reproduce HTD results within a very good agreement at a lower cost. Our HT-FBR expressions are more compact than the original HTD ones. Moreover, the analytical nature of HT-FBR enables the reconstruction (interpolation) to much denser tensor than the original one. HT-FBR belongs to the extended χ-FBR family for which χ = Tucker, and CP. However, the use of tree structures presents several advantages over Tucker or CP. First, in contrast to Tucker, its rank grows polynomically instead of exponentially. Second, similar to Tucker, the analogous of the factor matrices in Tucker, are also orthogonal in HT-FBR (also in HTD). This is very interesting for the algebraic manipulation of the HT-FBR object. And third, the rank we obtain is much closer than that of CP than that of Tucker but without running into the issues of CP optimisation. Finally, our analytical form depends on a small number of parameters thus rendering it suitable for direct optimisation. As future perspectives we will parallelise the HT-FBR structure tasks and then GPU optimise its computation.

Acknowledgements

The authors would like to gratefully acknowledge N. Nadoveza for fruitful discussions on CPD.

References

[1] D. Bacciu and D. P. Mandic. Tensor Decompositions in Deep Learning. *ESANN 2020 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2020. 114109.

[2] R. E. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, Princeton, 1961.

[3] Lars Grasedyck. Hierarchical singular value decomposition of tensors. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2029–2054, 2010.

[4] H.-D. Meyer, U. Manthe, and L. S. Cederbaum. The multi-configurational time-dependent Hartree approach. 165:73–78, 1990.

[5] Natasa Nadoveza, Ramón L. Panadés-Barrueta, Lei Shi, Fabien Gatti, and Daniel Peláez. Analytical high-dimensional operators in canonical polyadic finite basis representation (CP-FBR). *The Journal of Chemical Physics*, 158(11), 03 2023. 114109.

[6] Ramón Panadés-Barrueta, Daniel Peláez. Low-rank sum-of-products finite-basis-representation (sop-fbr) of potential energy surfaces. *The Journal of Chemical Physics*, 153, 11 2020.

[7] Daniel Peláez, Hans-Dieter Meyer. The multigrid potfit (mgpf) method: Grid representations of potentials for quantum dynamics of large systems. *The Journal of chemical physics*, 138:014108, 01 2013.

[8] Falk Richter, Majdi Hochlaf, Pavel Rosmus, Fabien Gatti, Hans-Dieter Meyer. A study of the mode-selective trans–cis isomerization in HONO using ab initio methodology. *The Journal of Chemical Physics*, 120(3):1306–1317, 01 2004.

[9] M. Schröder. Transforming high-dimensional potential energy surfaces into a canonical polyadic decomposition using Monte Carlo methods. 152:024108, 2020.

[10] H. Wang and M. Thoss. Multilayer formulation of the multiconfiguration time-dependent Hartree theory. 119:1289–1299, 2003.