TENSOR CHAIN DECOMPOSITION AND FUNCTION INTERPOLATION

Petr Tichavský[1] *and Anh-Huy Phan*[2]

[1]Institute of Information Theory and Automation,
Academy of Sciences of the Czech Republic, Prague, Czech Republic
[2]Tokyo University of Agriculture and Technology (TUAT), Tokyo, Japan

ABSTRACT

Tensor Chain (TC) decomposition represents a given tensor as a chain (circle) of order-3 tensors (wagons) connected through tensor contractions. In this paper, we show the link between the TC decomposition and a structured Tucker decompositions, and propose a variant of the Krylov-Levenberg-Marquardt optimization, tailored for this problem. Many extensions can be considered, here we only mention decomposition of tensor with missing entries, which enables the tensor completion. Performance of the proposed algorithm is demonstrated on tensor decomposition of the sampled Rosenbrock function. It can be better modeled both as TC and canonical polyadic (CP) decomposition, but with TC, the reconstruction is possible with a lower number of function values.

***Index Terms*—** Multilinear models; canonical polyadic decomposition; tensor train

1. INTRODUCTION

In the most recent three decades, low-rank tensor approximation has been established as a new tool in scientific computing to address large-scale linear and multilinear algebra problems appearing in many applications [1, 2]. The most popular tensor decompositions are the canonical polyadic decomposition (CPD) and the Tucker decomposition, and, more recently, tensor train (TT) [3, 4] and Tensor Chain (TC) [5, 6]. The relationship between CPD and TT is discussed in [7, 8].

The TC is an extension of the earlier tensor train (TT) [3]. It is also known under the name of tensor ring [9]. It was designed to remove problems occurring in TT decomposition with dimensions of the core tensors in the middle of the chain, which need to be quite high sometimes. Connecting the first and the last core tensors in the model makes it more balanced because then there are no first and last core tensors.

Recently, TC decomposition was used as a tool for tensor completion [10] and for compression of convolutive layers in neural networks [11].

[0]The work of P. Tichavský was supported by the Czech Science Foundation through Project No. 22-11101S.

In Section II, we show a relationship between the TC/TT decompositions and a Tucker decomposition. The link has already been mentioned in [12], but not in full generality, which we do apply here.

Both tensor models, TT and TC, are linear functions of their building blocks (core tensors, wagons) one by one, but not jointly. It follows that the TT/TC decomposition can be sought by an Alternating Least Squares (ALS) technique similar to ALS in the CP decomposition. It consists of a series of partial optimization, where all but one wagon are fixed, and optimization with respect to the remaining wagon is found in a closed form, see [5, 13].

As the CP decomposition can apply to incomplete tensors, so the TC decomposition. Once a TC model of the incomplete data is built, it provides an estimate of the missing tensor elements. It is just a way of tensors completion. Various principles of tensor completion are discussed and compared in [16].

The tensor decompositions can be used for a function interpolation, see e.g. [14] and [15]. Multivariate functions can be sampled and treated as tensors. If these tensors can be well approximated by a low-rank decomposition of some kind, e.g., CP, TT, or TC with low bond dimensions, then relatively low number function evaluation (tensor elements) are sufficient to identify the model. For some functions, the CP decomposition is useful, but for some other functions TT or TC is better, see, e.g., [24], and the references therein. The decomposition model can induce an accurate interpolation of the function (missing tensor elements).

The main novelty of this paper is the design of of the Krylov-Levenberg-Marquardt algorithm, originally proposed for computing the CP decomposition [18, 21]. We modify the KLM algorithm for the TC decomposition and TC decomposition of incomplete tensors, and show that the TC model is better than CP in the case of important nonlinar functions such as the Rosenbrock function.

This paper is organized as follows. Section 2 explains the connection between the structured Tucker decomposition (STD) and Tensor Chain (TC) Decomposition. Section 3 summarizes the Krylov-Levenberg-Marquardt algorithm. In Section 4, an application of the algorithm in the function interpolation is shown. Section 5 concludes the paper.

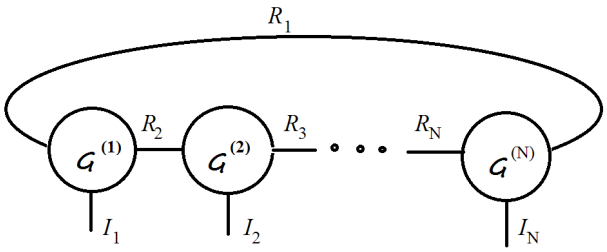

Fig. 1. Illustration of tensor chain.

2. TC AND TT AS CASES OF TUCKER DECOMPOSITION

Assume that we are given an order-N tensor $\mathcal{T}$ of the size $I_1 \times I_2 \times \ldots \times I_N$ with elements $T(i_1, \ldots, i_N)$. The Tucker decomposition of the tensor of the multilinear rank $(R_1, \ldots, R_N)$ is represented by a core tensor $\mathcal{K}$ of the size $R_1 \times \ldots \times R_N$ and N factor matrices $\mathbf{A}^{(n)}$ of the sizes $I_n \times R_n, n = 1, \ldots, N$ such that

$$T(i_1, \ldots, i_N) = \sum_{r_1, \ldots, r_N} K(r_1, \ldots, r_N) A^{(1)}_{i_1, r_1} \ldots A^{(N)}_{i_N, r_N} \tag{1}$$

Symbolically, we shall write

$$\mathcal{T} = [[\mathcal{K}; \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]] \, . \tag{2}$$

The tensor chain decomposition is determined as a contraction of N order-3 tensors $\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(N)}$, where $\mathcal{G}^{(n)}$ has the size $R_n \times I_n \times R_{n+1}$, $n = 1, \ldots, N$, and $R_{N+1} = R_1$, see Fig. 1. Here, there are N free indices, and the contractions, denoted by the connecting lines, representing summations over auxiliary (bond) indices $r_n = 1, \ldots, R_n$, $n = 1, \ldots, N$. In other words, elements of the tensor are given as

$$\begin{aligned} T(i_1, \ldots, i_N) = & \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \ldots \sum_{r_N=1}^{R_N} \mathcal{G}^{(1)}(r_1, i_1, r_2) \\ & \cdot \mathcal{G}^{(2)}(r_2, i_2, r_3) \ldots \mathcal{G}^{(N)}(r_N, i_N, r_1) \end{aligned} \tag{3}$$

The equation (3) can be also written as

$$T(i_1, \ldots, i_N) = \text{tr}(\mathbf{G}^{(1)}_{i_1} \mathbf{G}^{(2)}_{i_2} \ldots \mathbf{G}^{(N)}_{i_N}) \tag{4}$$

where $\mathbf{G}^{(n)}_{i_n} = G^{(n)}(:, i_n, :)$ is the i_n-th lateral slice of $\mathcal{G}^{(n)}$, $n = 1, \ldots, N$, having the size $R_n \times R_{n+1}$. Symbolically we shall write

$$\mathcal{T} = \{\{\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(N)}\}\} \, . \tag{5}$$

The tensor train [3] can be viewed as a special case of the tensor chain with $R_1 = R_{N+1} = 1$.

It was shown in [12] that the tensor chain model (5) can be viewed as a special case of the Tucker model (1). To see this fact in full generality, consider a multi-linear function ϕ of N real-valued matrices $\mathbf{X}_1, \ldots, \mathbf{X}_N$, where $\mathbf{X}_n$ has the size $R_n \times R_{n+1}$, $n = 1, \ldots, N$ such that

$$\phi(\mathbf{X}_1, \ldots, \mathbf{X}_N) = \text{tr}(\mathbf{X}_1 \mathbf{X}_2 \ldots \mathbf{X}_N) \, . \tag{6}$$

It is the trace of the product of the matrices, similar to (4). The function is linear in each of its arguments, and its output is scalar. Such a function is represented by a tensor, $\mathcal{M}$, of the size $R_1 R_2 \times R_2 R_3 \times \ldots \times R_N R_1$ such that

$$\begin{aligned} & \phi(\mathbf{X}_1, \ldots, \mathbf{X}_N) \\ & = \sum_{i_1, \ldots, i_N} M(i_1, \ldots, i_N) (\text{vec}\, \mathbf{X}_1)_{i_1} \ldots (\text{vec}\, \mathbf{X}_N)_{i_N} \end{aligned} \tag{7}$$

for all $\mathbf{X}_1, \ldots, \mathbf{X}_N$, where $M(i_1, \ldots, i_N)$ is the element of $\mathcal{M}$ in the position $(i_1, \ldots, i_N)$. The summation in (7) proceeds through all $i_n = 1, \ldots, I_n, n = 1, \ldots, N$. It can easily be seen that the elements of $\mathcal{M}$ are only zeros and ones.

Once we know the core tensor $\mathcal{M}$, it is possible to design a KLM for the tensor chain decomposition through the structured Tucker model with a fixed core tensor [12]. In this paper, however, we propose a much simpler way, without even evaluating the core tensor $\mathcal{M}$, because $\mathcal{M}$ is usually large in size.

3. KRYLOV-LEVENBERG-MARQUARDT ALGORITHM

The Levenberg-Marquardt algorithm to minimize the Frobenius (L2) norm of the fitting error consists of iterations

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}' = \boldsymbol{\theta} - (\mathbf{H} + \mu \mathbf{I})^{-1} \mathbf{g} \tag{8}$$

where Jacobi matrix $\mathbf{J}$, an error gradient $\mathbf{g}$, and an approximate Hessian $\mathbf{H}$ are defined as

$$\mathbf{J} = \frac{\partial \text{vec}(\mathcal{T})}{\partial \boldsymbol{\theta}} \tag{9}$$

$$\mathbf{g} = \mathbf{J}^T \text{vec}(\mathcal{T} - \hat{\mathcal{T}}) \tag{10}$$

$$\mathbf{H} = \mathbf{J}^T \mathbf{J} \tag{11}$$

μ is a damping parameter that is updated through the iterations, see [20] for more details.

In the Krylov-Levenberg-Marquardt algorithm, the expression $(\mathbf{H} + \mu \mathbf{I})^{-1} \mathbf{g}$ is replaced by its approximation

$$(\mathbf{H} + \mu \mathbf{I})^{-1} \mathbf{g} \approx \frac{1}{\mu} \mathbf{g} - \frac{1}{\mu} \mathbf{U} (\mu \mathbf{S}^{-1} + \mathbf{U}^T \mathbf{U})^{-1} (\mathbf{U}^T \mathbf{g}) \tag{12}$$

where columns of $\mathbf{U}$ form an orthogonal basis of the so-called Krylov subspace, which is the linear hull of

$$[\mathbf{g}, \mathbf{H}\mathbf{g}, \mathbf{H}^2 \mathbf{g}, \ldots, \mathbf{H}^{M-1} \mathbf{g}] \tag{13}$$

and $\mathbf{S} = \mathbf{U}^T \mathbf{H} \mathbf{U}$. The integer M is a design parameter controlling accuracy of the approximation. The matrix $\mathbf{U}$

is obtained through a Gramm-Schmidt orthogonalization process, and $\mathbf{S}$ is received as a side product of this process. See [18, 21] for more details.

The total complexity of computing (12) depends on the complexity of the products $\mathbf{Hx}$ and on the order of the approximation M. If the complexity of the product $\mathbf{Hx}$ is C flops, and θ has N_p elements, then the complexity of the update (12) is $O(MC + M^2N_p + M^3)$, see [12] for details; but if M is not large, then the complexity is about $O(MC)$.

It now remains to explain the fast implementation of computing the product $\mathbf{Hx}$ without even evaluating the Hessian $\mathbf{H}$. The arbitrary vector $\mathbf{x}$ of the appropriate length can be written as composed of N parts,

$$\mathbf{x} = [\mathrm{vec}\,\mathbf{X}_1; \ldots; \mathrm{vec}\,\mathbf{X}_N] \tag{14}$$

where $\mathbf{X}_n$ is a matrix of the shape of $\mathbf{G}^{(n)}$, i.e., $I_n \times (R_n R_{n+1})$ for $n = 1, \ldots, N$. Similarly, the outcome $\mathbf{y} = \mathbf{Hx}$ shares the same structure,

$$\mathbf{y} = [\mathrm{vec}\,\mathbf{Y}_1; \ldots; \mathrm{vec}\,\mathbf{Y}_N] \tag{15}$$

where the matrices $\mathbf{Y}_n$ have the same sizes as $\mathbf{X}_n$, $n = 1, \ldots, N$. Now, it can be shown that

$$\mathbf{Y}_n = \mathbf{Z}_{(n)}\mathbf{U}_n^T \tag{16}$$

where $\mathbf{Z}_{(n)}$ is the mode-n matrization of the tensor

$$\begin{aligned} \mathcal{Z} \;=\; & \{\{\mathcal{X}_1, \mathcal{G}^{(2)}, \ldots, \mathcal{G}^{(N)}\}\} \\ & + \{\{\mathcal{G}^{(1)}, \mathcal{X}_2, \mathcal{G}^{(3)}, \ldots, \mathcal{G}^{(N)}\}\} \\ & + \ldots + \{\{\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(N-1)}, \mathcal{X}_N\}\}\,. \end{aligned} \tag{17}$$

The tensor $\mathcal{Z}$ is written as a sum of N tensor chains. It follows that the product (16) can be computed through N tensor contractions without actually evaluating the tensor, see Fig. 2. The result (17) is obtained by considering TC decomposition as a structured Tucker decomposition, see the analysis in [12].

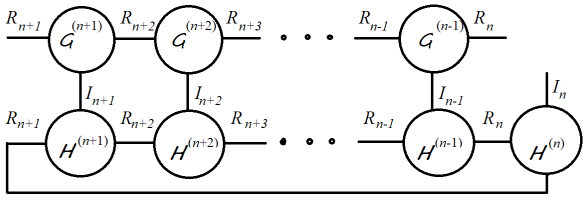

Fig. 2. Computing the product $\mathbf{Z}_{(n)}\mathbf{U}_n^T$ through tensor contractions. Here, the wagons $\mathcal{H}^{(n)}$ are equal to $\mathcal{G}^{(n)}$ up to one in position m which is replaced by $\mathcal{X}_m$, $m = 1, \ldots, N$.

We can see that computation of neither the product $\mathbf{y} = \mathbf{Hx}$ does not require any large tensors to be stored in the computer memory. The complexity of the product $\mathbf{Hx}$ then equals $C = O(N^2R^4I)$.

Finally, the error gradient $\mathbf{g}$ can be computed as

$$\mathbf{g} = [\mathbf{g}_1; \ldots; \mathbf{g}_N] \tag{18}$$

with blocks

$$\mathbf{g}_n \;=\; \mathrm{vec}\,[\mathbf{E}_{(n)}\mathbf{U}_n^T]\,. \tag{19}$$

where $\mathbf{E}_{(n)}$ is the mode-n matricization of the tensor $\mathcal{E}$, which is the difference between the given tensor and the model,

$$\mathcal{E} \;=\; \mathcal{T} - \{\{\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(N)}\}\}\,. \tag{20}$$

Assume now that we are given a weight tensor $\mathcal{W}$ which should have the same size as the data, $\mathcal{T}$, and is filled with nonnegative elements. The criterion to be minimized is changed to

$$\varepsilon_{\mathcal{W}}(\mathcal{T}, \boldsymbol{\theta}) = \|\mathcal{W}^{1/2} * (\mathcal{T} - \{\{\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(N)}\}\})\|_F^2\,.$$

Here, the square root is taken elementwise, and "*" denotes the Hadamard (elementwise) product.

Ideally, in probabilistic setting, the weight of each tensor element should be inversely proportional to variance of the element. In the case of tensor with missing elements, the weight tensor should have zeros in the corresponding positions.

As in [12], it can be easily seen that the KLM algorithm can be modified for the use in the weighted scenario. Let $\mathbf{W}$ denote a diagonal matrix containing all weight tensor elements along its main diagonal. The equations (10), (11), (20) and (17) are replaced with

$$\mathbf{g} \;=\; \mathbf{J}^T\mathbf{W}\mathrm{vec}(\mathcal{E}) = \mathbf{J}^T\mathrm{vec}(\mathcal{W} * \mathcal{E}) \tag{21}$$

$$\mathbf{H} \;=\; \mathbf{J}^T\mathbf{W}\mathbf{J} \tag{22}$$

$$\mathcal{E} \;=\; \mathcal{W} * (\mathcal{T} - \{\{\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(N)}\}\}) \tag{23}$$

$$\begin{aligned} \mathcal{Z} \;=\; & \mathcal{W} * (\{\{\mathcal{X}_1, \mathcal{G}^{(2)}, \ldots, \mathcal{G}^{(N)}\}\} \\ & + \{\{\mathcal{G}^{(1)}, \mathcal{X}_2, \mathcal{G}^{(3)}, \ldots, \mathcal{G}^{(N)}\}\} \\ & + \ldots + \{\{\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(N-1)}, \mathcal{X}_N\}\})\,. \end{aligned} \tag{24}$$

The other equations relate to the algorithm remain unchanged. The only difference is that one cannot use the tensor contractions to compute the product $\mathbf{Z}_{(n)}\mathbf{U}_n^T$, because the weighted tensor $\mathcal{Z}$ is not of the simple form any more. The weighted KLM algorithm will be thus somewhat slower than the unweighted one.

4. FUNCTION INTERPOLATION

Some multivariate functions can be well approximated by a low-rank decomposition TT, or TC with low bond dimensions, see [22]. One example of such function is a Rosenbrock function [23]. It is a non-convex function used as a performance test problem for optimization algorithms introduced by Howard H. Rosenbrock in 1960.

The simplest case of the Rosenbrock function is two-dimensional,

$$f_2(x,y) = (a-x)^2 + b(y-x^2)^2 \tag{25}$$

where a, b are constants, typically $a = 1$ and $b = 100$. Let this function be sampled in some rectangle it can be easily verified that the rank of the resultant matrix is always 3. The proof is easy: the function can be written as a sum of three terms, and each of them has, after the sampling, rank one:

$$f_2(x,y) = \underbrace{(a-x)^2 + bx^4}_{term1} \underbrace{- 2byx^2}_{term2} + \underbrace{by^2}_{term3} . \tag{26}$$

We can see that arbitrary three columns and three rows of the matrix are sufficient to interpolate the whole matrix. In dimension two, CP decomposition is equivalent to TT, TC, or ordinary SVD decomposition. The rank is 3.

A multidimensional generalization of the Rosenbrock function is defined as [23]

$$f_N(x_1,\dots,x_N) = \sum_{i=1}^{N-1} b(x_{i+1} - x_i^2)^2 + (a-x_i)^2 . \tag{27}$$

The function can be rewritten as a sum of $(2N-1)$ rank-one terms: There are N terms, each a function of a single variable, and $(N-1)$ products $2bx_{i+1}x_i^2$. It follows that the CP decomposition of the sampled tensor has rank of $(2N-1)$. The function can be interpolated through incomplete CP model with this rank.

We think that the tensor can be decomposed as a TT with bond dimension at most $(1, 3, \dots, 3)$. We can prove this conjecture for $N = 3$ and $N = 4$. For example, for $N = 3$, it holds

$$f_3(x,y,z) = \mathbf{A}(x)\mathbf{B}(y)\mathbf{C}(z) \tag{28}$$

where $\mathbf{A}(x) = [bx^4 - 2ax, x^2, 1]$, $\mathbf{C}(z) = [z^2, z, 1]^T$, and

$$\mathbf{B}(y) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1-2by \\ b & -2by^2 & by^4 + (b+1)y^2 + 2a(a-y) \end{bmatrix} .$$

The conjecture follows. The bond dimension is $(1, 3, 3)$. (A similar decomposition exists for $N = 4$.) A TC decomposition of the tensor is also possible. For $N = 3$, we can have decomposition $f_3(x,y,z) = \mathrm{tr}[\mathbf{A}(x)\mathbf{B}(y)\mathbf{C}(z)]$ where $\mathbf{A}(x), \mathbf{B}(y), \mathbf{C}(z)$ are suitable univariate matrix functions of dimensions 2×2, 2×3 and 3×2, respectively. It means that the bond dimensions are $(2, 3, 2)$. We show that the TT and TC decompositions are easier to fit than CP in the case of incomplete tensor.

In our numerical experiment, we sample the Rosenbrock function of $N = 3$ variables in the grid $\langle -2, 2\rangle \times \langle -2, 2\rangle \times \langle -1, 3\rangle$ with the step 0.05, to create a tensor $\mathcal{T}$ of the size $81 \times 81 \times 81$. First, we decomposed the full tensor with no added noise. The learning curves of ALS and KLM with various Krylov orders M are shown in Fig. 3. We can see the superlinear convergence of KLM in contrast to the ALS. Also, we can see how the Krylov order M influences the rate of the convergence.

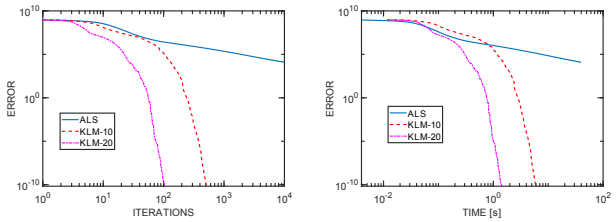

Fig. 3. Learning curves for modeling sampled Rosenbrock function through TC model with bond dimensions $(2, 3, 2)$.

Next, we aim to reconstruct the tensor using randomly chosen $p\%$ tensor elements that are, in addition, corrupted by additive white Gaussian noise with zero mean and variance 0.0001. The reconstruction is done by incomplete TC modeling with bond dimensions $(2, 3, 2)$, $(1, 3, 3)$, $(3, 3, 3)$, and CP model with rank $R = 5$, both with weighted KLM algorithms and $M = 20$. The average computational times of the TCs and CP (Tensorlab [17] with 10 random initializations) were 25s, 27s, 58s and 1.0s$\times$10, respectively. The resultant reconstruction error (median of 70 independent trials) is plotted in Fig.4. We can see that the best reconstruction is obtained by the TC model with the bond dimensions $(1, 3, 3)$ and $(2, 3, 2)$.

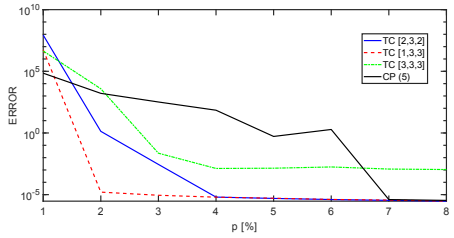

Fig. 4. Reconstruction error on the missing values of the order-3 Rosenbrock function versus percentage of the observed entries.

We can see that simpler models (TCs) can reconstruct the function from less measurements/observations than more complex models (CP). The reconstruction is not much sensitive to the rank selection.

5. CONCLUSIONS

We have presented improved algorithms for the tensor chain modeling. The TC model can be used a function interpolation, similarly to CP based interpolation. For some functions, the TC model is more suitable than CP. The matlab codes of the respective algorithms and the data sets are available on the Internet at https://github.com/Tichavsky/tensor-decomposition.

6. REFERENCES

[1] N.D. Sidiropoulos, L. De Lathauwer, X. Fu, K Huang, E.E Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning" *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551-3582, 2017.

[2] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, Sep. 2009.

[3] I.V. Oseledets and E.E. Tyrtyshnikov, "Breaking the curse of dimensionality, or how to use SVD in many dimensions," *SIAM Journal on Scientific Computing*, vol. 31, no. 5, pp. 3744–3759, 2009.

[4] I. V. Oseledets, "Tensor-train decomposition, *SIAM J. Sci. Comput.*, vol. 33, no. 5, pp. 2295–2317, Jan. 2011.

[5] M. Espig, W. Hackbusch, S. Handschuh, and R. Schneider, "Optimization problems in contracted tensor networks," *Computing and Visualization in Science*, vol. 14, no. 6, pp. 271–285, 2011.

[6] B.N. Khoromskij, "O(d log N)-quantics approximation of N-d tensors in high-dimensional numerical modeling," *Constructive Approximation*, vol. 34, no. 2, pp. 257–280, 2011.

[7] Y. Zniyed, R. Boyer, A. L.F. de Almeida, and G. Favier, "High-order tensor estimation via trains of coupled third-order CP and Tucker decompositions", *Linear Algebra and its Applications*, Vol. 588 (2020), pp. 304–337.

[8] A.H. Phan, and A. Cichocki, A. Uschmajew, P. Tichavský, G. Luta, and D.P. Mandic, "Tensor Networks for Latent Variable Analysis: Novel Algorithms for Tensor Train Approximation", *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, No. 11, pp. 4622-4636, November 2020.

[9] Q. Zhao, G. Zhou, S. Xie, L. Zhang, and A. Cichocki. Tensor ring decomposition. arXiv preprint arXiv:1606.05535, 2016.

[10] L. Yuan, C. Li, D. Mandic, J. Cao, and Q. Zhao, "Tensor ring decomposition with rank minimization on latent space: An efficient approach for tensor completion." *Proc. of the AAAI Conference on Artificial Intelligence*, Vol. 33, No. 1, pp. 9151–9158, 2019.

[11] W. Wang, Y. Sun, B. Eriksson, W. Wang, and V. Aggarwal, "Wide compression: Tensor ring nets." *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9329–9338, 2018.

[12] P. Tichavský, A.H. Phan, and A. Cichocki, "Krylov-Levenberg-Marquardt Algorithm for Structured Tucker Tensor Decompositions" *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no.3, pp. 550-559, April 2021.

[13] S. Handschuh, *Numerical Methods in Tensor Networks*, Ph.D. thesis, Faculty of Mathematics and Informatics, University Leipzig, Germany, Leipzig, Germany, 2015.

[14] N. Kargas and N.D. Sidiropoulos, "Nonlinear system identification via tensor completion", *Proceedings of AAAI 2020*, pp. 4420–4427.

[15] N. Kargas, N.D. Sidiropoulos, "Supervised learning and canonical decomposition of multivariate functions", *IEEE Transactions on Signal Processing* vol. 69, pp. 1097–1107, 2021.

[16] A. Sobral, E. Zahzah, "Matrix and Tensor Completion Algorithms for Background Model Initialization: A Comparative Evaluation", *Pattern Recognition Letters*, vol. 96, pp. 22-–33, 2017.

[17] N. Vervliet, O. Debals, et al., "Tensorlab 3.0", Available online at https://www.tensorlab.net, Mar. 2016

[18] P. Tichavský, A.H. Phan, and A. Cichocki, "Sensitivity in Tensor Decomposition" *IEEE Signal Processing Letters*, vol.26, no.11, pp. 1653–1657, November 2019.

[19] Q. Shi, Y.-M Cheung, and H. Lu, "Feature Extraction for Incomplete Data Via Low-Rank Tensor Decomposition With Feature Regularization", *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, Vol. 30, no.6, pp. 1803–1817, June 2019.

[20] J. Nocedal, S.J. Wright, *Numerical Optimization*, Springer 2006.

[21] P. Tichavský, A.H. Phan, and A. Cichocki, "Weighted Krylov-Levenberg-Marquardt method for canonical polyadic tensor decomposition", *IEEE ICASSP 2020*, Barcelona, Spain, pp. 3917–3921.

[22] K. Sozykin, A. Chertkov, R. Schutski, A.-H. Phan, A. Cichocki, and I. Oseledets, "TTOpt: A Maximum Volume Quantized Tensor Train-based Optimization and its Application to Reinforcement Learning", https://arxiv.org/abs/2205.00293.

[23] H.H. Rosenbrock, "An automatic method for finding the greatest or least value of a function". *The Computer Journal.*, Vol. 3, no. 3, pp. 175-–184, 1960.

[24] X.P. Li, Z-Y. Wang, Z-L. Shi, H.C. So, and N.D. Sidiropoulos, "Robust Tensor Completion via Capped Frobenius Norm", *IEEE Tr. Neural Networks and Learning Systems* 2023, DOI 10.1109/TNNLS.2023.3236415.