RECENT NUMERICAL AND CONCEPTUAL ADVANCES FOR TENSOR DECOMPOSITIONS — A PREVIEW OF TENSORLAB 4.0

Nico Vervliet[*]*, Michiel Vandecappelle*[*†]*, Martijn Boussé*[*]*, Rob Zink*[*]*, and Lieven De Lathauwer*[*†]

[*]KU Leuven, Dept. of Elec. Eng. (ESAT), Stadius, Kasteelpark Arenberg 10, B-3001 Leuven
[†]KU Leuven Kulak, Group Science, Engineering and Technology, E. Sabbelaan 53, B-8500 Kortrijk

ABSTRACT

The fourth release of Tensorlab — a Matlab toolbox which bundles state-of-the-art tensor algorithms and tools — introduces a number of algorithms which allow a variety of new types of problems to be solved. For example, Gauss–Newton type algorithms for dealing with non-identical noise distributions or implicitly given tensors are discussed. To deal with large-scale datasets, incomplete tensors are combined with constraints, and updating techniques enable real-time tracking of time-varying tensors. A more robust algorithm for computing the decomposition in block terms is presented as well. To make tensor algorithms more accessible, graphical user interfaces for computing a decomposition in rank-1 terms or to compress a tensor are given.

***Index Terms*—** tensor, decomposition, CPD, BTD, optimization

1. INTRODUCTION

Tensor decompositions are key tools in a variety of applications in signal processing, data analysis, machine learning, scientific computing, computational chemistry and so on. Decompositions such as the multilinear singular value decomposition (MLSVD) and tensor trains (TT) are often used to compress data or to find dominant subspaces, while the canonical polyadic decomposition (CPD) and the block term decomposition (BTD) allow interesting rank-1 or low multilinear rank components to be found thanks to mild uniqueness conditions. Computing these decompositions efficiently for a variety of tensors (large-scale, dense, sparse, incomplete, structured, ...) is often challenging, especially when one wants to incorporate prior knowledge in the form of constraints. See [1, 2, 3] for recent overviews.

In this paper, we mainly focus on optimization-based algorithms to compute a decomposition $\mathcal{M}(\mathbf{z})$ of a tensor $\mathcal{T}$: one usually minimizes the residual in a least squares sense, i.e., the variables $\mathbf{z}$ are computed such that

$$\min_{\mathbf{z}} \frac{1}{2} ||\mathcal{M}(\mathbf{z}) - \mathcal{T}||_{\mathrm{F}}^2 . \tag{1}$$

For example, in the case of a CPD of a third-order tensor with factors $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, $\mathbf{z} = \left[\mathrm{vec}\,(\mathbf{A})^{\mathrm{T}}, \mathrm{vec}\,(\mathbf{B})^{\mathrm{T}}, \mathrm{vec}\,(\mathbf{C})^{\mathrm{T}}\right]^{\mathrm{T}}$ and $\mathcal{M}(\mathbf{z}) = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$. Block coordinate descent type algorithms such as alternating least squares (ALS) have been popular to solve (1) thanks to their straightforward implementation. However, more advanced algorithms that update all variables at once typically require fewer iterations, are more robust and have proven convergence properties. Moreover, they are not necessarily much harder to implement [3].

Tensorlab is built upon the complex optimization toolbox [4], which properly deals with both real and complex $\mathbf{z}$ and contains many optimization routines. For tensor decompositions, we focus mainly on (inexact) Gauss–Newton (GN) type algorithms that are numerically well-founded. More technically, in every iteration the system

$$\mathbf{J}^{\mathrm{H}}\mathbf{J}\mathbf{p} = -\mathbf{J}^{\mathrm{H}}\mathbf{r} \tag{2}$$

is solved for the step $\mathbf{p}$. The Jacobian $\mathbf{J}$ contains the derivative of every residual entry in $\mathbf{r} = \mathrm{vec}\,(\mathcal{M}(\mathbf{z}) - \mathcal{T})$ w.r.t. every variable $\mathbf{z}$, and $\mathbf{J}^{\mathrm{H}}\mathbf{r}$ is the gradient of the objective function in (1). The GN step $\mathbf{p}$ is then used together with the gradient to update the estimate $\mathbf{z}_k$ using a trust region approach. For the computation of the CPD and the BTD, the Gramian of the Jacobian $\mathbf{J}^{\mathrm{H}}\mathbf{J}$ is a good approximation of the Hessian near an optimum, allowing (local) quadratic convergence [5]. Moreover, the multilinear structure can be exploited elegantly to allow efficient computations [5].

The first release of Tensorlab in 2013 provided algorithms and tools for basic tensor decompositions and operations. Tensorlab 2.0 introduced the structured data fusion (SDF) framework allowing multiple datasets to be analyzed jointly as well as the incorporation of parametric constraints

N. Vervliet is supported by Internal Funds KU Leuven (PDM/18/146). M. Vandecappelle is supported by an SB Grant from the Research Foundation — Vlaanderen (FWO). Research furthermore supported by: (1) Flemish Government: This work was supported by the Fonds de la Recherche Scientifique — FNRS and the Fonds Wetenschappelijk Onderzoek — Vlaanderen under EOS Project no 30468160 (SeLMA); (2) EU: The research leading to these results has received funding from the European Research Council under the European Unions Seventh Framework Programme (FP7/2007-2013) / ERC Advanced Grant: BIOTENSORS (no 339804). This paper reflects only the authors' views and the Union is not liable for any use that may be made of the contained information; (3) KU Leuven Internal Funds C16/15/059.

such as orthogonality, nonnegativity, smooth functions and Vandermonde structure [6]. In the third release, various techniques for large-scale tensors have been introduced including randomization [7, 8] and the structured tensor framework for tensors that can be represented with few parameters [9]. Techniques to (de-)tensorize data allow various signal processing assumptions to be modeled [10]. Improved visualization tools have been introduced as well. For an in-depth overview of the philosophy and history behind Tensorlab, we refer to [8].

In this paper, we discuss some of the new numerical methods and concepts that are introduced in Tensorlab 4.0. In Section 2, an algorithm for different assumptions on the noise/error distribution is presented. Updating techniques for online decompositions and improved incomplete tensor support are discussed in Section 3. Then, we elaborate on the concept of implicitly given tensors in Section 4. Finally, a new algorithm for BTDs is discussed and an overview of graphical user interfaces (GUIs) is given in Sections 5 and 6.

2. NON-IDENTICAL OR NON-GAUSSIAN NOISE DISTRIBUTIONS

When using the least-squares cost function (1), one implicitly assumes that noise contributions (or errors) are additive, independent over all tensor entries and come from identical distributions. Moreover, these distributions are normal with zero mean and equal variances, or can be approximated by normal distributions thanks to the central limit theorem. However, these assumptions do not always hold. For example, some sensors may be more accurate than others or some tensor entries may be more important. In both examples, the identical distribution assumption does not hold.

To deal with the case that the variance is not equal for all tensor entries, a weight tensor $\mathcal{W} \geq 0$ can be introduced into the cost function, i.e.,

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \frac{1}{2} ||\mathcal{W} * (\llbracket \mathbf{A}, \mathbf{B}, \mathbf{C} \rrbracket - \mathcal{T})||_{\mathrm{F}}^2 ,$$

in which $*$ is the Hadamard product. Implementing a GN type algorithm for a general, dense $\mathcal{W}$ is computationally (too) demanding. However, the weight tensor can often be approximated by a low-rank tensor, i.e., $\mathcal{W} = \llbracket \mathbf{X}, \mathbf{Y}, \mathbf{Z} \rrbracket$. Gauss–Newton methods can be generalized easily to accommodate this low-rank weight tensor: instead of solving (2), the GN step $\mathbf{p}$ is found by solving

$$\mathbf{J}^{\mathrm{H}}\mathbf{D}\mathbf{J}\mathbf{p} = -\mathbf{J}^{\mathrm{H}}\tilde{\mathbf{r}}.$$

The weight tensor appears (squared) on the diagonal matrix $\mathbf{D}$ and in the weighted residual $\tilde{\mathbf{r}}$. The CPD structure of $\mathcal{W}$ can again be exploited to achieve efficient algorithms. A rank-1 weight tensor is a common special case. Note that this rank-1 case is already a generalization of the unweighted case as the latter corresponds to an all-one weight tensor $\mathcal{W} = \mathbf{1} \otimes \mathbf{1} \otimes \cdots \otimes \mathbf{1}$ in which $\mathbf{1}$ is a vector with all ones and $\otimes$ denotes the outer product.

3. LARGE-SCALE TENSORS

In streaming or continuous monitoring applications, new slices of data regularly become available and an 'online' decomposition is required, i.e., the decomposition should be updated in real time to track the underlying signals. When dealing with large tensors that cannot be stored in memory, a decomposition can be computed loading only few slices at once. In both cases, updating can be used: a previously-computed decomposition is altered to accommodate the new data without recomputing the decomposition from scratch or accessing previously-seen data. ALS-based algorithms have been presented for the CPD [11, 12]. Second-order information can be included almost trivially using the structured tensor framework [9] leading to a highly efficient GN method [13]. Suppose at time t, the decomposition $\mathcal{T}^{(t)} = \llbracket \mathbf{A}, \mathbf{B}, \mathbf{C} \rrbracket$ has been computed and at time $t+1$, a new frontal slice $\mathbf{M}$ arrives. The mode-1 unfolding of the new tensor $\mathcal{T}^{(t+1)}$ is then given by

$$\mathbf{T}_{(1)}^{(t+1)} = \begin{bmatrix} \mathbf{A}(\mathbf{C} \odot \mathbf{B})^{\mathrm{T}} & \mathbf{M} \end{bmatrix},$$

hence the previously-seen data do not appear as such. This CPD-plus-slice structure is exploited when computing the objective function value and gradient, which avoids the explicit construction of $\mathcal{T}^{(t+1)}$ and reduces the storage and computational complexity significantly. In a tracking or streaming context, information from older data may grow stale. To reduce the influence of older data, windowing techniques that lower the weight for old data are introduced. Once the weight is below a threshold, the corresponding rows in $\mathbf{C}$ can be removed to avoid the cost of large factors and to allow 'infinite' streaming problems.

The previous discussion was limited to third-order tensors. However, in Tensorlab 4.0 the CPD of tensors of any order can be updated by adding slices of arbitrary order. The SDF framework is extended to allow updating well. Hence, it is possible to update a joint factorization of multiple tensors and matrices or to impose simple constraints such as nonnegativity. The updating technique can also be combined with weighting techniques from Section 2 [14].

A second large-scale technique involves incomplete tensors: combining tensor decompositions with incomplete tensors provides a good way to deal with the curse of dimensionality [15]. In [16], an improved GN-type algorithm to compute the CPD from an incomplete tensor is proposed. To improve the accuracy or to reduce the number of samples, prior knowledge can be incorporated. In the case of linear constraints on the factors, i.e., $\mathbf{A} = \mathbf{D}\hat{\mathbf{A}}$ with $\mathbf{D}$ a known matrix such as a dictionary or a set of evaluated smooth functions, the computational cost can be reduced significantly as shown in [16].

4. IMPLICITLY GIVEN TENSORS AND MULTILINEAR SYSTEMS

Tensors are usually given explicitly as dense, sparse or incomplete arrays, and many algorithms have appeared to deal with these kinds of tensors. If a tensor is given in a structured format, e.g., as a set of signals that create a Hankel tensor, the explicit construction of the tensor can be avoided to reduce the computational cost [9]. Constructing the tensor explicitly is still possible, though. In other applications, however, the tensor to be decomposed is only available implicitly. For example, the solution of a linear system of equations can admit a (vectorized) low-rank tensor decomposition, i.e.,

$$\mathbf{A}\mathbf{x} = \mathbf{b} \quad \text{with} \quad \mathbf{x} = \text{vec}\left(\left[\!\left[\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(N)}\right]\!\right]\right).$$

In some cases, e.g., if $\mathbf{A}$ has full column rank, it is possible to solve the linear system for $\mathbf{x}$ and then decompose $\mathcal{X} = \text{unvec}(\mathbf{x})$ using standard techniques. However, in many cases, e.g., if the linear system is underdetermined or $\mathbf{A}$ is rank-deficient, $\mathbf{x}$ cannot be determined uniquely. Moreover, explicitly constructing $\mathbf{x}$ is often too expensive in terms of computational cost and memory due the curse of dimensionality. Therefore, (second-order) optimization-based algorithms have been derived in [17]. To achieve an efficient algorithm, the potential structure in $\mathbf{A}$ should be taken into account as well [17]. By exploiting the low-rank structure at the same time as solving the linear system, the number of required equations (rows of $\mathbf{A}$) is reduced significantly. This is useful for compressed sensing applications where $\mathbf{x}$ is sparse in the sense that it can be represented by few parameters.

The rank-1 case, i.e., where $\mathbf{x} = \mathbf{u}^{(N)} \otimes \cdots \otimes \mathbf{u}^{(1)}$ and $\otimes$ denotes the Kronecker product, has also been called a Kronecker product equation (KPE) [18]. This special case can be seen as a generalization of linear systems to multilinear systems, e.g., for $N = 2$, we have

$$\mathcal{A} \cdot_2 \mathbf{u}^{(1)^{\mathrm{T}}} \cdot_3 \mathbf{u}^{(2)^{\mathrm{T}}} = \mathbf{b}. \tag{3}$$

(The mode-n tensor-matrix product is denoted by $\cdot_n$.) This problem appears in, e.g., system identification [17] and information retrieval applications [19].

The KPE formulation can be used in a new type of tensor-based classification by exploiting multilinear relations. Given a tensor of training data $\mathcal{A}$, e.g., a set (vectorized) faces of different persons under varying illumination, a truncated MLSVD $\mathcal{A} = \mathcal{S} \cdot_1 \mathbf{U} \cdot_2 \mathbf{V} \cdot_3 \mathbf{W}$ is computed to learn the principal variations, similar to multilinear principal component analysis. Each mode-1 vector $\mathbf{b}$ (in the example each face) can be written as

$$\mathbf{b} = \mathbf{U}\mathbf{S}(\mathbf{w} \otimes \mathbf{v}),$$

for some $\mathbf{v}$ and $\mathbf{w}$, which is an equivalent formulation for Eq. (3). Given a new, unseen sample (face) $\tilde{\mathbf{b}}$, the KPE is solved to find $\tilde{\mathbf{v}}$ and $\tilde{\mathbf{w}}$ which can be matched against the learned factors $\mathbf{V}$ and $\mathbf{W}$ to, e.g., identify the person and the illumination condition. The advantages of this approach are the interpretable multilinear map and the significant reduction of the number of parameters. Examples for face recognition and irregular heartbeat classification are discussed in [17] and references therein.

5. BLOCK TERM DECOMPOSITION

In signal processing and data analysis applications the CPD is prevalent compared to the BTD. However, the expressive power of a rank-1 term may sometimes be too limited. The BTD writes a tensor as a sum of low multilinear rank terms instead, making it more suitable for such problems. (Note that many papers discussing a BTD actually focus on a special case, i.e., the decomposition in multilinear rank-$(L_r, L_r, 1)$ terms (LL1) [20].) While ALS and GN-type algorithms have been discussed to compute a BTD [21, 22], they are often sensitive to the initial (usually random) guess and computationally intensive.

A special variant of the BTD, the so-called decomposition in multilinear rank $(M_r, N_r, \cdot)$ terms decomposition writes a tensor as

$$\mathcal{T} = \sum_{r=1}^{R} \mathcal{D}^{(r)} \cdot_1 \mathbf{A}^{(r)} \cdot_2 \mathbf{B}^{(r)}.$$

In [23], a new type of algorithm is presented to compute this decomposition: instead of using formulation (1), a matrix consisting of Kronecker products of frontal slices is constructed and it is shown that the kernel of this matrix admits a coupled LL1 decomposition. Both algebraic and optimization-based algorithms are derived. Similar to the CPD case, the result of the algebraic method can be used as an initialization for an optimization step, e.g., for standard BTD solvers [6, 24], to improve the results if noise is present.

6. GRAPHICAL USER INTERFACE

Visualization of tensor data is important for initial data exploration and to assess the quality of a tensor decomposition after it has been computed. Therefore, specialized GUIs are available in Tensorlab. However, for the computation of a decomposition, various (experimental) choices such as the (multilinear) rank, the initialization and the algorithm are required. While these experiments can be done perfectly by writing code, it may not be clear which steps to take and which options to check, especially for less experienced users. To facilitate a good workflow and quick experimentation, new GUIs, plotting functions and error metrics have been developed.

Two main groups can be distinguished. The first group focuses on the computation of a CPD and enforces good practices such as looking at multilinear singular values (see Figure 1), using proper initialization and compression methods,

and comparing various error metrics. The second group concentrates on compressing tensor data using the MLSVD or the LMLRA: the plot tools allow good cutoffs to be estimated from the multilinear singular values and automatically enforce theoretical bounds on multilinear ranks. The results of these GUIs are often starting points for more elaborate investigations. Therefore, automatic code generation is provided to get the user started quickly.

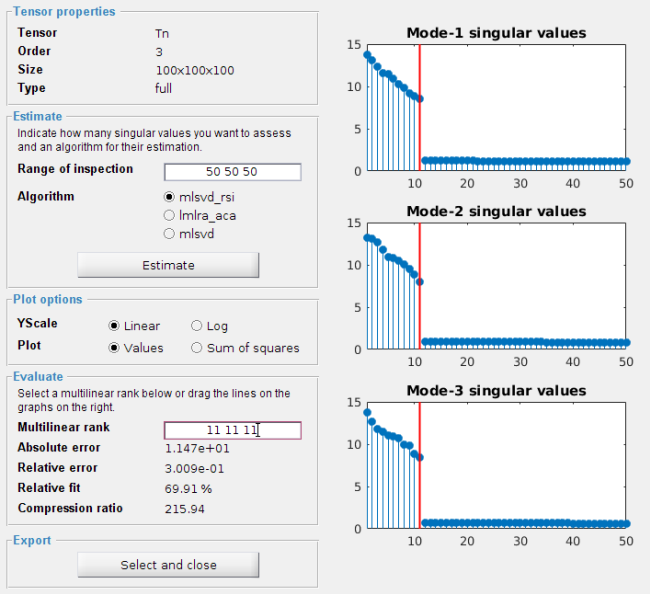


Fig. 1. Using the multilinear singular value inspection GUI, a quick estimate of the multilinear rank can be obtained. A beta-version of the GUI is shown here.

7. CONCLUSION

A brief and high-level overview of some new types of problems that can be solved using Tensorlab 4.0 is given. These problems involve other error distributions and implicitly given tensors. Two CPD algorithms for large-scale tensors are discussed as well as a new method to compute a BTD. To facilitate the proper use of tensor decompositions, new GUIs are introduced as well. Apart from these techniques, other improvements such as the redesign of the computational kernels which allows new types of decompositions and regularization terms to be added to structured data fusion framework, are to be expected as well. For a more elaborate discussion of new techniques outside of Tensorlab, we refer the reader to the recent overview papers.

8. REFERENCES

[1] A. Cichocki, D. Mandic, A.-H. Phan, C. Caiafa, G. Zhou, Q. Zhao, and L. De Lathauwer, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Process. Mag.*, vol. 32, no. 2, pp. 145–163, Mar. 2015.

[2] 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 Trans. Signal Process.*, vol. 65, no. 13, pp. 3551–3582, July 2017.

[3] N. Vervliet and L. De Lathauwer, "Numerical optimization based algorithms for data fusion," in *Data Fusion Methodology and Applications*, M. Cocchi, Ed. Elsevier, 2018, Accepted.

[4] L. Sorber, M. Van Barel, and L. De Lathauwer, "Complex optimization toolbox v1.0," Feb. 2013, Available online at `https://www.esat.kuleuven.be/sista/cot/`.

[5] L. Sorber, M. Van Barel, and L. De Lathauwer, "Optimization-based algorithms for tensor decompositions: Canonical polyadic decomposition, decomposition in rank-$(L_r, L_r, 1)$ terms, and a new generalization," *SIAM J. Optim.*, vol. 23, no. 2, pp. 695–720, Apr. 2013.

[6] L. Sorber, M. Van Barel, and L. De Lathauwer, "Structured data fusion," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 4, pp. 586–600, June 2015.

[7] N. Vervliet and L. De Lathauwer, "A randomized block sampling approach to canonical polyadic decomposition of large-scale tensors," *IEEE J. Sel. Topics Signal Process.*, vol. 10, no. 2, pp. 284–295, Mar. 2016.

[8] N. Vervliet, O. Debals, and L. De Lathauwer, "Tensorlab 3.0 — Numerical optimization strategies for large-scale constrained and coupled matrix/tensor factorization," in *2016 50th Asilomar Conference on Signals, Systems and Computers*, Nov. 2016, pp. 1733–1738.

[9] N. Vervliet, O. Debals, and L. De Lathauwer, "Exploiting efficient representations in tensor decompositions," *SIAM J. Sci. Comput.*, Dec. 2018, Accepted.

[10] O. Debals, *Tensorization and Applications in Blind Source Separation*, Ph.D. thesis, 2017.

[11] D. Nion and N. D. Sidiropoulos, "Adaptive algorithms to track the PARAFAC decomposition of a third-order tensor," *IEEE Trans. Signal Process.*, vol. 57, no. 6, pp. 2299–2310, June 2009.

[12] E. Gujral, R. Pasricha, and E. E. Papalexakis, "SamBaTen: Sampling-based batch incremental tensor decomposition," *Proceedings of the 2018 SIAM Int. Conf. on Data Mining*, p. 387–395, May 2018.

[13] M. Vandecappelle, N. Vervliet, and L. De Lathauwer, "Nonlinear least squares updating of the canonical polyadic decomposition," in *2017 25th European Signal Processing Conference (EUSIPCO17)*, Aug. 2017, pp. 693–697.

[14] M. Vandecapelle, M. Boussé, N. Vervliet, and L. De Lathauwer, "CPD updating using low-rank weights," in *IEEE Proc. of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing*, Aug. 2018, pp. 6368–6372.

[15] N. Vervliet, O. Debals, L. Sorber, and L. De Lathauwer, "Breaking the curse of dimensionality using decompositions of incomplete tensors: Tensor-based scientific computing in big data analysis," *IEEE Signal Process. Mag.*, vol. 31, no. 5, pp. 71–79, Sept. 2014.

[16] N. Vervliet, O. Debals, and L. De Lathauwer, "Canonical polyadic decomposition of incomplete tensors with linearly constrained factors," Technical Report 16–172, ESAT-STADIUS, KU Leuven, Belgium, Apr. 2017.

[17] M. Boussé, N. Vervliet, I. Domanov, O. Debals, and L. De Lathauwer, "Linear systems with a canonical polyadic decomposition constrained solution: Algorithms and applications," *Numer. Linear Algebra Appl.*, vol. 25, no. 6, pp. e2190, Aug. 2018.

[18] M. Boussé, N. Vervliet, O. Debals, and L. De Lathauwer, "Face recognition as a Kronecker product equation," in *2017 IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, Dec. 2017, pp. 276–280.

[19] X. Li and M. K. Ng, "Solving sparse non-negative tensor equations: algorithms and applications," *Frontiers of Mathematics in China*, vol. 10, no. 3, pp. 649–680, May 2014.

[20] L. De Lathauwer, "Decompositions of a higher-order tensor in block terms — Part II: Definitions and uniqueness," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 3, pp. 1033–1066, Sept. 2008.

[21] L. De Lathauwer and D. Nion, "Decompositions of a higher-order tensor in block terms — Part III: Alternating least squares algorithms," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 3, pp. 1067–1083, Jan. 2008.

[22] L. Sorber, M. Van Barel, and L. De Lathauwer, "Structured data fusion," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 4, pp. 586–600, 7 July 2015.

[23] I. Domanov, N. Vervliet, and L. De Lathauwer, "Decomposition of a tensor into multilinear rank-$(M_r, N_r, \cdot)$ terms," Techical Report 18-51, ESAT-STADIUS, KU Leuven, Belgium, Jan. 2019.

[24] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer, "Tensorlab 3.0," Mar. 2016, Available online at `https://www.tensorlab.net`.