Received: 23 March 2018 | Revised: 20 May 2019 | Accepted: 20 June 2019
DOI: 10.1002/nla.2262

RESEARCH ARTICLE

WILEY

Quantized CP approximation and sparse tensor interpolation of function-generated data

Boris N. Khoromskij[1] | **N. Kishore Kumar**[2] | **Jan Schneider**[3]

[1]Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany
[2]Department of Mathematics, Birla Institute of Technology and Science Pilani, Hyderabad Campus, Hyderabad, India
[3]Westsächsische Hochschule Zwickau, Zwickau, Germany

Correspondence
N. Kishore Kumar, Department of Mathematics, Birla Institute of Technology and Science Pilani, Hyderabad Campus, Hyderabad-500 078, India.
Email: thimmaki@gmail.com

Funding information
National Board for Higher Mathematics; Department of Atomic Energy (DAE)

Summary

In this article, we consider the iterative schemes to compute the canonical polyadic (CP) approximation of quantized data generated by a function discretized on a large uniform grid in an interval on the real line. This paper continues the research on the quantics-tensor train (QTT) method ("$O(d \log N)$-quantics approximation of N-d tensors in high-dimensional numerical modeling" in *Constructive Approximation*, 2011) developed for the tensor train (TT) approximation of the quantized images of function related data. In the QTT approach, the target vector of length 2^L is reshaped to a Lth-order tensor with two entries in each mode (quantized representation) and then approximated by the QTT tensor including $2r^2L$ parameters, where r is the maximal TT rank. In what follows, we consider the alternating least squares (ALS) iterative scheme to compute the rank-r CP approximation of the quantized vectors, which requires only $2rL \ll 2^L$ parameters for storage. In the earlier papers ("Tensors-structured numerical methods in scientific computing: survey on recent advances" in *Chemom Intell Lab Syst*, 2012), such a representation was called Q_{Can} format, whereas in this paper, we abbreviate it as the QCP (quantized canonical polyadic) representation. We test the ALS algorithm to calculate the QCP approximation on various functions, and in all cases, we observed the exponential error decay in the QCP rank. The main idea for recovering a discretized function in the rank-r QCP format using the reduced number of the functional samples, calculated only at $O(2rL)$ grid points, is presented. The special version of the ALS scheme for solving the arising minimization problem is described. This approach can be viewed as the sparse QCP-interpolation method that allows to recover all $2rL$ representation parameters of the rank-r QCP tensor. Numerical examples show the efficiency of the QCP-ALS-type iteration and indicate the exponential convergence rate in r.

KEYWORDS

alternating least squares iteration, canonical tensor approximation, CP rank, discretized function, Lth-order tensors, QCP data format, QTT tensor approximation, uniform grid

1 | INTRODUCTION

In many applications, the approximation or integration of functions inheriting the properties of e^{-kx}, e^{-kx^2}, $e^{-k|x|}$, $\sin(kx)$, or $1/|x|^\alpha$ on an interval in $\mathbb{R}$ and functions depending on many parameters lead to the challenging numerical problems.

Often, a very fine grid is required to approximate sharp functions like the Gaussians e^{-kx^2} for large values of k, highly oscillating functions, or functions with multiple local singularities or cusps arising, for example, as the solution of PDEs discretized on fine spatial grid. The storage of the function values and simple arithmetic's operations on data arising from sampling on large grids may easily become nontractable. The additional difficulty arises if each function evaluation has a very high cost, say, related to the solution of large linear system or spectral problem and to solving complicated PDE.

The quantics-tensor train (QTT) tensor approximation method, introduced and analyzed in the work of Khoromskij[1] for some classes of discretized functions, is now a well established technique for data compression of long function-generated vectors. It is based on the low-rank tensor approximation to the quantized image of a vector, where the TT format[2] was applied to the quantized multifold image. Based on the quantization (reshaping) of a long 2^L vector to a Lth-order tensor (quantics), the consequent QTT tensor approximation has been proven to have a low TT rank for a wide class of functional vectors. We refer to the work of Oseledets,[3] where the TT approximation to the reshaped $2^L \times 2^L$ Laplacian-type matrices was considered and analyzed numerically. The QTT tensor parametrization requires $O(2r^2L)$ storage size, where r is the upper bound on the TT rank parameters. Some examples on the successful application of the QTT tensor approximation to the solution of PDEs and in stochastic modeling can be found in other works.[4–15]

This paper continues the research on the QTT tensor approximation method[1] based on the use of TT format. In what follows, we investigate the numerical schemes to compute canonical polyadic (CP) tensor approximation of the quantized tensor. This data format was introduced in the work of Khoromskij[16] under the name Q_{Can} tensor representation. In this paper, we shall abbreviate the notion Q_{Can} as the QCP format. First, we briefly recall the main construction along the line of the QTT approximation. A given vector of size $N = 2^L$ is reshaped (quantized) by successive dyadic folding to a $\underbrace{2 \times 2 \times \dots \times 2}_{L}$ array. The rank r representation of this tensor in the canonical format reduces the number of representation parameters from 2^L down to $2rL$, which is smaller than for the QTT format, characterized by the storage size $O(2r^2L)$. The following simple example shows why the QCP approximation of a vector does a job by reducing the number of representation parameters.

Let $f(x) = e^{-x}$ in $[0, 1]$. Consider the nodes $0, h, 2h, \dots, 15h$ on the interval $[0, 1]$, where h is the step size of the uniform grid. The function values at these discrete points form a vector $\mathbf{q} = [1, q, q^2, \dots, q^{15}]$ of length $N = 2^4$, where $q = e^{-h}$. Now, reshape the vector $\mathbf{q}$ to obtain a fourth-order tensor $\mathbf{Q} \in \mathbb{R}^{2\times2\times2\times2}$ that is the quantized image of $\mathbf{q}$. Following the work of Khoromskij,[1] we recall that the CP rank of this tensor is 1, and the corresponding explicit CP tensor representation reads as

$$\mathbf{Q} = \begin{pmatrix} 1 \\ q \end{pmatrix} \otimes \begin{pmatrix} 1 \\ q^2 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ q^4 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ q^8 \end{pmatrix}.$$

One can see that the whole vector of size $N = 2^4$ is represented only by four parameters, which means the logarithmic complexity scaling $\log_2 N$. In general, the CP rank of the quantized image of a vector of length 2^L generated by $f(x) = e^{-\lambda x}$ is 1 and it is represented by only $\log_2 2^L = L$ parameters, such that its explicit one-term representation is given by[1]

$$\mathbf{Q} = \bigotimes_{p=1}^{L} \begin{pmatrix} 1 \\ q^{2^{p-1}} \end{pmatrix}.$$

We say that the CP rank of the quantized image of the discretized function $e^{-\lambda x}$ is 1. In general, the exact CP rank of rather simple functions like e^{-x^2}, $\sin k\pi x$, $|x|^\alpha$, ... and so forth, is not known. Construction and complexity analysis of alternating least squares (ALS)-type algorithms for computing the QCP approximation of quantized functions is the main aim of this article.

It is worth to note that the rank-r QCP tensor is represented only by small number of parameters, $2Lr$, whereas the QTT format based on rank r TT tensors is parametrized by $O(2Lr^2)$ numbers as it was already mentioned. Based on this observation, we propose the QCP interpolation scheme using only small number of functional calls (of the order of $O(2Lr)$), which recovers the quantized tensor image. This concept leads to the promising enhancement of the QCP approximation of the complete 2^L vector because of the small number of parameters in the arising minimization problem. For the practical implementation, we introduce the ALS-type scheme to compute the sparse QCP interpolant.

In numerical examples, we test the ALS iterative scheme implementing the CP approximation on 15th-order tensors representing 2^{15} vectors generated by various functions. In all cases, we observe the exponentially fast error decay in the QCP rank. Notice that the traditional ALS algorithm for CP tensors and its enhanced versions have been discussed in many articles.[13,17–26] Regularized ALS scheme was considered in other works.[27,28]

Recovering a tensor using only few of its entries is called tensor completion. Tensor completion has been widely studied in the literature. Various approaches like decomposition-based methods[29–34] are available in the literature. Various other approaches can be found in other works.[35–38]

The efficient representation and multilinear algebra of large multidimensional vectors (tensors) can be based on their low-rank tensor approximation by using different tensor formats. We refer to reviews on the multilinear algebra[23,39–42] and to recent surveys and monographs on tensor numerical methods and their application in scientific computing.[16,43–46]

The rest of this paper is organized as follows. Some auxiliary technical results concerning the ALS-canonical algorithm are presented in Section 2. Section 3 describes the particular QCP-ALS scheme, which uses the complete information about the tensor. In Section 4, we calculate the QCP approximation of some selected functions, which appears in various applications. The main idea and basic ALS scheme for computing the QCP approximation by using the information on only few entries of the target vector is described in Section 5, where the numerical illustrations are also presented. The approximation by using incomplete data can be viewed as the sparse QCP interpolation of function-generated vectors. Some useful notations, definitions, and a simple example of the scheme for QCP approximation of a fourth-order tensor are given in the Appendix.

In this article, we often use MATLAB notations, for example, $\mathbf{X} = \text{reshape}(\mathbf{x}, 2, \dots, 2)$. The Frobenius norm of a tensor $\mathbf{X} = [x_{i_1 i_2 \dots i_d}] \in \mathbb{R}^{n_1 \times n_2 \times \dots \times n_d}$ is defined as the square root of the sum of squares of all its elements $x_{i_1 i_2 \dots i_d}$, that is,

$$\|\mathbf{X}\|_F = \sqrt{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_d=1}^{n_d} x^2_{i_1 i_2 \dots i_d}}.$$

2 | TECHNICAL RESULTS FOR ALS

For convenience and better understanding of our notation and some technical results in the upcoming sections, we provide the proofs of some basic results (see the work of Van Loan[26]), which are useful in the construction of the ALS algorithm, and give some simple examples of ALS iteration for canonical approximation of a tensor.

A complete description of each iteration step of ALS for a fourth-order tensor χ is given in Appendix A.3. In Steps 1, 2, 3, and 4 of A.3, we need to obtain products such as $(C \odot B \odot A)^T(C \odot B \odot A)$ and $(C \odot B \odot A)^T \chi(:, :, :, 1)$, where $\odot$ is the Khatri–Rao product of matrices defined in Appendix A.2. An efficient way of obtaining these products is described below. First, we show it for products, which appear in the canonical approximation of a third-order tensor and then generalize it for dth-order tensors.

2.1 | Fast evaluation of $(B \odot C)^T(B \odot C)$

In the ALS method, to get the canonical approximation to a third-order tensor, we need to compute $(B \odot C)^T(B \odot C)$. This usually requires $O(r^2 n_1 n_2) + n_1 n_2 r$ arithmetic operations. Now, we show the efficient way to compute it.

Lemma 1. *If* $B = [\mathbf{b}_1|\mathbf{b}_2| \dots |\mathbf{b}_r] \in \mathbb{R}^{n_1 \times r}$ *and* $C = [\mathbf{c}_1|\mathbf{c}_2| \dots |\mathbf{c}_r] \in \mathbb{R}^{n_2 \times r}$, *then*

$$(B \odot C)^T (B \odot C) = (B^T B)\circ(C^T C),$$

where $\circ$ *denotes the Hadamard product of matrices.*

Proof. As defined in Appendix A.2,

$$B \odot C = [\mathbf{b}_1 \otimes \mathbf{c}_1|\mathbf{b}_2 \otimes \mathbf{c}_2| \dots |\mathbf{b}_r \otimes \mathbf{c}_r] \in \mathbb{R}^{n_1 n_2 \times r}.$$

Therefore,

$$(B \odot C)^T (B \odot C) = \begin{bmatrix} (\mathbf{b}_1 \otimes \mathbf{c}_1)^T \\ (\mathbf{b}_2 \otimes \mathbf{c}_2)^T \\ \vdots \\ (\mathbf{b}_r \otimes \mathbf{c}_r)^T \end{bmatrix}_{r \times n_1 n_2} [\mathbf{b}_1 \otimes \mathbf{c}_1 \,|\, \mathbf{b}_2 \otimes \mathbf{c}_2 \,|\, \dots \,|\, \mathbf{b}_r \otimes \mathbf{c}_r]_{n_1 n_2 \times r}$$

$$= \begin{bmatrix} (\mathbf{b}_1 \otimes \mathbf{c}_1)^T(\mathbf{b}_1 \otimes \mathbf{c}_1) & (\mathbf{b}_1 \otimes \mathbf{c}_1)^T(\mathbf{b}_2 \otimes \mathbf{c}_2) & \dots \dots & (\mathbf{b}_1 \otimes \mathbf{c}_1)^T(\mathbf{b}_r \otimes \mathbf{c}_r) \\ (\mathbf{b}_2 \otimes \mathbf{c}_2)^T(\mathbf{b}_1 \otimes \mathbf{c}_1) & (\mathbf{b}_2 \otimes \mathbf{c}_2)^T(\mathbf{b}_2 \otimes \mathbf{c}_2) & \dots \dots & (\mathbf{b}_2 \otimes \mathbf{c}_2)^T(\mathbf{b}_r \otimes \mathbf{c}_r) \\ \vdots & \vdots & & \vdots \\ (\mathbf{b}_r \otimes \mathbf{c}_r)^T(\mathbf{b}_1 \otimes \mathbf{c}_1) & (\mathbf{b}_r \otimes \mathbf{c}_r)^T(\mathbf{b}_2 \otimes \mathbf{c}_2) & \dots \dots & (\mathbf{b}_r \otimes \mathbf{c}_r)^T(\mathbf{b}_r \otimes \mathbf{c}_r) \end{bmatrix}$$

$$= \begin{bmatrix} \left(\mathbf{b}_1^T \otimes \mathbf{c}_1^T\right)(\mathbf{b}_1 \otimes \mathbf{c}_1) & \left(\mathbf{b}_1^T \otimes \mathbf{c}_1^T\right)(\mathbf{b}_2 \otimes \mathbf{c}_2) & \dots \dots & \left(\mathbf{b}_1^T \otimes \mathbf{c}_1^T\right)(\mathbf{b}_r \otimes \mathbf{c}_r) \\ \left(\mathbf{b}_2^T \otimes \mathbf{c}_2^T\right)(\mathbf{b}_1 \otimes \mathbf{c}_1) & \left(\mathbf{b}_2^T \otimes \mathbf{c}_2^T\right)(\mathbf{b}_2 \otimes \mathbf{c}_2) & \dots \dots & \left(\mathbf{b}_2^T \otimes \mathbf{c}_2^T\right)(\mathbf{b}_r \otimes \mathbf{c}_r) \\ \vdots & \vdots & & \vdots \\ \left(\mathbf{b}_r^T \otimes \mathbf{c}_r^T\right)(\mathbf{b}_1 \otimes \mathbf{c}_1) & \left(\mathbf{b}_r^T \otimes \mathbf{c}_r^T\right)(\mathbf{b}_2 \otimes \mathbf{c}_2) & \dots \dots & \left(\mathbf{b}_r^T \otimes \mathbf{c}_r^T\right)(\mathbf{b}_r \otimes \mathbf{c}_r) \end{bmatrix} \quad \text{(see P1 in } \mathbf{A}.1)$$

$$= \begin{bmatrix} \left(\mathbf{b}_1^T\mathbf{b}_1 \otimes \mathbf{c}_1^T\mathbf{c}_1\right) & \left(\mathbf{b}_1^T\mathbf{b}_2 \otimes \mathbf{c}_1^T\mathbf{c}_2\right) & \dots \dots & \left(\mathbf{b}_1^T\mathbf{b}_r \otimes \mathbf{c}_1^T\mathbf{c}_r\right) \\ \left(\mathbf{b}_2^T\mathbf{b}_1 \otimes \mathbf{c}_2^T\mathbf{c}_1\right) & \left(\mathbf{b}_2^T\mathbf{b}_2 \otimes \mathbf{c}_2^T\mathbf{c}_2\right) & \dots \dots & \left(\mathbf{b}_2^T\mathbf{b}_r \otimes \mathbf{c}_2^T\mathbf{c}_r\right) \\ \vdots & \vdots & & \vdots \\ \left(\mathbf{b}_r^T\mathbf{b}_1 \otimes \mathbf{c}_r^T\mathbf{c}_1\right) & \left(\mathbf{b}_r^T\mathbf{b}_2 \otimes \mathbf{c}_r^T\mathbf{c}_2\right) & \dots \dots & \left(\mathbf{b}_r^T\mathbf{b}_r \otimes \mathbf{c}_r^T\mathbf{c}_r\right) \end{bmatrix}. \quad \text{(see P3 in } \mathbf{A}.1)$$

Because $\mathbf{b}_i^T\mathbf{b}_j$ and $\mathbf{c}_i^T\mathbf{c}_j$ are scalars, $(\mathbf{b}_i^T\mathbf{b}_j) \otimes (\mathbf{c}_i^T\mathbf{c}_j) = (\mathbf{b}_i^T\mathbf{b}_j) \cdot (\mathbf{c}_i^T\mathbf{c}_j)$. Therefore,

$$(B \odot C)^T (B \odot C) = \begin{bmatrix} \mathbf{b}_1^T\mathbf{b}_1 & \mathbf{b}_1^T\mathbf{b}_2 & \dots & \mathbf{b}_1^T\mathbf{b}_r \\ \mathbf{b}_2^T\mathbf{b}_1 & \mathbf{b}_2^T\mathbf{b}_2 & \dots & \mathbf{b}_2^T\mathbf{b}_r \\ \\ \mathbf{b}_r^T\mathbf{b}_1 & \mathbf{b}_r^T\mathbf{b}_2 & \dots & \mathbf{b}_r^T\mathbf{b}_r \end{bmatrix} \circ \begin{bmatrix} \mathbf{c}_1^T\mathbf{c}_1 & \mathbf{c}_1^T\mathbf{c}_2 & \dots & \mathbf{c}_1^T\mathbf{c}_r \\ \mathbf{c}_2^T\mathbf{c}_1 & \mathbf{c}_2^T\mathbf{c}_2 & \dots & \mathbf{c}_2^T\mathbf{c}_r \\ \\ \mathbf{c}_r^T\mathbf{c}_1 & \mathbf{c}_r^T\mathbf{c}_2 & \dots & \mathbf{c}_r^T\mathbf{c}_r \end{bmatrix} = (B^T B) \circ (C^T C).$$

One can easily show that $B^T B$ requires $O(n_1 r^2)$ and $C^T C$ requires $O(n_2 r^2)$ arithmetic operations. Therefore, the computational complexity to compute $(B \odot C)^T (B \odot C)$ is $O((n_1 + n_2) r^2) + r^2$. □

Generalization of Lemma 1

Here, we generalize Lemma 1 to more than two matrices. Let us consider $A_1, A_2, \dots, A_L$ to be matrices of the same size $n \times r$. Then, by recursion, one can easily prove that

$$\begin{aligned} &[A_L \odot .. \odot A_{i+1} \odot A_{i-1} \odot A_{i-2} \odot .. \odot A_1]^T [A_L \odot .. \odot A_{i+1} \odot A_{i-1} \odot A_{i-2} \odot .. \odot A_1] \\ &= [(A_L \odot .. \odot A_{i+1} \odot A_{i-1} \odot A_{i-2} \odot .. \odot A_2) \odot A_1]^T [(A_L \odot .. \odot A_{i+1} \odot A_{i-1} \odot A_{i-2} \odot .. \odot A_2) \odot A_1] \\ &= (A_L \odot .. \odot A_{i+1} \odot A_{i-1} \odot A_{i-2} \odot .. \odot A_2)^T (A_L \odot .. \odot A_{i+1} \odot A_{i-1} \odot A_{i-2} \odot .. \odot A_2) \circ A_1^T A_1 \\ &\quad \vdots \\ &= \left(A_L^T A_L\right) \circ \left(A_{L-1}^T A_{L-1}\right) \circ \dots \circ \left(A_{i+1}^T A_{i+1}\right) \circ \left(A_{i-1}^T A_{i-1}\right) \circ \dots \circ \left(A_2^T A_2\right) \circ \left(A_1^T A_1\right). \end{aligned}$$

The computational complexity to compute the above is $O((L-1)nr^2)$, whereas the direct computation of this product requires $n^{L-1} r + O(r^2 n^{L-1})$. Therefore, this is much faster.

2.2 | Fast evaluation of $(B \odot C)^T \mathbf{x}$

In ALS, we also need to compute $(B \odot C)^T \mathbf{x}$ for $\mathbf{x} \in \mathbb{R}^{n_1 n_2}$. It would require $3 n_1 n_2 r - r$ arithmetic operations including $n_1 n_2 r$ operations for computing $(B \odot C)$. This complexity can be further improved in the following way.

Lemma 2. *If* $B = [\mathbf{b}_1 | \mathbf{b}_2 | \dots | \mathbf{b}_r] \in \mathbb{R}^{n_1 \times r}$ *and* $C = [\mathbf{c}_1 | \mathbf{c}_2 | \dots | \mathbf{c}_r] \in \mathbb{R}^{n_2 \times r}$ *and* $\mathbf{x} \in \mathbb{R}^{n_1 n_2}$, *then*

$$\mathbf{y} = (B \odot C)^T \mathbf{x} = \begin{bmatrix} \mathbf{c}_1^T X \mathbf{b}_1 \\ \mathbf{c}_2^T X \mathbf{b}_2 \\ \vdots \\ \mathbf{c}_r^T X \mathbf{b}_r \end{bmatrix},$$

where $X = reshape(\mathbf{x}, n_2, n_1)$.

Proof. Let $\mathbf{x} = [x_1, x_2, \dots, x_{n_1 n_2}]^T$. Reshape the vector $\mathbf{x}$ as an $n_2 \times n_1$ matrix X

$$X = \begin{bmatrix} x_1 & x_{n_2+1} & \dots & x_{(n_1-1)n_2+1} \\ x_2 & x_{n_2+2} & & x_{(n_1-1)n_2+2} \\ \vdots & \vdots & & \\ x_{n_2} & x_{2n_2} & & x_{n_1 n_2} \end{bmatrix}.$$

$(B \odot C)^T \mathbf{x}$ is given by

$$(B \odot C)^T \mathbf{x} = [\mathbf{b}_1 \otimes \mathbf{c}_1 | \mathbf{b}_2 \otimes \mathbf{c}_2 | \dots | \mathbf{b}_r \otimes \mathbf{c}_r]^T_{n_1 n_2 \times r} \mathbf{x}_{n_1 n_2 \times 1}$$

$$= \begin{bmatrix} (\mathbf{b}_1 \otimes \mathbf{c}_1)^T \\ (\mathbf{b}_2 \otimes \mathbf{c}_2)^T \\ \vdots \\ (\mathbf{b}_r \otimes \mathbf{c}_r)^T \end{bmatrix}_{r \times n_1 n_2} \mathbf{x}_{n_1 n_2 \times 1} = \begin{bmatrix} (\mathbf{b}_1 \otimes \mathbf{c}_1)^T \mathbf{x} \\ (\mathbf{b}_2 \otimes \mathbf{c}_2)^T \mathbf{x} \\ \vdots \\ (\mathbf{b}_r \otimes \mathbf{c}_r)^T \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{c}_1^T X \mathbf{b}_1 \\ \mathbf{c}_2^T X \mathbf{b}_2 \\ \vdots \\ \mathbf{c}_r^T X \mathbf{b}_r \end{bmatrix},$$

where $X =$ reshape$(\mathbf{x}, n_2, n_1)$. In the last step of the above equation, we have used $(\mathbf{b}_i^T \otimes \mathbf{c}_i^T)\mathbf{x} = \mathbf{c}_i^T X \mathbf{b}_i$. This can be shown easily in the following way.

Let $\mathbf{b}_i = [b_{1i}, b_{2i}, \dots, b_{n_1 i}]^T$ and $\mathbf{c}_i = [c_{1i}, c_{2i}, \dots, c_{n_2 i}]^T$. Then,

$$\begin{aligned}
(\mathbf{b}_i \otimes \mathbf{c}_i)^T \mathbf{x} &= \begin{bmatrix} b_{1i}c_{1i} \\ b_{1i}c_{2i} \\ \vdots \\ b_{1i}c_{n_2 i} \\ b_{2i}c_{1i} \\ \vdots \\ b_{n_1 i}c_{n_2 i} \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n_2} \\ x_{n_2+1} \\ \vdots \\ x_{n_2 n_1} \end{bmatrix} \\
&= b_{1i}c_{1i}x_1 + b_{1i}c_{2i}x_2 + \dots \dots.. + b_{1i}c_{n_2 i}x_{n_2} \\
&\quad + b_{2i}c_{1i}x_{n_2+1} + \dots \dots \dots \dots \dots \dots. + b_{2i}c_{n_2 i}x_{2n_2} \\
&\quad \vdots \\
&\quad + b_{n_1 i}c_{1i}x_{n_2(n_1-1)+1} + \dots \dots.. + b_{n_1 i}c_{n_2 i}x_{n_1 n_2} \\
&= c_{1i}\left(b_{1i}x_1 + b_{2i}x_{n_2+1} + \dots.. + b_{n_1 i}x_{n_2(n_1-1)+1}\right) \\
&\quad + c_{2i}\left(b_{1i}x_2 + b_{2i}x_{n_2+2} + ... + b_{n_1 i}x_{n_2(n_1-1)+2}\right) \\
&\quad \vdots \\
&\quad + c_{n_2 i}\left(b_{1i}x_{n_2} + b_{2i}x_{2n_2} + \dots \dots \dots \dots.. + b_{n_1 i}x_{n_2 n_1}\right) \\
&= [c_{1i}, c_{2i}, \dots, c_{n_2 i}] \begin{bmatrix} b_{1i}x_1 + b_{2i}x_{n_2+1} + \dots.. + b_{n_1 i}x_{n_2(n_1-1)+1} \\ b_{1i}x_2 + b_{2i}x_{n_2+2} + \dots.. + b_{n_1 i}x_{n_2(n_1-1)+2} \\ \vdots \\ b_{1i}x_{n_2} + b_{2i}x_{2n_2} + \dots \dots \dots \dots.. + b_{n_1 i}x_{n_2 n_1} \end{bmatrix} \\
&= [c_{1i}, c_{2i}, \dots, c_{n_2 i}] \begin{bmatrix} x_1 & x_{n_2+1} & \dots. & x_{(n_1-1)n_2+1} \\ x_2 & x_{n_2+2} & \dots. & x_{(n_1-1)n_2+2} \\ \vdots & \vdots & \vdots & \\ x_{n_2} & x_{2n_2} & \dots. & x_{n_1 n_2} \end{bmatrix} \begin{bmatrix} b_{1i} \\ b_{2i} \\ \vdots \\ b_{n_1 i} \end{bmatrix} \\
&= \mathbf{c}_i^T X \mathbf{b}_i.
\end{aligned}$$

Here, $n_2(2n_1-1)$ operations are required to compute $X_{n_2 n_1}\mathbf{b}_i$ and $(2n_2-1)$ operations to compute $\mathbf{c}_i^T(X_{n_2 n_1}\mathbf{b}_i)$. Therefore, the overall computational complexity is $2n_1 n_2 r + n_2 r - r$, which is less than the complexity for computing $(B \odot C)^T \mathbf{x}$ directly. □

2.2.1 | Generalization of Lemma 2

Let us look at Lemma 2 in the case of three matrices: $A_4 \in \mathbb{R}^{n_4 \times r}$, $A_3 \in \mathbb{R}^{n_3 \times r}$, and $A_2 \in \mathbb{R}^{n_2 \times r}$. Let $\mathbf{x} \in \mathbb{R}^{n_4 n_3 n_2}$. We look at $(A_4 \odot A_3 \odot A_2)^T \mathbf{x}$. Reshape the vector $\mathbf{x}$ into a matrix of size $n_2 n_3 \times n_4$. Let $X_4 = \text{reshape}(\mathbf{x}, n_2 n_3, n_4)$. Then,

$$(A_4 \odot A_3 \odot A_2)^T \mathbf{x} = \begin{bmatrix} (A_3 \odot A_2)_1^T X_4 (A_4)_1 \\ (A_3 \odot A_2)_2^T X_4 (A_4)_2 \\ \vdots \\ (A_3 \odot A_2)_r^T X_4 (A_4)_r \end{bmatrix}.$$

Here, $(A_4)_i$ is the ith column of A_4 with size $n_4 \times 1$. Therefore, the size of $X_4(A_4)_i$ is $n_2 n_3 \times 1$. Let us denote the vector $X_4(A_4)_i$ by

$$(\mathbf{x}_4)_i = X_4(A_4)_i, \ i = 1, 2, \ \ldots, r.$$

Now, reshape each $(\mathbf{x}_4)_i$, $i = 1, 2, \ \ldots, r$ into matrices $(X_3)_i \in \mathbb{R}^{n_2 \times n_3}$. Then,

$$(A_4 \odot A_3 \odot A_2)^T \mathbf{x} = \begin{bmatrix} (A_3 \odot A_2)_1^T (\mathbf{x}_4)_1 \\ (A_3 \odot A_2)_2^T (\mathbf{x}_4)_2 \\ \vdots \\ (A_3 \odot A_2)_r^T (\mathbf{x}_4)_r \end{bmatrix} = \begin{bmatrix} (A_2)_1^T (X_3)_1 (A_3)_1 \\ (A_2)_2^T (X_3)_2 (A_3)_2 \\ \vdots \\ (A_2)_r^T (X_3)_r (A_3)_r \end{bmatrix}.$$

Computational complexity

The computational complexity of the general product $(A_L \odot A_{L-1} \odot \ \ldots \ \odot A_2)^T \mathbf{x}$ by the above technique is $r(2n-1)\left(\frac{n^{L-1}-1}{n-1}\right)$, where $A_i \in \mathbb{R}^{n \times r}$ and $\mathbf{x} \in \mathbb{R}^{n^{L-1}}$, whereas the direct computation of this product is a bit more expensive as it requires $n^{L-1} r + O(rn^{L-1})$ arithmetic operations including the computation of $A_L \odot A_{L-1} \odot \ \ldots \ \odot A_2$.

3 | QCP ALGORITHM

Let f be a function discretized on a fine grid of size 2^L (for example $L = 15$) with uniform length in an interval. The function values at the grid points generate a vector of size 2^L. As described in the introduction, we can reshape this long vector as a tensor of order L and one can approximate it as a sum of products of vectors of length 2. Figure 1 shows an example of a (third-order tensor) quantized vector of length 2^3, $[\tau_1, \tau_2, \tau_3, \ \ldots, \tau_8]^T$. The construction of a rank r canonical approximation of such a tensor using ALS method is described below.

Let $I = [a, b]$. Consider a uniform mesh with mesh size $h = \frac{1}{2^L - 1}$. Let $\mathbf{f}$ be the vector of length 2^L whose entries are the values of the given function f at these 2^L points on the grid. Let us denote $\mathbf{f}$ by

$$\mathbf{f} = [\tau_1, \tau_2, \tau_3, \ \ldots, \tau_{2^L}]^T. \tag{1}$$

Let χ be the quantized Lth-order tensor, given by

$$\chi = \text{reshape}(\mathbf{f}, \underbrace{2, 2, \ \ldots, 2}_{L}) \in \mathbb{R}^{2 \times 2 \times \ldots \times 2}.$$

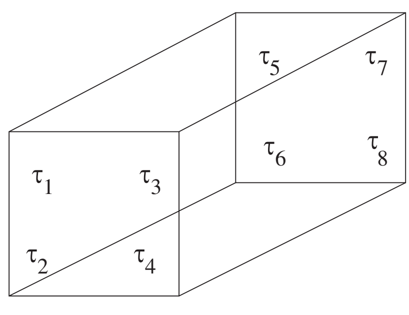

FIGURE 1 Third-order tensor

The precise definition of this operation is shortly recalled here.

The vector $\mathbf{f}$ is reshaped to its quantic image in $\otimes_{J=1}^{L}\mathbb{R}^2$ by dyadic folding,

$$\mathcal{G}_{2,L}\,:\,\mathbf{f}\to\chi=\chi(\mathbf{j})\in\otimes_{J=1}^{L}\mathbb{R}^2,\mathbf{j}=\{j_1,j_2,\ \dots,j_L\},\quad \text{with}\quad j_\upsilon\in\{1,2\},\upsilon=1,2,\ \dots,L,$$

where for fixed i, we have $\chi(\mathbf{j})\,:=\mathbf{f}(i)$ and $j_\upsilon=j_\upsilon(i)$ is defined via 2-coding, $j_\upsilon-1=C_{-1+\upsilon}$, such that the coefficients $C_{-1+\upsilon}$ are found from the dyadic representation of $i-1$,

$$i-1=C_0+C_1\,2+C_2\,2^2+\ \dots\ +C_{L-1}\,2^{L-1}\equiv\sum_{\upsilon=1}^{L}(j_\upsilon-1)\,2^{\upsilon-1}.$$

The rank r canonical approximation of the Lth-order tensor is

$$\chi\cong\sum_{k=1}^{r}\mathbf{a}_k^{(1)}\otimes\mathbf{a}_k^{(2)}\otimes\ \dots\ \otimes\mathbf{a}_k^{(L)},\tag{2}$$

where each $\mathbf{a}_k^{(i)}=\begin{bmatrix}a_{1,k}^{(i)}\\ a_{2,k}^{(i)}\end{bmatrix}$ is a 2×1 vector and $\otimes$ is the usual tensor product.

Let

$$A_1=\left[\mathbf{a}_1^{(1)},\mathbf{a}_2^{(1)},\ \dots,\mathbf{a}_r^{(1)}\right],A_2=\left[\mathbf{a}_1^{(2)},\mathbf{a}_2^{(2)},\ \dots,\mathbf{a}_r^{(2)}\right],\ \dots,A_L=\left[\mathbf{a}_1^{(L)},\mathbf{a}_2^{(L)},\ \dots,\mathbf{a}_r^{(L)}\right].$$

Here, $A_1,A_2,\ \dots,A_L$ are $2\times r$ matrices, corresponding to L different directions, whose columns are the unknown vectors in Equation (2).

Below is the formulation of the ALS:

$$\text{Minimize}\ \frac{1}{2}\left\|\chi-\sum_{k=1}^{r}\mathbf{a}_k^{(1)}\otimes\mathbf{a}_k^{(2)}\otimes\ \dots\ \otimes\mathbf{a}_k^{(L)}\right\|_F^2,\tag{3}$$

where $\|.\|_F$ is the Frobenius norm of a tensor.

In the ALS approach, this functional is minimized in an alternating way. ALS fixes all $A_j,\,j\neq i,j=1,2,\ \dots,L$ to minimize for A_i and continue this process until some convergence criterion is satisfied. That is, first fix $A_2,A_3,\ \dots,A_L$ to solve for A_1 and then fix $A_1,A_3,\ \dots,A_L$ to solve for A_2 and so on, and then fixes $A_1,A_2,\ \dots,A_{L-1}$ to solve for A_L and continue the process.

At each iteration of the ALS approach, we have L steps. First, we start with an initial guess on $A_2,A_3,\ \dots,A_L$ and solve for A_1, this gives the initial guess for the next step. Because we are fixing $L-1$ matrices and solving for one of the matrices $A_i,\ i=1,2,\ \dots,L$ at each step of an iteration, the problem is reduced to a linear least squares problem.

In the ith step of an iteration, we fix $A_1,A_2,\ \dots,A_{i-1},A_{i+1},\ \dots,A_L$ and solve for $A_i=\begin{bmatrix}a_{1,1}^{(i)}&a_{1,2}^{(i)}&\dots&a_{1,r}^{(i)}\\ a_{2,1}^{(i)}&a_{2,2}^{(i)}&\dots&a_{2,r}^{(i)}\end{bmatrix}$. The resulting linear least squares problem is

$$\text{minimize}\ \mathcal{F},$$
$$\text{where}\ \mathcal{F}=\frac{1}{2}\left\|\chi-\sum_{k=1}^{r}\mathbf{a}_k^{(1)}\otimes\mathbf{a}_k^{(2)}\otimes\ \dots\ \otimes\mathbf{a}_k^{(L)}\right\|_F^2\ \text{with}\ A_1,A_2,\ \dots,A_{i-1},A_{i+1},\ \dots,A_L\ \text{fixed}.\tag{4}$$

This gives the equations

$$\frac{\partial\mathcal{F}}{\partial a_{1,1}^{(i)}}=\frac{\partial\mathcal{F}}{\partial a_{1,2}^{(i)}}=\ \dots\ =\frac{\partial\mathcal{F}}{\partial a_{1,r}^{(i)}}=0\quad\text{and}\quad\frac{\partial\mathcal{F}}{\partial a_{2,1}^{(i)}}=\frac{\partial\mathcal{F}}{\partial a_{2,2}^{(i)}}=\ \dots\ =\frac{\partial\mathcal{F}}{\partial a_{2,r}^{(i)}}=0.$$

These equations can be written in the form

$$\begin{bmatrix} \hat{A}_i^T\hat{A}_i & \\ & \hat{A}_i^T\hat{A}_i \end{bmatrix} \begin{bmatrix} \begin{bmatrix} a_{1,1}^{(i)} \\ a_{1,2}^{(i)} \\ \vdots \\ a_{1,r}^{(i)} \end{bmatrix} \\ \begin{bmatrix} a_{2,1}^{(i)} \\ a_{2,2}^{(i)} \\ \vdots \\ a_{2,r}^{(i)} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \hat{A}_i^T\chi(:, \dots, :, 1, :, \dots, :) \\ \hat{A}_i^T\chi(:, \dots, :, 2, :, \dots, :) \end{bmatrix}. \tag{5}$$

Here, $\hat{A}_i = A_L \odot A_{L-1} \odot ... \odot A_{i+1} \odot A_{i-1} \odot ... \odot A_1$, where $\odot$ denotes the Khatri–Rao product of matrices (see A.2). This $\hat{A}_i$ is a $2^{L-1} \times r$ matrix and $\chi(:, \dots, :, j, :, \dots, :)$ $j = 1, 2$ is a vector of length 2^{L-1}. $\hat{A}_i^T\hat{A}_i$ is a $r \times r$ symmetric positive definite matrix. The direct computation of $\hat{A}_i^T\hat{A}_i$ and $\hat{A}_i^T\chi(:, \dots, :, j, :, \dots, :)$ is expensive. The fast computation of these products are described in Section 2.

Remark. For a better understanding of the structure of $\hat{A}_i$ and the derivation of (5) we refer to Appendix A.3. All steps of the ALS algorithm for Rank 2 canonical approximation of a fourth-order tensor are shown there in detail.

From Equation (5), one can see that we need to solve two $r \times r$ linear systems with the same matrix $\hat{A}_i^T\hat{A}_i$ and different right hand-side vectors at each step of an iteration. We continue the iterations until a convergence criterion is reached.

Algorithm

Define tolerance ϵ
Maximum iterations=Maxiter
Initialize $A_i \in \mathbb{R}^{2\times r}, i = 1, 2, \dots, L.$
while iter<=Maxiter
$C_i = A_i, i = 1, 2, \dots, L$

for $i = 1, 2, ..L$
Obtain $\hat{A}_i = A_L \odot A_{L-1} \odot \dots \odot A_{i+1} \odot A_{i-1} \odot \dots \odot A_1; \hat{A}_i^T\chi(:, \dots, :, j, :, \dots, :)$ *for* $j = 1, 2$

Solve $\hat{A}_i^T\hat{A}_i \begin{bmatrix} a_{1,1}^{(i)} \\ a_{1,2}^{(i)} \\ \vdots \\ a_{1,r}^{(i)} \end{bmatrix} = \hat{A}_i^T\chi(:, \dots, :, 1, :, \dots, :)$ *and* $\hat{A}_i^T\hat{A}_i \begin{bmatrix} a_{2,1}^{(i)} \\ a_{2,2}^{(i)} \\ \vdots \\ a_{2,r}^{(i)} \end{bmatrix} = \hat{A}_i^T\chi(:, :, , 2, :, .. :)$

end for

stop if $max\{max\,|A_i - C_i|\} < \epsilon$
iter=iter+1
end while

Computational complexity

Let the number of iterations in the above algorithm be *iter*. In each iteration step of ALS, we need to compute $\hat{A}_i = A_L \odot A_{L-1} \odot \dots \odot A_{i+1} \odot A_{i-1} \odot \dots \odot A_1$ and $\hat{A}_i^T\chi(:, \dots, :, 1, :, \dots, :)$ as well as $\hat{A}_i^T\chi(:, \dots, :, 2, :, \dots, :)$ for $i = 1, 2, \dots, L$ and need to solve a linear least squares system twice.

The computation $\hat{A}_i^T\hat{A}_i = (A_L^TA_L)\circ(A_{L-1}^TA_{L-1})\circ \dots \circ(A_{i+1}^TA_{i+1})\circ(A_{i-1}^TA_{i-1})\circ \dots \circ(A_2^TA_2)\circ(A_1^TA_1)$ requires $O((L-1)2r^2)$ arithmetic operations (look at section 2, and here $n = 2$) and $\hat{A}_i^T\chi(:, \dots, :, j, :, \dots, :)$ requires $3r(2^{L-1} - 1)$ operations (look at Section 2). $O(r^3)$ operations are required to solve a $r \times r$ linear system. Therefore, the total complexity of the algorithm is $O(L((L-1)2r^2 + 3r(2^{L-1} - 1) + r^3))$ for each iteration step. That is $O(2^{L-1})$ per iteration step.

Comments on the algorithm

This is a straightforward ALS algorithm applied to higher order tensors of order L. The initialization of $A_i \in \mathbb{R}^{2\times r}, i = 1, 2, \ldots, L$ is random. The condition number of the matrices $\hat{A}_i^T \hat{A}_i$ is large for large values of r.

4 | NUMERICAL EXAMPLES

In this section, we present the canonical approximation of some functions discretized on $[0, 1]$ and consider the approximation in the following format:

$$\sum_{k=1}^{r} \begin{pmatrix} 1 \\ a_{2,k}^{(1)} \end{pmatrix} \otimes \begin{pmatrix} 1 \\ a_{2,k}^{(2)} \end{pmatrix} \otimes \ldots \otimes \begin{pmatrix} a_{1,k}^{(L)} \\ a_{2,k}^{(L)} \end{pmatrix}.$$

The number of parameters in this format is almost half of the parameters required for the canonical representation given in Equation (2). Therefore, the computational complexity is here further reduced. The condition numbers of the matrices $\hat{A}_i^T \hat{A}_i$ are much better in this case.

In all the numerical examples given below, the functions are discretized on a uniform grid of size 2^{15}, so the reshaped tensor is of order 15. In all the tables below "*error*" denotes the maximum error in the canonical approximation of the discretized function. The initial matrices A_i are chosen randomly and the computations are carried out in MATLAB.

Example 1. Consider the function $f(x) = e^{-x^2}$ in $[0, 1]$. We have obtained the canonical approximation with different ranks; see Table 1. We have considered *Maxiter*=125.

Example 2. Consider the functions $\sin(\pi x)$, $\sin(2\pi x)$, and $\sin(4\pi x)$ in $[0, 1]$. Table 2 shows the error in the maximum norm for different values of r. In this example, we have considered *Maxiter*=125.

Example 3. Now, consider the functions $f(x) = x$ or $f(x) = x^2$ in $[0, 1]$. Table 3 shows the *error* for different values of r. In this example, we have considered *Maxiter*=125.

TABLE 1 Error in the maximum norm for different values of r

r	*error*
1	0.108596
2	0.031
3	0.0081
4	0.0023
5	0.00071
6	0.00024
7	0.00015
8	0.0000881
9	0.0000461
10	0.0000210

TABLE 2 Error for different ranks in the canonical approximation

	$\sin(\pi x)$	$\sin(2\pi x)$	$\sin(4\pi x)$
r	*error*	*error*	*error*
1	0.63658	1.000	1.0
2	0.164	0.250	0.162
3	0.0336	0.0723	0.067
4	0.00635	0.0341	0.0308
5	0.0014	0.00591	0.0059
6	0.000292	0.00168	0.0022
7	0.0000822	0.000389	0.0010
8	0.0000572	0.000172	0.000370
9	0.00000901	0.0000886	0.000142
10	0.00000671	0.0000317	0.000070

TABLE 3 Error for different values of r

r	x error	x^2 error
1	0.176	0.075
2	0.0186	0.0276
3	0.00576	0.00661
4	0.00133	0.00121
5	0.000346	0.000218
6	0.000082	0.00005
7	0.000022	0.0000125
8	0.00000652	0.00000927
9	0.00000268	0.00000351
10	0.000000728	0.00000252

In all the examples above, one can observe that the error decays exponentially with r such as μ^r, where $\mu < 1$. In addition, one can see that the function (or better, its discretized representation) has been well approximated by the QCP format using only 160 parameters, where the original size was 2^{15}. Please note that, so far, we have used complete information of the data to obtain the QCP approximation. A more effective way based on the QCP interpolation is sketched in the following section.

5 | THE QCP APPROXIMATION USING ONLY A FEW FUNCTION CALLS

In Section 3, we have seen the construction of a rank r canonical approximation using the complete data of size 2^L. Here, we describe the idea of constructing the rank r canonical approximation using function values at a few sampling points only. The more detailed presentation is the topic of our ongoing work. Let $M\,(= O(2Lr))$ be the number of sampling points, comparable to the number of unknown representation parameters. Many issues such as a good choice of the sampling points and the robust error analysis of the method will not be addressed in this article. This approach can be viewed as the sparse interpolation of a given function in the QCP format by using a small number of functional calls.

Consider the rank-r canonical approximation of the tensor χ

$$\chi \cong \sum_{k=1}^{r} \mathbf{a}_k^{(1)} \otimes \mathbf{a}_k^{(2)} \otimes \dots \otimes \mathbf{a}_k^{(L)}.$$

The method to evaluate the unknown parameters $\mathbf{a}_k^{(i)}, i = 1, 2, \dots, L,\ k = 1, 2, \dots, r$, using the information of the tensor χ only at M positions is given below. We let

$$A_1 = \left[\mathbf{a}_1^{(1)}, \mathbf{a}_2^{(1)}, \dots, \mathbf{a}_r^{(1)}\right], A_2 = \left[\mathbf{a}_1^{(2)}, \mathbf{a}_2^{(2)}, \dots, \mathbf{a}_r^{(2)}\right], \dots, A_L = \left[\mathbf{a}_1^{(L)}, \mathbf{a}_2^{(L)}, \dots, \mathbf{a}_r^{(L)}\right]$$

be the side matrices.

Suppose we have chosen M points $s_1, s_2, \dots, s_M$ on the grid with corresponding function values such that they represent the function well in the whole interval. The corresponding entries in the vector $\mathbf{f}$ are denoted by $\tau_{s_1}, \tau_{s_2}, \dots, \tau_{s_M}$. We can identify these entries at certain positions in the Lth-order tensor χ and one can obtain the subscripts in the tensor product corresponding to the linear index of the entries $\tau_{s_1}, \tau_{s_2}, \dots, \tau_{s_M}$. Let us denote the subscripts corresponding to each linear index by

$$\begin{aligned} s_1 &\to \left(i_1^{s_1}, i_2^{s_1}, \dots, i_L^{s_1}\right) \\ s_2 &\to \left(i_1^{s_2}, i_2^{s_2}, \dots, i_L^{s_2}\right) \\ &\vdots \\ s_M &\to \left(i_1^{s_M}, i_2^{s_M}, \dots, i_L^{s_M}\right). \end{aligned} \tag{6}$$

Remember that each subscript $i_j^{s_k}$ is either 1 or 2 for all $j = 1, 2, \dots, L, k = 1, 2, \dots, M$.

Analog to what is shown in Appendix A.3, we minimize the functional

$$\mathcal{F} = \frac{1}{2}\left(\left(\tau_{s_1} - \sum_{k=1}^{r} a^{(1)}_{i^{s_1}_1,k} a^{(2)}_{i^{s_1}_2,k} \dots a^{(L)}_{i^{s_1}_L,k}\right)^2 + \left(\tau_{s_2} - \sum_{k=1}^{r} a^{(1)}_{i^{s_2}_1,k} a^{(2)}_{i^{s_2}_2,k} \dots a^{(L)}_{i^{s_2}_L,k}\right)^2 + \dots + \left(\tau_{s_M} - \sum_{k=1}^{r} a^{(1)}_{i^{s_M}_1,k} a^{(2)}_{i^{s_M}_2,k} \dots a^{(L)}_{i^{s_M}_L,k}\right)^2\right)$$

with respect to the unknown side matrices. At each iteration of ALS, we have L steps. In the ith step of an iteration, we fix $A_1, A_2, \dots, A_{i-1}, A_{i+1}, \dots, A_L$ and solve for $A_i = \begin{bmatrix} a^{(i)}_{1,1} & a^{(i)}_{1,2} & \dots & a^{(i)}_{1,r} \\ a^{(i)}_{2,1} & a^{(i)}_{2,2} & \dots & a^{(i)}_{2,r} \end{bmatrix}$. This reduces the problem to a linear least squares problem. The linear system looks very similar to the system in (5), but with some differences. Here, we describe it in detail.

Among the M sampling points $s_1, s_2, \dots, s_M$, let $p_1, p_2, \dots, p_{N_1}$ be the linear indices having 1 as the ith subscript and $q_1, q_2, \dots, q_{N_2}$ be the linear indices having 2 as the ith subscript ($N_1 + N_2 = M$). Then, the linear system is given by

$$\hat{A}^T_{i,1}\hat{A}_{i,1}\begin{bmatrix} a^{(i)}_{1,1} \\ a^{(i)}_{1,2} \\ \vdots \\ a^{(i)}_{1,r} \end{bmatrix} = \hat{A}^T_{i,1} r_{i,1} \text{ and } \hat{A}^T_{i,2}\hat{A}_{i,2}\begin{bmatrix} a^{(i)}_{2,1} \\ a^{(i)}_{2,2} \\ \vdots \\ a^{(i)}_{2,r} \end{bmatrix} = \hat{A}^T_{i,2} r_{i,2},$$

$$\text{where} \quad \hat{A}_{i,1} = \begin{bmatrix} \hat{a}^{p_1}_1 & \hat{a}^{p_1}_2 & \hat{a}^{p_1}_r \\ \hat{a}^{p_2}_1 & \hat{a}^{p_2}_2 & \hat{a}^{p_2}_r \\ \\ \hat{a}^{p_{N_1}}_1 & \hat{a}^{p_{N_1}}_2 & \hat{a}^{p_{N_1}}_r \end{bmatrix} \text{with } \hat{a}^p_k = a^{(L)}_{i^p_L,k} a^{(L-1)}_{i^p_{L-1},k} \dots a^{(i+1)}_{i^p_{i+1},k} a^{(i-1)}_{i^p_{i-1},k} \dots a^{(1)}_{i^p_1,k},$$

$$\hat{A}_{i,2} = \begin{bmatrix} \hat{a}^{q_1}_1 & \hat{a}^{q_1}_2 & \hat{a}^{q_1}_r \\ \hat{a}^{q_2}_1 & \hat{a}^{q_2}_2 & \hat{a}^{q_2}_r \\ \\ \hat{a}^{q_{N_2}}_1 & \hat{a}^{q_{N_2}}_2 & \hat{a}^{q_{N_2}}_r \end{bmatrix} \text{with } \hat{a}^q_k = a^{(L)}_{i^q_L,k} a^{(L-1)}_{i^q_{L-1},k} \dots a^{(i+1)}_{i^q_{i+1},k} a^{(i-1)}_{i^q_{i-1},k} \dots a^{(1)}_{i^q_1,k},$$

$$\text{and} \quad r_{i,1} = \begin{bmatrix} \tau_{p_1} \\ \tau_{p_2} \\ \vdots \\ \tau_{p_{N_1}} \end{bmatrix}, r_{i,2} = \begin{bmatrix} \tau_{q_1} \\ \tau_{q_2} \\ \vdots \\ \tau_{q_{N_2}} \end{bmatrix}.$$

Remark. The matrices $\hat{A}_{i,1}$ and $\hat{A}_{i,2}$ are very similar to $A_L \odot A_{L-1} \odot \dots \odot A_{i+1} \odot A_{i-1} \odot \dots \odot A_1$, but with many rows missing. The sizes of the matrices $\hat{A}_{i,1}$ and $\hat{A}_{i,2}$ are $N_1 \times r$ and $N_2 \times r$, respectively, which are very small compared to $\hat{A}_i$ in (5).

This leads to a reduction of the computational complexity. Here, we present a numerical example to check the performance of the algorithm. We consider an approximation in the following format:

$$\sum_{k=1}^{r} \begin{pmatrix} a^{(1)}_{1,k} \\ a^{(1)}_{2,k} \end{pmatrix} \otimes \begin{pmatrix} a^{(2)}_{1,k} \\ a^{(2)}_{2,k} \end{pmatrix} \otimes \dots \otimes \begin{pmatrix} a^{(L)}_{1,k} \\ a^{(L)}_{2,k} \end{pmatrix}.$$

A further reduction of the number of unknowns is possible if one uses the format, which has been discussed in Section 4.

Example 4. Consider the function $f(x) = e^{-x^2}$ in $[0, 1]$ and $f(x) = e^{-50x^2}$ in $[0, 0.25]$. Let $L = 12$; therefore, the grid size is 2^{12}. We have obtained the canonical approximation with different ranks using the information of the function at $M = 2Lr$ or $M = 4Lr$ sampling points. The sampling points and initial matrices are chosen randomly. Table 4 shows "*error*" in the approximation for different values of the rank r (in analogy to Section 4, the maximum error is considered).

Table 4 also shows the number of sampling points used to obtain the canonical approximation. For the function $f(x) = e^{-x^2}$, the *error* decays very fast in the case of $M = 4Lr$ compared to the case of $M = 2Lr$. One can see that we have used function values only at 288 points to approximate the tensor to $O(10^{-4})$ accuracy instead of using the information at 4,096 points. In this case, we have considered *Maxiter*=125. The results are presented for $M = 4Lr$ in the case of the function $f(x) = e^{-50x^2}$. The error decays fast and we have used the information only at 384 points to approximate the tensor to

TABLE 4 Error of the QCP interpolation for different values of r and M

	e^{-x^2} in [0,1]				e^{-50x^2} in [0,0.25]	
r	$M = 2Lr$	*error*	$M = 4Lr$	*error*	$M = 4Lr$	*error*
1	24	0.219347	48	0.144140	48	0.2081219
2	48	0.056676	96	0.0291372	96	0.0291072
3	72	0.011712	144	0.0075389	144	0.0124090
4	96	0.006980	192	0.0036845	192	0.0040713
5	120	0.003715	240	0.0019918	240	0.0023895
6	144	0.002515	288	0.0002400	288	0.0013455
7	168	0.001142			336	0.00084574
8	192	0.000697			384	0.00026631

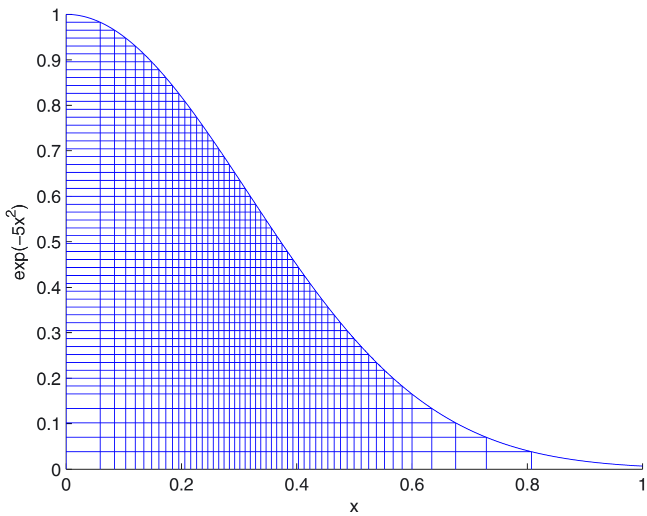

FIGURE 2 Adaptive grid for the sparse quantized canonical polyadic interpolation

$O(10^{-4})$ instead of 4,096. In this case, we have considered *Maxiter*=125. In both cases, we can see that the error decays exponentially like μ^r, where $\mu < 1$.

The sparse interpolation in the QCP format requires the information of the function only at $M\,(\sim 2Lr)$ points instead of the information at the full set 2^L of grid points. The overall computational complexity of the algorithm is reduced dramatically and it is $O(M)$ per iteration step of the ALS algorithm. Here, $M \ll N = 2^L$. In the above numerical example, the sampling points were chosen randomly. Clearly, there are many strategies for adaptive selection of sampling points based on some, a priori knowledge about the behavior of the underlying function, but this issue will not be discussed here in detail. Figure 2 shows an example of the adaptive choice of the interpolation grid for the function $y = f(x) = e^{-5x^2}$. This grid is obtained as an image or the uniform grid on the y-axis under the mapping by the inverse function $x = f^{-1}(y)$.

To complete this section, we briefly discuss the relation between the QCP and QTT rank parameters in the perspective of numerical complexity for the large number of grid points N. Recall that the number M of functional calls in our QCP interpolation scheme scales as $2r_{qcp}L$, which is on the order of magnitude smaller than N, in all numerical tests reported in Tables 1–4. The storage size for the QCP tensor scales as $O(2r_{qcp}\log N)$, whereas for the QTT tensor the corresponding cost amounts to $O(2r_{qtt}^2 \log N)$. Hence, the typical QCP rank r_{qcp} should be compared with r_{qtt}^2. For functions considered in our numerical example, we achieved the accuracy about $10^{-4} - 10^{-5}$ with the QCP rank parameter $r_{qcp} \sim 10$. For similar functions the QTT the accuracy about $10^{-5} - 10^{-6}$ was achieved with the QTT rank parameter about $2.5 \leq r_{qtt} \leq 4$ (see Section 4.1.8 in the work of Khoromskij[44] and Section 14.4 in the work of Khoromskaia et al.[46]). These numerical results indicate the very similar storage complexity for the QCP and QTT approximations providing the approximation error about $10^{-4} - 10^{-5}$.

Notice that the so-called TT-cross approximation in the TT format[47] requires asymptotically smaller number of functional calls than N, but only in the case of rather large N. More precisely, in numerical test, we observe that the number of functional calls in the QTT-cross interpolation algorithms behaves like $2Cr_{qtt}^2 \log N$ at the limit of large N, where the constant C is about $C \approx 10$; see the discussion in other works.[8,44]

6 | CONCLUSIONS AND FUTURE WORK

In this article, the ALS-type algorithms for approximation/interpolation of a function in QCP format have been described. The representation complexity of the rank-r QCP format is estimated by $2Lr$. As commented in Section 2, the condition numbers of the matrices appearing in each iteration of the ALS algorithm are large for large values of the rank r. Complete data of the tensor has been used to obtain the CP approximation at the computational cost $O(2^{L-1})$ per iteration. This complexity is reduced if the approximation can be obtained using only a few data points, which can be viewed as the sparse interpolation of a given function in the QCP format.

The idea of obtaining CP approximation using only a small number of functional calls is described and numerical examples are presented. In this case, the overall computational complexity of the QCP approximation is only $O(2Lr)$ per iteration step of the algorithm, that is, it is proportional to the number of representation parameters in the target QCP tensor. It is remarkable that the complexity of the QCP interpolation scales linearly in the CP rank and logarithmically in the full vector size.

A discussion of different strategies for clever choice of the sampling points as well as the error analysis of the method and the extension of the algorithm to functions of two or three variables is postponed to ongoing work. The QCP format can be used in the approximation of the solution of PDEs, integration of highly oscillating functions, and to approximate functions where the calculation of function values is computationally expensive. This format can also be used to just represent functions that depend on many parameters.

ACKNOWLEDGEMENTS

Kishore Kumar appreciates the support provided by the Max Planck Institute for Mathematics in the Sciences (MPI MIS; Leipzig, Germany) during his scientific visit in 2015. The authors are thankful to Dr. V. Khoromskaia (MPI MIS, Leipzig, Germany) for useful discussions. This work is partially supported by the National Board of Higher Mathematics, Department of Atomic Energy (DAE), India.

CONFLICT OF INTEREST

This work does not have any conflicts of interest.

ORCID

Boris N. Khoromskij https://orcid.org/0000-0002-5853-5521
N. Kishore Kumar https://orcid.org/0000-0003-1365-6319
Jan Schneider https://orcid.org/0000-0002-1229-2579

REFERENCES

1. Khoromskij BN. $O(d \log N)$-quantics approximation of N-d tensors in high-dimensional numerical modeling. Constructive Approximation. 2011;34(2):257–289.
2. Oseledets IV. Tensor-train decomposition. SIAM J Sci Comput. 2011;33(5):2295–2317.
3. Oseledets IV. Approximation of $2^d \times 2^d$ matrices using tensor decomposition. SIAM J Matrix Anal Appl. 2010;31(4):2130–2145.
4. Benner P, Dolgov S, Khoromskaia V, Khoromskij B. Fast iterative solution of the Bethe-Salpeter eigenvalue problem using low-rank and QTT tensor approximation. J Comput Phys. 2017;334:221–239.
5. Dolgov SV, Khoromskij B. Two-level Tucker-TT-QTT format for optimized tensor calculus. SIAM J Matrix Anal Appl. 2013;34(2):593–623.
6. Kazeev V, Oseledets I, Rakhuba M, Schwab C. QTT-finite-element approximation for multiscale problems I: model problems in one dimension. Adv Comput Math. 2017;43(2):411–442.
7. Kazeev V, Reichmann O, Schwab C. Low-rank tensor structure of linear diffusion operators in the TT and QTT formats. Linear Algebra Appl. 2013;438(11):4204–4221.
8. Khoromskij BN, Sauter S, Veit A. Fast quadrature techniques for retarded potentials based on TT/QTT tensor approximation. Comput Methods Appl Math. 2011;11(3):342–362.
9. Dolgov S, Khoromskij BN, Litvinenko A, Matthies HG. Computation of the response surface in the tensor train data format. SIAM/ASA J Uncertain Quantification. 2015;3:1109–1135.
10. Khoromskij BN, Oseledets I. Quantics-TT Collocation approximation of parameter-dependent and stochastic elliptic PDEs. Comput Methods Appl Math. 2010;10(4):376–394.

11. Khoromskij BN, Oseledets I. Quantics-TT approximation of elliptic solution operators in higher dimensions. Russ J Numer Anal Math Model. 2011;26(3):303–322.
12. Khoromskij BN, Repin SI. A fast iteration method for solving elliptic problems with quasiperiodic coefficients. Russ J Numer Anal Math Model. 2015;30(6):329–344.
13. Khoromskij BN, Schwab C. Tensor-structured Galerkin approximation of parametric and stochastic elliptic PDEs. SIAM J Sci Comput. 2011;33(1):364–385.
14. Khoromskaia V, Khoromskij BN. Grid-based lattice summation of electrostatic potentials by assembled rank-structured tensor approximation. Comput Phys Commun. 2014;185:3162–3174.
15. Khoromskaia V, Khoromskij BN, Schneider R. QTT representation of the hartree and exchange operators in electronic structure calculations. Comput Methods Appl Math. 2011;11(3):327–341.
16. Khoromskij BN. Tensors-structured numerical methods in scientific computing: survey on recent advances. Chemom Intell Lab Syst. 2012;110:1–19.
17. Carroll JD, Chang J-J. Analysis of individual differences in multidimensional scaling via an N-way generalization of 'Eckart-Young' decomposition. Psychometrika. 1970;35:283–319.
18. Comon P, Luciani X, de Almeida ALF. Tensor decomposition, alternating least squares and other tales. J Chemom. 2009;23:393–405.
19. Domanov I. Study of canonical polyadic decomposition of higher order tensors [doctoral thesis]. Leuven, Belgium: KU Leuven; 2013.
20. Espig M, Hackbusch W, Khachatryan A. On the convergence of alternating least squares optimisation in tensor format representations. Preprint:423. Aachen: RWTH Aachen University; 2015.
21. Golub GH, Van Loan CF. Matrix computations. 4th ed. Baltimore, MD: The Hopkins University Press; 2013. Johns Hopkins studies in the mathematical sciences.
22. Harshman RA. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-model factor analysis. Los Angeles, CA: University of California, Los Angeles; 1970;1–84. Issue 16 of UCLA working papers in phonetics. http://publish.uwo.ca/~harshman/wpppfac0.pdf
23. Kolda TG, Bader BW. Tensor decompositions and applications. SIAM Rev. 2009;51(3):455–500.
24. Rohwedder T, Holtz S, Schneider R. The alternation least square scheme for tensor optimisation in the TT-format. SIAM J Sci Comput. 2012;34(2):A683–A713.
25. Uschmajew A. Local convergence of the alternative least squares algorithm for canonical tensor approximation. SIAM J Matrix Anal Appl. 2012;33(2):639–652.
26. Van Loan CF. Lectures. http://issnla2010.ba.cnr.it/Course_Van_Loan.htm
27. Li N, Kindermann S, Navasca C. Some convergence results on the regularized alternating least-squares method for tensor decomposition. Linear Algebra Appl. 2013;438(2):796–812.
28. Navasca C, De Lathauwer L, Kindermann S. Swamp reducing technique for tensor decompositions. Proceedings of the 16th European Signal Processing Conference (EUSIPCO); 2008 Aug 25–29; Lausanne, Switzerland. Piscataway, NJ: IEEE.
29. Gandy S, Recht B, Yamada I. Tensor completion and low-n-rank tensor recovery via convex optimization. Inverse Problems. 2011;27:025010.
30. Grasedyck L, Krammer S. Stable ALS approximation in the TT-format for rank-adaptive tensor completion. arXiv preprint arXiv:1701.08045. 2017.
31. Karlsson L, Kressner D, Uschmajew A. Parallel algorithms for tensor completion in CP format. Parallel Computing. 2016;57:222–234.
32. Da Silva C, Hermann FJ. Optimization on the hierarchical Tucker manifold-applications of tensor completion. Lin Alg Appl. 2015;481:131–173.
33. Tomasi G, Bro R. PARAFAC and missing values. Chemom Intell Lab Syst. 2005;75(2):163–180.
34. Yokota T, Zhao Q, Cichocki A. Smooth PARAFAC decomposition for tensor completion. IEEE Trans Signal Process. 2016;64(20):5423–5436.
35. Candès EJ, Recht B. Exact matrix completion via convex optimization. Found Comput Math. 2009;9:717–772.
36. Kressner D, Steinlechner M, Vandereycken B. Low-rank tensor completion by Riemannian optimization. BIT Numer Math. 2014;54(2):447–468.
37. Rauhut H, Schneider R, Stojanac Z. Tensor completion in heierarchical tensor representations. In: Compressed sensing and its applications. Cham, Switzerland: Birkhäuser. 2015:419–450.
38. Song Q, Hancheng G, Caverlee J, Hu X. Tensor completion algorithms in big data analysis. arXiv preprint arXiv:1711.10105. 2018.
39. Grasedyck L, Kressner D, Tobler C. A literature survey of low-rank tensor approximation techniques. GAMM-Mitteilungen. 2013;36(1):53–78.
40. Hackbusch W. Tensor spaces and numerical tensor calculus. Berlin, Germany: Springer; 2012.
41. Kishore Kumar N, Schneider J. Literature survey on low rank approximation of matrices. Lin Multilin Alg. 2017;65(11):2212–2244.
42. Oseledets IV, Savostyanov DV, Tyrtyshnikov EE. Linear algebra for tensor problems. Computing. 2009;85:169–188.
43. Cichocki A, Lee N, Oseledets I, Phan A-H, Zhao Q, Mandic DP. Tensor networks for dimensionality reduction and large-scale optimization: part 1 low-rank tensor decompositions. Found Trends Mach Learn. 2016;9(4–5):249–429.
44. Khoromskij BN. Tensor numerical methods in scientific computing. Berlin, Germany: De Gruyter; 2018.
45. Khoromskaia V, Khoromskij BN. Tensor numerical methods in quantum chemistry: from Hartree–Fock to excitation energies. Phys Chem Chem Phys. 2015;17:31491–31509.

46. Khoromskaia V, Khoromskij BN. Tensor numerical methods in quantum chemistry: research monograph. Berlin, Germany: De Gruyter; 2018.
47. Oseledets IV, Tyrtyshnikov EE. TT-Cross Approximation for Multidimensional arrays. Linear Algebra Appl. 2010;432(1):70–88.

How to cite this article: Khoromskij BN, Kishore Kumar N, Schneider J. Quantized CP approximation and sparse tensor interpolation of function-generated data. *Numer Linear Algebra Appl.* 2019;e2262. https://doi.org/10.1002/nla.2262

APPENDIX

A.1 | Kronecker product

Let A be an $m \times n$ matrix and B be an $p \times q$ matrix, then the Kronecker product $A \otimes B$ is an $mp \times nq$ block matrix:

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ a_{21}B & a_{22}B & \dots & a_{2n}B \\ & & & \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{bmatrix}.$$

Properties of the Kronecker product

For matrices B, C, D, and F of suitable sizes, the following properties hold:

P1. $(B \otimes C)^T = B^T \otimes C^T$.
P2. $(B \otimes C) \otimes D = B \otimes (C \otimes D)$.
P3. $(B \otimes C)(D \otimes F) = BD \otimes CF$.
P4. $(B \otimes C)^{-1} = B^{-1} \otimes C^{-1}$.

A.2 | Khatri–Rao product

Let $B = [\mathbf{b}_1|\mathbf{b}_2| \dots |\mathbf{b}_r] \in \mathbb{R}^{n_1 \times r}$, where $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_r$ are the columns of the matrix B. Let $C = [\mathbf{c}_1|\mathbf{c}_2| \dots |\mathbf{c}_r] \in \mathbb{R}^{n_2 \times r}$. The Khatri–Rao product of B and C is defined as the $n_1 n_2 \times r$ matrix

$$B \odot C = [\mathbf{b}_1 \otimes \mathbf{c}_1|\mathbf{b}_2 \otimes \mathbf{c}_2| \dots |\mathbf{b}_r \otimes \mathbf{c}_r].$$

Here, $\mathbf{b}_i \otimes \mathbf{c}_i$ is

$$\mathbf{b}_i \otimes \mathbf{c}_i = \begin{bmatrix} b_{1i} \\ b_{2i} \\ \vdots \\ b_{n_1 i} \end{bmatrix} \otimes c_i = \begin{bmatrix} b_{1i}c_i \\ b_{2i}c_i \\ \vdots \\ b_{n_1 i}c_i \end{bmatrix} = \begin{bmatrix} b_{1i}c_{1i} \\ b_{1i}c_{2i} \\ \vdots \\ b_{1i}c_{n_2 i} \\ b_{2i}c_{1i} \\ \vdots \\ b_{n_1 i}c_{n_2 i} \end{bmatrix}.$$

$O(n_1 n_2 r)$ arithmetic operations are required to compute $B \odot C$.

A.3 | Rank 2 canonical approximation of a fourth-order tensor

Consider a fourth-order tensor. Imagine that the tensor is generated by reshaping a vector $\mathbf{x} = [\tau_1, \tau_2, \tau_3, \dots, \tau_{16}]^T$ of length 16. Let $\chi = \text{reshape}(\mathbf{x}, 2, 2, 2, 2)$.

We obtain a rank two canonical approximation to the tensor χ using ALS. The Rank 2 approximation in canonical format is given by

$$\chi \cong \begin{pmatrix} a_1^1 \\ a_2^1 \end{pmatrix} \otimes \begin{pmatrix} b_1^1 \\ b_2^1 \end{pmatrix} \otimes \begin{pmatrix} c_1^1 \\ c_2^1 \end{pmatrix} \otimes \begin{pmatrix} d_1^1 \\ d_2^1 \end{pmatrix} + \begin{pmatrix} a_1^2 \\ a_2^2 \end{pmatrix} \otimes \begin{pmatrix} b_1^2 \\ b_2^2 \end{pmatrix} \otimes \begin{pmatrix} c_1^2 \\ c_2^2 \end{pmatrix} \otimes \begin{pmatrix} d_1^2 \\ d_2^2 \end{pmatrix}.$$

To obtain the Rank 2 canonical approximation, we minimize the functional $\mathcal{F}$

$$\mathcal{F} = \frac{1}{2}\left\| \chi - \begin{pmatrix} a_1^1 \\ a_2^1 \end{pmatrix} \otimes \begin{pmatrix} b_1^1 \\ b_2^1 \end{pmatrix} \otimes \begin{pmatrix} c_1^1 \\ c_2^1 \end{pmatrix} \otimes \begin{pmatrix} d_1^1 \\ d_2^1 \end{pmatrix} + \begin{pmatrix} a_1^2 \\ a_2^2 \end{pmatrix} \otimes \begin{pmatrix} b_1^2 \\ b_2^2 \end{pmatrix} \otimes \begin{pmatrix} c_1^2 \\ c_2^2 \end{pmatrix} \otimes \begin{pmatrix} d_1^2 \\ d_2^2 \end{pmatrix} \right\|_F^2$$
$$= \frac{1}{2}\left(\left(\tau_1 - \left(a_1^1 b_1^1 c_1^1 d_1^1 + a_1^2 b_1^2 c_1^2 d_1^2\right)\right)^2 + \left(\tau_2 - \left(a_2^1 b_1^1 c_1^1 d_1^1 + a_2^2 b_1^2 c_1^2 d_1^2\right)^2 + \dots + \left(\tau_{16} - \left(a_2^1 b_2^1 c_2^1 d_2^1 + a_2^2 b_2^2 c_2^2 d_2^2\right)^2 \right).$$

Let us denote

$$A = \begin{bmatrix} a_1^1 & a_1^2 \\ a_2^1 & a_2^2 \end{bmatrix}, B = \begin{bmatrix} b_1^1 & b_1^2 \\ b_2^1 & b_2^2 \end{bmatrix}, C = \begin{bmatrix} c_1^1 & c_1^2 \\ c_2^1 & c_2^2 \end{bmatrix}, \text{ and } D = \begin{bmatrix} d_1^1 & d_1^2 \\ d_2^1 & d_2^2 \end{bmatrix}.$$

By ALS, $\mathcal{F}$ is minimized in an alternating way. ALS first fixes B, C, and D to minimize for A, then fixes A, C, and D to minimize for B, then fixes A, B, and D to minimize for C, and finally fixes A, B, and C to minimize for D. Because we are fixing all but one direction in each step of an iteration, the problem reduces to a linear least squares problem. All the steps of one iteration are described below.

Step 1. Fix B, C, and D and solve for A. The minimization leads to the equations

$$\frac{\partial \mathcal{F}}{\partial a_1^1} = 0, \frac{\partial \mathcal{F}}{\partial a_1^2} = 0 \quad \text{and} \quad \frac{\partial \mathcal{F}}{\partial a_2^1} = 0, \frac{\partial \mathcal{F}}{\partial a_2^2} = 0,$$

which give a decoupled diagonal system

$$\begin{bmatrix} \hat{A}^T\hat{A} & 0 \\ 0 & \hat{A}^T\hat{A} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} a_1^1 \\ a_1^2 \end{bmatrix} \\ \begin{bmatrix} a_2^1 \\ a_2^2 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \hat{A}^T\chi(1,:,:,:) \\ \hat{A}^T\chi(2,:,:,:) \end{bmatrix}, \tag{A1}$$

where (see A.2)

$$\hat{A} = \begin{bmatrix} d_1^1c_1^1b_1^1 & d_1^2c_1^2b_1^2 \\ d_1^1c_1^1b_2^1 & d_1^2c_1^2b_2^2 \\ d_1^1c_2^1b_1^1 & d_1^2c_2^2b_1^2 \\ d_1^1c_2^1b_2^1 & d_1^2c_2^2b_2^2 \\ d_2^1c_1^1b_1^1 & d_2^2c_1^2b_1^2 \\ d_2^1c_1^1b_2^1 & d_2^2c_1^2b_2^2 \\ d_2^1c_2^1b_1^1 & d_2^2c_2^2b_1^2 \\ d_2^1c_2^1b_2^1 & d_2^2c_2^2b_2^2 \end{bmatrix} = D \odot C \odot B, \quad \chi(1,:,:,:) = \begin{bmatrix} \tau_1 \\ \tau_3 \\ \tau_5 \\ \tau_7 \\ \tau_9 \\ \tau_{11} \\ \tau_{13} \\ \tau_{15} \end{bmatrix}, \quad \chi(2,:,:,:) = \begin{bmatrix} \tau_2 \\ \tau_4 \\ \tau_6 \\ \tau_8 \\ \tau_{10} \\ \tau_{12} \\ \tau_{14} \\ \tau_{16} \end{bmatrix}. \tag{A2}$$

Step 2. Fix A, C, D and solve for B. Then, the equations

$$\frac{\partial \mathcal{F}}{\partial b_1^1} = 0, \frac{\partial \mathcal{F}}{\partial b_1^2} = 0 \quad \text{and} \quad \frac{\partial \mathcal{F}}{\partial b_2^1} = 0, \frac{\partial \mathcal{F}}{\partial b_2^2} = 0$$

give the linear system

$$\begin{bmatrix} \hat{B}^T\hat{B} & 0 \\ 0 & \hat{B}^T\hat{B} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} b_1^1 \\ b_1^2 \end{bmatrix} \\ \begin{bmatrix} b_2^1 \\ b_2^2 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \hat{B}^T\chi(:,1,:,:) \\ \hat{B}^T\chi(:,2,:,:) \end{bmatrix}$$

with

$$\hat{B} = D \odot C \odot A \quad \text{and} \quad \chi(:,1,:,:) = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_5 \\ \tau_6 \\ \tau_9 \\ \tau_{10} \\ \tau_{13} \\ \tau_{14} \end{bmatrix}, \quad \chi(:,2,:,:) = \begin{bmatrix} \tau_3 \\ \tau_4 \\ \tau_7 \\ \tau_8 \\ \tau_{11} \\ \tau_{12} \\ \tau_{15} \\ \tau_{16} \end{bmatrix}. \tag{A3}$$

Step 3. Fix A, B, D and solve for C. Then, the equations

$$\frac{\partial \mathcal{F}}{\partial c_1^1} = 0, \frac{\partial \mathcal{F}}{\partial c_1^2} = 0 \quad \text{and} \quad \frac{\partial \mathcal{F}}{\partial c_2^1} = 0, \frac{\partial \mathcal{F}}{\partial c_2^2} = 0$$

give the linear system

$$\begin{bmatrix} \hat{C}^T\hat{C} & 0 \\ 0 & \hat{C}^T\hat{C} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} c_1^1 \\ c_1^2 \end{bmatrix} \\ \begin{bmatrix} c_2^1 \\ c_2^2 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \hat{C}^T \chi(:,:,1,:) \\ \hat{C}^T \chi(:,:,2,:) \end{bmatrix}$$

with

$$\hat{C} = D \odot B \odot A \quad \text{and} \quad \chi(:,:,1,:) = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \\ \tau_9 \\ \tau_{10} \\ \tau_{11} \\ \tau_{12} \end{bmatrix}, \quad \chi(:,:,2,:) = \begin{bmatrix} \tau_5 \\ \tau_6 \\ \tau_7 \\ \tau_8 \\ \tau_{13} \\ \tau_{14} \\ \tau_{15} \\ \tau_{16} \end{bmatrix}. \tag{A4}$$

Step 4. Fix A, B, C and solve for D. The equations

$$\frac{\partial \mathcal{F}}{\partial d_1^1} = 0, \frac{\partial \mathcal{F}}{\partial d_1^2} = 0 \quad \text{and} \quad \frac{\partial \mathcal{F}}{\partial d_2^1} = 0, \frac{\partial \mathcal{F}}{\partial d_2^2} = 0$$

give the linear system

$$\begin{bmatrix} \hat{D}^T\hat{D} & 0 \\ 0 & \hat{D}^T\hat{D} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} d_1^1 \\ d_1^2 \end{bmatrix} \\ \begin{bmatrix} d_2^1 \\ d_2^2 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \hat{D}^T \chi(:,:,:,1) \\ \hat{D}^T \chi(:,:,:,2) \end{bmatrix}$$

with

$$\hat{D} = C \odot B \odot A \quad \text{and} \quad \chi(:,:,:,1) = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \\ \tau_5 \\ \tau_6 \\ \tau_7 \\ \tau_8 \end{bmatrix}, \quad \chi(:,:,:,2) = \begin{bmatrix} \tau_9 \\ \tau_{10} \\ \tau_{11} \\ \tau_{12} \\ \tau_{13} \\ \tau_{14} \\ \tau_{15} \\ \tau_{16} \end{bmatrix}. \tag{A5}$$

Here, the matrices $\hat{A}$,$\hat{B}$,$\hat{C}$, and $\hat{D}$ are of size 8×2. This is $2^{4-1} \times r$, where $r = 2$. However, the matrices such as $\hat{A}^T A$ appearing in the decoupled linear systems are of very small size 2×2 for $r = 2$. In addition, one can see that the matrices $\hat{A}^T\hat{A}$, $\hat{B}^T\hat{B}$, $\hat{C}^T\hat{C}$, and $\hat{D}^T\hat{D}$ are symmetric and positive definite.