NFN - Nationales Forschungsnetzwerk

Geometry + Simulation

http://www.gs.jku.at

Greedy low-rank approximation in Tucker format of tensors and solutions of tensor linear systems

Irina Georgieva and Clemens Hofreither

G+S Report No. 68

March 2018

Der Wissenschaftsfonds.

JOHANNES KEPLER UNIVERSITY LINZ

GREEDY LOW-RANK APPROXIMATION IN TUCKER FORMAT OF TENSORS AND SOLUTIONS OF TENSOR LINEAR SYSTEMS

I. GEORGIEVA* AND C. HOFREITHER†

Abstract. We propose a method for the approximation of tensors, given either explicitly or implicitly as the solution of tensor linear systems, in the Tucker tensor format. It is an iterative method that greedily constructs a suitable tensor product subspace in which to approximate the tensor by means of successive rank one approximations. An approximation to the target tensor is then obtained either by orthogonal projection into this subspace (in the direct approximation case) or by approximately satisfying the given linear system in the subspace (in the implicit approximation case).

In several numerical experiments, we compare the method to the greedy rank one update (or Proper Generalized Decomposition) approach. The proposed method outperforms the rank one update method significantly, both in terms of error per iteration and error per unit of computation time.

Key words. Tensor approximation, Tucker tensors, greedy algorithms, tensor linear systems, Proper Generalized Decomposition

AMS subject classifications. 15A69, 65F10, 15A23

1. Introduction.

For low-rank approximation of tensors with order d higher than two, several different formats have been described in the literature. The most straightforward approach is the approximation by sums of outer products of d vectors, sometimes referred to as the canonical or CP (CANDECOMP/PARAFAC) format; see [12] and the references therein for a historical overview and properties of this representation. Unfortunately, there are certain difficulties associated with low-rank approximation in this format for orders $d > 2$ stemming from the fact that the set of tensors with a given maximum rank R is not closed if $d > 2$, a crucial difference from the matrix case [4].

These difficulties are not merely theoretical in nature, but significantly impact the practical computation of best or "good" rank R approximations for tensors with order $d > 2$. For instance, one of the oldest and most popular algorithms for this task, the Alternating Least Squares (ALS) method, is heavily dependent on the choice of its starting values and frequently runs into problems with slow or even stagnant convergence. Many improvements to ALS and different approaches to the rank R approximation problem have been proposed (see [7] and the references therein), but at this point it seems unclear whether any single method can consistently and significantly outperform ALS with a decent choice of starting values.

In order to sidestep the inherent difficulties of the canonical rank R approximation problem, alternative formats for low-rank tensor approximation have been devised. One of the earliest of these is the so-called Tucker format [21], also known as tensor subspace representation [8]. The idea here is to work with a small subspace of the entire linear space of tensors $\mathbb{R}^{N_1 \times \ldots \times N_d}$ which is represented as a tensor product of d spaces which are subspaces of $\mathbb{R}^{N_j}$, $j = 1, \ldots, d$. These subspaces are represented as the ranges of d matrices. The coefficients of a particular tensor within the tensor subspace are stored as a smaller so-called core or coefficient tensor. If, for all j, the

*Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Acad. G. Bonchev, Bl. 8, 1113, Sofia, Bulgaria (irina@math.bas.bg)

†Institute of Computational Mathematics, Johannes Kepler University, Altenberger Str. 69, 4040 Linz, Austria (chofreither@numa.uni-linz.ac.at)

j-th subspace is small compared to $\mathbb{R}^{N_j}$, this leads to significant savings in the amount of data to be stored.

The Tucker format still suffers from an exponential increase in required storage as the order d increases. To fully break this "curse of dimensionality," more advanced tensor representations such as the hierarchical Tucker format [9, 8] and the tensor train format [17] have been developed. Nevertheless, for problems with a moderate number of dimensions, the Tucker format is often adequate and attractive due to its comparable simplicity. In our case, we have applications in mind which stem from the discretization of partial differential equations (PDEs) in tensor product spaces, in particular in Isogeometric Analysis [11] where tensor product spline spaces are employed.

A number of algorithms for approximation of tensors in Tucker format is available in the literature; see [12, 7] and the references therein for an overview. The most popular one is the so-called Higher Order Singular Value Decomposition (HOSVD) [14] combined with a truncation step, which is known to yield quasi-optimal approximations [6]. Such an approximation can be improved by the so-called Higher Order Orthogonal Iteration (HOOI) [15], essentially a variant of ALS for the Tucker format with similar problems with respect to convergence guarantees. Other methods for computing Tucker approximations are based on a Newton-Grassman optimization approach [5].

Although ALS for the best rank R approximation may converge slowly or not at all, the ALS algorithm for the best rank one approximation typically converges very fast and globally, i.e., independently of the starting value [22]. One simple and popular method which exploits this fact is the Greedy Rank One Update (GROU) strategy, where a rank R approximation is built up by iteratively computing the best rank one approximation to the current error and adding it to the previous approximation (see, e.g., [1]). This approach is popular in applications for computing low-rank approximations to the solutions of tensor linear systems under the name Proper Generalized Decomposition (PGD); see [3] and the references therein. Whereas the GROU method yields optimal approximations in the matrix case ($d = 2$), its convergence is typically very slow for tensors with orders $d > 2$.

We propose a novel greedy iterative algorithm for computing Tucker approximations of tensors which are given either explicitly or as the solution of a tensor linear system. Like GROU, it uses best rank one approximations to the current error computed by ALS as its main building block. The core idea is to greedily construct the tensor subspaces used in Tucker approximation in such a way that they contain the rank one approximations computed by ALS. The matrices which represent the tensor subspace are kept orthonormal throughout, which permits fast orthogonal projection of the tensor to be approximated into the tensor subspace.

A crucial advantage of this method is that it applies not only to approximation of explicitly given tensors, but can be straightforwardly extended to approximation of tensors which are given implicitly as the solution of linear equations with a low-rank tensor linear operator. Such equations arise naturally in tensor product discretizations of partial differential equations [16] and, in particular, recently in Isogeometric Analysis [11, 10]. In the case of such implicitly given tensors, the error of the current approximation is unknown. Instead, we use a variant of ALS for computing a best rank one minimizer of the residual of the linear equation and iteratively enrich our tensor subspace to contain this rank one tensor. Since we cannot project the (unknown) solution tensor into this space, we instead solve a small, dense linear system which can be considered a projection of the original linear system into the small tensor

subspace.

There are other methods for approximating solutions of tensor linear systems in a number of tensor formats; see [7]. In general, they are either based on applying a standard iterative method and truncating the result to a desired small rank after each iteration, or on optimization considerations, such as ALS. We are not aware of any prior publications which specifically treat the approximate solution of general tensor linear systems in Tucker format, although a projection approach is taken in [2] for the Hierarchical Tucker format.

We give some numerical examples which demonstrate the performance of the proposed method for direct approximation and for approximately solving tensor linear systems. The achieved approximation errors, relative to the spent computation time, compare very favorably to the GROU scheme. In the direct approximation case, the method produces almost optimal approximations in our tests, which is a surprising result for a greedy algorithm.

The outline of the paper is as follows. In Section 2, we review some preliminaries on tensor rank, low-rank approximation, the Tucker format and existing algorithms. In Section 3, we describe our new algorithm for greedy approximation in Tucker format. In Section 4, we develop a variant of the algorithm for approximating tensors which are implicitly given as solutions of tensor linear systems. In Section 5, we present some numerical examples and evaluate the performance of the method. We summarize the main results in Section 6 and offer an outlook on future work.

2. Preliminaries.

2.1. Rank and low-rank approximation of tensors. Assume we are given a tensor of order d and dimensions $N_1, \dots, N_d$, i.e., $B \in \mathbb{R}^{N_1 \times \cdots \times N_d} =: \mathbb{R}^{\mathcal{I}}$ with $\mathcal{I} = \{1, \dots, N_1\} \times \dots \times \{1, \dots, N_d\}$. We seek to compute approximations which require much fewer than the $N_1 \cdots N_d$ coefficients of B.

DEFINITION 2.1. *The d-dimensional* tensor product *of vectors $x^j \in \mathbb{R}^{N_j}$, $j = 1, \dots, d$ is the tensor $x^1 \otimes \cdots \otimes x^d \in \mathbb{R}^{\mathcal{I}}$ with the entries*

$$[x^1 \otimes \cdots \otimes x^d]_{i_1,\dots,i_d} = x^1_{i_1} \cdots x^d_{i_d}.$$

DEFINITION 2.2 ([13]). *The* rank *of B is the smallest $R \in \mathbb{N}_0$ such that there exist vectors $x^r_j \in \mathbb{R}^{N_j}$, $j = 1, \dots, d$, $r = 1, \dots, R$ with*

$$B = \sum_{r=1}^{R} x^r_1 \otimes \cdots \otimes x^r_d.$$

Even if B does not have low rank, it is often still possible to obtain good low-rank approximations. In the sequel, we will always use the Frobenius norm

$$\|B\|^2 = \sum_{(i_1,\dots,i_d) \in \mathcal{I}} B^2_{i_1,\dots,i_d}.$$

DEFINITION 2.3. *If X^* minimizes the error $\|B - X\|$ among all tensors $X \in \mathbb{R}^{\mathcal{I}}$ of rank at most $R \in \mathbb{N}_0$, we call X^* a* best rank R approximation *to B.*

For higher order tensors, there are certain problems with this notion of best approximation [4]. In particular, for order $d > 2$, the set of tensors of rank at most $R > 1$ is not closed, meaning that a best rank R approximation may not exist, and the problem of finding such a best approximation is ill-posed. On the other hand, for

the matrix case ($d = 2$), there always exists a best rank R approximation given by the truncated singular value decomposition (SVD). Furthermore, for any order d, a best rank 1 approximation always exists.

2.2. Multiway tensor product and Tucker tensor format.

DEFINITION 2.4. *Given d matrices and a tensor of order d,*

$$U_j \in \mathbb{R}^{N_j \times r_j},\ j = 1, \dots, d, \qquad X \in \mathbb{R}^{r_1 \times \dots \times r_d},$$

where $N_j, r_j \in \mathbb{N}$, the multiway tensor product *is the tensor*

$$T = (U_1, \dots, U_d) \cdot X \in \mathbb{R}^{N_1 \times \dots \times N_d} \tag{2.1}$$

with the entries

$$T_{i_1,\dots,i_d} = \sum_{\alpha_1=1}^{r_1} \dots \sum_{\alpha_d=1}^{r_d} [U_1]_{i_1,\alpha_1} \cdots [U_d]_{i_d,\alpha_d} X_{\alpha_1,\dots,\alpha_d}, \quad (i_1, \dots, i_d) \in \mathcal{I}.$$

Below we give two important properties of this product which are easy to derive from the definition.

- For arbitrary matrices Q_j, U_j of compatible sizes, we have

$$(Q_1, \dots, Q_d) \cdot \big((U_1, \dots, U_d) \cdot X\big) = (Q_1 U_1, \dots, Q_d U_d) \cdot X. \tag{2.2}$$

- Let $\mathcal{A} = A_1 \otimes \cdots \otimes A_d : \mathbb{R}^{\mathcal{I}} \to \mathbb{R}^{\mathcal{I}}$ be a *rank one tensor linear operator*, where $A_j : \mathbb{R}^{N_j} \to \mathbb{R}^{N_j}$ are matrices. By this we mean that the application of $\mathcal{A}$ to a rank one tensor $x_1 \otimes \dots \otimes x_d$ is given by

$$\mathcal{A}(x_1 \otimes \cdots \otimes x_d) = (A_1 x_1) \otimes \cdots \otimes (A_d x_d).$$

  In practice, such an operator is often represented as the Kronecker product of the matrices A_j. The application of $\mathcal{A}$ to a tensor $B \in \mathbb{R}^{\mathcal{I}}$ is then given by

$$\mathcal{A}B = (A_1, \dots, A_d) \cdot B. \tag{2.3}$$

A tensor $T \in \mathbb{R}^{\mathcal{I}}$ is said to be in *Tucker format* [21] or *tensor subspace representation* [8] if it has the form (2.1), and we call the tuple $(r_1, \dots, r_d)$ its *Tucker rank*. (We use this convention instead of the one used by Hackbusch [8], where only the minimal tuple allowing such a representation of a given tensor is called the Tucker rank.)

A Tucker tensor can be represented on a computer by storing only the matrices U_j and the core tensor X. Assuming $N_j = N$ and $r_j = r$, this requires the storage of $dNr + r^d$ floating point numbers. If $r \ll N$, this is significantly less than the storage N^d required for the same tensor in full tensor format. Furthermore, (2.2) gives us an efficient method for applying the multiway tensor product to a Tucker tensor, producing again a Tucker tensor. In conjunction with (2.3), this describes how to efficiently apply a rank one operator to a Tucker tensor.

The storage required for the core tensor X still grows exponentially with the order d, which makes other representations such as the hierarchical Tucker format [9, 8] or the tensor train format [17] more desirable for high-dimensional problems. Nevertheless, for problems with moderate dimensionality, the simplicity of the Tucker format makes it an attractive choice.

If we denote the columns of U_j by $U_{j,\alpha} \in \mathbb{R}^{N_j}, \alpha = 1, \ldots, r_j$, the Tucker tensor T can be written as

$$T = \sum_{\alpha_1=1}^{r_1} \ldots \sum_{\alpha_d=1}^{r_d} (U_{1,\alpha_1} \otimes \cdots \otimes U_{d,\alpha_d}) X_{\alpha_1,\ldots,\alpha_d}. \tag{2.4}$$

An interpretation of this identity is that the columns of U_j, assuming that they are linearly independent, form a basis for a r_j-dimensional subspace of $\mathbb{R}^{N_j}$. A basis for a subspace of $\mathbb{R}^{N_1 \times \cdots \times N_d}$ is given by the tensor product of these univariate bases. Each choice of $(\alpha_1, \ldots, \alpha_d)$ in (2.4) enumerates one such basis tensor, which is multiplied by the corresponding entry in the coefficient tensor X. Thus, $T \in \operatorname{range}(U_1) \otimes \cdots \otimes \operatorname{range}(U_d)$, and the core tensor X gives the coefficients for T within this subspace. This motivates the name "tensor subspace representation."

2.3. The Alternating Least Squares algorithm. A popular method for computing best rank R approximations is the Alternating Least Squares (**ALS**) algorithm. It is based on sequentially fixing all but the k-th factor and solving the resulting linear least squares problem. It can be considered a block nonlinear Gauss-Seidel iteration. The variant for the best rank one approximation problem is given in Algorithm 2.1.

Algorithm 2.1 Alternating least squares (**ALS**)

function ALS($B \in \mathbb{R}^{\mathcal{I}}$)
  choose nonzero starting values $x_j \in \mathbb{R}^{N_j}$, $j = 1, \ldots, d$
  while not converged **do**
    for $k = 1, \ldots, d$ **do**
      solve the linear least squares problem

$$x_k \leftarrow \operatorname*{arg\,min}_{y \in \mathbb{R}^{N_k}} \|B - x_1 \otimes \cdots x_{k-1} \otimes y \otimes x_{k+1} \cdots \otimes x_d\|^2$$

    end for
  end while
  return $(x_1, \ldots, x_d)$
end function

Possible stopping criteria for Algorithm 2.1 include reaching a fixed number of iterations, the reduction of the target functional to a desired level, or stagnation of the changes in the factors x_j.

We do not go into details here on how to solve the linear least squares problems which occur within Algorithm 2.1 as this is by now well established in the literature; see, e.g., [12]. For a more general situation, we describe the procedure in Section 4.1.

There exist variants of Algorithm 2.1 for the best rank R approximation problem. However, for $R > 1$ and $d > 2$, convergence is very dependent on the chosen starting values and may be very slow or even stagnate. On the other hand, Algorithm 2.1 (for the case $R = 1$) has been proved to converge globally for almost all B (see [22]), and this is conjectured to hold for all B.

2.4. The Greedy Rank One Update algorithm. As discussed above, best rank one approximations are typically significantly faster to compute (by Algorithm 2.1) than best rank R approximations for $R > 1$, often by orders of magnitude. This can be exploited to build approximation algorithms which are based on successive rank one approximations.

In particular, a greedy rank-one update strategy permits fast computation of rank R approximations to tensors, which however usually have significantly larger approximation errors than the best rank R approximation. Starting with a zero tensor, we add in each iteration the best rank one approximation to the current approximation error. This best rank one approximation can be computed using Algorithm 2.1 (**ALS**), since it typically converges globally and quickly. After R iterations of the greedy rank one update method, we obtain a rank R approximation. The entire procedure is shown in Algorithm 2.2. Note that the approximation X_k is typically not stored and returned as a full tensor, but as a list of rank one contributions.

Algorithm 2.2 Greedy rank one updates (**GROU**)

function GROU($B \in \mathbb{R}^{\mathcal{I}}$)
  let $X_0 = 0$
  for $k = 1, 2, \ldots, R$ **do**
    $(x_1, \ldots, x_d) = \mathrm{ALS}(B - X_{k-1})$
    $X_k = X_{k-1} + x_1 \otimes \cdots \otimes x_d$
  end for
  return X_R
end function

In the matrix case ($d = 2$), **ALS** will, in the k-th iteration of Algorithm 2.2, compute approximately the k-th singular pair of B. Thus, **GROU** will approximate with high precision the truncated SVD and thus typically converge quite fast.

The situation is quite different for $d > 2$, where convergence of this algorithm is often very slow, meaning that many rank one terms are required to obtain a reasonable approximation of B. On the other hand, an advantage of this algorithm is that each iteration requires relatively low computational effort. In fact, the computational time is typically completely dominated by the **ALS** step.

3. A greedy algorithm for Tucker approximation. We propose a Greedy Tucker Approximation (**GTA**) algorithm for efficiently computing approximations of the form (2.1). The idea is to greedily build up the tensor product subspace $(U_1, \ldots, U_d)$ by one dimension in each iteration. In other words, after k iterations, we have $r_1 = \ldots = r_d = k$ and $U_j \in \mathbb{R}^{N_j \times k}$. As in **GROU**, the main building block is the computation of best rank one approximations to the current error by **ALS**.

Given an input tensor $B \in \mathbb{R}^{\mathcal{I}}$, we start with an empty Tucker approximation $T = 0$. In each iteration, we enrich the previous tensor subspace in such a way that it contains the best rank one approximation of the current error $B - T$ (computed by Algorithm 2.1). In other words, given the matrices $U_1, \ldots, U_d$ from the previous iteration and a new rank one approximation $x_1 \otimes \cdots \otimes x_d \in \mathbb{R}^{\mathcal{I}}$, we wish to construct new matrices $U_1', \ldots, U_d'$ with one additional column each such that $x_j \in \mathrm{range}(U_j')$. Instead of simply appending x_j to U_j, we append its orthonormalized version,

$$U_j' = \mathrm{orth}(U_j, x_j) := [U_j, x_j'/\|x_j'\|] \quad \text{with} \quad x_j' = (I - U_j U_j^T) x_j.$$

In this way, we keep the columns of the matrices U_j orthonormal throughout. Note that if $\|x_j'\| \ll \|x_j\|$, this means that x_j is already approximately contained in $\mathrm{range}(U_j)$, and we may skip enriching U_j in this iteration. For simplicity, we do not use this modification of the algorithm in the remainder of the paper.

Since each matrix U_j has orthonormal columns, it follows from elementary prop-

erties of the tensor product that the resulting tensor product basis used in (2.4),

$$\left\{U_{1,\alpha_1} \otimes \cdots \otimes U_{d,\alpha_d} \in \mathbb{R}^{\mathcal{I}} : \alpha_j \in \{1, \ldots, r_j\}, j = 1, \ldots, d\right\}$$

is orthonormal as well. This allows us to cheaply compute the core tensor corresponding to the best approximation in the tensor subspace by the orthogonal projection

$$X = (U_1^T, \ldots, U_d^T) \cdot B \in \mathbb{R}^{k \times \cdots \times k}.$$

Since the matrices U_j are unchanged from the previous iteration except for the addition of one column, it is not necessary to recompute the entries of X which were computed in previous iterations. We then obtain a new approximation $T = (U_1, \ldots, U_d) \cdot X$. It should be noted that it is not necessary to compute T as a full tensor, but instead it may be kept only in Tucker format. The entire procedure is shown in Algorithm 3.1.

Algorithm 3.1 Greedy Tucker approximation (**GTA**)

```
function GTA(B ∈ ℝ^ℐ)
    let U_j = {}, T = 0
    for k = 1, 2, ..., R do
        (x_1, ..., x_d) = ALS(B − T)
        for j = 1, ..., d do
            U_j ← orth(U_j, x_j)
        end for
        X ← (U_1^T, ..., U_d^T) · B
        T ← (U_1, ..., U_d) · X
    end for
    return (U_1, ..., U_d), X
end function
```

We may stop the algorithm either after a fixed number of iterations (as shown in Algorithm 3.1) or when the absolute or relative error is below a desired threshold.

4. A greedy algorithm for Tucker approximation of solutions of tensor linear systems. Assume that the tensor to be approximated is not given directly, but as the solution $Z \in \mathbb{R}^{\mathcal{I}}$ of a tensor linear equation $\mathcal{A}Z = B$, where $\mathcal{A} : \mathbb{R}^{\mathcal{I}} \to \mathbb{R}^{\mathcal{I}}$ is an invertible linear operator on tensors (often of low rank itself). Such equations arise naturally, for instance, in the discretization of partial differential equations (PDEs) over tensor product grids. We give an example of such a tensor linear system in Section 5.2. Recently, there has been increased interest in efficiently solving such tensor linear systems due to the development of the Isogeometric Analysis [11] approach for the discretization of PDEs based on tensor product spline spaces; see, e.g., [10].

4.1. Best rank one minimizers of the residual. Instead of computing best rank one approximations to the error, which is not accessible since Z is not known, we compute rank one tensors which minimize the residual: find $x_j \in \mathbb{R}^{N_j}$, $j = 1, \ldots, d$, such that

$$\|B - \mathcal{A}(x_1 \otimes \cdots \otimes x_d)\| \to \min.$$

The Alternating Least Squares algorithm generalizes in a straighforward fashion to this setting, as shown in Algorithm 4.1. We denote this algorithm by ALS-LS$(\mathcal{A}, B)$.

Algorithm 4.1 Alternating least squares for linear systems (**ALS-LS**)

function ALS-LS($\mathcal{A} : \mathbb{R}^{\mathcal{I}} \to \mathbb{R}^{\mathcal{I}}$, $B \in \mathbb{R}^{\mathcal{I}}$)
  choose nonzero starting values $x_j \in \mathbb{R}^{N_j}$, $j = 1, \dots, d$
  while not converged **do**
    for $k = 1, \dots, d$ **do**
      solve the linear least squares problem

$$x_k \leftarrow \underset{y \in \mathbb{R}^{N_k}}{\arg\min} \|B - \mathcal{A}(x_1 \otimes \cdots x_{k-1} \otimes y \otimes x_{k+1} \cdots \otimes x_d)\|^2$$

    end for
  end while
  return $(x_1, \dots, x_d)$
end function

Like Algorithm 2.1, it can be extended to compute approximations with rank $R > 1$, but we do not use this variant due to the generally poor convergence.

For this algorithm to be efficient, we generally prefer the linear operator $\mathcal{A}$ itself to be of low rank. By this we mean that there exists a rank $R_{\mathcal{A}} \in \mathbb{N}$ and (possibly sparse) matrices $A_j^\rho \in \mathbb{R}^{N_j \times N_j}$, $j = 1, \dots, d$, $\rho = 1, \dots, R_{\mathcal{A}}$, such that for all $x_j \in \mathbb{R}^{N_j}$, $j = 1, \dots, d$, we have

$$\mathcal{A}(x_1 \otimes \cdots \otimes x_d) = \sum_{\rho=1}^{R_{\mathcal{A}}} (A_1^\rho x_1) \otimes \cdots \otimes (A_d^\rho x_d). \tag{4.1}$$

In order to solve the k-th inner linear least squares problem in Algorithm 4.1 with such an $\mathcal{A}$ with low rank (see, e.g., [1]), we introduce the tensor linear operator

$$\begin{aligned} \mathcal{Z}_k = \mathcal{Z} &= \sum_{\rho=1}^{R_{\mathcal{A}}} (A_1^\rho x_1) \otimes \cdots \otimes (A_{k-1}^\rho x_{k-1}) \otimes A_k^\rho \otimes (A_{k+1}^\rho x_{k+1}) \otimes \cdots \otimes (A_d^\rho x_d) \\ &= \sum_{\rho=1}^{R_{\mathcal{A}}} z_1^\rho \otimes \cdots \otimes z_{k-1}^\rho \otimes A_k^\rho \otimes z_{k+1}^\rho \otimes \cdots \otimes z_d^\rho \quad : \quad \mathbb{R}^{N_k} \to \mathbb{R}^{\mathcal{I}} \end{aligned}$$

with the vectors $z_j^\rho = A_j^\rho x_j \in \mathbb{R}^{N_j}$. Then the solution of the k-th least squares problem is given by the solution $\hat{x}_k \in \mathbb{R}^{N_k}$ of the linear equation

$$\mathcal{Z}^T \mathcal{Z} \hat{x}_k = \mathcal{Z}^T B = \sum_{\rho=1}^{R_{\mathcal{A}}} ((z_1^\rho)^T, \dots, (z_{k-1}^\rho)^T, (A_k^\rho)^T, (z_{k+1}^\rho)^T, \dots, (z_d^\rho)^T) \cdot B$$

with the matrix $\mathcal{Z}^T \mathcal{Z} \in \mathbb{R}^{N_k \times N_k}$ and the right-hand side $\mathcal{Z}^T B \in \mathbb{R}^{N_k}$, where we used property (2.3). This shows that the input tensor $B \in \mathbb{R}^{\mathcal{I}}$ to Algorithm 4.1 is not required entrywise, but only in a form which allows the computation of multiway tensor products (2.1) with it. In particular, using property (2.2), we can realize **ALS-LS** efficiently with B given as a Tucker tensor, which we will exploit in the next section.

Before we proceed to introduce the variant of **GTA** for tensor linear systems, we point out that the strategy of greedy rank one updates generalizes easily to the case of

such linear systems by replacing the rank one approximation to the current error by the rank one minimizer of the residual. Algorithm 4.2 displays this variant **GROU-LS**. If the right-hand side B is given as a canonical tensor, then it is advantageous to keep also X_k in canonical form throughout. Along the lines of the remarks made above, it is possible to realize **ALS-LS** directly and efficiently on canonical tensors. This approach has enjoyed considerable success in many applications under the name Proper Generalized Decomposition (PGD); see [3] and the references therein. Some convergence results for this greedy rank one update method are given in [1].

Algorithm 4.2 Greedy rank one updates for linear systems (**GROU-LS**)

function GROU-LS($\mathcal{A} : \mathbb{R}^{\mathcal{I}} \to \mathbb{R}^{\mathcal{I}}, B \in \mathbb{R}^{\mathcal{I}}$)
  let $X_0 = 0$
  for $k = 1, 2, \ldots, R$ **do**
    $(x_1, \ldots, x_d) =$ ALS-LS($\mathcal{A}, B - \mathcal{A}X_{k-1}$)
    $X_k = X_{k-1} + x_1 \otimes \cdots \otimes x_d$
  end for
  return X_R
end function

4.2. The algorithm GTA-LS. Based on **ALS-LS**, we develop a variant of **GTA**, which we call **GTA-LS**, for approximating solutions of tensor linear systems in Tucker format. As in Algorithm 3.1, we start with an empty Tucker tensor $T = 0$ and empty matrices (i.e., with zero columns) $U_j, j = 1, \ldots, d$. In each iteration, we compute the rank one tensor $x_1 \otimes \cdots \otimes x_d$ which minimizes the residual $\|B - \mathcal{A}(T + x_1 \otimes \cdots \otimes x_d)\|$ by Algorithm 4.1. As in Algorithm 3.1, we extend each matrix U_j with the orthonormalized x_j.

Unlike in Algorithm 3.1, we cannot compute the core tensor X as the orthogonal projection of the sought tensor Z since Z is only given implicitly. Instead, we attempt to approximately satisfy the linear equation $\mathcal{A}Z = B$ in the tensor subspace spanned by the bases $(U_1, \ldots, U_d)$. To this end, we introduce the linear operators

$$\begin{aligned} \mathcal{U} &: \mathbb{R}^{r_1 \times \cdots \times r_d} \to \mathbb{R}^{N_1 \times \cdots \times N_d}, \qquad \mathcal{U}X = (U_1, \ldots, U_d) \cdot X, \\ \mathcal{U}^T &: \mathbb{R}^{N_1 \times \cdots \times N_d} \to \mathbb{R}^{r_1 \times \cdots \times r_d}, \qquad \mathcal{U}^T Y = (U_1^T, \ldots, U_d^T) \cdot Y. \end{aligned}$$

Roughly speaking, we now seek a coefficient tensor $X \in \mathbb{R}^{r_1 \times \cdots \times r_d}$ such that

$$\mathcal{A}(\mathcal{U}X) \approx B.$$

One way to approach this is to minimize the residual,

$$\|B - \mathcal{A}\mathcal{U}X\| \to \min,$$

which leads to the linear least squares problem

$$(\mathcal{A}\mathcal{U})^T \mathcal{A}\mathcal{U}X = (\mathcal{A}\mathcal{U})^T B. \tag{4.2}$$

Another approach which is tempting when the operator $\mathcal{A}$ stems from a Galerkin discretization of a partial differential equation is based on the variational interpretation

$$\langle \mathcal{A}Z, \psi \rangle = \langle B, \psi \rangle \qquad \forall \psi \in \mathbb{R}^{\mathcal{I}},$$

where $\langle \cdot, \cdot \rangle$ denotes the Frobenius inner product. Performing a Galerkin projection of this variational problem to the tensor subspace spanned by $\mathcal{U}$, we obtain the problem: find $\varphi \in \operatorname{range}(\mathcal{U}) \subset \mathbb{R}^{\mathcal{I}}$ such that

$$\langle \mathcal{A}\varphi, \psi \rangle = \langle B, \psi \rangle \qquad \forall \psi \in \operatorname{range}(\mathcal{U}). \tag{4.3}$$

This is equivalent to finding $X \in \mathbb{R}^{r_1 \times \cdots \times r_d}$ such that

$$\langle \mathcal{A}\mathcal{U}X, \mathcal{U}Y \rangle = \langle B, \mathcal{U}Y \rangle \qquad \forall Y \in \mathbb{R}^{r_1 \times \cdots \times r_d}.$$

Thus, (4.3) is solved by $\varphi = \mathcal{U}X$ with X being the solution of the linear system

$$(\mathcal{U}^T \mathcal{A}\mathcal{U})X = \mathcal{U}^T B. \tag{4.4}$$

If $\mathcal{A}$ stems from the Galerkin discretization of a coercive and bounded bilinear form, then, by Céa's lemma, the unique solution φ is a quasi-optimal approximation to Z in the space spanned by $\mathcal{U}$. The constant of quasi-optimality depends only on the underlying variational problem (namely, on the constants in the coercivity and boundedness estimates), but not on $\mathcal{U}$. If these assumptions are not satisfied, the least squares approach (4.2) may be more appropriate.

The entire algorithm **GTA-LS** is given in Algorithm 4.3. As a stopping criterion, we can either use a fixed number of iteration steps or the reduction of the initial residual by a desired tolerance.

Algorithm 4.3 Greedy Tucker approximation for linear systems (**GTA-LS**)

function GTA-LS($\mathcal{A} : \mathbb{R}^{\mathcal{I}} \to \mathbb{R}^{\mathcal{I}}, B \in \mathbb{R}^{\mathcal{I}}$)
  let $U_j = \{\}$ for $j = 1, \ldots, d$, $X = 0$
  for $k = 1, 2, \ldots, R$ **do**
    $T \leftarrow (U_1, \ldots, U_d) \cdot X$
    $(x_1, \ldots, x_d) = \text{ALS-LS}(\mathcal{A}, B - \mathcal{A}T)$
    for $j = 1, \ldots, d$ **do**
      $U_j \leftarrow \operatorname{orth}(U_j, x_j)$
    end for
    $X \leftarrow (\mathcal{U}^T \mathcal{A}\mathcal{U})^{-1} \mathcal{U}^T B$ (or $((\mathcal{A}\mathcal{U})^T \mathcal{A}\mathcal{U})^{-1} (\mathcal{A}\mathcal{U})^T B$, see (4.2))
  end for
  return $(U_1, \ldots, U_d), X$
end function

Assuming that B is given in low-rank Tucker format, we also obtain the residual $B - \mathcal{A}T$ in low-rank Tucker format by combining properties (2.3) and (2.2). Here we require a method for adding Tucker tensors, which is described in detail in [8]. The trivial addition of Tucker tensors also increases the resulting Tucker rank additively. Therefore, we apply HOSVD projection as described in [8] to reduce the Tucker rank of the residual before invoking **ALS-LS**. In our examples we found that truncation with a relatively large relative tolerance of 10^{-2} was sufficient in order not to degrade the overall convergence rate.

In each iteration, we obtain X by solving the smaller ($r_1 \cdots r_d$ unknowns) linear system (4.2) or (4.4). The matrix to be inverted here generally is dense and has size $k^d \times k^d$ in the k-th iteration. Thus the solution of this linear system grows increasingly computationally expensive with increasing k, and we will discuss some strategies to speed up this step in Section 4.3.

4.3. Solution of the linear system for the coefficients X. In every iteration of Algorithm 4.3, we have to solve a linear system for the linear operator $\mathcal{U}^T\mathcal{A}\mathcal{U} : \mathbb{R}^{r_1\times\cdots\times r_d} \to \mathbb{R}^{r_1\times\cdots\times r_d}$ (assuming the Galerkin approach (4.4)). After matricizing this operator, and taking into account that in the k-th iteration $r_j = k$, we obtain a matrix $C \in \mathbb{R}^{k^d\times k^d}$. Since the matrices U_j have orthonormal columns, C inherits many properties from the global linear system $\mathcal{A}$ such as invertability, symmetry, and positive definiteness. If we use instead the operator $\mathcal{U}^T\mathcal{A}^T\mathcal{A}\mathcal{U}$ from the least squares approach (4.2), C is guaranteed to be symmetric and positive definite.

However, in general, C is dense even if $\mathcal{A}$ is sparse due to the density of the factor matrices U_j. This makes the direct inversion of C relatively expensive as the rank grows, as the complexity for this inversion scales with $\mathcal{O}(k^{3d})$. Although these routines are highly optimized in modern BLAS/LAPACK implementations, it is worth considering alternatives for situations where higher rank is required.

At the k-th iteration, we have the core tensor $X_{k-1} \in \mathbb{R}^{(k-1)\times\cdots\times(k-1)}$ from the previous iteration available to us. Since all the previous basis vectors in the matrices U_j remain unchanged, it seems reasonable to use X_{k-1} as the starting value of an iterative method by extending it with zeros along each coordinate axis, resulting in a tensor $\widetilde{X}_k \in \mathbb{R}^{k\times\cdots\times k}$. We then apply a few iterations of Gauss-Seidel relaxation with the matrix C and the starting values $\widetilde{X}_k$ to obtain an approximation to the solution of the linear system (assuming that the conditions for convergence of Gauss-Seidel, e.g., symmetric positive definiteness, are satisfied).

The number of Gauss-Seidel iterations should be chosen in such a way that the convergence rate of Algorithm 4.3 is not degraded. In the example in Section 5.2, we see that the convergence rate of Gauss-Seidel is bounded from above independently of the rank (and hence the size of C). This is rather surprising and motivates the use of an iterative method for the solution of this linear system. In this case, a constant number of Gauss-Seidel iterations per iteration of Algorithm 4.3 is sufficient to maintain the optimal convergence rate.

Further ways to improve the performance of this step may be sparsification of the matrix C, block preconditioning, or, since C has tensor product structure itself, the use of tensor methods also on this level. We leave these ideas for future work.

5. Numerical examples. In this section, we present numerical results from examples both for the direct and the implicit tensor approximation problem. The tests were performed on a Linux workstation with an Intel® Core™ i5-7500 CPU with 3.40GHz and 32 GB RAM. The used implementation in the Python programming language can be found on the second author's homepage[1].

5.1. Example 1: Approximation. We consider low-rank approximation of the tensor $B \in \mathbb{R}^{100\times100\times100}$ with the entries

$$B_{ijk} = 1/\sqrt{i^2+j^2+k^2}, \qquad i,j,k \in \{1,\dots,100\}.$$

This tensor was used as a benchmark problem in several previous publications [19, 18, 20]. We compare the errors in the Frobenius norm which are obtained after R iterations of Algorithm 2.2 (**GROU**) and Algorithm 3.1 (**GTA**). Recall that **GROU** results in a rank R approximation, whereas **GTA** results in a Tucker approximation with Tucker rank (R, R, R).

As an additional point of comparison, we compute the so-called higher-order SVD (HOSVD; see [14]) of B and truncate it to Tucker rank (R, R, R). Unlike in the matrix

[1] http://www.numa.uni-linz.ac.at/~chofreither/software/

case, where the truncated SVD yields the best rank R approximation, the truncated HOSVD is generally not the best approximation with a given Tucker rank. However, it is quasi-optimal up to a constant $\sqrt{d}$ (see [6]) and thus provides a very good upper bound for the best Tucker approximation error for our needs.

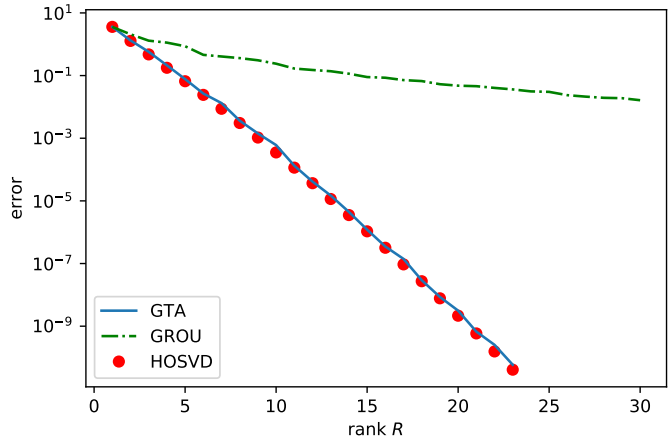

FIGURE 1. *Errors in Frobenius norm for Example 1 plotted over the rank R (i.e., the number of iterations for GTA and GROU and the truncation rank for HOSVD).*

The errors for these three methods are shown in Figure 1, plotted over the number of iterations, i.e., the rank R. Convergence of the greedy rank one update strategy is rather slow. On the other hand, the greedy Tucker approximation algorithm exhibits errors which are essentially identical to those of the truncated HOSVD, and thus to the best approximation error with a given Tucker rank. It is somewhat surprising that a greedy approximation algorithm produces almost optimal approximations.

Of course, the comparison in terms of iteration numbers does not give the full picture as algorithm **GTA** performs more work per iteration and produces larger outputs in terms of the required memory storage. Therefore, we also give the computation times together with the achieved errors in the Frobenius norm for different ranks R in Table 1. We see that the computation times grow mildly superlinearly with R for both algorithms, and **GTA** takes only by a small factor more time than **GROU**. In fact, the invokation of Algorithm 2.1 (**ALS**) requires the majority of time in both methods. As a result, **GTA** is the dramatically superior method of approximation also when considered with respect to computation time. The achieved errors are plotted over the required times in Figure 2.

5.2. Example 2: Solution of tensor linear system. We compute discretized solutions of the Poisson equation

$$-\Delta u = 1 \qquad \text{on } (0,1)^3$$

with homogeneous Dirichlet boundary conditions. We introduce a subdivision of $(0,1)$ into $n+1$ uniform intervals and construct a space of univariate, piecewise linear and continuous finite element functions over this grid. As a basis for this space, we use standard Lagrange hat functions, i.e., functions which are 1 in the j-th grid point and 0 in all others; denote this basis function by ϕ_j. We then build a tensor product basis

TABLE 1
Computation times (in seconds) and Frobenius errors for algorithms GROU and GTA in Example 1. Numbers are averaged over 25 runs.

R	time GROU	time GTA	error GROU	error GTA
1	0.017	0.017	3.572×10^{0}	3.572×10^{0}
2	0.038	0.042	2.080×10^{0}	1.327×10^{0}
3	0.055	0.079	1.301×10^{0}	5.503×10^{-1}
4	0.094	0.104	1.126×10^{0}	2.062×10^{-1}
5	0.125	0.139	8.636×10^{-1}	8.160×10^{-2}
6	0.144	0.180	4.585×10^{-1}	2.992×10^{-2}
7	0.177	0.212	4.067×10^{-1}	1.255×10^{-2}
8	0.237	0.254	3.611×10^{-1}	3.635×10^{-3}
9	0.262	0.285	3.078×10^{-1}	1.237×10^{-3}
10	0.283	0.343	2.388×10^{-1}	4.638×10^{-4}
11	0.319	0.397	1.703×10^{-1}	1.398×10^{-4}
12	0.337	0.440	1.505×10^{-1}	4.407×10^{-5}
13	0.409	0.467	1.377×10^{-1}	1.445×10^{-5}
14	0.436	0.517	1.165×10^{-1}	4.634×10^{-6}
15	0.476	0.517	9.847×10^{-2}	1.402×10^{-6}
16	0.530	0.551	8.554×10^{-2}	3.974×10^{-7}

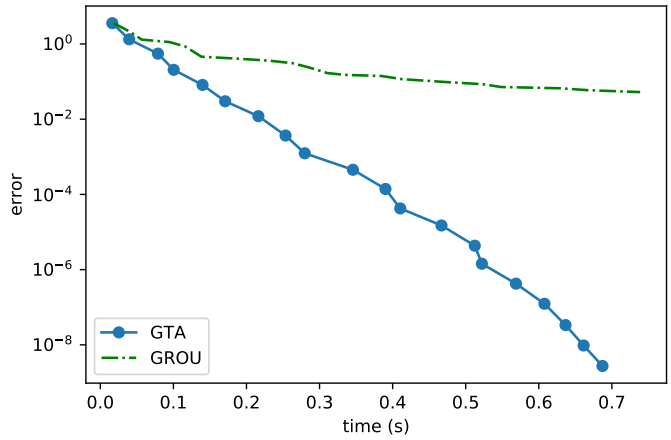

FIGURE 2. *Errors in Frobenius norm for Example 1 plotted over computation time for algorithms GTA and GROU. Numbers are averaged over 25 runs.*

of trilinear basis functions

$$\phi_{ijk}(x,y,z) = \phi_i(x)\phi_j(y)\phi_k(z)$$

over the unit cube and perform a standard Galerkin finite element discretization of the Poisson equation. By standard arguments [16], we know that the stiffness matrix $K \in \mathbb{R}^{n^3\times n^3}$ for this problem can be written as a sum of Kronecker products,

$$K = K_1 \otimes M_1 \otimes M_1 + M_1 \otimes K_1 \otimes M_1 + M_1 \otimes M_1 \otimes K_1,$$

where $M_1, K_1 \in \mathbb{R}^{n\times n}$ with the entries

$$[M_1]_{ij} = \int_0^1 \phi_i(x)\phi_j(x)\,dx, \quad [K_1]_{ij} = \int_0^1 \phi_i'(x)\phi_j'(x)\,dx, \quad i,j = 1,\dots,n$$

are the mass and stiffness matrices for the univariate bases, respectively. Here n is the number of univariate basis function after eliminating boundary degrees of freedom.

Interpreting this problem as a tensor linear system as described in Section 4, we seek a tensor $Z \in \mathbb{R}^{\mathcal{I}} = \mathbb{R}^{n\times n\times n}$ such that $\mathcal{A}Z = B$, where the application of the tensor linear operator $\mathcal{A} : \mathbb{R}^{\mathcal{I}} \to \mathbb{R}^{\mathcal{I}}$ to a rank one tensor $x_1 \otimes x_2 \otimes x_3$ with $x_j \in \mathbb{R}^n$ is given by

$$\begin{aligned}\mathcal{A}(x_1 \otimes x_2 \otimes x_3) = K_1x_1 \otimes M_1x_2 \otimes M_1x_3 + M_1x_1 \otimes K_1x_2 \otimes M_1x_3 \\ + M_1x_1 \otimes M_1x_2 \otimes K_1x_3\end{aligned}$$

and $B \in \mathbb{R}^{\mathcal{I}}$ is a constant (hence, rank one) tensor with entries $b_{ijk} = \int_\Omega \phi_{ijk}\,dx$.

We apply Algorithm 4.3 (**GTA-LS**) to this tensor linear system, using the Galerkin formulation (4.4) of the core linear system. We test both the variant with exact solution of the dense core linear system and with inexact solution by 5 Gauss-Seidel steps. For comparison, we also test the greedy rank one update method Algorithm 4.2 (**GROU-LS**). Furthermore, we compare to the truncated HOSVD of the exact solution Z. Note that for computing the HOSVD, we first need to compute the exact solution Z as a full tensor. Thus, this is not a viable method for computing the solution of large tensor linear systems and only serves as a point of reference for the best Tucker rank (R, R, R) approximation of the exact solution.

The decay of the residual norm depending on the rank R for Example 2 with $n = 50$ and $n = 200$ is shown in Figure 3. As in the direct approximation case (Example 1), the convergence of the greedy rank one update method is slow. The Greedy Tucker Approximation strategy attains rates which are much closer to the quasi-optimal approximation error of the truncated HOSVD. We point out that the variant which uses 5 Gauss-Seidel steps for the core linear system (GS) does not have significantly worse convergence rate than the one using a direct solver (DS).

Unlike in the approximation case, there is a gap between the convergence rate of **GTA-LS** and the optimal rate (which is very well approximated by truncated HOSVD). In the case $n = 50$, the geometric mean of the residual reductions from one iteration to the next is 5.19× for HOSVD (measured up to $R = 15$, where maximum accuracy is reached) and 3.25× for **GTA-LS**; thus, the latter achieves 63% of the quasi-optimal rate. In the case $n = 200$, the geometric mean of the residual reductions from one iteration to the next is 3.00× for HOSVD and 2.13× for **GTA-LS**; thus, the latter achieves 71% of the quasi-optimal rate. The quotients between the rates for the error (instead of the residual) behave very similarly. Thus, somewhat surprisingly, **GTA-LS** performs better, relatively to the best approximation error, for $n = 200$ than for $n = 50$.

The above results show that the gap is not directly related to the condition number of the linear system, which scales like $\mathcal{O}(n^2)$. Nevertheless, an interesting topic for future study is the use of preconditioners. Assume we have an operator $\mathcal{C} : \mathbb{R}^{\mathcal{I}} \to \mathbb{R}^{\mathcal{I}}$ such that the preconditioned problem

$$\mathcal{C}^{-1}\mathcal{A}Z = \mathcal{C}^{-1}B$$

is better conditioned than the original problem. For $\mathcal{C}$ having rank one, its inverse has rank one as well and can be applied very efficiently. In the ideal (but unrealistic)

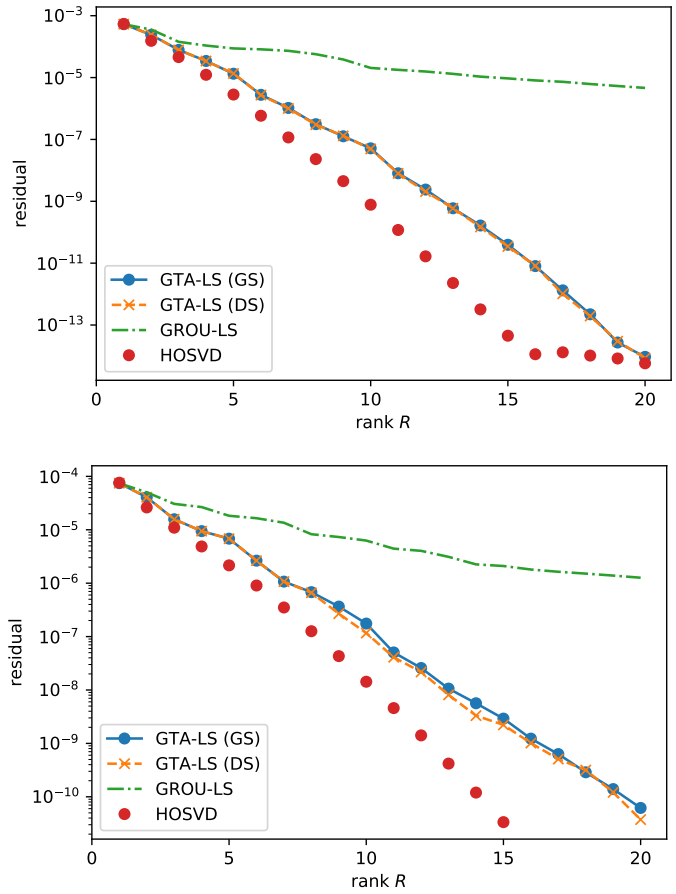

FIGURE 3. *Residual norm in Example 2 (solution of Poisson equation) plotted over the rank R. Top: $n = 50$, bottom: $n = 200$. We compare Algorithm 4.3 with direct solution of the core linear system (GTA-LS (DS)), Algorithm 4.3 with 5 Gauss-Seidel steps for the core linear system (GTA-LS (GS)), and Algorithm 4.2 (GROU-LS). In addition, we show the residuals for the truncated HOSVD, which was computed from the full exact solution. Numbers are averaged over 20 runs.*

case that $\mathcal{C}^{-1}\mathcal{A} = I$, the residual is identical to the error and we can hope for optimal convergence rates by the evidence from Example 1. If $\mathcal{C}^{-1}\mathcal{A}$ is in some sense closer to identity than $\mathcal{A}$, we can expect the convergence rates to be closer to the optimal ones, i.e., the gap to narrow.

In Figure 4, we plot the decay of the residual norm over the computation time. Again, **GROU-LS** is not competitive. Up to a certain point, as long as the dense core linear systems are small, the residual is reduced exponentially with time for both variants of **GTA-LS**. As the solution of the dense core linear system begins to dominate, we see significant differences in performance between the direct solution and the Gauss-Seidel variants, stemming from the difference in computational complexity for this step (cubic for exact solution, quadratic for Gauss-Seidel).

In order to evaluate the performance of the algorithm for varying problem size and varying condition number, we display the computation times and residual reduction factors for a constant number $R = 15$ of iterations for different choices of n in Table 2.

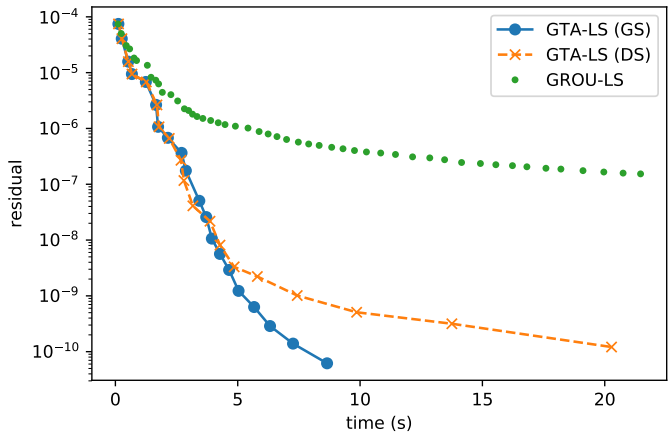

FIGURE 4. *Residual norm in Example 2 (solution of Poisson equation with $n = 200$) plotted over computation time. We compare Algorithm 4.3 with direct solution of the core linear system (GTA-LS (DS)), Algorithm 4.3 with 5 Gauss-Seidel steps for the core linear system (GTA-LS (GS)), and Algorithm 4.2 (GROU-LS). Numbers are averaged over 20 runs.*

TABLE 2

Algorithm GTA-LS with fixed rank $R = 15$ and varying problem size n in Example 2 (Poisson equation). Shown are the computation time and the reduction in Frobenius norm of the initial residual. The small core linear systems were solved directly (variant GTA-LS (DS)).

n	time	res. reduction
25	2.87 s	2.330×10^{-12}
50	3.78 s	1.023×10^{-7}
100	5.26 s	5.533×10^{-6}
200	5.49 s	3.009×10^{-5}
400	8.19 s	1.325×10^{-4}
800	8.17 s	4.025×10^{-4}
1600	11.24 s	7.450×10^{-4}

The core linear systems were solved directly here. We point out that the largest problem considered here, $n = 1600$, corresponds to a linear system with roughly 4.1 billion degrees of freedom.

We observe that the computation times increase sublinearly with n. This can be explained by the fact that in addition to steps which scale essentially linearly with n, such as the invocation of **ALS-LS**, we also have the solution of the small, dense core linear system, whose size does not depend on n. As usual, implementation issues and hardware characteristics, such as multithreading and cache sizes, also impact the time measurements.

The reduction of the residual for a fixed number of iterations decreases with n in Table 2. As the discussion of Figure 3 indicated, this effect may be mostly due to the fact that the error of the best Tucker rank (R, R, R) approximation, with fixed R, increases with n. However, preconditioning could improve these rates, as discussed above.

Finally, we present some numerical evidence that Gauss-Seidel performs well as

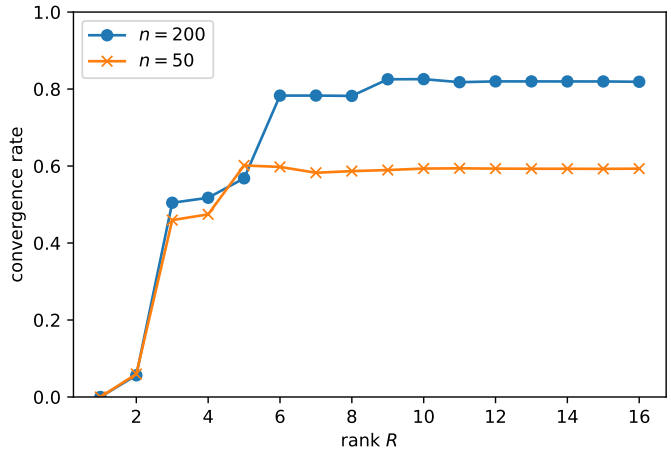

FIGURE 5. *Convergence rate of Gauss-Seidel for the dense core linear system, computed as the spectral radius $\rho(I - L_C^{-1}C)$, for Poisson problem (Example 2) with $n = 50$ and $n = 200$, plotted over the rank R. The size of the dense matrix C is $R^3 \times R^3$.*

an approximate solver for the core linear system (see Section 4.3). The convergence rate of Gauss-Seidel for a matrix C is given by the spectral radius $\rho(I-L_C^{-1}C)$, where L_C denotes the lower triangular part of C. In Figure 5, we plot these numbers for the matrices $C \in \mathbb{R}^{R^3 \times R^3}$ arising in Algorithm 4.3 after R iterations, for the Poisson equation example with $n = 50$ and $n = 200$. The rates stay bounded away from 1 in both examples, independently of the rank. This shows that a fixed number of Gauss-Seidel iterations is sufficient to solve the core linear system to an accuracy where it does not degrade the overall convergence of Algorithm 4.3. However, for increasing n, it may be necessary to increase the number of Gauss-Seidel steps. This is generally acceptable since the effort for the **ALS-LS** step of the algorithm scales roughly linearly with n, whereas the cost of one Gauss-Seidel iteration on C does not depend on n.

6. Conclusion and outlook. We have presented algorithms for greedy approximation in the Tucker tensor format of tensors which are given either explicitly or implicitly as the solutions of tensor linear systems. We have compared their performance with the greedy rank one update (GROU) strategy (also known as Proper Generalized Decomposition [3] in applications), with which they share a main building block, namely the computation of successive rank one approximations.

Both in the direct and in the implicit approximation case, the proposed methods significantly outperform the GROU algorithms, both in terms of error per iteration and error per unit of computation time.

In the implicit approximation case, our algorithm solves a small, dense linear system in each iteration, usually of size $k^d \times k^d$ in the k-th iteration. If k becomes large, this step may become a bottleneck. We have described how using Gauss-Seidel iteration instead of the direct solution can help mitigate this issue. Additional ways to improve the performance of this step, such as sparsification or using tensor methods, may be the subject of future research.

The application of the **GTA-LS** algorithm to tensor product discretizations of partial differential equations, in particular those arising in Isogeometric Analysis, will

be the topic of a forthcoming publication. An interesting issue will be the construction of suitable low-rank or rank one preconditioners in order to improve the convergence rates, as discussed in Section 5.

Acknowledgements. The authors acknowledge the support by the bilateral project DNTS-Austria 01/3/2017 (WTZ BG 03/2017), funded by Bulgarian National Science Fund and OeAD (Austria), and by Grant DN 02/14/2016, funded by the Bulgarian National Science Fund. The second author was supported by the National Research Network "Geometry + Simulation" (NFN S117), funded by the Austrian Science Fund (FWF).

REFERENCES

[1] A. Ammar, F. Chinesta, and A. Falcó, *On the convergence of a greedy rank-one update algorithm for a class of linear systems*, Archives of Computational Methods in Engineering, 17 (2010), pp. 473–486, https://doi.org/10.1007/s11831-010-9048-z.

[2] J. Ballani and L. Grasedyck, *A projection method to solve linear systems in tensor format*, Numerical Linear Algebra with Applications, 20 (2012), pp. 27–43, https://doi.org/10.1002/nla.1818.

[3] F. Chinesta, P. Ladeveze, and E. Cueto, *A short review on model order reduction based on proper generalized decomposition*, Archives of Computational Methods in Engineering, 18 (2011), pp. 395–404, https://doi.org/10.1007/s11831-011-9064-7.

[4] V. de Silva and L.-H. Lim, *Tensor rank and the ill-posedness of the best low-rank approximation problem*, SIAM Journal on Matrix Analysis and Applications, 30 (2008), pp. 1084–1127, https://doi.org/10.1137/06066518x.

[5] L. Eldén and B. Savas, *A Newton-Grassmann method for computing the best multilinear rank-(r_1, r_2, r_3) approximation of a tensor*, SIAM Journal on Matrix Analysis and Applications, 31 (2009), pp. 248–271, https://doi.org/10.1137/070688316.

[6] L. Grasedyck, *Hierarchical singular value decomposition of tensors*, SIAM Journal on Matrix Analysis and Applications, 31 (2010), pp. 2029–2054, https://doi.org/10.1137/090764189.

[7] L. Grasedyck, D. Kressner, and C. Tobler, *A literature survey of low-rank tensor approximation techniques*, GAMM-Mitteilungen, 36 (2013), pp. 53–78.

[8] W. Hackbusch, *Tensor Spaces and Numerical Tensor Calculus*, Springer Series in Computational Mathematics, Springer Berlin Heidelberg, 2012.

[9] W. Hackbusch and S. Kühn, *A new scheme for the tensor representation*, Journal of Fourier Analysis and Applications, 15 (2009), pp. 706–722, https://doi.org/10.1007/s00041-009-9094-9.

[10] C. Hofreither, *A black-box low-rank approximation algorithm for fast matrix assembly in isogeometric analysis*, Computer Methods in Applied Mechanics and Engineering, 333 (2018), pp. 311–330, https://doi.org/10.1016/j.cma.2018.01.014.

[11] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs, *Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement*, Computer Methods in Applied Mechanics and Engineering, 194 (2005), pp. 4135–4195, https://doi.org/10.1016/j.cma.2004.10.008.

[12] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Review, 51 (2009), pp. 455–500, https://doi.org/10.1137/07070111X.

[13] J. B. Kruskal, *Rank, decomposition, and uniqueness for 3-way and n-way arrays*, in Multiway data analysis, R. Coppi and S. Bolasco, eds., Elsevier, North-Holland, Amsterdam, 1989, pp. 7–18.

[14] L. D. Lathauwer, B. D. Moor, and J. Vandewalle, *A multilinear singular value decomposition*, SIAM Journal on Matrix Analysis and Applications, 21 (2000), pp. 1253–1278, https://doi.org/10.1137/S0895479896305696.

[15] L. D. Lathauwer, B. D. Moor, and J. Vandewalle, *On the best rank-1 and rank-$(r_1, r_2, \ldots, r_n)$ approximation of higher-order tensors*, SIAM Journal on Matrix Analysis and Applications, 21 (2000), pp. 1324–1342, https://doi.org/10.1137/S0895479898346995.

[16] R. E. Lynch, J. R. Rice, and D. H. Thomas, *Tensor product analysis of partial difference equations*, Bulletin of the American Mathematical Society, 70 (1964), pp. 378–384.

[17] I. V. Oseledets, *Tensor-train decomposition*, SIAM Journal on Scientific Computing, 33 (2011), pp. 2295–2317, https://doi.org/10.1137/090752286.

[18] I. V. Oseledets, D. V. Savostianov, and E. E. Tyrtyshnikov, *Tucker dimensionality*

reduction of three-dimensional arrays in linear time, SIAM Journal on Matrix Analysis and Applications, 30 (2008), pp. 939–956, https://doi.org/10.1137/060655894.

[19] I. V. Oseledets and D. V. Savost'yanov, *Minimization methods for approximating tensors and their comparison*, Computational Mathematics and Mathematical Physics, 46 (2006), pp. 1641–1650, https://doi.org/10.1134/S0965542506100022.

[20] H. D. Sterck and K. Miller, *An adaptive algebraic multigrid algorithm for low-rank canonical tensor decomposition*, SIAM Journal on Scientific Computing, 35 (2013), pp. B1–B24, https://doi.org/10.1137/110855934.

[21] L. R. Tucker, *Some mathematical notes on three-mode factor analysis*, Psychometrika, 31 (1966), pp. 279–311, https://doi.org/10.1007/BF02289464.

[22] L. Wang and M. T. Chu, *On the global convergence of the alternating least squares method for rank-one approximation to generic tensors*, SIAM Journal on Matrix Analysis and Applications, 35 (2014), pp. 1058–1072, https://doi.org/10.1137/130938207.