Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

An algorithm for low-rank approximation of bivariate functions using splines

I. Georgieva [a], C. Hofreither [b,*]

[a] Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Acad. G. Bonchev St., Bl. 8, 1113, Sofia, Bulgaria
[b] Institute of Computational Mathematics, Johannes Kepler University Linz, Altenbergerstr. 69, 4040 Linz, Austria

ARTICLE INFO

Article history:
Received 11 December 2015
Received in revised form 22 March 2016

Keywords:
Low-rank approximation
Splines
Cross approximation
Bivariate approximation
Isogeometric analysis

ABSTRACT

We present an algorithm for the approximation of bivariate functions by "low-rank splines", that is, sums of outer products of univariate splines. Our approach is motivated by the Adaptive Cross Approximation (ACA) algorithm for low-rank matrix approximation as well as the use of low-rank function approximation in the recent extension of the `chebfun` package to two dimensions. The resulting approximants lie in tensor product spline spaces, but typically require the storage of far fewer coefficients than tensor product interpolants. We analyze the complexity and show that our proposed algorithm can be efficiently implemented in terms of the cross approximation algorithm for matrices using either full or row pivoting.

We present several numerical examples which show that the performance of the algorithm is reasonably close to the best low-rank approximation using truncated singular value decomposition and leads to dramatic savings compared to full tensor product spline interpolation.

The presented algorithm has interesting applications in isogeometric analysis as a data compression scheme, as an efficient representation format for geometries, and in view of possible solution methods which operate on tensor approximations.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction

In recent years, low-rank and tensor approximation methods have increasingly found applications to many problems in numerical analysis and scientific computing. In boundary element methods (BEM), it was found that the use of so-called $\mathcal{H}$-matrices for the data-sparse representation of otherwise dense BEM matrices allows for the quasi-optimal realization of the BEM in three dimensions. The core idea is the hierarchical decomposition of the computational domain and the approximation of the resulting submatrices in low-rank form by the so-called Adaptive Cross Approximation (ACA) algorithm. These approximations have been extensively studied, and we can cite here only a few publications by Hackbusch [1], Hackbusch and Khoromskij [2], and Bebendorf [3]. The idea of low-rank approximation has been extended to tensors of orders higher than two, and the resulting tensor approximation schemes have been applied very successfully to many computational problems. We refer to the survey [4] and the monograph [5] as well as the references therein for an overview of this rapidly expanding field.

Isogeometric analysis (IGA) is a discretization method for partial differential equations introduced in [6] based on the idea that geometry representations from industry-standard CAD systems in terms of tensor product spline spaces should

* Corresponding author.
E-mail addresses: irina@math.bas.bg (I. Georgieva), chofreither@numa.uni-linz.ac.at (C. Hofreither).

http://dx.doi.org/10.1016/j.cam.2016.03.023
0377-0427/© 2016 Elsevier B.V. All rights reserved.

be used directly in analysis, and also solution fields should be represented in such spline spaces. This approach has become increasingly popular in the last decade, and tensor product B-spline and NURBS spaces still seem to be the most popular choices as approximation spaces in the IGA literature.

Therefore, one motivation of the present work is to study to what extent low-rank approximation methods can yield gains in computational efficiency in isogeometric analysis. As a first step, we study here the approximation of functions using low-rank approximations, which is important both to give an idea of the savings that can be expected and as a prerequisite for the future development of PDE solvers operating on tensor approximations directly. We restrict ourselves to the two-dimensional case, where we do not have to use the more sophisticated models for high-dimensional tensor approximation and can instead rely on simple sums of rank 1 matrices.

Some related work is given by Townsend and Trefethen [7] who describe an extension of the `chebfun` software to two dimensions. `chebfun` is a Matlab package for numerical computation with functions based on polynomial interpolation in Chebyshev nodes. The two-dimensional extension, `chebfun2`, is based on low-rank approximation of bivariate functions and representing the univariate factors by the techniques developed in `chebfun`. Their work also contains an overview of related techniques for low-rank approximation of functions; see [7, Section 2.3].

We base our algorithm on similar ideas as `chebfun2`, however we use splines instead of polynomial interpolation as the underlying univariate representation. Furthermore, we show that the use of a fixed tensor grid leads to advantages in the computational realization of the algorithm. We also give an algorithm using row pivoting rather than the full pivoting used in [7], which leads to dramatic time savings in large-scale approximation problems with only modestly increased error.

The remainder of the paper is structured as follows.

In Section 2, we collect some preliminaries on low-rank approximation of bivariate functions, in particular known results on best approximation by the truncated singular value decomposition, as well as on B-splines. We also state the cross approximation algorithm for low-rank approximation of matrices.

In Section 3, we develop our algorithm for low-rank approximation of bivariate functions. We base our algorithm on the translation of the ACA algorithm to functions and then introduce a discrete version of this algorithm by means of spline interpolation. We show how the resulting method can be efficiently realized in terms of the matrix ACA algorithm and a postprocessing step by spline interpolation. We give an analysis of the computational complexity.

In Section 4, we present several numerical examples. In particular, we study how close the results from the cross approximation algorithm are to the best possible approximation by truncated singular value decomposition, and we compare the errors obtained from full pivoting and the more efficient row pivoting. We also make some observations on how the choice of the underlying spline space influences the approximation quality of the algorithm.

In Section 5, we summarize the obtained results and discuss possible applications and further developments. We also compare our approach to prior work.

2. Preliminaries

2.1. Low-rank approximation of bivariate functions and the singular value decomposition

In this section, we summarize some results on the low-rank approximation of bivariate functions and in particular the best approximation result by the truncated singular value decomposition of a compact operator. In the matrix case, the fact that the best low-rank approximation can be obtained from the singular value decomposition is a classical result due to Eckart and Young [8]. The results for bivariate functions given here are the straightforward generalization of the matrix case using the spectral theory of compact operators on Hilbert spaces.

Let

$$\Omega = (-1, 1)^2, \qquad f : \Omega \to \mathbb{R}.$$

Our aim is to find univariate functions and coefficients

$$u_k, v_k \in L_2(-1, 1) \quad \text{and} \quad \sigma_k \in \mathbb{R} \quad \forall k \in \{1, \dots, K\}$$

such that f is close in some sense to the *low-rank approximation*

$$f_K(x, y) = \sum_{k=1}^{K} \sigma_k (u_k \otimes v_k)(x, y) := \sum_{k=1}^{K} \sigma_k u_k(x) v_k(y). \tag{1}$$

The best possible approximation for a given rank K is given by the truncated singular value decomposition (SVD). Assuming that $f \in L_2(\Omega)$, we introduce the integral operator

$$\begin{aligned} \mathcal{S} : L_2(-1, 1) &\to L_2(-1, 1) \\ u &\mapsto \int_{-1}^{1} f(x, \cdot) u(x)\, dx. \end{aligned}$$

This is a Hilbert–Schmidt integral operator and thus compact, and therefore admits a singular value decomposition (see, e.g., [9]). In other words, there exist two orthonormal bases of $L_2(-1, 1)$ denoted by $\{u_k : k \in \mathbb{N}\}$ and $\{v_k : k \in \mathbb{N}\}$ as well as singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$ tending to 0 such that

$$\mathcal{S}w = \sum_{k=1}^{\infty} \sigma_k \langle w,\ u_k \rangle v_k.$$

In particular, we have $\mathcal{S}u_\ell = \sigma_\ell v_\ell$. It follows that the expansion of $f \in L_2(\Omega)$ in the orthonormal basis $(u_k \otimes v_\ell)_{k,\ell \in \mathbb{N}}$ has the coefficients $(\delta_{k,\ell}\sigma_k)_{k,\ell}$, and hence

$$f(x, y) = \sum_{k \in \mathbb{N}} \sigma_k u_k(x) v_k(y).$$

By Parseval's identity,

$$\|f\|^2_{L_2(\Omega)} = \sum_{k \in \mathbb{N}} \sigma_k^2,$$

and the best rank-K approximation with respect to the L_2-norm is given by the truncated SVD

$$f_K(x, y) := \sum_{k=1}^{K} \sigma_k u_k(x) v_k(y)$$

with error

$$\|f - f_K\|^2_{L_2(\Omega)} = \sum_{k=K+1}^{\infty} \sigma_k^2. \tag{2}$$

2.2. B-splines and tensor product B-splines

We fix some spline degree $p \in \mathbb{N}$ and assume that we have a knot vector

$$\underline{t} = (t_1, t_2, \ldots, t_{N+p+1})$$

of monotonically increasing knots, $t_j \leq t_{j+1}$, where both the first knot $-1 = t_1 = \cdots = t_{p+1}$ and the last knot $1 = t_{N+1} = \cdots = t_{N+p+1}$ are repeated $p + 1$ times each. Furthermore, we assume that no interior knot is repeated more than p times. Such a knot vector is referred to as *open*.

Over this knot vector, we introduce a B-spline basis of degree p (or order $p + 1$). There are various ways to define the B-splines; we give here a recurrence formula for $B_{i,p}$, the ith normalized B-spline of degree p (see [10,11]).

$$B_{i,0}(t) = \begin{cases} 1, & \text{if } t \in [t_i, t_{i+1}) \\ 0, & \text{otherwise} \end{cases}$$

$$B_{i,p}(t) = \frac{t - t_i}{t_{i+p} - t_i} B_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} B_{i+1,p-1}(t).$$

The set $\{B_{1,p}, \ldots, B_{N,p}\}$ is then the B-spline basis of degree p over the knot vector $\underline{t}$. It satisfies the partition of unity property. Each B-spline is a nonnegative polynomial of degree p in each nonempty knot span $[t_i, t_{i+1})$. Furthermore, the support of $B_{i,p}$ is $[t_i, t_{i+p+1}]$.

We now choose a unisolvent system of interpolation nodes $\underline{\xi} = \{\xi_1, \ldots, \xi_N\}$ for the B-spline basis. That is to say that for arbitrary given values $(y_i)_{i=1}^N$, we can find a unique interpolating spline $v \in \text{span}\{B_{1,p}, \ldots, B_{N,p}\}$ such that $v(\xi_i) = y_i$ for all $i = 1, \ldots, N$. The following theorem states that it is sufficient to choose the ith interpolation node in the interior of the support of the ith B-spline function.

Theorem 1 (*Schönberg–Whitney [12]*). *Let $\underline{\xi} = \{\xi_1, \ldots, \xi_N\}$ be strictly increasing. Then the nodes $\underline{\xi}$ are unisolvent for $\{B_{i,p}\}_{i=1}^N$ if and only if*

$$B_i(\xi_i) \neq 0 \quad \forall i = 1, \ldots, N.$$

In the following, we always choose the interpolation nodes

$$\xi_i = \tfrac{1}{p}(t_{i+1} + \cdots + t_{i+p}), \quad i = 1, \ldots, N,$$

also known as *Gréville abscissae*, which are unisolvent due to Theorem 1.

We denote by

$$\mathcal{B} := \text{span}\{B_{i,p} \otimes B_{j,p} : i, j \in \{1, \ldots, N\}\}$$

the space of tensor product splines of degree p over Ω with dimension N^2, given in terms of the tensor product B-spline basis. The tensor product of interpolation nodes

$$\Xi = \underline{\xi} \otimes \underline{\xi} = \{(\xi_i, \xi_j) : i, j \in \{1, \dots, N\}\} \subset \Omega$$

is unisolvent for $\mathcal{B}$. Therefore, for arbitrary given values $(y_{ij})_{i,j=1}^N$, we can find a unique interpolating tensor product spline $v \in \mathcal{B}$ such that $v(\xi_i, \xi_j) = y_{ij}$ for all $i, j = 1, \dots, N$.

2.3. *Cross approximation of matrices*

When applied to a matrix, the truncated SVD described in Section 2.1 gives the best possible approximation for a rank K approximation in both the Frobenius and spectral norms [8]. However, the SVD is rather expensive to compute, requiring $\mathcal{O}(N^3)$ operations for a matrix of size $N \times N$. Therefore, more efficient low-rank approximation algorithms have been developed which generally do not yield the best possible approximation, but often a reasonably close result, and run faster than the SVD. The following algorithm is well-known in the literature and typically referred to as "Cross Approximation" or "Adaptive Cross Approximation" (ACA). See for instance the monographs [3,5] for the statement of the algorithm and some analytical results.

In the following, we let $\langle n \rangle = \{1, \dots, n\}$ and write $A_{I,J} \in \mathbb{R}^{|I|\times|J|}$ for the submatrix of A formed by taking the rows from the index set I and the columns from the index set J.

Algorithm 1 (*Cross Approximation with Row Pivoting*)**.** Given is a matrix $A \in \mathbb{R}^{m\times n}$. Let $E^{(0)} = A$ and $i_1 = 1$. For $k = 1, 2, \dots,$ choose j_k such that

$$j_k = \operatorname*{argmax}_{j\in\langle n\rangle} |E^{(k-1)}_{i_k,j}|.$$

(If the i_kth row is zero and hence the maximum above is zero, increase i_k by one until we find a nonzero row. If it happens that $i_k > m$ in this search, reset i_k to 1.) Then choose

$$u_k = E^{(k-1)}_{\langle m\rangle,j_k} \in \mathbb{R}^m, \qquad v_k = \left(E^{(k-1)}_{i_k,\langle n\rangle}\right)^\top \in \mathbb{R}^n$$

and set

$$E^{(k)} = E^{(k-1)} - \sigma_k u_k v_k^\top$$

with

$$\sigma_k = 1/E^{(k-1)}_{i_k,j_k}.$$

Choose the next row as $i_{k+1} = \operatorname{argmax}_{i\in\langle m\rangle} |(u_k)_i|$. Then increase k and repeat.

We terminate the iteration once we find that the Kth error $E^{(K)}$ is small enough in some suitable sense. The low-rank approximation A_K to A is then given by

$$A_K = \sum_{k=1}^K \sigma_k u_k v_k^\top.$$

Remark 1. As written, Algorithm 1 has space complexity $\mathcal{O}(mn + K\max\{m, n\})$ and time complexity $\mathcal{O}(Kmn)$ due to the full update of the error matrix $E^{(k)}$ in every iteration. This is faster than computing the SVD for $K < \min\{m, n\}$, but still not tractable for large matrices. However, in every iteration, the algorithm actually needs only the matrix entries in a single row and a single column of $E^{(k)}$. For large matrices, it is therefore more efficient not to compute and store $E^{(k)}$ explicitly, but to use the expression

$$E^{(k)} = A - A^{(k)} = A - \sum_{j=1}^k \sigma_j u_j v_j^\top$$

for computing the needed rows and columns of $E^{(k)}$ on the fly. Thus we do not even need to store the matrix A; instead, it is sufficient to be able to generate its rows and columns as needed. The resulting algorithm runs in $\mathcal{O}(K^2\max\{n, m\})$ and has space complexity $\mathcal{O}(K\max\{m, n\})$. This reduction in complexity is the main motivation for using row pivoting over full pivoting, since the latter will always require at least $\mathcal{O}(Kmn)$ operations.

3. Cross approximation of bivariate functions by spline interpolation

3.1. *Motivation: cross approximation of bivariate functions*

As motivation, we consider the following procedure which is the direct translation of Algorithm 1 to bivariate functions. We point out that this is the same algorithm that is used as a starting point in [7, p. C498].

Let $e_0 = f$. For $k = 1, 2, \ldots$, we find a pivot point by full pivoting,

$$(x_k, y_k) = \operatorname*{argmax}_{(x,y)\in\Omega} |e_{k-1}(x, y)|,$$

or by row pivoting analogously to the matrix algorithm. Then set

$$e_k(x, y) = e_{k-1}(x, y) - \frac{e_{k-1}(x, y_k)e_{k-1}(x_k, y)}{e_{k-1}(x_k, y_k)}.$$

Note that this choice ensures that

$$e_k(x_k, y) = e_k(x, y_k) = 0 \quad \forall(x, y) \in \Omega,$$

and in fact it is easy to prove that this property is preserved throughout all iterations, i.e.,

$$e_k(x_\ell, y) = e_k(x, y_\ell) = 0 \quad \forall(x, y) \in \Omega \text{ and } \ell \le k.$$

A low-rank approximation to f is then given by (1) with the choices

$$\sigma_k = 1/e_{k-1}(x_k, y_k), \qquad u_k(x) = e_{k-1}(x, y_k), \qquad v_k(y) = e_{k-1}(x_k, y).$$

This procedure is difficult to realize in practice. Finding the maximum over the error function might be intractable, and storing the univariate functions u_k and v_k exactly is not possible in general. Therefore, in the sequel we describe an algorithm which considers the values of f only in a pre-determined grid and stores spline interpolants of u_k and v_k.

3.2. Cross approximation by spline interpolation

Let

$$\Phi = \operatorname{span}\{\phi_1, \phi_2, \ldots, \phi_N\} \subset L_2(-1, 1)$$

denote a spline space with a B-spline basis $\phi_i = B_{i,p}$ of degree p with unisolvent point set $\underline{\xi} = \{\xi_1, \ldots, \xi_N\}$ as introduced in Section 2.2. In other words, for arbitrary given values $(y_i)_{i=1}^N$, there exists a unique spline $v \in \Phi$ with $v(\xi_i) = y_i$ for all $i = 1, \ldots, N$.

Everything that follows can be immediately generalized to the case where we use different spaces in x- and in y-directions, but to simplify the notation we use the same space Φ in both directions.

Algorithm 2. Let $e_0 = f$ and $x_1 = \xi_1$. For $k = 1, 2, \ldots$, find a row pivot element by

$$y_k = \operatorname*{argmax}_{y\in\underline{\xi}} |e_{k-1}(x_k, y)|.$$

(If the maximum is zero, set x_k to the next node from $\underline{\xi}$ until a non-zero row is found.) Then find spline interpolants $u_k, v_k \in \Phi$ such that

$$u_k(\xi_i) = e_{k-1}(\xi_i, y_k) \quad \forall i = 1, \ldots, N,$$
$$v_k(\xi_i) = e_{k-1}(x_k, \xi_i) \quad \forall i = 1, \ldots, N,$$

and set

$$e_k(x, y) = e_{k-1}(x, y) - \sigma_k u_k(x)v_k(y) \tag{3}$$

with

$$\sigma_k = \frac{e_{k-1}(x_k, y_k)}{u_k(x_k)v_k(y_k)} = 1/e_{k-1}(x_k, y_k).$$

Then set x_{k+1} to the next node from $\underline{\xi}$ after x_k, increase k, and repeat.

We terminate the iteration once we find that the Kth error function e_K is small enough. The low-rank approximation f_K to f is then given by (1).

Remark 2. Note that Algorithm 2 constructs a series of pivots $(x_k, y_k) \in \Xi$ by maximizing the magnitude of the pivot element over a single row in each iteration. The corresponding algorithm with full pivoting is obtained if one instead maximizes the pivot over both dimensions in each step, i.e.,

$$(x_k, y_k) = \operatorname*{argmax}_{(x,y)\in\Xi} |e_{k-1}(x, y)| \quad \forall k = 1, \ldots, K.$$

Algorithm 2 satisfies a similar property as the procedure from Section 3.1 concerning preservation of zeros, however now only on discrete crosses formed from the grid points Ξ. To be precise, we have the following statement.

Lemma 1. *The kth error function in Algorithm 2 vanishes on all rows and columns of the point grid Ξ on which we have chosen a pivot element, that is,*

$$e_k(x_\ell, \xi_j) = e_k(\xi_j, y_\ell) = 0 \quad \forall j \in \{1, \dots, N\} \text{ and } \ell \le k.$$

Proof. By construction, we have for all $j = 1, \dots, N$ that

$$\sigma_k u_k(x_k) v_k(\xi_j) = e_{k-1}(x_k, \xi_j) \quad \text{and} \quad \sigma_k u_k(\xi_j) v_k(y_k) = e_{k-1}(\xi_j, y_k).$$

Due to the definition of e_k in (3), we obtain

$$e_k(x_k, \xi_j) = e_k(\xi_j, y_k) = 0 \quad \forall j = 1, \dots, N. \tag{4}$$

We now consider the next update, where we see that

$$u_{k+1}(x_k) = e_k(x_k, y_{k+1}) = 0 \quad \text{and} \quad v_{k+1}(y_k) = e_k(x_{k+1}, y_k) = 0$$

due to (4), and it follows that

$$e_{k+1}(x_k, \xi_j) = e_{k+1}(\xi_j, y_k) = 0 \quad \forall j = 1, \dots, N.$$

By induction, the statement of the lemma follows. □

Remark 3. For each of the interpolants u_k and v_k, we denote their coefficient vectors with respect to the basis $\{\phi_i\}$ by $\underline{u}_k, \underline{v}_k \in \mathbb{R}^N$. Then we have

$$\begin{aligned} f_K(x, y) &= \sum_{k=1}^{K} \left(\sum_{i=1}^{N} \underline{u}_{k,i} \phi_i(x) \right) \left(\sum_{j=1}^{N} \underline{v}_{k,j} \phi_j(y) \right) \\ &= \sum_{i,j=1}^{N} \phi_i(x) \phi_j(y) \left(\sum_{k=1}^{K} \underline{u}_{k,i} \underline{v}_{k,j} \right) = \sum_{i,j=1}^{N} \phi_i(x) \phi_j(y) \left[\sum_{k=1}^{K} \underline{u}_k \underline{v}_k^\top \right]_{i,j}. \end{aligned}$$

This shows that f_K lies in the tensor product spline space $\mathcal{B}$ and has the coefficient matrix $\sum_{k=1}^{K} \underline{u}_k \underline{v}_k^\top$ with rank at most K.

The above remark together with Lemma 1 implies that Algorithm 2 converges to the exact tensor product interpolant after at most N iterations. However, we can usually achieve a good approximation already after $K \ll N$ iterations and thus obtain a low-rank approximation.

3.3. *Reduction to the matrix approximation algorithm*

We observe that Algorithm 2 depends only on the matrix

$$F := f(\Xi) := (f(\xi_i, \xi_j))_{i,j=1}^{N}$$

of function values of f in Ξ. In fact, we have $e_0(\Xi) = f(\Xi)$, and then by construction $e_k(\Xi) = e_{k-1}(\Xi) - \sigma_k c_k r_k^\top$, where c_k and r_k are a column and a row, respectively, of $e_{k-1}(\Xi)$. The functions u_k and v_k are the interpolants of the vectors c_k and r_k, respectively, at the nodes $\underline{\xi}$.

In this formulation, it becomes clear that we never need the function values of u_k and v_k in any other points but $\underline{\xi}$, and these values are given by c_k and r_k. Thus, Algorithm 2 is equivalent to applying Algorithm 1 to F, resulting in the low-rank approximation

$$F_K = \sum_{k=1}^{K} c_k r_k^\top$$

and finally, as a post-processing step, computing the spline interpolants $u_k, v_k \in \Phi$ as

$$u_k(\xi_i) = c_{k,i}, \qquad v_k(\xi_i) = r_{k,i} \quad \forall k = 1, \dots, K, \; i = 1, \dots, N.$$

The above discussion leads to a very efficient implementation of the approximation procedure. As discussed in Remark 1, it is not needed to compute the entire matrix F beforehand; instead one can compute single rows and columns of it as needed by the cross approximation algorithm. Note that the same is not true, in general, for the coefficient matrix of the tensor product interpolant of f, which may depend on all entries of F.

The main part of the algorithm is the computation of the low-rank matrix F_K. We point out that the computational effort for this step does not depend on the spline degree p, but only on the dimension N of the used spline space Φ. Only the final interpolation step to obtain u_k and v_k becomes more expensive when increasing the degree.

As per the discussion in Remark 1, the matrix ACA step of the algorithm has time complexity $\mathcal{O}(K^2N)$ when using row pivoting. For the post-processing step, we need to solve $2K$ interpolation problems for splines with N unknowns and degree p. Thus we need to invert a band matrix with size N and bandwidth p, which can be done using banded LU factorization in $\mathcal{O}(Np^2)$ operations (assuming $p \ll N$; cf. [13]). The backward substitution then only requires $\mathcal{O}(Np)$ operations per interpolation problem to be solved. Thus our algorithm has total complexity $\mathcal{O}(N(K^2 + p^2 + Kp))$, which is linear in N.

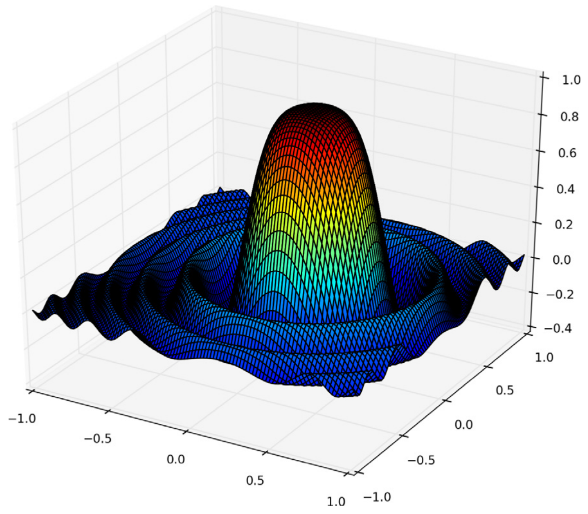

Fig. 1. (Example 1) Mexican hat function f.

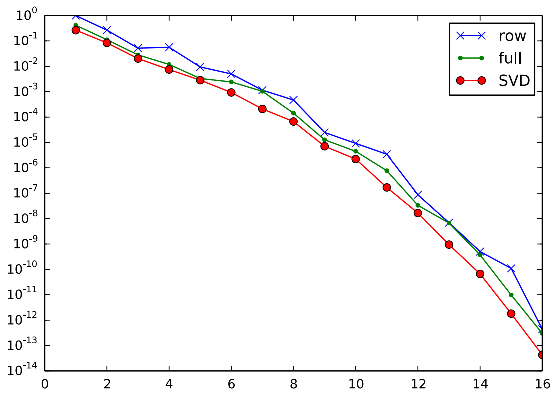

Fig. 2. (Example 1) Comparison of error $\|F - F_K\|_C$ for approximation of the matrix F by truncated SVD, cross approximation with full pivoting, cross approximation with row pivoting for different ranks.

4. Numerical experiments

Below we consider the approximation of three different functions by the proposed algorithm.

4.1. *Example 1*

We consider the off-center Mexican hat function in $(-1, 1)^2$ (see Fig. 1) given by

$$f(x, y) = \frac{\sin(5\pi((x - 0.2)^2 + y^2))}{5\pi((x - 0.2)^2 + y^2)}$$

and fix a uniform interpolation grid Ξ of 100×100 nodes. Let $F = f(\Xi)$ denote the matrix of function values. We determine the approximation error of full and row pivoted cross approximation of F compared to the best possible approximation by truncated SVD.

Fig. 2 shows the maximum matrix entry error $\|F - F_K\|_C$ for different ranks K by these three approximation methods. (By $\|\cdot\|_C$ we denote the Chebyshev matrix norm, i.e., the maximum entrywise error.) The plot indicates that both cross approximation methods achieve an asymptotically optimal error. Keeping in mind that row pivoting is much cheaper in terms of computational complexity, this makes cross approximation with row pivoting seem very attractive in practice. The results in the Frobenius norm are very similar, and we do not reproduce them here.

Next we set up spline spaces $\Phi(N, p)$ of different degrees p and dimensions N using uniform spacing h between the knots and compute low-rank spline approximations to f by the fast cross approximation algorithm described in Section 3.3 with full or row pivoting. In each run, the rank K at which to terminate the approximation algorithm was chosen as $K(N, p)$ such that the L_2-error $\|f - f_{K(N,p)}\|_{L_2(\Omega)}$ was at most 5% higher than the interpolation error in the full tensor product spline space

Table 1
(Example 1) Optimal ranks K to achieve L_2-error within 5% of the tensor product interpolant for truncated SVD approximation, ACA with full pivoting, ACA with row pivoting.

	h^{-1}	25	50	100	200	400	800
	SVD	3	4	6	6	7	8
$p = 1$	Full	4	5	6	7	8	9
	Row	5	6	7	7	9	9
	SVD	4	6	8	9	10	11
$p = 2$	Full	4	7	8	10	11	12
	Row	5	7	9	10	12	12
	SVD	5	7	9	11	12	13
$p = 3$	Full	5	8	9	12	13	13
	Row	6	9	10	12	12	14

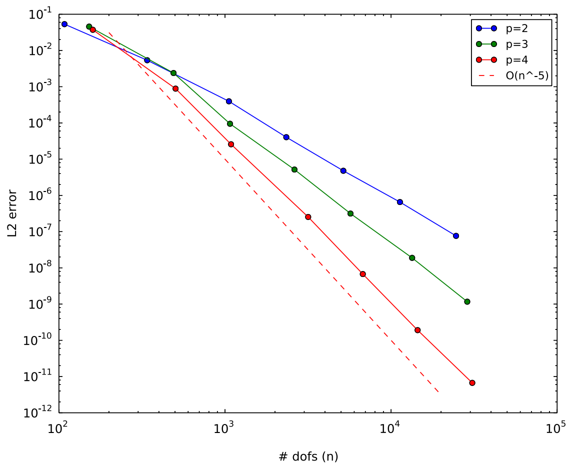

Fig. 3. (Example 1) L_2 errors for cross approximation with full pivoting with different spline degrees p plotted over the number of degrees of freedom $n = 2KN$.

$\mathcal{B} = \Phi(N, p) \otimes \Phi(N, p)$ with the same knots and degree. This is the "optimal" rank since additional terms cannot decrease the L_2-error significantly below the error of the tensor product interpolant (cf. Remark 3). The resulting ranks for full and row pivoting are shown in Table 1. We also compare them with the optimal ranks obtained when using the truncated SVD approximation of the matrix F.

We observe that the optimal ranks increase moderately with N or h^{-1}, as well as with the degree p. Furthermore, the ranks obtained with the row-pivoted cross approximation algorithm are only very slightly higher than those obtained using the truncated SVD. Again, these results strongly favor the use of the row-pivoted algorithm.

In Fig. 3, we plot the resulting L_2-errors over the number of degrees of freedom that we need to store when using the optimal ranks, $n = 2NK(N, p)$, with full pivoting. In Table 2, we give these errors in numerical form. In addition, we compute the approximate convergence rates with respect to h and to n: if h_i and h_{i+1} are the mesh sizes from two successive examples, and e_i and e_{i+1} the obtained L_2-errors, the rate is approximated as $\operatorname{roc}(h) = \log(e_{i+1}/e_i)/\log(h_{i+1}/h_i)$, and analogously for the number n of degrees of freedom.

The rates with respect to h show that we achieve approximately the rate $\mathcal{O}(h^{p+1})$ that one would expect using tensor product spline interpolation. However, the cross approximation scheme is much more efficient with respect to the number of degrees of freedom: the table shows that we can almost achieve the convergence rate $\mathcal{O}(n^{-(p+1)})$ for the L_2-error. Using d-dimensional tensor product spline interpolation with $n = N^d$ degrees of freedom, it is known that we can at best hope for the rate $\mathcal{O}(n^{-(p+1)/d})$ (see, e.g., [14]).

4.2. Example 2

We consider the function

$$f(x, y) = \frac{2}{3}\left(\exp(-\sqrt{(10x-3)^2 + (10y-3)^2}) + \exp(-\sqrt{(10x+3)^2 + (10y+3)^2})\right)$$

with two exponential peaks (see Fig. 4, left). The error function for a rank 12 spline approximation with degree $p = 2$ and $N = 200$ intervals computed using row pivoting is shown in the right plot of Fig. 4. Due to the presence of singularities,

Table 2
(Example 1) L_2 errors and convergence rates for cross approximation with full pivoting.

	h^{-1}	K	Error	roc(h)	roc($1/n$)
	16	3	5.334e−02	–	–
	32	5	5.338e−03	3.31	2.00
	64	8	3.954e−04	3.77	2.30
$p = 2$	128	9	4.043e−05	3.29	2.87
	256	10	4.782e−06	3.08	2.70
	512	11	6.549e−07	2.87	2.53
	1024	12	7.627e−08	3.10	2.76
	16	4	4.559e−02	–	–
	32	7	2.385e−03	4.26	2.52
	64	8	9.484e−05	4.65	4.12
$p = 3$	128	10	5.169e−06	4.20	3.26
	256	11	3.151e−07	4.04	3.60
	512	13	1.885e−08	4.06	3.30
	1024	14	1.164e−09	4.02	3.64
	16	4	3.722e−02	–	–
	32	7	8.851e−04	5.39	3.26
	64	8	2.574e−05	5.10	4.60
$p = 4$	128	12	2.539e−07	6.66	4.32
	256	13	6.741e−09	5.24	4.79
	512	14	1.909e−10	5.14	4.69
	1024	15	6.687e−12	4.84	4.42

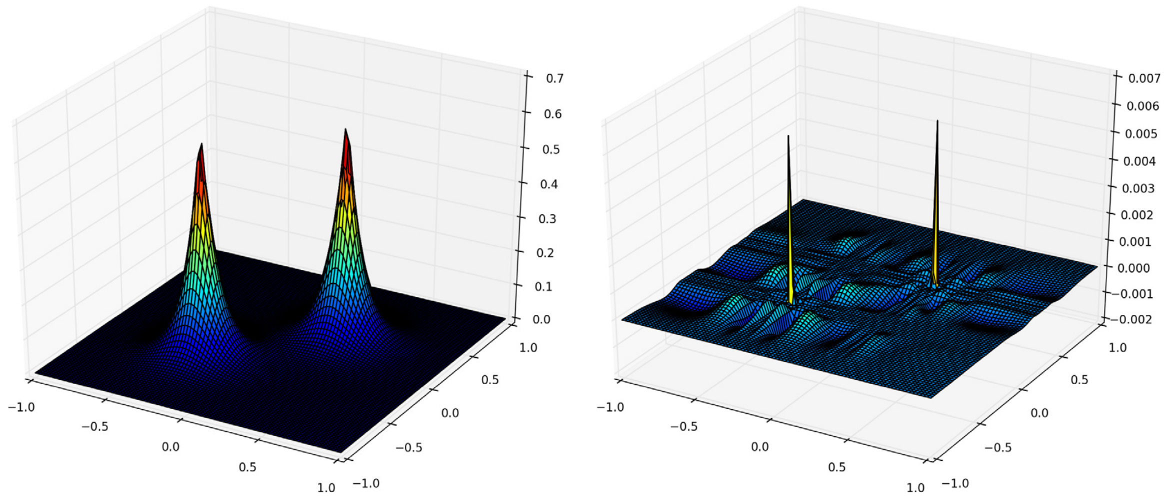

Fig. 4. (Example 2) *Left*: Function f. *Right*: Error $f - f_{12}$ of rank 12 spline approximation with $p = 2$, $N = 200$.

increasing the spline degree does not significantly reduce the approximation error for this function, and therefore all our tests are done with $p = 2$ in this example.

We perform low-rank approximation as described in Section 3.3 using row pivoting, degree $p = 2$, and varying N. The resulting L_2-errors are plotted in Fig. 5 over the rank K. For each choice of N, the algorithm converges to an approximation with L_2-error essentially identical to that of full tensor product spline interpolation. For instance, for $N = 400$, the optimal error is reached with rank $K = 18$, which requires storing $2KN = 14\,400$ coefficients, whereas storing the tensor product interpolant achieving the same error would require $N^2 = 160\,000$ coefficients.

4.3. Example 3

We consider the function

$$f(x, y) = \frac{\cos(10x(1 + y^2))}{1 + 10(x + 2y)^2},$$

an example lifted from [15]. See Fig. 6 for a contour plot of the function.

We perform low-rank approximation of this function using splines of varying degree p according to the algorithm from Section 3.3. We compare the full pivoting strategy and the more efficient row pivoting strategy in terms of the resulting L_2 errors. In each test, we use the same number $N = 200$ of knot spans for the spline spaces.

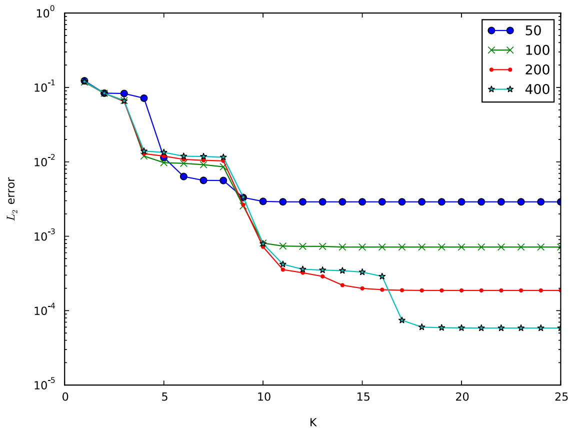

Fig. 5. (Example 2) L_2 errors plotted over rank K, with degree $p = 2$ and varying N.

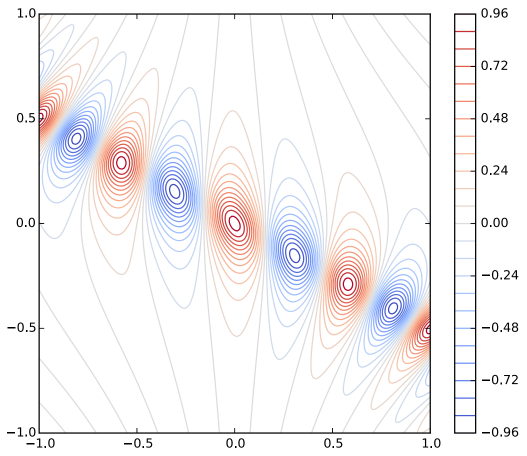

Fig. 6. (Example 3) Contour plot of function f.

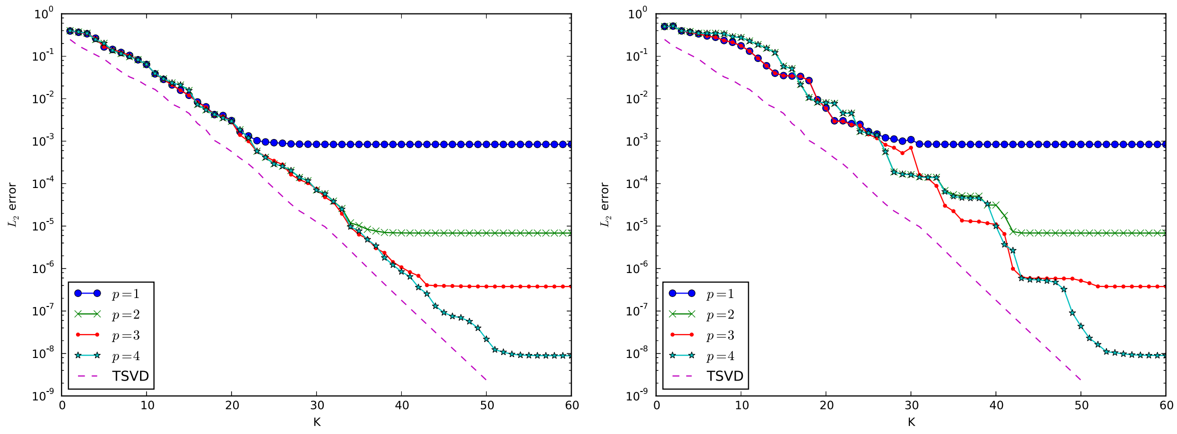

Fig. 7. (Example 3) L_2 errors plotted over rank K for $N = 200$ and varying spline degree p. *Left*: full pivoting. *Right*: row pivoting.

Fig. 7 shows the L_2 errors for low-rank approximation plotted over the rank K using full pivoting (left plot) and row pivoting (right plot). We observe that both pivoting strategies converge to the same error level, and row pivoting seems to require only slightly more iterations. In both plots, the dashed line labeled "TSVD" is an approximation of the best possible rank K error using truncated SVD as given by (2).

The obtained final errors for sufficiently high rank are essentially the same as if using full tensor product spline interpolation. Therefore, and since the function f is smooth, increasing the spline degree p reduces the obtainable error if one is willing to increase the rank of the approximation. It is interesting that if only low accuracy is required, increasing the degree has no benefits as the speed of convergence in the initial part of the plot is essentially independent of the spline degree, and seems to depend only on the decay of the singular values of the function f.

5. Conclusion and outlook

5.1. Conclusion

We have presented an algorithm for the approximation of bivariate functions by sums of separable splines which is based on the idea of cross approximation. The resulting approximants lie in tensor product spline spaces, but typically require the storage of much fewer degrees of freedom to achieve the same accuracy as full tensor product interpolants. Indeed, we have seen that using low-rank approximation we can achieve approximation errors close to $\mathcal{O}(n^{-(p+1)})$, where n is the number of degrees of freedom, whereas using tensor product interpolation in d dimensions we can at best hope for the rate $\mathcal{O}(n^{-(p+1)/d})$. Furthermore, the complexity considerations in Section 3.3 show that the algorithm is linear in N, the unknowns per coordinate direction, when using row pivoting. Thus we can treat very large approximation problems, even to the point where the tensor product interpolant would not fit into memory anymore and/or be too expensive to compute, as long as the rank K is relatively small.

There are three parameters to be chosen in our algorithm: the spline degree p, the dimension of the univariate spline spaces N, and the approximation rank K. The parameters (p, N) should be chosen according to the smoothness of the function to be approximated and the final desired accuracy. For the rank K, there is an "optimal" choice depending on (p, N) in the sense that there is usually a rank where the low-rank approximant achieves the same error as tensor product interpolation with N^2 degrees of freedom. Increasing the rank beyond this point does not further decrease the error since the low-rank approximations lie in the space of tensor product splines.

We have demonstrated the approximation properties of our algorithm using several numerical examples. In particular, we have seen that the use of row pivoting, which is much faster for large problems, does not significantly increase the approximation error.

The closest work to ours available in the literature seems to be by Townsend and Trefethen [7] on `chebfun2`. Therefore it seems interesting to compare our approaches in some more detail. Both algorithms approximate bivariate functions by sums of outer products of univariate functions. Also, both algorithms use the same basic idea of cross approximation which is, as Townsend and Trefethen point out, a variant of Gaussian elimination. We can find two main differences:

1. `chebfun2` approximates the univariate factors using polynomial interpolation with Chebyshev nodes, whereas our method uses spline approximation.
2. `chebfun2` chooses the number of interpolation nodes adaptively, whereas we use a fixed, pre-determined spline space. This is because for `chebfun2` the motivation is to compute with bivariate functions while retaining, as close as possible, machine precision, whereas our focus was on possible applications in Isogeometric Analysis.

5.2. Future work and possible applications

Although the cross approximation algorithm for matrices is widely known throughout the literature, no comprehensive error analysis seems to be available. Bebendorf [3] gives some results in this direction, however the estimates involve best-approximation errors which seem inaccessible in practice. Bebendorf also shows that the resulting approximations can be viewed as so-called skeleton approximations, which opens the door to applying the theory from [16,17]. The maximum volume concept used in these works is however difficult to realize in practice. Schneider [18] gives some new results concerning error analysis of an ACA algorithm for functions, as well as a comprehensive overview of the existing literature on this topic. He uses techniques first introduced by Babaev [19]. Again, these estimates make use of the maximum-volume concept and are in terms of a certain best-approximation error which seems difficult to quantify.

A possible analysis for our proposed algorithm could be pursued along either of two lines: either on the level of matrix algebra by analyzing the underlying matrix ACA algorithm, or directly on the level of functions similar to the approach of Schneider [18]. At present we have no conclusive results for either of these approaches and leave the rigorous error analysis for future work.

Low-rank approximations can be of great benefit in isogeometric analysis, where geometries are often represented in tensor product spaces. Applying the low-rank approximation algorithm as a form of compression can result in greatly reduced storage requirements as well as enable faster assembling of stiffness matrices. Furthermore, the large savings obtained here motivate the study of solution methods for partial differential equations discretized using isogeometric schemes which work directly on low-rank approximations of the input data and also produce a solution in this format. Promising results in this direction in other fields are given in [20]. The algorithm we presented could then be used as a preprocessing step to convert input data given in tensor product or analytic form to the low-rank format.

A crucial extension of the present method will be to higher dimensions. The structure of tensors with more than two dimensions is known to be much more complicated than that of matrices. In particular, the best approximation in "canonical" form, that is, as a sum of rank 1 tensors, is known to be an ill-posed problem in general. To work around this problem, other formats such as Tucker decomposition, tensor train decomposition, or hierarchical Tucker decomposition are commonly used. We refer to the survey [4] and the references therein for the related literature. Nevertheless, canonical approximations can still be useful in applications, and it would be interesting to study a generalization of the present algorithm to higher dimensions. Hackbusch [5] gives some details on generalizing the cross approximation algorithm to tensors of order greater than two.

Acknowledgments

The second author was supported by the National Research Network "Geometry + Simulation" (NFN S117, 2012–2016), funded by the Austrian Science Fund (FWF). The authors acknowledge the support by Grant No. DFNI-T01/0001, "Effective methods and algorithms for geometrical modeling", funded by the Bulgarian Science Fund.

References

[1] W. Hackbusch, A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices, Computing (ISSN: 0010-485X) 62 (2) (1999) 89–108. http://dx.doi.org/10.1007/s006070050015.
[2] W. Hackbusch, B.N. Khoromskij, A sparse $\mathcal{H}$-matrix arithmetic, Computing (ISSN: 0010-485X) 64 (1) (2000) 21–47. http://dx.doi.org/10.1007/PL00021408.
[3] M. Bebendorf, Hierarchical Matrices, in: Lecture Notes in Computational Science and Engineering, vol. 63, Springer, Berlin, Heidelberg, 2008.
[4] L. Grasedyck, D. Kressner, C. Tobler, A literature survey of low-rank tensor approximation techniques, GAMM-Mitt. 36 (1) (2013) 53–78.
[5] W. Hackbusch, Tensor Spaces and Numerical Tensor Calculus, in: Springer Series in Computational Mathematics, Springer, Berlin, Heidelberg, ISBN: 9783642280276, 2012.
[6] T.J.R. Hughes, J.A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Comput. Methods Appl. Mech. Engrg. (ISSN: 00457825) 194 (39–41) (2005) 4135–4195. http://dx.doi.org/10.1016/j.cma.2004.10.008.
[7] A. Townsend, L.N. Trefethen, An extension of Chebfun to two dimensions, SIAM J. Sci. Comput. 35 (6) (2013) 495–518.
[8] C. Eckart, G. Young, The approximation of one matrix by another of lower rank, Psychometrika (ISSN: 0033-3123) 1 (3) (1936) 211–218. http://dx.doi.org/10.1007/BF02288367.
[9] D. Werner, Funktionalanalysis, in: Springer-Lehrbuch, Springer-Verlag, Berlin, Heidelberg, ISBN: 978-3-642-21016-7, 2011. http://dx.doi.org/10.1007/978-3-642-21017-4.
[10] C. de Boor, On calculating with B-splines, J. Approx. Theory (ISSN: 0021-9045) 6 (1) (1972) 50–62. http://dx.doi.org/10.1016/0021-9045(72)90080-9.
[11] L. Piegl, W. Tiller, The NURBS Book, in: Monographs in Visual Communications, Springer-Verlag GmbH, ISBN: 9783540615453, 1997.
[12] C. de Boor, A Practical Guide to Splines, in: Applied Mathematical Sciences, vol. 27, Springer, 2001. (Revised edition).
[13] G.H. Golub, C.F. Van Loan, Matrix Computations, fourth ed., in: Johns Hopkins Studies in the Mathematical Sciences, vol. 3, Johns Hopkins University Press, 2012.
[14] L. Schumaker, Spline Functions: Basic Theory, in: Cambridge Mathematical Library, Cambridge University Press, ISBN: 9780521705127, 2007.
[15] A. Townsend, L.N. Trefethen, Gaussian elimination as an iterative algorithm. Technical report, University of Oxford Mathematical Institute, 2013, URL: http://eprints.maths.ox.ac.uk/1670/.
[16] S.A. Goreinov, E.E. Tyrtyshnikov, N.L. Zamarashkin, A theory of pseudoskeleton approximations, Linear Algebra Appl. 261 (1) (1997) 1–21. http://dx.doi.org/10.1016/S0024-3795(96)00301-1.
[17] S.A. Goreinov, N.L. Zamarashkin, E.E. Tyrtyshnikov, Pseudo-skeleton approximations by matrices of maximal volume, Math. Notes (ISSN: 0001-4346) 62 (4) (1997) 515–519. http://dx.doi.org/10.1007/BF02358985.
[18] J. Schneider, Error estimates for two-dimensional cross approximation, J. Approx. Theory (ISSN: 0021-9045) 162 (9) (2010) 1685–1700. http://dx.doi.org/10.1016/j.jat.2010.04.012.
[19] M.-B.A. Babaev, Best approximation by bilinear forms, Math. Notes 46 (2) (1989) 588–596. http://dx.doi.org/10.1007/BF01137621.
[20] B. Khoromskij, Tensor numerical methods for high-dimensional PDEs: Basic theory and initial applications. Technical report, 2014, URL: http://arxiv.org/abs/1408.4053.