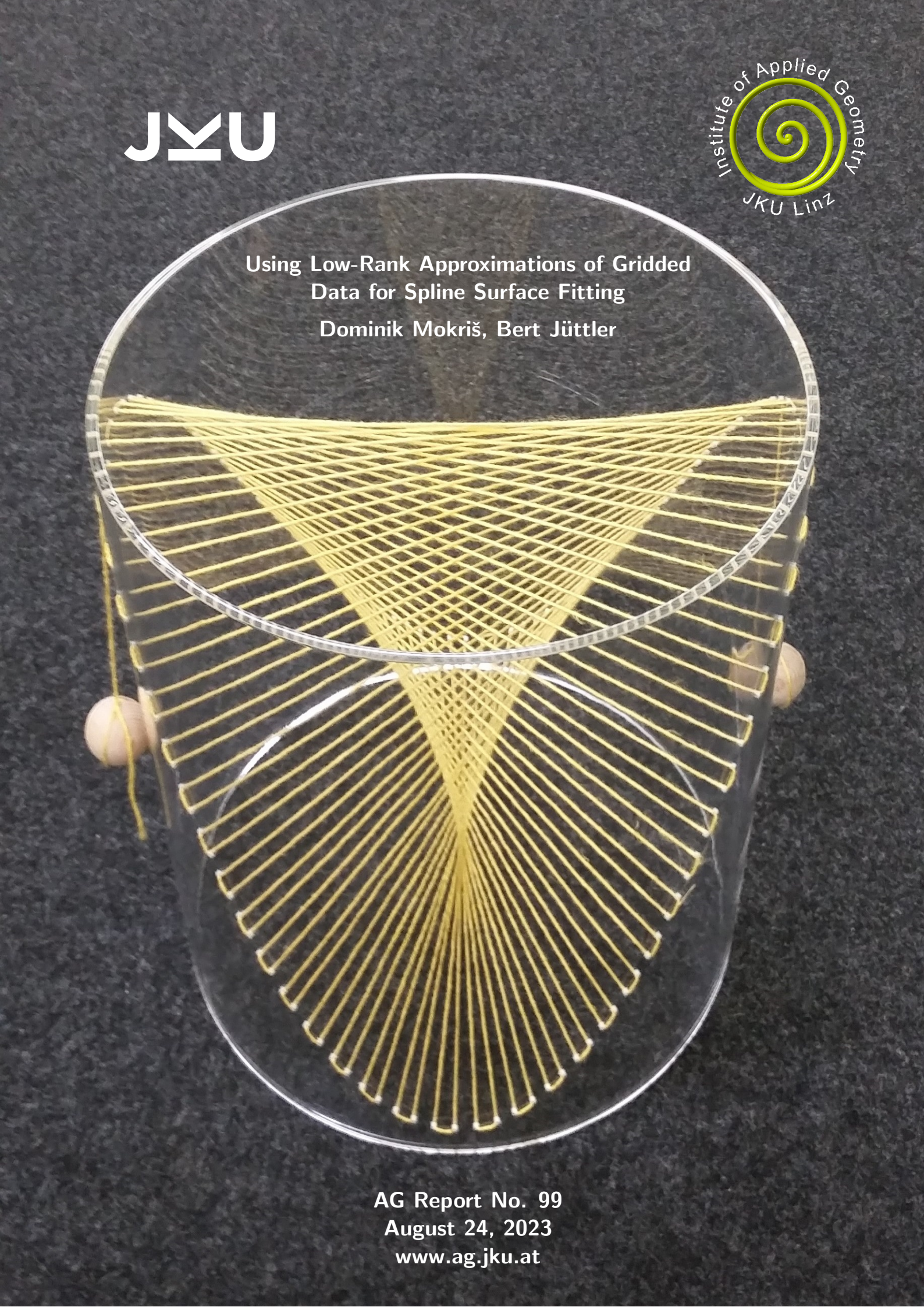

Using Low-Rank Approximations of Gridded Data for Spline Surface Fitting

Dominik Mokriš, Bert Jüttler

AG Report No. 99
August 24, 2023
www.ag.jku.at

Using Low-Rank Approximations of Gridded Data for Spline Surface Fitting

Dominik Mokriš[a,*], Bert Jüttler[b]

[a] *MTU Aero Engines AG, Germany*
[b] *Institute of Applied Geometry, Johannes Kepler University Linz, Austria*

Abstract

The paper is devoted to the problem of finding bivariate tensor-product spline (or similar) functions that approximate gridded data in the least-squares sense. We propose to apply a low rank approximation of matrices to the data, to find the result by solving a sequence of univariate fitting problems that can be handled efficiently. This can be seen as a generalization of the method proposed by Georgieva and Hofreither [1] that combines cross approximation (which is a particular method for low rank matrix approximation) with spline interpolation.

While the algorithm yields the best least-squares approximation after r steps, where r denotes the rank of the data matrix, terminating it earlier yields a low-rank approximation, which often provides a sufficient level of accuracy. We also present a stopping criterion (based on a lower error estimate) that allows to use the method efficiently in the situation when the required number of degrees of freedom is not known in advance.

Keywords: least-squares methods, low-rank approximation, surface fitting, tensor-product splines
2010 MSC: 65D07

1. Introduction

Surface (re-)construction with tensor-product splines via least squares approximation is a classical and well-established technique, which is ubiquitous in industrial practice, see [2] and the references cited therein. The fundamentals of the method are covered by several textbooks [3, 4]. More advanced techniques have been summarized in [5]. The (seemingly) alternative approach of Least Squares Progressive Iterative Approximation [6] can be seen as applying the gradient descent method (or, equivalently, the Richardson iteration method) to the problem of least squares fitting [7]. This approach enables various accelerating measures [8] and is beneficial for large data sets.

*Corresponding author
Email address: dominik.mokris@mtu.de (Dominik Mokriš)
URL: www.ag.jku.at (Bert Jüttler)

The particular case of *structured* (i.e., *gridded*) *data* allows to process surfaces by simply applying methods that were established for curves, and this opens the path to employ various performance-improving measures. The tensor-product formulation of the *abstract linear interpolation scheme*, which was already studied by de Boor [9, Chapter XVII], includes interpolation, least-squares approximation and collocation of a linear differential equation. It can be employed to parallelize the computation [10]. In the case of surface (re-) construction from structured data, this corresponds to first interpolating/approximating data slices in one parametric direction and then interpolating/approximating through the control points (i.e., performing *lofting*) in the other parametric direction, cf. [11, Section 7.2]. The result is independent of the ordering of the parametric directions. Contrary to the corresponding statement in the NURBS book of Piegl and Tiller [4, Section 9.4.3], this is also true when using least-squares approximations that interpolate at the endpoints.

Semi-structured data is used for constructing surfaces via lofting, see [12]. Even for unstructured data, the process of matrix assembly for least-squares fitting can be significantly accelerated with the help of a grid projection step (also if the resulting grid is only sparsely populated) via sum factorization [13].

Georgieva and Hofreither [1] pioneered the use of low rank matrix approximation for surface reconstruction from structured data, focusing on *interpolation*, proposing a two-step procedure that projects a bivariate function into a suitably chosen tensor-product spline space. Firstly, they create a matrix that contains sampled values of the function at the Greville abscissae and perform a low rank approximation of that matrix. Secondly, they sum the contributions of sufficiently many spline interpolants to the obtained rank-1 matrices.

Three methods for low rank matrix approximation were considered by them: Truncated Singular Value Decomposition (SVD) and Adaptive Cross Approximation (ACA) with and without pivoting, see [1] and the references cited therein. When the rank is not prescribed by the application, various stopping (rank selection) criteria can be used instead [14, 15, 16]. In another interesting paper, Pan et al. [17] proposed a method for compact implicit surface reconstruction via low-rank tensor approximation.

Georgieva and Hofreither [1] compare the performance of these decomposition methods and notice that while truncated SVD is known to provide the best approximation of a given rank [18], ACA with pivoting is only slightly less precise and can be efficiently computed on the fly. While the nice geometric interpretation of the full tensor-product interpolation method is lost, the possibility of early termination is a substantial advantage from the efficiency viewpoint.

The present paper presents a generalization of Georgieva and Hofreither's method. Instead of focusing on interpolation, we employ (weighted) *least-squares fitting*, thus gaining more flexibility concerning the sampling of the data. Each step of the low rank approximation of the data matrix yields a tensor-product of two vectors. We show that by fitting them separately and re-assembling the control points afterwards we arrive at the full tensor-product least-squares approximation of the data after r iterations, where r is the rank of the data matrix. Additionally, we use a lower error estimate in order to derive a stopping criterion. We demonstrate that this criterion is very efficient in the situation when least-squares fitting is combined with refinement of the basis.

The remainder of the paper is organized as follows. Section 2 describes in detail how to reformulate a bivariate (i.e., matrix-valued) least-squares problem as a collection of several univariate (i.e., vector-valued) problems. Additionally, a lower error estimate is derived. Section 3 recalls the method of least-squares fitting of gridded data with tensor-product B-splines and transforms the theory that was presented in the previous section into an efficient algorithm. This algorithm is then tested on several examples in Section 4.

2. Decomposing least-square solutions of matrix equations

Given matrices $X \in \mathbb{R}^{m\times p}$, $Y \in \mathbb{R}^{n\times q}$ and $F \in \mathbb{R}^{m\times n}$ with dimensions satisfying $m > p$ and $n > q$, we discuss how to solve an over-determined system of equations

$$F = XCY^T \tag{1}$$

for the unknown matrix $C \in \mathbb{R}^{p\times q}$ in the least-squares sense,

$$C = \underset{C'\in\mathbb{R}^{p\times q}}{\operatorname{argmin}} \|F - XC'Y^T\|_{2,2}^2 \ , \tag{2}$$

where $\|\cdot\|_{2,2}$ denotes the Frobenius norm. The number of equations (mn) exceeds the number of unknowns (pq). The rows of X and Y are assumed to be linearly independent. Our specific motivation for addressing this problem will be presented in the next section.

It is well known (see [19, Section 1] or [20, Theorem 4.6]) that the solution evaluates to

$$C = \underbrace{\big((X^\top X)^{-1}X^\top\big)}_{=X^+} F \underbrace{\big((Y^\top Y)^{-1}Y^T\big)^T}_{=(Y^+)^T} \ , \tag{3}$$

with the two Moore-Penrose pseudoinverses of X and Y, as the Gramian matrices X^TX and Y^TY are invertible [21, Lemma 8.4]. This can be equivalently expressed as the fact that the pseudoinverse of the Kronecker product of two matrices is equal to the Kronecker product of their pseudoinverses.

This observation leads to what we call the *standard method* (which has been described in [22, Section 10.2] in the context of tensor-product spline surface fitting) for solving over-determined systems of equations (1) in the least-squares sense:

1. Find the auxiliary matrix $A = \big((X^TX)^{-1}X^T\big)F$ by solving the linear system

   $$(X^TX)A = X^TF$$

   with m equations and n right-hand side vectors for the column vectors of A.
2. Find the solution $C = A\big((Y^TY)^{-1}Y^T\big)^T$ by solving the linear system

   $$C(Y^TY) = AY$$

   with q equations and p right-hand side vectors for the row vectors of C.

In the remainder of the paper we consider a decomposition

$$F = \sum_{j=1}^{r} \sigma_j \mathbf{u}_j \otimes \mathbf{v}_j = U\Sigma V^T \tag{4}$$

of the given matrix F into a sum of r rank-1 matrices with $r \geq \operatorname{rank}(F)$, where $\mathbf{u}_1, \ldots \mathbf{u}_r$ are the column vectors of a matrix $U \in \mathbb{R}^{m\times r}$, $\mathbf{v}_1, \ldots \mathbf{v}_r$ are the column vectors of a matrix $V \in \mathbb{R}^{n\times r}$, and $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_r)$. The solution C inherits the structure of this decomposition. This is described in the following result:

Lemma 1. *The solution* (2) *possesses the decomposition*

$$C = \sum_{j=1}^{r} \sigma_j \mathbf{g}_j \otimes \mathbf{h}_j \ , \tag{5}$$

where the vectors $\mathbf{g}_j$ *and* $\mathbf{h}_j$ *are the solutions of the least-squares problems*

$$\mathbf{g}_j = \underset{\mathbf{g}'_j \in \mathbb{R}^p}{\operatorname{argmin}} \|\mathbf{u}_j - X\mathbf{g}'_j\|_2^2 \quad \textit{and} \quad \mathbf{h}_j = \underset{\mathbf{h}'_j \in \mathbb{R}^q}{\operatorname{argmin}} \|\mathbf{v}_j - Y\mathbf{h}'_j\|_2^2 \ .$$

Proof. The vectors $\mathbf{g}_j$ and $\mathbf{h}_j$ satisfy the equations

$$X^T X\mathbf{g}_j = X^T\mathbf{u}_j \quad \text{and} \quad Y^T Y\mathbf{h}_j = Y^T\mathbf{v}_j \ ,$$

for $j = 1, \ldots, r$, which can be compactly rewritten as

$$X^T XG = X^T U \quad \text{and} \quad Y^T YH = Y^T V \ , \tag{6}$$

where the matrices $G \in \mathbb{R}^{p\times r}$ and $H \in \mathbb{R}^{q\times r}$ possess these vectors as their column vectors. We substitute (4) into (3) and obtain

$$C = \underbrace{\big((X^\top X)^{-1}X^\top\big)U}_{=G}\, \Sigma\, \underbrace{V^T\big((Y^\top Y)^{-1}Y^T\big)^T}_{=H^T} = G\Sigma H^T \ ,$$

since the matrices G and H fulfill the equations (6). This completes the proof, as the right–hand side of (5) can be compactly rewritten as $G\Sigma H^T$. □

Our goal is to construct a low-rank approximation of C by using a partial sum of the representation given in Lemma 1. For future reference we state a *lower* error estimate for the least squares error

$$\varepsilon := \|F - XCY^T\|_{2,2} \ ,$$

which is based on the individual residuals

$$R_i = \mathbf{u}_i \otimes \mathbf{v}_i - X(\mathbf{g}_i \otimes \mathbf{h}_i)Y^T$$

that are obtained by considering the decompositions (4) and (5).

Lemma 2. *The least squares error is bounded from below by*

$$\left\|F - XCY^T\right\|_{2,2} \geq \varepsilon_j - \left\|F - \sum_{i=1}^{j} \sigma_i \mathbf{u}_i \otimes \mathbf{v}_i\right\|_{2,2} \ , \tag{7}$$

for any $j \leq r$, where $\varepsilon_j = \| \sum_{i=1}^{j} \sigma_i R_i \|_{2,2}$ is the error of the approximation to the first j terms in the expansion (4)*.*

Proof. The reverse triangle inequality implies

$$\left\|F - XCY^T\right\|_{2,2} = \Big\| \sum_{i=1}^{r} \sigma_i R_i \Big\|_{2,2} \geq \varepsilon_j - \Big\| \sum_{i=j+1}^{r} \sigma_i R_i \Big\|_{2,2} \ .$$

The result then follows by first noting that

$$\Big\| \sum_{i=j+1}^{r} \sigma_i R_i \Big\|_{2,2} = \min_{C'' \in \mathbb{R}^{p \times q}} \Big\| \sum_{i=j+1}^{r} \sigma_i \mathbf{u}_i \otimes \mathbf{v}_i - XC''Y^T \Big\|_{2,2} \leq \Big\| \sum_{i=j+1}^{r} \sigma_i \mathbf{u}_i \otimes \mathbf{v}_i \Big\|_{2,2}$$

where the equality is implied by Lemma 1 and the inequality is obtained by considering $C'' = 0$, and then taking (4) into account. □

It should be noted that the lower bound can be evaluated even if only the first j terms of the expansion (4) are known. This fact will be beneficial when using this lower bound in an adaptive approximation procedure, as will be described later.

Remark 3. By using the triangular inequality instead of the inverse one in the proof of Lemma 2 one can obtain an upper bound on the error. These two estimates can then be summarized as

$$|\varepsilon - \varepsilon_j| \leq \left\|F - \sum_{i=1}^{j} \sigma_i \mathbf{u}_i \otimes \mathbf{v}_i\right\|_{2,2} \ .$$

3. An algorithm for least-squares fitting of gridded data

We apply Lemma 1 to the problem of least squares fitting of gridded data with separable weights by tensor-product spline functions. For given points (ξ_k, η_ℓ) and associated data values $d_{k\ell}$, $k = 1, \ldots, m$ and $\ell = 1, \ldots, n$, we construct a tensor-product spline function

$$s(\xi, \eta) = \sum_{i=1}^{p} \sum_{j=1}^{q} c_{ij} M_i(\xi) N_j(\eta)$$

with control points c_{ij} and B-splines M_i and N_j that minimizes the sum of the squared residuals

$$\sum_{k=1}^{m} \sum_{\ell=1}^{n} \omega_k^2 \nu_\ell^2 \big(d_{k\ell} - s(\xi_k, \eta_\ell)\big)^2 \ ,$$

where ω_k and ν_ℓ are user-defined non-negative weights.

This is an instance of Problem (2) with

$$X = \begin{pmatrix} \omega_1 M_1(\xi_1) & \cdots & \omega_1 M_p(\xi_1) \\ \vdots & \ddots & \vdots \\ \omega_m M_1(\xi_m) & \cdots & \omega_m M_p(\xi_m) \end{pmatrix} \text{ and } Y = \begin{pmatrix} \nu_1 N_1(\eta_1) & \cdots & \nu_1 N_q(\eta_1) \\ \vdots & \ddots & \vdots \\ \nu_n N_1(\eta_n) & \cdots & \nu_n N_q(\eta_n) \end{pmatrix}, \quad (8)$$

where c_{ij} and $f_{k\ell} = \omega_k \nu_\ell d_{k\ell}$ are the elements of matrices C and F, respectively. The matrices X and Y possess maximum rank (i.e., linearly independent rows) if the Schoenberg-Whitney conditions [23] are satisfied by a subset of the coordinates ξ_k and η_ℓ.

We now present Algorithm `LowRankFit`, which is based on Lemmas 1 and 2. It generates an (approximate) solution to the problem of least-squares fitting of gridded data with tensor-product spline functions.

Algorithm `LowRankFit`**:** Construct a low-rank approximation of least-squares fitting of gridded data.

Input : matrices $X \in \mathbb{R}^{m\times p}$, $Y \in \mathbb{R}^{n\times q}$ and matrix $F \in \mathbb{R}^{m\times n}$ of rank r, tolerances $\varepsilon_{\text{accept}}$ and $\varepsilon_{\text{abort}}$

Output: matrix $C' \in \mathbb{R}^{p\times q}$ approximating the solution C of Equation (2), resulting status

```
C' ← 0^{p×q};
for j = 1, ..., r do
    (σ_j, u_j, v_j) ← LowRankMatrixApprox(F, j);
    g_j ← LSsolve(X, u_j);
    h_j ← LSsolve(Y, v_j);
    C' ← C' + σ_j g_j ⊗ h_j;
    ε_j ← ‖F − X C' Y^T‖_{2,2};
    if ε_j < ε_accept then
        return Success
    else if ε_j − ‖F − Σ_{i=1}^{j} σ_i u_i ⊗ v_i‖_{2,2} > ε_abort then
        return CannotReachTolerance
return MaxIterReached
```

The algorithm proceeds by iterating over r up to the rank of F. In each iteration, the next instances of σ_r, $\mathbf{u}_r$ and $\mathbf{v}_r$ that appear in the decomposition (4) are constructed by `LowRankMatrixApprox`; this can be any algorithm capable of constructing a decomposition (4), see below. The coefficient vectors $\mathbf{g}_r$ and $\mathbf{h}_r$ are computed as (univariate) least-squares approximations of $\mathbf{u}_r$ and $\mathbf{v}_r$, respectively, and their tensor product is added to C'. Finally, a stopping criterion based on Lemma 2 is invoked in order to check whether the algorithm still has a possibility to satisfy the user-defined error bound.

A few comments are in order. First, it follows from Lemma 1 that C' is equal to C as j reaches r unless interrupted by one of the stopping criteria. As we will see in the

examples, a significantly lower rank is usually sufficient for a very good approximation. Second, in order for `LowRankMatrixApprox` to be reasonably efficient, one would be executing an adaptive cross approximation (ACA) algorithm (or a similar method for constructing the decomposition (4), cf. [1]) in parallel. Third, the systems for computing $\mathbf{g}_r$ and $\mathbf{h}_r$ are always the same and vary only in the right hand-side; one can thus save effort by precomputing, e.g., LU-decompositions of the systems on lines 4 and 5 of the algorithm. Finally, the algorithm generates not only a decent approximation of the data but the coefficients are also structured in the form $C' = \sum_{j=1}^{r} \sigma_j \mathbf{g}_j \otimes \mathbf{h}_j$. While we do not have an immediate practical application at hand, this form of the solution may be exploited in the future.

Algorithm `LowRankFit` is a generalization of [1, Algorithm 2] that replaces interpolation with weighted approximation. One can see this by choosing $m = p$ and $n = q$ (i.e., requiring that there is a one-to-one mapping between the data points and basis functions in each direction), setting all weights equal to one and by choosing the parameters $\xi_1, \ldots, \xi_m$ and $\eta_1, \ldots, \eta_n$ to be the Greville abscissae [23] of the B-spline bases $\{M_1, \ldots, M_p\}$ and $\{N_1, \ldots, N_q\}$. Then the matrices X and Y from Equation (8) are square and invertible due to Schoenberg-Whitney Theorem [23]. Equations (6) then simplify to

$$XG = U \quad \text{and} \quad YH = V \ ,$$

respectively; these are the formulae for the computation of control points of the component bases if the interpolation problem is unisolvent.

4. Numerical examples

We illustrate the performance of Algorithm `LowRankFit` by several examples implemented using the G+Smo library [24]. We will be using truncated SVD and ACA with and without pivoting for `LowRankMatrixApprox`; these algorithms are described in [1] in more detail.

Example 1. We approximate the function

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

which was also considered in [25], at 300×300 uniformly spaced points, yielding a data matrix of rank 81 in double floating arithmetic. Figure 1 depicts the results of using Algorithm `LowRankFit` with $\varepsilon_{\text{abort}} = +\infty$ and $\varepsilon_{\text{accept}}$ equal to 101% of the error obtained when performing standard tensor-product fitting with the same basis. We compare the error in the ℓ^2-norm using adaptive cross approximation with and without pivoting in `LowRankMatrixApprox` for various uniform cubic B-spline bases. The low rank approximations quickly reach the error of the full least-squares solutions with the same basis. On the one hand, this process is slightly slower when pivoting is used. On the other hand, ACA with pivoting was about three times faster in our tests, hence there is clearly a trade-off between computational time and number of iterations.

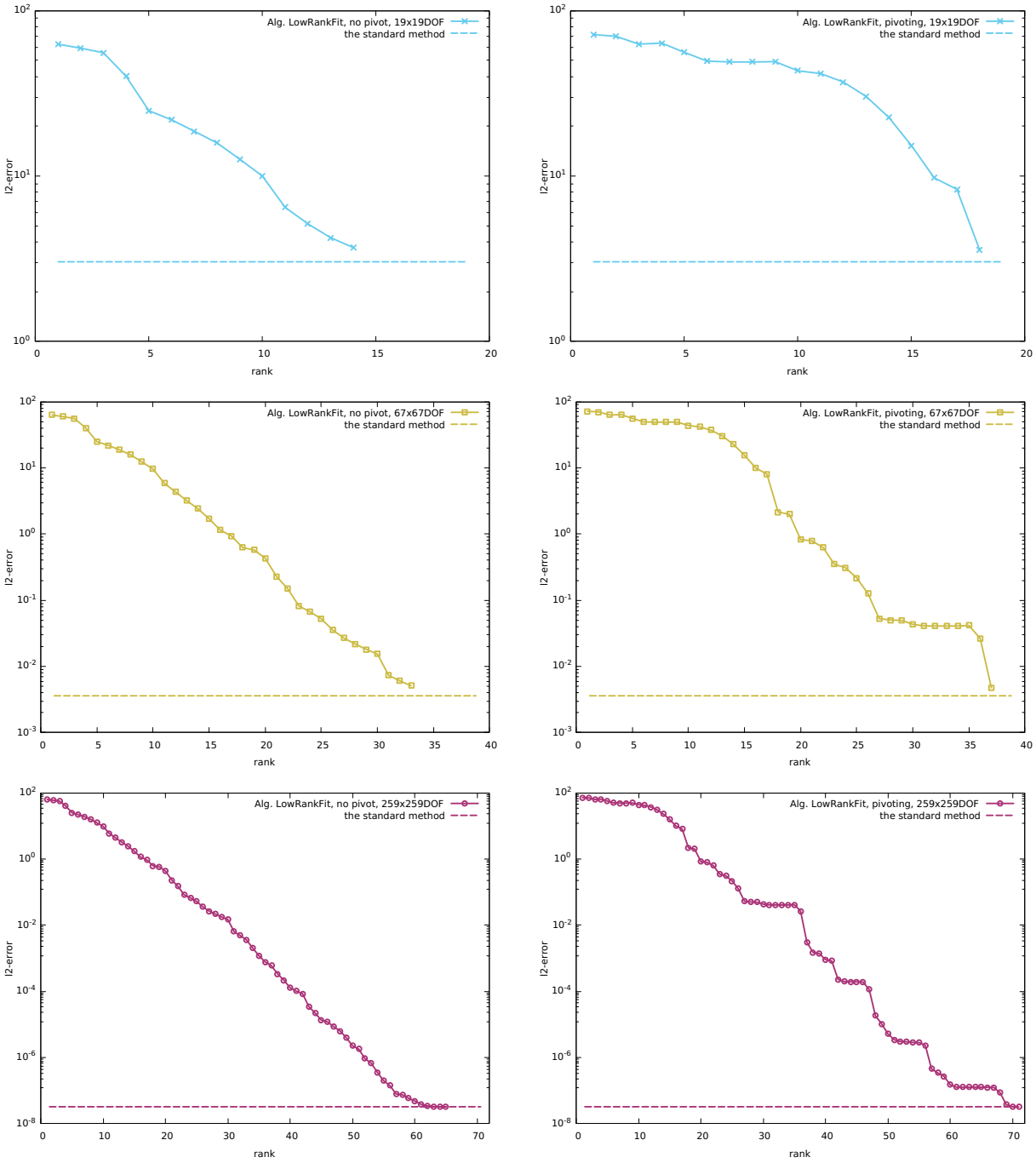

Figure 1: Example 1 – The ℓ^2-errors obtained when using ACA without (left) and with (right) pivoting in `LowRankMatrixApprox` for various tensor-product spline bases.

Figure 2 depicts the costs. We compare standard least-squares fitting and the low-rank fitting described, with and without row pivoting. Clearly, the use of Algorithm `LowRankFit` provides a significant speed-up compared to standard least-squares fitting.

The costs in the left-hand side figure are expressed in terms of the number of calls to (univariate) `LSsolve`. While in practice one would perform an LU decomposition beforehand, backward substitutions still constitute significant part of the total cost. Even though these numbers should be interpreted with caution, since we are comparing calls to `LSsolve` with different number of unknowns, they nevertheless confirm that the new algorithm has a computational advantage.

The right-hand side figure depicts the costs of the two crucial steps: solution of linear systems (full lines) and the low-rank matrix approximation (dashed lines). We compare three approaches: Algorithm `LowRankFit` using the ACA with and without pivoting for `LowRankMatrixApprox` and the standard method (see Section 2). The computations have been performed on a laptop computer and averaged over 10 runs. Note that we have used the highly optimized Eigen library [26] for solving the linear systems; applying a similar same amount of work on code acceleration to our implementation of low-rank matrix approximation is likely to significantly speed up this step. This explains why the reported computation times do not confirm the advantages of the new method.

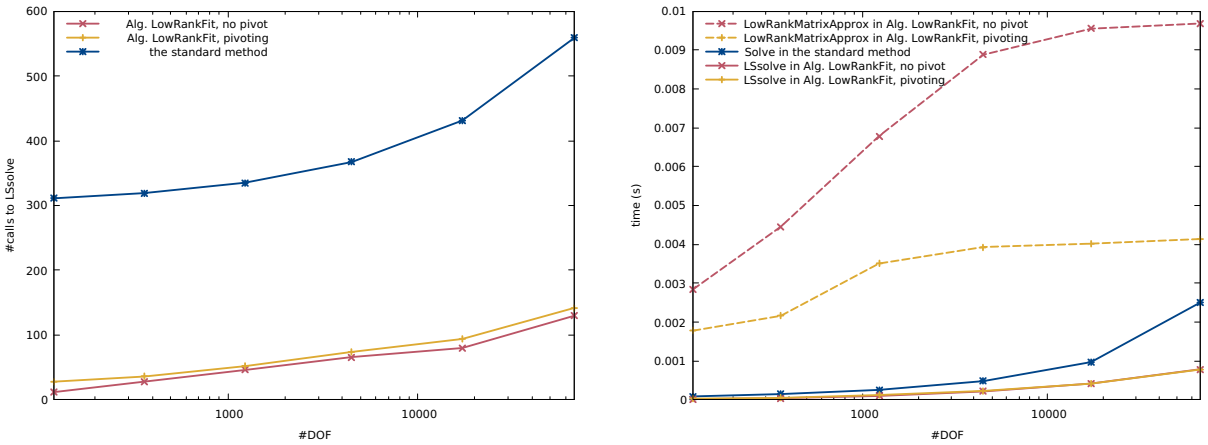

Figure 2: Example 1 – Computational effort (left: number of calls to `LSsolve`, right: computation time) depending on the size of the basis.

Example 2. We approximate the CRESCENDO airfoil presented in [27]. It is represented by 41×151 sampled points – which we parameterize uniformly in the radial direction and use the centripetal parametrization of the first profile in the circumferential direction – shown in Figure 3, left. As the points possess three coordinates, we apply Algorithm `LowRankFit` component-wise and assemble the results. All the data matrices have full rank (41).

Figure 3 compares the standard tensor-product fitting using 40×128 uniform cubic B-splines with the results of Algorithm `LowRankFit` for the same basis using ACA with pivoting, $\varepsilon_{\text{abort}} = +\infty$ and $\varepsilon_{\text{accept}} = 5\text{e}{-4}$; this terminates the computation after 8, 12 and 8 iterations in the x-, y- and z-component, respectively. The two surfaces are virtually identical.

Figure 4 depicts the decomposition error of the data matrix (in Frobenius norm) and the fitting error (in the maximum norm) of the x-coordinate, obtained when using ACA with

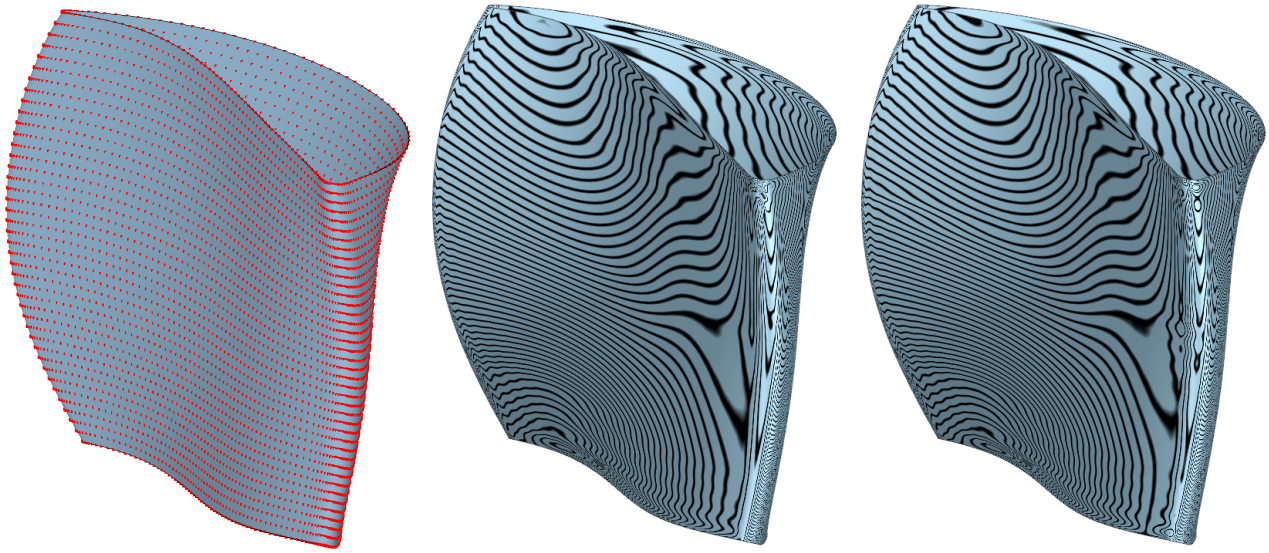

Figure 3: Example 2 – Left: input points; middle: full tensor-product fitting; right: The approximating surface generated by Algorithm `LowRankFit`.

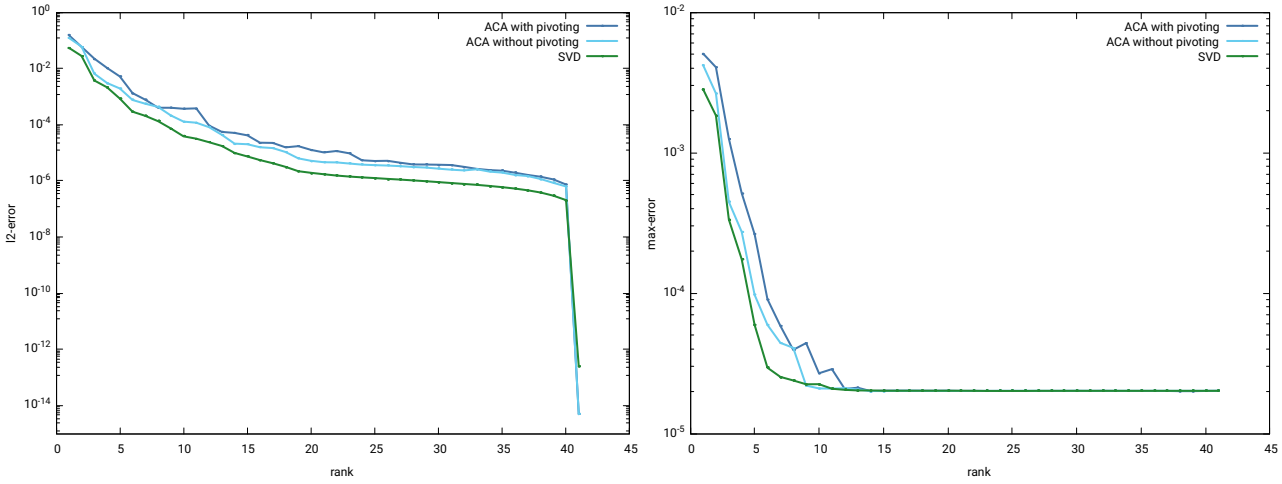

Figure 4: Example 2 – errors of the x-component. Left: matrix decomposition error; right: maximum fitting error.

and without pivoting and truncated SVD. The other two components give similar results. The approximation error does not significantly decrease from the 15-th iteration onwards. Note that Algorithm `LowRankFit` performs well even with respect to the maximum norm, which is not covered by the theoretical results.

Again, the computational effort compares favorably with the standard method, since we need less than 30 calls (for each coordinate) to `LSsolve` instead of the $\min(m+q, n+p) = 169$ calls required by standard tensor-product fitting, to get virtually identical results. (Again these numbers should be interpreted with caution, since we are comparing calls to `LSsolve` with different numbers of unknowns. Still they confirm that the new algorithm has a compu-

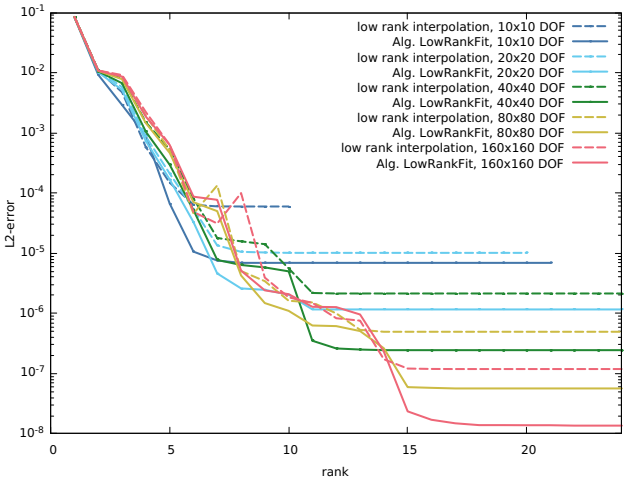

Figure 5: Example 3 – Approximate L^2-errors.

tational advantage.) Note that ACA with and without pivoting for `LowRankMatrixApprox` perform only marginally worse than the much more costly truncated SVD.

Example 3. The minimization problem (2) covers also the situation of weighted least-squares with separable (i.e., rank 1) weights, cf. [11, Section 7]. By choosing the parameters and weights to be the nodes and weights of a quadrature rule, this leads to an approximate L^2-projection. In this example, we approximate the function

$$f(\xi, \eta) = \frac{1}{4} e^{\sqrt{\xi^2+\eta^2}}$$

with the domain $[0,1] \times [0,1]$ using various uniform cubic tensor-product B-spline bases. We compare the two approaches of (a) sampling f in the Greville abscissae and performing low-rank interpolation as in [1, Algorithm 2] and (b) sampling points and weights from a 3-point Gauß-Legendre quadrature rule [28] and performing low-rank weighted fitting of Algorithm `LowRankFit` with $\varepsilon_{\text{abort}} = +\infty$ and $\varepsilon_{\text{accept}} = 0$. In both cases we use adaptive cross approximation with pivoting for `LowRankMatrixApprox`.

The results are reported in Figure 5. In both cases, using a low rank is sufficient for the approximation. However, the L^2-error (approximated using the same quadrature rule) is lower for (b) by approximately one order of magnitude.

Example 4. We demonstrate how one may use Algorithm `LowRankFit` adaptively if the required number of degrees of freedom is not known in advance. One starts with a coarse basis and proceeds until Algorithm `LowRankFit` returns either `Success` (in which case the procedure is terminated) or `CannotReachTolerance`, indicating that a refinement is necessary. The already computed values of σ_r, $\mathbf{u}_r$ and $\mathbf{v}_r$ are re-used, thus spreading the cost of constructing the decomposition (4) over several runs of Algorithm `LowRankFit`.

We test this procedure by approximating again the function from Example 1, this time with $\varepsilon_{\text{abort}} = \varepsilon_{\text{accept}} = 10^{-6}$. Figure 6 depicts the ℓ^2-error depending on r. The horizontal

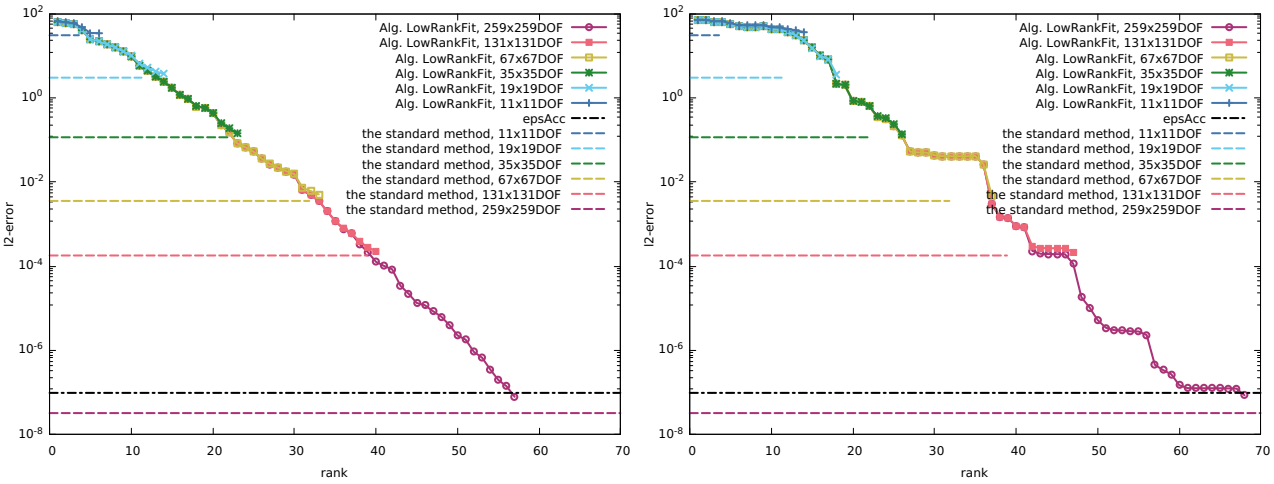

Figure 6: Example 4 – The ℓ^2-errors obtained when using ACA without (left) and with (right) pivoting in `LowRankMatrixApprox` together with the errors of the full gridded fitting and iterations where Algorithm `LowRankFit` terminates.

dashed lines indicate the error of the standard least-squares fitting with the same basis. When using a coarse basis, Algorithm `LowRankFit` exits with `CannotReachTolerance` soon. Once again, using full adaptive cross approximation (instead of pivoting) in `LowRankMatrixApprox` gives slightly better results at higher computational costs.

Table 1 compares the costs by counting the number of calls to `LSsolve`. For each particular basis (corresponding to one of the first six columns), Algorithm `LowRankFit` – regardless whether using ACA with or without pivoting for `LowRankMatrixApprox` – requires much fewer calls to `LSsolve` than the standard tensor-product fitting. The total costs of the entire adaptive procedure, which is obtained by summing up the costs of all the steps, are presented in the last column of Table 1. Note that the total costs (346 or 420 with and without pivoting, respectively) are lower by one order of magnitude, compared to the costs of the full tensor-product fitting. Once more, although these numbers should be interpreted with caution, since we are referring to calls to `LSsolve` with different numbers of unknowns, they nevertheless show the computational advantage of the new algorithm.

$p = q$:	11	19	35	67	131	259	sum
Alg. `LowRankFit` without pivoting	12	28	46	66	80	114	346
Alg. `LowRankFit` with pivoting	28	36	52	74	94	136	420
Full LS fitting	311	319	335	367	431	559	2322

Table 1: Example 4 – Computational costs (in terms of the number of calls to `LSsolve`).

5. Conclusions and outlook

We have presented an efficient algorithm for constructing low-rank approximations of least-squares fitting of gridded data with bivariate B-splines by decomposing it into a series of univariate fitting problems. Future work could be devoted to several topics:

An obvious question is whether it is possible to generalize Algorithm `LowRankFit` to approximations with more than two variables. While generalization of Lemma 1 is straightforward and requires only suitable notation, constructing multivariate analogues to the decomposition (4) is known to be challenging.

Another direction of research could be devoted to the extension of Algorithm `LowRankFit` to non-structured data. This could be attacked by extending the algorithm to general (i.e., not necessarily separable) weights and by exploring the idea of grid projection, see [13]. Alternatively one could also apply a suitable preprocessing step to the data, which might be similar to the idea (create a mesh from the unstructured data points and decompose this piecewise surface into tensor-products of its slices) presented in [25].

Last, but not least, we mention that a few improvements are required before the algorithm can be used in industrial practice. First, applications such as Example 2 require the result to be periodic (and $\mathcal{C}^2$-continuous) in one direction and the algorithm would have to be modified accordingly. Second, choosing $\varepsilon_{\text{accept}}$ and $\varepsilon_{\text{abort}}$ is challenging and suitable values are different for each component. Finally, the algorithm could be made more efficient by exploiting synergies between the results obtained for the three coordinates.

References

[1] I. Georgieva, C. Hofreither, An algorithm for low-rank approximation of bivariate functions using splines, Journal of Computational and Applied Mathematics 310 (2017) 80–91.

[2] G. Kiss, C. Giannelli, U. Zore, B. Jüttler, D. Großmann, J. Barner, Adaptive CAD model (re-)construction with THB-splines, Graphical Models 76 (5) (2014) 273–288.

[3] J. Hoschek, D. Lasser, Fundamentals of computer aided geometric design, AK Peters, Ltd., 1993.

[4] L. Piegl, W. Tiller, The NURBS Book, 2nd Edition, Springer, 1997.

[5] V. Weiss, L. Andor, G. Renner, T. Várady, Advanced surface fitting techniques, Computer Aided Geometric Design 19 (1) (2002) 19–42.

[6] C. Deng, H. Lin, Progressive and iterative approximation for least squares B-spline curve and surface fitting, Computer-Aided Design 47 (2014) 32–44.

[7] D. Rios, B. Jüttler, LSPIA, (stochastic) gradient descent, and parameter correction, Journal of Computational and Applied Mathematics 406 (2022) article no. 113921.

[8] S. Sajavičius, Hyperpower least squares progressive iterative approximation, Journal of Computational and Applied Mathematics 422 (2023) article no. 114888.

[9] C. de Boor, A Practical Guide to Splines, Springer, 1978.

[10] F. Cheng, A. Goshtasby, A parallel B-spline surface fitting algorithm, ACM Transactions on Graphics 8 (1) (1988) 41–50.

[11] T. Lyche, K. Mørken, Spline Methods Draft, Department of Mathematics, University of Oslo, 2018.

[12] N. Engleitner, B. Jüttler, Lofting with patchwork B-splines, in: Advanced Methods for Geometric Modeling and Numerical Simulation, Springer, 2019, pp. 77–98.

[13] S. Merchel, B. Jüttler, D. Mokriš, M. Pan, Fast formation of matrices for least-squares fitting by tensor-product spline surfaces, Computer-Aided Design (2022) article no. 103307.

[14] M. Bebendorf, Approximation of boundary element matrices, Numerische Mathematik 86 (4) (2000) 565–589.

[15] K. Frederix, M. van Barel, Solving a large dense linear system by adaptive cross approximation, Journal of Computational and Applied Mathematics 234 (11) (2010) 3181–3195.

[16] T. Mach, L. Reichel, M. van Barel, R. Vandebril, Adaptive cross approximation for ill-posed problems, Journal of Computational and Applied Mathematics 303 (2016) 206–217.

[17] M. Pan, W. Tong, F. Chen, Compact implicit surface reconstruction via low-rank tensor approximation, computer-aided design 78 (2016) 158–167.
[18] C. Eckart, G. Young, The approximation of one matrix by another of lower rank, Psychometrika 1 (3) (1936) 211–218.
[19] D. W. Fausett, C. T. Fulton, Large least squares problems involving Kronecker products, SIAM Journal on Matrix Analysis and Applications 15 (1) (1994) 219–227.
[20] L. Sun, B. Zheng, C. Bu, Y. Wei, Moore-Penrose inverse of tensors via Einstein product, Linear and Multilinear Algebra 64 (4) (2016) 686–698.
[21] H. Dym, Linear Algebra in Action, 2nd Edition, Americal Mathematical Society, 2013.
[22] P. Dierckx, Curve and surface fitting with splines, Oxford University Press, 1993.
[23] H. Prautzsch, W. Boehm, M. Paluszny, Bézier and B-spline Techniques, Springer, 2002.
[24] A. Mantzaflaris, An overview of geometry plus simulation modules, in: Mathematical Aspects of Computer and Information Sciences: 8th International Conference, MACIS 2019, Gebze, Turkey, November 13–15, 2019, Revised Selected Papers 8, Springer, 2020, pp. 453–456.
[25] A. Townsend, L. N. Trefethen, Gaussian elimination as an iterative algorithm, SIAM News 46 (2) (2013) page 3.
[26] G. Guennebaud, B. Jacob, et al., Eigen v3, http://eigen.tuxfamily.org (2010).
[27] M. Luers, M. Sagebaum, S. Mann, J. Backhaus, D. Großmann, N. R. Gauger, Adjoint-based volumetric shape optimization of turbine blades, Multidisciplinary Analysis and Optimization Conference (2018) article no. AIAA 2018–3638.
[28] J. Stoer, R. Bulirsch, Introduction to Numerical Analysis, Springer, 1980.