NFN - Nationales Forschungsnetzwerk

Geometry + Simulation

http://www.gs.jku.at

Low rank interpolation of boundary spline curves

Bert Jüttler and Dominik Mokriš

G+S Report No. 53

March 2017

Der Wissenschaftsfonds.

JOHANNES KEPLER UNIVERSITY LINZ

Low rank interpolation of boundary spline curves[☆]

Bert Jüttler[a,b], Dominik Mokriš[a,*]

[a] *Institute of Applied Geometry, Johannes Kepler University, Linz, Austria*
[b] *Radon Institute for Computational and Applied Mathematics, Austrian Academy of Sciences*

Abstract

The coefficients of a tensor-product spline surface in $\mathbb{R}^d$ with $m \times n$ control points form a tensor of order 3 and dimension (m, n, d). Motivated by applications in isogeometric analysis we analyze the *rank* of this tensor. In particular, we propose a new construction for low rank tensor-product spline surfaces from given boundary curves. While the results of this construction are generally not affinely invariant, we propose a simple standardization procedure that guarantees affine invariance for $d = 2$. In addition we provide a detailed comparison with existing constructions of spline surfaces from boundary data.

Keywords: spline surface interpolation, domain parametrization, low rank approximation, Coons interpolation, biharmonic interpolation
2010 MSC: 65D17 68U07

1. Introduction and overview

The construction of (tensor-product) spline surfaces from four given boundary curves is one of the classical problems in Computer Aided Geometric Design. *Coons interpolation* is a well-established construction of such surfaces (Coons, 1964; Farin, 2001). A linear interpolant is created separately in each of the two parametric directions and a bilinear interpolant of the patch corners is subtracted from their sum. The method is robust and simple.

Farin and Hansford (1999) introduced the discrete version of Coons interpolation as a special instance of using masks. They propose the class of *permanence patches*. These are spline surfaces where the mask generating the interior control points is a linear blend of the Lagrange-Euler mask (corresponding to discrete Coons patch) and the Laplace mask (corresponding to Laplacian smoothing). In

[☆] Dedicated to the memory of the late Professor Gerald E. Farin.

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

the special case of Bézier surfaces, the discrete Coons patch is the same as Coons patch (Farin, 1992; Farin and Hansford, 1999).

In the case of Bézier surfaces, Monterde and Ugail (2004, 2006) choose the inner control points such that the resulting surface satisfies certain fourth-order partial differential equation (see also Jüttler et al., 2006). This also includes Coons interpolation as a special case. Similarly to Coons interpolation, this approach can also be modified to allow for Hermite interpolation (Centella et al., 2009).

Constructions of surfaces from their boundary curves have found new applications in isogeometric analysis, since they are able to generate spline parametrizations for the computational domain of a numerical simulation from a given boundary representation (see Falini et al., 2015, and the references therein). In this context it has been observed that an analysis and subsequent optimization of the *rank* of a parametrization (which will be discussed in more detail in Section 2 of this paper) can lead to substantial improvements of the overall efficiency of the numerical simulation (Mantzaflaris et al., 2014). This observation has motivated us to explore interpolation techniques that are able to generate low rank spline surfaces.

In the context of implicit *surfaces*, low rank spline representations were used recently by Pan et al. (2016). Their research was motivated by the need to reduce the memory consumption.

After recalling the concept of tensor rank and adapting to the context of spline surfaces we propose an algorithm for coordinate-wise rank-2 interpolation of boundary curves. We note that the results are not affinely invariant as the algorithm does not commute with affine transformations. For the case of planar data we use a transformation to a reference position in order to restore affine invariance.

We show that the new interpolation method satisfies a permanence principle, which is similar to the case of Coons surfaces (Farin and Hansford, 1999), and that it reproduces bilinear surfaces. Using a series of examples we compare the new method with several existing techniques, which include biharmonic interpolation (Monterde and Ugail, 2004), Laplacian smoothing and Coons interpolation. These examples allow us to conclude that both Coons interpolation and our new method produce low rank parametrizations. It should be noted that the methods discussed in this paper do not provide direct theoretical guarantees for injectivity, since they do not possess properties such as a min-max principle, which is known for harmonic mappings. We refer to Gravesen et al. (2014) for a thorough discussion of the injectivity of planar domain parametrizations.

The remainder of the paper is organized as follows. We first introduce the notation and recall the notion of tensor rank. We then formulate our algorithm CR2I for coordinate-wise rank-2 interpolation in Section 3, and we discuss the permanence principle. The following section shows how to restore affine invariance

in the bivariate case by introducing a standard position. Other approaches to boundary interpolation are summarized in Section 5, and this is followed by an example-based comparison with the new methods. Finally we conclude the paper.

2. Preliminaries

We recall the notion of tensor rank and adapt it to the case of spline surfaces.

2.1. Rank of tensor-product spline surfaces

We consider two univariate B-spline bases

$$\boldsymbol{\beta}(s) = [\beta_i(s)]_{i=1,\dots,m} \quad \text{and} \quad \boldsymbol{\tau}(t) = [\tau_j(t)]_{j=1,\dots,n},$$

which are defined by two open knot vectors with boundary knots 0 and 1. It is not assumed that the degrees of the two spline bases are equal but we assume that $m, n \geq 3$. Recall that their tensor product

$$\boldsymbol{\beta}(s) \otimes \boldsymbol{\tau}(t) = [\beta_i(s)\tau_j(t)]_{i=1,\dots,m;j=1,\dots,n}$$

defines the tensor product spline basis. Given a *coefficient tensor*

$$\mathbf{C} = [c_{ijk}]_{i=1,\dots,m;j=1,\dots,n;k=1,\dots,d} \in \mathbb{R}^{m\times n\times d}$$

of order 3 and dimension (m, n, d), we define a *tensor product spline surface* in $\mathbb{R}^d$,

$$\mathbf{p}(s,t) = \mathbf{C} : (\boldsymbol{\beta}(s) \otimes \boldsymbol{\tau}(t)) = \left[\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ijk}\beta_i(s)\tau_j(t)\right]_{k=1,\dots,d}, \tag{1}$$

with the parameter domain $[0, 1]^2$. Here we use the symbol **:** to denote the Frobenius product, which compactly expresses the summation with respect to the indices of the tensor-product basis. In the special case $d = 1$ we call $\mathbf{p}$ *scalar-valued spline function* and sometimes write simply p instead of $\mathbf{p}$. If $d = 2$, we call $\mathbf{p}$ a *planar parametrization*, because if $\mathbf{p}$ is bijective, it parametrizes the domain enclosed by the planar boundary curves.

Any tensor of order 3 admits a representation as a finite sum of tensor-products of vector triplets,

$$\mathbf{C} = \sum_{r=1}^{R} \mathbf{u}^r \otimes \mathbf{v}^r \otimes \mathbf{w}^r, \quad \text{or, equivalently,} \quad c_{ijk} = \sum_{r=1}^{R} u_i^r v_j^r w_k^r \tag{2}$$

for some vectors

$$\mathbf{u}^r = [u_i^r]_{i=1,\dots,m} \in \mathbb{R}^m, \quad \mathbf{v}^r = [v_j^r]_{j=1,\dots,n} \in \mathbb{R}^n \quad \text{and} \quad \mathbf{w}^r = [w_k^r]_{k=1,\dots,d} \in \mathbb{R}^d.$$

Clearly, each tensor possesses infinitely many representations of this form. The minimum number R for all possible representations is called the *tensor rank* of $\mathbf{C}$. In particular, the null tensor has rank 0.

This representation was introduced in order to address the "curse of dimension" concerning the memory requirements, especially for higher order tensors. Storing the representation (2) requires $O(R(m+n+d))$ memory, while the full coefficient tensor has mnd elements. Using the representation (2) is advantageous if the tensor rank satisfies $R \ll \min\{m, n\}$.

Any four vectors $\mathbf{q}$, $\mathbf{r}$, $\mathbf{s}$, $\mathbf{t}$ satisfy the identity $(\mathbf{q} \otimes \mathbf{r}) : (\mathbf{s} \otimes \mathbf{t}) = (\mathbf{q} \cdot \mathbf{s})(\mathbf{r} \cdot \mathbf{t})$. Combining this fact with the representation (2), we see that the associated spline surface can be written as

$$\mathbf{p}(s,t) = \sum_{r=1}^{R} (\mathbf{u}^r \cdot \boldsymbol{\beta}(s)) \, (\mathbf{v}^r \cdot \boldsymbol{\tau}(t)) \, \mathbf{w}^r, \tag{3}$$

where "$\cdot$" denotes the standard inner product of vectors. We then say that $\mathbf{p}$ is a *rank R spline surface* and R is simply called the *rank* of $\mathbf{p}$.

It should be noted that the representation (3) can be used to evaluate a spline surface. It shows that a specific point on a rank-R spline surface can be computed by evaluating $2R$ univariate spline functions, whereas using the standard tensor-product representation (1) requires $d(\delta + 2)$ such evaluations, where δ is the polynomial degree. The evaluation of each spline function requires $O(\delta^2)$ operations if performed by knot insertion. This, however, may not be the optimal approach since one may re-use values of B-splines.

2.2. Affine mappings

The tensor rank of the coefficient tensor $\mathbf{C}$ is generally not preserved by affine transformations of $\mathbb{R}^d$, i.e., the coefficients of the spline functions $\mathbf{p}$ and $\alpha \circ \mathbf{p}$, where α is an affine mapping, may have different tensor ranks. Indeed, consider the spline functions $\mathbf{p} = 0$ and $\mathbf{p}' = 1$ for $d = 1$, which are related by an affine mapping since $x' = x + 1$. Their coefficients have rank 0 and 1.

We analyze the effect that affine mappings can have on the rank.

Lemma 1. *Affine mappings increase the rank of a spline surface by at most one.*

Proof. We consider a spline surface (3) with coefficients possessing tensor rank R, and an affine mapping

$$\alpha : \mathbf{x} \mapsto \alpha(\mathbf{x}) = \mathbf{a} + A\mathbf{x}$$

defined by a vector $\mathbf{a} \in \mathbb{R}^d$ and a matrix $A \in \mathbb{R}^{d \times d}$. We distinguish whether the function identically equal to one can be spanned by $\boldsymbol{\beta}(s)$ and $\boldsymbol{\tau}(t)$ or not, i.e., whether

$$1 \in \operatorname{span}\{ \, (\mathbf{u}^r \cdot \boldsymbol{\beta}(s))(\mathbf{v}^r \cdot \boldsymbol{\tau}(t)) \mid r = 1, \ldots, R \, \}. \tag{4}$$

First, we assume that (4) is satisfied. Thus, there exist coefficients ζ^r satisfying

$$1 = \sum_{r=1}^{R} (\mathbf{u}^r \cdot \boldsymbol{\beta}(s))(\mathbf{v}^r \cdot \boldsymbol{\tau}(t))\zeta^r.$$

The transformed spline surface

$$(\alpha \circ \mathbf{p})(s,t) = \sum_{r=1}^{R} (\mathbf{u}^r \cdot \boldsymbol{\beta}(s)) \, (\mathbf{v}^r \cdot \boldsymbol{\tau}(t)) \, (\zeta^r \mathbf{a} + A\mathbf{w}^r),$$

has coefficients

$$\sum_{r=1}^{R} \mathbf{u}^r \otimes \mathbf{v}^r \otimes (\zeta^r \mathbf{a} + A\mathbf{w}^r).$$

The tensor rank of the transformed coefficients thus does not exceed R.

Second, we consider the situation where (4) is not satisfied. We use the partition of unity property of B-splines. The transformed spline surface

$$(\alpha \circ \mathbf{p})(s,t) = \left(\sum_{r=1}^{R} (\mathbf{u}^r \cdot \boldsymbol{\beta}(s)) \, (\mathbf{v}^r \cdot \boldsymbol{\tau}(t)) \, A\mathbf{w}^r \right) + (\mathbb{1}^m \cdot \boldsymbol{\beta}(s)) \, (\mathbb{1}^n \cdot \boldsymbol{\tau}(t))\mathbf{a}$$

then has coefficients

$$\left(\sum_{r=1}^{R} \mathbf{u}^r \otimes \mathbf{v}^r \otimes (A\mathbf{w}^r) \right) + \mathbb{1}^m \otimes \mathbb{1}^n \otimes \mathbf{a},$$

where $\mathbb{1}^k \in \mathbb{R}^k$ is a vector with all elements equal to 1. The tensor rank then does not exceed $R + 1$. □

The group structure of regular affine mappings implies that *the ranks of any two affinely equivalent spline surfaces differ at most by one.*

2.3. Bounds on the tensor rank

Computing the tensor rank is a difficult problem (Kolda and Bader, 2009); in fact, the problem is known to be NP-hard (Håstad, 1990). A notable exception is the case $d = 2$, where the tensor rank can be obtained by analyzing the Kronecker form of a linear matrix pencil (Ja'Ja', 1979; Landsberg, 2012). Even in this situation, however, the computation of the tensor rank is quite involved. We will therefore work with well-known upper and lower bounds instead.

Recall that a *matricization* $\mathrm{mat}(\mathbf{C})$ of a tensor $\mathbf{C}$ is obtained by splitting the indices into two subsets and applying lexicographic ordering (possibly after permuting the indices) to each subset individually, cf. Kolda and Bader (2009).

As a first observation, we note that *the tensor rank of* $\mathbf{C}$ *is greater than or equal to the matrix rank of any matricization* $\mathrm{mat}(\mathbf{C})$. For a proof, we consider the matricization obtained by keeping the first index and combining the last two indices lexicographically and obtain

$$\mathrm{mat} \sum_{r=1}^{R} \mathbf{u}^r \otimes \mathbf{v}^r \otimes \mathbf{w}^r = \sum_{r=1}^{R} \mathbf{u}^r \otimes \mathrm{vec}(\mathbf{v}^r \otimes \mathbf{w}^r) = [\mathbf{u}^r]_{r=1,\dots,R} \cdot [\mathrm{vec}(\mathbf{v}^r \otimes \mathbf{w}^r)]_{r=1,\dots,R}.$$

The rank of the matricization thus does not exceed $\mathrm{rank}[\mathbf{u}^r]_{r=1,\dots,R} \leq R$. The same argument applies to any other matricization.

Second, if we consider the matrix slices of $\mathbf{C}$ obtained by considering fixed values for one of the three indices, then *the tensor rank does not exceed the sum of the ranks of these matrix slices.* To see this, note that using the canonical unit vectors $\mathbf{e}_k = [\delta_\ell^k]_{\ell=1,\dots,d}$, we obtain

$$\mathbf{C} = \sum_{k=1}^{d} \sum_{r=1}^{R_k} \mathbf{u}_k^r \otimes \mathbf{v}_k^r \otimes \mathbf{e}_k, \quad \text{where} \quad [c_{ijk}]_{ij} = \sum_{r=1}^{R_k} \mathbf{u}_k^r \otimes \mathbf{v}_k^r$$

are rank R_k decompositions of the tensor slices obtained by fixing the index k.

For simplicity, assume $d = 2$ now. The rank of a spline parametrization gives upper bounds on ranks of several derived quantities, provided that these can again be represented as spline functions (possibly with different degrees and knot vectors). These quantities include the determinant of the Jacobian matrix (which arises frequently in isogeometric analysis and differential geometry) and the elements g_{ij} of the first fundamental form.

Lemma 2. *The ranks of the spline functions* $\det \nabla \mathbf{p}$*,* g_{ii} *for* $i = 1, 2$*, and* g_{12}*, are bounded by* $R(R-1)$*,* $\binom{R+1}{2}$ *and* R^2*, respectively, if* $\mathbf{p} : \mathbb{R}^2 \to \mathbb{R}^2$ *is a rank* R *planar parametrization.*

Proof. Using $\mathbf{w}^r = [w_k^r]_{k=1,2}$ in (3) gives

$$\det(\nabla \mathbf{p}) = \sum_{r=1}^{R} \sum_{p=1}^{R} \underbrace{(\mathbf{u}^r \cdot \boldsymbol{\beta}(s)')(\mathbf{u}^p \cdot \boldsymbol{\beta}(s))}_{(a)} \underbrace{(\mathbf{v}^r \cdot \boldsymbol{\tau}(t))(\mathbf{v}^p \cdot \boldsymbol{\tau}(t)')}_{(b)} (w_1^r w_2^p - w_2^r w_1^p).$$

The right-hand side can again be transformed into a form analogous to (3) by introducing representations of the terms (a) and (b) with respect to suitable spline bases. This proves the first bound as the sum has $(R^2 - R)$ non-zero elements. Similar techniques can be used to prove the remaining two bounds if we exploit the symmetry of the sum in the case of g_{ii}. □

3. Coordinate-wise rank-2 interpolation

We consider the following problem: Given four spline curves in $\mathbb{R}^d$ that form a simple closed loop, we want to find a bivariate spline surface $\mathbf{p}(s,t)$ in $\mathbb{R}^d$ that interpolates these four curves along the patch boundaries.

More precisely, denote these curves by

$$\mathbf{p}(s,0) = [[c_{i1k}\ ]_{i=1,\dots,m} \cdot \boldsymbol{\beta}(s)]_{k=1,\dots,d}, \tag{5}$$
$$\mathbf{p}(0,t) = [[c_{1jk}\ ]_{j=1,\dots,n} \cdot \boldsymbol{\tau}(t)\ ]_{k=1,\dots,d}, \tag{6}$$
$$\mathbf{p}(1,t) = [[c_{mjk}]_{j=1,\dots,n} \cdot \boldsymbol{\tau}(t)\ ]_{k=1,\dots,d}, \tag{7}$$
$$\mathbf{p}(s,1) = [[c_{ink}\ ]_{i=1,\dots,m} \cdot \boldsymbol{\beta}(s)]_{k=1,\dots,d}. \tag{8}$$

The control points of the boundary curves, i.e., $[c_{ijk}]_{k=1,\dots,d} \in \mathbb{R}^d$ with $i \in \{1,m\}$ or $j \in \{1,n\}$, are given. Our task consists in finding the remaining elements of the coefficient tensor $\mathbf{C}$. Among the many possible solutions we identify one that gives a low rank spline surface.

The four boundary curves are called *matching* if they form a simple loop and if the knot vectors of $\mathbf{p}(s,0)$ and $\mathbf{p}(0,t)$ are equal to the knot vectors of $\mathbf{p}(s,1)$ and $\mathbf{p}(1,t)$, respectively. Furthermore, the four corner points are called *feasible* if the condition

$$\begin{vmatrix} c_{11k} & c_{1nk} \\ c_{m1k} & c_{mnk} \end{vmatrix} \neq 0$$

is satisfied for each coordinate, i.e., for $k = 1, \dots, d$.

3.1. Rank 2d interpolation algorithm

Before proposing our first algorithm we present a technical lemma.

Lemma 3. *Consider a real matrix*

$$B = \begin{pmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{m1} & \cdots & b_{mn} \end{pmatrix} \quad \textit{with} \quad \begin{vmatrix} b_{11} & b_{1n} \\ b_{m1} & b_{mn} \end{vmatrix} \neq 0.$$

Then B possesses rank 2 if and only if

$$b_{ij} = \frac{b_{i1}\begin{vmatrix} b_{1j} & b_{1n} \\ b_{mj} & b_{mn} \end{vmatrix} + b_{in}\begin{vmatrix} b_{11} & b_{1j} \\ b_{m1} & b_{mj} \end{vmatrix}}{\begin{vmatrix} b_{11} & b_{1n} \\ b_{m1} & b_{mn} \end{vmatrix}} \tag{9}$$

for all $1 < i < m$ and $1 < j < n$.

Proof. Firstly, the rank of B does not exceed 2, because the j-th column of B is a linear combination of the first column and the last column:

$$[b_{ij}]_{i=1,\dots,m} = \frac{\begin{vmatrix} b_{1j} & b_{1n} \\ b_{mj} & b_{mn} \end{vmatrix}}{\begin{vmatrix} b_{11} & b_{1n} \\ b_{m1} & b_{mn} \end{vmatrix}} [b_{i1}]_{i=1,\dots,m} + \frac{\begin{vmatrix} b_{11} & b_{1j} \\ b_{m1} & b_{mj} \end{vmatrix}}{\begin{vmatrix} b_{11} & b_{1n} \\ b_{m1} & b_{mn} \end{vmatrix}} [b_{in}]_{i=1,\dots,m}.$$

This is a consequence of (9) for $1 < j < n$ and it its trivially satisfied for $j = 1$ and $j = n$.

Secondly, the rank of B is then equal to 2, as the elements at the four corners form a regular 2×2 submatrix. □

We now formulate a simple method for Coordinate-wise Rank-2 Interpolation in Algorithm CR2I. It takes the control points of the four matching boundary curves with feasible corner points as input and returns the control points of an interpolating spline surface. The d slices of the coefficient tensor are considered individually. The k-th coordinate of the interior control points is computed from the k-th coordinate of the boundary control points by applying Lemma 3 to the k-th slice.

Algorithm CR2I: Coordinate-wise rank-2 interpolation

input : Control points of the boundary curves, i.e., c_{ijk} with (i, j, k) from $(\{1, m\} \times \{1, \dots, n\} \times \{1, \dots, d\}) \cup (\{1, \dots, m\} \times \{1, n\} \times \{1, \dots, d\})$

output: All control points, i.e., c_{ijk} with $(i, j, k) \in \{1, \dots, m\} \times \{1, \dots, n\} \times \{1, \dots, d\}$

for $k = 1, \dots, d$ **do**

  $\Delta = \begin{vmatrix} c_{11k} & c_{1nk} \\ c_{m1k} & c_{mnk} \end{vmatrix}$;

  for $j = 2, \dots, n-1$ **do**

    $\lambda = \frac{1}{\Delta} \begin{vmatrix} c_{1jk} & c_{1nk} \\ c_{mjk} & c_{mnk} \end{vmatrix}$; $\quad \rho = \frac{1}{\Delta} \begin{vmatrix} c_{11k} & c_{1jk} \\ c_{m1k} & c_{mjk} \end{vmatrix}$;

    for $i = 2, \dots, m-1$ **do**

      $c_{ijk} = \lambda c_{i1k} + \rho c_{ink}$;

    end

  end

end

Corollary 4. *Algorithm CR2I returns a spline surface* $\mathbf{p} : \mathbb{R}^2 \to \mathbb{R}^d$ *with the property that the rank of each coordinate function is equal to 2 if the given boundary*

curves in $\mathbb{R}^d$ are matching and the corner points are feasible. Consequently, the rank of $\mathbf{p}$ does not exceed $2d$.

Proof. These facts are implied by Lemma 3 and by the upper bound on the tensor rank which was presented in Section 2.3. □

Interestingly, (9) has the following continuous analogue.

Proposition 5. *Consider a scalar-valued spline surface $p(s,t)$ with*

$$\begin{vmatrix} p(0,0) & p(0,1) \\ p(1,0) & p(1,1) \end{vmatrix} \neq 0. \tag{10}$$

Then $p(s,t)$ is of rank 2 if and only if

$$\forall s,t \in [0,1]: \begin{vmatrix} p(0,0) & p(0,t) & p(0,1) \\ p(s,0) & p(s,t) & p(s,1) \\ p(1,0) & p(1,t) & p(1,1) \end{vmatrix} = 0. \tag{11}$$

Proof. Since $d = 1$, Eq. (1) simplifies to

$$p(s,t) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} \beta_i(s) \tau_j(t).$$

We use the endpoint interpolation property to rewrite (11) as

$$\begin{vmatrix} c_{11} & \sum_j c_{1j}\tau_j(t) & c_{1n} \\ \sum_i c_{i1}\beta_i(s) & \sum_i \sum_j c_{ij}\beta_i(s)\tau_j(t) & \sum_i c_{in}\beta_i(s) \\ c_{m1} & \sum_j c_{mj}\tau_j(t) & c_{mn} \end{vmatrix} = 0,$$

which can be rearranged into

$$\sum_{i=1}^{m} \sum_{j=1}^{n} \left(c_{i1} \begin{vmatrix} c_{1j} & c_{1n} \\ c_{mj} & c_{mn} \end{vmatrix} - c_{ij} \begin{vmatrix} c_{11} & c_{1n} \\ c_{m1} & c_{mn} \end{vmatrix} + c_{in} \begin{vmatrix} c_{11} & c_{1j} \\ c_{m1} & c_{mj} \end{vmatrix} \right) \beta_i(s)\tau_j(t) = 0.$$

All coefficients of this spline surface are equal to zero and this is exactly the condition of Lemma 3, hence $\operatorname{rank}([c_{ij}]_{ij}) = 2$. □

Corollary 6. *There exists exactly one rank-2 scalar-valued spline function $p(s,t)$ that interpolates a given loop of matching boundary curves with feasible corners.*

Proof. Algorithm CR2I guarantees the existence and Lemma 3 the uniqueness. □

3.2. Permanence principle

Other interpolation methods often possess the *permanence principle*: if one performs the interpolation and restricts the surface to $[s_1, s_2] \times [t_1, t_2] \subset [0,1]^2$, the result is the same as applying the interpolation to the boundary curves of this restricted surface. The permanence principle is satisfied for Coons interpolation (Farin and Hansford, 1999) and we now show that it applies to our method as well. We start the derivation with several technical observations for scalar-valued spline functions, i.e., with $d = 1$.

Lemma 7. *Neither knot insertion nor degree elevation increases the rank of a scalar-valued spline function $p(s,t)$.*

Proof. Since $d = 1$ and p is of rank R, Eq. (3) can be simplified into

$$p(s,t) = \sum_{r=1}^{R} (\mathbf{u}^r \cdot \boldsymbol{\beta}(s)) \, (\mathbf{v}^r \cdot \boldsymbol{\tau}(t)). \tag{12}$$

Knot insertion and degree elevation (without loss of generality in s-direction) do not change the value of any of the summands,

$$(\mathbf{u}^r \cdot \boldsymbol{\beta}(s)) \, (\mathbf{v}^r \cdot \boldsymbol{\tau}(t)) = (\widehat{\mathbf{u}}^r \cdot \widehat{\boldsymbol{\beta}}(s)) \, (\mathbf{v}^r \cdot \boldsymbol{\tau}(t)), \quad r = 1, \ldots, R, \tag{13}$$

where $\widehat{\boldsymbol{\beta}}(s)$ is the basis $\boldsymbol{\beta}(s)$ after applying knot insertion or degree elevation and $\widehat{\mathbf{u}}^r$ is the corresponding updated coefficient vector. This carries over to (12),

$$p(s,t) = \sum_{r=1}^{R} (\widehat{\mathbf{u}}^r \cdot \widehat{\boldsymbol{\beta}}(s)) \, (\mathbf{v}^r \cdot \boldsymbol{\tau}(t)),$$

and we arrive at another rank-R representation of p. □

Corollary 8. *Let $p(s,t)$ be a scalar-valued spline function of rank R. Then the rank of its restriction*

$$(s,t) \mapsto p(s_1 + s(s_2 - s_1), t_1 + t(t_2 - t_1)), \quad (s,t) \in [0,1]^2,$$

to $[s_1, s_2] \times [t_1, t_2] \subseteq [0,1]^2$ does not exceed R.

Proof. The coefficients of the restriction are obtained by inserting the knots s_1, s_2 and t_1, t_2 repeatedly and taking a submatrix of the resulting coefficient matrix. □

Theorem 9 (Permanence principle)**.** *Let $\mathbf{p}(s,t)$ be a spline surface constructed by Algorithm CR2I and let $0 \le s_1 < s_2 \le 1$ and $0 \le t_1 < t_2 \le 1$ be chosen so that the*

corner points $\mathbf{p}(s_1,t_1)$, $\mathbf{p}(s_2,t_1)$, $\mathbf{p}(s_1,t_2)$ *and* $\mathbf{p}(s_2,t_2)$ *are feasible. Furthermore, let* $\widehat{\mathbf{p}}(s,t)$ *be the surface generated by applying Algorithm CR2I to the curves*

$$\mathbf{p}(s_1{+}\cdot(s_2{-}s_1),t_1),\ \mathbf{p}(s_1,t_1{+}\cdot(t_2{-}t_1)),\ \mathbf{p}(s_1{+}\cdot(s_2{-}s_1),t_2) \text{ and } \mathbf{p}(s_2,t_1{+}\cdot(t_2{-}t_1)).$$

Then the restriction of the first surface to $[s_1,s_2]\times[t_1,t_2]$ *is equal to the second one,*

$$\mathbf{p}(s_1+s(s_2-s_1),t_1+t(t_2-t_1))=\widehat{\mathbf{p}}(s,t)\quad\forall s,t\in[0,1].$$

Proof. It suffices to prove the theorem for scalar-valued spline functions p and $\widehat{p}$ since Algorithm CR2I deals with each coordinate individually. We use Corollary 4 with $d=1$ and the feasibility of the corner points to conclude that p and $\widehat{p}$ have rank 2. The restriction of the first surface to $[s_1,s_2]\times[t_1,t_2]$ has the same rank due to Corollary 8. Finally we use Corollary 6 to complete the proof. □

3.3. Numerical results

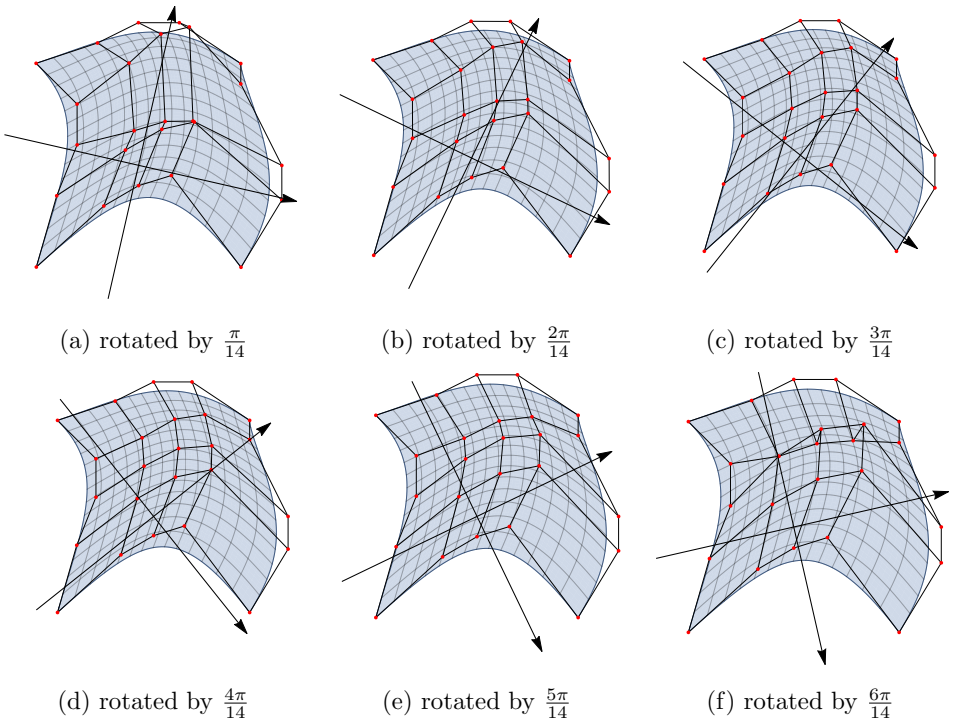
We conclude this section with three examples.

Example 10. We consider a given loop of quartic Bézier curves. Algorithm CR2I cannot be applied directly as the corner points are infeasible. We restored feasibility by applying rotations (with different angles) to the coordinate system. The origin remains in the center of gravity of the four corner points.

Figure 1 shows the results produced by Algorithm CR2I after applying rotations with angles $i\frac{\pi}{14}$ with $i=1,\ldots,6$. We consider only these values of i since a rotation by $\frac{\pi}{2}$ is just a swap and a sign change of the coordinate axes.

Example 11. Algorithm CR2I can be used for $d=3$ as well. We consider an input loop which is similar to that from Fig. 2 of Monterde and Ugail (2004). It has been translated so that the origin is in the centroid of the four corner points and then rotated around the x, y and z axes by $\frac{\pi}{3}$, $-\frac{\pi}{7}$ and $-\frac{\pi}{18}$, respectively, in order to obtain feasible input. Figure 2(a) depicts the result transformed back to the original position.

Example 12. Figure 2(b) shows the results of Algorithm CR2I on the input from Fig. 5 of Monterde and Ugail (2004); since it is again an invalid input, it has been rotated around the x, y and z axes by $\pi/3$, $\pi/4$ and $-\pi/6$, respectively. However, different choices of these angles lead to very similar results. It is also interesting to compare with the examples in Fig. 1, 2 and 3 of Farin and Hansford (1999), where a similar input is used, except that it has more control points (Adding further points would not change the result, see Proposition 17); our result is close to what they term "optimal" control net, in the sense it coincides with "designer's intent".



(a) rotated by $\frac{\pi}{14}$ (b) rotated by $\frac{2\pi}{14}$ (c) rotated by $\frac{3\pi}{14}$

(d) rotated by $\frac{4\pi}{14}$ (e) rotated by $\frac{5\pi}{14}$ (f) rotated by $\frac{6\pi}{14}$

Figure 1: Results of Algorithm CR2I for quartic boundary curves (Example 10) and rotated coordinate systems.

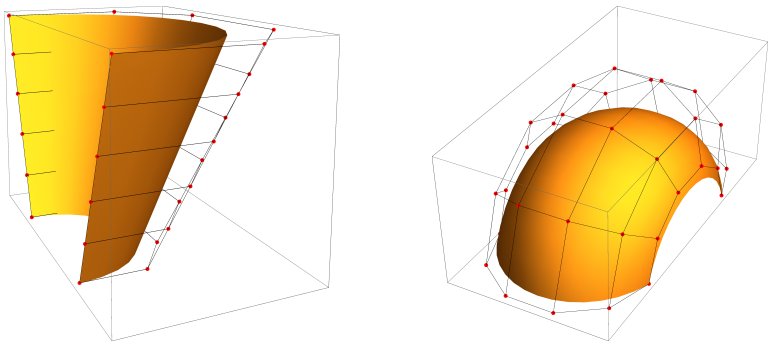
We have seen that Algorithm CR2I is not affinely invariant, i.e., that it does not commute with affine transformations. Although this can be considered a disadvantage, the choice of an appropriate affine coordinate system gives additional degrees of freedom (for example, five of them if $d = 2$) that can be used as design parameters. One might try several coordinate systems and choose the one where some fairness measure of the result – such as the maximal aspect ratio of the elements – is optimal.

3.4. Invariance in special situations

The application of Algorithm CR2I commutes with any non-uniform scaling of the input data, where each coordinate is multiplied by a non-zero factor. Indeed, such a scaling does not change the rank of the coefficient matrices for each coordinate, and the algorithm treats every coordinate separately.

We identify conditions on the input data which guarantee that Algorithm CR2I commutes with two other, more general, classes of affine mappings.

Two curves $\mathbf{f}, \mathbf{g} : [0, 1] \to \mathbb{R}^d$ are said to be *TS-equivalent* if there is a composition of a translation and a non-uniform scaling that transforms the first curve



(a) Example 11 (b) Example 12

Figure 2: Surfaces to compare with Fig. 2 of Monterde and Ugail (2004) and Fig. 2 of Farin and Hansford (1999).

into the second one, i.e., if there exist a vector $\mathbf{a} \in \mathbb{R}^d$ and non-zero scalars $q_1, \ldots, q_d \in \mathbb{R}$ such that

$$\mathbf{g}(s) = \mathbf{a} + \operatorname{diag}(q_1, \ldots, q_d)\mathbf{f}(s) \text{ for all } s \in [0, 1].$$

Proposition 13. *Consider admissible data and translations that transform it into data which are again admissible. Algorithm CR2I commutes with these translations if one of the two pairs of opposite boundaries are TS-equivalent.*

Proof. It suffices to consider the one-dimensional case ($d = 1$), since both the translations and Algorithm CR2I deal with each coordinate separately. We assume that the northern and the southern boundary curve are TS-equivalent, i.e., $c_{mj} = a + qc_{1j}$ for $j = 1, \ldots, n$ and some $a, q \in \mathbb{R}$. The data are admissible only if $a \neq 0$.

Using the TS equivalence of the two boundaries we obtain the identity

$$\begin{vmatrix} c_{11}+v & c_{1j}+v & c_{1n}+v \\ c_{i1}+v & c_{ij}+v & c_{in}+v \\ c_{m1}+v & c_{mj}+v & c_{mn}+v \end{vmatrix} = (a+v-qv)[(c_{11}-c_{1n})c_{ij}+(c_{1j}-c_{11})c_{in}+(c_{1n}-c_{1j})c_{i1}],$$

where $v \in \mathbb{R}$ describes the translation. First we consider the case of the initial data, i.e., we choose $v = 0$. Clearly, the determinant vanishes if the central element takes the value c_{ij} which is generated by Algorithm CR2I. Consequently, the term in the square brackets vanishes, as the admissibility of the initial data implies $a \neq 0$. Therefore, the determinant also vanishes for the translated data (i.e., for a non-zero value of v), provided that the same translation is applied to the central element. Consequently, Algorithm CR2I assigns the value $c_{ij} + v$ to the central element due to the uniqueness of rank-2 interpolation for admissible data. The proof for TS-equivalent eastern and western boundaries is analogous. □

Two curves $\mathbf{f}, \mathbf{g} : [0,1] \to \mathbb{R}^d$ are said to be *L-equivalent* if there exists a linear mapping (i.e., an affine transformation with fixed point $\mathbf{0}$) that transforms the first boundary into the second one, i.e., if there exists a matrix A such that

$$\mathbf{g}(s) = A\mathbf{f}(s) \text{ for all } s \in [0,1].$$

Proposition 14. *Consider admissible data in the planar case ($d = 2$) and linear transformations that transform it into data which are again admissible. Algorithm CR2I commutes with linear mappings if one of the two pairs of opposite boundaries are L-equivalent.*

Proof. We assume that the northern and the southern boundary curve are L-equivalent, i.e.,

$$\begin{bmatrix} c_{mj1} \\ c_{mj2} \end{bmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{bmatrix} c_{1j1} \\ c_{1j2} \end{bmatrix}$$

for $j = 1, \ldots, n$. The data are admissible only if $a_{12} \neq 0$ and $a_{21} \neq 0$.

Take a linear mapping described by the matrix

$$\hat{A} = \begin{pmatrix} \hat{a}_{11} & \hat{a}_{12} \\ \hat{a}_{21} & \hat{a}_{22} \end{pmatrix}.$$

Using the L-equivalence of the two boundaries we obtain the identity

$$\begin{vmatrix} \hat{a}_{k1}c_{111} + \hat{a}_{k2}c_{112} & \hat{a}_{k1}c_{1j1} + \hat{a}_{k2}c_{1j2} & \hat{a}_{k1}c_{1n1} + \hat{a}_{k2}c_{1n2} \\ \hat{a}_{k1}c_{i11} + \hat{a}_{k2}c_{i12} & \hat{a}_{k1}c_{ij1} + \hat{a}_{k2}c_{ij2} & \hat{a}_{k1}c_{in1} + \hat{a}_{k2}c_{in2} \\ \hat{a}_{k1}c_{m11} + \hat{a}_{k2}c_{m12} & \hat{a}_{k1}c_{mj1} + \hat{a}_{k2}c_{mj2} & \hat{a}_{k1}c_{mn1} + \hat{a}_{k2}c_{mn2} \end{vmatrix}$$

$$= \left(\hat{a}_{k1} \begin{vmatrix} a_{11} & a_{12} \\ \hat{a}_{k1} & \hat{a}_{k2} \end{vmatrix} + \hat{a}_{k2} \begin{vmatrix} a_{21} & a_{22} \\ \hat{a}_{k1} & \hat{a}_{k2} \end{vmatrix} \right) \left(\hat{a}_{k1} \underbrace{\begin{vmatrix} c_{111} & c_{1j1} & c_{1n1} \\ c_{112} & c_{1j2} & c_{1n2} \\ c_{i11} & c_{ij1} & c_{in1} \end{vmatrix}}_{(i)} + \hat{a}_{k2} \underbrace{\begin{vmatrix} c_{111} & c_{1j1} & c_{1n1} \\ c_{112} & c_{1j2} & c_{1n2} \\ c_{i12} & c_{ij2} & c_{in2} \end{vmatrix}}_{(ii)} \right)$$

for $k = 1, 2$. First we consider the case of the initial data, where $\hat{a}_{ij} = \delta_{ij}$ with δ_{ij} denoting the Kronecker delta. For $k = 1$ the determinant on the left-hand side vanishes if the central element takes the value c_{ij1} which is generated by Algorithm CR2I. The right-hand side simplifies to the first 3×3 determinant (i), multiplied by a_{12}. Consequently, this determinant is equal to zero, as the admissibility of the initial data implies $a_{12} \neq 0$. Analogously, for $k = 2$ when considering c_{ij2} generated by Algorithm CR2I and taking the admissibility condition $a_{21} \neq 0$ into account, we conclude that the second 3×3 determinant (ii) on the right-hand side vanishes also.

Thus, the entire right-hand side is equal to zero for any choice of $\hat{A}$. Consequently, Algorithm CR2I assigns the value $\hat{a}_{k1}c_{ij1} + \hat{a}_{k2}c_{ij2}$ to the central element, due to the uniqueness of rank-2 interpolation for admissible data.

The proof for L-equivalent eastern and western boundaries is analogous. □

In particular, the algorithm commutes with rotations around the origin $\mathbf{0}$ if the assumptions of this proposition are satisfied. One should also note that Algorithm CR2I commutes with arbitrary affine transformations if one of the two pairs of opposite boundaries are linearly parametrized line segments and $d = 2$. This can be seen by combining the two sufficient conditions.

4. Ensuring affine invariance

We complement Algorithm CR2I by suitable pre- and postprocessing steps to achieve affine invariance. The results are limited to planar spline parametrizations ($d = 2$).

4.1. Description of the algorithm

We propose Algorithm AR5I, which is illustrated by Figure 3. It consists of three steps, namely transformation to a standard position (step 1), coordinate-wise rank-2 interpolation (step 2), and back transformation of the interpolating surface (step 3).

Algorithm AR5I: Affinely invariant rank-5 interpolation.

input : Control points of the boundary curves, i.e., c_{ijk} with (i, j, k) from $(\{1, m\} \times \{1, \ldots, n\} \times \{1, 2\}) \cup (\{1, \ldots, m\} \times \{1, n\} \times \{1, 2\})$

output: All control points, i.e., c_{ijk} with $(i, j, k) \in \{1, \ldots, m\} \times \{1, \ldots, n\} \times \{1, 2\}$

`/* Step 1: Transform the input into standard position: */`

A=`transformationMatrix(` $[c_{11k}]_{k=1,2}$, $[c_{1nk}]_{k=1,2}$, $[c_{m1k}]_{k=1,2}$, $[c_{mnk}]_{k=1,2}$`);`

for *boundary control points* **do**

  $[x_{ij}, y_{ij}, 1]^T = A[c_{ij1}, c_{ij2}, 1]^T$;

end

`/* Step 2: Apply Algorithm CR2I: */`

$[\tilde{c}_{ijk}]_{i=1,\ldots,m,j=1,\ldots,n,k=1,2}$ = Algorithm CR2I($[x_{ij}, y_{ij}]_{ij}$);

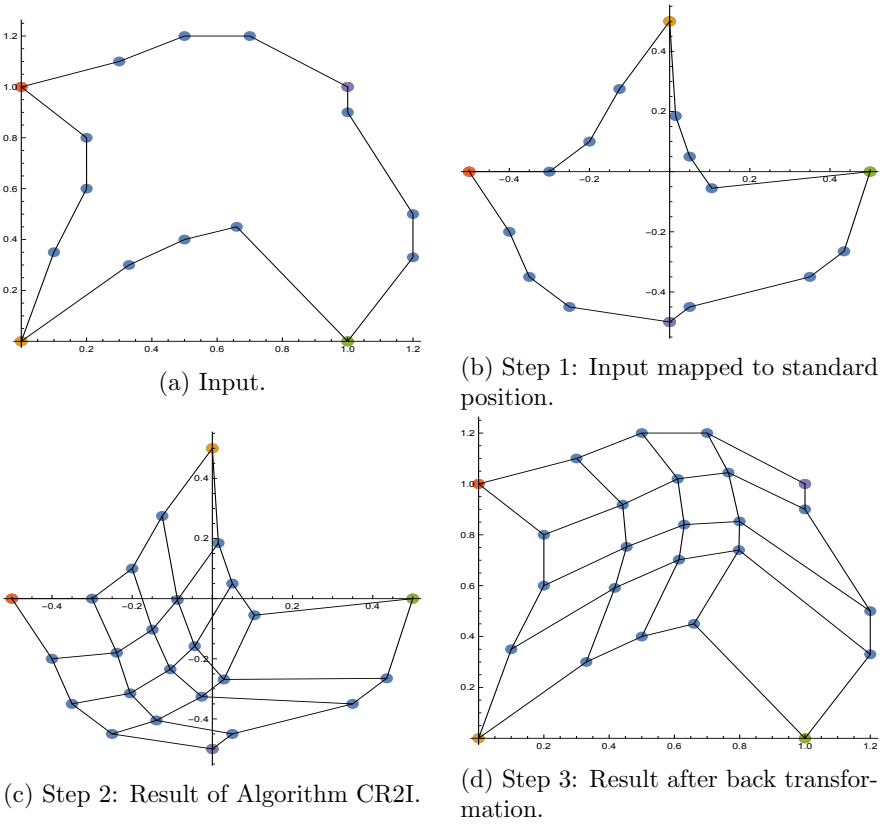
`/* Step 3: Back transformation of the result: */`

for *all control points* **do**

  $[c_{ij1}, c_{ij2}, 1]^T = A^{-1}[\tilde{c}_{ij1}, \tilde{c}_{ij2}, 1]^T$;

end

The algorithm uses the procedure `transformationMatrix` to generate the matrix A that described the transformation to a *standard position*. More precisely, we transform the diagonals $[c_{1nk} - c_{m1k}]_{k=1,2}$ and $[c_{11k} - c_{mnk}]_{k=1,2}$ into vectors of unit length on x and y axes, respectively, and we map their intersection to the



(a) Input.

(b) Step 1: Input mapped to standard position.

(c) Step 2: Result of Algorithm CR2I.

(d) Step 3: Result after back transformation.

Figure 3: The three steps of Algorithm AR5I. The dots represent the control points.

origin $[0, 0]$, see Figure 3(b). Using homogeneous coordinates,

$$A = \begin{pmatrix} M_{11} & M_{12} & -M_{11}c_{111} - M_{12}c_{112} \\ M_{21} & M_{22} & -M_{21}c_{1n1} - M_{22}c_{1n2} \\ 0 & 0 & 1 \end{pmatrix}, \tag{14}$$

where

$$M = \begin{pmatrix} c_{1n1} - c_{m11} & c_{111} - c_{mn1} \\ c_{1n2} - c_{m12} & c_{112} - c_{mn2} \end{pmatrix}^{-1}. \tag{15}$$

We summarize the properties of Algorithm AR5I.

Theorem 15. *Algorithm AR5I produces an interpolating planar spline parametrization* $\mathbf{p}$ *with a rank not exceeding 5 if no three of the corner points are collinear and the diagonals are non-parallel. The result is invariant under affine mappings.*

Proof. If the diagonals are non-parallel, then the matrix M in (15) and thus also A in (14) can be constructed and is invertible, thus enabling the transformation to the standard position and back.

After transforming into standard position we need to make sure that the assumptions of Lemma 3 are satisfied, since we cannot apply Algorithm CR2I otherwise. The elements of the denominator determinant in (9) for $k = 1$ are

$$b_{11} = 0, \quad b_{1n} = \frac{\begin{vmatrix} 1 & c_{mn1} & c_{mn2} \\ 1 & c_{111} & c_{112} \\ 1 & c_{m11} & c_{m12} \end{vmatrix}}{|M|}, \quad b_{m1} = \frac{\begin{vmatrix} 1 & c_{111} & c_{112} \\ 1 & c_{1n1} & c_{1n2} \\ 1 & c_{mn1} & c_{mn2} \end{vmatrix}}{|M|}, \quad b_{mn} = 0.$$

This determinant is non-zero as all triplets of corner points are non-collinear. The same argument applies to $k = 2$.

Since we use Algorithm CR2I for $d = 2$, we are certain that the rank of the coefficient tensor in the standard position does not exceed four, see Corollary 4. According to Lemma 1, the back transformation increases it by one at most. Finally we note that affinely equivalent inputs are transformed to the same standard position, thus guaranteeing the affine invariance. □

Note that parallel diagonals would lead to intersecting boundary curves, see Figure 4. It is therefore natural to exclude these situations.

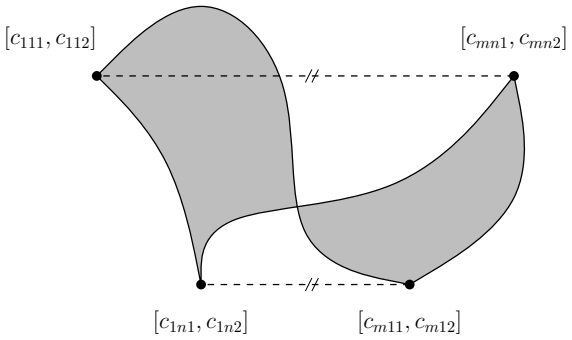


Figure 4: Parallel diagonals of the control polygon lead to intersecting boundary curves.

Example 16. Algorithm AR5I does not satisfy the permanence principle. The result obtained by applying Algorithm AR5I to the data from Example 10 has been restricted to $[0.1, 0.9] \times [0.3, 0.7]$ (Fig. 5, left). We applied Algorithm AR5I to the boundary curves and compared it with the restricted surface. The two meshes in Fig. 5 (right) visualize the results, which are clearly different.

Finding a suitable standard position for higher values of the dimension d is beyond the scope of the present paper.

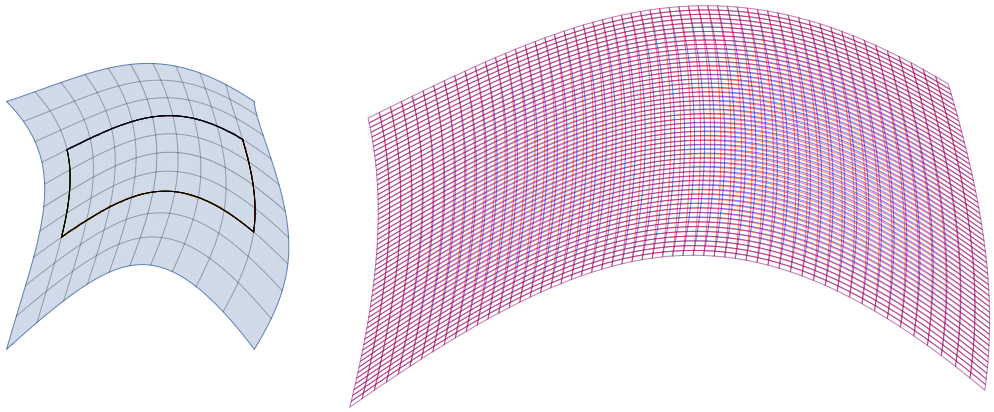


Figure 5: A counterexample to the permanence principle for Algorithm AR5I, see Example 16.

4.2. Bilinear precision

We show that both algorithms possess *bilinear precision*, i.e., if the input data are the boundary curves of a bilinear patch, then we obtain the bilinear patch as a result. We start by proving two technical lemmas.

Lemma 17. *The results of Algorithms AR5I and CR2I do not change if one applies knot insertion or degree elevation to the input curves.*

Proof. According to Corollaries 4 and 6, the spline surfaces with admissible corner points and rank 2 in each component are exactly those generated by Algorithm CR2I. Condition (11) depends on the values of the boundary curves and thus does not change with knot insertion or degree elevation. Neither does Algorithm AR5I, as the boundary curves transform to the same standard position regardless of knot insertion and degree elevation. □

Lemma 18. *All coordinate functions of a bilinear surface* $\mathbf{p}(s,t) = [p_k(s,t)]_{k=1,\dots,d}$ *have rank 2 if and only if the corners are admissible.*

Proof. Denote

$$\mathbf{p}(s,t) = ((1-s),s) \cdot \begin{pmatrix} \mathbf{p}(0,0) & \mathbf{p}(0,1) \\ \mathbf{p}(1,0) & \mathbf{p}(1,1) \end{pmatrix} \begin{pmatrix} (1-t) \\ t \end{pmatrix}.$$

The coefficient tensor of $p_k(s,t)$ is equal to

$$\begin{pmatrix} p_k(0,0) & p_k(0,1) \\ p_k(1,0) & p_k(1,1) \end{pmatrix}.$$

Clearly, requiring each p_k to be of rank 2 is equivalent to assuming the corner points to be admissible. □

Theorem 19. *Algorithms CR2I and AR5I reproduce bilinear patches if the assumptions of Corollary 4 and Theorem 15 are satisfied, respectively.*

Proof. Lemma 18 implies that a given bilinear patch with admissible corner points is a spline surface of rank 2 in each coordinate. If we insert knots or raise the degree so that $m, n \geq 3$, the claim for Algorithm CR2I follows from Corollary 6.

Since affine transformations preserve lines, any bilinear patch satisfying the assumptions of Theorem 15 is mapped into a bilinear patch with admissible corners in reference position of Algorithm AR5I, thus reducing to the previous case. □

Algorithms CR2I and AR5I can be combined with other bases than B-splines. In fact, they only require that the surface can be written in the form (1) and that the values on the boundary of the surface are determined by the control points on the boundary as we specified in (5)–(8). Consequently, our algorithms can be used in combination with, e.g., the tensor-product Lagrange basis.

5. Comparison with existing approaches

We discuss other approaches from the literature and analyze the resulting ranks.

5.1. Biharmonic method

Monterde and Ugail (2004) proposed a method based on their earlier work on biharmonic Bézier surfaces. They formulate conditions on the control points of a surface which ensure that its bilaplacian is equal to zero (cf. Theorem 2 therein). For given boundary curves, the linear system formed by these conditions possesses – under certain assumptions identified by Jüttler et al. (2006) – a unique solution.

However, the method is restricted to Bézier surfaces and the rank of the resulting surface is possibly large.

In our experience (see Section 6), the biharmonic method seems to be suffering from two practical drawbacks. Firstly, the linear system to solve is often quite ill-conditioned, rendering solution in floating point arithmetic less reliable, especially for larger values of m and n. Secondly, the method tends to generate patches that "spill out" of the area between the curves, leading to a useless result.

5.2. Coons interpolation

A Coons patch is constructed as a Boolean sum, i.e., as a sum of two linear interpolants of the boundary curves (one in s-direction and the other in t-direction) from which a bilinear interpolant of the corner points is subtracted. The method can be traced back to Coons (1964). It is effective for a wide class of curves and can be applied to arbitrary dimension d.

Proposition 20. *The rank of a Coons patch does not exceed* $4d$.

Proof. As noticed by Farin and Hansford (1999), a Coons patch can be written as

$$\begin{aligned}(1-s)(\mathbf{p}(0,t)-t\mathbf{p}(0,1))\\ +(1-t)(\mathbf{p}(s,0)-(1-s)\mathbf{p}(0,0))\\ +s(\mathbf{p}(1,t)-(1-t)\mathbf{p}(1,0))\\ +t(\mathbf{p}(s,1)-s\mathbf{p}(1,1)).\end{aligned}$$

Using this decomposition for each coordinate, the rank of the corresponding slice – and hence of the associated slice of the coefficient tensor – is bounded by 4. The bound on the tensor follows from the upper bound in Section 2.3. □

In particular, for $d = 2$ we obtain that the tensor rank of a Coons patch cannot exceed 8. There is no general lower bound on the tensor rank of a Coons patch, but for specific instances we obtained a lower bound of 6, see next section. The question whether the upper bound is tight remains open.

As noted by Farin (1992) and Farin and Hansford (1999) the Coons patch is the same as the discrete Coons patch in the case of Bézier surfaces. Consequently, it can be implemented using the *Coons mask*

$$[c_{ijk}]_k = \begin{matrix}-1/4 & 1/2 & -1/4\\ 1/2 & \bullet & 1/2\\ -1/4 & 1/2 & -1/4\end{matrix}$$

on the control points, which is a discrete version of an Euler-Lagrange equation. Farin and Hansford (1999) discuss a wider class of surfaces generated by certain masks, which they call *permanence patches*. They also provide an interpretation in terms of a variational principle.

5.3. Laplacian smoothing

Laplacian smoothing is another example of a method where the control points are computed using a mask. The *Laplace mask* reads

$$[c_{ijk}]_k = \begin{matrix}0 & 1/4 & 0\\ 1/4 & \bullet & 1/4.\\ 0 & 1/4 & 0\end{matrix}$$

In fact, the masks generating the aforementioned permanence patches are blends between the Coons and Laplace masks, cf. Farin and Hansford (1999) and references therein for a more thorough discussion.

The applicability of the method in the present form is restricted to matching boundary curves. Laplacian smoothing works for arbitrary value of d.

The advantage of the mask-based methods is the simplicity of their implementation. Furthermore, the generated parametrizations have a certain appeal, as Laplacian smoothing often alleviates from acute elements. However, the rank of the resulting surfaces can be quite high.

6. Examples

6.1. Bézier boundary curves

Now, we compare the results obtained for several numerical examples. More precisely, we compare

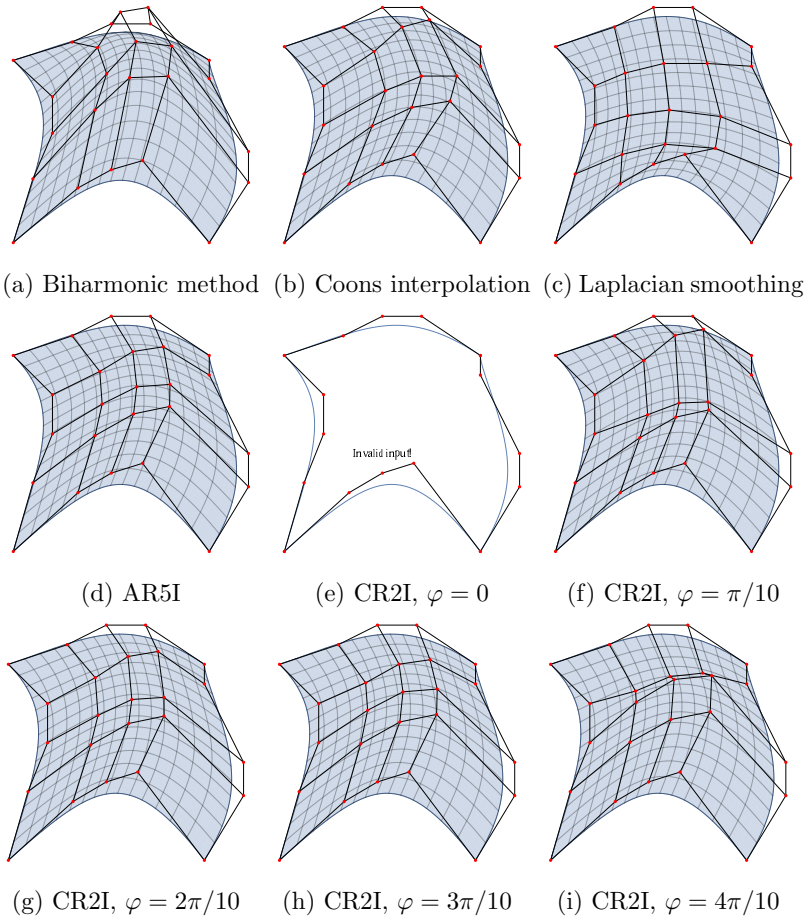
- the biharmonic method,
- Coons interpolation,
- Laplacian smoothing,
- Algorithm AR5I, and
- Algorithm CR2I.

For the latter algorithm we rotate the data around the origin (which is chosen as the centroid of the corner points) by $\varphi = i\pi/10$ with $i = 0, \ldots, 4$. All the examples have been computed symbolically in Mathematica. Table 1 presents several estimates on the tensor rank as well as the maximal rank of g_{ij} and the rank of $\det(\nabla \mathbf{p})$ for all methods and the following six examples.

Example A. We use the same input curves as in Example 10. The differences between the results are most pronounced in the upper part of the domain, see Figure 6. Note that the biharmonic method produces a small overlap (hence a non-regular surface) in the top part of the domain.

Example B. This example shows the sensitivity of the methods with respect to changes of a single control point on the boundary, see Figure 7. Laplacian smoothing leads to smallest changes in the interior of the domain. Algorithms AR5I and CR2I and Coons interpolation produce sensible results, and the ranks are lower than the one obtained by Laplacian smoothing. The biharmonic method is quite sensitive to the input. The results of Algorithm CR2I remain the same for further rotations and are thus not shown.

Example C. The star-like domain in Figure 8 reveals significant differences between the methods. Both the biharmonic method and Coons interpolation perform badly, producing an overlapping parametrization in the center of the domain (see bottom row). Laplacian smoothing produces the most regular elements near the center of the domain. The results of Algorithms AR5I and CR2I are very similar and do not change with further rotations.



(a) Biharmonic method (b) Coons interpolation (c) Laplacian smoothing

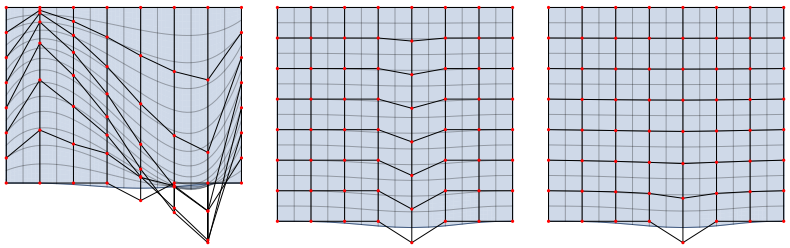
(d) AR5I (e) CR2I, $\varphi = 0$ (f) CR2I, $\varphi = \pi/10$

(g) CR2I, $\varphi = 2\pi/10$ (h) CR2I, $\varphi = 3\pi/10$ (i) CR2I, $\varphi = 4\pi/10$

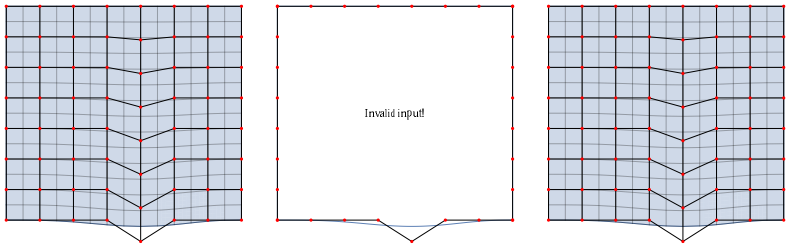
Figure 6: Parametrizations obtained in Example A.

Example D. This example shows that both Laplacian smoothing and the biharmonic method can lead to quite high ranks of the resulting parametrizations, whereas the Coons interpolation and our algorithms give parametrizations with bounded ranks. See Figure 9 for the results. The corners form almost a perfect square. This led to the useless output for Algorithm CR2I with $\phi = 0$, since the input is almost invalid: Figure 9 (e) is in a different scale and the input curves are covered by the dot in the bottom right corner. However, rotating the input leads to sensible results (f)-(i).

Example E. This example has been generated by randomly perturbing the points on a square, see Figure 10. Coons interpolation and Algorithm AR5I show a regular pattern of the control points, which seems to account for the low rank. In contrast,



(a) Biharmonic method (b) Coons interpolation (c) Laplacian smoothing



(d) AR5I (e) CR2I, $\varphi = 0$ (f) CR2I, $\varphi = \pi/10$

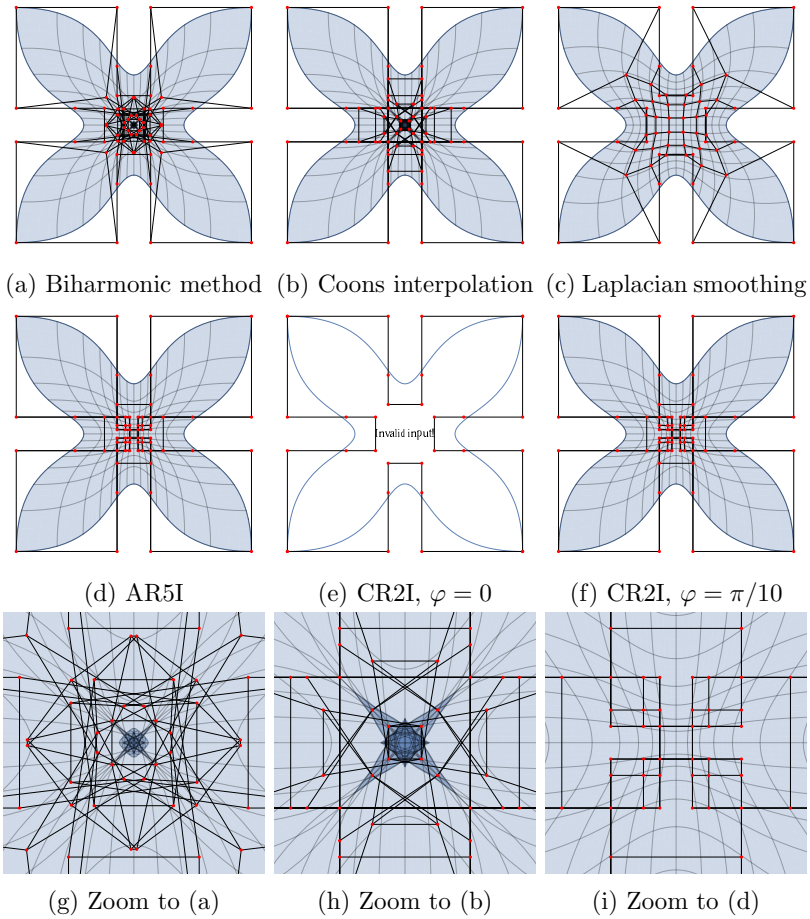
Figure 7: Parametrizations obtained in Example B.

Laplacian smoothing produces a very regular control net at the cost of a rather high rank. The parametrization generated by the biharmonic method is shown in a different scale, since it has "spilled out": The input curves are covered by the red dot in the center of the picture.

Example F. We consider a difficult domain in Figure 11. Laplacian smoothing seems to give the best results around the cavities at the sides, whereas both Coons interpolation and our algorithms do not manage to capture the non-convex part. However, all these three methods lead to singularities (overlaps) near the top and bottom of the domain. The biharmonic method fails. The results of Algorithm CR2I remained the same for further rotations and thus are not shown.

Example G. Figure 12 shows a domain that has been taken from the database of the G+Smo library (Jüttler et al., 2014). Notice that Algorithm AR5I and Algorithm CR2I with $\varphi = \frac{\pi}{10}$ and with $\varphi = \frac{2\pi}{10}$ are the only methods that manage to capture the nonconvex region in the upper part.

In addition, we compare the ranks of the parametrizations produced in these examples in Table 1. For each example, we consider all the parametrizations obtained in the examples. In the case of CR2I, we report in the rows labeled by



(a) Biharmonic method (b) Coons interpolation (c) Laplacian smoothing

(d) AR5I (e) CR2I, $\varphi = 0$ (f) CR2I, $\varphi = \pi/10$

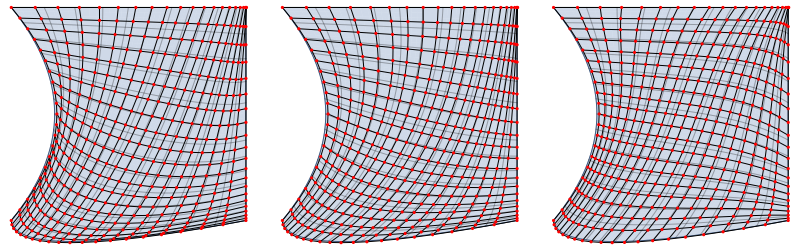
(g) Zoom to (a) (h) Zoom to (b) (i) Zoom to (d)

Figure 8: Parametrizations obtained in Example C. Note that the darker blue is used in areas where the mapping is not one-to-one.

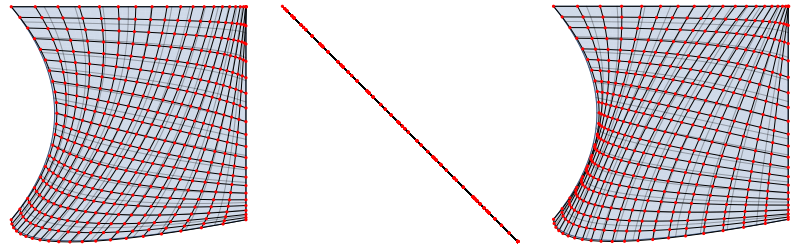
(0) and (φ) the ranks obtained for the reference position and for the remaining four positions, respectively. The latter four positions gave the same ranks in all cases.

As it is quite costly to obtain the exact tensor ranks, we consider ranks of derived quantities (which can be evaluated by computing the ranks of matrices) and numerical estimates. More precisely, we compare

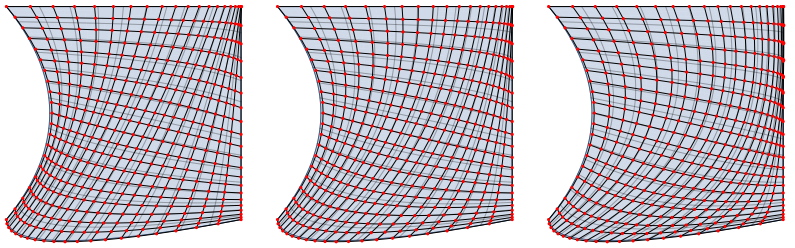
- the rank of the x- (and y-) coordinate function, which is a lower bound on the rank of the parametrization,
- the rank of the matricization obtained by concatenating the x- and y-slice of the coefficient tensor, which is an upper bound on the rank of the parametriza-



(a) Biharmonic method (b) Coons interpolation (c) Laplacian smoothing



(d) AR5I (e) CR2I, $\varphi = 0$ (f) CR2I, $\varphi = \pi/10$



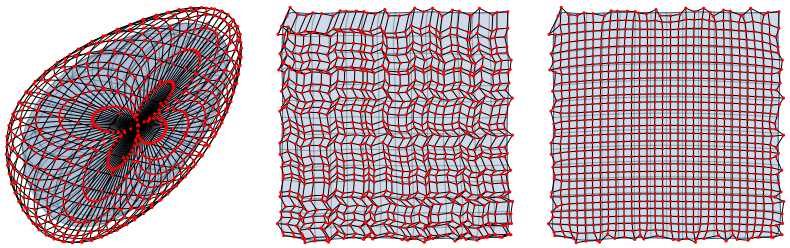
(g) CR2I, $\varphi = 2\pi/10$ (h) CR2I, $\varphi = 3\pi/10$ (i) CR2I, $\varphi = 4\pi/10$

Figure 9: Parametrizations obtained in Example D.

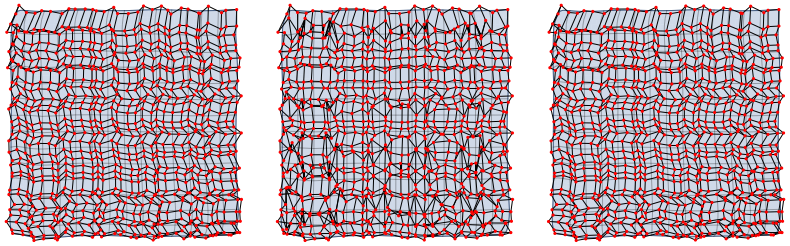
tion,

- a numerically computed estimate for the rank of the parametrization, obtained by calling `rankest` from the Matlab library Tensorlab 3.0 (Vervliet et al., 2016) with `MaxRelErr` $= 10^{-11}$ and `MinRelErr` $= 10^{-12}$,
- the maximum ranks of the spline functions g_{ij}, and
- the rank of the spline function $\det \nabla \mathbf{p}$.

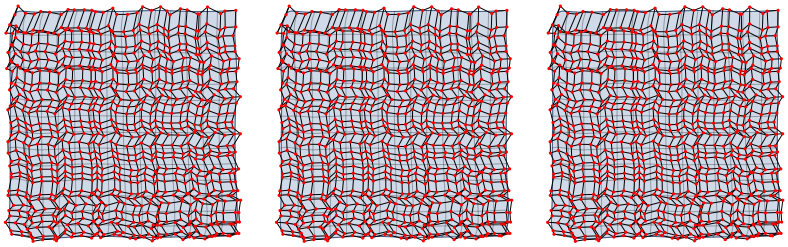
For each example and each considered rank, the lowest rank is highlighted in green, the second lowest one in yellow, and the remaining ones in red. Results in brackets indicate badly shaped parametrizations, and hyphens indicate invalid inputs. The numerical rank computation with `rankest` failed in three instances (marked with



(a) Biharmonic method (b) Coons interpolation (c) Laplacian smoothing



(d) AR5I (e) CR2I, $\varphi = 0$ (f) CR2I, $\varphi = \pi/10$



(g) CR2I, $\varphi = 2\pi/10$ (h) CR2I, $\varphi = 3\pi/10$ (i) CR2I, $\varphi = 4\pi/10$

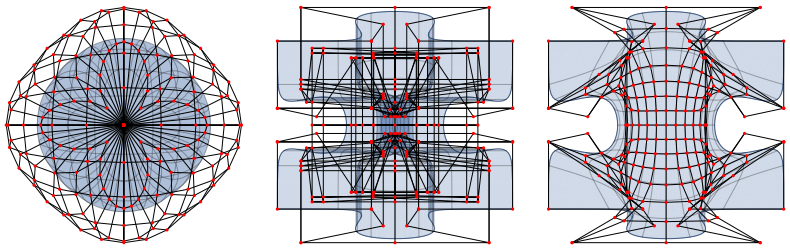
Figure 10: Parametrizations obtained in Example E.

`err`). One may conclude that algorithms CR2I and AR5I give slightly lower ranks than Coons interpolation in almost all cases, while the other two methods give significantly higher ranks.

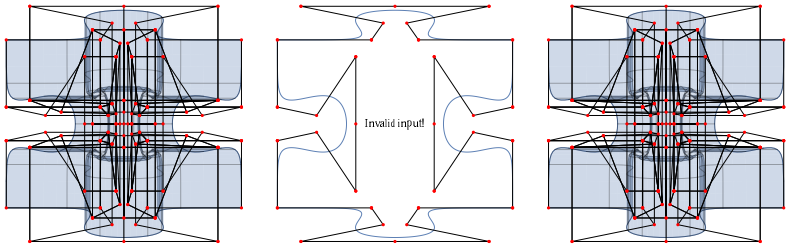
Upper bounds on the ranks of the derived quantities can be obtained from Lemma 2, by combining it with the results concerning the ranks of the parametrizations (Corollary 4, Theorem 15 and Proposition 20). It turns out that these upper bounds are not very tight.

6.2. Spline boundary curves

We conclude with two additional examples, where the boundary curves are spline curves. Therefore the biharmonic method cannot be applied. Also, we do not show control points, as we obtain Coons interpolation not through a mask but



(a) Biharmonic method (b) Coons interpolation (c) Laplacian smoothing



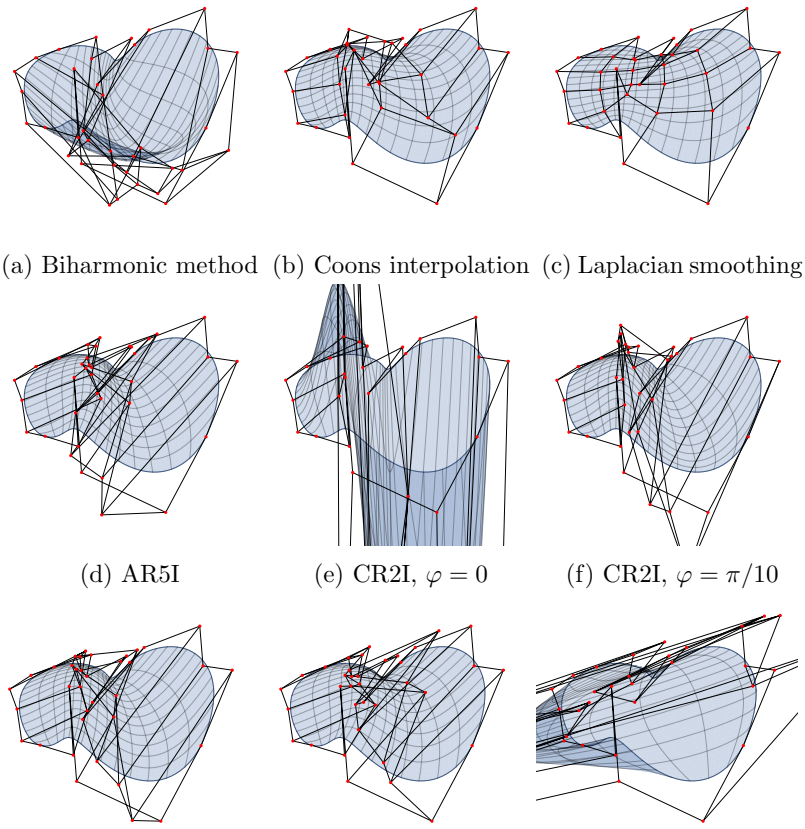
(d) AR5I (e) CR2I, $\varphi = 0$ (f) CR2I, $\varphi = \pi/10$

Figure 11: Parametrizations obtained in Example F.

by a direct application of the formula from the proof of Proposition 20.

Example 21. The boundary curves are approximations of the border of the US state Indiana taken from Giannelli et al. (2016). They are cubic spline curves with 259 control points each. One can see from Figure 13 that none of the simple methods (i.e., methods that work without solving a large linear system) discussed in this paper creates a valid parametrization for the use in isogeometric analysis. Only Laplacian smoothing is able to create a valid parametrization in this case. In this and similar situations, one would need to create a global bijective parametrization and apply our Algorithms to the boundaries of patches obtained by subdivision.

Example 22. The boundary curves are quadratic spline curves with uniform knots and 4 and 33 control points, respectively. The entire geometry is a cross-section of a Japanese tea cup, which is a geometry available in the G+Smo library (Jüttler et al., 2014) except for a slight modification of the short edges at the top of the domain in order to satisfy the assumptions of Algorithm AR5I. Figure 14 shows the results of Coons interpolation, Laplacian smoothing, Algorithm AR5I, direct application of Algorithm CR2I and Algorithm CR2I with $\varphi = \frac{\pi}{6}$.



(a) Biharmonic method (b) Coons interpolation (c) Laplacian smoothing

(d) AR5I (e) CR2I, $\varphi = 0$ (f) CR2I, $\varphi = \pi/10$

(g) CR2I, $\varphi = 2\pi/10$ (h) CR2I, $\varphi = 3\pi/10$ (i) CR2I, $\varphi = 4\pi/10$

Figure 12: Parametrizations obtained in Example G.

7. Conclusions

We introduced two new algorithms that create tensor-product spline surfaces from given boundary curves in $\mathbb{R}^d$. The first one, which fulfills the permanence principle, guarantees that the rank of each coordinate function is equal to 2, but the result is not invariant under affine transformations. The second one, which restores affine invariance by using a reference position and is currently available for dimension $d = 2$ only, guarantees the tensor rank not to exceed 5. We proved that both algorithms possess bilinear precision. We also performed a series of numerical experiments to compare the new algorithms with other interpolation schemes.

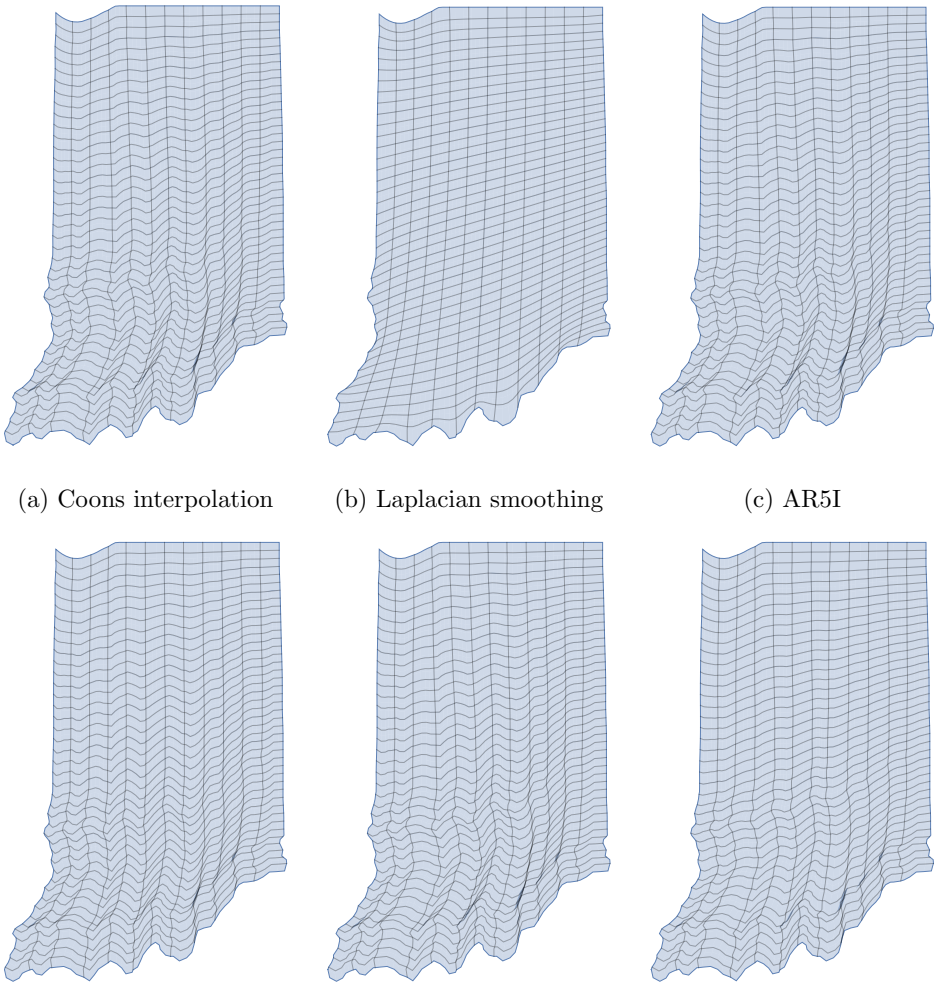
Future work will consider generalizations to volumes and possibly also to the case of C^1 (or even higher order) boundary data. In addition, we plan to investigate the approximation order of the new surface interpolation schemes.

Table 1: Ranks of parametrizations and derived quantities for examples A-F.

Quantity	Method	Example A	B	C	D	E	F	G
x-slice	biharmonic	5	1	4	21	(30)	(7)	6
	Coons	4	1	2	4	4	2	4
	Laplacian	5	1	4	21	30	7	6
	AR5I	4	1	1	5	5	1	5
	CR2I(0)	-	-	-	(2)	2	-	2
	CR2I(φ)	2	2	2	2	2	2	2
y-slice	biharmonic	5	4	4	21	(30)	(7)	6
	Coons	4	2	2	3	4	2	4
	Laplacian	5	7	4	21	30	7	6
	AR5I	4	2	1	5	5	1	5
	CR2I(0)	-	-	-	(2)	2	-	2
	CR2I(φ)	2	2	2	2	2	2	2
matricization	biharmonic	5	4	8	21	(30)	(14)	6
	Coons	5	2	4	5	6	4	6
	Laplacian	5	8	8	21	30	14	6
	AR5I	4	2	2	5	5	2	5
	CR2I(0)	-	-	-	(4)	4	-	4
	CR2I(φ)	4	2	2	4	4	2	4
`rankest`	biharmonic	6	5	8	41	(6)	(13)	6
	Coons	`err`	3	4	6	`err`	4	`err`
	Laplacian	5	8	8	18	58	14	6
	AR5I	4	3	2	4	5	2	5
	CR2I(0)	-	-	-	(4)	4	-	4
	CR2I(φ)	4	3	2	4	4	2	4
maximum rank of g_{ij}	biharmonic	8	6	7	13	(9)	(10)	10
	Coons	7	2	4	5	14	4	8
	Laplacian	8	13	7	24	51	14	10
	AR5I	8	2	2	5	16	2	10
	CR2I(0)	-	-	-	(3)	8	-	8
	CR2I(φ)	8	2	2	5	8	2	8
$\det(\nabla \mathbf{p})$	biharmonic	7	3	7	12	(9)	(8)	10
	Coons	7	1	5	4	14	5	8
	Laplacian	8	7	7	17	45	14	10
	AR5I	8	1	2	4	8	2	8
	CR2I(0)	-	-	-	(6)	8	-	8
	CR2I(φ)	8	1	2	4	8	2	8

Acknowledgments

The authors have been supported by the Austrian Science Fund (FWF, NFN S117 "Geometry + Simulation").



(a) Coons interpolation (b) Laplacian smoothing (c) AR5I

(d) CR2I, $\varphi = 0$ (e) CR2I, $\varphi = \frac{\pi}{6}$ (f) CR2I, $\varphi = \frac{\pi}{3}$

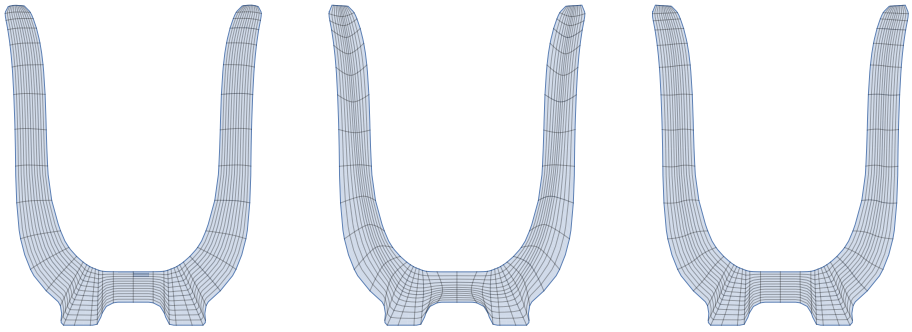
Figure 13: Indiana geometry.

References

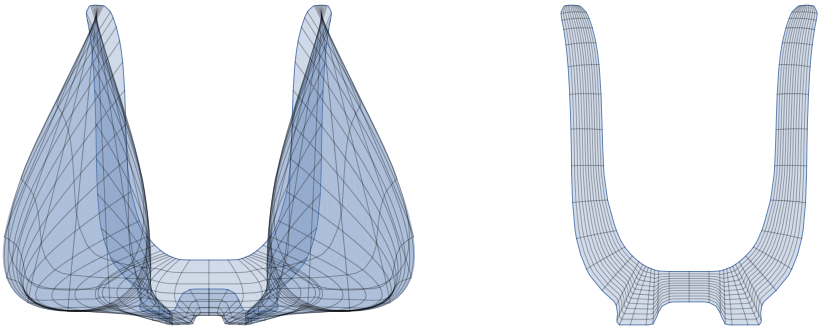
Centella, P., Monterde, J., Moreno, E. and Oset, R. (2009). Two C1-methods to generate Bézier surfaces from the boundary, *Comput. Aided Geom. Design* **26**(2): 152–173.

Coons, S. A. (1964). Surfaces for computer aided design, *Technical Report, MIT* .

Falini, A., Špeh, J. and Jüttler, B. (2015). Planar domain parameterization with THB-splines, *Comput. Aided Geom. Design* **35**: 95–108.



(a) Coons interpolation (b) Laplacian smoothing (c) AR5I



(d) CR2I, $\varphi = 0$ (e) CR2I, $\varphi = \frac{\pi}{6}$

Figure 14: Japanese tea cup geometry.

Farin, G. (1992). Commutativity of Coons and tensor product operators, *Rocky Mtn. J. Math.* **22**(2): 541–547.

Farin, G. E. (2001). *Curves and Surfaces for CAGD: A Practical Guide*, Elsevier. 5th ed.

Farin, G. and Hansford, D. (1999). Discrete Coons patches, *Comput. Aided Geom. Design* **16**(7): 691–700.

Giannelli, C., Jüttler, B., Kleiss, S. K., Mantzaflaris, A., Simeon, B. and Špeh, J. (2016). THB-splines: An effective mathematical technology for adaptive refinement in geometric design and isogeometric analysis, *Comput. Methods Appl. Mech. Engrg.* **299**: 337–365.

Gravesen, J., Evgrafov, A., Nguyen, D.-M. and Nørtoft, P. (2014). Planar parametrization in isogeometric analysis, *in* M. Floater, T. Lyche, M.-L. Mazure, K. Mørken and L. L. Schumaker (eds), *Mathematical Methods for Curves and Surfaces*, Lecture Notes in Computer Science, pp. 189–212.

Håstad, J. (1990). Tensor rank is NP-complete, *J. of Algorithms* **11**(4): 644–654.

Ja'Ja', J. (1979). Optimal evaluation of pairs of bilinear forms, *SIAM Journal on Computing* **8**(3): 443–462.

Jüttler, B., Langer, U., Mantzaflaris, A., Moore, S. E. and Zulehner, W. (2014). Geometry + simulation modules: Implementing isogeometric analysis, *Proc. Appl. Math. Mech.* **14**(1): 961–962.

Jüttler, B., Oberneder, M. and Sinwel, A. (2006). On the existence of biharmonic tensor-product Bézier surface patches, *Comput. Aided Geom. Design* **23**(7): 612–615.

Kolda, T. G. and Bader, B. W. (2009). Tensor decompositions and applications, *SIAM Review* **51**(3): 455–500.

Landsberg, J. M. (2012). *Tensors: geometry and applications*, Vol. 128 of *Graduate Studies in Mathematics*, American Mathematical Society Providence, RI, USA.

Mantzaflaris, A., Jüttler, B., Khoromskij, B. and Langer, U. (2014). Matrix generation in isogeometric analysis by low rank tensor approximation, *in* J.-D. Boissonnat, A. Cohen, O. Gibaru, C. Gout, T. Lyche, M.-L. Mazure and L. L. Schumaker (eds), *Curves and Surfaces*, Lecture Notes in Computer Science, pp. 321–340.

Monterde, J. and Ugail, H. (2004). On harmonic and biharmonic Bézier surfaces, *Comput. Aided Geom. Design* **21**(7): 697–715.

Monterde, J. and Ugail, H. (2006). A general 4th-order PDE method to generate Bézier surfaces from the boundary, *Comput. Aided Geom. Design* **23**(2): 208–225.

Pan, M., Tong, W. and Chen, F. (2016). Compact implicit surface reconstruction via low-rank tensor approximation, *Compput. Aided Des.* **78**: 158–167.

Vervliet, N., Debals, O., Sorber, L., Van Barel, M. and De Lathauwer, L. (2016). Tensorlab 3.0. Available online.
URL: *http://www.tensorlab.net*