Piecewise polynomial monotonic interpolation of 2D gridded data

Léo Allemand-Giorgis, Georges-Pierre Bonneau, Stefanie Hahmann, Fabien Vivodtzev

Piecewise polynomial monotonic interpolation of 2D gridded data

Léo Allemand-Giorgis, Georges-Pierre Bonneau, Stefanie Hahmann and Fabien Vivodtzev

Abstract A method for interpolating monotone increasing 2D scalar data with a monotone piecewise cubic C^1-continuous surface is presented. Monotonicity is a sufficient condition for a function to be free of critical points inside its domain. The standard axial monotonicity for tensor-product surfaces is however too restrictive. We therefore introduce a more relaxed monotonicity constraint. We derive sufficient conditions on the partial derivatives of the interpolating function to ensure its monotonicity. We then develop two algorithms to effectively construct a monotone C^1 surface composed of cubic triangular Bézier surfaces interpolating a monotone gridded data set. Our method enables to interpolate given topological data such as minima, maxima and saddle points at the corners of a rectangular domain without adding spurious extrema inside the function domain. Numerical examples are given to illustrate the performance of the algorithm.

Key words: Monotone surfaces, interpolation, visualization

Léo Allemand-Giorgis
Inria - Laboratoire Jean Kuntzmann, University of Grenoble
e-mail: leo.allemand-giorgis@inria.fr

Georges-Pierre Bonneau
Inria - Laboratoire Jean Kuntzmann, University of Grenoble
e-mail: Georges-Pierre.Bonneau@inria.fr

Stefanie Hahmann
Inria - Laboratoire Jean Kuntzmann, University of Grenoble
e-mail: Stefanie.Hahmann@inria.fr

Fabien Vivodtzev
CEA: French Atomic Energy Commission and Alternative Energies
e-mail: Fabien.Vivodtzev@cea.fr

1 Introduction

Preserving meaningful features in scalar fields when simplifying the data set is a requirement in Scientific Visualization for efficiently visualizing and understanding very large data. One class of meaningful features of a scalar function is locations and values of local extrema. Large isolated extrema are indeed salient features and have usually an important meaning. So they have to be preserved in a visualization, whereas nearby local extrema with similar values can be neglected in many application domains. It is even better to remove these spurious extrema in order to enhance the visibility of significant features and thus to improve the understanding of the data set. It is equally important to avoid adding spurious extraneous features when visualizing a data set. In particular if a data set samples a function without local extrema, then the visualization of this discrete data should be free of local extrema as well.

Morse theory [17] is an example of a concept dealing with critical points and their importance. The Morse-Smale (MS) complex, which is based on Morse theory, segments the domain into a set of regions inside which the function is monotonous. A combinatorial simplification of the MS complex has been introduced in [6]. The MS complex is simplified by removing adjacent pairs of critical points in the complex while preserving most significant critical points. Then the original data is however not coherent anymore with the simplified complex, because monotonicity got lost inside the MS cells. The new data should be monotonic inside a region. So, techniques which compute monotonic data knowing some values of the function on a grid are needed.

This paper presents a novel approach for computing monotone scalar functions interpolating gridded 2D data sets using smooth piecewise polynomial representations. We built the interpolant in such a way that no local extrema exist in the interior of the function domain if the input data fulfills a simple monotonicity criterion. In contrast to prior related works we do not require the input data to be axial monotonic, meaning that the data does not have to be strictly monotonous along all grid rows and columns, instead we base on a less restrictive monotonicity criterion. In this paper we make the following contributions:

- A monotonicity constraint is used which is more general than the standard axial monotonicity for tensor-product surfaces.
- In concordance with the monotonicity constraint we introduce a modified Sibson split interpolant.
- We derive sufficient conditions on the partial derivatives to ensure monotonicity of the interpolating function.
- We then develop two algorithms to effectively construct a monotone C^1 surface composed of cubic triangular Bézier surfaces.

The main contribution of this paper is a new piecewise polynomial monotonic interpolant. It has been inspired by earlier work in the field of Constrained Shape Design within Computer Aided Geometric Design (CAGD). In this field the objective is to interpolate or approximate a set of points by analytically defined curves or

surfaces with a given constraint such as positivity, convexity or monotonicity. Our new interpolant is based on relaxed monotonicity constraints that still ensure that no critical points exist in the interior of the function domain.

The remainder of the present paper is structured as follows. In Section 2 we review related works dealing with shape preserving interpolation. Section 3 presents our contributions on piecewise polynomial monotone interpolation. We first introduce our new interpolant. It is based on a relaxed monotonicity constraint. We give sufficient conditons ensuring that no critical point exists in the interior of the domain with our new interpolant. We then develop two algorithms to effectively compute monotone interpolants. Finally we show results in Section 4 and conclude the paper in Section 5 with a dicussion on future directions of research.

2 Related Works

Shape preserving interpolation is a well studied problem in scientific literature. It consists of computing a curve or surface which interpolates given scalar data while preserving the shape of the underlying data.

The problem we are dealing with in the present paper is closely related to shape preserving interpolation. Indeed, we aim to construct a function free of critical points interpolating a maximum and a minimum function value at opposite vertices of a rectangular domain. Since a monotone surface is sufficient for this purpose, we derive an algorithm which is able to construct a surface that preserves monotonicity along a diagonal direction.

Convexity [4, 3, 13] and monotonicity are typical shape properties to be preserved. Concerning monotonicity preserving surface fitting most research focussed on monotone bivariate interpolation. In [1, 2, 12] sufficient (and sometimes also necessary) conditions were derived for the monotonicity of piecewise polynomial patches interpolating data given at the grid points of a rectangular mesh. These conditions were transformed into a system of linear inequalities which in turn formed the basis for an algorithm. Since the interpolating functions are determined by a function value and first order derivatives at the grid points, the algorithms compute feasable solutions of the derivatives. All these methods provide surfaces preserving axial monotonicity. Even though our method is similar to [12], we base it on a relaxed monotonicity preservation in diagonal direction only. This is sufficient for our goal to interpolate local extrema without any other critical point inside the function domain.

In [10] it was shown that tensor product Berstein and B-spline bases preserve monotonicity in all directions. In [11] three kinds of monotonicity preservation of systems of bivariate functions on a triangle are studied. We also use the fact that Bernstein polynomials on triangles preserve monotonicity. However, we do not only derive sufficient conditions for monotonicity, in addition we provide an effective algorithm to compute a monotone surface. Our approach splits the rectangular domain

into four cubic triangle patches and computes the Bézier control points with a modified Sibson split in order to get a globally C^1-continuous surface.

Let us finally mention that monotonicity preservation has also been investigated for scattered data approximation [19], for subdivision curves [9, 14], convolution of B-splines [18], for rational surfaces [5] and for non-polynomial functions [15].

3 Monotonic Polynomial Interpolation

In this section we propose a novel solution to the following problem.

Problem: Given a 2D rectangular grid of scalar values sampled from a monotone function, compute a smooth interpolating function which is monotone as well.

This is a typical problem encountered in *Shape Preserving* interpolation methods [16], where the general goal is to compute an interpolating surface that mimics the shape of the input data.

The method we present in this paper is inspired by the algorithms given in [1, 2, 12], where C^1 monotone increasing spline functions interpolating gridded data are constructed by iteratively adjusting initially estimated gradient values at each grid point. It is a kind of Hermite interpolation, where the gradient values are adjusted to ensure monotonicity. This idea can easily be illustrated on 1D cubic Hermite interpolation of monotone data, where the choice of the derivative (slopes of the tangents) at these data points decide whether the interpolating function is monotonic or not, see Figure 1.

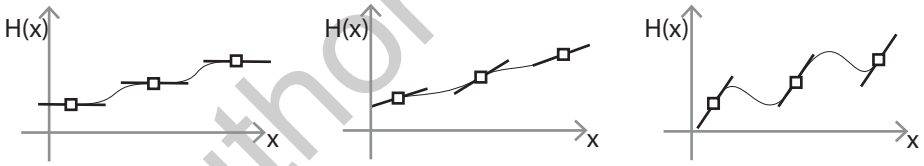

Fig. 1 Cubic Hermite interpolation of monotone increasing data. Different derivatives prescribed at the data points lead to different curves. Setting derivatives equal zero produces a monotone increasing curve (left), but critical points are generated. Monotonicity of the function gets lost when the derivative values are too big (right).

We use piecewise polynomial functions defined on a triangular subdivision of the gridded domain, where each rectangle is subdivided into four triangles by drawing the main diagonals, as shown in Figure 3. Each polynomial piece of function $f : T \to \mathbb{R}$ is a cubic triangular Bézier surface (Figure 2)

$$f(\tau) = \sum_{\substack{i+j+k=3 \\ i,j,k\geq 0}} b_{ijk} B^3_{ijk}(\tau), \qquad \tau \in T$$

defined on a domain triangle $T \subset \mathbb{R}^2$ given by three non-colinear points p_0, p_1, p_2. $\tau = (\tau_0, \tau_1, \tau_2)$ is the triplet of barycentric coordinates of a point in T. The 10 coefficients $b_{ijk} \in \mathbb{R}$ are called the *Bézier ordinates* of f corresponding to the parameter values $(i/3, j/3, k/3)$. $B^3_{ijk}(\tau) = \frac{3!}{i!j!k!}\tau_0^i\tau_1^j\tau_2^k$ are the Berstein polynomials of degree 3. All fundamentals on triangular Bézier surfaces can be found in [7].

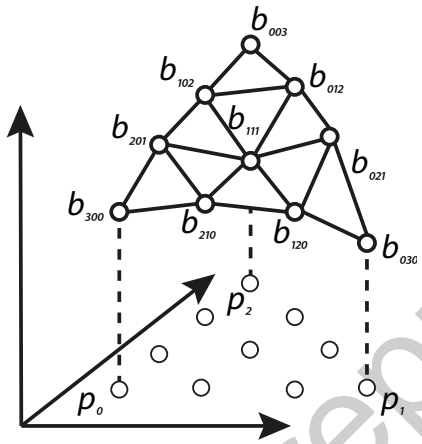

Fig. 2 Triangular Bézier surface patch of degre 3.

The polynomial pieces of our surface are computed by interpolating Hermite data (i.e. function values and partial derivatives) given at the grid points using a modified Sibson split interpolant, as explained in the next section. An advantage of using Hermite data is that C^1-continuity can be ensured more easily. The problem we solve is therefore to find upper bounds on the gradients, which guarantee that the functions interpolating these Hermite data is monotone, as illustrated in Figure 1 for the 1D case.

However, instead of solving for functions with monotone increasing values along the grid lines, i.e. along the x- and y-axis as it was done in previous works, we construct our interpolating function to be monotone only in diagonal $(x+y)$-direction. Additionally, we require that the function has to be strictly monotone increasing, i.e. be free of critical points inside its domain. Note that the latter property is not guaranteed by the standard axis aligned monotonicity definition as used in [1, 2, 12].

We begin in Section 3.1 by first describing the standard Sibson split method, which subdivides each rectangle into four cubic bivariate polynomials joining with C^1 continuity. Then the modified Sibson split interpolant is presented. Section 3.2 introduces our relaxed monotonicity constraints and explains the difference to the standard monotonicity. Then our new monotonicity preserving interpolant is intro-

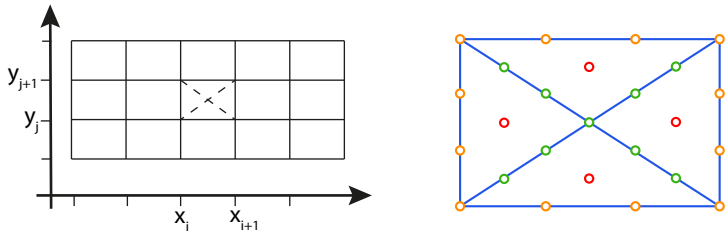

Fig. 3 Left: Rectangular function domain D with vertices (x_i, y_j). Right: Sibson split and Bezier coefficients of one rectangular patch.

duced in Section 3.3, and a proof of monotonicity is given. And finally, we derive two algorithms ensuring monotonicity and strict monotonicity respectively for a function interpolating monotone input data in Sections 3.4 and 3.5.

3.1 Modified Sibson split Interpolant

Cubic Sibson split patches (SSP) interpolate positional and partial derivative values given at the vertices of a rectangular gridded domain. Each patch is composed of four non-parametric cubic triangular Bézier patches with 25 Bézier ordinates as illustrated in Figure 4. The resulting surface is globally C^1-continuous since two adjacent SSP which share the same boundary curve along the common edge, interpolate the same partial cross-derivatives given at each end-point of the edge and have the same linear cross-boundary derivatives along the edge.

Following [8] the orange control vertices are used to interpolate positional and partial derivative values at the four corners. The green control vertices are constrained by the C^1 continuity between patches. Each green vertex is the mid-point on the straight line between the two vertices indicated by the arrows. The green middle vertex is the midpoint of the plane of its four surrounding vertices. These last conditions are sufficient to ensure C^1 continuity between the four patches, for an arbitrary choice of the red vertices [7]. The red vertices are chosen so that the *normal cross-boundary derivatives* (derivative orthogonal to the boundary) are linear. This leads to the following formulas for these four vertices:

$$\begin{cases} c_{17} = (2c_{13} + 2c_{14} + c_5 + c_6 - c_1 - c_2)/4 \\ c_{18} = (2c_{14} + 2c_{15} + c_7 + c_6 - c_1 - c_2)/4 \\ c_{19} = (2c_{15} + 2c_{16} + c_9 + c_6 - c_1 - c_2)/4 \\ c_{20} = (2c_{16} + 2c_{13} + c_{11} + c_6 - c_1 - c_2)/4. \end{cases} \quad (1)$$

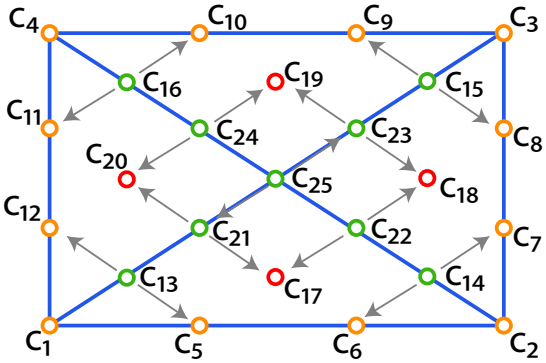

Fig. 4 Cubic (modified) Sibson interpolant. Input data: positional and gradient values located on a rectangular grid. Each grid cell is subdivided into four triangles and yields a C^1 piecewise cubic interpolant. Orange points are computed from given data, red points from linear cross-boundary derivatives and green points from C^1-continuity conditions.

The detailed formulas for all other coefficients are given in the Appendix.

Modified Sibson Interpolant

Instead of having linear normal cross-boundary derivatives, we require the *directional cross-boundary derivatives* $\frac{\partial f}{\partial(x+y)}(x,y)$ to be linear. This results in the following formulas for the Bézier ordinates marked in red in Figure 4 and which replace the formulas (1) :

$$\begin{cases} c_{17} = (-c_1 + 2c_5 - c_6 + c_{13} + c_{14})/2 \\ c_{18} = (-c_3 + 2c_8 - c_7 + c_{14} + c_{15})/2 \\ c_{19} = (-c_3 + 2c_9 - c_{10} + c_{15} + c_{16})/2 \\ c_{20} = (-c_1 + 2c_{12} - c_{11} + c_{13} + c_{16})/2. \end{cases} \tag{2}$$

All other coefficients remain the same.

Lemma 1. *The modified Sibson Interpolant* (2) *is* C^1 *continuous for given position and gradient values at the corners of a rectangular domain.*

Proof. Formulas (2) correspond to linear interpolation of $\frac{\partial f}{\partial(x+y)}(x,y)$ along the outer boundaries of each SSP. Since the SSPs share the same partial derivatives at their corners, the directional derivatives in the direction $x+y$ must be continuous across the domain D. Furthermore directional derivatives along the boundary curves of the SSPs are trivially continuous since the SSPs are C^0 continuous. It follows that the interpolant is C^1 across the domain. □

3.2 Relaxed Monotonicity

Monotonicity constraints : Let D be a rectangular domain in $\mathbb{R}^2$, subdivided into rectangles $D_{ij} = [x_i, x_{i+1}] \times [y_j, y_{j+1}]$ with $1 \leq i < n_x$ and $1 \leq j < n_y$ and $h_i^x = x_{i+1} - x_i$, $h_j^y = y_{j+1} - y_j$.
The data set $\{(x_i, y_j, z_{ij})\}_{i,j=1}^{n_x,n_y}$ is called *diagonal monotone increasing* if

$$z_{ij} < z_{i+1,j+1} \tag{3}$$

for all $1 \leq i < n_x$ and $1 \leq j < n_y$.

The problem we seek to solve is to compute a C^1-continuous function $f : D \to \mathbb{R}$ such that f interpolates a diagonal monotone increasing data set, i.e.

$$f(x_i, y_j) = z_{ij}$$

for all $1 \leq i \leq n_x$ and $1 \leq j \leq n_y$, and f is *monotone increasing in* $(x+y)$*-direction*, i.e.

$$\frac{\partial f}{\partial (x+y)}(x,y) \geq 0 \tag{4}$$

for all (x,y) in D.

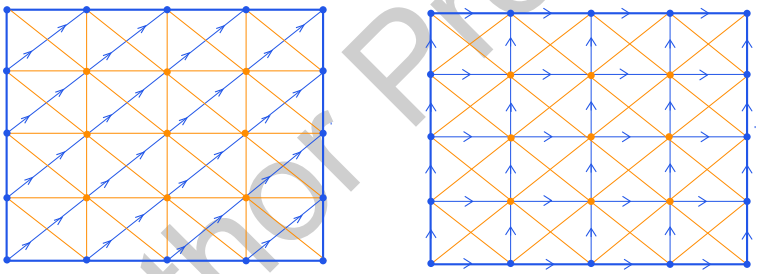

Fig. 5 Left: Diagonal monotone increasing data. The function values z_{ij} at the grid points are increasing along the grid diagonals (blue arrows). Right: Axis-aligned monotone increasing data. The functions values z_{ij} at the grid points are increasing along the x- and the y-axis (blue arrows).

Remark: A function is called *strictly monotone increasing* if (4) is replaced by

$$\frac{\partial f}{\partial (x+y)}(x,y) > 0. \tag{5}$$

In contrast to the usually used axis-aligned monotonicity as in [12] our *diagonal monotonicity* (3) of the input data does not require that the given function values increase along rows and columns, but only along the grid diagonals, see Figure 5. Note that this is a more general framework since if the data is increasing along the

rows and columns then it is also increasing along diagonals. It is thus possible to deal with horizontal or vertical grid lines that are not monotone increasing.

3.3 Sufficient monotonicity conditions

We now introduce a new monotonicity preserving interpolation scheme for gridded data, that only assumes the input values to increase along diagonals, as illustrated in Figure 5-left.

Let us assume without loss of generality that all D_{ij} are squares of the same size $h := h_i^x = h_j^y$. The next theorem gives sufficient conditions on the partial derivatives so that the modified SSP interpolant is (strictly) monotonic.

Theorem 1. *The modified SSP given by* (2) *which interpolates positional values* z_{kl} *and partial derivatives* z_{kl}^x, z_{kl}^y *on* D_{ij} $(k = i, i+1,\ l = j, j+1)$ *satisfies* (5) *providing the following conditions hold:*

$$z_{ij}^x + z_{ij}^y > 0\,, \tag{6}$$

$$z_{i+1j}^x + z_{i+1j}^y > 0\,, \tag{7}$$

$$z_{ij+1}^x + z_{ij+1}^y > 0\,, \tag{8}$$

$$z_{i+1j+1}^x + z_{i+1j+1}^y > 0, \tag{9}$$

$$\frac{12}{h}(z_{i+1j+1} - z_{ij}) > 5z_{ij}^x + z_{ij}^y + 2z_{i+1j}^x + 2z_{i+1j}^y + z_{i+1j+1}^x + 5z_{i+1j+1}^y\,, \tag{10}$$

$$\frac{12}{h}(z_{i+1j+1} - z_{ij}) > z_{ij}^x + 5z_{ij}^y + 2z_{ij+1}^x + 2z_{ij+1}^y + 5z_{i+1j+1}^x + z_{i+1j+1}^y\,. \tag{11}$$

Proof. In essence the proof of theorem 1 follows from three arguments. The first is that conditions (6) to (9) imply the monotonicity constraint (5) at the four corners of the SSP. The second argument comes from the modified Sibson-Split interpolation formulas (2), which imply that (5) is fulfilled along the four outer boundary edges of the SSP. Eventually, conditions (10) and (11) enable to propagate the monotonicity constraint (5) inside the SSP.

From [7] we know that condition (5) is satisfied by the modified SSP provided that for all couples of coefficients $(c_j \to c_k)$, as shown in Figure 6, the relation $c_j < c_k$ holds. This leads to 18 sufficient conditions on the 25 coefficients c_i.

Hypotheses (6) to (9) imply $c_{13} - c_1 > 0,\ c_{14} - c_6 = c_7 - c_{14} > 0,\ c_3 - c_{15} > 0,\ c_{16} - c_{11} = c_{10} - c_{16} > 0$ (see the Appendix for the values of the control-points). Adding (6) and (7) leads to $2(c_{17} - c_5) = (c_{13} - c_1) + (c_{14} - c_6) > 0$. Analogously we have $c_8 - c_{18} > 0,\ c_9 - c_{19} > 0$, and $c_{20} - c_{12} > 0$. Furthermore $2(c_{21} - c_{13}) = (c_{20} - c_{12}) + (c_{17} - c_5) > 0$, and analogously we have $c_{15} - c_{23} > 0$. $c_{22} - c_{17}$ can be computed from the interpolation conditions using (2) and the control-point formulas given in the Appendix. This leads to

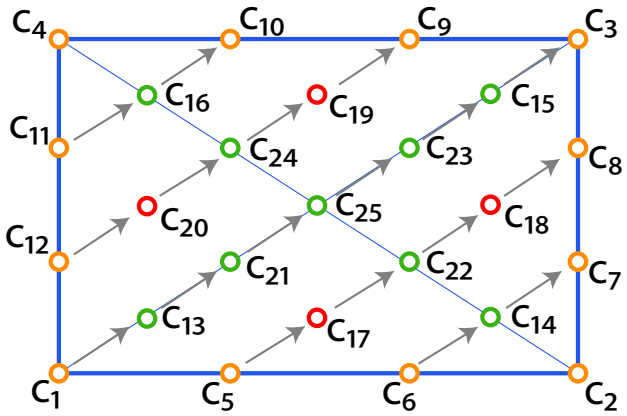

Fig. 6 If for every arrow the coefficient at its basis is strictly smaller than the coefficients at its tip then the four cubic triangular Bézier patches satisfy the diagonal monotonicity constraint (5), i.e. they are strictly increasing in the diagonal direction $x+y$.

$$c_{22}-c_{17}=\frac{1}{2}\left[(z_{i+1j+1}-z_{ij})-\frac{h}{12}(5z^x_{ij}+z^y_{ij}+2z^x_{i+1j}+2z^y_{i+1j}+z^x_{i+1j+1}+5z^y_{i+1j+1})\right]$$

Therefore condition (10) implies $c_{22}-c_{17}>0$. Analogously condition (11) implies $c_{24}-c_{20}>0$. From the Appendix we have $2*(c_{25}-c_{21})=(c_{22}-c_{17})+(c_{24}-c_{20})$ and therefore $c_{25}-c_{21}>0$. The remaining three conditions follow from the Appendix: $c_{18}-c_{22}=c_{22}-c_{17}>0$, $c_{23}-c_{25}=c_{25}-c_{21}>0$, $c_{19}-c_{24}=c_{24}-c_{20}>0$. □

Remark: (**Theorem 1'**)
Replacing the strict inequality signs by $\geq$ in (6) to (11) yields an analogous theorem for the weaker monotonicity constraint (4). The proof is analogous as well.

3.4 Algorithm 1

Assume now we are given as input gridded values z_{ij} which satisfy (3). Our goal is to compute partial derivatives z^x_{ij} and z^y_{ij} such that the sufficient conditions given in Theorem 1 are satisfied. Let us call them *admissible* partial derivatives. Using these admissible partial derivatives together with the input values z_{ij} for the modified SSP interpolant makes the resulting function being monotonic.

We present two algorithms for computing admissible partial derivatives, one for the weaker Theorem 1' and one for Theorem 1. Also the strategies followed by the algorithms are different. While the first algorithm modifies some given estimations of the partial derivatives, the second algorithm computes admissible partial derivatives only from the input function values.

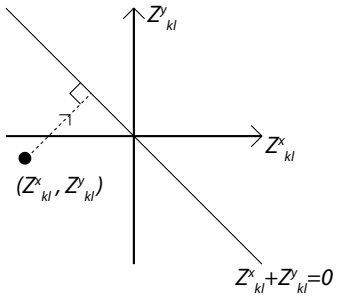

Fig. 7 Projection of the partial derivatives (z^x_{kl}, z^y_{kl}) on $z^x_{kl} + z^y_{kl} = 0$ when their sum is negative.

The first part of Algorithm 1 corresponds to the conditions (6) to (9) of Theorem 1. If the sum of the partial derivatives $z^x_{kl} + z^y_{kl}$ is negative in any corner of the SSP, we apply an orthogonal projection as illustrated in Fig.7. This procedure results in a gradient that is closest to the original gradient. The modified partial derivatives are thus given by $((z^x_{kl} - z^y_{kl})/2, (z^y_{kl} - z^x_{kl})/2))$.

The second part of Algorithm 1 decreases the partial derivative values so that conditions (10) and (11) of the Theorem 1 are satisfied. Absolute values are required since the partial derivatives as input may have a negative value.

3.5 Algorithm 2

The previous algorithm needs estimated gradient values as input. We now propose a second algorithm which directly computes admissable gradient values for the strict monotonicity of Theorem 1.

This second algorithm ensures that the right hand sides in conditions (10) and (11) are always smaller than $\lambda \frac{12}{h} \left(z_{i+1\,j+1} - z_{ij}\right)$, where $\lambda \in]0,1[$ is a user-specified constant. It also ensures that all partial derivatives are positive, so that conditions (6) to (9) are satisfied as well.

Proof. (Algorithm 2)

Because $z_{i+1\,j+1} > z_{ij}$, all K_{ij} are positive. Therefore all $KMin_{ij}$ are positive, and all partial derivatives z^x_{ij} and z^y_{ij} computed by Algorithm 2 are also positive. It follows that conditions (6) to (9) are fulfilled. It remains to prove conditions (10) and (11).

Let S be the right part of the condition (10):

$$\begin{aligned} S &= 5z^x_{ij} + z^y_{ij} + 2z^x_{i+1\,j} + 2z^y_{i+1\,j} + z^x_{i+1\,j+1} + 5z^y_{i+1\,j+1}, \\ &= 6\frac{\lambda}{2} KMin_{ij} + 4\frac{\lambda}{2} KMin_{i+1\,j} + 6\frac{\lambda}{2} KMin_{i+1\,j+1}. \end{aligned}$$

Algorithm 1 Modify the input partial derivatives for one SSP in order to satisfy the sufficient monotonicity conditions given by Theorem 1.

Require: z_{kl} with $z_{ij} < z_{i+1j+1}$, and partial derivatives z^x_{kl}, z^y_{kl} $(k = i, i+1, l = j, j+1)$
  {First part of the algorithm}
  for $k \in \{i, i+1\}$ **do**
    for $l \in \{j, j+1\}$ **do**
      {Verification of the four conditions (6)-(9)}
      if $z^x_{kl} + z^y_{kl} < 0$ **then**
        $old_x \leftarrow z^x_{kl}$
        $old_y \leftarrow z^y_{kl}$
        {Orthogonal projection of the point (z^x_{kl}, z^y_{kl}) onto $z^x_{kl} + z^y_{kl} = 0$}
        $z^x_{kl} \leftarrow \frac{old_x - old_y}{2}$
        $z^y_{kl} \leftarrow \frac{old_y - old_x}{2}$
      end if
    end for
  end for
  {Second part of the algorithm}
  $\Delta z \leftarrow \frac{12}{h}(z_{i+1j+1} - z_{ij})$
  $S_2 \leftarrow 5|z^x_{ij}| + |z^y_{ij}| + 2|z^x_{i+1j}| + 2|z^y_{i+1j}| + |z^x_{i+1j+1}| + 5|z^y_{i+1j+1}|$
  $S_3 \leftarrow |z^x_{ij}| + 5|z^y_{ij}| + 2|z^x_{ij+1}| + 2|z^y_{ij+1}| + 5|z^x_{i+1j+1}| + |z^y_{i+1j+1}|$
  $S_{max} \leftarrow Maximum(S_2, S_3)$
  {Verification of the last two conditions (10),(11)}
  if $\Delta z < S_{max}$ **then**
    $c \leftarrow \frac{\Delta z}{S_{max}}$
    $z^x_{kl} \leftarrow c\, z^x_{kl}$ for $k = i, i+1, l = j, j+1$
    $z^y_{kl} \leftarrow c\, z^y_{kl}$ for $k = i, i+1, l = j, j+1$
  end if

Algorithm (2) computes the coefficients $KMin_{ij}$ such that:

$$KMin_{ij},\ KMin_{i+1j},\ KMin_{i+1j+1} \leq \frac{3}{2h}(z_{i+1j+1} - z_{ij}).$$

And therefore:

$$S \leq \lambda \frac{12}{h}(z_{i+1j+1} - z_{ij}),$$

which means that condition (10) is verified.
Condition (11) can be proven analogously. □

The parameter λ can be used to control the shape of the interpolant. A value of λ close to 1 tends to increase the partial derivatives at the corners of the SSP, and to decrease the partial derivatives in the interior of the SSP. On the other hand, a value of λ close to 0 implies almost 0 partial derivatives at the corners of the SSP. In our examples we have used $\lambda = \frac{2}{3}$.

Algorithm 2 Compute partial derivatives for all SSPs in order to fulfill the sufficient monotonicity conditions given by Theorem 1.

Require: z_{ij} for $1 \leq i \leq n_x$ and $1 \leq j \leq n_y$, such that $z_{ij} < z_{i+1\,j+1}$
Require: a constant λ with $0 < \lambda < 1$
  for $i = 1 \cdots n_x - 1$ **do**
    for $j = 1 \cdots n_y - 1$ **do**
      $K_{ij} \leftarrow \frac{3}{2h}\left(z_{i+1\,j+1} - z_{ij}\right)$
    end for
  end for
  $KMin_{11} \leftarrow K_{11}$
  $KMin_{n_x n_y} \leftarrow K_{n_x-1,n_y-1}$
  $KMin_{n_x,1} \leftarrow K_{n_x-1,1}$
  $KMin_{1,n_y} \leftarrow K_{1,n_y-1}$
  {traverse the interior of domain}
  for $i = 1 \cdots n_x - 2$ **do**
    for $j = 1 \cdots n_y - 2$ **do**
      $KMin_{i+1,j+1} \leftarrow min\left(K_{ij}, K_{i+1\,j}, K_{i\,j+1}, K_{i+1\,j+1}\right)$
    end for
  end for
  {traverse left and right domain boundaries}
  for $j = 1 \cdots n_y - 2$ **do**
    $KMin_{1,j+1} \leftarrow min\left(K_{1\,j}, K_{1,j+1}\right)$
    $KMin_{n_x,j+1} \leftarrow min\left(K_{n_x-1\,j}, K_{n_x-1,j+1}\right)$
  end for
  {traverse bottom and top domain boundaries}
  for $i = 1 \cdots n_x - 2$ **do**
    $KMin_{i+1,1} \leftarrow min\left(K_{i1}, K_{i+1,1}\right)$
    $KMin_{i+1,n_y} \leftarrow min\left(K_{i,n_y-1}, K_{i+1,n_y-1}\right)$
  end for
  {initialize partial derivatives from *KMin* array}
  for $i = 1 \cdots n_x$ **do**
    for $j = 1 \cdots n_y$ **do**
      $z^x_{ij} \leftarrow \lambda \frac{1}{2} KMin_{ij}$
      $z^y_{ij} \leftarrow z^x_{ij}$
    end for
  end for

4 Results

The first and simple example shown in Figure 8 corresponds to a 4×2 grid of scalar values that satisfy the relaxed monotonicity constraints (3). The interpolated values are increasing along the rows, but not along all columns. They are diagonal monotone increasing but not axis-aligned monotone. This simple data set can thus not be handled by previous works on monotonicity preserving interpolation. In contrast, our relaxed monotonicity constraints enable to built an interpolant that is free of critical points as it can be seen from the isolines shown in the left side of Figure 8. On the right side, the same interpolant is shown together with the isoparametric lines. Since the interpolant is composed of triangular Bézier patches, there are 3 isoparametric directions.

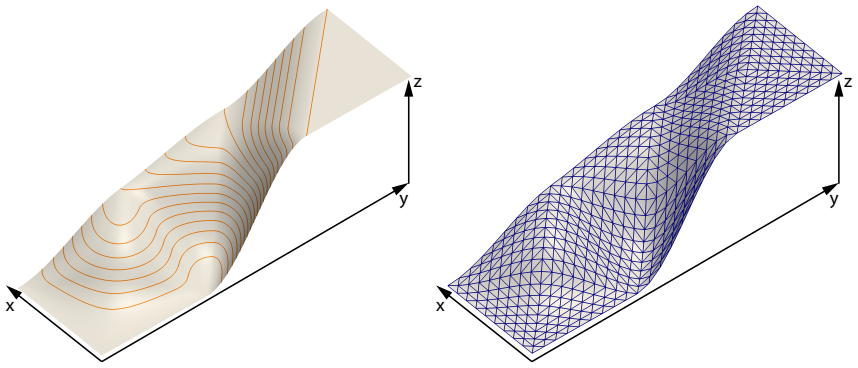

Fig. 8 A 4×2 grid of scalar values is interpolated by our C^1 cubic interpolant. This simple example can not be handled by prior works on monotonicity preserving interpolation since some of the rows have increasing and other have decreasing values. Nevertheless our interpolant does not exhibit critical points as it can be seen from the isolines in the left. The isoparametric lines of the 12 cubic Bézier patches are shown on the right.

The second example in Figure 9 illustrates the ability to use estimates of gradient values thanks to Algorithm 1 presented in section 3.4. In this example a grid of 5×5 sampled values in the domain $[-0.4, +0.4]^2$ for the function $(x+y)^3$ is interpolated. We give as input to Algorithm 1 the exact partial derivatives sampled from the function. We show our modified Sibson-Split interpolant on the left side of Figure 9 without correction of the partial derivatives, and on the right side with the corrected partial derivatives computed by Algorithm 1. As it can be seen from the isolines at the bottom of Figure 9, the interpolant with the exact partial derivatives exhibits critical points, which are removed when the corrected partial derivatives computed by Algorithm 1 are used for the interpolant.

The third example shows the interpolation of a grid of size 10×10. The values at the grid vertices are computed randomly but satisfy the diagonal monotonicity constraint (3). For this example Algorithm 2, described in Section 3.5, is applied to compute admissible values for the partial derivatives at the grid vertices. This data set can not be handled by previous works since neither the rows nor the columns are monotone increasing. Nevertheless, our new method is able to produce a monotone C^1 interpolant free of critical points as it can be seen from the isolines in Figure 10. The inset on the top right shows a closeup view with the isoparametric lines of the Bézier patches.

The last example in Figure 11 illustrates the ability of *piecewise* monotone interpolants to interpolate a grid of function values, where local minima, maxima and saddles are prescribed. In contrast to the previous settings, where a globally monotone function perfoms shape preserving interpolation, we now apply the monotonicity property only locally. We compute a surface which is piecewise monotone (inside each grid cell) without generating any extraneous critical points except at the grid vertices where critical points are prescribed. The red vertices are local maxima,

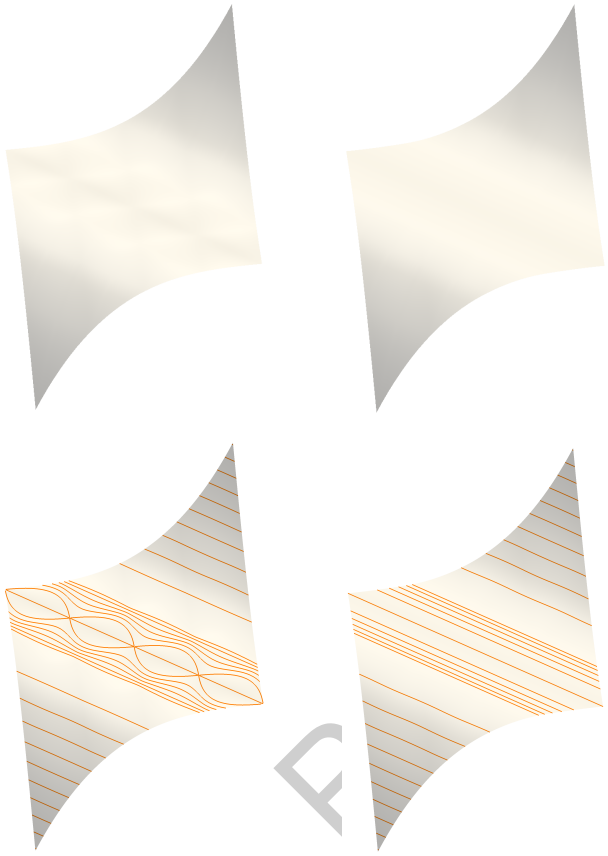

Fig. 9 Interpolation of a grid sampled from an analytical function. The left part shows the result when the exact partial derivatives of the function are used for the interpolant. The isolines on the bottom left clearly indicate critical points. In the right part these critical points are removed by modifying the partial derivatives with our Algorithm 1 (Section 3.4).

the blue vertices are local minima and the green vertices are saddles. At all these prescribed critical points the partial derivatives are fixed to be 0. There are only two grid vertices in yellow, which are regular vertices and where the choice of the partial derivatives is determinant for the monotonicity of the neighbouring patches. Indeed, this example also allows to illustrate the influence of the gradient values on the shape of a function computed by a Sibson interpolant. Figure 12 shows two results with different gradient values at the yellow vertices, reproducing thus the phenomena described in Figure 1-right for the 2D case. In fact, on can observe that the choice of gradient values too large in size produces extra unwanted critical points, see Figure 11-right, whereas Figure 11-left shows the resulting interpolant with properly chosen partial derivatives the yellow vertices. In both surfaces the local extrema are exactly interpolated, and the shape of the interpolant is smooth around the yellow

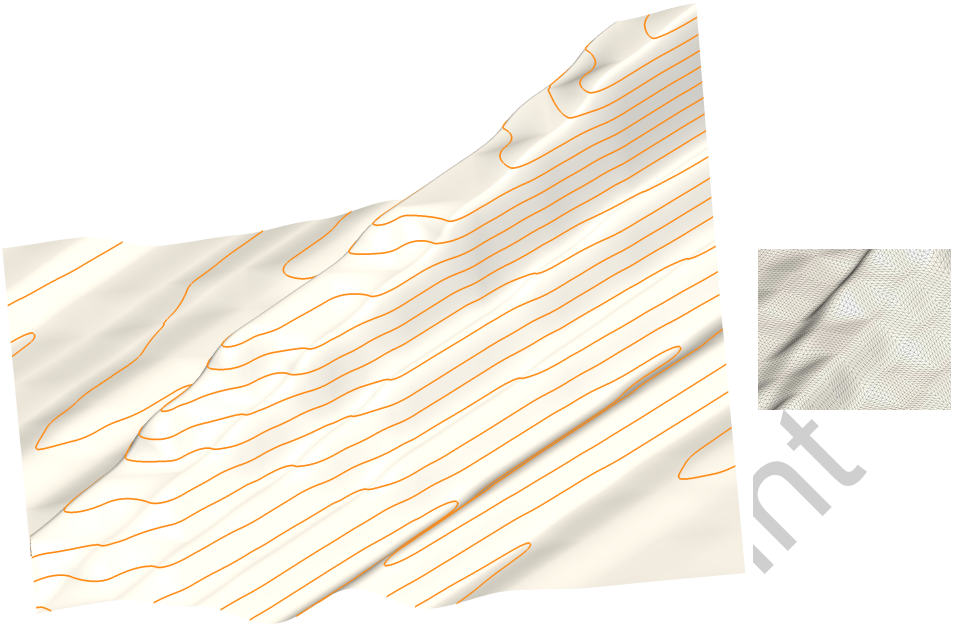

Fig. 10 Interpolation of a grid of 10×10 function values. The grid values are randomly chosen such that our relaxed monotonicity constraint (3) is fulfilled. Since neither the rows nor the columns are increasing, this grid can not be handled by previous works on constraint monotonic interpolation. Our new method can produce an interpolant free of critical points in the interior of the definition domain, as shown by the isolines. The partial derivatives are computed from Algorithm 2, described in section 3.5. The inset on the top right shows a closeup view with the isoparametric lines

regular vertices. All the patches are individually monotone increasing and join with c^1-continuity.

Finally, let us provide some statistics: All examples have been computed in less then $1ms$. Even though we only show small size examples to appreciate geometric and topological properties of the resulting functions, the method runs in real time also for very large data sets. We generated data sets with 10^6 grid cells and computed valid gradient values in $3ms$. Moreover, the computation of Bézier surfaces can be performed in parallel. Indeed, once the derivatives known at the corners of the grid cell ths modified Sibson interpolant can be computed independly from its neighbours.

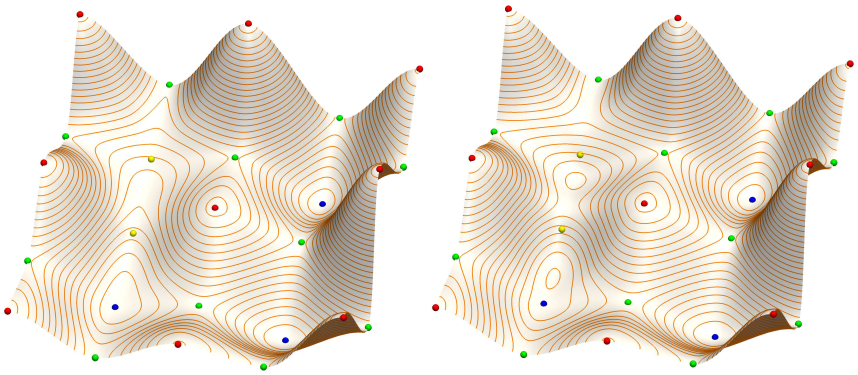

Fig. 11 Local maxima (red), minima (blue), saddles (green) and regular (yellow) vertices are interpolated by a C^1 piecewise cubic interpolant. Left: no unwanted local extrema exist in the interior of the cubic patches. Right: partial derivatives too large in size are chosen for the yelllow regular vertices implying that additional unwanted local extrema appear inside the cubic polynomial patches.

5 Conclusion

In this paper we propose a new method to interpolate a 2D grid of scalar values by a C^1 piecewise cubic function with no critical points and no local extrema in the interior of the domain. In comparison with prior related works, we do not require the values to increase along all rows and columns of the grid. Instead, we introduce a relaxed monotonicity constraint in which the values are required to increase monotonously only along diagonals of the grid cells. We also introduce a modified Sibson-split interpolant which is coherent with this relaxed monotonicity constraint. We give sufficient conditions on the partial derivatives at the grid vertices such that the Sibson-split interpolant is free of local extrema. And we propose two algorithms to actually compute admissible partial derivatives satisfying these sufficient conditions. The first algorithm takes as input estimated values of the partial derivatives and modifies them in order to ensure monotonicity. The second algorithm computes partial derivatives without requiring any initial guess. As shown in Section 4, such a C^1 piecewise cubic monotonic interpolant can also be used to interpolate a grid of prescribed local minima, maxima and saddles.

Our method is a step towards reconstructing function from MS complexes, even though it can't be applied directly in its present form. Indeed, the current work is limited to regular grids. Therefore, we are currently investigating the generalization of our results to monotone interpolation of function values defined on a triangular mesh instead of a grid. This would enable us to extend the example shown in Figure 11 to an arbitrary setting of local extrema. Furthermore, we plan to apply our method to the reconstruction of monotonic functions within MS cells. Another direction

of future research is to extend the present approach to C^2 continuity using quintic triangular Bézier patches.

Appendix — Sibson split interpolant

Let D be a rectangular domain in $\mathbb{R}^2$, regularly subdivided into rectangles $D_{ij} = [x_i, x_{i+1}] \times [y_j, y_{j+1}]$, $1 \le i < n_x, 1 \le j < n_y$ and the following ordinates z_{ij} and gradients z^x_{ij}, z^y_{ij} given at the grid points. Let $h^x = x_{i+1} - x_i, h^y = y_{j+1} - y_j$. Each rectangle is splitted into 4 sub-triangles by drawing both diagonals.

The Sibson split (cf. [7]) is a cubic C^1-continuous function $f : D \to \mathbb{R}$ interpolating the input data with

$$f(x_i, y_i) = z_{ij}, \qquad f_x(x_i, y_i) = z^x_{ij}, \qquad f_y(x_i, y_i) = z^y_{ij},$$

where each patch defined on D_{ij} is composed of four cubic polynomials with in total 25 Bézier coefficients (see Figure 4), computed uniquely as follows:

$$\begin{aligned}
c_1 &= z_{ij} & c_2 &= z_{i+1,j} & c_3 &= z_{i+1,j+1} \\
c_4 &= z_{i,j+1} & c_5 &= c_1 + \frac{h^x}{3} z^x_{ij} & c_6 &= c_2 - \frac{h^x}{3} z^x_{i+1,j} \\
c_9 &= c_3 - \frac{h^x}{3} z^x_{i+1,j+1} & c_{10} &= c_4 + \frac{h^x}{3} z^x_{i,j+1} & c_{12} &= c_1 + \frac{h^y}{3} z^y_{ij} \\
c_7 &= c_2 + \frac{h^y}{3} z^y_{i+1,j} & c_8 &= c_3 - \frac{h^y}{3} z^y_{i+1,j+1} & c_{11} &= c_4 - \frac{h^y}{3} z^x_{i,j+1} \\
c_{13} &= \frac{1}{2}(c_5 + c_{12}) & c_{14} &= \frac{1}{2}(c_6 + c_7) & c_{15} &= \frac{1}{2}(c_8 + c_9) \\
c_{16} &= \frac{1}{2}(c_{11} + c_{10}) & c_{21} &= \frac{1}{2}(c_{20} + c_{17}) & c_{22} &= \frac{1}{2}(c_{17} + c_{18}) \\
c_{23} &= \frac{1}{2}(c_{18} + c_{19}) & c_{24} &= \frac{1}{2}(c_{19} + c_{20}) & c_{25} &= \frac{1}{2}(c_{21} + c_{23}) = \frac{1}{2}(c_{22} + c_{24}).
\end{aligned}$$

References

1. Beatson, R., Ziegler, Z.: Monotonicity preserving surface interpolation. SIAM J. Numer. Anal. **22**(2), 401–411 (1985)
2. Carlson, R., Fritsch, F.: An algorithm for monotone piecewise bicubic interpolation. SIAM Journal on Numerical Analysis **26**(1), 230–238 (1989)
3. Carnicer, J., Floater, M., Peña, J.: Linear convexity conditions for rectangular and triangular bernstein-bézier surfaces. Computer Aided Geometric Design **15**(1), 27 – 38 (1997). DOI http://dx.doi.org/10.1016/S0167-8396(97)81783-9
4. Cavaretta A.S., J., Sharma, A.: Variation diminishing properties and convexity for the tensor product bernstein operator. In: B. Yadav, D. Singh (eds.) Functional Analysis and Opera-

tor Theory, *Lecture Notes in Mathematics*, vol. 1511, pp. 18–32. Springer Berlin Heidelberg (1992). DOI 10.1007/BFb0093794
5. Delgado, J., Peòa, J.M.: Are rational bézier surfaces monotonicity preserving? Comput. Aided Geom. Des. **24**(5), 303–306 (2007). DOI 10.1016/j.cagd.2007.03.006
6. Edelsbrunner, H., Harer, J., Zomorodian, A.: Hierarchical morse—smale complexes for piecewise linear 2-manifolds. Discrete & Computational Geometry **30**(1), 87–107 (2003). DOI 10.1007/s00454-003-2926-5. URL http://dx.doi.org/10.1007/s00454-003-2926-5
7. Farin, G.E.: Triangular bernstein-bézier patches. Computer Aided Geometric Design **3**(2), 83–127 (1986)
8. Farin, G.E.: Curves and Surfaces for Computer-Aided Geometric Design: A Practical Code, 4th edn. Academic Press, Inc., Orlando, FL, USA (1996)
9. Floater, M., Beccari, C., Cashman, T., Romani, L.: A smoothness criterion for monotonicity-preserving subdivision. Advances in Computational Mathematics **39**(1), 193–204 (2013). DOI 10.1007/s10444-012-9275-y
10. Floater, M., Peña, J.: Tensor-product monotonicity preservation. Advances in Computational Mathematics **9**(3-4), 353–362 (1998). DOI 10.1023/A:1018906027191
11. Floater, M.S., Peña, J.M.: Monotonicity preservation on triangles. Math. Comput. **69**(232), 1505–1519 (2000)
12. Han, L., Schumaker, L.L.: Fitting monotone surfaces to scattered data using c1 piecewise cubics. SIAM J. Numer. Anal. **34**(2), 569–585 (1997). DOI 10.1137/S0036142994268582. URL http://dx.doi.org/10.1137/S0036142994268582
13. Jüttler, B.: Surface fitting using convex tensor-product splines. Journal of Computational and Applied Mathematics **84**(1), 23 – 44 (1997). DOI http://dx.doi.org/10.1016/S0377-0427(97)00100-3
14. Kuijt, F., van Damme, R.: Monotonicity preserving interpolatory subdivision schemes. Journal of Computational and Applied Mathematics **101**(1–2), 203 – 229 (1999). DOI http://dx.doi.org/10.1016/S0377-0427(98)00220-9
15. Mainar, E., Peña, J.M.: Monotonicity preserving representations of non-polynomial surfaces. J. Comput. Appl. Math. **233**(9), 2161–2169 (2010). DOI 10.1016/j.cam.2009.09.045
16. McAllister, D.F., Passow, E., Roulier, J.A.: Algorithms for computing shape preserving spline interpolations to data. Mathematics of Computation **31**(139), 717–725 (1977)
17. Smale, S.: On gradient dynamical systems. Annals of Mathematics **74**(1), pp. 199–206 (1961). URL http://www.jstor.org/stable/1970311
18. Strøm, K.: On convolutions of b-splines. Journal of Computational and Applied Mathematics **55**(1), 1 – 29 (1994). DOI http://dx.doi.org/10.1016/0377-0427(94)90182-1
19. Willemans, K., Dierckx, P.: Smoothing scattered data with a monotone powell-sabin spline surface. Numerical Algorithms **12**(1), 215–231 (1996). DOI 10.1007/BF02141749