Chapter 1

Scalar and Parametric Spline Curves and Surfaces

Horacio Florez and Belsay Borges

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.74929

Abstract

A common engineering task consists of interpolating a set of discrete points that arise from measurements and experiments. Another traditional requirement implies creating a curve that mimics a given array of points, namely, a polyline. Any of these problems require building an analytical representation of the given discrete set of points. If the geometrical shape represented by the input polyline is complicated, then we may expect that a global interpolant or polynomial will be of a high degree, to honor all imposed constraints, which makes its use prohibited. Indeed, a global interpolant often experiences inflection points and sudden changes in curvature. To avoid these drawbacks, we often seek solving the interpolation/approximation problem using piecewise polynomial functions called "splines."

Keywords: cubic splines, tension splines, Bèzier curves, B-splines, NURBS

1. Introduction

In this chapter, we tackle passing a curve/surface through a given set of data points. We first focus in the case in which the curve to be constructed can be described as $S(x) = (x, f(x))$. We refer this data set as scalar data. We describe herein cubic and tension splines, which are powerful interpolants suitable to tackle large data sets. We then introduce a parametric case for a vector-valued curve $\underline{S}(\xi) = (x(\xi), y(\xi))^T$ and hence, it is able to represent arbitrary topologies. We explain how to construct piecewise continuous cubic Bèzier curves called "B-splines." We cover the interpolation and approximation problems with B-splines, this latter denoted as well as "inverse" design. We extend our treatment to tensor product surfaces that are referred as piecewise bicubic B-splines. Applications encompass translational and interpolation surfaces. We briefly introduce nonuniform rational B-spline curves and surfaces (*NURBS*).

We present applications such as approximating conic sections. We finalize the chapter introducing Duchon splines that are radial basis functions to interpolate scattered data sets in two or three dimensions.

2. Scalar splines

We cover herein the scalar case in which a spline function $S(x) = (x, f(x))$ fits a given set of sorted point pairs. We introduce cubic splines and their specialized version that offers a "tension" parameter that allows attracting the interpolant toward the polyline that connects the input points, i.e., linear spline. We refer to this latter as tension splines. The last section presents a couple of numerical examples.

2.1. Cubic splines

A spline is a piecewise continuous function consisting of several polynomials, each specified in a subinterval, bound themselves by certain continuity conditions. Let $x_0, \ldots, x_n$ be $(n+1)$ sorted points such that $x_0 < x_1 < x_2 < \ldots < x_n$ whose corresponding values are denoted by $y_0, \ldots, y_n$. A spline of k degree with knots $x_0, \ldots, x_n$ is a function $S : \mathbb{R} \to \mathbb{R}$ such that:

1. S_i is a polynomial of degree $\leq k$ that is continuous up to kth derivative over $[x_i, x_{i+1}]$.

2. Two adjacent splines need to have C^0 continuity at the junction points:

$$S(x) = \begin{cases} S_0(x); x \in [x_0, x_i) \\ S_i(x); x \in [x_i, x_{i+1}) \\ \vdots \\ S_{n-1}(x); x \in [x_{n-1}, x_n) \end{cases} \tag{1}$$

We thus enforce $C^m, m = 0, \ldots, (k-1)$ continuity conditions at the $(n-1)$ junction points which yields to $(4n-2)$ equations to determine $4n$ unknown spline coefficients. We omit details herein but refer the reader to [1, 2]. We end up with a tridiagonal system for the unknown curvature values κ_i, at the junction points:

$$h_{i-1} \cdot \kappa_{i-1} + 2 \cdot (h_i + h_{i-1}) \cdot \kappa_i + h_i \cdot \kappa_{i+1} = \frac{6}{h_i}\left(y_{i+1} - y_i\right) - \frac{6}{h_{i-1}}\left(y_i - y_{i-1}\right), \tag{2}$$

where $i = 1, .., n-1$ and $h_i = x_{i+1} - x_i$. The last equation provides a system of $(n-1)$ conditions for $\kappa_0, \ldots, \kappa_n$. Since both κ_0 and κ_n are arbitrary, a logical choice is choosing $\kappa_0 = \kappa_n \equiv 0$, which we refer as "natural spline." For the latter, we can write in matrix form:

$$\begin{bmatrix} u_1 & h_1 & & & & & \\ h_1 & u_2 & h_2 & & & & \\ & h_2 & u_3 & h_3 & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & \cdot & \cdot & \\ & & & h_{n-3} & u_{n-2} & h_{n-2} & \\ & & & & h_{n-2} & u_{n-1} & \end{bmatrix} \cdot \begin{bmatrix} \kappa_1 \\ \kappa_2 \\ \kappa_3 \\ \cdot \\ \cdot \\ \cdot \\ \kappa_{n-2} \\ \kappa_{n-1} \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \cdot \\ \cdot \\ \cdot \\ v_{n-2} \\ v_{n-1} \end{bmatrix}, \tag{3}$$

where

$$u_i = 2 \cdot (h_i + h_{i-1}) \;\; ; \;\; b_i = \frac{6}{h_i} \cdot \left(y_{i+1} - y_i\right) \;\; ; \;\; v_i = b_i - b_{i-1}. \tag{4}$$

Once we determine the curvature values κ_i, by solving Eq. (3), we define the spline function as

$$S_i(x) = y_i + A_i \cdot (x - x_i)^3 + B_i \cdot (x - x_i)^2 + C_i \cdot (x - x_i); \;\; i = 0, \ldots, n-1, \tag{5}$$

where the coefficients are given by

$$A_i = \frac{1}{6 \cdot h_i} \cdot (\kappa_{i+1} - \kappa_i) \;\; ; \;\; B_i = \frac{\kappa_i}{2} \;\; ; \;\; C_i = -\frac{h_i}{6} \cdot \kappa_{i+1} - \frac{h_i}{3} \cdot \kappa_i + \frac{1}{h_i} \cdot \left(y_{i+1} - y_i\right). \tag{6}$$

2.2. Tension splines

In some problems of adjusting discrete data, it is useful to have a parameter called "tension, τ." When τ has a small value, the resulting curve approaches a cubic spline. When τ tends to $+\infty$, the resulting curve approaches a linear spline. For the same sequence of sorted point pairs mentioned above, the tension spline satisfies.

1. $T \in C^2[x_0, x_n]$ and $T(x_i) = y_i \;\; ; \;\; i = 0, \ldots, n$.

2. On every interval $[x_i, x_{i+1}] : T^{(IV)} - \tau^2 \cdot T^{(II)} = 0$.

That is, $T : \mathbb{R} \to \mathbb{R}$ has continuity C^4 globally, interpolates to the data, and satisfies certain ordinary differential equation in each subinterval. It is clear that the prescription $\tau = 0$ leads to cubic polynomials when solving the equation. To determine T, we proceed similarly to the case of cubic splines, i.e., $\kappa_i \equiv T''(x_i)$:

$$T^{(IV)} - \tau \cdot T^{(II)} = 0 \;\; ; \;\; T(x_i) = y_i \;\; ; \;\; T(x_{i+1}) = y_{i+1}. \tag{7}$$

The solution is given by [2]:

$$T(x) = \frac{[\kappa_i \cdot \sinh(\tau \cdot \widehat{x}) + \kappa_{i+1} \cdot \sinh(\tau \cdot \tilde{x})]}{\tau^2 \cdot \sinh(\tau \cdot h_i)} + \left(y_i - \frac{\kappa_i}{\tau^2}\right) \cdot \frac{\widehat{x}}{h_i} + \left(y_{i+1} - \frac{\kappa_{i+1}}{\tau^2}\right) \cdot \frac{\tilde{x}}{h_i}, \tag{8}$$

where $\widehat{x} = (x_{i+1} - x)$ and $\tilde{x} = (x - x_i)$, and we compute the curvatures by solving the system:

$$\alpha_{i-1}\kappa_{i-1} + (\beta_{i-1} + \beta_i) \cdot \kappa_i + \alpha_i\kappa_{i+1} = (\gamma_i - \gamma_{i-1}), \tag{9}$$

and $1 \leq i \leq (n-1)$, and the arguments are ($\kappa_o = \kappa_n = 0$):

$$\alpha_i = \frac{1}{h_i} - \frac{\tau}{\sinh(\tau \cdot h_i)} \;\; ; \;\; \beta_i = \frac{\tau \cdot \cosh(\tau \cdot h_i)}{\sinh(\tau \cdot h_i)} - \frac{1}{h_i} \;\; ; \;\; \gamma_i = \frac{\tau^2 (y_{i+1} - y_i)}{h_i}. \tag{10}$$

2.3. Numerical examples

2.3.1. Example 1

Fit the following collection of point pairs using a natural cubic spline.

x	0	1	2	3	4
y	−8	−7	0	19	56

We assume $\kappa_0 = \kappa_4 \equiv 0$; thus $h_o = x_1 - x_0 = 1 = h_1 = h_2 = h_3$, and $u_1 = 2 \cdot (h_1 - h_0) = 4 = u_2 = u_3$. The tridiagonal system (3) yields

$$\begin{bmatrix} 4 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} \kappa_1 \\ \kappa_2 \\ \kappa_3 \end{bmatrix} = \begin{bmatrix} 36 \\ 72 \\ 108 \end{bmatrix} \Rightarrow \begin{bmatrix} \kappa_1 \\ \kappa_2 \\ \kappa_3 \end{bmatrix} \cong \begin{bmatrix} 6.4285 \\ 10.2857 \\ 24.4285 \end{bmatrix}. \tag{11}$$

As an illustration, $S_2(x)$ is given by

$$S_2(x) = 2.3571 \cdot (x-2)^3 + 5.1428 \cdot (x-2)^2 + 11.5 \cdot (x-2), \tag{12}$$

for instance, $S_2(3) \cong 18.9999$ and $S_2(2.5) \cong 7.3303$.

2.3.2. Example 2

Figure 1 depicts radial velocity profiles that represent the laminar fluid flow within a pipeline.

These velocity profiles were obtained by solving the Navier-Stokes equations under simplifying assumptions. The symbols represent the discrete point pairs, the abscissas correspond to the normalized radial coordinate from the center, and the y-coordinates are the normalized radial velocities. We fit all data sets by using natural splines. To solve the system (3), we recommend the Thomas method, i.e., a direct frontal solver for tridiagonal matrixes [1–3]. We also recommend employing a quick-search algorithm to evaluate the piecewise function. Indeed, for an arbitrary x, we need to determine what is the interval where this abscissa lies, i.e., $x \in [x_i, x_{i+1}]$.

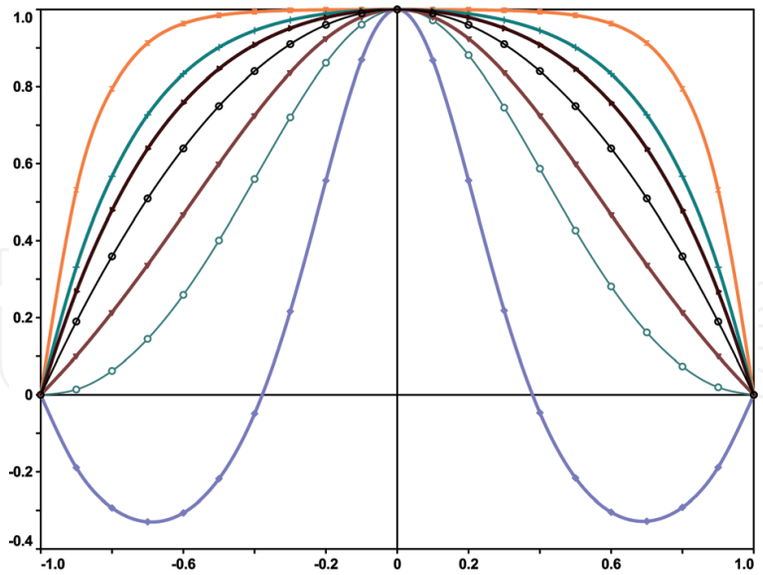

Figure 1. Discrete velocity profiles that were fitted by splines.

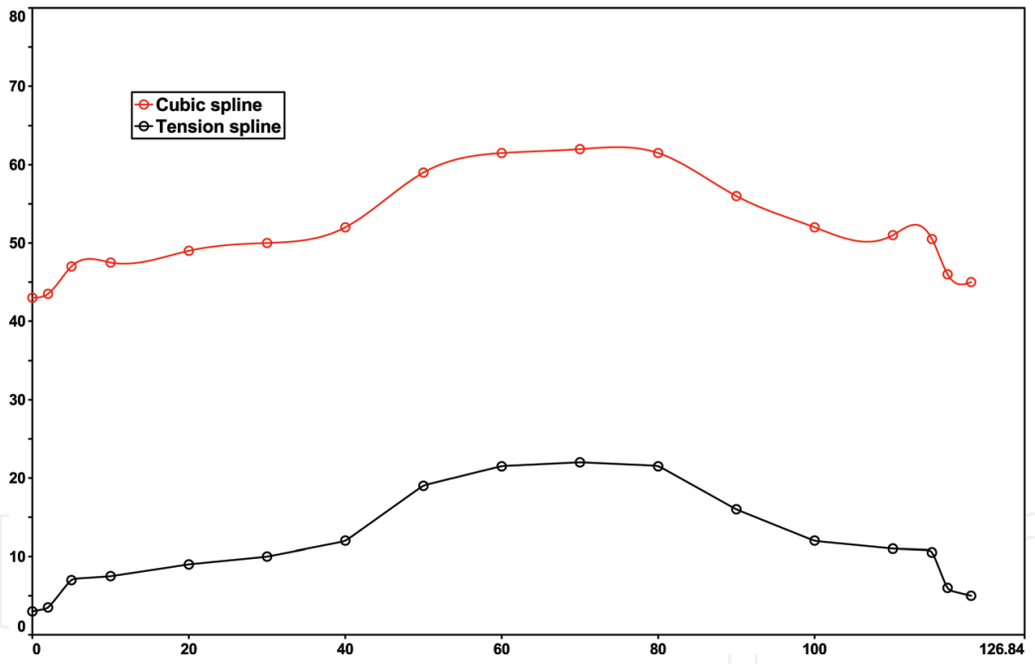

Figure 2. It depicts a car-like profile that we fit by cubic and tension splines.

2.3.3. Example 3

We finalize the examples by comparing cubic and tension splines. **Figure 2** depicts a car-like profile polygon that we would like to interpolate. We try both splines mentioned above. We highlight in red color the cubic spline (top) interpolant, while the tension spline is black (bottom curve). We observe that the cubic spline experiences inflection points because the car-shaped

polygon is challenging. This latter is the kind of application for tension splines where we seek to attract the spline toward the input polyline. We notice that we achieve that goal herein.

3. Bèzier, B-spline, and NURBS curves

The appropriate representation and meshing of the computational domain for the physical problem under study are necessary premises for a satisfactory computer simulation. In fact, one of the most demanding computational tasks in a simulation is defining the geometry because it will impact many aspects of the study such as the grid generation process [4]. Therefore, special methods must be applied to fit discrete data without sudden changes in curvature. The approach should be free of inflection points, and at minimum, it must enforce continuity C^2 of the fitted curve. In this chapter, this goal is achieved by using Bèzier, B-spline, and NURBS curves and surfaces [5, 6].

A Bèzier curve (*BC*), $\underline{\mathcal{B}}$, shown in **Figure 3**, is obtained by specifying the coordinates of a series of points in space, such that only the first and last ones fall on the originally given curve. All these points are known as control points, and the polyline resulting from connecting them with straight lines is called control polygon, which mimics the original curve, allowing an easy control of its shape. Although inflection points may be present in Bèzier curves, they are less common than in polynomials or other analytical functions [5, 6].

Global Bèzier curves, i.e., only one curve represents the given polyline, provide a powerful tool in geometry definition; however, complex shapes require a large number of constraints,

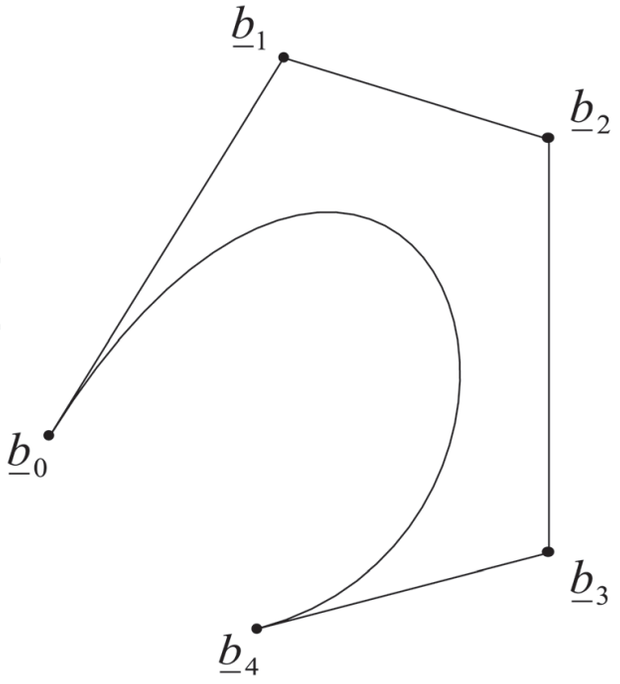

Figure 3. Fourth-order Bèzier curve with highlighted control points.

making their use prohibitive. It is therefore beneficial to represent them by using piecewise continuous Bèzier curves called B-spline curves [5]. In fact B-spline curves are a widely utilized representation for geometrical entities in computer-aided geometric design (*CAGD*) systems. Their convex hull, local support, shape-preserving forms, affine invariance, and variation-diminishing properties are extremely attractive in engineering design applications [4].

A particular Bèzier curve is set up by its parametric representation; let $\underline{\mathcal{B}} : \mathbb{R} \rightarrow \mathbb{R}^2$ be defined by

$$\underline{\mathcal{B}}(t) = \sum_{i=0}^{m} \underline{b}_i \cdot B_i^{(m)}(t), \quad t \in I = [0,1], \tag{13}$$

here, m denotes the order or degree of the curve, $B_i^{(m)}(t)$ are the Bernstein polynomials, defined as

$$B_i^{(m)}(t) = \frac{m!}{i!(m-i)!} t^i \cdot (1-t)^{m-i} \quad ; \quad \sum_{i=0}^{m} B_i^m(t) = 1, \tag{14}$$

and $\underline{b}_i$ are the control points. Notice in Eq. (14) that Bernstein polynomials satisfy the barycentric property, meaning that they add up to 1, which explains why a given curve cannot be outside its control polygon that is the convex-hull property. The control points of a given BC can be calculated in several ways since the Bèzier curve evaluated in $t = t_k$ must provide the corresponding base point $\underline{p}_k$; a linear system of equations can be formed for the unknown control points as

$$\sum_{i=0}^{m} \underline{b}_i \cdot B_i^{(m)}(t_k) = \underline{p}_k, \quad k = 0, .., m, \tag{15}$$

where the number of base points equals $(m+1)$; we compute the value of the parameter t_k by [6, 7]

$$t_k = \frac{s_k}{s_m}; \quad s_0 = 0; \quad s_k = s_{k-1} + \left\| \underline{p}_k - \underline{p}_{k-1} \right\|, \quad k = 1, \ldots, m, \tag{16}$$

which is the well-known chord-length parametrization.

The last approach is a powerful tool in curve design, but it has a limitation: if the geometry that we model has a complex shape (i.e., a significant number of base points), then its Bèzier curve representation may be of a prohibitively high degree. Since the Bèzier curve is forced to satisfy several constraints according to Eq. (15), the resulting curve may experience inflection points and sudden changes in curvature (see **Figure 4**, where $\underline{p}_k$ points in Eq. (15) are represented by circles). For practical purposes, degrees exceeding 10 are prohibitive [5, 6].

Such complex geometries can be modeled using piecewise polynomial curves named B-spline curves [5, 6] (see **Figure 5**). B-spline curves are a set of Bèzier curves of mth degree that must

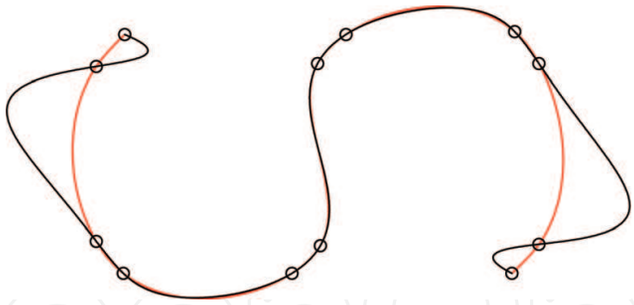

Figure 4. We interpolated with a Bèzier (black) and a cubic B-spline (red) curves.

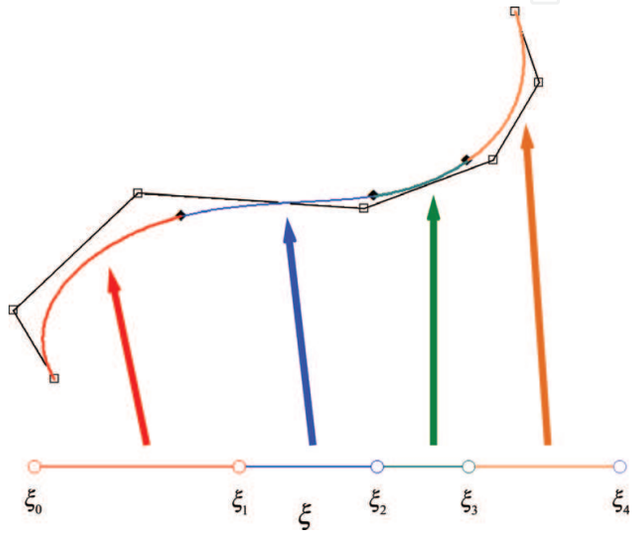

Figure 5. A B-spline curve, $\underline{C}(\xi)$, is the union of piecewise continuous curves.

satisfy at least the $C^{(m-1)}$ continuity. A spline curve $\underline{C}$ is the continuous mapping of a collection of global parameter values $\xi_0, \xi_1, \ldots, \xi_{L-1}, \xi_L$ into $\mathbb{R}^2$, where each interval $[\xi_i, \xi_{i+1}]$ is mapped onto a polynomial curve segment as shown in **Figure 5**. We define $\overline{\Omega} = [\xi_0, \xi_L]$ as the computational space. A local coordinate t for the interval $[\xi_i, \xi_{i+1}]$ can be defined by setting [5]:

$$t = \frac{\xi - \xi_i}{\xi_{i+1} - \xi_i} = \frac{\xi - \xi_i}{\Delta_i}, \quad \xi \in [\xi_i, \xi_{i+1}]. \tag{17}$$

3.1. C^2 cubic curves

Let $\underline{d}_{-1}, \underline{d}_0, \ldots, \underline{d}_L, \underline{d}_{L+1}$ be a set of $(L+3)$ points defining the de Boor's polygon that generates L individual cubic curves as shown in **Figure 6**. The required $(3L+1)$ Bèzier control points are calculated with the aid of C^1 and C^2 continuity criteria. C^1 conditions lead to

$$\underline{b}_{3i} = \frac{\Delta_i}{\Delta_{i-1} + \Delta_i} \underline{b}_{3i-1} + \frac{\Delta_{i-1}}{\Delta_{i-1} + \Delta_i} \underline{b}_{3i+1}, \quad i = 1, \ldots, L-1, \tag{18}$$

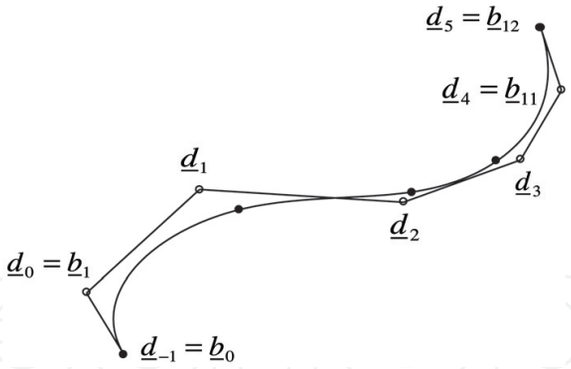

Figure 6. A C^2 cubic curve with highlighted de Boor's and junction points.

while C^2 conditions require that

$$\begin{aligned} \underline{b}_{3i-2} &= \frac{\Delta_{i-1}+\Delta_i}{\Delta}\underline{d}_{i-1} + \frac{\Delta_{i-2}}{\Delta}\underline{d}_i, \\ \underline{b}_{3i-1} &= \frac{\Delta_i}{\Delta}\underline{d}_{i-1} + \frac{\Delta_{i-2}+\Delta_{i-1}}{\Delta}\underline{d}_i. \end{aligned} \tag{19}$$

where $i = 2, \ldots, L-1,$ and $\Delta = \Delta_{i-2} + \Delta_{i-1} + \Delta_i$. The end points are

$$\begin{aligned} &\underline{b}_0 = \underline{d}_{-1} \;\; ; \;\; \underline{b}_1 = \underline{d}_0 \;\; ; \;\; \underline{b}_2 = \frac{\Delta_1}{\Delta_0+\Delta_1}\underline{d}_0 + \frac{\Delta_0}{\Delta_0+\Delta_1}\underline{d}_1, \\ &\underline{b}_{3L-2} = \frac{\Delta_{L-1}}{\Delta_{L-2}+\Delta_{L-1}}\underline{d}_{L-1} + \frac{\Delta_{L-2}}{\Delta_{L-2}+\Delta_{L-1}}\underline{d}_L \;\; ; \;\; \underline{b}_{3L-1} = \underline{d}_L \;\; ; \;\; \underline{b}_{3L} = \underline{d}_{L+1}. \end{aligned} \tag{20}$$

This construction is due to W. Boehm [5].

For cubic curves more parametrizations are available [5, 6], for instance:

1. Uniform parametrization

$$\xi_i = i \;\; ; \;\; i = 0, \ldots, L. \tag{21}$$

2. Chord-length parametrization [5]

$$\begin{aligned} &\xi_0 = 0.0; \;\; \xi_1 = \|\underline{d}_1 - \underline{d}_{-1}\|, \\ &\xi_i = \xi_{i-1} + \left\|\underline{d}_i - \underline{d}_{i-1}\right\|; \;\; i = 2, \ldots, L-1, \\ &\xi_L = \xi_{L-1} + \left\|\underline{d}_{L+1} - \underline{d}_{L-1}\right\|. \end{aligned} \tag{22}$$

3. A parametrization proposed by the author [6]

$$\begin{aligned} &\xi_0 = 0.0, \\ &\xi_{i+1} = \xi_i + \left\|\underline{d}_i - \underline{d}_{i-1}\right\| + \left\|\underline{d}_{i+1} - \underline{d}_i\right\| + \left\|\underline{d}_{i+2} - \underline{d}_{i+1}\right\|, \\ &i = 0, \ldots, L-1. \end{aligned} \tag{23}$$

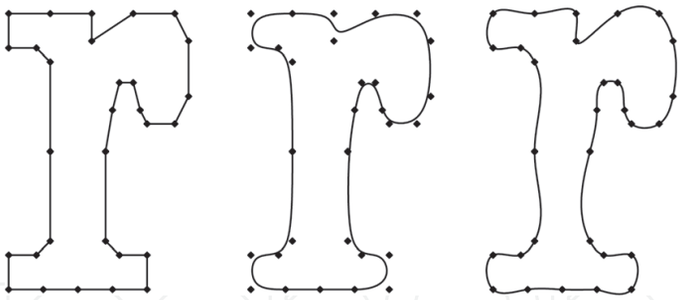

Figure 7. The inverse design and interpolation problems.

This latest parametrization has the advantage that it yields to a symmetric curve if the control polygon is symmetric as well, which may be interesting for certain applications.

3.2. Inverse design and interpolation problems

A two-dimensional geometric description based on B-spline curves requires the definition of a control polygon (the de Boor's polygon) that mimics the curve. Therefore two approaches are possible. The first method consists of providing the set of the de Boor's points (i.e., define the de Boor's polygon interactively from user's input) that defines the composite curve, which is known as "inverse design," and it has its application in "TrueType" font technology as shown in **Figure 7**. The second approach consists of defining the set of base points and then solves a linear system of equations for the de Boor's points such that the resulting curve passes through them. This latter is known as the "interpolation" problem.

Figure 7 shows the difference between the above approaches; from left to right, it has the de Boor's control polygon, inverse design, and interpolation problems both taking into account the same polygon as an argument with cubic curves.

3.3. Interpolation with cubic curves

In order to interpolate with cubic B-spline curves, we find unknown junction points such that they pass through a given set of data points $\underline{x}_0, \ldots, \underline{x}_L$ and corresponding parameter values $\xi_0, \xi_1, \ldots, \xi_{L-1}, \xi_L$. A composite cubic curve $\underline{\mathcal{C}}$, determined by its de Boor's polygon [4, 5] $\underline{d}_{-1}, \ldots, \underline{d}_{L+1}$ such that $\underline{\mathcal{C}}(\xi_i) = \underline{x}_i$, is required. The solution to this problem is obtained by finding the relationship between the data points $\underline{x}_i$ and the control vertices $\underline{d}_i$. This leads to the following linear system of equations for the unknown de Boor's points [5]:

$$\alpha_i \cdot \underline{d}_{i-1} + \beta_i \cdot \underline{d}_i + \gamma_i \cdot \underline{d}_{i+1} = (\Delta_{i-1} + \Delta_i) \cdot \underline{x}_i; \quad i = 1 \ldots L-1, \tag{24}$$

where (with $\Delta_{-1} = \Delta_L = 0$)

$$\begin{aligned} \alpha_i &= \frac{(\Delta_i)^2}{\Delta_{i-2} + \Delta_{i-1} + \Delta_i}, \\ \beta_i &= \frac{\Delta_i(\Delta_{i-2} + \Delta_{i-1})}{\Delta_{i-2} + \Delta_{i-1} + \Delta_i} + \frac{\Delta_{i-1}(\Delta_i + \Delta_{i+1})}{\Delta_{i-1} + \Delta_i + \Delta_{i+1}}, \\ \gamma_i &= \frac{(\Delta_{i-1})^2}{\Delta_{i-1} + \Delta_i + \Delta_{i+1}}. \end{aligned} \tag{25}$$

If the two Bèzier points $\underline{b}_1$ and $\underline{b}_{3L-1}$ are arbitrarily chosen, the following linear system of equations is obtained [5, 6]:

$$\begin{bmatrix} 1 & & & & & & \\ \alpha_1 & \beta_1 & \gamma_1 & & & & \\ & & \ddots & & & & \\ & & & \alpha_{L-1} & \beta_{L-1} & \gamma_{L-1} & \\ & & & & & 1 \end{bmatrix} \cdot \begin{bmatrix} \underline{d}_0 \\ \underline{d}_1 \\ \vdots \\ \underline{d}_{L-1} \\ \underline{d}_L \end{bmatrix} = \begin{bmatrix} \underline{r}_0 \\ \underline{r}_1 \\ \vdots \\ \underline{r}_{L-1} \\ \underline{r}_L \end{bmatrix}, \tag{26}$$

where

$$\begin{aligned} \underline{r}_0 &= \underline{x}_0, \\ \underline{r}_i &= (\Delta_{i-1} + \Delta_i) \cdot \underline{x}_i \;\; ; \;\; i = 1 \ldots L-1, \\ \underline{r}_L &= \underline{x}_L. \end{aligned} \tag{27}$$

The first and last vertices of the polygon are given by

$$\underline{d}_{-1} = \underline{x}_0, \quad \underline{d}_{L+1} = \underline{x}_L. \tag{28}$$

The points $\underline{b}_1$ and $\underline{b}_{3L-1}$ can be calculated from a given end condition. There are two possibilities for the choice of B-spline ending conditions. A natural spline requires that

$$\begin{aligned} \frac{d^2}{dt^2}\underline{s}_0(0) &= 6 \cdot (\underline{b}_2 - 2\underline{b}_1 + \underline{b}_0) = \underline{0}, \\ \frac{d^2}{dt^2}\underline{s}_{L-1}(1) &= 6 \cdot (\underline{b}_{3L} - 2\underline{b}_{3L-1} + \underline{b}_{3L-2}) = \underline{0}. \end{aligned} \tag{29}$$

Using the relations in Eq. (29), we obtain that

$$\begin{aligned} \left[2 - \frac{\Delta_1}{\Delta_0 + \Delta_1}\right] \cdot \underline{d}_0 - \frac{\Delta_0}{\Delta_0 + \Delta_1}\underline{d}_1 &= \underline{x}_0, \\ \left[2 - \frac{\Delta_{L-2}}{\Delta_{L-2} + \Delta_{L-1}}\right] \cdot \underline{d}_L - \frac{\Delta_{L-1}}{\Delta_{L-2} + \Delta_{L-1}}\underline{d}_{L-1} &= \underline{x}_L. \end{aligned} \tag{30}$$

These two equations replace the first and last rows of the linear system of equations given in Eq. (26). Notice that in either case, natural ending conditions or prescribed tangent vectors, the linear system in Eq. (24) is a tridiagonal matrix. Since the coefficient matrix is real and scalar, and the left- and right-hand-side vectors are in fact hypervectors (i.e., an array of vectors), it is recommendable to use a type of Gaussian elimination method against multiple right-hand sides to achieve performance [6, 7]. If we recompute the interpolant in **Figure 4** but this time with a B-spline cubic curve, we then obtain a smoother and steadier interpolant free of unwelcome inflection points.

3.4. NURBS curves

NURBS curves are useful when we require an exact geometrical representation of some entities, such as circles, parabolas, ellipses, spheres, cylinders, etc. This is precisely the case in

various applications in aerospace and mechanical engineering where NURBS are quite popular [4, 5, 8–10]. For instance, a NURBS curve is defined by its rational representation in Eq. (31):

$$\underline{\mathcal{R}}(t) = \frac{\sum_{i=0}^{m} w_i \cdot B_i^m(t) \cdot \underline{b}_i}{\sum_{i=0}^{m} w_i \cdot B_i^m(t)}, \tag{31}$$

where the weights w_i are positive real scalars. The usual B-spline definition is recovered if all those weights equal 1. Generally speaking, the weights play the role of attracting the curve toward its control polygon when we increase their values [5, 8]. It turns out that specific weights lead to exact representation of circles, for instance, as shown in **Figure 8**, where the real numbers depicted are the given weights. A circle can be exactly represented by three- or four-quadratic arcs (left and right side, respectively, in **Figure 8**); this latter alternative is more attractive, for instance, to generate a surface of revolution [4, 5, 8]. NURBS also provide exact representation for 3D surfaces such as spheres and cylinders as well as volumes [4, 10]. The reader may refer to [4, 5, 8–11] for further details for this well-established area of computational geometry. We depict an example of practical interest, in the context of geomechanics, in **Figure 9**. Herein we precisely represent a near-borehole section. To describe the borehole geometry, we employ four line segments and a quadratic arc as shown.

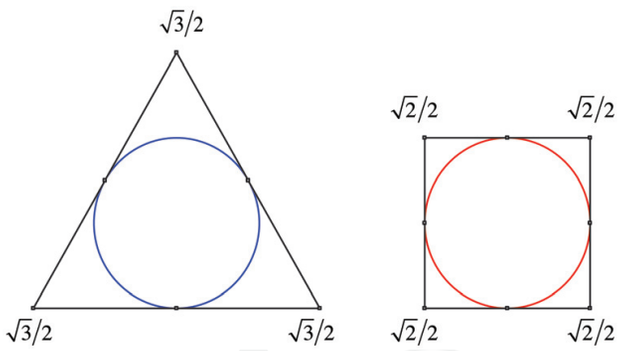

Figure 8. Two exact equivalent representations for a circle (the remaining weights equal 1).

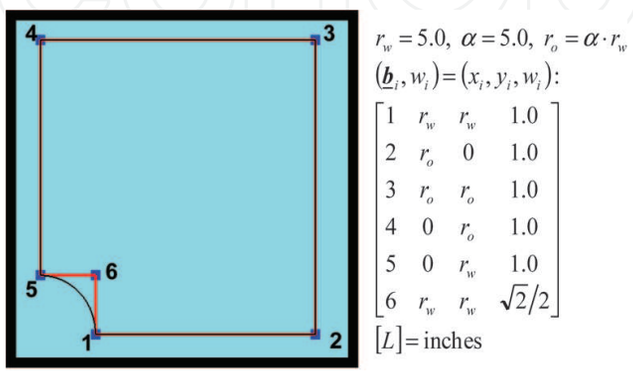

Figure 9. We represent a near borehole "exactly" by NURBS as shown.

In order to interpolate a set of points with a NURBS curve in two or three dimensions, one may follow the same procedure described with B-spline curves except that a mapping to $\mathbb{R}^4$ must be carried out first. The new input points $\underline{\tilde{x}}_i$ are given by

$$\underline{\tilde{x}}_i = w_i \cdot \begin{pmatrix} x_i & y_i & z_i & 1 \end{pmatrix}^T; \quad \underline{\tilde{x}}_i \in \mathbb{R}^4, \tag{32}$$

then the linear system in Eq. (24) with the right boundary conditions can be solved with $\underline{x}_i$ replaced by $\underline{\tilde{x}}_i$, as defined by the Eq. (32). After solving this system, the solution control polygon is still in $\mathbb{R}^4$, which implies that a mapping back to $\mathbb{R}^3$ is required:

$$\underline{d}_i = \frac{1}{w_i} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \underline{\tilde{d}}_i \;\; ; \;\; \underline{d}_i \in \mathbb{R}^3. \tag{33}$$

The latter is a straightforward procedure to reuse the subroutines already developed for B-spline curves.

3.5. Numerical examples

We implemented the proposed approaches in a computer code named "LogProc" which is a graphical user interface application developed with the C++ programming language. LogProc is proprietary software, but a free community version will be available for download from www.logproc.com. All examples were obtained applying the proposed knot sequence (23) to construct cubic composite curves. The empty circles represent de Boor's points (sample inverse designs) or base points (interpolation problems). We utilized the natural end condition in examples in **Figures 10** and **11** and the prescribed tangent in the remaining cases. We depict a typical font design application in **Figure 10** where we represent some alphabet's letters. Notice that an approach like this is appropriate to construct font outlines because they can be scaled

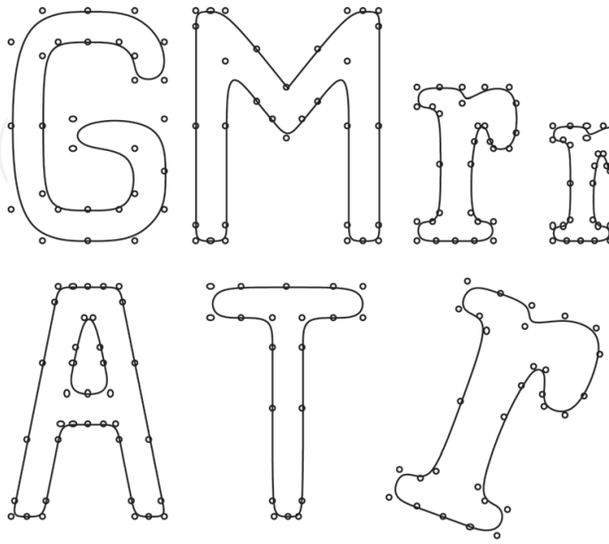

Figure 10. A typical font design application.

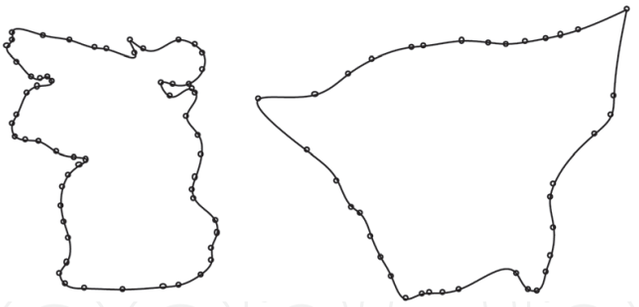

Figure 11. A pair of petroleum reservoir outlines.

and rotated. This feature is of significant interest in the "TrueType" font technology where outlines that are insensible to the resolution of the physical device in which they will be shown, such as monitors or printers, must be obtained [6]. We added a bonus section on the numerical implementation for computing splines that is available online, https://www.logproc.com/book-chapter-splines. We also include most sample datasets for downloading. We also recommend there suitable open-source libraries that are convenient to use. All vector and raster plots were prepared by logproc which can create high-quality PDF files in Windows 7 and 10, for instance.

Figure 11 shows a zenith view of a pair of petroleum reservoir outlines. Few base points permit to approximate their complex shape. These shapes could be used as arguments to generate an unstructured mesh appropriate to simulate the flow in a porous media [7].

We interpolate a discrete blade geometry approximation in both **Figures 12** and **13**. A3K7 profiles are constructed in the same way as NACA 65, but they present rounded trailing edges [6]. A3K7's shape is represented by 46 base points and circle arcs both in leading and trailing edges. Therefore the curves that interpolate the data points are tangents to circle arcs to ensure C^1 continuity between these entities. The profiles in the left of **Figure 12** were obtained applying Eq. (15) to construct C^1 single Bèzier curves (two curves, each of them approximating to suction and pressure sides, respectively). However, note that the resulting curves have unexpected inflection points on the suction side. The enforcement of continuity conditions

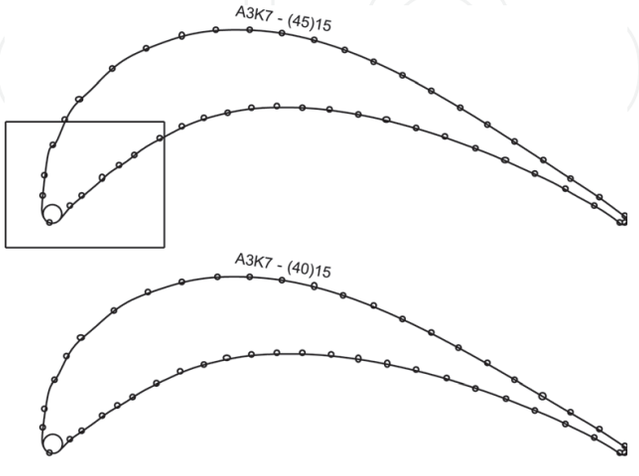

Figure 12. A3K7 interpolated by Bèzier curves.

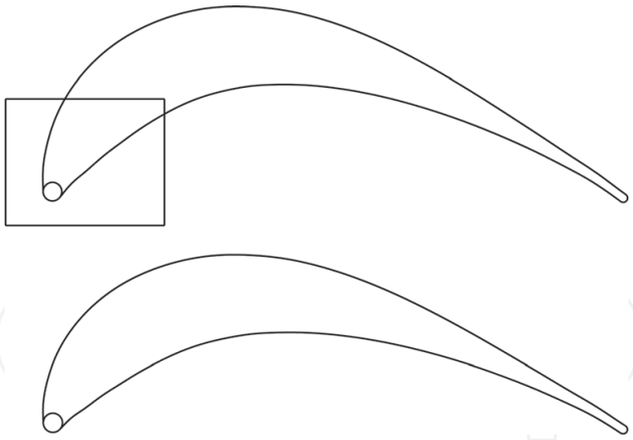

Figure 13. A3K7 interpolated by B-spline curves.

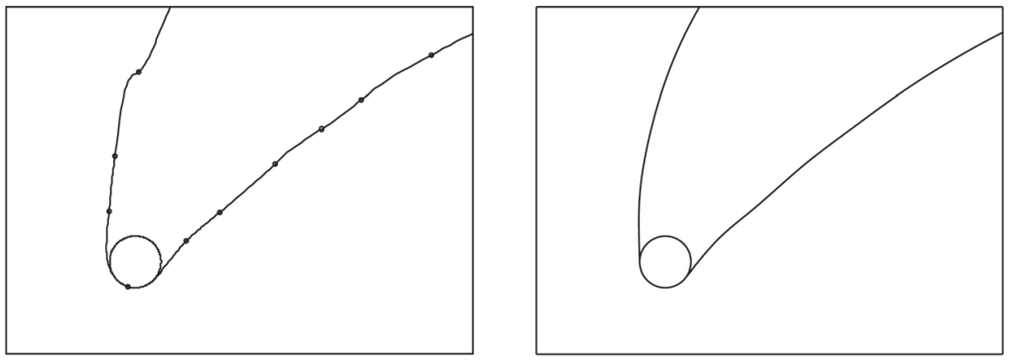

Figure 14. Details highlighted in **Figures 12** and **13**.

implies a large number of constraints after Eq. (15), which explains this behavior. If we replace the former Bèzier curves by two B-spline curves, computed from Eqs. (24) and (29), we obtain a smooth A3K7 geometry description (see profiles in the right-hand side of **Figure 12**). **Figure 14** zooms in to show that the resulting geometrical description is smooth and free of unwelcome features such as inflection points.

4. Interpolation surfaces

Let $\underline{\mathcal{S}}^{Int} : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ be a two-parameter mapping which represents a given surface. If structured data, i.e., tensor product data, needs to be interpolated, one may expect to come up with tensor product surfaces as well, where two parameters (ξ, η) allow covering two different directions associated with the surface. In the computational space, i.e., in the plane (ξ, η), the domain, $\overline{\Omega} = \left[\xi_0, \xi_{L_u}\right] \times \left[\eta_0, \eta_{L_v}\right]$, is a rectangle, and its image is the surface in 3D as shown in **Figure 15**, where L_ξ and L_η are the number of curves in their respective directions.

B-spline tensor product surfaces allow interpolating structured data, and they are defined as the tensor product of two families of curves $\underline{\mathcal{C}}_i^k(\xi)$ and $\underline{\mathcal{D}}_j^l(\eta)$, which is

$$\underline{\mathcal{L}}^{Int}\left(\xi_i,\eta_j\right)=\underline{x}_{ij};\quad \underline{\mathcal{S}}^{Int}_{klij}(\xi,\eta)=\underline{\mathcal{C}}^{k}_{i}(\xi)\otimes\underline{\mathcal{D}}^{l}_{j}(\eta);\quad (\xi,\eta)\in\overline{\Omega} \tag{34}$$

In applications of practical interest, usually cubic piecewise continuous curves are preferred because they provide a global C^2 representation that is smooth enough, called a bicubic surface [5, 8, 9].

4.1. Creating a surface of interpolation

The following steps describe creating a surface of interpolation:

1. An input control polygon, whose points are in $\mathbb{R}^3$, is provided. They correspond to data that is structured and ordered, which is usually a matrix-type array of points (see left side of **Figure 16**). For simplicity, points in the i-direction are associated with the ξ parameter while $j's$ are associated with η.

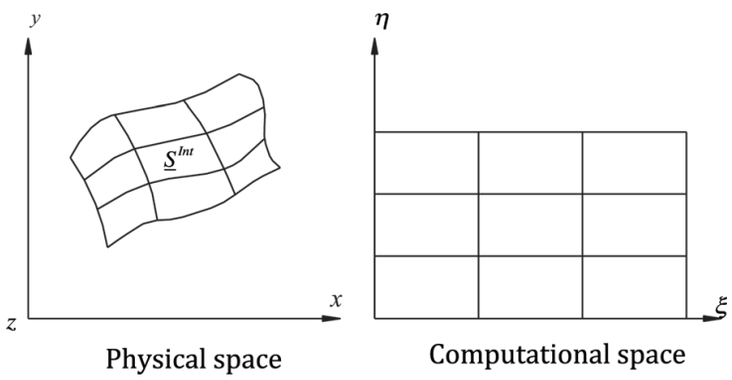

Figure 15. We depict the mapping between physical and computational spaces.

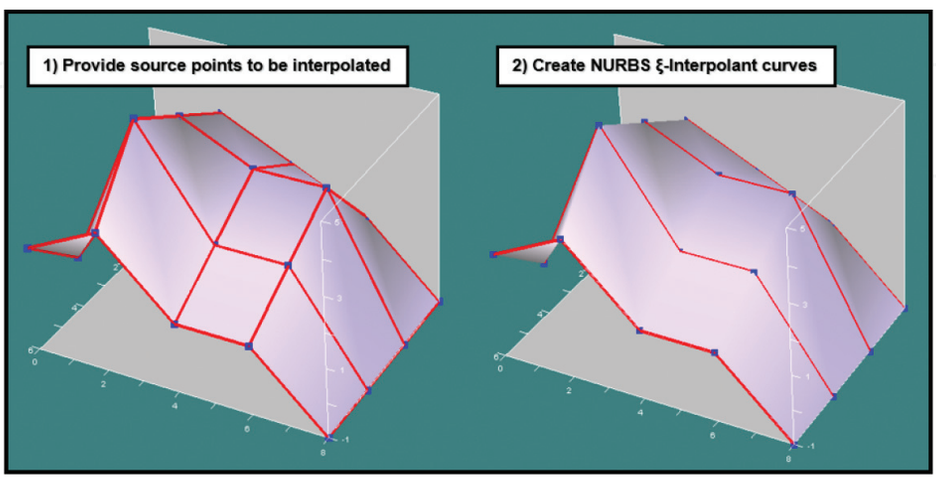

Figure 16. The first two steps to interpolate structured data are depicted: input control polygon (left) and ξ-interpolants (right) are shown.

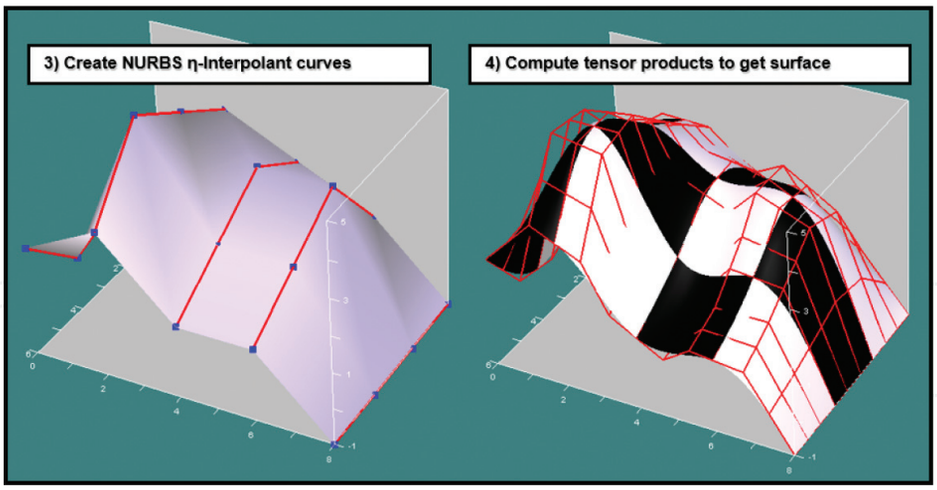

Figure 17. The last two steps to interpolate structured data are depicted: η-interpolants (left) and resulting bicubic patches are shown. The de Boor's control polygon is highlighted in red lines.

2. Create interpolation curves with constant values of η, so-called ξ-interpolants (see right side of **Figure 16**).

3. Proceed accordingly with previous step, interpolation curves with constant values of ξ; so-called η-interpolants are created this time (see left side of **Figure 17**).

4. Compute the tensor product between ξ- and η-interpolants in order to get bicubic patches (see right side of **Figure 17**).

The right-hand side in **Figure 17** shows typical bicubic patches as a chessboard surface emphasizing that we deal with a piecewise continuous entity. The computational cost associated with the above algorithm is reasonable because the most expensive part is computing the interpolants (see Section 3).

5. Translational surfaces

These surfaces are again a two-parameter mapping, $\underline{\sigma}^T : \mathbb{R}^2 \rightarrow \mathbb{R}^3$, but their construction procedure is simpler than interpolation surfaces; see, for instance, [5, 8, 9]. The idea here is just literally translating a curve $\underline{\alpha}$ along another curve $\underline{\beta}$, which yields

$$\underline{\sigma}^T(\xi, \eta) = \underline{\alpha}(\xi) + \underline{\beta}(\eta). \tag{35}$$

This idea became very popular in CAGD systems long time ago. Those systems usually support a command which allows extruding a geometrical entity, for instance, a cylinder can be easily created by extruding a circle along a straight line. **Figure 18** shows the above procedure applied to an aircraft wing where an NURBS airfoil profile was translated or extruded along a straight line accordingly. We interpolated an NACA 65 polyline with a NURBS curve as we mentioned in Section 3.4.

This procedure becomes very useful in the geometrical reconstruction of oil reservoirs (*RS*). Indeed, we reconstructed the geometry of RS in [12] by using B-spline surfaces. The technique exploits input mesh's simplicity to build a robust piecewise continuous geometrical representation using Bèzier bicubic patches. We manage the reservoir's topology with interpolation

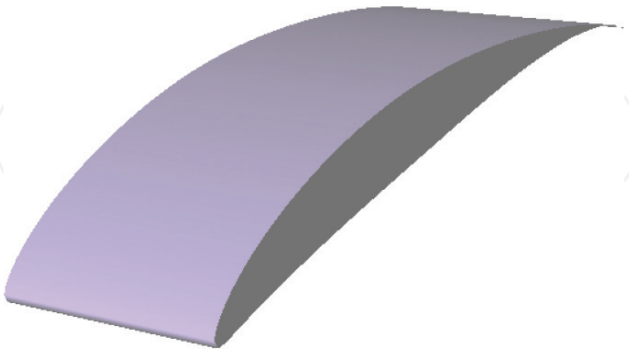

Figure 18. An aircraft wing by translating an NACA profile accordingly.

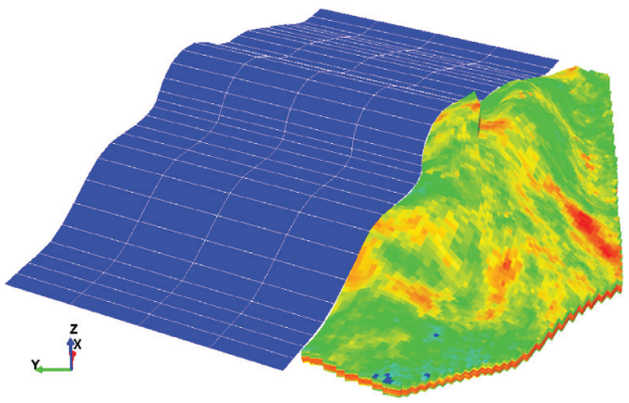

Figure 19. A translational surface.

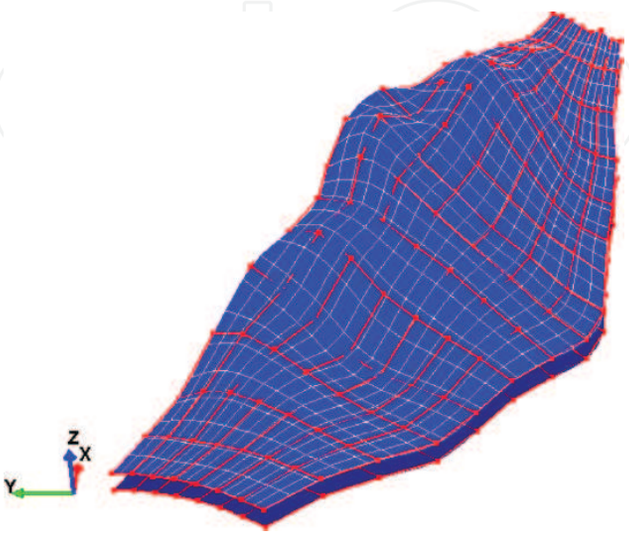

Figure 20. An interpolation surface.

surfaces, while translational surfaces allow extrapolating it toward its side burdens. After that, transfinite interpolation can be applied to generate decent hexahedral meshes. **Figure 19** shows a sample translational surface that we obtain by extruding a curve that interpolates the reservoir's edge as shown. We render the surfaces in blue color with a white wireframe, while the RS is the color-contoured surface that represents the porosity, a scalar property. We tackle the RS itself after interpolating the control polygon that **Figure 20** highlights in red color. The polygon is a 17×9 array of points representing the RS topology. The procedure works well for a variety of so-called open-to-the-public RS data sets that we reconstructed in [12]. It is also possible to utilize these NURBS curves and surfaces as interfaces for gluing nonmatching interfaces for the finite element method as we showed in [13].

6. Duchon splines

In the context of applications in statistical analysis involving very high dimensional data sets, response surfaces are growing popularity. By running the simulations at a set of points (e.g., experimental design) and fitting response surfaces, i.e., splines, for instance, to the resulting input-output data that is characterized by sparsity, we can obtain fast surrogates for the objective function for optimization purposes [14, 15]. The appeal of the latter approach goes beyond reducing runtime. Since the method begins with experimental design, statistical analyses can be done to identify which input variables are the most important, and thus we can create "main effect plots" to visualize input-output relationships [14]. We must recognize interpolation methods in which the basis functions are fixed and those in which they have parameters that are tuned (e.g., kriging, which has a statistical interpretation that allows one to construct an estimate of the potential error in the interpolator). We refer the reader to [14, 15] for further reading.

There are different ways to approximate a function of several variables: multivariate piecewise polynomials, splines, and tensor product methods, among others. All these approaches have advantages and drawbacks, but if the rank of the linear system to solve may become large, a natural choice is radial basis functions, which are also useful in lower dimensional problems [14, 16, 17]. This may be particularly true if the input data is scattered, which excludes tensor product methods at first glance. Duchon splines are a class of positive definite and compactly supported radial functions, which consist of univariate polynomial within their support. It can be proven that they are of minimal degree and unique up to a constant factor, for given smoothness and space dimension [18]. They are particularly suitable to compute interpolants for very large scatter datasets [17].

Duchon splines, denoted herein as s, are defined by [17, 18]

$$\begin{aligned} s(\underline{x}) &= \sum_j \lambda_j \cdot \varphi\left(\rho_j\right) + p_n(\underline{x}) \;\; ; \;\; n = 2, 3 \\ \rho_j &= \left\| \underline{x} - \underline{x}^j \right\| \\ \varphi(\rho) &= \rho^2 \ln \rho, \end{aligned} \tag{36}$$

where $p_n(\underline{x})$ is a linear polynomial in two or three dimensions:

$$\begin{aligned} p_2(\underline{x}) &= ax + by + c \\ p_3(\underline{x}) &= dx + ey + fz + g \;\; ; \;\; \lambda_j, a, \ldots, g \in \mathbb{R}. \end{aligned} \tag{37}$$

Notice that λ_j and the polynomial coefficients are all scalar quantities. In order to guarantee existence and uniqueness for these splines, an orthogonality condition with respect to linear polynomials is enforced, for instance, in two dimensions this yields to

$$\sum_j \lambda_j = \sum_j \lambda_j x_j = \sum_j \lambda_j y_j = 0. \tag{38}$$

By considering this result, the interpolation problem becomes

$$s(\underline{x}^i) = \sum_j \lambda_j \cdot \varphi\left(\rho_j^i\right) + p_n(\underline{x}^i) = F^i, \tag{39}$$

which implies m points plus $n+1$ orthogonality conditions; here, F^i are the nodal values to be interpolated. The resultant linear system to solve for is of $(m+n+1)$ rank.

Duchon splines are certainly suitable to interpolate scattered data sets that we cannot tackle with the tensor product surfaces that we discussed before. Indeed, **Figure 21** depicts such an application, in optimization, where an objective function that we wish to minimize was sampled randomly by Monte-Carlo (*MC*) realizations. To compute a minimum, we interpolate the black dots, and then we minimize the resulting spline with standard Newton stochastic techniques [15]. It is true that Duchon splines are a valid choice for "surrogate" models for such applications.

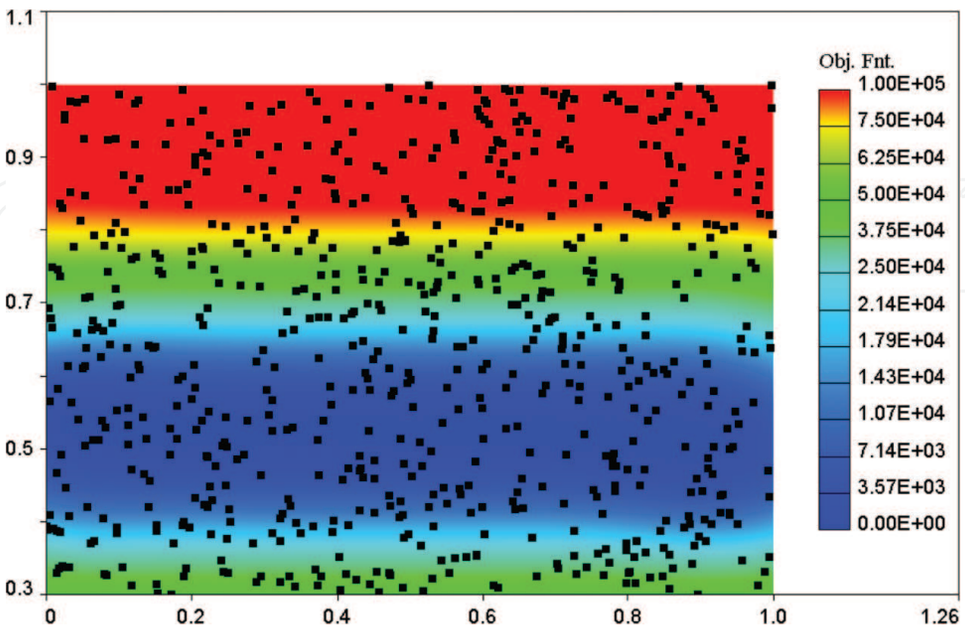

Figure 21. Discrete MC data with Duchon splines.

7. Concluding remarks

We presented a concise introduction to scalar and parametric spline interpolants. We introduced cubic and tension splines for scalar functions, and then we generalized them for the parametric case via Bèzier, B-spline, and NURBS curves. These latter entities are of the particular interest for applications in CAGD. We thus elaborated on topics such as inverse design and interpolation. We extended the treatment also to cover interpolation and translational surfaces with examples in mechanical and petroleum engineering. We wrapped up with the topic of interpolating sparse very high dimensional data sets via Duchon splines which are a kind of response surfaces suitable for applications in statistical analysis and optimization.

Acknowledgements

We acknowledge the financial support of the project "Reduced-Order Parameter Estimation for Underbody Blasts" funded by Army Research Laboratory.

Author details

Horacio Florez* and Belsay Borges

*Address all correspondence to: florezg@gmail.com

Computer Science Department, The University of Texas at El Paso, El Paso, USA

References

[1] Curtis F, Wheatley P. Applied Numerical Analysis. Pearson: USA; 2004

[2] Kincaid D, Cheney W. Numerical Analysis. USA: Brooks/Cole Publishing Company; 1991

[3] Atkinson K. An Introduction to Numerical Analysis. New York: John Wiley & Sons; 1978

[4] Thompson J, Weatherill N. Handbook of Grid Generation. CRC Press: Boca Raton; 1999

[5] Farin G. Curves and Surfaces for Computer-aided Geometric Design A Practical Guide. 4th ed. San Diego: Academic Press; 1993

[6] Florez H. A new method for building B-spline curves and its application to geometry design and structured grid generation. In: ASME International 2001 DETC Conference; Pittsburgh, Pennsylvania. 2001

[7] Florez H. Domain decomposition methods for geomechanics [Ph.D. thesis]. In: The University of Texas at Austin. 2012

[8] Farin G. NURBS from Projective Geometry to Practical Use. 2nd ed. A K Peters: Massachusetts; 1999

[9] Piegl L, Tiller W. The NURBS Book. 2nd ed. Berlin: Springer; 1997

[10] Hughes T, Cottrell J, Bazilevs Y. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Computer Methods in Applied Mechanics and Engineering. 2005;**194**:4135-4195

[11] De Berg M, van Kreveld M, Overmars M, Schwarzkopf O. Computational Geometry Algorithms and Applications. Berlin: Springer; 2000

[12] Florez H, Manzanilla-Morillo R, Florez J, Wheeler M. Spline-based reservoir's geometry reconstruction and mesh generation for coupled flow and mechanics simulation. Computational Geosciences. 2014;**18**:949-967

[13] Florez H, Wheeler M. A mortar method based on NURBS for curved interfaces. Computer Methods in Applied Mechanics and Engineering. 2016;**310**:535-566. ISSN 0045-7825. DOI: 10.1016/j.cma.2016.07.030

[14] Jones DR. A taxonomy of global optimization methods based on response surfaces. Journal of Global Optimization. 2001;**21**(4):345-383

[15] Schuëller GI, Jensen HA. Computational methods in optimization considering uncertainties—An overview. Computer Methods in Applied Mechanics and Engineering. 2008;**198**(1):2-13

[16] Hagen H. Curve and Surface Design. Chapter 8: A Survey of Parametric Scattered Data Fitting Using Triangular Interpolants. USA: SIAM; 1992

[17] Buhmann M. Radial basis functions. Acta Numerica. 2000:1-38

[18] Wendland H. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. Advances in Computational Mathematics. 1995;**4**:389-396