# 1
# Applications and motivations

In practical applications over a wide field of study one often faces the problem of reconstructing an unknown function $f$ from a finite set of discrete data. These data consist of data sites $X = \{x_1, \dots, x_N\}$ and data values $f_j = f(x_j)$, $1 \le j \le N$, and the reconstruction has to approximate the data values at the data sites. In other words, a function $s$ is sought that either *interpolates* the data, i.e. that satisfies $s(x_j) = f_j$, $1 \le j \le N$, or at least *approximates* the data, $s(x_j) \approx f_j$. The latter case is in particular important if the data contain noise.

In many cases the data sites are scattered, i.e. they bear no regular structure at all, and there is a very large number of them, easily up to several million. In some applications, the data sites also exist in a space of very high dimensions. Hence, for a unifying approach methods have to be developed which are capable of meeting this situation. But before pursuing this any further let us have a closer look at some possible applications.

## 1.1 Surface reconstruction

Probably the most obvious application of scattered data interpolation and approximation is the reconstruction of a surface $\mathcal{S}$. Here, it is crucial to distinguish between explicit and implicit surfaces. *Explicit* surfaces play an important role in terrain modeling, for example. They can be represented as the graph of a function $f : \Omega \to \mathbb{R}$ defined on some region $\Omega \subseteq \mathbb{R}^d$, where $d$ is in general given by $d = 2$. Staying with the terminology of terrain modeling, the data sites $X \subseteq \Omega$ depict certain points on a map, while a data value $f_j = f(x_j)$ describes the height at the point $x_j$. The data sites might form a regular grid, they might be situated on isolines (as in Figure 1.1), or they might have no structure at all. The region $\Omega$ itself might also carry some additional information; for example, it could represent the earth. Such additional information should be taken into account during the reconstruction process.

The reconstruction of an *implicit* surface, or more precisely of a compact, orientable manifold, is even more demanding. Such surfaces appear for example as sculptures, machine parts, and archaeological artifacts. They are often digitized using laser scanners, which easily produce huge *point clouds* $X = \{x_1, \dots, x_N\} \subseteq \mathcal{S}$ consisting of several million points in $\mathbb{R}^3$. In this situation, the surface $\mathcal{S}$ can no longer be represented as the graph of a single

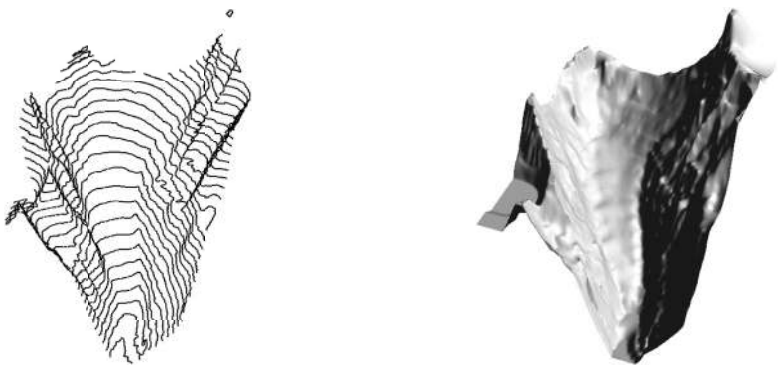

Fig. 1.1 Reconstruction of a glacier from isolines.

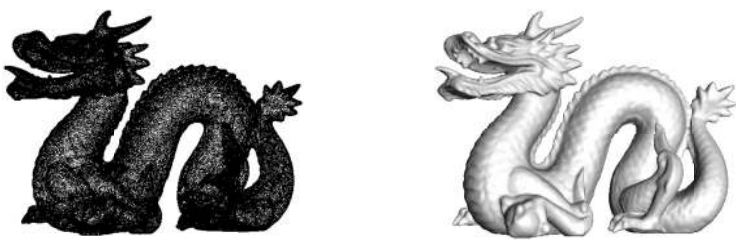

Fig. 1.2 Reconstruction (on the right) of the Stanford dragon (on the left).

function $f$. There are in the main two different approaches to building accurate models for implicit surfaces. In the first approach, one tries to find local *parameterizations* of the object that allow an efficient rendering. However, for complicated models (such as the dragon shown in Figure 1.2) this approach is limited. In the second approach, one tries to describe $\mathcal{S}$ as the zero-level set of a function $F$, i.e. $\mathcal{S} = \{x \in \Omega : F(x) = 0\}$. Such an implicit representation easily delivers function-based operations, for example shape blending or deformation or any other constructive solid geometry (CSG) operation such as the union, difference, or intersection of two or more objects.

The function $F$ can be evaluated everywhere, which allows stepless zooming and smooth detail-extraction. Furthermore, it gives, to a certain extent, a measure of how far away a point $x \in \Omega$ is from the surface. Moreover, the surface normal is determined by the gradient of $F$ whenever the representation is smooth enough.

The price we have to pay for such flexibility is that an implicit surface does not automatically lead to a fast visualization. An additional step is necessary, which is normally provided by either a ray-tracer or a polygonizer. But, for both, sufficiently good and appropriate solutions exist. Since our measured point cloud $X$ is a subset of the surface $\mathcal{S}$ we are looking for an approximate solution $s$ that satisfies $s(x_j) = 0$ for all $x_j \in X$. Obviously these interpolation conditions do not suffice to determine an accurate approximation to the

surface, since, for example, the zero function satisfies them. The remedy for this problem is to add additional *off-surface* points. To make this approach work, we assume that our surface is the boundary of a compact set and that the function $F$ can be chosen such that $F$ is positive inside and negative outside that set. We also need surface normals to the unknown surface. If the data comes from a mesh or from a laser scanner that provides also normal information via its position during the scanning process these normals are immediately to hand. Otherwise, they have to be estimated from the point cloud itself, which can be done in two steps. In the first step, for each point $x_j \in X$ we search its $K \ll N$ nearest neighbors in $X$ and try to determine a local tangent plane. This can be done by a *principal component analysis*. Let us assume that $\mathcal{N}(x_j)$ contains the indices of these neighbors. Then we compute the *center of gravity* of $\{x_k : k \in \mathcal{N}(x_j)\}$, i.e. $\hat{x}_j := K^{-1} \sum_{k\in\mathcal{N}(x_j)} x_k$, and the associated *covariance matrix*

$$\mathrm{Cov}\,(x_j) := \sum_{k\in\mathcal{N}(x_j)} (x_k - \hat{x}_j)(x_k - \hat{x}_j)^T \in \mathbb{R}^{3\times 3}.$$

The eigenvalues of this positive semi-definite matrix can be computed numerically or even analytically. They indicate how closely the neighborhood $\{x_k : k \in \mathcal{N}(x_j)\}$ of $x_j$ determines a plane. To be more precise, if we have two eigenvalues that are close together and a third one, which is significantly smaller than the others, then the eigenvectors for the first two eigenvalues determine the plane, while the eigenvector for the smallest eigenvalue determines the normal to this plane. Hence, we have a tool for not only determining the normal but also deciding whether a normal can be fitted at all.

The second step deals with orienting consistently the normals just created. If two data points $x_j$ and $x_k$ are close then their associated normalized normals $\eta_j$ and $\eta_k$ must point in nearly the same direction, which means that $\eta_j^T \eta_k \approx 1$. This relation should hold for all points that are sufficiently close. To make this more precise, we use graph theory. First, we build a *Riemann graph*. This graph has a vertex for every normal $\eta_j$ and an edge $e_{j,k}$ between the vertices of $\eta_j$ and $\eta_k$ if and only if $j \in \mathcal{N}(x_k)$ or $k \in \mathcal{N}(x_j)$. The *cost* or *weight* $w(e_{j,k})$ of such an edge measures the deviation of the normals $\eta_j$ and $\eta_k$; for example, we could choose $w(e_{j,k}) = 1 - |\eta_j^T \eta_k|$. Hence, the normals are taken to be consistently oriented if we can find directions $b_j \in \{-1, 1\}$ such that $\sum_{e_{j,k}} b_j b_k w(e_{j,k})$ is minimized. Unfortunately, it is possible to show that this problem is NP-hard and hence that we can only find an approximate solution. The idea is simply to start with an arbitrary normal and then to propagate the orientation to neighboring normals. To this end, we compute the *minimal spanning tree* or *forest* for the Riemann graph. Since the number of edges in this graph is proportional to $N$, any reasonable algorithm for this problem, for example Kruskal's algorithm, will work fine in an acceptable amount of time. After that, we propagate the orientations by traversing the minimal spanning tree.

Once we have oriented the normals, this allows us to extend the given data sets by off-surface points. This can be done by for example adding one point along each normal on the outside and one on the inside of the surface. Special care is necessary to avoid the

situation where an outside point belonging to one normal is actually an interior point in another part of the surface or that a supposedly interior point is so far away from its associated surface point that it is actually outside the surface at another place. The associated function values that $s$ should attain are chosen to be proportional to the signed distance of the point from the surface.

Another possible way of adding off-surface points is based on the following fact. Suppose that $x$ is a point which should be added. If $x_j$ denotes its nearest neighbor in $X$ and if $X$ is a sufficiently dense sample of $\mathcal{S}$, then $x_j$ comes close to the projection of $x$ onto $\mathcal{S}$. Hence if $x_j$ is approximately equal to $x$ then the latter is a point of $\mathcal{S}$ itself. Otherwise, if the angle between the line through $x_j$ and $x$ on the one hand and the normal $\eta_j$ (pointing outwards) on the other hand is less than 90 degrees then the point is outside the surface; if the angle is greater than 90 degrees then it is inside the surface.

After augmenting our initial data set by off-surface points, we are now back to a classical interpolation or approximation problem.

## 1.2 Fluid–structure interaction in aeroelasticity

Aereolasticity is the science that studies, among other things, the behavior of an elastic aircraft during flight. This behavior is influenced by the interaction between the deformations of the elastic structure caused by the fluid flow, and the impact that the aerodynamic forces would have on a rigid structural framework. To model these different aspects in a physically correct manner, different models have been developed, adapted to the specific problems.

The related aeroelastic problem can be described in a coupled-field formulation, where the interaction between the fluid and structural models is limited to the exchange of boundary conditions. This *loose* coupling has the advantage that each component of the coupled problem can be handled as an isolated entity. However, the challenging task is to reconcile the benefits of this isolated view with a realistic treatment of the new physical effects arising from the interaction.

Let us make this more precise. Suppose at first that we are interested only in computing the flow field around a given aircraft. This can be modeled mathematically by the Navier–Stokes or the Euler equations, which can be solved numerically using for example a finite-volume code. Such a solver requires a detailed model of the aircraft and its surroundings. In particular, the surface of the aircraft has to be rendered with a very high resolution, as indicated in the right-hand part of Figure 1.3. Let us suppose that our solver has computed a solution, which consists of a velocity field and a pressure distribution. For the time being, we are not interested in the problem of how such a solution can be computed. For us, it is crucial that the pressure distribution creates loads on the aircraft, which might and probably will lead to a deformation. So the next step is to compute the deformation from the loads or forces acting on the aircraft.

Obviously, though, a model having a fine resolution of the surface of the aircraft is not necessary for describing its structure; this might even impede the numerical stability. Hence,

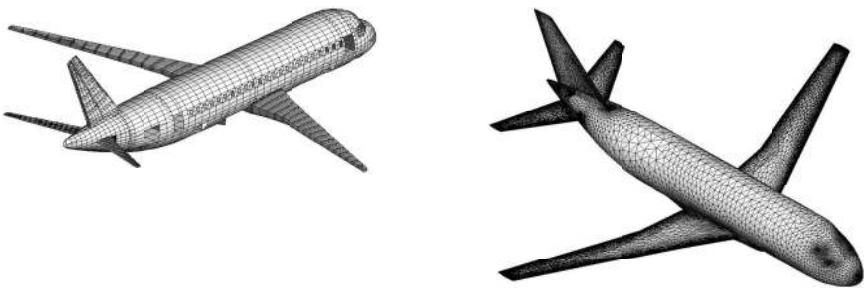

Fig. 1.3 The structural and aerodynamical model of a modern aircraft.

another model is required which is better suited to describing the structural deformation, for example the one shown in Figure 1.3 on the left. Again, along with the model comes a partial differential equation, this time from elasticity theory, which can again be solved, for example by finite elements. But before this can be done, the loads have to be transferred from one mesh to the other in a physically reasonable way. If this has been done and the deformation has been computed then we are confronted with another coupling problem. This time, the deformations have to be transferred from the structural to the aerodynamical model. If all these problems can be solved we can start to iterate the process until we find a steady state, which presumably exists.

Since we have the aerodynamical model, the structural model, and the coupling problem, one usually speaks in this context of a *three-field formulation*. As we said earlier, here we are interested only in the coupling process, which can be described as a scattered data approximation problem, as follows. Suppose that $X$ denotes the nodes of the structural mesh and $Y$ the nodes of the aerodynamical mesh (neither actually has to be a mesh). To transfer the deformations $u(x_j) \in \mathbb{R}^3$ from $X$ to $Y$ we need to find a vector-valued interpolant $s_{u,X}$ satisfying $s_{u,X}(x_j) = u(x_j)$. Then the deformations of $Y$ are given simply by $s_{u,X}(y_j)$, $y_j \in Y$. Conversely, if $f(y_j) \in \mathbb{R}$ denotes the load at $y_j \in Y$ then we need another function $s_{f,Y}$ to interpolate $f$ in $Y$. The loads on the mesh $X$ are again simply given by evaluation at $X$. A few more things have to be said. First of all, if the loads are constant or if the displacements come from a linear transformation, this situation should be recovered exactly, which means that our interpolation process has to be exact for linear polynomials. Furthermore, certain physical entities such as energy and work should be conserved. This means at least that

$$\sum_{y\in Y} f(y) = \sum_{x\in X} s_{f,Y}(x)$$

and

$$\sum_{y\in Y} f(y)s_{u,X}(y) = \sum_{x\in X} s_{f,Y}(x)u(x),$$

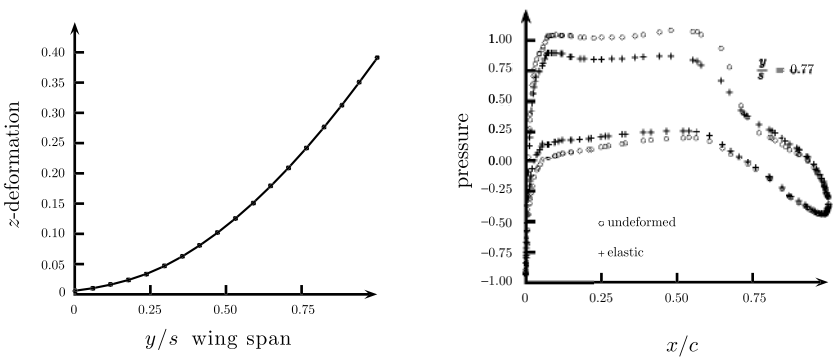


Fig. 1.4 Steady state of the deformed aircraft.

where the last equation is to be taken component-wise. If the models differ too much then both equations have to be understood in a more local sense. However, these equations make it obvious that in certain applications more has to be satisfied than just simple point evaluations. It is important to note that interpolation is crucial in this process since otherwise each coupling step would result in a loss of energy.

The advantage of this *scattered data* approach is that it allows us to couple any two models that have at least some node information. There is no additional information such as the elements or connectivity of the nodes involved. Moreover, the two models can be quite different. It often happens that the boundary of the aerodynamical aircraft has no joint node with the structural model. The latter might even degenerate into a two-dimensional object.

Figure 1.4 shows a typical result for the example from Figure 1.3 based on a speed $M = 0.8$, an angle of attack $\alpha = -0.087^\circ$, and an altitude $h = 10\,498$ meters. On the left the deformation of a wing is shown, while the right-hand graph gives the negative pressure distribution at 77% wing span, for a static and an elastic computation. The difference between the two pressure distributions indicates that elasticity causes a loss of buoyancy, which can become critical for highly flexible structures, as found for example in the case of a large civil aircraft.

It should, be clear that the coupling process described here is not limited to the field of aeroelasticity. It can be applied in any situation where a given problem is decomposed into several subproblems, provided that these subproblems exchange data over specified nodes.

## 1.3 Grid-free semi-Lagrangian advection

In this section we will discuss briefly how the scattered data approximation can be used to solve advection equations. For simplicity, we restrict ourselves here to the two-dimensional case and to the *transport equation*, which is given by

$$0 = \frac{\partial}{\partial t}u(x, y, t) + a_1(x, y)\frac{\partial}{\partial x}u(x, y, t) + a_2(x, y)\frac{\partial}{\partial y}u(x, y, t). \tag{1.1}$$

It describes, for example, the advection of a fluid with velocity field $a = (a_1, a_2)$ and it will serve us as a model problem straight away. Suppose that $(x(t), y(t))$ describes a curve for which the function $\widetilde{u}(t) := u(x(t), y(t), t)$ is constant, i.e. $\widetilde{u}(t) = \text{const}$. Such a curve is called a *characteristic curve* for (1.1). Differentiating $\widetilde{u}$ yields

$$0 = \frac{\partial u}{\partial t} + \dot{x}(t)\frac{\partial u}{\partial x} + \dot{y}(t)\frac{\partial u}{\partial y},$$

where $\dot{x} = dx/dt$. The similarity to (1.1) allows us to formulate the following approximation scheme for solving the transport equation (1.1) with initial data given by a known function $u_0$. Suppose that we know the distribution $u$ at time $t_n$ and at sites $X = \{(x_1, y_1), \dots, (x_N, y_N)\}$ approximately, meaning that we have a vector $u^{(n)} \in \mathbb{R}^N$ with $u_j^{(n)} \approx u(x_j, y_j, t_n)$. To find the values of $u$ at site $(x_j, y_j)$ and time $t_{n+1}$ we first have to find the *upstream point* $(x_j^-, y_j^-)$ with $c := u(x_j^-, y_j^-, t_n) = u(x_j, y_j, t_{n+1})$ and then have to estimate the value $c$ from the values of $u$ at $X$ and time $t_n$. Hence, in the first step we have to solve $N$ ordinary differential equations

$$(\dot{\xi}_j, \dot{\eta}_j) = a(\xi_j, \eta_j), \qquad 1 \le j \le N,$$

with initial value $(\xi(t_{n+1}), \eta(t_{n+1})) = (x_j, y_j)$. The upstream point is the solution at $t_n$, i.e. $(x_j^-, y_j^-) = (\xi(t_n), \eta(t_n))$. Since this point will in general not be contained in $X$, the value $u(x_j^-, y_j^-, t_n)$ has to be estimated from $u^{(n)}$. This can be written as an interpolation problem. We need to find a function $s_u$ that satisfies $s_u(x_j) = u_j^{(n)}$ for $1 \le j \le N$.

The method just described is called a *semi-Lagrangian method*. It is obviously not restricted to a two-dimensional setting. It also applies to advection equations other than the transport problem (even nonlinear ones), but then an interpolatory step might also be necessary when solving the ordinary differential equations.

Moreover, it is not necessary at all to use the same set of sites $X$ in each time step. It is much more appropriate to adapt the set $X$ as required.

Finally, if the concept of scattered data approximation is generalized to allow also functionals other than pure point-evaluation functionals, there are plenty of other possibilities for solving partial differential equations. We will discuss some of them later in this book.

## 1.4 Learning from splines

The previous sections should have given some insight into the application of scattered data interpolation and approximation in the multivariate case.

To derive some concepts, we will now have a closer look at the univariate setting. Hence we will suppose that the data sites are ordered as follows,

$$X : a < x_1 < x_2 < \ldots < x_N < b, \tag{1.2}$$

and that we have certain data values $f_1, \dots, f_N$ to be interpolated at the data sites. In other

words, we are interested in finding a continuous function $s : [a, b] \to \mathbb{R}$ with

$$s(x_j) = f_j, \qquad 1 \le j \le N.$$

At this point it is not necessary that the data values $\{f_j\}$ actually stem from a function $f$, but we will keep this possibility in mind for reasons that will become clear later.

In the univariate case it is well known that $s$ can be chosen to be a polynomial $p$ of degree at most $N - 1$, i.e. $p \in \pi_{N-1}(\mathbb{R})$. Or, more generally, if a *Haar space* $S \subseteq C(\mathbb{R})$ of dimension $N$ is fixed then it is always possible to find a unique interpolant $s \in S$. In this context the space $S$ has the remarkable property that it depends only on the number of points in $X$ and not on any other information about the data sites, let alone about the data values. Thus it would be reasonable to look for such spaces also in higher dimensions. Unfortunately, a famous theorem of Mairhuber and Curtis (Mairhuber [115], see also Chapter 2) states that this is impossible. Thus if working in space dimension $d \ge 2$ it is impossible to fix an $N$-dimensional function space beforehand that is appropriate for *all* sets of $N$ distinct data sites. However, probably no one with any experience in approximation theory would, even in the univariate case, try to interpolate a hundred thousand points with a polynomial.

The bottom line here is that for a successful interpolation scheme in $\mathbb{R}^d$ either conditions on the involved points have to be worked out, in such a way that a stable interpolation with polynomials is still possible, or the function space has to depend on the data sites. The last concept is also well known in the univariate case. It is a well-established fact that a large data set is better dealt with by splines than by polynomials. In contrast to polynomials, the accuracy of the interpolation process using splines is not based on the polynomial degree but on the spacing of the data sites. Let us review briefly properties of univariate splines in the special case of cubic splines. The set of cubic splines corresponding to a decomposition (1.2) is given by

$$S_3(X) = \{s \in C^2[a, b] : s|[x_i, x_{i+1}] \in \pi_3(\mathbb{R}), 0 \le i \le N\}, \tag{1.3}$$

where $x_0 := a$, $x_{N+1} := b$. It consists of all twice differentiable functions that coincide with cubic polynomials on the intervals given by $X$. The space $S_3(X)$ has dimension $\dim(S_3(X)) = N + 4$, so that the interpolation conditions $s(x_i) = f_i$, $1 \le i \le N$, do not suffice to guarantee a unique interpolant. Different strategies are possible to enforce uniqueness and one of these is given by the concept of natural cubic splines. The set of natural cubic splines

$$\mathcal{N}S_3(X) = \{s \in S_3(X) : s|[a, x_1], s|[x_N, b] \in \pi_1(\mathbb{R})\}$$

consists of all cubic splines that are linear polynomials on the outer intervals $[a, x_1]$ and $[x_N, b]$. It is easy to see that a cubic spline $s$ is a natural cubic spline if and only if it satisfies $s''(x_1) = s^{(3)}(x_1) = 0$ and $s''(x_N) = s^{(3)}(x_N) = 0$. Since we have imposed four additional conditions it is natural to assume that the dimension of $\mathcal{N}S_3(X)$ is $\dim(\mathcal{N}S_3(X)) = N$, which is indeed true. Even more, it can be shown that the initial interpolation problem has a unique solution in $\mathcal{N}S_3(X)$. For this and all the other results on splines we refer the reader to Greville's article [75] or to the books by Schumaker [175] and de Boor [43].

This is not the end of the story, however; splines have several important properties and we state some of them for the cubic case.

(1) They are piecewise polynomials.
(2) An interpolating natural cubic spline satisfies a minimal norm property. This can be formulated as follows. Suppose $f$ comes from the Sobolev space $H^2[a,b]$, i.e. $f \in C[a,b]$ has weak first- and second-order derivatives also in $L_2[a,b]$. (We will give a precise definition of this later on). Assume further that $f$ satisfies $f(x_j) = f_j$, $1 \le j \le N$. If $s_{f,X}$ denotes the natural cubic spline interpolant then

$$(f'' - s''_{f,X}, s''_{f,X})_{L_2[a,b]} = 0.$$

This leads immediately to the Pythagorean equation

$$\|f'' - s''_{f,X}\|^2_{L_2[a,b]} + \|s''_{f,X}\|^2_{L_2[a,b]} = \|f''\|^2_{L_2[a,b]},$$

which means that the natural cubic spline interpolant is that function from $H^2[a,b]$ that minimizes the semi-norm $\|f''\|_{L_2[a,b]}$ under the conditions $f(x_j) = f_j$, $1 \le j \le N$.
(3) They possess a local basis (B-splines). These basis functions can be defined in various ways: by recursion, by divided differences, or by convolution.

Of course, this list gives only a few properties of splines. For more information, we refer the interested reader to the previously cited sources on splines.

The most dominant feature of splines, which has contributed most to their success, is that they are piecewise polynomials. This feature together with a local basis not only allows the efficient computation and evaluation of spline functions but also is the key ingredient for a simple error analysis. Hence, the natural way of extending splines to the multivariate setting is based on this property. To this end, a bounded region $\Omega \subseteq \mathbb{R}^d$ is partitioned into essentially disjoint subregions $\{\Omega_j\}_{j=1}^N$. Then the spline space consists simply of those functions $s$ that are piecewise polynomials on each patch $\Omega_j$ and that have smooth connections on the boundaries of two adjacent patches. In two dimensions the most popular partition of a polygonal region is based on a triangulation. Even in this simplest case, however, the dimension of the spline space is in general unknown (see Schumaker [176]). Moreover, when coming to higher dimensions it is not at all clear what an appropriate replacement for the triangulation would be. Hence, even if substantial progress has been made in the two-dimensional setting, the method is not suited for general dimensions. Another possible generalization to the multivariate setting is based on the third property. In particular a construction based on convolution has led to the theory of Box splines (see de Boor *et al.* [44]). Again, even the two-dimensional setting is tough to handle, not to speak of higher-dimensional cases.

Hence, we want to take the second property as the motivation for a framework in higher dimensions. This approach leads to a remarkably beautiful theory, where all space dimensions can be handled in the same way. Since the resulting approximation spaces no longer consist of piecewise polynomials, we do not want to call the functions splines. The buzz phrase, which has become popular in this field, is *radial basis functions*.

To get an idea of radial basis functions let us stick a little longer with natural cubic splines. It is well known that the set $S_3(X)$ has the basis $(\cdot - x_j)_+^3$, $1 \le j \le N$, plus an arbitrary basis for $\pi_3(\mathbb{R})$. Here, $x_+$ takes the value of $x$ for nonnegative $x$ and zero in the other case. Hence, every $s \in \mathcal{N}S_3(X)$ has a representation of the form

$$s(x) = \sum_{j=1}^{N} \alpha_j (x - x_j)_+^3 + \sum_{j=0}^{3} \beta_j x^j, \qquad x \in [a, b]. \tag{1.4}$$

Because $s$ is a natural spline we have the additional information that $s$ is linear on the two outer intervals. On $[a, x_1]$ it has the representation $s(x) = \sum_{j=0}^{3} \beta_j x^j$ so that necessarily $\beta_2 = \beta_3 = 0$. Thus, (1.4) becomes

$$s(x) = \sum_{j=1}^{N} \alpha_j (x - x_j)_+^3 + \beta_0 + \beta_1 x, \qquad x \in [a, b]. \tag{1.5}$$

To derive the representation of $s$ on $[x_N, b]$ we simply have to remove all subscripts $+$ on the functions $(\cdot - x_j)_+^3$ in (1.5). Expanding these cubics and rearranging the sums leads to

$$s(x) = \sum_{\ell=0}^{3} \binom{3}{\ell} (-1)^{3-\ell} \left( \sum_{j=1}^{N} \alpha_j x_j^{3-\ell} \right) x^\ell + \beta_0 + \beta_1 x, \qquad x \in [x_N, b].$$

Thus, for $s$ to be a natural spline, the coefficients of $s$ have to satisfy

$$\sum_{j=1}^{N} \alpha_j = \sum_{j=1}^{N} \alpha_j x_j = 0. \tag{1.6}$$

This is a first characterization of natural cubic splines. But we can do more. Using the identity $x_+^3 = (|x|^3 + x^3)/2$ leads, because of (1.6), to

$$\begin{aligned} s(x) &= \sum_{j=1}^{N} \frac{\alpha_j}{2} |x - x_j|^3 + \sum_{j=1}^{N} \frac{\alpha_j}{2} (x - x_j)^3 + \beta_0 + \beta_1 x \\ &= \sum_{j=1}^{N} \frac{\alpha_j}{2} |x - x_j|^3 + \sum_{\ell=0}^{3} \frac{1}{2} \binom{3}{\ell} (-1)^{3-\ell} \sum_{j=1}^{N} \alpha_j x_j^{3-\ell} x^\ell + \beta_0 + \beta_1 x \\ &= \sum_{j=1}^{N} \widetilde{\alpha}_j |x - x_j|^3 + \widetilde{\beta}_0 + \widetilde{\beta}_1 x, \end{aligned}$$

with $\widetilde{\alpha}_j = \frac{1}{2}\alpha_j$, $1 \le j \le N$, and $\widetilde{\beta}_0 = \beta_0 - \frac{1}{2}\sum \alpha_j x_j^3$, $\widetilde{\beta}_1 = \beta_1 + \frac{3}{2}\sum \alpha_j x_j^2$.

**Proposition 1.1** *Every natural cubic spline $s$ has a representation of the form*

$$s(x) = \sum_{j=1}^{N} \alpha_j \phi(|x - x_j|) + p(x), \qquad x \in \mathbb{R}, \tag{1.7}$$

*where $\phi(r) = r^3$, $r \ge 0$, and $p \in \pi_1(\mathbb{R})$. The coefficients $\{\alpha_j\}$ have to satisfy (1.6). Vice versa, for every set $X = \{x_1, \dots, x_N\} \subseteq \mathbb{R}$ of pairwise distinct points and for every $f \in \mathbb{R}^N$*