Remarks on Meshless Local Construction of Surfaces

Robert Schaback

March 27, 2000

Abstract

This contribution deals with techniques for the construction of surfaces from N given data at irregularly distributed locations. Such methods should ideally have the properties

- computational efficiency,
- smoothness of the resulting surface, if required, and
- quality of reproduction,

but these goals turn out to be hard to meet by a single algorithm. Methods are split into a single construction or precalculation part and subsequent pointwise evaluations. Both parts are analyzed with respect to their complexity. It turns out that one has to expect the main workload on the side of geometric subproblems rather than within numerical techniques. Furthermore, if exact reconstruction at the data locations is required, and if the user wants to avoid solving non–local linear systems, there is no way around localized Langrange–type interpolation formulae. Thus two instances of such techniques are studied in some detail:

- interpolation by weighted local Lagrangians based on radial basis functions and
- moving least squares.

While the former is much more simple than the latter, it still has some deficiencies in theory and practice. Moving least squares, if equipped with certain additional features, turn out to be widely satisfactory, even in difficult cases.

1 Introduction

Ignoring more general and more subtle definitions, we consider surfaces here as sets Y of points $y \in \mathbb{R}^3$ that are either

- *implicitly* represented via an equation $g(y) = 1$ for a scalar function g on $\mathbb{R}^3$ or

- *explicitly* represented as images $y = F(x)$ of a function F defined on a subset Ω of $I\!R^2$ called the *parameter domain*.

Implicit representations have the advantage that one can often define a body with surface Y as the set of points y with $g(y) \leq 1$, while all points with $g(y) > 1$ are "outside". This feature is very convenient for ray tracing algorithms, because one has a quick test for points y on the ray for being inside or outside the body. The transition between implicit and explicit representations of the same surface is a difficult problem that we ignore here. An explicit representation is called *nonparametric*, if the defining map F has the simple form $F(x) = (x, f(x))$ with a scalar function f on Ω.

We focus on the construction of surfaces from given data. These can come in different forms. The most standard case is *quantitative point data* as

- a set $\{y_1, \ldots, y_N\} \subset I\!R^3$ of points on (or near) the surface, or
- a set $X = \{x_1, \ldots, x_N\} \subset \Omega \subseteq I\!R^2$ together with a set $\{y_1, \ldots, y_N\} \subset I\!R^3$ such that $y_j = F(x_j)$ for all $j,\ 1 \leq j \leq N$, either exactly or approximately.

Again, the nonparametric setting specializes to the case $y_j = (x_j, f(x_j)),\ 1 \leq j \leq N$. In general, derivative values can be specified, but we skip over such extensions here. More serious are *qualitative* data like "smoothness", "good shape" or whatever the user may prescribe. Here, we ignore everything except smoothness, and we shall restrict the latter to the classical mathematical definition.

The construction of explicit representations of surfaces from data of the form $y_j = F(x_j)$ can clearly be done by any multivariate vector–valued scattered data interpolation or approximation technique. This will be the main topic of this paper. But before that, and for completeness, we want to point at the specific problems coming up in the case of unstructured data $\{y_1, \ldots, y_N\} \subset I\!R^3$. Imagine 8 data points to be given, forming the vertices of the unit cube in $I\!R^3$. Whatever method is used to find a surface containing these points, there is an intrinsic ambiguity, described by the following possible solutions:

1. A closed, bounded and connected surface, e.g. a sphere,
2. three solutions consisting of two connected components, each picking up the four points of opposite facets, e.g. two parallel planes,
3. twelve different U–shaped solutions formed by picking up the vertices of a chain of three adjacent facets.

This ambiguity even concerns the global topological structure of the solution, and the arising problem is much more serious than finding a numerical method that actually constructs *some* surface containing the data points. For instance, given an arbitrary multivariate scalar–valued interpolation technique for scattered data, one can easily construct a function g on $I\!R^3$ such that $g(y_j) = 1$ for all j. Now, except for certain degenerations, the set of points y with $g(y) = 1$ will define a surface that picks up the given data, but it is not clear how the method behaves in situations like the one above. In principle, it also does not help to do a local triangulation first, because the same problem arises with the triangulation. We leave this interesting area to future research.

2 Construction, Evaluation and Complexity

For the rest of this paper, we focus on constructing explicit representations of surfaces from given data in the form $y_j = F(x_j)$, $1 \leq j \leq N$ for some unknown function F. In many cases, the actual and final evaluation of surface points will not be based on the given data, but rather on some intermediate data needed for the representation. For instance, many CAD packages evaluate surfaces from Bernstein–Bézier control nets, and then these nets form the intermediate data for the representation. We thus split the process in

$$\text{Input data} \overset{Construction}{\longrightarrow} \text{Representation data} \overset{Evaluation}{\longrightarrow} \text{Surface points.}$$

The construction step can contain some data reduction. Typical cases are provided by the lower levels of hierarchical or multilevel schemes for representing surfaces (see [35] for example), or by greedy methods like [32]. We do not consider such methods here. There are also cases where the intermediate data are much larger than the original data. We mentioned an example at the beginning of this section.

The construction step will often be much more complex than the evaluation step, but the evaluation usually has to be performed many times. This is why it is prohibitive to have an $\mathcal{O}(N)$ complexity of evaluation. But if evaluation at a point x is to be done a $\mathcal{O}(1)$ cost and reasonable quality, one needs at least some geometric information about data near to x. This geometric part of the reconstruction process turns out to be much more important than expected, and we conjecture the following:

> In the development of *efficient* techniques for reconstruction of multivariate functions (or surfaces, in particular), the major computational complexity lies within the *geometric* algorithms, not the numerical techniques.

This fact should have been widely recognized in the past, but the scientific focus still is very much on the side of Numerical Analysis than on Computational Geometry. For instance, in any case of univariate spline interpolation, we need for each evaluation point x the smallest knot interval $[x_j, x_{j+1}]$ containing x. This is what we called a "geometric" information above. A naive way getting this information is to use a sorting algorithm at cost $\mathcal{O}(N \log N)$ within the construction step, followed by an $\mathcal{O}(\log N)$ search within each evaluation. The actual numerical construction step via solving a banded system will take $\mathcal{O}(N)$ operations, while the numerical evaluation is of $\mathcal{O}(1)$ complexity for a fixed degree. Our statement is valid already in this simple example, but things will naturally be worse in the multivariate case. This is why we deal with geometric issues in the next section.

To fix our efficiency goals somewhat more precisely, let us look at the relative computational complexity of construction and evaluation of surfaces, provided that N data are given.

- We consider a *construction* technique to be efficient, if it produces $\mathcal{O}(N)$ intermediate data at a computational cost of $\mathcal{O}(N)$ operations for a fixed accuracy requirement. This will rule out precalculations involving triangulations, sorting methods, or full–size linear systems, and it will normally require some additional assumptions on the geometry of the data.
- We consider an *evaluation* technique to be efficient, if it takes $\mathcal{O}(1)$ operations to evaluate the surface at a single point. This rules out all nonlocal methods, methods based on the evaluation of sums with more than $\mathcal{O}(1)$ terms, or methods that require nontrivial search techniques for each evaluation.

The rest of the paper is concentrating on techniques that at least promise to meet these goals, together with the ability to yield surfaces of any prescribed smoothness. The reader will wonder how and why we drop the additional $\log N$ complexity factor that already arises in univariate spline algorithms. But we shall show below that this is justified for "reasonable" data geometries, and in the univariate case it turns out that this is possible whenever there is an upper bound ρ on the mesh ratio

$$\frac{\max\limits_{1\le j<N} |x_{j+1}-x_j|}{\min\limits_{1\le j<N} |x_{j+1}-x_j|}.$$

3 Efficient Geometric Algorithms

If there are no additional assumptions on the data locations, any geometric algorithm with a complexity of $\mathcal{O}(N\log N)$ within the construction step and $\mathcal{O}(\log N)$ for each evaluation must be considered to be efficient, as we are taught by univariate spline theory. But if the set $X=\{x_1,\ldots,x_N\}\subset\Omega\subseteq I\!R^2$ of data locations is not too badly distributed, we hope to get away with $\mathcal{O}(N)$ and $\mathcal{O}(1)$, respectively. The first basic idea is to assume *quasi–uniformity* of the data locations x_j, $1\le j\le N$ on a bounded domain Ω which at least contains the convex hull of the data. This property requires that the quotient of the *fill distance*

$$h := h(X,\Omega) := \max_{y\in\Omega}\min_{x_j\in X} \|y-x_j\|_2 \tag{1}$$

and the *separation distance*

$$q := q(X) := \frac{1}{2} \min_{x_j \neq x_k \in X} \|x_j - x_k\|_2 \le h(X, \Omega)$$

is bounded above by a constant $\rho > 1$.

The second basic idea is to ignore sorting and triangulations in favour of the *k nearest neighbor problem.* The goal is to do some geometric preprocessing at $\mathcal{O}(N)$ cost such that for every given evaluation point x it takes only $\mathcal{O}(k)$ operations to get the k nearest neigbours from the data set X.

The standard folklore recipe, described in d dimensions here, implements a space decomposition technique like those used in Computer Graphics. By a first $\mathcal{O}(N)$ scan over the given N data locations, a bounding box for the whole data set is constructed, defined by maximal and minimal coordinates. Then there are several possible strategies for splitting the global box into $\mathcal{O}(N)$ smaller boxes, hopefully containing only $\mathcal{O}(1)$ data points each.

A standard grid–type decomposition of the global bounding box does the job for quasi–uniform data sets. To see this, let us first prove that h^{-d}, q^{-d}, and N have the same asymptotics for $N \to \infty$. In fact, since each data point has a ball of radius q around it such that the ball does not contain any other data point, these balls are disjoint and the sum of their volumes must be bounded above by a constant. Thus $N = \mathcal{O}(q^{-d})$. On the other hand, the union of the balls of radius h around the data points must cover the domain Ω, and thus the sum of their values is bounded below by a constant, proving $h^{-d} = \mathcal{O}(N)$.

Now let n_B be the maximum number of data points in each box. The balls of radius q around these points will be disjoint and contained in the box of volume $\mathcal{O}(1/N) = \mathcal{O}(q^d)$ plus a surrounding volume that can be bounded by $\mathcal{O}(q^d)$, too. Therefore n_B is bounded above by a constant.

If the data distribution is not quasi–uniform, a decomposition via median splits into a binary tree of boxes will work at the price of $\mathcal{O}(N \log N)$ operations. We prefer the former case and suggest to drop excess points of clusters, keeping the number of points in each grid box at $\mathcal{O}(1)$ by brute force. The treatment of details of surfaces related to data clusters can always be postponed to a second problem, working locally at a finer scale, and having the residuals of the first step as input data. As a byproduct, the above strategy provides a simple "thinning" algorithm along the lines of papers by Floater and Iske [13, 14, 15].

Anyway, it takes $\mathcal{O}(N)$ or $\mathcal{O}(N \log N)$ operations to distribute the given N data into $\mathcal{O}(N)$ boxes with $\mathcal{O}(1)$ points in each box. For any given point x, it takes $\mathcal{O}(1)$ operations to find the box containing x, if the data are quasi–uniform. In the general case, however, one has to go down the binary tree at $\mathcal{O}(\log N)$ cost.

The basic data structure will consist of a list of point indices for each box. The implementation of such a structure can use standard techniques from sparse matrices. To cope with allocation problems, we prefer to use a second scan over all data points that just counts the number of points in each box. Then allocation can be done once and precisely, and the actual placement of points into the correct boxes is done by a third scan over all data points. The overall storage requirement is $\mathcal{O}(N)$.

Since the number of points in each box is $\mathcal{O}(1)$, one can then easily use the data structure to solve nearest neighbour problems for any point x with a complexity of $\mathcal{O}(k)$ (in the quasi–uniform case, or $\mathcal{O}(k \log N)$ in general) for a fixed number k of required neighbours of a point x. The idea is to go into the box of x first and then into all neighbouring boxes with increasing distance, picking up all the data points in those boxes. Whenever one has finished the boxes covering a full ball of radius r around x, one can be sure that at least all neighbours of x at distance at most r are found. The process is stopped if one has found at least k such points, and these are then sorted with respect to their distance to x. No more that $\mathcal{O}(k^d) = \mathcal{O}(1)$ boxes need to be checked in the quasi–uniform case, because the k nearest neighbours cannot be further away from x than $h + 2(k-1)q$.

Note that a univariate simplification of this algorithm allows to sort N real numbers in $\mathcal{O}(N)$ operations, provided that they are quasi–uniformly distributed in a bounded interval. Such algorithms are called "sorting by distribution", and their prototype is the well-known *radix sort.* Furthermore, a subsequent search algorithm can then be implemented at $\mathcal{O}(1)$ cost.

Methods for constructing good triangulations will cost at least $\mathcal{O}(N \log N)$ operations in the two–dimensional case, but they work for general point distributions. Whether they can be reduced to $\mathcal{O}(N)$ computational complexity for quasi–uniform data, is beyond the knowledge of the author.

4 Localization and Oversampling

We now go back to numerical techniques and introduce some more notation. If both the calculation and the evaluation step are linear, one can write

$$\begin{aligned} b_k &= \sum_{\ell=1}^{N} \beta_{k\ell} y_\ell,\ 1 \le k \le M \\ F(x) &= \sum_{k=1}^{M} u_k(x) b_k \end{aligned} \tag{2}$$

with certain evaluation functions u_k and intermediate data b_k, $1 \le k \le M$, starting from input data y_ℓ at x_ℓ for $1 \le \ell \le N$. The number M can be much larger that N, and the index k of intermediate data b_k and basis functions u_k

need not have any relation to the index ℓ of the data. For example, if we define the u_k as basis functions of some finite element space or as Bernstein–Bézier or NURBS basis functions with respect to some representation of the surface by many standard patches, we require the intermediate data to be nodal data for finite elements or to be control points with respect to the various standard surface patches. In such a case, the value of M is much larger than N, and it may be not at all obvious that the constructed surface has sufficient smoothness, unless certain linear equations for the control points b_k are satisfied. We want to ignore the "patching problem" in this contribution, but we shall see later how it arises unexpectedly.

The resulting surface mapping is

$$\begin{aligned} F(x) &= \sum_{k=1}^{M} u_k(x) \sum_{\ell=1}^{N} \beta_{k\ell} y_\ell \\ &= \sum_{\ell=1}^{N} y_\ell \sum_{k=1}^{M} u_k(x)\beta_{k\ell} \\ &= \sum_{\ell=1}^{N} y_\ell L_\ell(x), \end{aligned} \tag{3}$$

and the final form uses Lagrange–type functions

$$L_\ell(x) := \sum_{k=1}^{M} u_k(x)\beta_{k\ell},\ 1 \le \ell \le N$$

that have to satisfy $L_\ell(x_j) = \delta_{j\ell}$ if exact reproduction of the data is required.

In principle, one could fix the evaluation functions u_k beforehand, depending on the final application, and maybe even in a very convenient local form, via finite elements, bicubic spline patches or NURBS. The matrix B with entries $\beta_{k\ell}$ should then be a one–sided inverse to the matrix U with entries $u_k(x_j)$. For $M \ge N$, and if U has full rank N, the determination of such an inverse is possible in theory, but we already mentioned the additional conditions on the intermediate data that will be required to guarantee smoothness, if the u_k are not automatically smooth enough.

At this point, the reader should have understood why we do not want to get into serious trouble with smoothness conditions defined indirectly via additional conditions on the intermediate data. We restrict ourselves to cases where the functions u_k or L_ℓ have the required smoothness, and then we are free to find convenient linear mappings to generate the intermediate data we need.

But there is an important point to be noted at this stage. If we want to make both steps local and carry them out as they are in (2), i.e. without reformulation of the first equation as a linear system for the b_k, the matrices U and B should be sparse, and at the same time be one–sided inverses of each other. This is a very serious obstacle. In general, matrices with a fixed, but irreducible sparsity structure can have full inverses, if perturbations of the matrix entries are allowed ([8], p. 271). This rules out the case $M = N$ except for the standard situation $U = I$ that we analyze below. For $M >> N$ the chances are better, but there are just a few results on such "oversampling" techniques. Roughly speaking, the above argument amounts to the following:

> If locality or sparsity of both the construction and the evaluation process for exact reconstruction of surfaces from parametric data is required, and if both steps are carried out by linear formulae without solving linear systems, one has to do oversampling or to stick to a variation of Lagrange interpolation.

We refrain from casting this guideline into the shape of a theorem, but we shall follow it throughout the paper.

For $M = N$, most of the well–known U matrices (e.g. from splines or radial basis functions) are non–sparse. Note that they are the inverses of the (possibly sparse) matrices of the linear systems for calculating the intermediate data. The pitfall of the above principle is avoided by not using the inverses as linear mappings. Instead, one solves the (possibly sparse) linear systems.

Let us describe a univariate example. Imagine a standard scalar univariate interpolation problem with data $y_j = f(x_j)$, $1 \leq j \leq N$ for nodes $x_1 < x_2 < \ldots < x_N$. We already used this example for pointing at the bulk of work induced by generating the necessary geometric information. We now focus on numerical techniques for construction of intermediate data and evaluation. Linear splines have a Lagrange formulation without any construction step. This follows the above principle via local Lagrange interpolation. Splines of higher degree are usually treated via a nonlocal construction step involving a sparse system with a non–sparse inverse. This follows the principle by resorting to solving a system. There is no local formula that allows circumvention of solving a system in case of $M = N$ and higher–degree splines. This is what the above principle enforces, if neither Lagrange interpolation, nor solving a system, nor oversampling is done. But oversampling can possibly avoid both the nonlocal evaluation and the linear system. In fact, if sufficiently many derivatives at the knots are approximated using the point data by any of the standard techniques, and if piecewise odd–degree Hermite interpolation is done on the oversampled data, we get away without any system, using local construction and evaluation.

The general trick is to use oversampling in such a way that sufficiently many local intermediate data are constructed, such that a subsequent local construction step finds all the data it needs. Finding good multivariate oversampling strategies is a major open problem. But note that the example also shows that we are back to a situation that we did not want to pursue here: the introduction of the “patching problem” through the back door via oversampling. This is easy in the univariate case, but serious in multivariate settings. We close this section with the remark that there may be sparse *approximate* inverses. Examples are in [34]. Transition from interpolation to approximation will thus be another feasible workaround, but note that in this context approximation coincides with quasi–interpolation.

5 Local Lagrange Interpolation

Let us go back to interpolation and consider the simple case $U = I$ implied by Lagrange interpolation on the original data, and look at localized techniques. In such a situation there are no intermediate data, and there is no preprocessing required and no system to be solved. On the downside, we now need Lagrange–type evaluation functions which have a prescribed global smoothness and a cheap $\mathcal{O}(1)$ local evaluation. Such functions do exist, but the race for practically good functions is open. The early Shepard–type techniques were nonlocal, and their localized extensions were nonsmooth. On the other hand, any sufficiently smooth and sufficiently localized peak function u_k which is one at x_k will do the job, but at the price of a useless resulting surface, looking like a bed of nails. The approximation quality comes in as a third criterion, besides smoothness and computational complexity.

But there are simple and cheap methods that do better than localized peaks. A good class of methods with limited smoothness is provided by *natural neighbour coordinates*. Originally due to Sibson [36, 37] as a method yielding a continuous surface, there was an extension by Farin [11] to a continuously differentiable interpolant. If implemented naively, natural neighbour coordinates require a preprocessed Dirichlet tesselation at a cost of at least $\mathcal{O}(N \log N)$, which violates our efficiency goals. If a preprocessing at $\mathcal{O}(N)$ is done for solving the k nearest neighbour problem as described in section 3, one can possibly calculate the natural neighbour coordinates locally within each evaluation step, getting a $\mathcal{O}(1)$ cost per evaluation. But since smoothness is limited, we do not pursue natural neighbour techniques in this paper.

Let us describe a rather general recipe for calculating smooth local interpolants. Around any of the data points $x_j \in \Omega$ we consider a ball $B_r(x_j)$ of some fixed radius $r > 0$. Then we take all points x_k in this ball and construct a local Lagrange function L_j^{loc} with respect to these points by an arbitrary method for local scattered data interpolation, provided that the solution has the required smoothness. Thus we have

$$L_j^{loc}(x_k) = \delta_{jk} \text{ for all } x_k \text{ with } \|x_j - x_k\| < r,\ 1 \le j, k \le N,$$

but we cannot use these functions globally, because they fail to work on far–away points. But there is an easy remedy. Take any nonnegative scalar function w on

$\mathbb{R}$ with $w(0) = 1$ and support $[-r,r]$ such that $w(\|x - x_j\|_2)$ has the required smoothness. Then

$$L_j(x) := L_j^{loc}(x) w(\|x - x_j\|_2) \tag{4}$$

will be a global Lagrange function, and the surface construction can proceed via (3). Note that this recipe allows for a wide range of possible cases, and the contest for good examples is open. We shall show some cases after we have described how to solve the local scattered data interpolation problems.

If the data distribution is quasi–uniform, the calculation of a local Lagrangian function at x_j will require only $\mathcal{O}(1)$ operations. Precalculation of all Lagrangians can be done at $\mathcal{O}(N)$ complexity, and local evaluation at a single point x will only require $\mathcal{O}(1)$ Lagrangians. Thus we have an efficient method in the sense of section 2, independent of the type of local interpolation used. The race for cases with good reproduction qualities is open.

6 Radial Basis Functions

Now it is time to explicitly describe the tools we want to use for local interpolation to scattered data. The presentation can be brief, because there are many survey articles on the subject (in chronological order: [17, 16, 9, 23, 10, 19, 25, 7, 30, 34]). By a fundamental observation of Mairhuber [20], nontrivial spaces for multivariate scattered data interpolation must necessarily depend on the data locations. To make this dependence as simple as possible, one uses functions of the form

$$\begin{array}{rcl} s(x) & = & \displaystyle\sum_{j=1}^{N} \alpha_j \phi(\|x - x_j\|_2) + \sum_{i=1}^{Q} \beta_i p_i(x) \\ 0 & = & \displaystyle\sum_{j=1}^{N} \alpha_j p_i(x_j),\ 1 \le i \le Q \end{array} \tag{5}$$

with a *radial basis function* ϕ on $[0,\infty)$ and a basis $p_1,\dots,p_Q$ of the space $\mathbb{P}_m$ of bivariate polynomials of degree up to $m-1$, where $Q = m(m+1)/2$. The function ϕ and the number m are related by the requirement that ϕ must be (strictly) conditionally positive definite of some order $m_0 \le m$, and this property makes sure that the systems

$$\begin{array}{rcl} s(x_k) & = & \displaystyle\sum_{j=1}^{N} \alpha_j \phi(\|x_k - x_j\|_2) + \sum_{i=1}^{Q} \beta_i p_i(x_k) = y_k,\ 1 \le k \le N \\ 0 & = & \displaystyle\sum_{j=1}^{N} \alpha_j p_i(x_j),\ 1 \le i \le Q \end{array} \tag{6}$$

arising for arbitrary scattered data problems are uniquely solvable, if there is no nontrivial polynomial in $\mathbb{P}_m, m \ge m_0$, that vanishes at all data locations

$x_1, \dots, x_N$. The coefficients α_j and β_i are scalars in the case of nonparametric data, and vectors in the general case. We shall ignore this in the sequel, restricting ourselves to the scalar case without loss of generality.

The most prominent examples of radial basis functions are

$$\begin{array}{rcll}
\phi(r) & = & r^\beta, \quad \beta > 0, \beta \notin 2I\!N_0, & m_0 = \lceil \beta/2 \rceil \\
\phi(r) & = & r^{2k}\log(r), \quad k \in I\!N \quad \text{(thin-plate splines)} & m_0 = k+1 \\
\phi(r) & = & (c^2+r^2)^\beta, \quad \beta < 0, \quad \text{(inverse multiquadrics)} & m_0 = 0 \\
\phi(r) & = & (c^2+r^2)^\beta, \quad \beta > 0, \beta \notin I\!N_0 \quad \text{(multiquadrics)} & m_0 = \lceil \beta \rceil \\
\phi(r) & = & e^{-\alpha r^2}, \quad \alpha > 0 \quad \text{(Gaussians)} & m_0 = 0 \\
\phi(r) & = & (1-r)_+^4(1+4r) & m_0 = 0
\end{array}$$

together with their orders m_0 of conditional positive definiteness. A comprehensive presentation of these functions together with full proofs of their fundamental properties is in [33].

Note that in the context of section 2 we have representation data consisting of two vectors $\alpha \in I\!R^N$, $\beta \in I\!R^Q$, which already is a weak form of oversampling in case of $Q > 0$. But, except for trivial choices of scaling, the system (6) has no sparse inverse, even if a compactly supported function like $(1-r)_+^4(1+4r)$ is used. The latter function is C^2 on R^2 when written as a radial function of two variables, and it can act as a weight function in (4). Other reasonable weight functions are

$$w_{\delta,k}(r) = \left\{ \begin{array}{ccl} 1 & r & \le 1-\delta \\ \delta^{-2k}(1-r)^k(r-(1-2\delta))^k & 1-\delta < r & \le 1 \\ 0 & r & > 1 \end{array} \right. \tag{7}$$

for $\delta \in (0,1)$ and $k > 0$, yielding prescribed degrees of smoothness.

7 Global Interpolation by Radial Basis Functions

We do not consider global solutions of large–scale scattered data interpolation problems in detail here. For completeness, we only point out the two current lines of research and mark their fundamental differences. The starting point is the behaviour of radial basis function interpolants with respect to scaling. It is a standard technique, arising already in finite elements and being the background of the convergence theory initiated by Strang and Fix [38], to scale the interpolants in a way that is proportional to the data density, using "narrow" basis functions for dense data and "wide" basis functions for coarse data. For historical reasons this is called a *stationary* setting, while the *nonstationary* case uses the same radial basis function for all possible interpolation problems, irrespective of the data density.

Let us first look at computational issues. In the stationary setting, the arising matrices will have a condition that is basically independent of the data density. For compactly supported basis functions, the sparsity structure is fixed and the evaluation of approximants will be cheap due to localization. In the nonstationary setting the condition will dramatically increase when the data get dense, because rows and columns of the system matrix tend to be more and more similar. Furthermore, sparse matrices arising from compactly supported radial basis functions get filled up, and the complexity of evaluation increases.

But the situation is different, if we look at approximation properties. In the nonstationary setting, all radial basis functions have good approximation properties which are closely related to the numerical condition: the better the approximation properties, the worse the condition [29]. On the other hand, in the stationary case there is no convergence for interpolation problems based on integrable radial basis functions [6], while thin–plate splines and multiquadrics show good approximation properties. But the latter do not share the advantage of the stationary setting with respect to the matrix structure: the systems will always be non–sparse.

Thus there is no fully satisfying way out, if users look at problems on varying scales. Using the stationary setting with global radial basis functions like thin–plate splines, powers or multiquadrics will cause no convergence problems, but the user is forced to add strategies for dealing with large full matrices and a costly evaluation process. The groups around M.J.D. Powell [26, 28] and R. Beatson [5, 3, 4, 2] have made great progress in this direction. A second approach uses compactly supported radial basis functions and exploits sparsity as much as possible. If fill–in is to be limited, one is bound to a stationary setting, but then there are problems getting good approximation quality, because there is no convergence in theory. As long as the data are not too dense, the stationary technique improves with data density, but there is a small final error level that cannot be improved by adding more data. This phenomenon was called *approximate approximation* my Mazỳa and Schmidt [21], and it deserves further study. The approximation quality of the final level is mainly determined by the admitted amount of fill–in [31], but the natural way out of this is to go over to multiscale techniques [12, 13, 22] applying the steps of a stationary setting recursively to residuals. This is quite successful, but still needs theoretical work. First steps are in [18].

The method of section 5, using local weighted Lagrangians, avoids solving large systems and guarantees locality without using compactly supported basis functions. Its properties will be discussed in the next section.

8 Local Weighted Interpolation by Radial Basis Functions

Let us now look at some specific cases, implementing the cut–off Lagrangian technique of section 5 via local interpolants based on radial basis functions from section 6. For ease of publication in printed form, we confine ourselves here to simple 2D graphics and present much more sophisticated 3D images at the conference. It is a rather convenient rule–of–thumb to use about 50 local data around each evaluation point, and thus we start in Figure 1 with presenting a one–dimensional cross–section of the Lagrangian calculated via thin–plate splines and linear polynomials for 49 local neigbours on a two–dimensional grid. These 49 neighbors are within a circle of radius 0.5 on a grid with spacing 1/8, and thus the cross section of the Lagrange function along an axis has 8 symmetric zeros in $[-0.5, 0.5]$, being regularly distributed at distance 1/8, if zero is added. To see the behaviour outside $[-0.5, 0.5]$, we replaced the values inside by zero to get the second plot in Figure 1. The outside peaks have a maximum height of 0.000717, and this is a coarse upper bound of the relative deviation between the global and local Lagrangian. This unexpected behaviour of thin–plate spline Lagrangians was first observed by Powell [24]. The decay for arguments tending to infinity is exponential, and thus a weighted cutoff does no serious harm. In our figures, we have not yet multiplied the local Lagrangian with a weight function, but we prefer to use weights that are equal to 1 for most of $[0, 1]$, because otherwise the peak of the Lagrangian gets too sharp. A good strategy for the Lagrangian based on 49 points was (7) for $\delta = 0.1$.

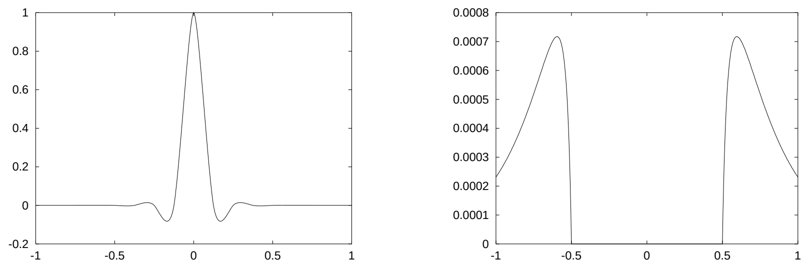

Figure 1: Local Lagrange Function for Thin–plate Splines

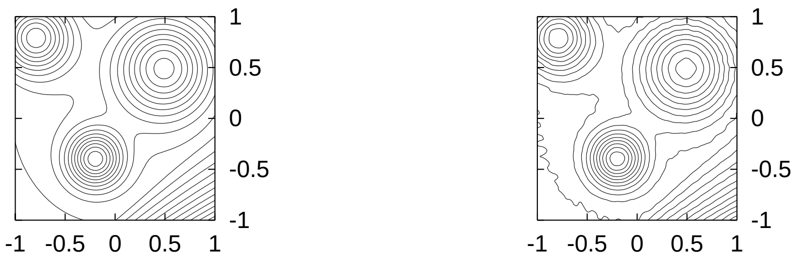

Figure 2: Contour Plots

More sophisticated examples reveal that at high graphical resolutions the smooth cut–off induced by the weight function shows up considerably, though it is quantitatively of a small order of magnitude. Thus there is quite some work to be done on methods of this kind. An example is provided by Figure 2, where contours of the reproduced Franke–type surface, plotted at high resolution on the right–hand side, get rough in comparison to the original function on the left.

We now want to focus on the reproduction quality and start with the remark that the classical error bounds for radial basis function interpolation in the nonstationary setting are local. This is not directly stated in the literature, but can be read between the lines of the various proof techniques, e.g. [40, 27]. In principle, if the fill distance $h := h(X, \Omega)$ of (1) is small enough, and if local reconstruction is to be done at some point $x \in \Omega$, one can confine the local interpolant to data at points x_j with $\|x - x_j\|_2 \leq ch$ with a suitable constant $c > 1$. Thus the number of locally required data points can be bounded independent of h for reasonably distributed data sets, but due to the nonstationary setting the condition will not be bounded above. However, the numerically feasible range is much larger than in global problems. In cases that are scale invariant (powers and thin–plate splines), there is no difference between the stationary and nonstationary settings, and then the local systems have no serious stability problems.

But, unfortunately, there is a subtle difference to the techniques of section 5. In local radial basis function interpolation as described by the localized standard convergence analysis, the selection of data points depends on a selection of nearest neighbours of the evaluation point x, while in section 5 we used precomputed selections based on neighbours of each x_k. The local Lagrangians of the two cases will not be comparable, and the proof of local convergence orders does not cover the situation of section 5. Furthermore, the local interpolant in case of radial basis functions is a true linear combination of the $\phi(\|x - x_j\|_2)$ with $\|x - x_j\|_2 \leq ch$, while in case of section 5 such functions are multiplied with the weight function. Thus, unfortunately, there is no easy way to carry standard results on radial basis functions over to this situation.

9 Fully Local Methods with Polynomial Reproduction

We now look generally at localized techniques in the sense of the previous section. We depart from radial basis functions for a while and describe a folklore argument proving m–th order of convergence for stable local methods with local reproduction of polynomials of order up to m. At a point $x \in \Omega$ we want to take only a subset $X(x) := \{x_j \in X \ : \ \|x - x_j\|_2 \leq ch\} \subseteq X$ of the data set x with fill distance h as in (1), where $c > 1$ is a constant. We simply assume that we have a linear local process at x that is based on data $X(x)$ and that locally reproduces polynomials of order at most m. In particular, we keep x fixed and write

$$R_f(x) := \sum_{x_j \in X(x)} f(x_j) u_j(x) \tag{8}$$

with certain real numbers $u_j(x)$ such that

$$R_p(x) = p(x)$$

holds for all polynomials p up to order m. Note that the reproduction of polynomials is confined to the single point x. Now we assume that f has continuous derivatives up to order m around x, and thus the Taylor expansion $T_{x,f,m}$ of f at x of order at most m satisfies

$$|f(x_j) - T_{x,f,m}(x_j)| \leq Ch^m$$

for all $x_j \in X(x)$, where C depends on c and the derivatives of f near x. Now we can bound the local error via

$$\begin{array}{rcl} |f(x) - R_f(x)| & = & |T_{x,f,m}(x) - R_f(x)| \\ & = & |R_{T_{x,f,m}}(x) - R_f(x)| \\ & = & \left| \displaystyle\sum_{x_j \in X(x)} \left(T_{x,f,m}(x_j) - f(x_j)\right) u_j(x) \right| \\ & \leq & Ch^m \displaystyle\sum_{x_j \in X(x)} |u_j(x)|, \end{array}$$

and we see that the "Lebesgue constants"

$$L(x) := \sum_{x_j \in X(x)} |u_j(x)|$$

should be bounded independent of h, which is the stability condition we mentioned at the outset.

Let us look at simple examples first. For $m = 1$, we can get local reproduction of constants by always picking the function value at the nearest neighbor. The Lebesgue constant is 1. If f is continuously differentiable on Ω, and because any $x \in \Omega$ has a nearest neighbour from X at distance at most h, we get a method of order 1. The reconstruction is piecewise constant on the Dirichlet tesselation induced by X, though the tesselation is never actually calculated. For $m = 2$ and if Ω is the convex hull of X, we can use barycentric coordinates with respect to triangles containing x, taking the data at the vertices. Any triangulation of Ω via X will then lead to a piecewise linear and continous reconstruction by linear finite elements. The Lebesgue constant is 1 again.

Natural neighbour interpolation is another case fitting into this framework, The original version by Sibson [36, 37] is continuous and reproduces linear polynomials with Lebesgue constant 1, while the C^1 extension by Farin [11] even reproduces quadratic polynomials.

Of course, the general approach above can be combined with radial basis function techniques and a sufficiently large order m of polynomial reproduction. By an argument in [29], the quantity

$$\sum_{x_j \in X(x)} u_j^2(x)$$

can be bounded above in all relevant cases, even in the nonstationary situation. This is not precisely what we require for the above line of argumentation, but if the local data sets $X(x)$ consist of $\mathcal{O}(1)$ points, which is what we can assume for quasi–uniform data distributions, the Lebesgue constants are uniformly bounded. However, it always has to assume that the local data do not allow a vanishing nontrivial polynomial of order m, and under this assumption one can go back to m–th order polynomials right away. This is why we do not pursue this setting any further.

Here is a little digression. One is tempted to consider the optimization problem

$$\begin{array}{rcl}\displaystyle\sum_{j=1}^N |u_j(x)| & = & \text{Minimum}\\ \displaystyle\sum_{j=1}^N u_j(x)x_j^\alpha & = & x^\alpha,\ 0 \le |\alpha| < m\end{array}$$

to hope for a reasonable method with *automatic* localization near x. The standard split of the variables $u_j(x) = u_j^+(x) - u_j^-(x)$ into nonnegative parts leads to a linear programming problem of simple form. But due to reproduction of constants via

$$1 = \sum_{j=1}^{N} (u_j^+(x) - u_j^-(x)),$$

we have

$$\sum_{j=1}^{N} u_j^+(x) \geq 1$$

and the objective function always satisfies

$$\text{Minimum} = \sum_{j=1}^{N} (u_j^+(x) + u_j^-(x)) \geq 1.$$

Thus for $m = 2$ all cases with interpolation via local barycentric coordinates in a triangle containing x will be optimal, irrespective of the size or position of the triangle. There is no automatic selection of local neighbours via this optimization problem.

Things are even worse when the point x is outside the convex hull of the data. If the problem is solvable at all, linear programming tells us that there always is an optimal solution based on three points for $m = 2$, and the solution must be determined by barycentric coordinates again, at least one of which must now be negative. The optimum is attained for choices of triangles where the sum of negative barycentric coordinates is minimal in absolute value. Closer inspection reveals that those optimal triangles are geometrically awful, because negative barycentric coordinates of a point x outside a triangle are small in absolute value, if the vertices “antipodal” to x are far away from x.

Similarly bad results are obtained if we replace the L_1 objective function by L_2 or L_∞, and we conclude that optimal stability does not imply locality, finishing our digression.

For upsampling of gridded data, there are simple and useful folklore formulae obtainable via the arguments of this section. For linear precision, upsampling at the midpoint of edges or at the center of a square should use the arithmetic mean of the data values. Again, we have Lebesgue constants bounded by 1, and the process will be of second order in terms of the meshwidth. Of course, such a process yields the bilinear local interpolant when started on four values at the vertices of a square and repeated indefinitely. Note that though the order is 2 for data from C^2 functions or surfaces, the resulting function or surface will not be C^2. Schemes with quadratic precision in two variables should use 6 points in general. A simple recipe can be obtained from looking at quadratic polynomials in Bernstein–Bézier representation, but the result will not yield a smooth surface.

10 Moving Least Squares

The examples above had the disadvantage that they generate surfaces with little smoothness, because the local schemes depend on the evaluation point x and the point selection $X(x)$ in a nontrivial and possibly noncontinuous way. We now look at a general recipe that overcomes this drawback and allows arbitrary smoothness and approximation order, at least in theory.

For a fixed evaluation point $x \in \Omega$ we consider the weighted least–squares problem

$$\text{Minimize} \sum_{j=1}^{N} (f(x_j) - p(x_j))^2 \, \phi(\|x - x_j\|_2)$$

over all polynomials $p \in I\!P_m$. Here, the weight function is a smooth nonnegative radial basis function ϕ with compact support, and this is how the above problem turns out to be localized. The resulting process, if well–defined, will reproduce polynomials up to order m, but we still have to write it in the form (8) and show that the functions u_j come out to be smooth.

Since the resulting linear system has a right–hand side that is a linear function of the data $f(x_j)$, we get (8) without further arguments, but we have to find a representation of the $u_j(x)$. To this end, we introduce self–explanatory matrix notation to write the objective function as $\|D_x f - D_x A a\|_2^2$ with a diagonal $N \times N$ matrix D_x having entries $\sqrt{\phi(\|x - x_j\|_2)}$ and an $N \times Q$ matrix A with entries $p_k(x_j)$ for a basis $p_1, \ldots, p_Q$ of $I\!P_m$. The solution vector $a_x \in I\!R^Q$ with respect to the data vector $f = (f(x_1), \ldots, f(x_N))^T$ is uniquely determined by the system

$$A^T D_x D_x A a_x = A^T D_x D_x f,$$

provided that the coefficient matrix $A^T D_x D_x A$ has full rank $Q \leq N$. We assume this for a moment, and we proceed to construct a vector $u(x) \in I\!R^N$ such that for $p(x) := (p_1(x), \ldots, p_Q(x))^T$ we can write $R(x) := a_x^T p(x) = u(x)^T f$. This is easy, if we look at

$$\begin{array}{rcl} A^T D_x D_x A v(x) & = & p(x) \\ u(x) & = & D_x D_x A v(x) \end{array} \tag{9}$$

and solve the first system for $v(x)$, putting the solution into the second equation. Thus we get $A^T u(x) = p(x)$ for free, which is the polynomial reproduction property at x. The entries of $A^T D_x D_x A$ are

$$\sum_{i=1}^{N} \phi(\|x - x_i\|_2) p_j(x_i) p_k(x_i),$$

and the matrix has full rank, if we define

$$X(x) := \{x_j \in X \ : \ \phi(\|x - x_i\|_2) > 0\}$$

and assume that there is no nontrivial polynomial in $\mathbb{P}_m$ that vanishes on $X(x)$. One can see clearly how the weight function localizes the least–squares problem if it is of compact support, but the support must be large enough to host at least a set of points near x that are in general position with respect to $\mathbb{P}_m$.

Since we can write the reconstruction in the form $R(x) = u^T(x)f$ without taking care of the localization explicitly, we see from the system (9) that the smoothness of the overall approximation is completely determined by the smoothness of the weight function. Thus we are left with the highly nontrivial problem of bounding the Lebesgue constants. A thorough treatment of this, giving all constants in explicit form, is due to Wendland [39]. Thus moving least squares are a technique that satisfies all requirements: it is effective in the sense of section 2, and it can produce surfaces with any prescribed smoothness. However, in its standard form it is an approximation rather than an interpolation.

One of the main computational problems of moving least squares is the proper determination of the local point selection $X(x)$. In particular, there may be great variations in the data density, and these variations should be flexibly incorporated into the algorithm. We propose to use all data points in a ball with varying radius around the evaluation point x, i.e.

$$X(x) := X \cap B_{\delta(x)}(x) := \{x_j \in X \ : \ \|x - x_j\|_2 \leq \delta(x)\}$$

where δ is a smooth function that is calculated beforehand, preferably by another moving least squares appproximation. For instance, one can generate $\mathcal{O}(N)$ regularly distributed points $y_1, \ldots, y_N$ in the domain Ω and find a “good calculation radius” δ_j for $X(y_j) := B_{\delta_j}(x)$ for each of these points. Then $\delta(x)$ is constructed via an intermediate moving least squares algorithm, and the result is inserted into the actual surface construction technique.

We finish the paper with examples provided by R. Baule [1], illustrating the use of a varying calculation radius. We pick the glacier data ($N = 8345$) from R. Franke’s website `http://www.math.nps.navy.mil/~rfranke/`, because it has a very inhomogeneous data distribution (see Figure 3). The main problem of any reconstruction method is to produce good results where the data are scarce, while keeping a good overall reproduction quality of the data. Naive and direct application of moving least squares can either result in a staircase or an overdose of smoothing (see Figures 4,5 and the examples from [39]). If the calculation radius varies as in Figure 6, one gets the much more realistic results of Figures 7 and 8. In fact, the L_∞ error on the data goes down from 81 to 21 when variable radii are used. A further variation, not described here in detail, includes interpolation via infinite weights, and then we get the same visual appearance as in Figure 8, but with zero error on the data.

Acknowledgement

Special thanks go to Rainer Baule and Holger Wendland for proofreading.

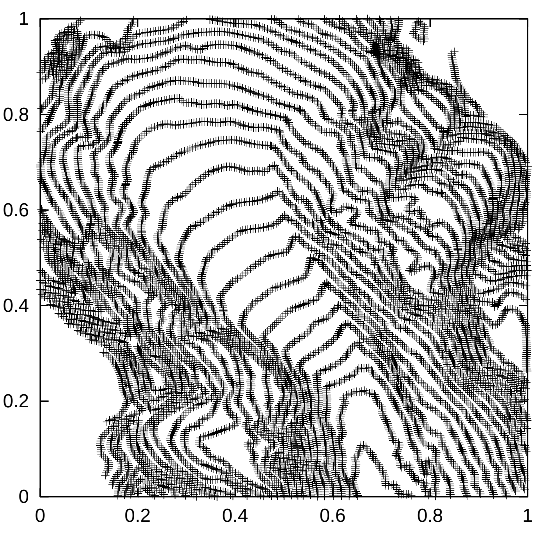

Figure 3: Glacier data

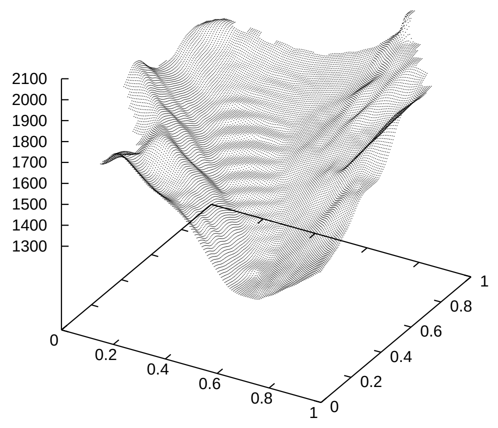

Figure 4: Moving least squares with fixed radius 0.06

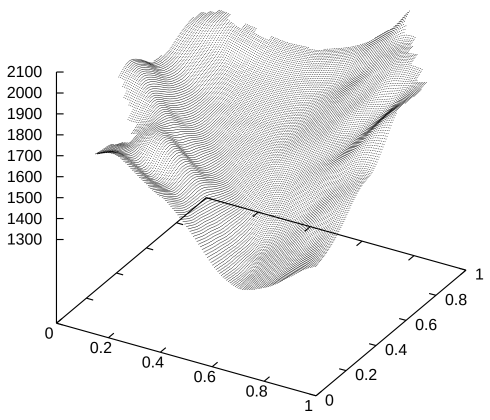

Figure 5: Moving least squares with fixed radius 0.12

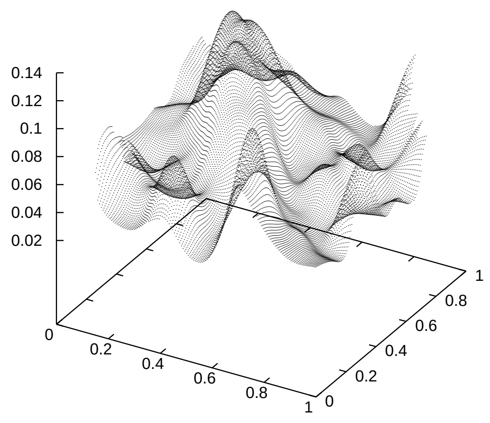

Figure 6: Variable radius used in Figure 7

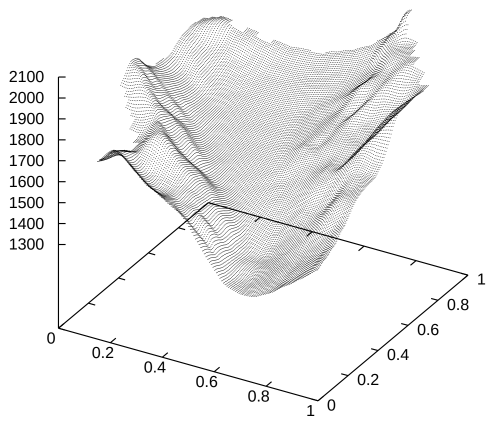

Figure 7: Moving least squares with variable radius, degree zero

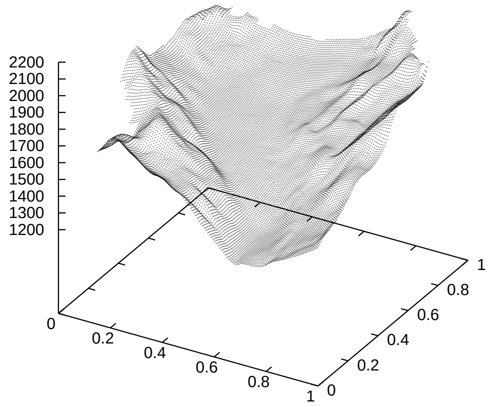

Figure 8: Moving least squares with variable radius, degree 2

References

[1] R. Baule. Moving Least Squares Approximation mit parameterabhängigen Gewichtsfunktionen. Diplomarbeit, Universität Göttingen, 2000.

[2] R.K. Beatson, J.B. Cherrie, and C.T. Mouat. Fast fitting of radial basis functions: Methods based on preconditioned GMRES iteration. *Advances in Computational Mathematics*, 11:253–270, 1999.

[3] R.K. Beatson and L. Greengard. A short course on fast multipole methods. In M. Ainsworth, J. Levesley, W. Light, and M. Marletta, editors, *Wavelets, Multilevel Methods and Elliptic PDEs*, pages 1–37. Oxford University Press, 1997.

[4] R.K. Beatson and W.A. Light. Fast evaluation of radial basis functions: Methods for two–dimensional polyharmonic splines. *IMA Journal of Numerical Analysis*, 17:343–372, 1997.

[5] R.K. Beatson and G.N. Newsam. Fast evaluation of radial basis functions: I. Advances in the theory and applications of radial basis functions. *Comput. Math. Appl.*, 24(12):7–19, 1992.

[6] M.D. Buhmann. *Multivariable Interpolation using Radial Basis Functions.* PhD thesis, University of Cambridge, 1989.

[7] M.D. Buhmann. New developments in the theory of radial basis function interpolation. In K. Jetter and F.I. Utreras, editors, *Multivariate Approximations: From CAGD to Wavelets*, pages 35–75. World Scientific, Singapore, 1993.

[8] Iain S. Duff, Albert M. Erisman, and John K. Reid. *Direct Methods for Sparse Matrices.* Monographs on Numerical Analysis. Clarendon Press, Oxford, 1986.

[9] N. Dyn. Interpolation of scattered data by radial functions. In C.K. Chui, L.L. Schumaker, and F.I. Utreras, editors, *Topics in Multivariate Approximation*, pages 47–61. Academic Press, Boston, 1987.

[10] N. Dyn. Interpolation and approximation by radial and related functions. In C.K. Chui, L.L. Schumaker, and J.D. Ward, editors, *Approximation Theory VI*, volume 1, pages 211–234. Academic Press, New York, 1989.

[11] G. Farin. Surfaces over Dirichlet tessellations. *Comput. Aided Geom. Design*, 7:281–292, 1990.

[12] M.S. Floater and A. Iske. Multistep scattered data interpolation using compactly supported radial basis functions. *J. Comp. Appl. Math.*, 73:65–78, 1996.

[13] M.S. Floater and A. Iske. Thinning and approximation of large sets of scattered data. In F. Fontanella, K. Jetter, and P.-J. Laurent, editors, *Advanced Topics in Multivariate Approximation*, pages 87–96. World Scientific Publishing, Singapore, 1996.

[14] M.S. Floater and A. Iske. Thinning, inserting, and swapping scattered data. In A. Le Mehaute, C. Rabut, and L. L. Schumaker, editors, *Surface Fitting and Multiresolution Methods*, pages 139–144. Vanderbilt University Press, Nashville, 1996.

[15] M.S. Floater and A. Iske. Thinning algorithms for scattered data interpolation. *BIT*, 38:4:705–720, 1998.

[16] R. Franke. Scattered data interpolation. Test of some methods. *Mathematics of Computation*, pages 181–200, 1982.

[17] R.L. Hardy. Multiquadric equations of topography and other irregular surfaces. *J. Geophys. Res.*, 76:1905–1915, 1971.

[18] S. Hartmann. *Multilevel-Fehlerabschätzung bei der Interpolation mit radialen Basisfunktionen.* Dissertation, Universität Göttingen, 1998.

[19] W. Light. Some aspects of radial basis function approximation. In S.P. Singh, editor, *Approximation Theory, Spline Functions and Applications*, volume 356, pages 163–190. Kluwer, 1992.

[20] J. C. Mairhuber. On Haar's theorem concerning Chebychev approximation problems having unique solutions. *Proc. Amer. Math. Soc.*, 7:609–615, 1956.

[21] V. Maz'ya and G. Schmidt. On approximate approximations using Gaussian kernels. *IMA Journal of Numerical Analysis*, 16:13–29, 1996.

[22] F. J. Narcowich, R. Schaback, and J. D. Ward. Multilevel interpolation and approximation. *Appl. Comput. Harmonic Anal.*, 7:243–261, 1999.

[23] M.J.D. Powell. Radial basis functions for multivariable interpolation: a review. In D.F. Griffiths and G.A. Watson, editors, *Numer. Algorithms*, pages 223–241. Longman Scientific & Technical (Harlow), 1987.

[24] M.J.D. Powell. Tabulation of thin plate splines on a very fine two–dimensional grid. In *Numerical Methods of Approximation Theory*, pages 221–244. Birkhäuser, Basel, 1992.

[25] M.J.D. Powell. The theory of radial basis function approximation in 1990. In W.A. Light, editor, *Advances in Numerical Analysis II: Wavelets, Subdivision Algorithms, and Radial Basis Functions*, pages 105–210. Oxford Univ. Press, Oxford, 1992.

[26] M.J.D. Powell. Truncated Laurent expansions for the fast evaluation of thin plate splines. *Numer. Algorithms*, 5(1–4):99–120, 1993.

[27] M.J.D. Powell. The uniform convergence of thin–plate spline interpolation in two dimensions. *Numer. Math.*, 67:107–128, 1994.

[28] M.J.D. Powell. A review of algorithms for thin plate spline interpolation in two dimensions. In F. Fontanella, K. Jetter, and P.-J. Laurent, editors, *Advanced Topics in Multivariate Approximation*, pages 303–322. World Scientific Publishing, 1996.

[29] R. Schaback. Error estimates for approximations from control nets. *Comput. Aided Geom. Design*, 10:57–66, 1993.

[30] R. Schaback. Multivariate interpolation and approximation by translates of a basis function. In Charles K. Chui and L.L. Schumaker, editors, *Approximation Theory VIII*, volume 1: Approximation and interpolation, pages 491–514. World Scientific Publishing, 1995.

[31] R. Schaback. On the efficiency of interpolation by radial basis functions. In A. LeMéhauté, C. Rabut, and L.L. Schumaker, editors, *Surface Fitting and Multiresolution Methods*, pages 309–318. Vanderbilt University Press, Nashville, TN, 1997.

[32] R. Schaback and H. Wendland. Adaptive greedy techniques for approximate solution of large RBF systems. manuscript, Göttingen, 2000.

[33] R. Schaback and H. Wendland. Characterization and construction of radial basis functions. In N. Dyn, D. Leviatan, and D Levin, editors, *Eilat proceedings*. Cambridge University Press, 2000.

[34] R. Schaback and H. Wendland. Numerical techniques based on radial basis functions. In Albert Cohen, Christophe Rabut, and Larry Schumaker, editors, *Curve and Surface Fitting*. Vanderbilt University Press, Nashville, TN, 2000.

[35] H.-P. Seidel. Hierarchical methods in computer graphics. In László Szirmay Kalos, editor, *14th Spring Conference on Computer Graphics*, pages 1–16. Comenius University, Bratislava, Slovakia, 1998.

[36] R. Sibson. A vector identity for the Dirichlet tesselation. *Mathematical Proceedings of the Cambridge Philosophical Society*, 87:151–155, 1980.

[37] R. Sibson. A brief description of natural neighbor interpolation. In V. Barnett, editor, *In Interpreting Multivariate Data*, pages 21–36. John Wiley, Chichester, 1981.

[38] G. Strang and G. Fix. A Fourier analysis of the finite element variational method. In G. Geymonat, editor, *Constructive Aspects of Functional Analysis*, C.I.M.E. II Ciclo 1971, pages 793–840. ???, 1973.

[39] H. Wendland. Local polynomial reproduction and moving least squares approximation. to appear in IMA Journal of Numerical Analysis, 2000.

[40] Z. Wu and R. Schaback. Local error estimates for radial basis function interpolation of scattered data. *IMA Journal of Numerical Analysis*, 13:13–27, 1993.