Multilevel Scattered Data Approximation by Adaptive Domain Decomposition

Armin Iske and Jeremy Levesley

Abstract. A new multilevel approximation scheme for scattered data is proposed. The scheme relies on an adaptive domain decomposition strategy using quadtree techniques (and their higher-dimensional generalizations). It is shown in the numerical examples that the new method achieves an improvement on the approximation quality of previous well-established multilevel interpolation schemes.

1 Introduction

Multivariate scattered data approximation requires the recovery of an unknown function $f : \mathbb{R}^d \to \mathbb{R}$ from given function values $f(x_1), \ldots, f(x_N) \in \mathbb{R}$ sampled at a finite set $X = \{x_1, \ldots, x_N\} \subset \mathbb{R}^d$ of pairwise distinct locations. Especially in situations where the number N is extremely large, and the points in X are unevenly distributed, multilevel approximation schemes are appropriate tools [3, 5].

The starting point of the scheme in [3] is a decomposition of the data into a hierarchy

$$X_1 \subset X_2 \subset \cdots \subset X_{L-1} \subset X_L = X \tag{1}$$

of nested subsets. The data hierarchy (1) is *a priori* computed in [3] by using *thinning algorithms*, recursive point removal schemes, as discussed in more detail in [4]. In a subsequent *synthesis* of the data, a sequence $s_1, \ldots, s_L$ of approximations to f is then recursively computed by the following multilevel interpolation scheme.

Let $s_0 \equiv 0$. For $j = 1, \ldots, L$, compute an interpolant $\Delta s_j : \mathbb{R}^d \to \mathbb{R}$ to the residual $f - s_{j-1}$ on X_j. Then let $s_j = s_{j-1} + \Delta s_j$. Altogether, the following L interpolation problems are to be solved one after the other:

$$\begin{array}{rcl@{\qquad}rcl} f|_{X_1} &=& \Delta s_1|_{X_1}; & s_1 &=& \Delta s_1; \\ (f - s_1)|_{X_2} &=& \Delta s_2|_{X_2}; & s_2 &=& s_1 + \Delta s_2; \\ \vdots & & & \vdots & & \\ (f - s_{L-1})|_{X_L} &=& \Delta s_L|_{X_L}; & s_L &=& s_{L-1} + \Delta s_L. \end{array} \tag{2}$$

Note that every function s_j in (2) matches f on the subset X_j, i.e.,

$$s_j|_{X_j} = f|_{X_j} \quad \text{for all } 1 \le j \le L. \tag{3}$$

In the multilevel scheme proposed in [3], and later improved in [5], *radial basis functions* were used for solving the interpolation problems in (2). Radial basis functions are popular tools for multivariate scattered data interpolation and in applications [1, 6, 11, 12].

In (2), we make use of *polyharmonic spline* interpolation at the coarsest level in order to compute the initial interpolant s_1. In this case, the interpolant $s_1 : \mathbb{R}^d \to \mathbb{R}$ in (2) has the form

$$s_1 = \sum_{x \in X_1} c_x \phi_{d,k}(\| \cdot - x\|) + p, \tag{4}$$

where the polyharmonic spline $\phi_{d,k} : [0, \infty) \to \mathbb{R}$ is given by

$$\phi_{d,k}(r) = \left\{ \begin{array}{ll} r^{2k-d} \log(r), & \text{for } d \text{ even}, \\ r^{2k-d}, & \text{for } d \text{ odd}. \end{array} \right\}, \quad 2k > d, \tag{5}$$

and where $p : \mathbb{R}^d \to \mathbb{R}$ is a function in Π^d_{k-1}, the linear space of all d-variate polynomials of degree at most $k-1$. Moreover, the unknown coefficients c_x in (4) annihilate the polynomials in Π^d_{k-1}, i.e.,

$$\sum_{x \in X_1} c_x q(x) = 0, \qquad \text{for all } q \in \Pi^d_{k-1}.$$

Application of the interpolation conditions $f|_{X_1} = s_1|_{X_1}$ in (2), and using the above side conditions, give a square system which is uniquely solvable as long as the points in X_1 are *unisolvent* with respect to Π^d_{k-1}, i.e., there is no non-trivial polynomial in Π^d_{k-1} vanishing on all points in X_1.

Then, at the levels $j = 2, \ldots, L$, we work with *compactly supported* radial basis functions. In this case, for a fixed compactly supported positive definite radial function $\phi : [0, \infty) \to \mathbb{R}$ of support radius $r = 1$, each interpolant Δs_j, $2 \le j \le L$, in (2), has the form

$$\Delta s_j = \sum_{x \in X_j} c_x \phi_{\varrho_j}(\| \cdot - x\|), \qquad 2 \le j \le L, \tag{6}$$

where, by putting $\phi_{\varrho_j} = \phi(\cdot/\varrho_j)$, $\varrho_j > 0$, the monotonically decreasing sequence of numbers ϱ_j are the basis function's support radii at the L different levels.

As discussed in [5], the method's performance relies heavily on the *quality* of the data hierarchy (1). In particular, in the selection of the coarsest set X_1, special attention has to be paid to the approximation quality of the

initial interpolant s_1. For the purpose of obtaining good approximation quality, much of the effort in [5, 7] has been spent on the selection of a suitable sequence (1). To this end, a recursive filtering scheme has been designed.

Rather than further tuning the quality of the nested sequence in (1), in this paper we drop the restriction of requiring the sets in (1) to be *subsets* of X. Instead of this, we prefer to work with a data hierarchy of the form

$$C_1 \subset C_2 \subset \cdots \subset C_{L-1} \subset C_L, \tag{7}$$

where each subset C_j, $1 \leq j \leq L$, represents a collection of *clusters* in X. But, in contrast to the multilevel schemes in [3, 5], the representing sets C_j in (7) do not necessarily need to be subsets of X. This amounts to working with multilevel *approximation* rather than multilevel *interpolation*. Therefore, in this new setting, each subset X_j in (2) and (3) is replaced by a corresponding C_j in (7). The construction of the data hierarchy in (7) is the subject of the discussion in Sections 2 and 3.

Compared with the results in the previous papers [3, 5], the proposed multilevel scheme yields significantly better approximation quality. This will be confirmed in the numerical examples in Section 4 by using a real-world example from terrain modelling.

2 Adaptive Domain Decomposition

Let $\Omega \subset \mathbb{R}^d$ be a *bounding box* for X, i.e., Ω is a hypercuboid in $\mathbb{R}^d$ containing the point set X. In what follows, we intend to decompose Ω into a collection $\{\omega\}_{\omega \in \mathcal{L}}$ of smaller *cells* of different sizes and satisfying

$$\Omega = \bigcup_{\omega \in \mathcal{L}} \omega. \tag{8}$$

By letting $X_\omega = X \cap \omega$, for $\omega \in \mathcal{L}$, this yields a partition $\{X_\omega\}_{\omega \in \mathcal{L}}$ of X into accordingly many point clusters. The decomposition of Ω (and thus the partition of X) will be computed by recursively splitting the cells, with Ω being the initial cell; see the following Subsection 2.1. The decision on the splitting of a single cell will be made according to two different customized splitting criteria, to be explained in Subsections 2.2 and 3.1.

2.1 Generalized Quadtrees

For the purpose of computing and maintaining a decomposition of Ω into a collection of cells, we make use of the well-known data structure *quadtree*

(for the special case of two dimensions, where $d = 2$) and its generalizations in higher dimensions, e.g. *octtree* for $d = 3$. Recall that a quadtree is a tree, where each of its nodes has either four *children* or none. A node with no children is called a *leaf*. For a fixed space dimension $d \geq 2$, in a *generalized quadtree* each node has either 2^d children or none. For the sake of simplicity, we will from now use the expression *quadtree* throughout this text rather than *generalized quadtree*.

The nodes in the quadtree are the cells. Initially, we let (the cell) Ω be the root of the quadtree. At this stage, Ω is the only cell in the tree and thus is a leaf. Furthermore, let $\mathcal{L}$ denote the set of leaves of the quadtree. Now a leaf

$$\omega = [x_{\ell_1}, x_{r_1}] \times \cdots \times [x_{\ell_d}, x_{r_d}] \subset \Omega \subset \mathbb{R}^d \tag{9}$$

of the quadtree is split by first computing its decomposition into 2^d subcells $\omega_{\mathbf{i}} \subset \omega$, $\mathbf{i} = (i_1, \ldots, i_d) \in \{0, 1\}^d$, of equal size, each being of the form

$$\omega_{\mathbf{i}} = [x_{\ell_1} + i_1\Delta_1, x_{m_1} + i_1\Delta_1] \times \cdots \times [x_{\ell_d} + i_d\Delta_d, x_{m_d} + i_d\Delta_d], \tag{10}$$

where $x_\omega = (x_{m_1}, \ldots, x_{m_d})^T \in \mathbb{R}^d$ is the centre of ω and $\Delta_j = (x_{r_j} - x_{\ell_j})/2$, $1 \leq j \leq d$, is half the length of a corresponding edge of ω.

The quadtree is then modified by attaching the new leaves $\omega_{\mathbf{i}}$, $\mathbf{i} \in \{0, 1\}^d$, to its *parent* cell ω, i.e., the cells $\{\omega_{\mathbf{i}}\}$ are the 2^d children of ω in the quadtree, and thus ω is no longer a leaf. Finally, the set $\mathcal{L}$ is updated by letting

$$\mathcal{L} = (\mathcal{L} \setminus \omega) \bigcup_{\mathbf{i} \in \{0,1\}^d} \omega_{\mathbf{i}}.$$

Note that each of the leaves $\omega_{\mathbf{i}}$ has the form (9), so that the splitting of any $\omega \in \mathcal{L}$ by using (10) is well-defined at any stage of the recursion.

For the special case of two dimensions, by the splitting of one $\omega \in \mathcal{L}$ we obtain the four subcells

$$\begin{aligned} \omega_{00} &= [x_\ell, x_m] \times [y_\ell, y_m], & \omega_{01} &= [x_\ell, x_m] \times [y_m, y_r], \\ \omega_{10} &= [x_m, x_r] \times [y_\ell, y_m], & \omega_{11} &= [x_m, x_r] \times [y_m, y_r], \end{aligned}$$

where $x_\omega = (x_m, y_m) = ((x_\ell + x_r)/2, (y_\ell + y_r)/2)$ is the centre of ω, see Figure 1. Note that one could also consider non-uniform splittings of ω, but we want to avoid long, thin cells.

2.2 Splitting of Cells at the Coarsest Level

Initially, the cells are split according to the spatial distribution of the points in X. Our aim is to construct the coarsest set C_1 in (7). To this end, a

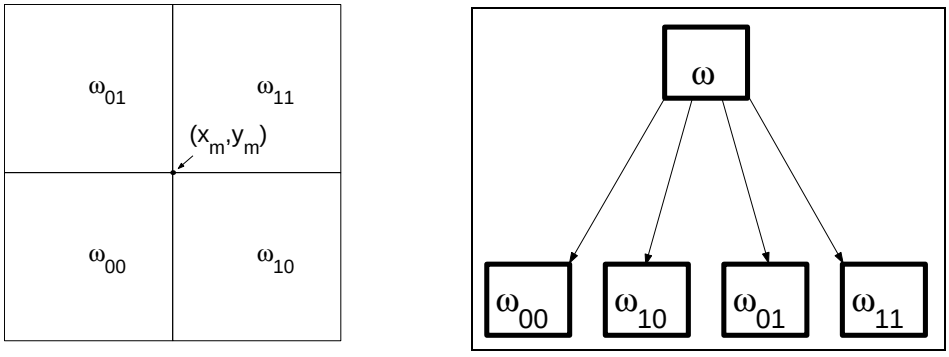


Figure 1: The cell $\omega \subset \mathbb{R}^2$ is split into four subcells $\omega_{00}, \omega_{01}, \omega_{10}, \omega_{11}$ **(a)**, and the quadtree is updated accordingly **(b)**.

decomposition (8) of Ω is computed, such that the number $|X_\omega|$ of points in each resulting point cluster X_ω is not greater than a predetermined number $M \ll |X| = N$.

Definition 1 *Let M be given. We say that a leaf $\omega \in \mathcal{L}$ is* `splittable`*, iff the size $|X_\omega|$ of X_ω is greater than M, i.e., $|X_\omega| > M$.*

Having specified the definition for a splittable leaf, the recursive domain decomposition of Ω is accomplished as follows.

Algorithm 1 (Domain Decomposition) *Let the point set X and a bounding box Ω for $X \subset \Omega$ be given. Then, Ω is decomposed into a collection of cells as follows.*

(1) Let Ω be the root of the quadtree, and let $\mathcal{L} = \{\Omega\}$.

(2) **WHILE (** *$\mathcal{L}$ contains a splittable leaf* **)**

(2a) Locate a splittable leaf $\omega \in \mathcal{L}$;

(2b) Split $\omega \in \mathcal{L}$ and obtain the 2^d new leaves $\{\omega_{\mathbf{i}} : \mathbf{i} \in \{0,1\}^d\}$;

(2c) Update the quadtree and let $\mathcal{L} = (\mathcal{L} \setminus \omega) \bigcup_{\mathbf{i} \in \{0,1\}^d} \omega_{\mathbf{i}}$.

(3) **OUTPUT:** *Cell collection $\{\omega\}_{\omega \in \mathcal{L}}$ satisfying (8).*

Note that the above Algorithm 1 yields a partition $\{X_\omega\}_{\omega\in\mathcal{L}}$ of X, where each cluster X_ω contains not more than M points. Figure 3 shows an example in two dimensions with $|X| = 23,092$ data points given, and where $M = 60$ was selected. Algorithm 1 computes a decomposition of Ω into 841 cells, shown in Figure 3 **(a)**. This decomposition is used in order to define the coarsest set C_1 in (7). To this end, we let

$$C_1 = \{x_\omega : \omega \in \mathcal{L} \text{ and } X_\omega \text{ not empty}\} \tag{11}$$

be the union of all cell centres whose cells are not empty, and so any $x_\omega \in C_1$ represents the point cluster $X_\omega \subset X$. In the situation of our example in Figure 3, one cell is empty. Therefore, the resulting set C_1, displayed in Figure 3 **(b)**, comprises $|C_1| = 840$ cell centres.

3 Multilevel Approximation

In this section, the details of the proposed multilevel approximation scheme are explained. The algorithm is a modification of the multilevel interpolation scheme (2). In contrast to the corresponding schemes in [3, 5], the data hierarchy (1) is adaptively constructed at run time.

The starting point of our multilevel approximation scheme is the coarsest point set C_1 in (11). But the points in C_1 do not have any function values, yet. To this end, we assign to each $x_\omega \in C_1$ a *cell average value* $s_\omega(x_\omega)$ of f on ω, where s_ω is the polyharmonic spline interpolant of the form (4) satisfying $s_\omega|_{X_\omega} = f|_{X_\omega}$. In other words, we consider solving the interpolation problem

$$s_\omega(\xi) = f(\xi), \qquad \text{for all } \xi \in X_\omega, \tag{12}$$

by using polyharmonic splines.

We need to make a few comments concerning the numerical stability of local polyharmonic spline interpolation. We first remark that the interpolation problem (12) is ill-posed, whenever the point set X_ω is not unisolvent with respect to the polynomials Π^d_{k-1}. In this situation, for the sake of numerical stability, we prefer to compute the *mean value*

$$\frac{1}{|X_\omega|}\sum_{\xi\in X_\omega} f(\xi)$$

rather than solving (12) by polyharmonic spline interpolation.

Moreover, we remark that the linear system resulting from (12) may be ill-conditioned, even if the interpolation problem (12) is itself well-conditioned.

To be more precise, due to Narcowich and Ward [9], the spectral condition number of the resulting coefficient matrix is bounded above by a monotonically decreasing function of the interpolation points' *separation distance*

$$q_{X_\omega} = \min_{\substack{\xi,\eta \in X_\omega \\ \xi \neq \eta}} \|\xi - \eta\|.$$

This in turn implies that one should, for the sake of numerical stability, avoid solving the linear system resulting from (12) directly in situations where the minimal distance q_{X_ω} between two points in X_ω is small. For further details on this, see [9] and the more general discussion in [10].

The previous paper [8], however, offers a numerically stable algorithm for the evaluation of the interpolant s_ω in (12). The algorithm works with a rescaling of the interpolation points X_ω, so that their separation distance is q_{X_ω} increased. The evaluation scheme in [8] can be viewed as a simple way of preconditioning of the linear system resulting from (12). The construction of this particular preconditioner relies on the scale-invariance of the interpolation scheme's *Lebesgue constant*.

The latter leads to further consequences concerning the *local approximation order* of polyharmonic spline interpolation, which in turn has impact on the approximation quality of the interpolant s_ω in (12). In order to briefly explain the relevant result from [8], we remark that the approximation order of local polyharmonic spline interpolation, with using $\phi_{d,k}$ in (5), is k for C^k-functions. To be more precise, we obtain for any target function f, which is in C^k locally around x_ω, the asymptotic bound

$$|s^h_\omega(x_\omega + h(x - x_\omega)) - f(x_\omega + h(x - x_\omega)| = \mathcal{O}(h^k), \quad h \to 0, \tag{13}$$

where s^h_ω denotes for $h > 0$ the unique polyharmonic spline interpolant satisfying the (scaled) interpolation conditions

$$s^h_\omega(x_\omega + h(\xi - x_\omega)) = f(x_\omega + h(\xi - x_\omega)) \qquad \text{for all } \xi \in X_\omega.$$

For further details concerning the analysis for the asymptotic bound in (13), we refer to the paper [8].

Now let us return to the discussion of multilevel approximation. Having computed the cell average values $s_\omega(x_\omega)$, the interpolation on C_1 by s_1 at the first level of the multilevel scheme (2) is well-defined.

3.1 Adaptive Splitting of Cells at Finer Levels

As to the construction of the subsequent point sets C_j, $j > 1$, in (7), we make use of the approximation behaviour of the current approximation s_j.

More precisely, at each level j, we evaluate the local approximation quality of s_j at the current decomposition $\{\omega\}_{\omega\in\mathcal{L}}$ by computing the error

$$\eta_\omega = \max_{x\in X_\omega} |f(x) - s_j(x)| \tag{14}$$

for every cell $\omega \in \mathcal{L}$. If, however, X_ω happens to be the empty set, we let $\eta_\omega = 0$.

On the basis of the error indicators $\{\eta_\omega\}_{\omega\in\mathcal{L}}$, further leaves of the quadtree will be split as follows. First, the cells are sorted according to their errors η_ω. Then, for a predetermined number $m \equiv m_j$, we split the m cells whose errors (at level j) are largest.

Altogether, this leads us to the following recursion for the adaptive cell splitting at level $j > 1$, where we initially (i.e. when $j = 2$) assume that the subset C_1 in (7) and the approximations of f at the cell centres in C_1 have already been computed as explained above. Moreover, the availability of the initial domain decomposition $\mathcal{L}_1 = \{\omega\}_{\omega\in\mathcal{L}}$, computed by Algorithm 1, is assumed.

Algorithm 2 (Multilevel Approximation) *For level $j \geq 1$, let a decomposition $\mathcal{L}_j = \{\omega\}_{\omega\in\mathcal{L}}$ of Ω, the subset C_j in (7) and the approximation s_j of f in (2) satisfying $s_j|_{C_j} = f|_{C_j}$ be given. Then, the next subset C_{j+1} in (7), the next finer decomposition $\mathcal{L}_{j+1}$ of Ω, and the approximation s_{j+1} in (2) are computed as follows.*

(1) For a predetermined m, split the m cells in $\mathcal{L}_j$ whose error η_ω in (14) are largest, and so obtain the new decomposition $\mathcal{L}_{j+1} = \{\omega\}_{\omega\in\mathcal{L}}$ of Ω.

(2) Let $C_{j+1} = C_j \cup \{x_\omega : \omega \in \mathcal{L}_{j+1}\}$.

(3) For each $\omega \in \mathcal{L}_{j+1}$, assign an approximation $\Delta s_\omega(x_\omega)$ to the cell centre x_ω by evaluating the polyharmonic spline interpolant Δs_ω satisfying $\Delta s_\omega|_{X_\omega} = (f - s_j)|_{X_\omega}$ at the cell centre x_ω.

(4) For some support radius ϱ_{j+1}, compute the interpolant Δs_{j+1} of the form (6) satisfying $\Delta s_{j+1}|_{C_{j+1}} = (f - s_j)|_{C_{j+1}}$.

(5) Let $s_{j+1} = s_j + \Delta s_{j+1}$.

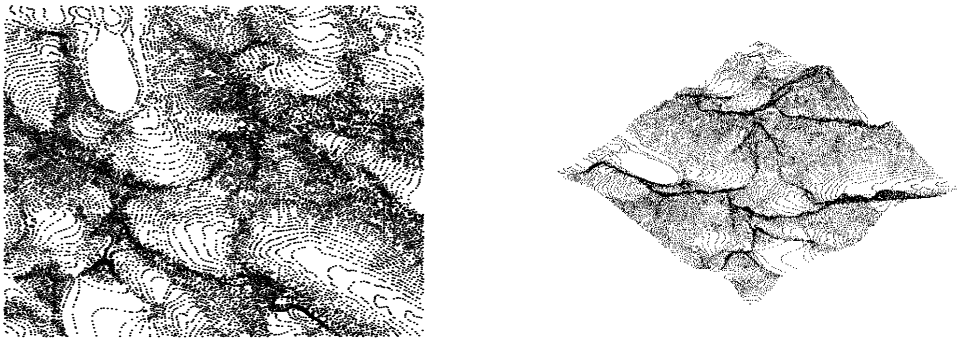
4 Numerical Results

We have implemented the proposed multilevel approximation scheme for *arbitrary* space dimension d. For the pupose of illustration, however, this

section shows numerical results for the special case of two dimensions, where $d = 2$, and so we prefer to work with *thin plate spline* interpolation, i.e., $k = 2$ in (4).

In our numerical examples, we decided to work with one real-world data set from terrain modelling. This particular data set, called *Hurrungane*, is displayed in Figure 2 **(a)** (2D view), and **(b)** (3D view). The data is a sample of height values $\{f(x)\}_{x\in X}$ taken at $|X| = 23,092$ distinct geographic locations of a Norwegian mountain area, where the rectangular domain is given by $\Omega = [437000, 442000] \times [6812000, 6817000]$. The minimum height of this data set is $\min_{x\in X} f(x) = 1100$ meters, and the maximum height is $\max_{x\in X} f(x) = 2400$ meters above sea-level.

Now we compare the performance of our scheme, **QT**, against two very recent ones, [3, 5] (multilevel interpolation using scattered data filtering), **SF**, and [2] (multilevel interpolation using adaptive thinning), **AT**. For the purpose of illustration, it is sufficient to work with just two levels, and so $L = 2$.

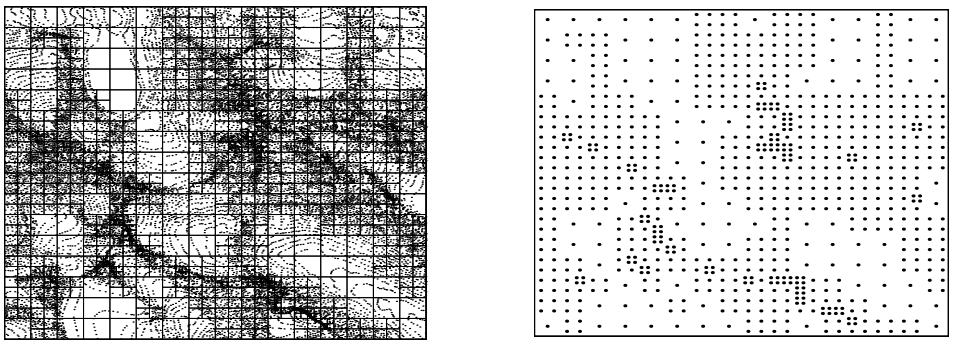


(a) **(b)**

Figure 2: The data set Hurrungane, 2D view **(a)**, and 3D view **(b)**.

Figure 3 **(a)** shows the initial decomposition of Ω into 841 cells, generated by using Algorithm 1. This splitting yields the coarsest subset C_1 in (7), displayed in Figure 3 **(b)**. We found that one cell is empty, and so the size of the subset C_1 is $n_1 = |C_1| = 840$.

Next, we split leaves in the quadtree according to the magnitude of their aforementioned *cell error* in (14). The splitting is done by using Algorithm 2. We have chosen $m = 419$ in step (1) of Algorithm 2, so that the splitting yields $4 \times 419 = 1676$ new leaves, and thus 1676 new cell centres. But 21 out of these 1676 new leaves were empty. The union of the cell centres, belonging



(a) **(b)**

Figure 3: Hurrungane: Partition of the domain into 841 cells **(a)**, and the centres of the leaves **(b)**, yielding the subset C_1 in (7) of size $|C_1| = 840$.

to the $1676 - 21 = 1655$ *non-empty* leaves, and the $n_1 = |C_1| = 840$ points in the initial set C_1 yields, according to step (2) in Algorithm 2, the subsequent subset C_2 in (7) of size $n_2 = |C_2| = 840 + 1655 = 2495$. The two subsets C_1 and C_2 are displayed in Figure 4 **(a)** and **(b)**.

For the purpose of comparison, we have generated two nested subsets X_1, X_2 in (1) of the same size, i.e., $|X_1| = n_1$ and $|X_2| = n_2$, by using the two alternative methods **SF** and **AT**. The two subsets X_1 and X_2 output by these two different methods are also displayed in Figure 4, **(c)** and **(d)** (generated by **SF**), **(e)** and **(f)** (generated by **AT**).

Then, we have used the subsets C_1, C_2 (generated by **QT**) and X_1, X_2 (generated by **SF** and **AT**) for computing the two corresponding interpolants s_1 and s_2 in (2) for each of the three different methods. Recall that s_1 in (4) is the polyharmonic spline interpolant of the data on the set C_1, when using the method **QT**, or on X_1, when using **SF**, **AT**. Moreover, $s_2 = s_1 + \Delta s_2$, where Δs_2 in (6) is the interpolant of the resulting residual $f - s_1$ on the set C_2 (or on X_2 in (2) when working with either **SF** or **AT**). In either case, we have selected $\varrho_2 = 250.0$ for the support radius of the compactly supported radial basis function ϕ_{ϱ_2} in (6), where we let $\phi(r) = (1-r)_+^4(4r+1)$. For details on the construction of compactly supported radial basis functions, see [13]. In our numerical experiments, we have also considered using polyharmonic spline interpolation at level $j = 2$ in (2) in order to compute Δs_2 of the form (4), satisfying $(f - s_1)|_{X_2} = \Delta s_2|_{X_2}$. The approximation quality of the resulting interpolant s_2 turned out to be about as good as when working with compactly supported radial basis

Method	n_1	$\eta_\infty^{(1)}$	$\eta_2^{(1)}$	n_2	ϱ_2	$\eta_\infty^{(2)}$	$\eta_2^{(2)}$
QT	840	153.410	21.822	2495	250.0	77.682	11.550
SF	840	226.030	31.863	2495	250.0	134.069	16.164
AT	840	341.381	33.998	2495	250.0	229.369	23.156

Table 1: Comparison of the three different methods **QT**, **SF**, and **AT**: Approximation quality of the interpolants s_1 and s_2.

functions. For the purpose of making a fair comparison between the three different methods **QT**, **SF** and **AT**, especially on the basis of the previous results in [3, 5], we stick to using compactly supported radial basis functions at level $j = 2$, as originally suggested in [3].

Now, for evaluating the approximation quality of s_1 and s_2 we have recorded both the L_∞-error

$$\eta_\infty^{(j)} = \max_{x \in X} |f(x) - s_j(x)|, \qquad j = 1, 2,$$

and the (discrete) L_2-error

$$\eta_2^{(j)} = \left(\frac{1}{|X|} \sum_{x \in X} |f(x) - s_j(x)|^2 \right)^{1/2}, \qquad j = 1, 2,$$

for each of the three different methods. Table 1 shows our results.

Given the numerical results in Table 1, the method **QT** is the best, followed by **SF** and **AT**. We remark that the method **SF** is very similar to the one proposed in [3]. Indeed, the method in [3] works with a data hierarchy (1) of *uniformly distributed* subsets. This results in the fairly good behaviour of the initial approximation s_1 for **SF**. In contrast to this, the method **AT** works with unevenly distributed subsets, which leads to undesirable overshoots of s_1 near the clusters in X_1 output by **AT** (corresponding to the ridges of the mountains; see Figure 4 **(e)**). This explains why the method **AT** is inferior to **SF** already at level $j = 1$, with a much larger L_∞-error $\eta_\infty^{(1)}$ than **SF**. The poor approximation quality of **AT** at the initial level cannot be recovered by the subsequent interpolant s_2. This is also supported by the numerical results in [5].

In conclusion, the performance of multilevel interpolation relies heavily on the data hierarchy (1). Moreover, as also shown in [5], the approximation quality of the initial interpolant s_1 has a strong effect on the approximation

quality of subsequent interpolants. Now note that, when compared with **SF** and **AT**, the approximation quality of the method **QT** is much better at the initial level. Indeed, the method **QT** reduces the approximation errors $\eta_\infty^{(1)}$ and $\eta_2^{(1)}$ of the method **SF** by approximately a third. This is due to the well-balanced distribution of the points in the coarse set C_1; see Figure 4 **(a)**. The distribution of the points in C_1 is not as clustered as in the set X_1 output by **AT**; see Figure 4 **(e)**. This helps to avoid the abovementioned overshoots of the initial interpolant s_1. The method **QT** continues to be superior to both **SF** and **AT** at the coarse level $j = 2$. This agrees with our above explanation concerning the corresponding comparison between **SF** and **AT**.

In summary, the adaptive multilevel approximation scheme **QT** yields a good alternative to previous multilevel interpolation schemes. The good performance of the method **QT** is mainly due to the sophisticated construction of the subsets C_1 and C_2. On the one hand, unlike **SF**, this construction is *data-dependent*. On the other hand, in contrast to **AT**, dense clusters are avoided, and this helps to damp down possible overshoots of the interpolants s_j, $j = 1, 2$.

Acknowledgment

The first author was partially supported by EPSRC grant GR/R07769. The data set *Hurrungane* was provided by the Norwegian Mapping Authority.

References

[1] M.D. Buhmann: Radial basis functions. *Acta Numerica*, 2000, 1–38.

[2] N. Dyn, M.S. Floater, and A. Iske: Adaptive thinning for bivariate scattered data. *J. Comput. Appl. Math.* **145**(2), 2002, 505–517.

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

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

[5] A. Iske: Hierarchical scattered data filtering for multilevel interpolation schemes. *Mathematical Methods for Curves and Surfaces: Oslo 2000*, T. Lyche and L. L. Schumaker (eds.), Vanderbilt University Press, Nashville, 2001, 211–220.

[6] A. Iske: Scattered data modelling using radial basis functions. *Tutorials on Multiresolution in Geometric Modelling*, A. Iske, E. Quak, M.S. Floater (eds.), Springer-Verlag, Heidelberg, 2002, 205–242.

[7] A. Iske: Progressive scattered data filtering. *J. Comput. Appl. Math.* **158**(2), 2003, 297–316.

[8] A. Iske: On the approximation order and numerical stability of local Lagrange interpolation by polyharmonic splines. *Modern Developments in Multivariate Approximation*, W. Haussmann, K. Jetter, M. Reimer, J. Stöckler (eds.), ISNM 145, Birkhäuser, Basel, 153–165.

[9] F.J. Narcowich and J.D. Ward: Norm estimates for the inverses of a general class of scattered-data radial-function interpolation matrices. J. Approx. Theory **69**, 1992, 84–109.

[10] R. Schaback: Stability of radial basis function interpolants. *Approximation Theory X: Wavelets, Splines, and Applications*, C.K. Chui, L.L. Schumaker, and J. Stöckler (eds.), Vanderbilt Univ. Press, Nashville, 2002, 433–440.

[11] R. Schaback: Creating surfaces from scattered data using radial basis functions. *Mathematical Methods for Curves and Surfaces*, M. Dæhlen, T. Lyche, and L.L. Schumaker (eds.), Vanderbilt University Press, Nashville, 1995, 477–496.

[12] R. Schaback: Multivariate interpolation and approximation by translates of a basis function. *Approximation Theory VIII, Vol. 1: Approximation and Interpolation*, C.K. Chui and L.L. Schumaker (eds.), World Scientific, Singapore, 1995, 491–514.

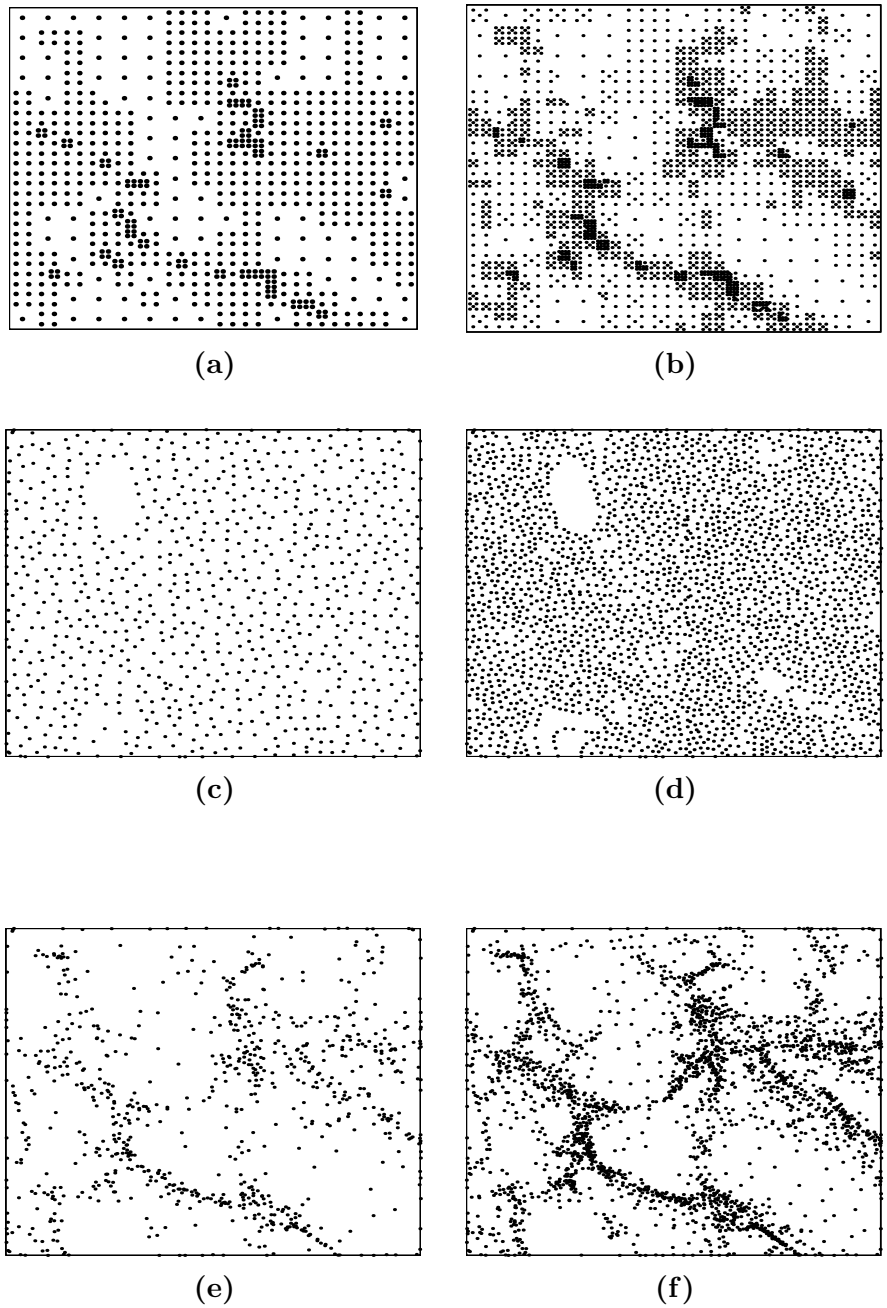
[13] H. Wendland: Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics* **4**, 1995, 389–396.

Authors' addresses:

Armin Iske
Zentrum Mathematik
Technische Universität München
D-85747 Garching, GERMANY
iske@ma.tum.de

Jeremy Levesley
Department of Mathematics and Computer Science
University of Leicester
University Road
Leicester, LE1 7RH, ENGLAND
jl1@mcs.le.ac.uk

Figure 4: The subsets C_1, C_2 generated by the quadtree method, **(a)**,**(b)**; the subsets X_1, X_2 output by scattered data filtering, **(c)**,**(d)**, and adaptive thinning, **(e)**,**(f)**.



(a) **(b)**

(c) **(d)**

(e) **(f)**