University of West Bohemia
Department of Computer Science and Engineering
Univerzitní 8
30614 Plzeň
Czech Republic

Interpolation and Approximation Methods for Large Geometric Datasets

State of the Art and Concept of Ph.D. Thesis

Zuzana Majdišová

Technical Report No. DCSE/TR-2016-05
July, 2016

Distribution: public

Technical Report No. DCSE/TR-2016-05
July 2016

Interpolation and Approximation Methods for Large Geometric Datasets

Zuzana Majdišová

Abstract

A surface reconstruction of large scattered datasets using interpolation or approximation methods is often a task in many engineering problems. Several techniques have been developed for the surface reconstruction, but they mostly require the conversion of a scattered dataset to an ordered dataset, i.e. a semi-regular mesh is obtained by using some tessellation techniques, which is computationally expensive. Therefore, we focus to the Radial Basis Function (RBF) methods which are appropriate for large scattered datasets in $d-$dimensional space.

The RBF methods are non-separable as it is based on the distance between two points, and lead to a solution of a linear system of equations. Using RBF methods; the implicit or explicit analytical representation of the surface can be obtained. It is one of the advantages over the classical triangulation methods.

The following report contains the state of the art in the given computer graphics area; it aims to the description of important data structures for storage of the large scattered datasets and several existing RBF methods. Then, the report shows the common problems of these methods. Finally, the report focuses on the presumptive future work.

This work was supported by the Ministry of Education, Youth and Sports, project LH12181 and by the projects SGS-2013-029 and SGS-2016-013.

Copies of this report are available on
http://www.kiv.zcu.cz/publications/
or by surface mail on request sent to the following address:

University of West Bohemia
Department of Computer Science and Engineering
Univerzitní 8
30614 Plzeň
Czech Republic

Acknowledgements

I would like to thank my colleagues for their numerous and practical advices. Great thank belongs to my supervisor Prof. Ing. Václav Skala, CSc. for his time, patience and valuable comments. Furthermore, my thanks also belong to my family and my friends whose support was very important to me during my studies.

Contents

Used Notation

a	Scalar value
$\boldsymbol{a}$	Column vector, i.e. $\boldsymbol{a} = [a_1, \ldots, a_N]^T$
$\boldsymbol{A}$	Matrix
$\boldsymbol{I}$	Identity matrix
$\mathcal{A}$	Set of elements
$\mathbb{R}$	Real numbers
$\mathbb{E}^d$	d-dimensional Euclidean space
∇f	Gradient of function f
$\frac{\partial f}{\partial x}$	Partial derivative of function f with respect to variable x
$\|.\|$	Euclidean norm

Chapter 1

Introduction

Interpolation and approximation are the most frequent operations used in computational techniques. Several techniques have been developed for data interpolation or approximation, but an ordered dataset is mostly expected, e.g. a rectangular mesh, a structured mesh, an unstructured mesh, etc. However, in many engineering problems, data are not ordered, and they are scattered in $d-$dimensional space, in general. In fact, the conversion of a scattered dataset to a semi-regular grid is commonly performed in technical applications using some tessellation techniques. However, this approach is quite prohibitive for the case of $d-$dimensional data due to the computational cost.

Interesting techniques are based on the Radial Basis Function (RBF) method which was originally introduced by [Har71]. They are widely used across many fields solving technical and non-technical problems. RBF techniques are effective tools for solving partial differential equations in engineering and sciences [HSfY15], [LCC13], [Isk04]. Moreover, RBF applications can be found in neural networks, fuzzy systems, pattern recognition, data visualization [PRF14], medical applications, surface reconstruction [IdSPT14], [SPN14], [SPN13], [PS11a], [PS11b], [KHS03], [TO02], [DTS02], [CBC$^+$01], reconstruction of corrupted images [US05], [ZVS09], etc. Note that some of the methods for surface reconstruction mentioned above provide implicit representation of surface, and some other methods give explicit representation. The RBF techniques are really meshless and are based on collocation in a set of scattered nodes. These methods are independent with respect to the dimension of the space and lead to the solution of linear system of equations. The computational cost of RBF approximation increases nonlinearly (almost cubic) with the number of points in the given dataset, and linearly with the dimensionality of data.

The processed point clouds are mostly created by $3D$ scanner, and they contain very large amount of points. Moreover, points in the cloud may not be uniformly distributed, and the holes can be formed. Therefore, it is necessary to develop

the methods which are fast and able to reconstruct corrupted datasets [OBS06], [OBS05], [TRS04], [OBA+03], etc. for mathematical representation of surface. It is possible to obtain further acceleration of calculation using high performance computing, such approaches are introduced e.g. in [TGB14] and [CGGS13].

The RBF approximation is generally faster than the RBF interpolation, but the larger errors and inaccuracies are produced. To solve such problems, the modified, robust moving least square method was presented in [JCW+15].

As mentioned above, RBF methods lead to the solution of a linear system of the size equal to the number of data points, further, current $3D$ data scanners allow acquisition of tens of millions points, thus, there is also an important task to find appropriate data structures for storing the point clouds, reconstructed surfaces and representation of matrix of the linear system. There are many publications [SMP+15], [Law13], [LK11], [Šim09], [BG09], [SHK09] [LD08], [BG08], [LZ06], [Mas03] etc. in which introduce a description of several useful data structures.

1.1 Organization

The rest of this work is divided into three parts. The first part (Chapter 2) provides an overview of data structures for large scattered datasets. The second part (Chapter 3) describes radial basis functions (RBFs) and the basic idea of meshless interpolation and approximation methods.

Chapter 4 covers the main topic of this work - reconstruction of large geometric datasets. In this chapter the important existing interpolation or approximation methods for large geometric datasets are briefly outlined. Finally, the proposal of future work is described.

Chapter 2

Data Structures for Large Scattered Datasets

Meshless interpolation and approximation lead to the solution of linear system of equation. According to the used basis function the matrix of linear system can be sparse or dense. Moreover, the input dataset contains many points which are scattered in space.

Therefore, in this chapter, we focus on existing data structures which can help us with storage of large scattered dataset or necessary sub-results, e.g. sparse matrix etc.

2.1 Sparse Matrix Format

The matrix of linear system of equations can be sparse for some approximation or interpolation methods when the CS-RBFs are used. Moreover, the performance of mathematical operations with sparse matrices depends strongly on the storage format used. Therefore, the most important part of each approximation is a data structure used to store the approximation matrix. There are a number of existing sparse matrix representations, each with different computational characteristics, storage requirements, and methods of accessing and manipulating entries of the matrix. The main difference among existing storage formats is the sparsity pattern, or the structure of the nonzero elements, for which they are best suited. We also have to consider general representations which efficiently store matrices with arbitrary sparsity patterns. There is the summarization of several existing sparse matrix representations performed in the remainder of this section which is based on [BG08], [BG09], [Šim09], [Law13] and [SMP+15].

2.1.1 Diagonal Format

When nonzero values are restricted to a small number of matrix diagonals N_D, then diagonal format (DIA) is an appropriate representation. This format efficiently encodes matrices arising from the common discretization methods, the application of stencils to regular grids etc. A 25×25 sparse matrix with 5 occupied diagonals can be seen in Figure 2.1.

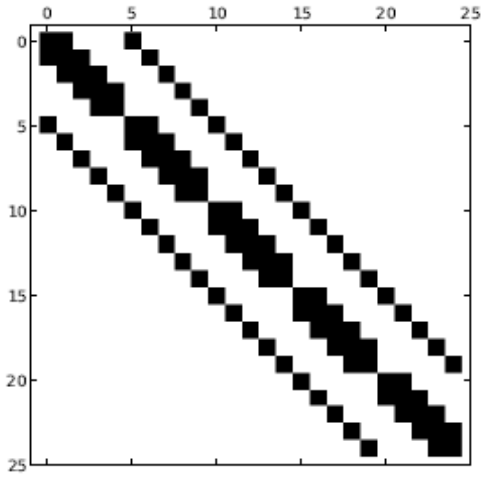


Figure 2.1: A 25×25 sparse matrix with 5 occupied diagonals [BG08].

The diagonal format is formed by two arrays for representation of matrix: `data`, where the nonzero values are stored, and `offsets`, where the offset of each diagonal from the main diagonal is stored. By convection, the main diagonal is represented by offset 0, while $i > 0$ defines the i-th super-diagonal and $i < 0$ the i-th sub-diagonal.

Example of the DIA format for matrix $\boldsymbol{A}$ with four occupied diagonals:

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 & 6 & 0 & 0 \\ 9 & 2 & 0 & 7 & 0 \\ 0 & 1 & 3 & 0 & 8 \\ 4 & 0 & 2 & 4 & 0 \\ 0 & 5 & 0 & 3 & 5 \end{pmatrix}$$

$$\texttt{data} = \begin{bmatrix} * & * & 1 & 6 \\ * & 9 & 2 & 7 \\ * & 1 & 3 & 8 \\ 4 & 2 & 4 & * \\ 5 & 3 & 5 & * \end{bmatrix} \qquad \texttt{offsets} = \begin{bmatrix} -3 & -1 & 0 & 2 \end{bmatrix}$$

$$\text{Usage: } \boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} : \boldsymbol{y}_i = \sum_{j=0}^{N_D-1} \texttt{data}\Big[i, \texttt{offsets}[j] + i\Big] \cdot \boldsymbol{x}_{\texttt{offsets}[j]+i}$$

Note that elements marked with the symbol $*$ are used for padding and may store an arbitrary value.

The benefits of the diagonal format are two. First, this format avoids the need to store row/column indices of nonzero elements because indices are implicitly defined by their position in `data` array and the corresponding offset of the diagonal, i.e. row index of the matrix coincides with row index of the `data` array and column index of the matrix corresponds to sum of the row index of the `data` array and offset value, which has the same column index as nonzero element in `data` array. Second, all memory accesses to arrays are contiguous which improves the efficiency of memory transactions.

The disadvantages of the DIA format are also clear. This representation is appropriate only for certain patterns. Some patterns of sparse matrices, which are inappropriate for DIA, can be seen in Figure 2.2. Moreover, it can also potentially waste storage since it allocates memory for values "outside" the matrix and explicitly stores zero values that occur in occupied diagonals.

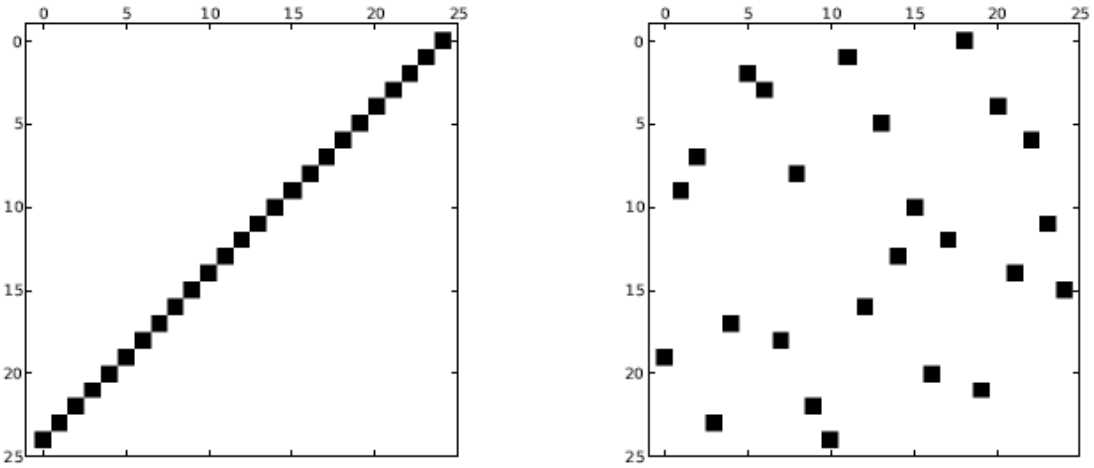


Figure 2.2: Sparsity patterns that are ill-suited to the sparse diagonal format [BG08].

2.1.2 ELLPACK Format

ELLPACK (ELL) format is more general than DIA format and is well-suited to vector architectures. An $M \times N$ sparse matrix which has a maximum of K nonzero values per row is stored as a dense $M \times K$ array `data`, where rows with fewer that K nonzero values are zero-padded, and $M \times K$ array `indices`, where the column index for each nonzero element is stored. Based on empirical results, this format is profitable if at least one third of the matrix rows contain K nonzero values.

Example of the ELL format for matrix $\boldsymbol{A}$ with a maximum of three nonzero values per row:

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 & 6 & 0 & 0 \\ 9 & 2 & 0 & 7 & 0 \\ 0 & 1 & 3 & 0 & 8 \\ 4 & 0 & 2 & 4 & 0 \\ 0 & 5 & 0 & 3 & 5 \end{pmatrix}$$

$$\texttt{data} = \begin{bmatrix} 1 & 6 & * \\ 9 & 2 & 7 \\ 1 & 3 & 8 \\ 4 & 2 & 4 \\ 5 & 3 & 5 \end{bmatrix} \qquad \texttt{indices} = \begin{bmatrix} 0 & 2 & * \\ 0 & 1 & 3 \\ 1 & 2 & 4 \\ 0 & 2 & 3 \\ 1 & 3 & 4 \end{bmatrix}$$

$$\text{Usage: } \boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} : \boldsymbol{y}_i = \sum_{j=0}^{K-1} \texttt{data}\Big[i, \texttt{indices}[i,j]\Big] \cdot \boldsymbol{x}_{\texttt{indices}[i,j]}$$

The ELL format is more general than the DIA format since the nonzero columns need not follow any particular pattern. Note that row indices are defined implicitly and column indices are stored explicitly.

The ELL format is most efficient when the maximum number of nonzero values per row does not substantially differ from average. Therefore, this storage scheme is, for example, appropriate for semi-structured or unstructured meshes, in which the maximum vertex degree is not significantly greater that average degree, see Figure 2.3. In contrast, the example of an unstructured mesh, which is inappropriate for the ELL format, can be seen in Figure 2.4.

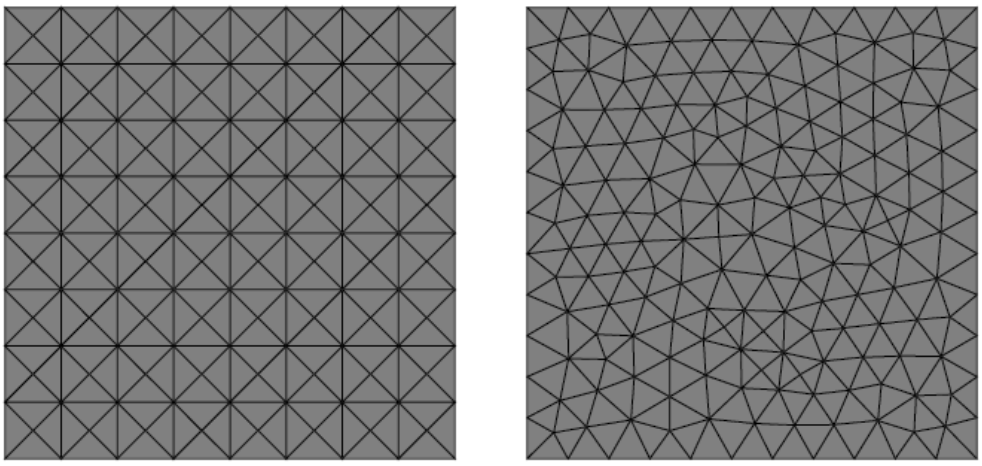


Figure 2.3: The example semi-structured (left) and unstructured (right) meshes whose vertex-edge connectivity is efficiently encoded in ELL format. In each case the maximum vertex degree is not significantly greater that the average degree [BG08].

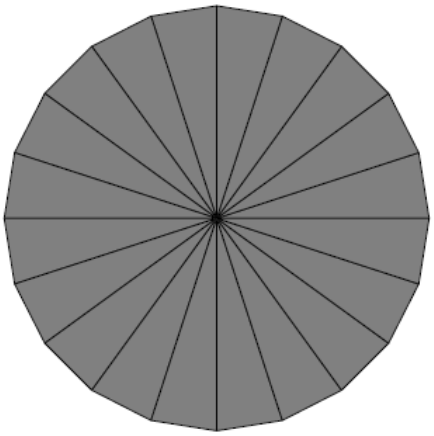


Figure 2.4: The vertex-edge connectivity of the wheel is not efficiently encoded in ELL format. Since the center vertex has high degree relative to average degree, the vast majority of the entries in the `data` and `indices` arrays representation will be wasted [BG08].

2.1.3 Coordinate Format

The coordinate (COO) format is the simplest storage scheme. The sparse matrix is represented by three arrays: `data`, where the N_{NZ} nonzero values are stored, `row`, where the row index of each nonzero element is kept, and `col`, where the column indices of the nonzero values are stored.

Example of the COO format for matrix $\boldsymbol{A}$:

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 & 6 & 0 & 0 \\ 9 & 2 & 0 & 7 & 0 \\ 0 & 1 & 3 & 0 & 8 \\ 4 & 0 & 2 & 4 & 0 \\ 0 & 5 & 0 & 3 & 5 \end{pmatrix}$$

$$\texttt{row} = [\ 0\ \ 0\ \ 1\ \ 1\ \ 1\ \ 2\ \ 2\ \ 2\ \ 3\ \ 3\ \ 3\ \ 4\ \ 4\ \ 4\ ]$$
$$\texttt{col} = [\ 0\ \ 2\ \ 0\ \ 1\ \ 3\ \ 1\ \ 2\ \ 4\ \ 0\ \ 2\ \ 3\ \ 1\ \ 3\ \ 4\ ]$$
$$\texttt{data} = [\ 1\ \ 6\ \ 9\ \ 2\ \ 7\ \ 1\ \ 3\ \ 8\ \ 4\ \ 2\ \ 4\ \ 5\ \ 3\ \ 5\ ]$$

Usage: $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} : \boldsymbol{y}_{\text{row}[i]} = \boldsymbol{y}_{\text{row}[i]} + \texttt{data}[i] \cdot \boldsymbol{x}_{\text{col}[i]}, \quad i = 0, \ldots, N_{NZ}$

The benefit of the COO format is its generality, i.e. arbitrary sparse matrix can be represented by COO format and the required storage is always proportional to the number of nonzero values.

The disadvantage of the COO format is that both row and column indices are stored explicitly which declines the efficiency of memory transactions.

2.1.4 Compressed Sparse Row Format

The compressed sparse row (CSR) format is the most common format for storing sparse matrices. The CSR format can be viewed as a natural extension of the COO representation with a simple compression scheme, which is applied to the row indices. Like the COO format, CSR explicitly stores column indices and nonzero values in arrays `indices` and `data`. A third array `pts`, which is used by CSR format, is array of row pointers. This array allows the CSR format to represent rows of varying length. For $M \times N$ sparse matrix, `ptr` has length $M + 1$ and stores the offset into the i-th row in `ptr`$[i]$. The last element in `ptr`, which would otherwise correspond to the $(M + 1)$-st row, stores the number of nonzero values in the sparse matrix.

Example of the CSR format for matrix $\boldsymbol{A}$:

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 & 6 & 0 & 0 \\ 9 & 2 & 0 & 7 & 0 \\ 0 & 1 & 3 & 0 & 8 \\ 4 & 0 & 2 & 4 & 0 \\ 0 & 5 & 0 & 3 & 5 \end{pmatrix}$$

$$\texttt{ptr} = [\ 0\ \ 2\ \ 5\ \ 8\ \ 11\ \ 14\ ]$$

$$\texttt{indices} = [\ 0\ \ 2\ \ 0\ \ 1\ \ 3\ \ 1\ \ 2\ \ 4\ \ 0\ \ 2\ \ 3\ \ 1\ \ 3\ \ 4\ ]$$

$$\texttt{data} = [\ 1\ \ 6\ \ 9\ \ 2\ \ 7\ \ 1\ \ 3\ \ 8\ \ 4\ \ 2\ \ 4\ \ 5\ \ 3\ \ 5\ ]$$

$$\text{Usage: } \boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} : \boldsymbol{y}_i = \sum_{j=\texttt{ptr}[i]}^{\texttt{ptr}[i+1]-1} \texttt{data}[j] \cdot \boldsymbol{x}_{\texttt{indices}[j]}$$

The benefits of the CSR format are two. First, row pointers facilitate fast querying of matrix values. Second, row pointers allow readily computed the other quantities of interest, such as the number of nonzero values in a particular row. For these reasons, the CSR format is commonly used for sparse matrix computations.

2.1.5 Hybrid Format

While ELL format is well-suited to vector architectures, its efficiency rapidly decreases when the number of nonzero values per row varies. In contrary, efficiency of the COO format is invariant to the distribution of nonzero values. To obtain the benefits of both, these are combined into a hybrid ELL/COO format.

The main idea of the hybrid (HYB) format is to store the typical number of nonzero values per row in the ELL data structure and the remaining elements of exceptional rows in the COO format. The typical number of nonzero values per row is often known as a priori. However, in the general case this number must be

determined from the sparse matrix. It can be performed so that the histogram of the row sizes is computed and the number K is determined. In this case, the number K is the largest number such that using K columns per row in the ELL portion of the HYB format meets a certain objective measure. Based on empirical results, it is shown that the fully-occupied ELL format is roughly three times faster than COO. Therefore, it is profitable to add a K-th column to the ELL representation if at least one third of the matrix rows contains least K nonzero values. The example of the HYB format can be seen in Figure 2.5.

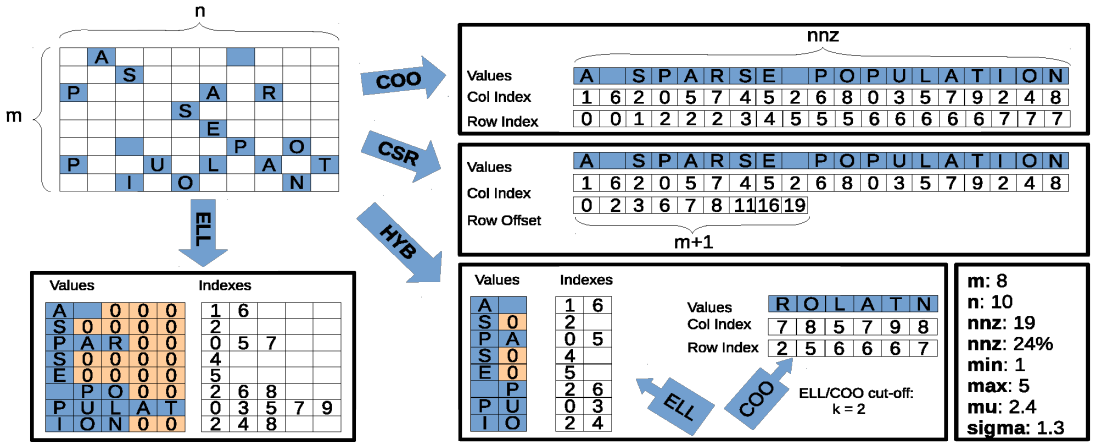


Figure 2.5: Sparse representations for matrix in COO, CSR, ELL and HYB format [SMP+15].

2.1.6 Quadtree Data Format

The consequence of indirect addressing is that previous formats are slow. Therefore, the quadtree data format has been developed. This format is based on quadtree (4-ary tree) which is the recursive data structure. In this case, the quadtree represents a partition of the matrix into submatrices (nodes), see Figure 2.6. Each node of the tree is assigned with one type. Inner nodes of this tree can be divided into "Mixed" or "Empty" nodes. "Empty" node is represented by the NULL pointer. Leaves of the quadtree are divided into "Full" or "Sparse" nodes. The "Sparse" node is sparse submatrix represented by COO format.

Benefits of using quadtree are the following. It can be performed an easy and fast conversion from standard sparse matrix storage formats such as COO or CSR. In comparison to standard formats; quadtree can be relatively easily modified (the exact complexity depends on the type of modified node). Moreover, the better performance against standard storage formats, which is achieved in that the matrix is stored as small dense submatrices, is provided.

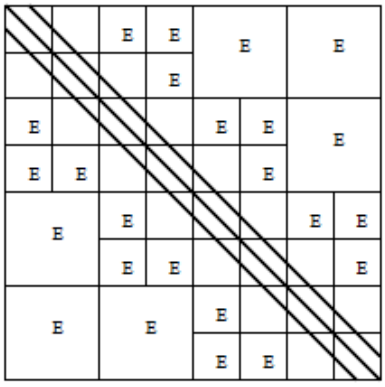


Figure 2.6: Tridiagonal matrix in the quadtree storage format. Locations of nonzero elements and "Empty" nodes are marked [Šim09].

2.2 Space Subdivision

The space subdivision techniques are very often used for determination which object from the given dataset lying at least partially within the given area. These techniques uniformly divide space into smaller areas, see Figure 2.7. Their aim is to enable fast test whether a geometric object resides at least partially within the given area. Standard techniques lead to memory requirements which are possible to approximately estimated as $O(p \cdot q \cdot M^d)$, where p is a number of objects, d is a dimension of space, M is a number of subdivision in one axis and q is a probability that an "average" object hits the given interval, generally q is a function defined as $q = q(M, p, s)$, where s is a size of an "average" object. It is obvious that memory consumption is rapidly growing with the refinement of the subdivision and the dimension of space. So those requirements are not acceptable in the case of small objects.

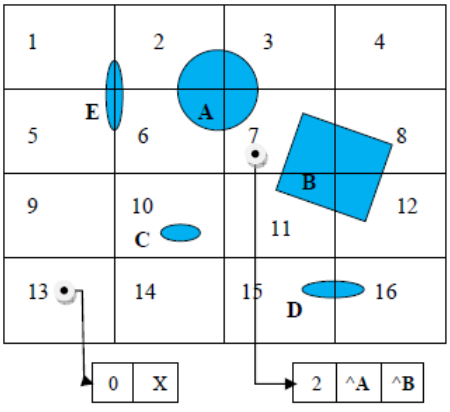


Figure 2.7: Space partitioning in $\mathbb{E}^2$ [Ska12].

2.2.1 Residency Mask (RM)

The residency mask [Cyc92] is a bit vector in which each bit is used for the cell within the partitioned $d-$dimensional space. The residency mask is determined for each object in this space. Generally, an appropriate bit is set to 1 if the object resides at least partially within the given cell of $d-$dimensional space. Residency masks for objects in Figure 2.7 correspond to:

Item	**Bit Mask**			
object **A**	[0000	0000	0110	0110]
object **B**	[0000	1100	1100	0000]
object **C**	[0000	0010	0000	0000]
object **D**	[1100	0000	0000	0000]
object **E**	[0000	0000	0011	0011]

It can be seen that memory requirements of residency masks are possible to estimate as:

$$mem_{RM} = \frac{p \cdot M^k}{8}[\text{Bytes}],$$

where p is a number of objects, d is the dimension of space and M is a number of the subdivision in one axis. The technique mentioned above allows a quick determination of a potential intersection of the objects using logical AND operation (**land**), i.e. if the object with residency mask RM_i and the object with residency mask RM_j are intersected, then a condition:

$$RM_i \textbf{ land } RM_j \neq [0, \ldots, 0] \qquad i, j \in 1, \ldots, p, \tag{2.1}$$

where p is a number of object in space, is true. Therefore, if condition (2.1) is false, then no intersection is possible for these objects.

This technique is useful for finding all cells which interfere with the given object. The disadvantage of this technique is that if the number of subdivisions M is higher and objects are small, then very long binary vectors are obtained containing almost zeros.

2.2.2 Binary Mask (BM)

The binary masks [Ska12] are again represented by bit vectors. For simplicity, we assume that we have two dimensional case. However, note that this approach is generally extensible for $d-$dimensional space. Let us define ${}^{i}\boldsymbol{R}$, resp. ${}^{j}\boldsymbol{C}$, sets of all objects that interfere with the given $i-$th row slice, resp. $j-$th column slice. Then a set ${}^{ij}\boldsymbol{W}$ is defined as:

$${}^{ij}\boldsymbol{W} = {}^{i}\boldsymbol{R} \cap {}^{j}\boldsymbol{C} \tag{2.2}$$

and determines all object that can interfere with the subinterval at the (i, j) position, see Figure 2.8.

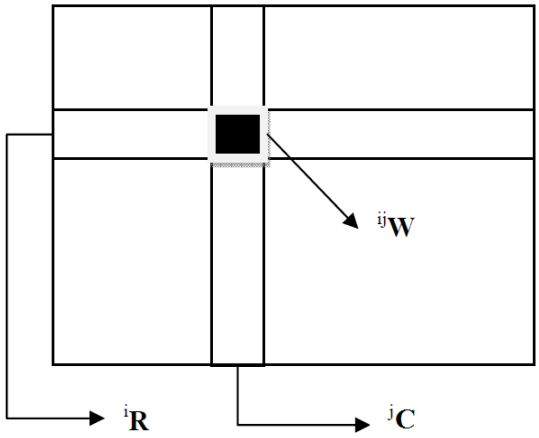


Figure 2.8: Schema of binary mask [Ska12].

The sets ${}^{i}\boldsymbol{R}$, resp. ${}^{j}\boldsymbol{C}$, are represented using bit vectors in which each bit is used for one object in the given space, i.e. $k-$th bit expresses whether the $k-$th object interferes with the given slice, and length of bit vector is p, where p is a number of objects in given space. Thus, term (2.2) is possible to express using bitwise logical AND operation (**land**) as:

$$ {}^{ij}\boldsymbol{W} = {}^{i}\boldsymbol{R} \textbf{ land } {}^{j}\boldsymbol{C} \qquad i, j \in 1, \ldots, M, \tag{2.3} $$

where M is a number of subdivision in one axis.

It can be seen that memory requirements of binary masks are given as:

$$ mem_{BM} = \frac{d \cdot p \cdot M}{8} [\text{Bytes}], $$

where p is a number of objects, d is dimension of space and M is a number of subdivision in one axis.

This technique is useful for finding all objects which interfere with the given cell. Another advantage of binary masks is that we can easily check the consistency of the given scene. Let us define:

$$ \boldsymbol{X} = \cup^{i}\boldsymbol{R} = [x_1, \ldots, x_p], \tag{2.4} $$

where p is number of objects. Then the $k-$th object is not contained in any subinterval and the given scene is inconsistent if

$$ \exists k \in \{1, \ldots, p\} : x_k = 0. $$

Similarly the other axes are handled.

2.3 Summary

Sparse matrix formats are useful for sparse matrix-vector multiplication on GPU which is one of the important operations for calculation of RBF methods. Moreover, the quadtree data format allows the effective multiplication of sparse matrix by a sparse matrix which can be applied for RBF approximation.

If we want to perform the approximation or interpolation of the given dataset using the domain decomposition method, which is one of the possible approaches, then we will certainly find a usage for the space subdivision techniques.

Chapter 3

Meshless Interpolation and Approximation

Interpolation and approximation are probably the most frequent operations used in computational techniques. Several techniques have been developed for data interpolation or approximation, but an ordered dataset is mostly expected, e.g. a rectangular mesh, a structured mesh, an unstructured mesh, etc. However, in many engineering problems, data are not ordered and they are scattered in $d-$dimensional space, in general. In fact, the conversion of a scattered dataset to a semi-regular grid is commonly performed in technical applications using some tessellation techniques. However, this approach is quite prohibitive for the case of $d-$dimensional data due to the computational cost. Interesting techniques which are appropriate for large scattered (unordered) datasets are meshless interpolation resp. approximation. These meshless techniques do not require conversion to a semi-regular grid.

The RBF techniques are based on the distance between two points and lead to a solution of a linear system of equations.

3.1 Radial Basis Function (RBF)

Radial basis functions (RBFs) are traditional and powerful tools for the meshless interpolation and approximation of scattered data. These functions are real-valued functions which depend only on the distance from the fixed center point. More precisely, let us consider an univariate function:

$$\phi : [0, \infty) \to \mathbb{R} \tag{3.1}$$

then the radial basis function $\Phi_i : \mathbb{R}^d \to \mathbb{R}$ is defined as:

$$\Phi_i(\boldsymbol{x}) = \phi(r_i) = \phi(\|\boldsymbol{x} - \boldsymbol{x}_i\|), \tag{3.2}$$

where $\{\boldsymbol{x}_i\}_{i=1}^N \subset \mathbb{R}^d$ is a set of N different points which are so-called the centers and $\|.\|$ is some norm in $\mathbb{R}^d$. The Euclidean norm is usually used.

Name of RBF (specifically word radial) is based on the following property. The value of Φ at any point at a certain fixed distance from the fixed center point is constant, i.e.

$$\|\boldsymbol{x}_1\| = \|\boldsymbol{x}_2\| \quad \Rightarrow \quad \Phi(\boldsymbol{x}_1) = \Phi(\boldsymbol{x}_2), \qquad \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^d. \tag{3.3}$$

Thus, Φ is radially (or spherically) symmetric around its center. Example of a such function can be seen in Figure 3.1.

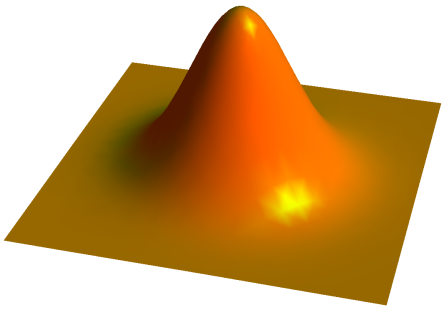


Figure 3.1: Example of RBF

Nice property of RBF interpolants is invariance for all Euclidean transformations (i.e. translations, rotations and reflections). This means that it does not matter whether we first compute the RBF interpolant and then apply a Euclidean transformations, or if the transform of the data is performed first and then compute the interpolant. This is a consequence of fact that Euclidean transformations are 2-norm-invariant.

There are two main groups of basis functions: global RBFs (e.g. [Duc77] and [Sch79]) and "local" Compactly Supported RBFs (CS-RBFs) [Wen06]. Fitting scattered data with CS-RBFs leads to a simpler and faster computation because a system of linear equations has a sparse matrix. However, approximation using CS-RBFs is sensitive to the density of approximated scattered data and to the choice of a shape parameter. Global RBFs lead to a linear system of equation with a dense matrix and their usage is based on sophisticated techniques, such as, the fast multipole method [Dar00]. The global RBFs are useful in repairing incomplete datasets, and they are less sensitive to the density of data. Table 3.1 presents typical examples of global RBFs and Table 3.2 presents examples of "local" Wendland's CS-RBFs. Note that the notation $(1-\alpha r)_+^q$ means:

$$(1-\alpha r)_+^q = \begin{cases} (1-\alpha r)^q & \text{if} \quad 0 \leq \alpha r \leq 1 \\ 0 & \text{if} \quad \alpha r > 1 \end{cases} \tag{3.4}$$

Table 3.1: Typical examples of global RBFs

Global RBF	$\boldsymbol{\phi(r)}$
Gauss function [Sch79]	$e^{-(\alpha r)^2}$
Inverse Quadric (IQ)	$\frac{1}{1+(\alpha r)^2}$
Inverse Multiquadric (IMQ)	$\frac{1}{\sqrt{1+(\alpha r)^2}}$
Multiquadric (MQ)	$\sqrt{1+(\alpha r)^2}$
Thin-Plate Spline (TPS) [Duc77]	$(\alpha r)^2 \log(\alpha r)$

and α is a shape parameter. Figure 3.2 shows behavior of selected Wendland's CS-RBFs for shape parameter $\alpha = 1$; on the x axis is a radius value r (negative part is just for illustration of the symmetry properties).

Table 3.2: Typical examples of "local" Wendland's CS-RBFs [Wen95]

ID	**CS-RBF**	$\boldsymbol{\phi(r)}$
1	Wendland's $\phi_{1,0}$	$(1-\alpha r)_+$
2	Wendland's $\phi_{1,1}$	$(1-\alpha r)_+^3(3\alpha r+1)$
3	Wendland's $\phi_{1,2}$	$(1-\alpha r)_+^5(8(\alpha r)^2+5\alpha r+1)$
4	Wendland's $\phi_{3,0}$	$(1-\alpha r)_+^2$
5	Wendland's $\phi_{3,1}$	$(1-\alpha r)_+^4(4\alpha r+1)$
6	Wendland's $\phi_{3,2}$	$(1-\alpha r)_+^6(35(\alpha r)^2+18\alpha r+3)$
7	Wendland's $\phi_{3,3}$	$(1-\alpha r)_+^8(32(\alpha r)^3+25(\alpha r)^2+8\alpha r+1)$
8	Wendland's $\phi_{5,0}$	$(1-\alpha r)_+^3$
9	Wendland's $\phi_{5,1}$	$(1-\alpha r)_+^5(5\alpha r+1)$
10	Wendland's $\phi_{5,2}$	$(1-\alpha r)_+^7(16(\alpha r)^2+7\alpha r+1)$

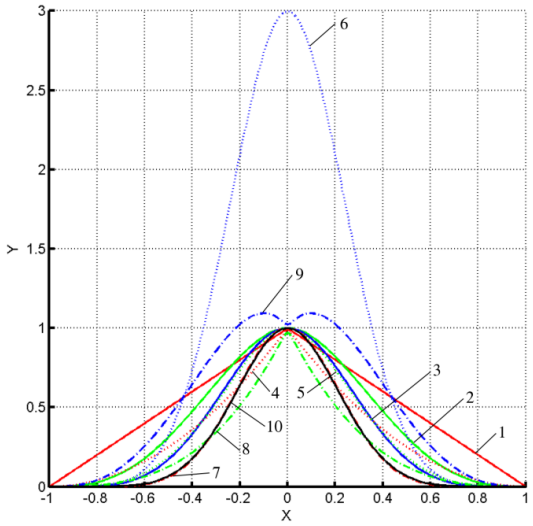


Figure 3.2: Geometrical properties of CS-RBFs [Ska13]

3.2 RBF Interpolation Methods

We assume that we have an unordered dataset $\{\boldsymbol{x}_i\}_1^N \in \mathbb{E}^2$. However, this approach is applicable for $d-$dimensional space, in general. Moreover, each point $\boldsymbol{x}_i$ is associated with vector $\boldsymbol{h}_i \in \mathbb{E}^p$ of the given values, where p is dimension of vector, or scalar value $h_i \in \mathbb{E}^1$. For an explanation of the RBF interpolation, we consider the case when each point $\boldsymbol{x}_i$ is associated with scalar value h_i. The RBF interpolation is based on computing the distance of two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ from the dataset.

3.2.1 RBF Interpolation

In this section, the RBF interpolation method, recently introduced e.g. in [Fas07] (Chapter 2), [ALP14], [Ska15], etc., and its properties are described.

The interpolated value can be determined as:

$$f(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \phi(r_j) = \sum_{j=1}^{N} c_j \phi(\|\boldsymbol{x} - \boldsymbol{x}_j\|), \tag{3.5}$$

where the interpolating function $f(\boldsymbol{x})$ is represented as a sum of N RBFs, each centered at a different data point $\boldsymbol{x}_j$, and weighted by an appropriate weight c_j.

It can be seen, to solve the interpolation problem, the distance matrix and a radial basis expansion are used. This leads to solve a linear system of equations

for the given dataset:

$$h_i = f(\boldsymbol{x}_i) = \sum_{j=1}^{N} c_j \phi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|) = \sum_{j=1}^{N} c_j \phi_{i,j} \qquad i = 1, \ldots, N. \tag{3.6}$$

This linear system of equation can be represented as the matrix equation:

$$\boldsymbol{A}\boldsymbol{c} = \boldsymbol{h}, \tag{3.7}$$

where matrix $\boldsymbol{A}$ is symmetric matrix. Equation (3.7) can be expressed in the form:

$$\begin{pmatrix} \phi_{1,1} & \cdots & \phi_{1,i} & \cdots & \phi_{1,N} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \phi_{i,1} & \cdots & \phi_{i,i} & \cdots & \phi_{i,N} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \phi_{N,1} & \cdots & \phi_{N,i} & \cdots & \phi_{N,N} \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_i \\ \vdots \\ c_N \end{pmatrix} = \begin{pmatrix} h_1 \\ \vdots \\ h_i \\ \vdots \\ h_N \end{pmatrix}. \tag{3.8}$$

This linear system of equations can be solved by the Gauss elimination method, the LU decomposition, etc.

3.2.2 RBF Interpolation with Polynomial Reproduction

The method which was introduced in Section 3.2.1 can theoretically have problems with stability and solvability because there is not guaranteed that the matrix of linear system is well-conditioned. Therefore, the radial basis function expression is extended by polynomial function $P_k(\boldsymbol{x})$ of degree k. This approach was introduced e.g. in [Fas07] (Chapter 6), [Ska13], [ALP14], etc.

The RBF interpolation of the given dataset can be analytically expressed by using this approach as:

$$f(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \phi(r_j) + P_k(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \phi(\|\boldsymbol{x} - \boldsymbol{x}_j\|) + P_k(\boldsymbol{x}), \tag{3.9}$$

where $\boldsymbol{x}_j$ is point from the given dataset at which RBF is centered. Now, the vector of weights $\boldsymbol{c} = (c_1, \ldots, c_N)^T$ has to be necessary determined.

In practice, a linear polynomial $P_1(\boldsymbol{x})$, i.e. polynomial of first order, is used. Thus, the following linear system of equations is obtained:

$$\begin{aligned} h_i = f(\boldsymbol{x}_i) &= \sum_{j=1}^{N} c_j \phi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|) + a_x x_i + a_y y_i + a_0 \\ &= \sum_{j=1}^{N} c_j \phi_{i,j} + a_x x_i + a_y y_i + a_0 \qquad i = 1, \ldots, N, \end{aligned} \tag{3.10}$$

where vector $\boldsymbol{x}_i$ is given as $\boldsymbol{x}_i = (x_i, y_i)^T$ and vector $\boldsymbol{a} = (a_y, a_y, a_0)^T$ is vector of linear polynomial coefficients. However, now we have $N + 3$ unknowns, namely the vector of weights $\boldsymbol{c} = (c_1, \ldots, c_N)^T$ and vector of linear polynomial coefficients $\boldsymbol{a} = (a_y, a_y, a_0)^T$, and we have only N conditions to determine them, namely the linear system (3.10). Therefore, additional conditions must be found. Specifically, we can add the following conditions:

$$\sum_{i=1}^{N} c_i = 0 \qquad \sum_{i=1}^{N} c_i \boldsymbol{x}_i = \boldsymbol{0}. \tag{3.11}$$

It leads to the square linear system of equations which is possible represented by matrix form:

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{P} \\ \boldsymbol{P}^T & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{c} \\ \boldsymbol{a} \end{pmatrix} = \begin{pmatrix} \boldsymbol{h} \\ \boldsymbol{0} \end{pmatrix} \tag{3.12}$$

and can be solved by the Gauss elimination method, the LU decomposition, etc.

3.2.3 Regularized RBF Interpolation

Sometimes, it may happen that the RBF interpolation result (Section 3.2.1) will pass through the given dataset, but somewhere else this RBF result will behave unpredictably, e.g. Figure 3.3. Intuitively, this may be due to the fact that the

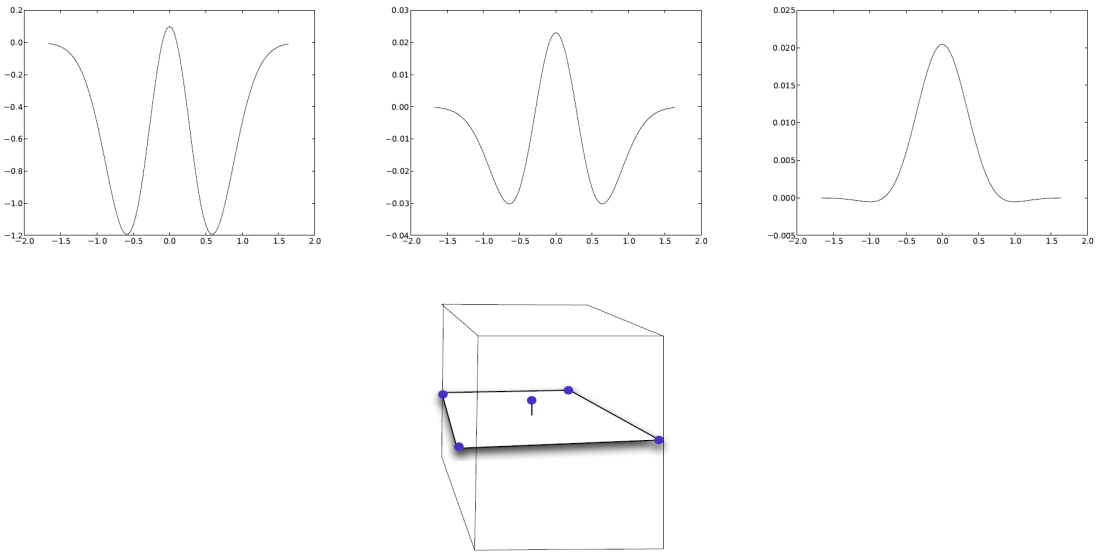


Figure 3.3: Illustration of ill-conditioning and regularization. Top: From left to right, the regularization parameter is 0, 0.01 and 0.1. Note, the vertical scale is changing. Bottom: The geometry of this interpolation problem. The data are zero at the corners of a square with a single non-zero value in the center [ALP14].

matrix is divided by "nearly zero" somewhere, resulting in large overshoots. This behavior occurs when the matrix of the linear system (3.7) is not singular, but is poorly conditioned. In the case mentioned above, a simple solution is to apply the regularization, see e.g. [EPP00], thus, obtaining a minimization problem. The result of this minimization problem can be expressed by following matrix equation:

$$(\boldsymbol{A}^T\boldsymbol{A} + \omega\boldsymbol{I})\boldsymbol{c} = \boldsymbol{A}^T\boldsymbol{f}, \tag{3.13}$$

where ω is the regularization parameter and $\boldsymbol{I}$ is the identity matrix. The regularization parameter ω is chosen as a very small adjustable number such as 0.00001.

3.2.4 Summary

The disadvantage of RBF interpolation is large and usually ill-conditioned matrix of the linear system of equations. Moreover, in the case of an oversampled dataset or intended reduction; we want to reduce the given problem, i.e. reduce the number of weights and used basis functions, and, furthermore, preserve good precision of the approximated solution. The approach, which includes the reduction, is called a RBF approximation, and will be described in following.

3.3 RBF Approximation Methods

For simplicity, we assume that we have an unordered dataset $\{\boldsymbol{x}_i\}_1^N$ in $\mathbb{E}^2$. However, note that this approach is generally applicable for d-dimensional space. Further, each point $\boldsymbol{x}_i$ from the dataset is associated with vector $\boldsymbol{h}_i \in \mathbb{E}^p$ of the given values, where p is the dimension of the vector, or scalar value $h_i \in \mathbb{E}^1$. For an explanation of the RBF approximation, let us consider the case when each point $\boldsymbol{x}_i$ is associated with scalar value h_i, e.g. a $2\,1/2\,D$ surface. Let us introduce a set of new reference points $\{\boldsymbol{\xi}_j\}_1^M$, see Figure 3.4.

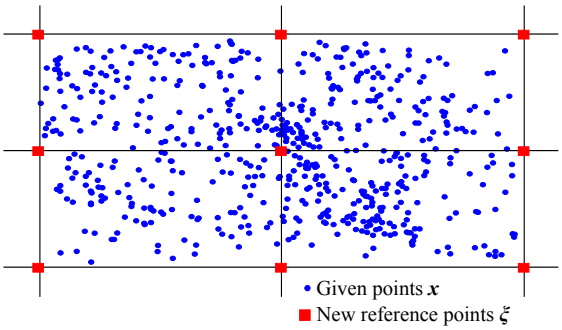


Figure 3.4: RBF approximation and reduction of points.

These reference points may not necessarily be in a uniform grid. It is appropriate that their placement reflects the given surface (e.g. the terrain profile, etc.) as well as possible. The number of added reference points $\boldsymbol{\xi}_j$ is M, where $M \ll N$. The RBF approximation is based on computing the distance of given point $\boldsymbol{x}_i$ and reference point $\boldsymbol{\xi}_j$.

3.3.1 RBF Approximation

In this section, the RBF approximation method, which was recently introduced in [Ska13], and its properties are decribed.

The approximated value can be determined similarly as for interpolation (see [Ska13]):

$$f(\boldsymbol{x}) = \sum_{j=1}^{M} c_j \phi(r_j) = \sum_{j=1}^{M} c_j \phi(\|\boldsymbol{x} - \boldsymbol{\xi}_j\|), \tag{3.14}$$

where the approximating function $f(\boldsymbol{x})$ is represented as a sum of M RBFs, each associated with a different reference point $\boldsymbol{\xi}_j$, and weighted by an appropriate coefficient c_j.

It can be seen that we get an overdetermined linear system of equations for the given dataset:

$$h_i = f(\boldsymbol{x}_i) = \sum_{j=1}^{M} c_j \phi(\|\boldsymbol{x}_i - \boldsymbol{\xi}_j\|) = \sum_{j=1}^{M} c_j \phi_{i,j} \qquad i = 1, \ldots, N. \tag{3.15}$$

The linear system of equations (3.15) can be represented as the matrix equation:

$$\boldsymbol{A}\boldsymbol{c} = \boldsymbol{h}, \tag{3.16}$$

where the number of rows is $N \ll M$ and M is the number of unknown weights $[c_1, \ldots, c_M]^T$, i.e. the number of reference points. (3.16) can be expressed in the form:

$$\begin{pmatrix} \phi_{1,1} & \cdots & \phi_{1,M} \\ \vdots & \ddots & \vdots \\ \phi_{i,1} & \cdots & \phi_{i,M} \\ \vdots & \ddots & \vdots \\ \phi_{N,1} & \cdots & \phi_{N,M} \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_M \end{pmatrix} = \begin{pmatrix} h_1 \\ \vdots \\ h_i \\ \vdots \\ h_N \end{pmatrix}. \tag{3.17}$$

Thus, the presented system is overdetermined, i.e. the number of equations N is higher than number of variables M. This linear system of equations can be solved by the least squares method as $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{c} = \boldsymbol{A}^T\boldsymbol{h}$ or singular value decomposition, etc.

3.3.2 RBF Approximation using Lagrange Multipliers

The RBF approximation introduced by Fasshauer [Fas07] (Chapter 19) is based on Lagrange multipliers. In this section, the properties of this method will be briefly summarized.

This RBF approximation is formulated as a constrained quadratic optimization problem. The goal of this method is to approximate the given dataset by function:

$$f(\boldsymbol{x}) = \sum_{j=1}^{M} c_j \phi(\|\boldsymbol{x} - \boldsymbol{\xi}_j\|), \tag{3.18}$$

where the approximating function $f(\boldsymbol{x})$ is represented as a sum of M RBFs, each associated with a different reference point $\boldsymbol{\xi}_j$, and weighted by an appropriate coefficient c_j. Therefore, it is necessary to determine the vector of weights $\boldsymbol{c} = (c_1, \ldots, c_M)^T$, which leads to the minimization of the quadratic form:

$$\frac{1}{2}\boldsymbol{c}^T \boldsymbol{Q} \boldsymbol{c}, \tag{3.19}$$

where $\boldsymbol{Q}$ is some $M \times M$ symmetric positive definite matrix. This quadratic form is minimized subject to the N linear constraints $\boldsymbol{A}\boldsymbol{c} = \boldsymbol{h}$, where $\boldsymbol{A}$ is an $N \times M$ matrix with full rank, and the right-hand side $\boldsymbol{h} = (h_1, \ldots, h_N)^T$ is given. Thus, the constrained quadratic minimization problem can be described as a linear system of equations:

$$F(\boldsymbol{c}, \boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{c}^T \boldsymbol{Q} \boldsymbol{c} - \boldsymbol{\lambda}^T (\boldsymbol{A}\boldsymbol{c} - \boldsymbol{h}), \tag{3.20}$$

where $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_N)^T$ is the vector of Lagrange multipliers, and we need to find the minimum of (3.20) with respect to $\boldsymbol{c}$ and $\boldsymbol{\lambda}$. This leads to solving the following system:

$$\begin{aligned} \frac{\partial F(\boldsymbol{c}, \boldsymbol{\lambda})}{\partial \boldsymbol{c}} &= \boldsymbol{Q}\boldsymbol{c} - \boldsymbol{A}^T \boldsymbol{\lambda} = \boldsymbol{0} \\ \frac{\partial F(\boldsymbol{c}, \boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}} &= \boldsymbol{A}\boldsymbol{c} - \boldsymbol{h} = \boldsymbol{0} \end{aligned} \tag{3.21}$$

or, in matrix form:

$$\begin{pmatrix} \boldsymbol{Q} & -\boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{c} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{h} \end{pmatrix}, \tag{3.22}$$

where $Q_{i,j} = \phi(\|\boldsymbol{\xi}_i - \boldsymbol{\xi}_j\|)$ and $\boldsymbol{Q}$ is a symmetric matrix. (3.22) is then solved.

It should be noted that we want to minimize M in order to reduce the computational cost of the approximated value $f(\boldsymbol{x})$ as much as possible.

3.3.3 Original RBF Approximation with Polynomial Reproduction

The method which was introduced in Section 3.3.1 can theoretically have problems with stability and solvability. Therefore, the RBF approximant (3.14) is usually extended by polynomial function $P_k(\boldsymbol{x})$ of degree k. The original approach of RBF approximation with polynomial reproduction was introduced by Fasshauer [Fas07] (Chapter 19.4).

The goal of this approach is to approximate the given dataset of N points by function:

$$f(\boldsymbol{x}) = \sum_{j=1}^{M} c_j \phi(\|\boldsymbol{x} - \boldsymbol{\xi}_j\|) + P_k(\boldsymbol{x}). \tag{3.23}$$

where $\boldsymbol{\xi}_j$ are reference points specified by a user. Now, it is necessary to determine the vector of weights $c = (c_1, \ldots, c_M)^T$. It is achieved by solving an overdetermined linear system of equations:

$$\begin{aligned} h_i = f(\boldsymbol{x}_i) &= \sum_{j=1}^{M} c_j \phi(\|\boldsymbol{x}_i - \boldsymbol{\xi}_j\|) + P_k(\boldsymbol{x}_i) \\ &= \sum_{j=1}^{M} c_j \phi_{i,j} + P_k(\boldsymbol{x}_i) \qquad i = 1, \ldots, N, \end{aligned} \tag{3.24}$$

where $\boldsymbol{x}_i$ is point from the given dataset and is associated with scalar value h_i.

In practice, a linear polynomial:

$$P_1(\boldsymbol{x}) = \boldsymbol{a}^T \boldsymbol{x} + a_0 \tag{3.25}$$

is used and additional conditions are applied:

$$\sum_{i=1}^{M} c_i = 0 \qquad \sum_{i=1}^{M} c_i \boldsymbol{\xi}_i = \boldsymbol{0}. \tag{3.26}$$

Geometrically, the coefficient a_0 determines the placement of the hyperplane and the expression $\boldsymbol{a}^T \boldsymbol{x}$ represents the inclination of the hyperplane.

It can be seen that for d-dimensional space a linear system of $(N+d+1)$ equations in $(M + d + 1)$ variables has to be solved, where N is number of points in the given dataset, M is number of reference points and d is dimensionality of data.

For $d = 2$ vectors $\boldsymbol{x}_i$, $\boldsymbol{\xi}_j$ and $\boldsymbol{a}$ are given as $\boldsymbol{x}_i = (x_i, y_i)^T$, $\boldsymbol{\xi}_j = (\xi_j, \eta_j)^T$ and $\boldsymbol{a} = (a_x, a_y)^T$. Thus, for $\mathbb{E}^2$ and the given dataset we can write this linear system of equations in the following matrix form:

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{P} \\ \boldsymbol{\Xi} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{c} \\ \boldsymbol{a} \\ a_0 \end{pmatrix} = \begin{pmatrix} \boldsymbol{h} \\ \boldsymbol{0} \end{pmatrix}. \tag{3.27}$$

This system is overdetermined ($M \ll N$) and can be solved by the least squares method as:

$$\begin{pmatrix} \boldsymbol{A}^T\boldsymbol{A} + \boldsymbol{\Xi}^T\boldsymbol{\Xi} & \boldsymbol{A}^T\boldsymbol{P} \\ \boldsymbol{P}^T\boldsymbol{A} & \boldsymbol{P}^T\boldsymbol{P} \end{pmatrix} \begin{pmatrix} \boldsymbol{c} \\ \boldsymbol{a} \\ a_0 \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}^T\boldsymbol{h} \\ \boldsymbol{P}^T\boldsymbol{h} \end{pmatrix} \tag{3.28}$$

where

$$\boldsymbol{A}^T\boldsymbol{A} + \boldsymbol{\Xi}^T\boldsymbol{\Xi} = \begin{pmatrix} \sum_{i=1}^{N} \phi_{i1}\phi_{i1} + \boldsymbol{\xi}_1 \cdot \boldsymbol{\xi}_1 + 1 & \cdots & \sum_{i=1}^{N} \phi_{i1}\phi_{iM} + \boldsymbol{\xi}_1 \cdot \boldsymbol{\xi}_M + 1 \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^{N} \phi_{iM}\phi_{i1} + \boldsymbol{\xi}_M \cdot \boldsymbol{\xi}_1 + 1 & \cdots & \sum_{i=1}^{N} \phi_{iM}\phi_{iM} + \boldsymbol{\xi}_M \cdot \boldsymbol{\xi}_M + 1 \end{pmatrix}$$

$$\boldsymbol{P}^T\boldsymbol{A} = (\boldsymbol{A}^T\boldsymbol{P})^T = \begin{pmatrix} \sum_{i=1}^{N} x_i\phi_{i1} & \cdots & \sum_{i=1}^{N} x_i\phi_{iM} \\ \sum_{i=1}^{N} y_i\phi_{i1} & \cdots & \sum_{i=1}^{N} y_i\phi_{iM} \\ \sum_{i=1}^{N} \phi_{i1} & \cdots & \sum_{i=1}^{N} \phi_{iM} \end{pmatrix} \qquad \boldsymbol{A}^T\boldsymbol{h} = \begin{pmatrix} \sum_{i=1}^{N} \phi_{i1}h_i \\ \vdots \\ \sum_{i=1}^{N} \phi_{iM}h_i \end{pmatrix}$$

$$\boldsymbol{P}^T\boldsymbol{P} = \begin{pmatrix} \sum_{i=1}^{N} x_i^2 & \sum_{i=1}^{N} x_iy_i & \sum_{i=1}^{N} x_i \\ \sum_{i=1}^{N} y_ix_i & \sum_{i=1}^{N} y_i^2 & \sum_{i=1}^{N} y_i \\ \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} y_i & \sum_{i=1}^{N} 1 \end{pmatrix} \qquad \boldsymbol{P}^T\boldsymbol{h} = \begin{pmatrix} \sum_{i=1}^{N} x_ih_i \\ \sum_{i=1}^{N} y_ih_i \\ \sum_{i=1}^{N} h_i \end{pmatrix}$$

It should be noted that additional conditions (3.26) introduce inconsistency to the least squares method. Specifically, the inconsistency is caused by the adding the term $\boldsymbol{\Xi}^T\boldsymbol{\Xi}$ to $\boldsymbol{A}^T\boldsymbol{A}$. Therefore, the mentioned RBF approximation with the linear reproduction is inconveniently formulated as it mixes variables which have a different physical meaning. Thus, we propose another approach in the following section.

3.3.4 Proposed RBF Approximation with Polynomial Reproduction

The method which was introduced in Section 3.3.3 can establish inconsistency to the RBF approximation method as mentioned above. Therefore, we introduce

another approach [MS16a] in this section.

The approximated value can be expressed as:

$$f(\boldsymbol{x}) = \sum_{j=1}^{M} c_j \phi(\|\boldsymbol{x} - \boldsymbol{\xi}_j\|) + P_k(\boldsymbol{x}). \tag{3.29}$$

where $\boldsymbol{\xi}_j$ are reference points specified by a user. The approximating function $f(\boldsymbol{x})$ is represented as a sum of M RBFs, each associated with a different reference point $\boldsymbol{\xi}_j$, and weighted by an appropriate coefficient c_j, and $P_k(\boldsymbol{x})$ is polynomial function of degree k. For simplicity, we assume that we have linear polynomial:

$$P_1(\boldsymbol{x}) = \boldsymbol{a}^T\boldsymbol{x} + a_0. \tag{3.30}$$

However, note that this approach is generally applicable for the arbitrary polynomial function.

Now it can be seen that for $\mathbb{E}^2$, linear polynomial and given dataset, we get following overdetermined linear system of equations:

$$\boldsymbol{A}\boldsymbol{c} + \boldsymbol{P}\boldsymbol{k} = \boldsymbol{h}, \tag{3.31}$$

where $A_{ij} = \phi(\|\boldsymbol{x}_i - \boldsymbol{\xi}_j\|)$ is entry of matrix in the $i-$th row and $j-$th column, $\boldsymbol{c} = (c_1, \ldots, c_M)^T$ is the vector of weights, $\boldsymbol{P}_i = (\boldsymbol{x}_i^T, 1)^T$ is the vector, $\boldsymbol{k} = (\boldsymbol{a}^T, a_0)^T$ is vector of coefficient for linear polynomial and $\boldsymbol{h} = (h_1, \ldots, h_N)^T$ is vector of values in the given points.

The error is then defined as:

$$R = \|\boldsymbol{A}\boldsymbol{c} + \boldsymbol{P}\boldsymbol{k} - \boldsymbol{h}\|, \tag{3.32}$$

then

$$\begin{aligned} R^2 &= (\boldsymbol{A}\boldsymbol{c} + \boldsymbol{P}\boldsymbol{k} - \boldsymbol{h})^T(\boldsymbol{A}\boldsymbol{c} + \boldsymbol{P}\boldsymbol{k} - \boldsymbol{h}) \\ &= \boldsymbol{c}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{c} + \boldsymbol{k}^T\boldsymbol{P}^T\boldsymbol{P}\boldsymbol{k} + 2\boldsymbol{c}^T\boldsymbol{A}^T(\boldsymbol{P}\boldsymbol{k} - \boldsymbol{h}) - 2\boldsymbol{k}^T\boldsymbol{P}^T\boldsymbol{h} + \boldsymbol{h}^T\boldsymbol{h}. \end{aligned} \tag{3.33}$$

Our goal is to minimize the square of error, i.e. find the minimum of R^2 (3.33). This minimum is obtained by differentiating of equation (3.33) with respect to $\boldsymbol{c}$ and $\boldsymbol{k}$ and finding the zeros of those derivatives. It leads to equations:

$$\begin{aligned} \frac{\partial R^2}{\partial \boldsymbol{c}} &= 2(\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{P}\boldsymbol{k} - \boldsymbol{A}^T\boldsymbol{h}) = \boldsymbol{0}, \\ \frac{\partial R^2}{\partial \boldsymbol{k}} &= 2(\boldsymbol{P}^T\boldsymbol{A}\boldsymbol{c} + \boldsymbol{P}^T\boldsymbol{P}\boldsymbol{k} - \boldsymbol{P}^T\boldsymbol{h}) = \boldsymbol{0}, \end{aligned} \tag{3.34}$$

which leads to a system of linear equations:

$$\begin{pmatrix} \boldsymbol{A}^T\boldsymbol{A} & \boldsymbol{A}^T\boldsymbol{P} \\ \boldsymbol{P}^T\boldsymbol{A} & \boldsymbol{P}^T\boldsymbol{P} \end{pmatrix} \begin{pmatrix} \boldsymbol{c} \\ \boldsymbol{k} \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}^T\boldsymbol{h} \\ \boldsymbol{P}^T\boldsymbol{h} \end{pmatrix}, \tag{3.35}$$

i.e.

$$\boldsymbol{Q}\boldsymbol{\gamma} = \boldsymbol{f}. \tag{3.36}$$

The matrix $\boldsymbol{Q}$ is a $(M+3) \times (M+3)$ symmetric positively semidefinite matrix. The equation (3.35) can be expressed in the form:

$$\begin{pmatrix} \sum_{i=1}^{N} \phi_{i1}\phi_{i1} & \cdots & \sum_{i=1}^{N} \phi_{i1}\phi_{iM} & \sum_{i=1}^{N} \phi_{i1}x_i & \sum_{i=1}^{N} \phi_{i1}y_i & \sum_{i=1}^{N} \phi_{i1} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \sum_{i=1}^{N} \phi_{iM}\phi_{i1} & \cdots & \sum_{i=1}^{N} \phi_{iM}\phi_{iM} & \sum_{i=1}^{N} \phi_{iM}x_i & \sum_{i=1}^{N} \phi_{iM}y_i & \sum_{i=1}^{N} \phi_{iM} \\ \sum_{i=1}^{N} x_i\phi_{i1} & \cdots & \sum_{i=1}^{N} x_i\phi_{iM} & \sum_{i=1}^{N} x_i^2 & \sum_{i=1}^{N} x_iy_i & \sum_{i=1}^{N} x_i \\ \sum_{i=1}^{N} y_i\phi_{i1} & \cdots & \sum_{i=1}^{N} y_i\phi_{iM} & \sum_{i=1}^{N} y_ix_i & \sum_{i=1}^{N} y_i^2 & \sum_{i=1}^{N} y_i \\ \sum_{i=1}^{N} \phi_{i1} & \cdots & \sum_{i=1}^{N} \phi_{iM} & \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} y_i & \sum_{i=1}^{N} 1 \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_M \\ a_x \\ a_y \\ a_0 \end{pmatrix}$$

$$= \begin{pmatrix} \sum_{i=1}^{N} \phi_{i1}h_i & \cdots & \sum_{i=1}^{N} \phi_{iM}h_i & \sum_{i=1}^{N} x_ih_i & \sum_{i=1}^{N} y_ih_i & \sum_{i=1}^{N} h_i \end{pmatrix}^T \tag{3.37}$$

where $\phi_{ij} = \phi(\|\boldsymbol{x}_i - \boldsymbol{\xi}_j\|)$, point $\boldsymbol{x}_i = (x_i, y_i)^T$ and vector $\boldsymbol{a} = (a_x, a_y)^T$. It can be seen that this approach eliminated the inconsistency introduced in the Section 3.3.3.

3.4 RBF Approximation for Large Data

In practice, the real datasets contain a large number of points which results into high memory requirements for storing the matrix $\boldsymbol{A}$ of the overdetermined linear system of equations (3.16). For example, when dataset which contains $3,000,000$ points, number of reference points is $10,000$, and the double precision floating point is used, then we need 223.5 GB memory for storing the matrix $\boldsymbol{A}$ of the overdetermined linear system of equations (3.16). Unfortunately, we do not have an unlimited capacity of RAM memory, therefore, the calculation of unknown weights c_j for RBF approximation would be prohibitively computationally expensive due to memory swapping, etc. In this section, a proposed solution to this problem [MS16b] is described.

In Section 3.3.1, it was introduced that overdetermined system of equations can be

solved by the least squares method. For this method the square $M \times M$ matrix:

$$\boldsymbol{B} = \boldsymbol{A}^T \boldsymbol{A} \tag{3.38}$$

is to be determined. Advantages of matrix $\boldsymbol{B}$ are that it is a symmetric matrix and, moreover, only two vectors of length N are needed for determination of one entry, i.e.:

$$b_{ij} = \sum_{k=1}^{N} \phi_{ki} \cdot \phi_{kj}, \tag{3.39}$$

where b_{ij} is entry of matrix $\boldsymbol{B}$ in the $i-$th row and $j-$th column.

To save memory requirements and data bus (PCI) load, block operations with matrices are used. Based on the above properties of matrix $\boldsymbol{B}$, only the upper triangle of this matrix is computed. Moreover, the matrix is partitioned into $M_B \times M_B$ blocks, see Figure 3.5, and the calculation is performed sequentially for each block:

$$\begin{gathered} \boldsymbol{B}_{kl} = (\boldsymbol{A}_{*,k})^T (\boldsymbol{A}_{*,l}) \\ k = 1, \ldots, \frac{M}{M_B}, \qquad l = k, \ldots, \frac{M}{M_B}, \end{gathered} \tag{3.40}$$

where $\boldsymbol{B}_{kl}$ is sub-matrices in the $k-$th row and $l-$th column and $\boldsymbol{A}_{*,k}$ is defined as:

$$\boldsymbol{A}_{*,k} = \begin{pmatrix} \phi_{1,(k-1)\cdot M_B+1} & \cdots & \phi_{1,k\cdot M_B} \\ \vdots & \ddots & \vdots \\ \phi_{i,(k-1)\cdot M_B+1} & \cdots & \phi_{i,k\cdot M_B} \\ \vdots & \ddots & \vdots \\ \phi_{N,(k-1)\cdot M_B+1} & \cdots & \phi_{N,k\cdot M_B} \end{pmatrix}. \tag{3.41}$$

The size of block M_B is chosen so that M_B is multiple of M and there is no swapping, i.e.:

$$M_B \cdot (M_B + 2 \cdot N) \cdot prec < \text{size of RAM [B]}, \tag{3.42}$$

where *prec* is the size of data type in bytes.

Note that this approach is possible to modify easily for RBF approximation with polynomial reproduction Section 3.3.3 and Section 3.3.4.

3.5 Experimental Results of RBF Approximation

The presented methods of the RBF approximation have been tested on synthetic and real datasets. Moreover, different radial basis functions with shape parameter

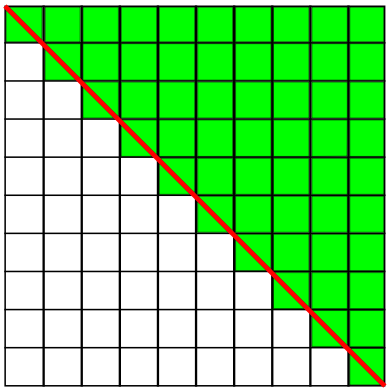


Figure 3.5: $M \times M$ square matrix which is partitioned into $M_B \times M_B$ blocks. The color red is used to denote the main diagonal of matrix and illustrates the symmetry of matrix. The color green is used to denote the blocks which must be computed.

α and different sets of reference points have been used for testing. These types of distribution are described in Section 3.5.1. The presented results are based on results which are introduced in [MS16a], [MS16b], [Maj16] and [MS].

3.5.1 Types of Reference Points Distribution

The following sets of reference points were used for experiments on synthetic datasets:

Points on regular grid
: This set contains the points on a regular grid in $\mathbb{E}^2$.

Epsilon points
: This distribution of reference points is described in the following text.

Epsilon points + AABB corners
: This set of points is determined in the same manner as the previous case. Moreover, the corners of axis aligned bounding box (AABB) of Epsilon points are added to the set of reference points.

Halton points
: This distribution of points is described in the following text in detail. However, note that this set of reference points equals the subset of the given dataset, for which we determine the RBF approximation.

Halton points + AABB corners
: This set of reference points is determined in the same manner as Halton points. Moreover, the corners of AABB are added to this set.

Halton points

Construction of a Halton sequence is based on a deterministic method. This sequence generates well-spaced "draws" points from the interval $[0, 1]$. The sequence uses a prime number as its base and is constructed based on finer and finer prime-based divisions of sub-intervals of the unit interval. The Halton sequence [Fas07] can be described by the following recurrence formula:

$$Halton(p)_k = \sum_{i=0}^{\lfloor \log_p k \rfloor} \frac{1}{p^{i+1}} \left(\left\lfloor \frac{k}{p^i} \right\rfloor \bmod p \right), \tag{3.43}$$

where p is the prime number and k is the index of the calculated element.

For the $\mathbb{E}^2$ space, subsequent prime numbers are used as a base. In this test, $\{2, 3\}$ were used for the Halton sequence and the following sequence of points in a rectangle (a, b) was derived:

$$\begin{aligned} Halton(2,3) = \Big\{ &\left(\frac{1}{2}a, \frac{1}{3}b\right), \left(\frac{1}{4}a, \frac{2}{3}b\right), \left(\frac{3}{4}a, \frac{1}{9}b\right), \left(\frac{1}{8}a, \frac{4}{9}b\right), \left(\frac{5}{8}a, \frac{7}{9}b\right), \\ &\left(\frac{3}{8}a, \frac{2}{9}b\right), \left(\frac{7}{8}a, \frac{5}{9}b\right), \left(\frac{1}{16}a, \frac{8}{9}b\right), \left(\frac{9}{16}a, \frac{1}{27}b\right), \dots \Big\}, \end{aligned} \tag{3.44}$$

where a is the width of the rectangle and b is the height of the rectangle.

Visualization of the dataset with 10^3 points of the Halton sequence from (3.44) can be seen in Figure 3.6. We can see that the Halton sequence in $\mathbb{E}^2$ space covers this space more evenly than randomly distributed uniform points in the same rectangle.

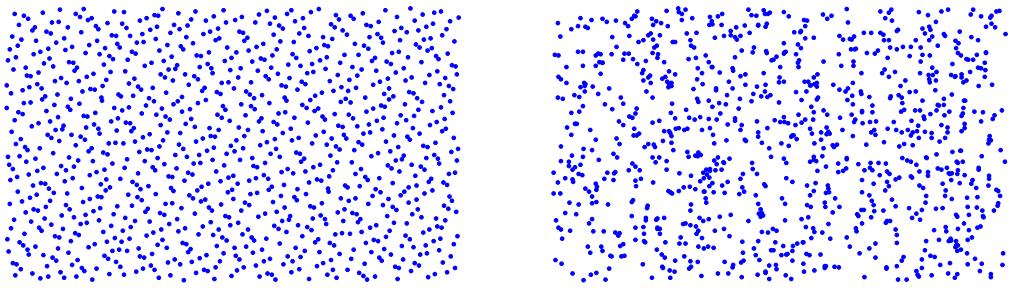


Figure 3.6: Halton points in $\mathbb{E}^2$ generated by $Halton(2, 3)$ (left) and random points in a rectangle with uniform distribution (right). The number of points is 10^3 in both cases.

Epsilon points

This is a special distribution of points in $\mathbb{E}^2$, which is based on a regular grid. Each point is determined as follows:

$$\begin{aligned} P_{ij} = \Big[& i \cdot \Delta x + \text{rand}(-\varepsilon_x, \varepsilon_x), j \cdot \Delta y + \text{rand}(-\varepsilon_y, \varepsilon_y) \Big], \\ & \varepsilon_x \approx 0.25 \cdot \Delta \text{x}, \qquad i = 0, \ldots, N_x, \\ & \varepsilon_y \approx 0.25 \cdot \Delta \text{y}, \qquad j = 0, \ldots, N_y, \end{aligned} \tag{3.45}$$

where Δx and Δy are real numbers representing the grid spacing, N_x indicates the number of grid columns, N_y is the number of grid rows and $\text{rand}(-\varepsilon_x, \varepsilon_x)$ or $\text{rand}(-\varepsilon_y, \varepsilon_y)$ is a random drift with a uniform distribution from $-\varepsilon_x$ to ε_x or from $-\varepsilon_y$ to ε_y.

Figure 3.7 presents the dataset with 40×25, (i.e. 10^3) epsilon points. Moreover, we can see the comparison of this distribution of points with points on a regular grid.

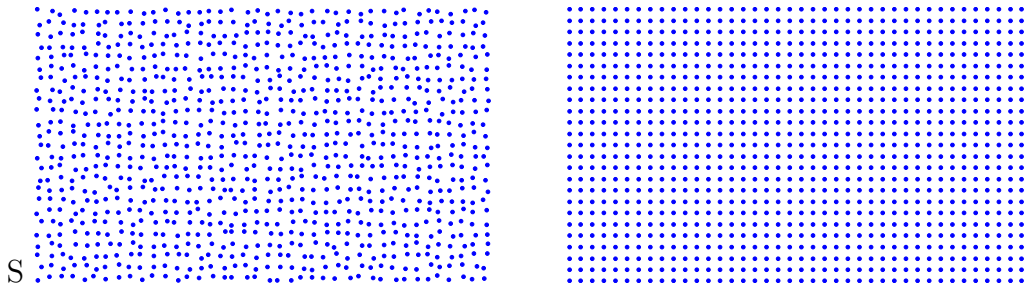


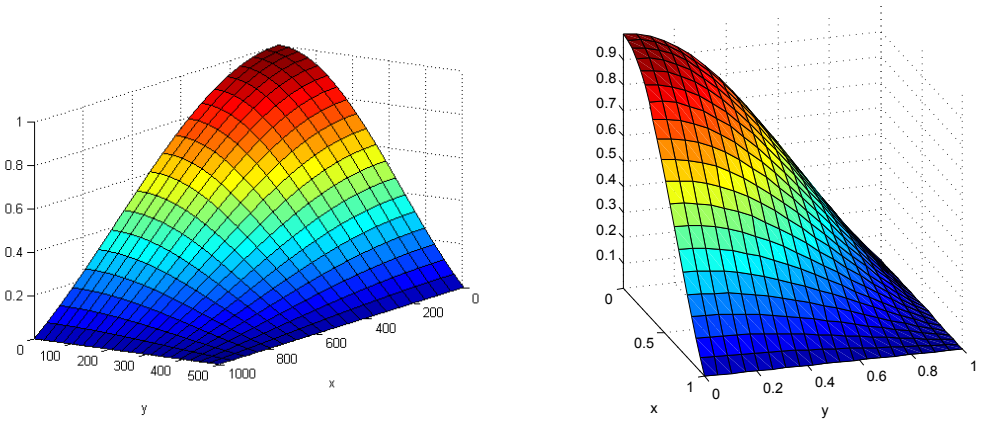
Figure 3.7: Epsilon points (left) and points on a $2D$ regular grid (right). The number of points is $40 \times 25 = 10^3$ in both cases.

3.5.2 Synthetic Datasets

A Halton distribution of points was used for synthetic data. Moreover, each point from such a dataset is associated with a function value at this point. For this purpose, different functions have been used for experiments. Results for those two functions are presented here. The first is a $2D$ sinc function, see Figure 3.8, and the second is Franke's function [Fra79], see Figure 3.9.

Comparison of RBF Approximation Methods with Polynomial Reproduction

In this section, the original RBF approximation with polynomial reproduction, see Section 3.3.3, and proposed RBF approximation with polynomial reproduction,



(a) $s_x = 1000$, $s_y = 500$ (b) $s_x = 1$, $s_y = 1$

Figure 3.8: 2D sinc function $\text{sinc}(\frac{\pi x}{s_x}) \cdot \text{sinc}(\frac{\pi y}{s_y})$ whose domain is restricted to $[0, s_x] \times [0, s_y]$.

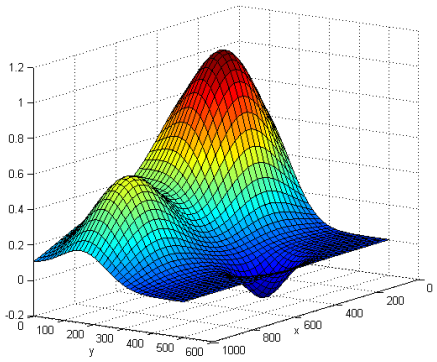
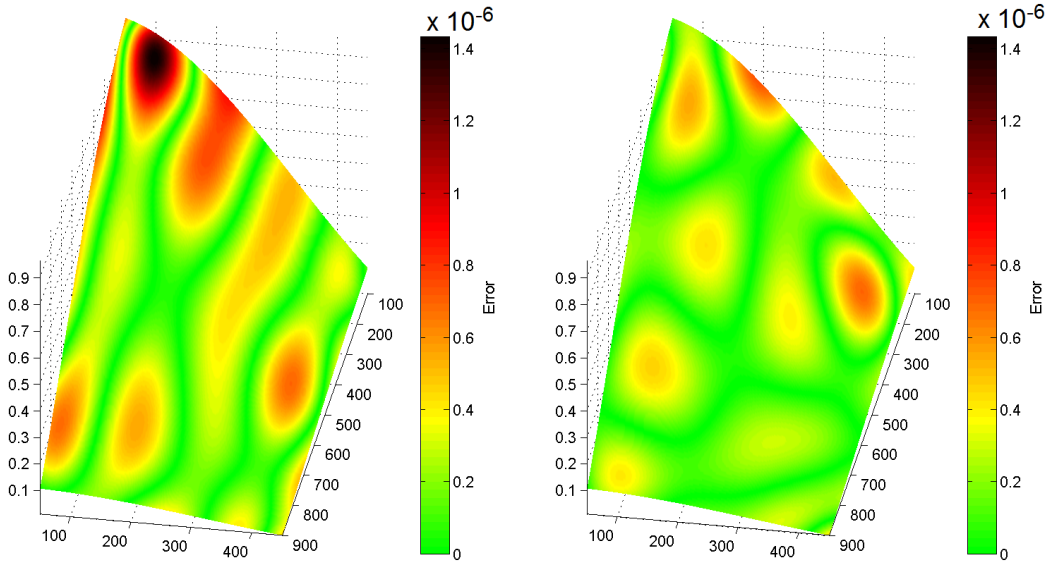


Figure 3.9: Franke's function whose domain is restricted to $[0, 1000] \times [0, 600]$.

see Section 3.3.4, are compared. An example of the RBF approximation of 1089 Halton data points sampled from a 2D sinc function, for a Halton set of reference points which consists of 81 points, using both approaches is shown in Figure 3.10. The graphs are false-colored according to the magnitude of the error.

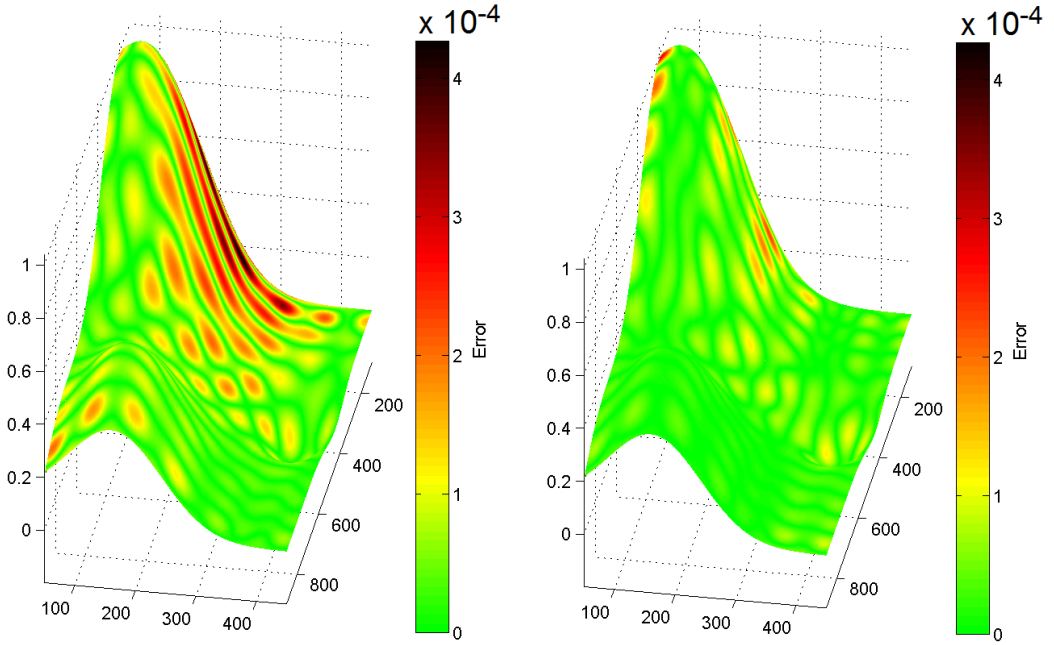
A further example of the RBF approximation of 4225 Halton data points sampled from a Franke's function and for a set of reference points which consists of 289 points on a regular grid, using both approaches is shown in Figure 3.11. The graphs are again false-colored by magnitude of the error.

It can be seen that the original RBF approximation with a linear reproduction returns a worse result in terms of the error in comparison with the proposed RBF approximation with a linear reproduction. Moreover, we can see from Figure 3.10



(a) Original approach (b) Proposed approach

Figure 3.10: Approximation of 1089 data points sampled from a $2D$ sinc function, i.e. $\text{sinc}\left(\frac{\pi x}{1000}\right) \cdot \text{sinc}\left(\frac{\pi y}{500}\right)$, where $(x, y) \in [0, 1000] \times [0, 500]$, with 81 Halton-spaced Gaussian functions with $\alpha = 0.001$, false-colored by magnitude of the error.



(a) Original approach (b) Proposed approach

Figure 3.11: Approximation of 4225 points sampled from a Franke's function with 289 regularly spaced IQ with $\alpha = 0.005$, false-colored by magnitude of the error.

and Figure 3.11 that for the presented cases **the maximum magnitude of the error for the original approach is approximately two times greater than the maximum magnitude of the error for the proposed approach**.

This raises the question how the RBF approximation depends on the shape parameter α selection. Many papers have been published about choosing the optimal shape parameter α, e.g. [Fra82], [Rip99], [FZ07], [Sch11]. In the following, a comparison depending on the choice of shape parameter α is performed.

Figure 3.12 presents the ratio of mean error of the original RBF approximation

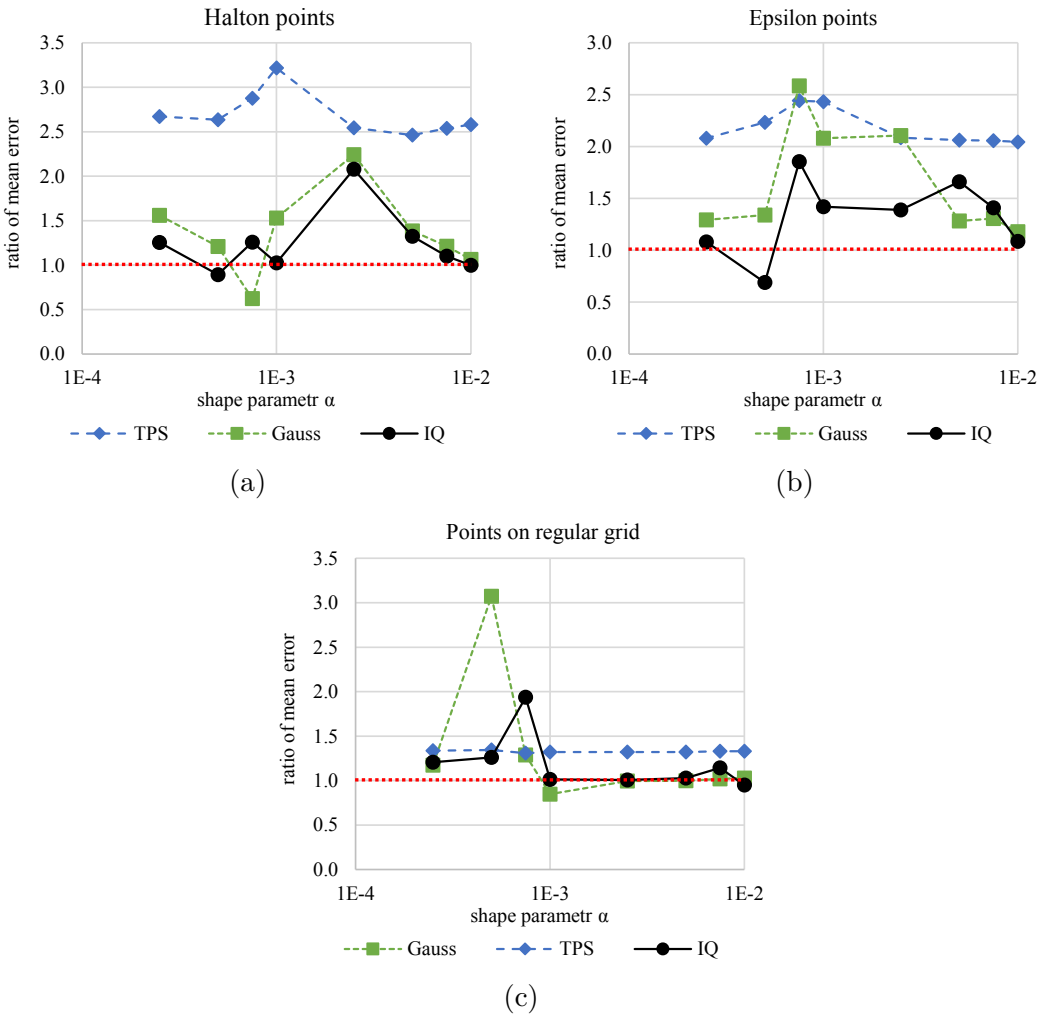


Figure 3.12: The ratio of the mean error of the original approach to the mean error of the proposed approach of RBF approximation of 1089 data points sampled from a $2D$ sinc function with 81 reference points for different RBFs and different shape parameters. The used sets of reference points are: Halton points (a), Epsilon points (b) and points on a regular grid (c).

with the linear reproduction to the mean error of the proposed RBF approximation with the linear reproduction, i.e.:

$$ratio = \frac{mean\ error_{original}}{mean\ error_{proposed}}, \tag{3.46}$$

for a dataset which consists of 1089 Halton points in the range $[0, 1000] \times [0, 500]$, sampled from a 2D sinc function. The set of reference points contains 81 points with different behavior of the distribution, and for different global RBFs. Graphs in Figure 3.12 represent the experimentally obtained ratio according to the shape parameter α of the used RBFs.

We can see that for the TPS, the mean errors of the proposed approach are significantly smaller than those of the original approach (ratio is greater than one). Furthermore, this ratio is not significantly different for the different shape parameters α. The proposed RBF approximation gives better results than the original approach in terms of the mean error for the Gaussian function and epsilon reference points. The proposed approach is also better, with five exceptions, in the remaining cases.

The experiments prove that the proposed approach to the RBF approximation is correct and gives better and more stable results than the original approach, see Section 3.3.3. Therefore, in the following we will use only proposed approach, see Section 3.3.4.

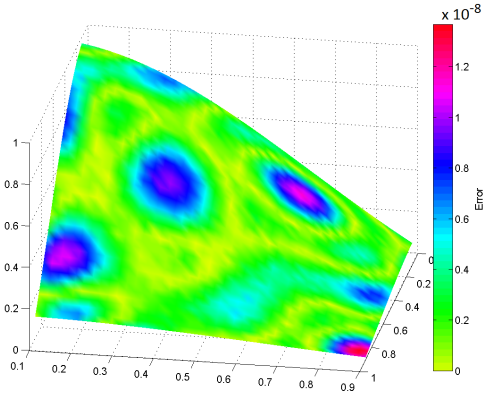
Examples of RBF Approximation Results

Some examples of the RBF approximation to 1089 Halton data points sampled from a $2D$ sinc function, for a Halton set of reference points, which consists of 81 points, and different RBFs are shown in Figure 3.13.

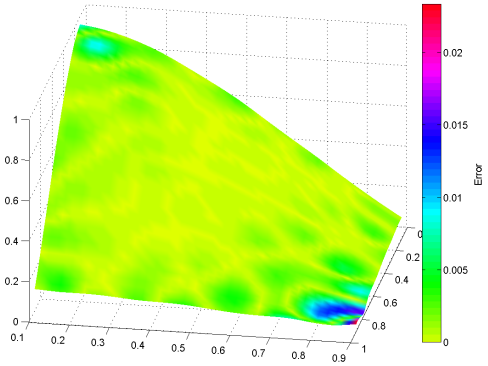
It can be seen that the RBF approximation using Lagrange multiplies (Fasshauer [Fas07]) returns the worst result in terms of the error in comparison with the proposed methods. Moreover, in Figure 3.13 (b), (d) and (f) it can also be seen that using the TPS brings the biggest error to the result.

There is a question how the RBF approximation depends on the α value selection. Consequently, it is described in the following section.

RBF approximation

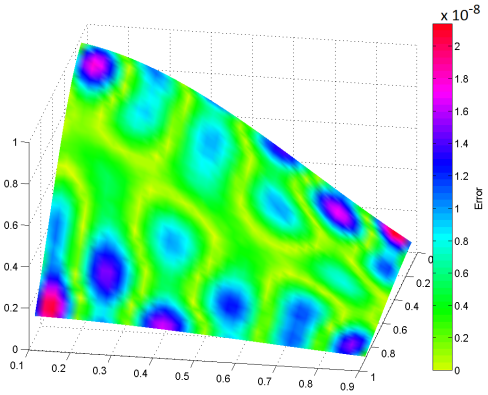


(a) Gauss, $\alpha = 1$, Halton points

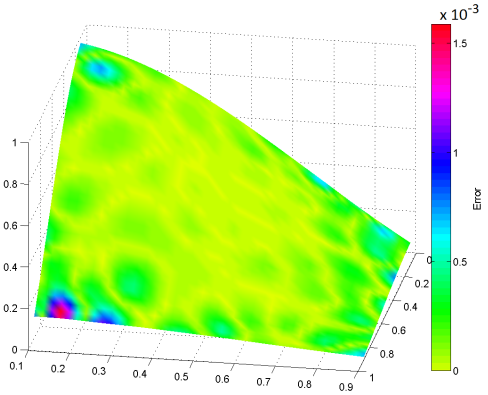


(b) TPS, $\alpha = 1$, Halton points

RBF approximation with linear reproduction

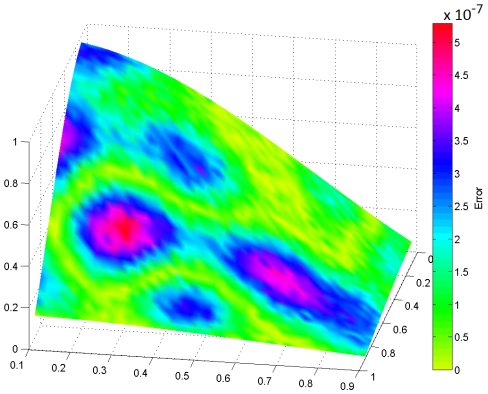


(c) Gauss, $\alpha = 1$, Halton points

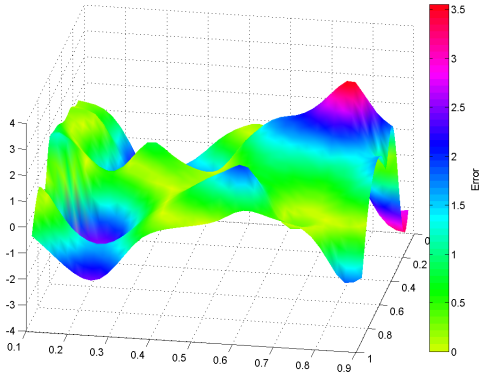


(d) TPS, $\alpha = 1$, Halton points

RBF approximation using Lagrange multipliers



(e) Gauss, $\alpha = 1$, Halton points



(f) TPS, $\alpha = 1$, Halton points

Figure 3.13: Approximation to 1089 data points sampled from a $2D$ sinc function with 81 Halton spaced basis functions false-colored by magnitude of error. Note, the scale is changing.

Comparison of Methods

In this section, the different versions of RBF approximation which were presented in Section 3.3.1, Section 3.3.2 and Section 3.3.4 are compared. Figure 3.14 presents the mean error of the RBF approximation for the dataset, which consists of 1089 Halton points in the range $[0, 1] \times [0, 1]$, sampled from a $2D$ sinc function, while the set of reference points contains 81 points with Halton behavior of the distribution, and for different global radial basis functions. The graphs represent the mean error according to a shape parameter α of used RBFs. We can see that we obtain a higher mean error for the RBF approximation using Lagrange multipliers (Fasshauer [Fas07]). Mean errors for RBF approximation and RBF approximation with linear reproduction are almost the same. Moreover, the

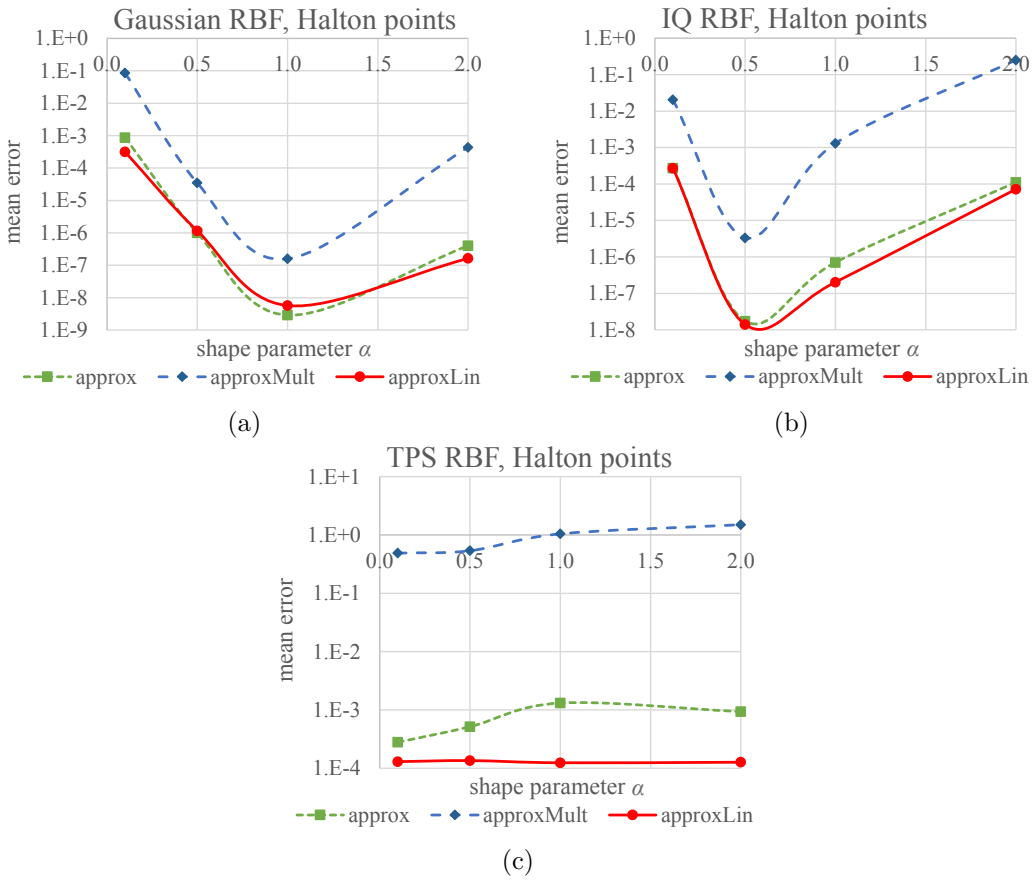


Figure 3.14: The mean error of approximation to 1089 data points sampled from a $2D$ sinc function with 81 reference Halton points for different RBF approximation methods, different RBFs and different shape parameters. The used approximation methods are: RBF approximation (approx), RBF approximation using Lagrange multipliers (approxMult) and proposed RBF approximation with linear reproduction (approxLin). RBFs are: (a) Gauss function, (b) IQ, (c) TPS.

Gaussian RBF gives the best result for shape parameter $\alpha = 1$ and the inverse quadric for $\alpha = 0.5$. Further, the TPS function is not appropriate to solve the given problem, see Figure 3.14 (c). Note, the standard deviation of errors was also measured and the same behavior and order of magnitude was obtained as for the mean errors.

Comparison by Placement of the Dataset in $\mathbb{E}^2$

This section is focused on a placement of the actual dataset in the domain space, and the used function generating associated scalar values in $\mathbb{E}^2$. The given dataset has a range of one in both axes and the function generating associated scalar values is a $2D$ sinc function. Two configurations for the placement of the origin of the dataset and the maximum of the $2D$ sinc function were used. The first configuration is at point $(0; 0)$; the second is moved to point $(3,951,753; 2,785,412)$.

Figure 3.15 presents the mean error for these configurations, when the Gaussian basis functions and Halton set of reference points were chosen. We can see that the proposed RBF approximation with linear reproduction gives a higher error for the second configuration, i.e. the placement at point $(3,951,753; 2,785,412)$. The decision is not ambiguous for RBF approximation using Lagrange multiplier. Note that a graph for the RBF approximation is not presented because both configurations give the same results.

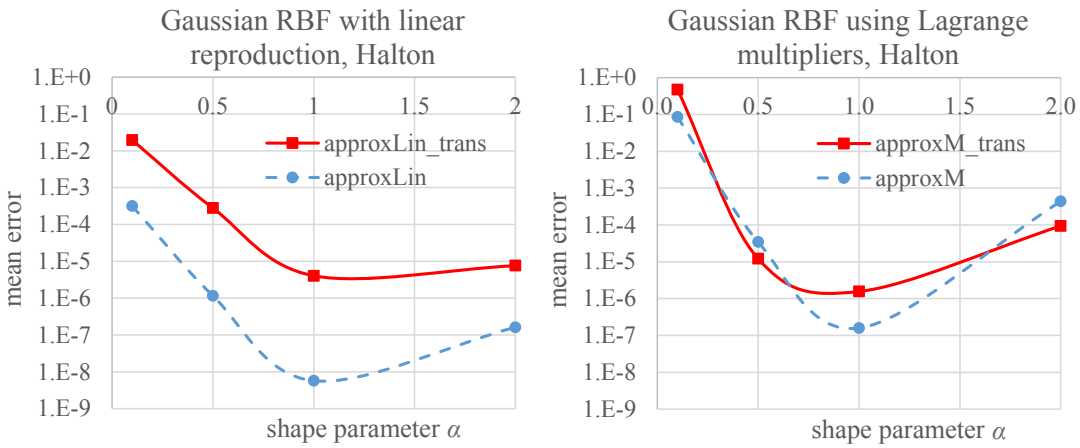


Figure 3.15: The mean error of approximation to 1089 data points sampled from a $2D$ sinc function with 81 spaced Gaussian basis functions for a Halton set of reference points, different RBF approximation methods and different shape parameters. The placement of the given dataset and the maximum of the $2D$ sinc function are at point $(0; 0)$ (circles) or at point $(3,951,753; 2,785,412)$ (squares). Versions of approximation are proposed RBF approximation with linear reproduction (left) and RBF approximation using Lagrange multipliers (right).

Comparison of Different Distribution of Reference Points

In this section, we focus on a comparison of the presented RBF approximation methods due to used distribution of reference points. Measurements of errors were performed for different type of RBFs with different shape parameters. Mean error according to shape parameter α for Gaussian RBF is presented in Figure 3.16.

We can see that for all versions of RBF approximation the worst result is obtained

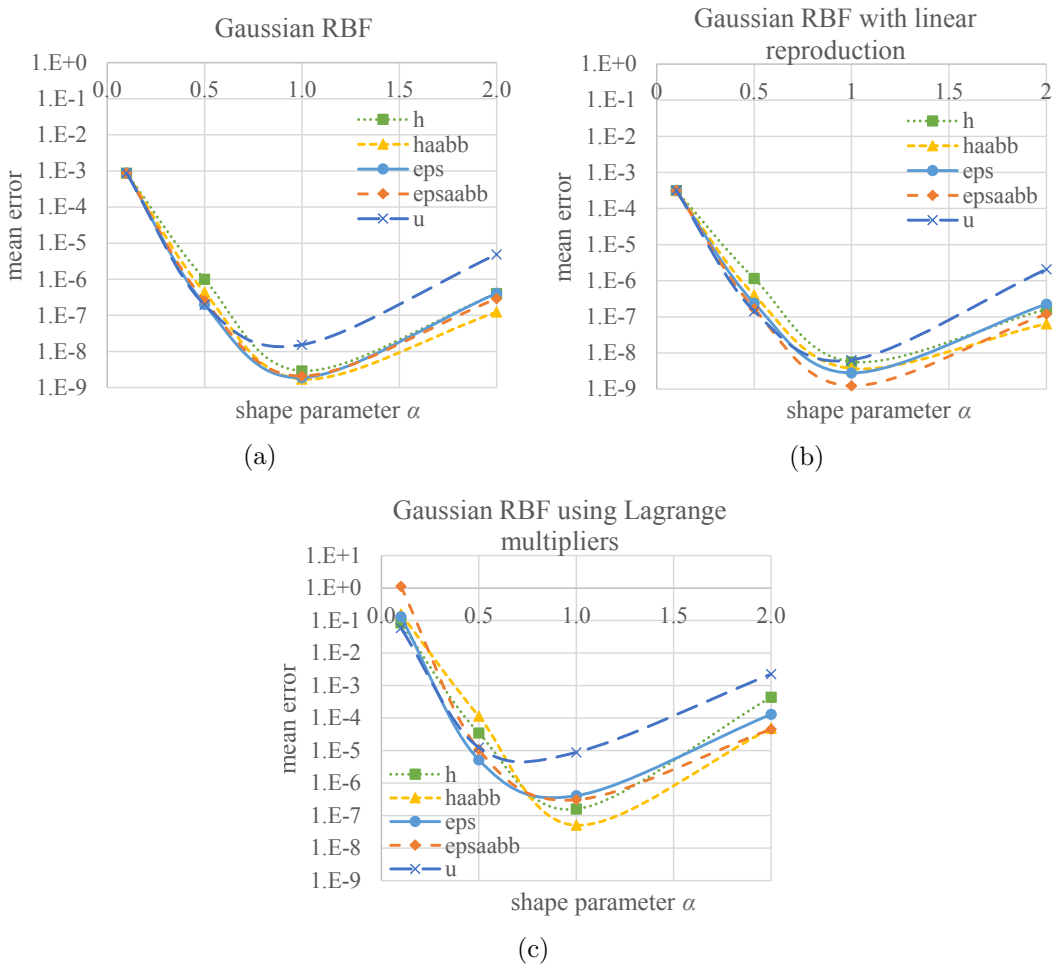


Figure 3.16: The mean error of approximation to 1089 data points sampled from a $2D$ sinc function with 81 spaced Gaussian basis functions for different RBF approximation methods, different shape parameters and different sets of reference points. The sets of reference points are: Halton points (h), Halton points + AABB (haabb), epsilon points (eps), epsilon points + AABB (epsaabb), points on a regular grid (u). Their description is in Section 3.5.1. Versions of approximation are: (a) RBF approximation, (b) proposed RBF approximation with linear reproduction, (c) RBF approximation using Lagrange multipliers.

for reference points on a regular grid (u). For the RBF approximation, the remaining sets of reference points give almost the same results. Reference points corresponding to epsilon points + AABB (epsaabb) almost always give the best result for proposed RBF approximation with linear reproduction. For RBF approximation using Lagrange multipliers, the best results are for the reference points which have a Halton distribution.

Optimal Number of Reference Points

This section focuses on the influence of the number of reference points. The number of reference points is determined relative to the number of points in the given dataset. Measurements for different shape parameters were performed many times, and average mean errors were computed, see Figure 3.17 - Figure 3.19. Note that the reference points were distributed by Halton distribution. Figure 3.17 presents the mean error for the Gaussian RBF approximation. Experimental results for the IQ are shown in Figure 3.18. We can see that the mean errors are almost constant for the small shape parameter α. However, the mean error decreases with the increasing number of reference points for greater shape parameters.

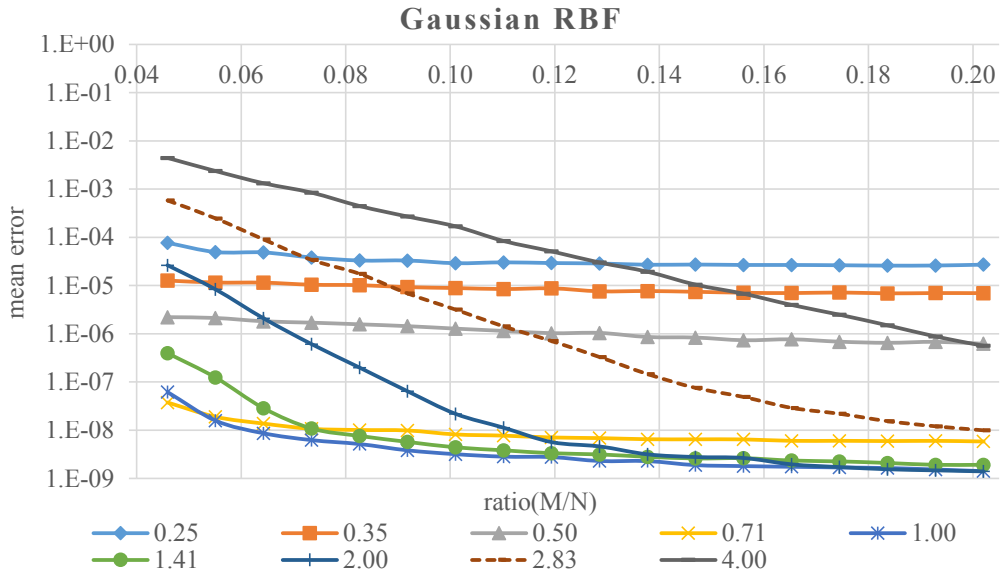


Figure 3.17: The mean error of the RBF approximation to 1089 data points sampled from a $2D$ sinc function for different numbers of reference points, Gaussian RBF with different shape parameters α.

Figure 3.19 presents experimental results obtained for TPS. We can see that the mean error decreases with the increasing number of reference points as would be expected.

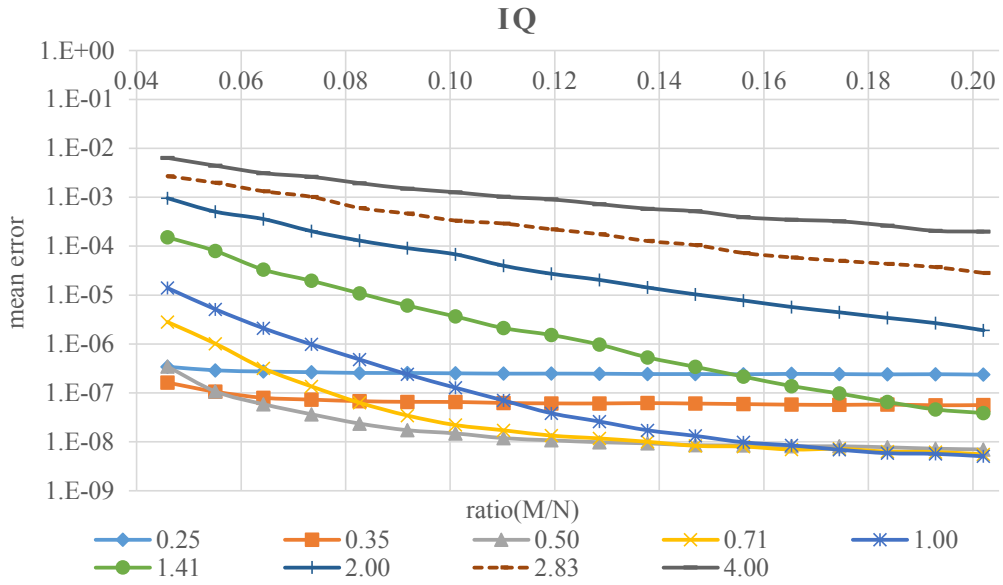


Figure 3.18: The mean error of the RBF approximation to 1089 data points sampled from a $2D$ sinc function for different numbers of reference points, IQ RBF with different shape parameters α.

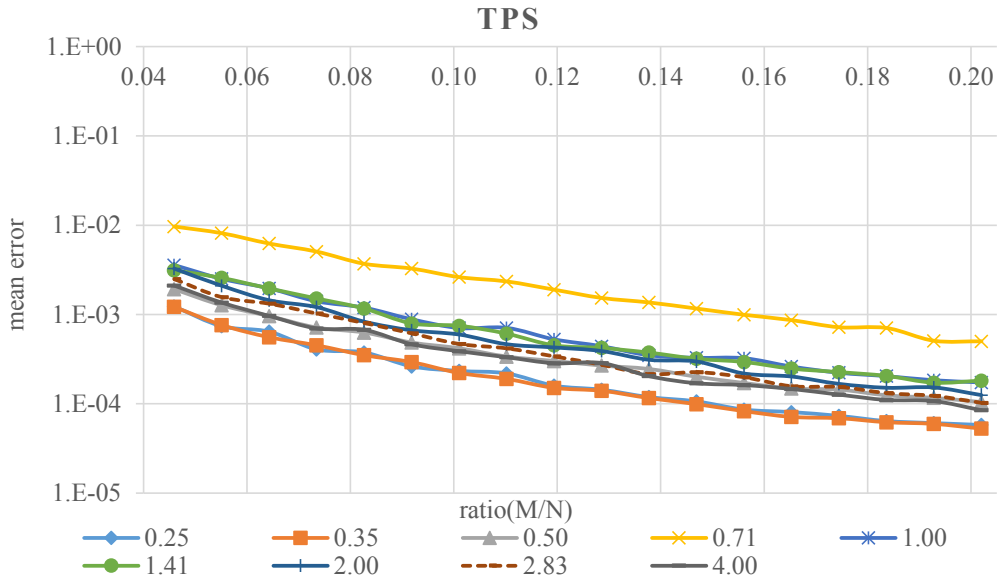


Figure 3.19: The mean error of the RBF approximation to 1089 data points sampled from a $2D$ sinc function for different numbers of reference points, TPS RBF with different shape parameters α.

Finally, note that the results for proposed RBF with linear reproduction are very similar to the RBF approximation. The RBF using Lagrange multipliers has unpredictable behavior and no trend can be established.

Comparison of CS-RBFs

The RBF approximation method presented above, see Section 3.3.1, has been tested on the dataset with 1089 Halton points and each point is associated with a value from Franke's function [Fra79]. The set of reference points is a subset of the given dataset, for which we determine the RBF approximation, and its cardinality is 81. Moreover, reference points are uniformly distributed within a given area.

All CS-RBFs from the catalog of RBFs in [Fas07] (see D.2.7) have been used for experiments. Depending on the quality, the obtained results are divided into three groups. Results are presented for the representative of each group, i.e. Wendland's $\phi_{3,0}$, Wendland's $\phi_{3,1}$ and Wendland's $\phi_{3,3}$, see Table 3.2.

The shape parameters α for used CS-RBFs were determined experimentally with regard to the quality of approximation, and they are presented in Table 3.3.

Table 3.3: Experimentally determined shape parameters α for the synthetic dataset and the used CS-RBFs

CS-RBF	**shape parameter**
Wendland's $\phi_{3,0}$	$\alpha = 0.707$
Wendland's $\phi_{3,1}$	$\alpha = 0.500$
Wendland's $\phi_{3,3}$	$\alpha = 0.250$

In Figure 3.20, the approximations of the given dataset for all CS−RBFs are shown. In this figure, the surfaces are false-colored by the magnitude of the error. It can be seen that for the given dataset the RBF approximation with Wendland's $\phi_{3,3}$ basis function returns the best result in terms of the error. Contrary, the worst result is obtained for RBF approximation with Wendland's $\phi_{3,0}$ basis function.

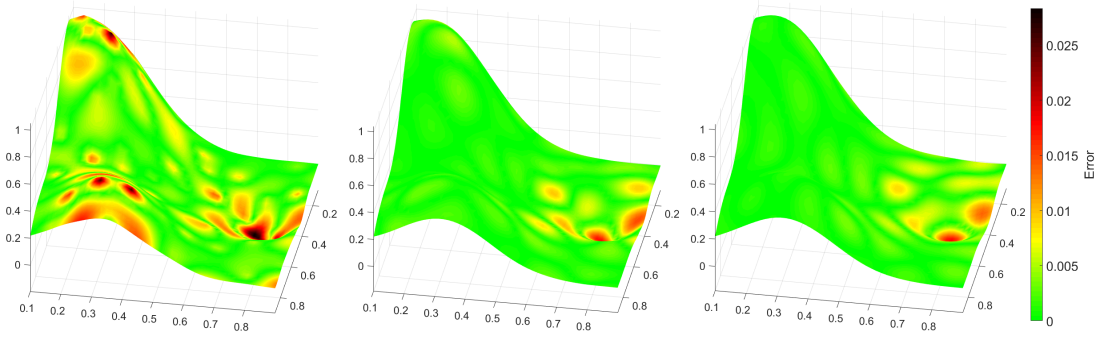


Figure 3.20: Wendland's RBF $\phi_{3,0}$, $\alpha = 0.707$ (left); Wendland's RBF $\phi_{3,1}$, $\alpha = 0.500$ (center) and Wendland's RBF $\phi_{3,3}$, $\alpha = 0.250$ (right)

Table 3.4: The RBF approximation error for the testing dataset and different radial basis functions.

Error	**Wendland's**		
	$\phi_{3,0}$	$\phi_{3,1}$	$\phi_{3,3}$
mean absolute error	0.0041	0.0021	0.0019
deviation of error	1.92E-5	6.06E-6	5.25E-6
mean relative error [%]	0.0151	0.0076	0.0072

Table 3.4 shows three different error measures of the dataset depending on the chosen basis functions: the mean absolute error, the deviation and the mean relative error.

3.5.3 Real Datasets

The presented modification of the RBF approximation for large data Section 3.4 has been tested on real data. Let us introduce results for two real datasets.

The first dataset was obtained from LiDAR data of the Serpent Mound in Adams Country, Ohio[1]. The second dataset is LiDAR data of the Mount Saint Helens in Skamania Country, Washington[1]. Each point of these datasets is associated with its elevation. The summary of the dimensions of terrain for the given datasets is in Table 3.5.

Table 3.5: Summary of the dimensions of terrain for tested datasets. Note that one feet [ft] corresponds to 0.3048 meter [m].

Dimensions	**Serpent Mound**	**St. Helens**
number of points	3,265,110	6,743,176
number of ref. pts.	10,000	10,000
lowest point [ft]	166.7800	3,191.5269
highest point [ft]	215.4800	8,330.2219
width [ft]	1,085.1199	26,232.3696
length [ft]	2,698.9601	35,992.6861

For experiments, two different radial basis functions have been used. These RBFs are Gauss function, see Table 3.1, and Wendland's $\phi_{3,1}$, see Table 3.2. Shape parameters α for used RBFs were determined experimentally with regard to the

[1] http://www.liblas.org/samples/

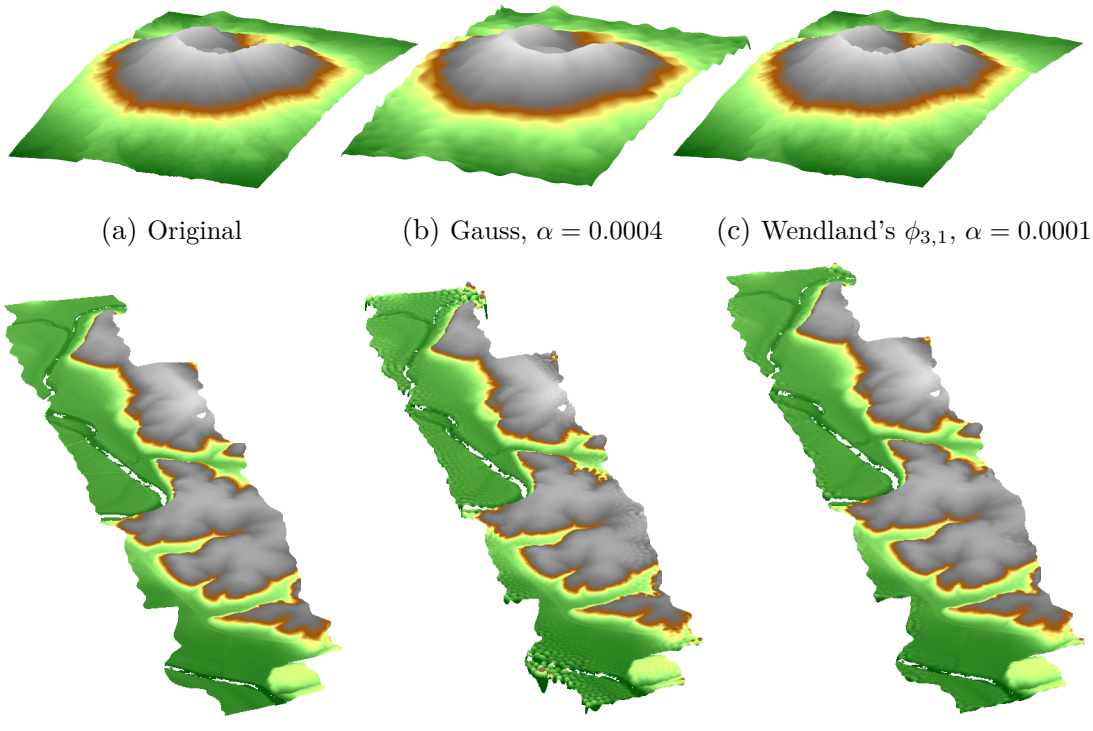
quality of approximation and they are presented in Table 3.6. Note that value of shape parameter α is inversely proportional to range of datasets.

Table 3.6: Experimentally determined shape parameters α for used RBFs

RBF	**shape parameter**	
	Serpent Mound	**St. Helens**
Gaussian RBF	$\alpha = 0.05$	$\alpha = 0.0004$
Wendland's $\phi_{3,1}$	$\alpha = 0.01$	$\alpha = 0.0001$

The set of reference points equals to the subset of the given dataset for which the RBF approximation is determined. Moreover, the distribution of reference points is uniform.

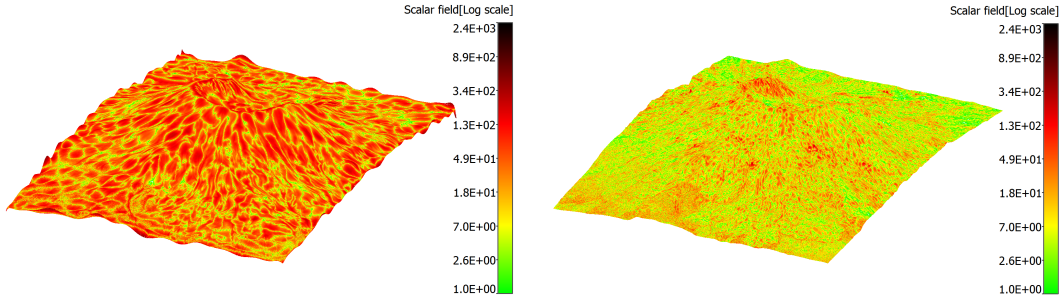
Approximation of Mount Saint Helens for both RBFs and its original are shown in Figure 3.21a-3.21c. In Figure 3.21b, it can be seen that the RBF approximation with the global Gaussian RBFs cannot preserve the sharp rim of a crater. Further, the magnitude of error visualization at each point of the original point cloud is



(a) Original (b) Gauss, $\alpha = 0.0004$ (c) Wendland's $\phi_{3,1}$, $\alpha = 0.0001$

(d) Original (e) Gauss, $\alpha = 0.05$ (f) Wendland's $\phi_{3,1}$, $\alpha = 0.01$

Figure 3.21: Mount Saint Helens is Skamania Country, Washington (top) and Serpent Mound in Adams Country, Ohio (bottom)

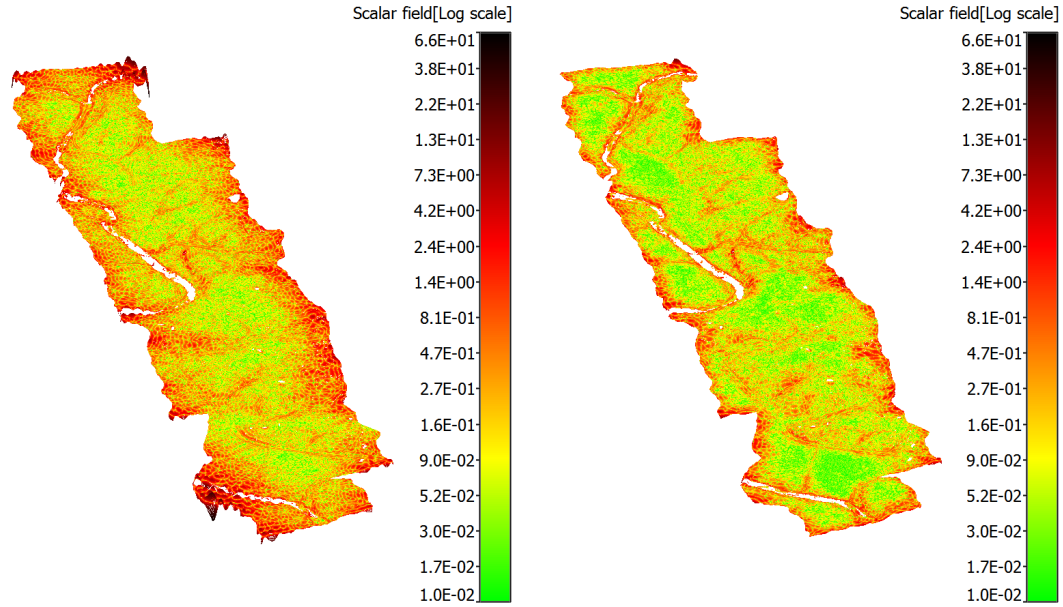


(a) Gauss, $\alpha = 0.0004$ (b) Wendland's $\phi_{3,1}$, $\alpha = 0.0001$

Figure 3.22: Approximation of Mount Saint Helens with 10,000 radial basis functions false-colored by magnitude of error.

presented in Figure 3.22. It can be seen that the RBF approximation with the global Gaussian RBFs returns worse result than RBF approximation with local Wendland's $\phi_{3,1}$ basis functions in terms of the error. Not only, the value of the mean absolute error, but also its deviation and the mean relative error for both approximations can be seen in Table 3.7.

The results of the RBF approximation for Serpent Mound and its original are shown in Figure 3.21d-3.21f. In Figure 3.23 the magnitude of error at each point of original point cloud is visualized. It can be seen that the approximation using



(a) Gauss, $\alpha = 0.05$ (b) Wendland's $\phi_{3,1}$, $\alpha = 0.01$

Figure 3.23: Approximation of the Serpent Mound with 10,000 radial basis functions false-colored by magnitude of error.

Table 3.7: The RBF approximation error for testing datasets and different radial basis functions. Note that one feet [ft] corresponds to 0.3048 meter [m].

Error	**Serpent Mound**		**St. Helens**	
	Gauss	$\phi_{3,1}$	**Gauss**	$\phi_{3,1}$
mean absolute error [ft]	0.4477	0.2289	44.4956	12.1834
deviation of error [ft]	1.4670	0.1943	680.3659	169.2800
mean relative error [%]	0.0024	0.0012	0.0087	0.0023

local Wendland's $\phi_{3,1}$ basis function (Figure 3.21f) returns again better result than approximation using the global Gaussian RBF (Figure 3.21e) in terms of the error.

The mutual comparison of both datasets in terms of the mean relative error (Table 3.7) indicates that the mean relative error for Serpent Mound is smaller than for Mount Saint Helens. It is caused by the presence of vegetation, namely a forest, in LiDAR data of the Mount Saint Helens. This vegetation operates in our RBF approximation as noise, therefore, the resulting mean relative error is higher.

The implementation of the RBF approximation has been performed in Matlab and tested on PC with the following configuration:

- CPU: Intel® Core™ i7-4770 (4× 3.40GHz + hyper-threading),
- memory: 32 GB RAM,
- operating system Microsoft Windows 7 64bits.

For the approximation of the Serpent Mound with $10,000$ local Wendland's $\phi_{3,1}$ basis function with shape parameter $\alpha = 0.01$; the running times for different sizes of blocks were measured. These times were converted relative to the time for 100×100 blocks and are presented in Figure 3.24. We can see that the time performance is large for the approximation matrix which is partitioned into small

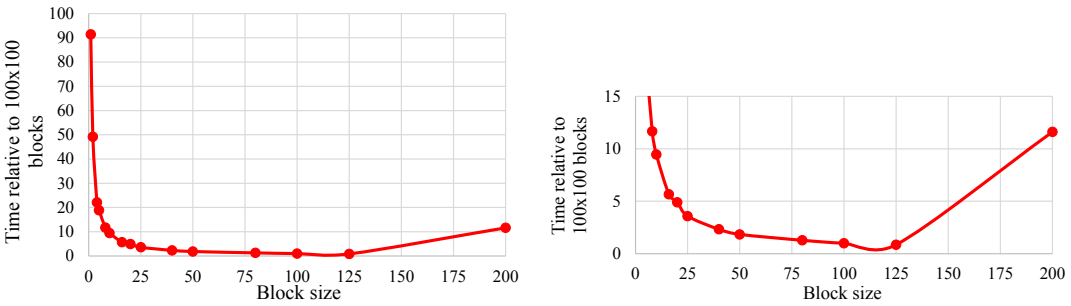


Figure 3.24: Time performance for approximation of the Serpent Mound depending on the block size. The times are presented relative to the time for 100×100 blocks.

blocks (i.e. smaller than 25×25 blocks). This is caused by overhead costs. On the other hand, the running time begins to grow above the permissible limit due to memory swapping for the approximation matrix which is partitioned into large blocks (i.e. larger than 125×125 blocks).

Comparison of CS-RBFs

The proposed RBF approximation for large data, see Section 3.4, has been tested on the dataset which was obtained from LiDAR data of the Serpent Mound in Adams Country, Ohio[1] and each point is associated with its elevation. The set of reference points is a subset of the given dataset, for which the RBF approximation is determined. Moreover, reference points are uniformly distributed within a given area. Table 3.5 gives an overview of the datasets.

All CS-RBFs from the catalog of RBFs in [Fas07] (see D.2.7) have been used for experiments. Depending on the quality, the obtained results are divided into three groups. The results are presented for an representative of each group, i.e. Wendland's $\phi_{3,0}$, Wendland's $\phi_{3,1}$ and Wendland's $\phi_{3,3}$, see Table 3.2.

The shape parameters α for the used CS-RBFs were determined experimentally with regard to the quality of approximation and they are presented in Table 3.8.

Table 3.8: Experimentally determined shape parameters α for the Serpent Mound and the used CS-RBFs

CS-RBF	**shape parameter**
Wendland's $\phi_{3,0}$	$\alpha = 0.0071$
Wendland's $\phi_{3,1}$	$\alpha = 0.0100$
Wendland's $\phi_{3,3}$	$\alpha = 0.0050$

The RBF approximation for the Serpent Mound was solved block-wise where the size of blocks is 100. The results of RBF approximation are shown in Figure 3.25. It illustrates the magnitude of the error at each point of the original points cloud. It can be seen that the RBF approximation with Wendland's $\phi_{3,1}$ basis function returns the best result in terms of the error for the Serpent Mound dataset. The RBF approximation with Wendland's $\phi_{3,3}$ basis function returns the worst results. However, the overall variation of the error is smaller for this configuration. Moreover, we can see that the highest errors occur on the boundary of the terrain for Wendland's $\phi_{3,1}$ and Wendland's $\phi_{3,0}$. Three error measures of the elevation for all used CS-RBFs are gathered in Table 3.9. Moreover, the signed errors for the Serpent Mound dataset and Wendland's $\phi_{3,1}$ basis function are shown in

[1] http://www.liblas.org/samples/

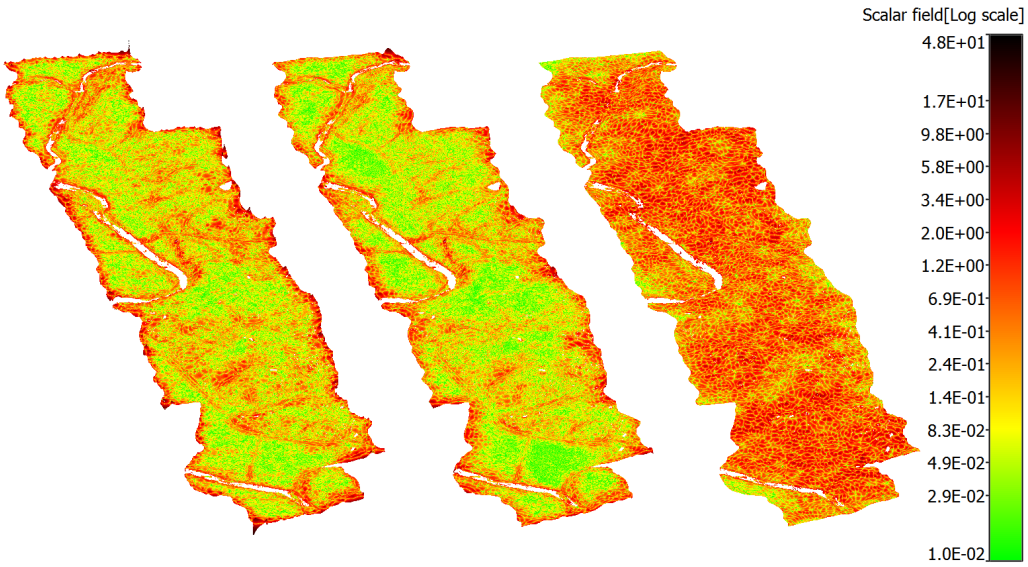


Figure 3.25: Serpent Mound dataset: Wendland's RBF $\phi_{3,0}$, $\alpha = 0.0071$ (left); Wendland's RBF $\phi_{3,1}$, $\alpha = 0.0100$ (center) and Wendland's RBF $\phi_{3,3}$, $\alpha = 0.0050$ (right)

Figure 3.26. We can see that the signs are different at various locations. The similar results are obtained for the rest of experiments.

Figure 3.26: The signed errors for the Serpent Mound dataset and Wendland's RBF $\phi_{3,1}$ with $\alpha = 0.01$: the color red is used to denote the positive error, and the color cyan is used to denote the negative error.

Table 3.9: The RBF approximation error for the Serpent Mound dataset and different radial basis functions. Note that one feet [ft] corresponds to 0.3048 meter [m].

Error	**Wendland's**		
	$\phi_{3,0}$	$\phi_{3,1}$	$\phi_{3,3}$
mean absolute error [ft]	0.3043	0.2289	0.7480
deviation of error [ft]	0.3990	0.1943	0.5093
mean relative error [%]	0.0016	0.0012	0.0040

3.5.4 Summary

The new RBF approximation with a polynomial reproduction (Section 3.3.4) was proposed. This approach eliminates inconsistency which occurs in the original RBF approximation with a polynomial reproduction (Section 3.3.3). This inconsistency is caused by adding additional conditions to the polynomial part. The experiments made prove that the proposed approach gives significantly better results than the original method in terms of accuracy.

Further, the comparisons of different methods of RBF approximation with respect to various criteria were presented. The RBF approximation which was introduced in Section 3.3.1 gives the best results due to the smallest error. The proposed RBF approximation with linear reproduction can be influenced by the placement of the given dataset in space. Therefore, it is appropriate that the translation of the estimated center of gravity to the origin of the coordinate system is made as the first step. The worst results according to the error were obtained using the RBF approximation using Lagrange multipliers. Moreover, this method of approximation has unpredictable behavior, the matrix for RBF approximation using Lagrange multipliers is mostly ill-conditioned and its size is high, i.e. it is of the $(M + N) \times (M + N)$ size.

Moreover, the RBF approximation introduced in Section 3.3.1 was used for the approximation of synthetic and real datasets. The RBF approximation for large data which was introduced in Section 3.4 was used for large datasets. The main idea of this modification is that the symmetric matrix is partitioned and solved block-wise enabling the computation on systems with limited main memory. The experiments made prove that the proposed approach is able to determine the RBF approximation for large dataset. Moreover we can see, from the experimental results, that the use of a local RBFs is better than global RBFs if data are sufficiently sampled. Futher, it is obvious that the approximation using the global Gaussian RBFs has problems with the preservation of sharp edges. The experiments made also prove that the RBF methods have problems with the

accuracy of calculation on the boundary of an object, which is a well known property, and the magnitude of the RBF approximation error is influenced by the presence of a noise.

Finally, the comparison of results stemming from CS-RBFs approximation was performed. This comparison was done for synthetic and real datasets. The experiments showed that it is possible to divide the CS-RBFs from the catalog of RBFs (D.2.7 in [Fas07]) into three groups depending on the quality of the results. Moreover, we have found, from the results of the experiments, that CS-RBFs which return the best results in terms of the error for the synthetic datasets; may not be the best choice for real datasets.

Chapter 4

Reconstruction of Geometric Datasets

So far we dealt with explicit surface representation using RBF methods, i.e. we solved $f(\boldsymbol{x}_i) = h_i$. Now, we will deal with implicit representation of surface using RBF methods, i.e. we will solve $F(\boldsymbol{x}_i) = 0$, which is more difficult.

4.1 Carr's Method

The algorithm proposed by Carr et al in [CBC+01] deals with the reconstruction and the representation of large point clouds with RBF. An surface of point cloud is defined implicitly as the zero set of an RBF fitted to the given data.

The algorithm involves three steps. These steps are:

- Constructing a signed-distance function.
- Fitting an RBF to the resulting distance function.
- Iso-surfacing the fitted RBF using e.g. marching tetrahedra algorithm.

4.1.1 Fitting an Implicit Function to a Surface

Now, we define a problem to be solved more precisely. We have given n distinct points $\{(x_i, y_i, z_i)\}_{i=1}^{n}$ on a surface $S \in \mathbb{R}^3$, then find a surface S' that is a reasonable approximation to S. As the definition of the solved problem is clear; it implies that we wish to find a function f which implicitly defines a surface S', and the following equation is satisfied for all the given points:

$$f(x_i, y_i, z_i) = 0 \qquad i = 1, \ldots, n. \tag{4.1}$$

Moreover, off-surface points have to be appended to the input data, and non-zero values have to be given to avoid the trivial solution that f is zero everywhere. Thus, we get a more useful interpolation problem: Find f such that

$$\begin{aligned} f(x_i, y_i, z_i) &= 0 && i = 1, \ldots, n && \text{(on-surface points)}, \\ f(x_i, y_i, z_i) &= h_i \neq 0 && i = n+1, \ldots, N_c && \text{(off-surface points)}. \end{aligned} \tag{4.2}$$

Then, the problem of generating the off-surface points $\{(x_i, y_i, z_i)\}_{i=n+1}^{N_c}$ and the corresponding values h_i arises. An obvious choice for function f is a signed-distance function, i.e. the values h_i are chosen as the distance to the nearest on-surface point. The positive value of this distance indicates a point outside the object while the negative value indicates a point inside object. The off-surface points are generated by projecting along surface normals, see Figure 4.1.

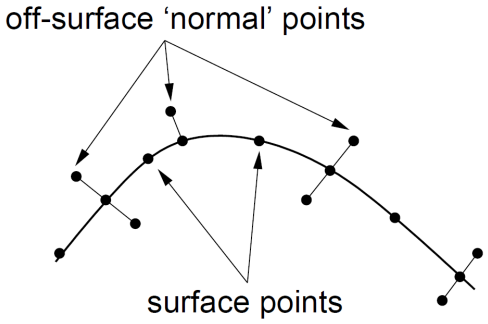


Figure 4.1: A signed-distance function is constructed from the surface data by specifying off-surface points along surface normals. These points may be specified on either or both sides of the surface, or not at all [CBC+01].

In Figure 4.2(a) the example of point cloud (green) for hand from a laser scan can be seen. Moreover, this point cloud is supplemented with off-surface points where points outside the object are represented by hot colors and inner points are colored by cold colors.

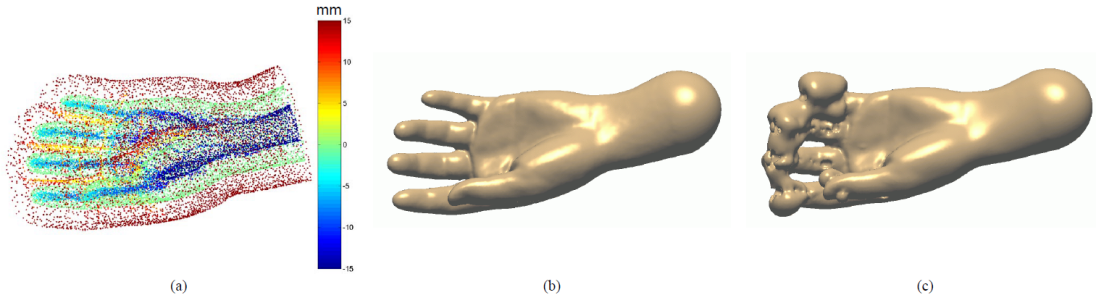


Figure 4.2: Reconstruction of a hand from a cloud of points with and without validation of normal lengths [CBC+01].

Estimating surface normals and determining the appropriate projection distance are two problems which have to be solved. If the partial mesh is known then normals are implied by the mesh connectivity at each vertex. However, in the case of unorganized dataset, it is required to estimate normals from a local neighborhood of points.

When the off-surface points are projecting along the normals, it has to be ensured that they do not intersect other parts of the surface. In the opposite case, i.e. when the inappropriate projecting distances are chosen, the off-surface points with the incorrect sign and magnitude of associated values d_i are generated and the resulting surface is distorted, see Figure 4.2(c). Therefore, the validation of off-surface distances and dynamic projection is performed to ensure that off-surface points produce a distance field consistent with the surface data, see Figure 4.2(a) and (b).

Now, when we have given a set of zero-valued surface points and non-zero off-surface points, our problem is possible to be solved as a scattered data interpolation problem which can be solved by the RBF interpolation with polynomial reproduction, see Section 3.2.2.

4.1.2 RBF Center Reduction

The method mentioned above uses all the input data points as centers of the RBF. However, it is possible to obtain the resulting surface with the desired accuracy using significantly fewer centers, see Figure 4.3. Therefore, a greedy algorithm can be used to iteratively fit an RBF to within the desired *fitting accuracy*. The steps of this greedy algorithm are introduced in Algorithm 1. Note that if a different accuracy δ_i is specified at each point then the condition may be replaced in step 3 by $|\varepsilon_i| < \delta_i$. An example of using this algorithm can be seen in Figure 4.4.

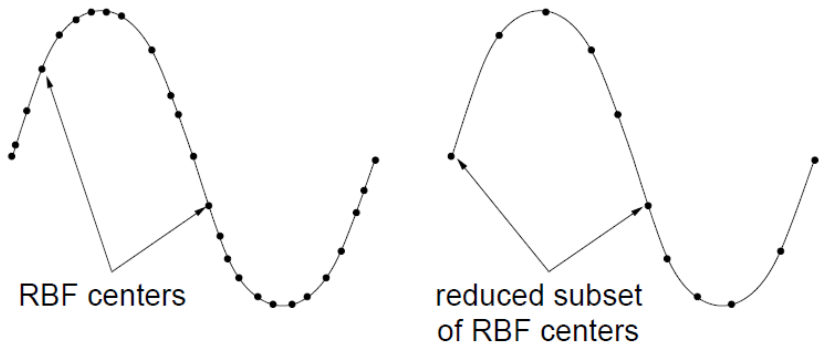


Figure 4.3: Illustration of center reduction [CBC+01].

Algorithm 1 Simple greedy algorithm for iteratively fit an RBF

1: Choose a subset from the interpolation nodes $\boldsymbol{x}_i$ and fit an RBF only to these.
2: Evaluate the residual, i.e. $\varepsilon_i = h_i - f(\boldsymbol{x}_i)$, at all nodes.
3: **if** $\max\{|\varepsilon_i|\} <$ *fitting accuracy* **then**
4: Stop
5: **else**
6: Append new centers where ε_i is large.
7: Re-fit RBF and goto 2.

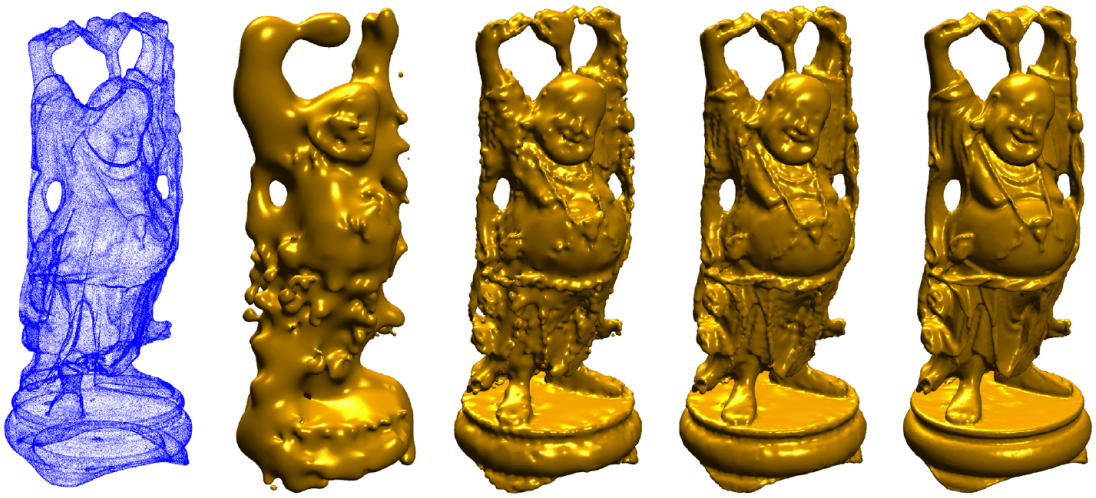


Figure 4.4: A greedy algorithm iteratively fits an RBF to a point cloud resulting in fewer centers in the final function. In this case the $544,000$ points is represented by $80,000$ centers to a relative accuracy of 5×10^{-4} in the final frame [CBC+01].

4.1.3 Reconstruction of Noisy Data

The real data is often noisy and irregular due to the limited range resolution and mis-registration between scans taken at different scanner positions. The linear system (3.12) is too strict for those data. Therefore, we use the idea of regularization (Section 3.2.3), and mentioned linear system of equations is modified to:

$$\begin{pmatrix} \boldsymbol{A} - 8N_c\pi\varrho\boldsymbol{I} & \boldsymbol{P} \\ \boldsymbol{P}^T & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{c} \\ \boldsymbol{a} \end{pmatrix} = \begin{pmatrix} \boldsymbol{h} \\ \boldsymbol{0} \end{pmatrix}, \tag{4.3}$$

where the regularization parameter $\varrho \geq 0$ can be through of as the stiffness of the RBF $f(\boldsymbol{x})$. Note that for parameter ϱ it is possible to the choose global value or to specify it for each data point or group of points.

4.2 Multilevel Compactly Supported RBF Interpolation

The algorithm proposed by Othake et al in [OBS05], and [OBS03] combines advantages provided by locally and globally supported RBFs and uses CS-RBFs in a hierarchical fashion.

Let us consider that we have given a set of N points $\mathcal{P} = \{\boldsymbol{p}_i\}_{i=1}^{N}$ scattered along a surface. Moreover, we assume that each point $\boldsymbol{p}_i$ in this set is associated with its inner unit normal $\boldsymbol{n}_i$ defining an orientation. We want to calculate a 3D scalar field $f(\boldsymbol{x})$ such that its zero level-set $f(\boldsymbol{x}) = 0$ interpolates $\mathcal{P}$.

At first, we build a multiscale hierarchy of point sets $\{\mathcal{P}^1, \mathcal{P}^2, \ldots, \mathcal{P}^L = \mathcal{P}\}$. To construct this hierarchy, we fit $\mathcal{P}$ into a parallelepiped and then we start an octree-based subdivision of this parallelepiped. Point set $\mathcal{P}$ is clustered with respect to the cells of created octree. After that each cell is represented by centroid of points contained in this cell. A unit normal; associated with the centroid, is obtained by averaging the normals associated with the points of $\mathcal{P}$ inside the cell and its normalizing. Set $\mathcal{P}^1$ corresponds to the subdivision of the initial parallelepiped into eight equal octants.

After constructing the multiscale hierarchy of point sets, it is possible to proceed to multilevel interpolation via offsetting in the coarse-to-fine way, i.e. the interpolation function of a point set $\mathcal{P}^m$ is used for interpolation of a point set $\mathcal{P}^{m+1}$. The main steps of this approach are demonstrated in Figure 4.5.

First, we define an initial base function:

$$f^0(\boldsymbol{x}) = -1 \tag{4.4}$$

and then recursively define the set of interpolating functions:

$$f^k(\boldsymbol{x}) = f^{k-1}(\boldsymbol{x}) + o^k(\boldsymbol{x}) \qquad k = 1, \ldots, L, \tag{4.5}$$

where $f^k(\boldsymbol{x}) = 0$ interpolates the set $\mathcal{P}^k$. The offsetting function $o^k(\boldsymbol{x})$ is determined as

$$o^k(\boldsymbol{x}) = \sum_{\boldsymbol{p}_i^k \in \mathcal{P}^k} \left[g_i^k(\boldsymbol{x}) + c_i^k \right] \phi_{\sigma^k}(\|\boldsymbol{x} - \boldsymbol{p}_i^k\|), \tag{4.6}$$

where $\phi_{\sigma^k}(r) = \phi(r/\sigma^k)$, $\phi(r) = (1-r)_+^4(4r+1)$ is Wendland's CS-RBFs, σ^k is its support size and $g_i^k(\boldsymbol{x})$ and c_i^k are unknown functions and coefficients to be determined.

The function $g_i^k(\boldsymbol{x})$ is a local shape function which can be estimated on the basis of number of points in the appropriate cell and the distribution of normals of those points [OBA+03]. In this section, the function $g_i^k(\boldsymbol{x})$ is defined such that $g_i^k(\boldsymbol{x}) = 0$ is the local quadratic approximations of $\mathcal{P}^k$ in a small neighborhood of

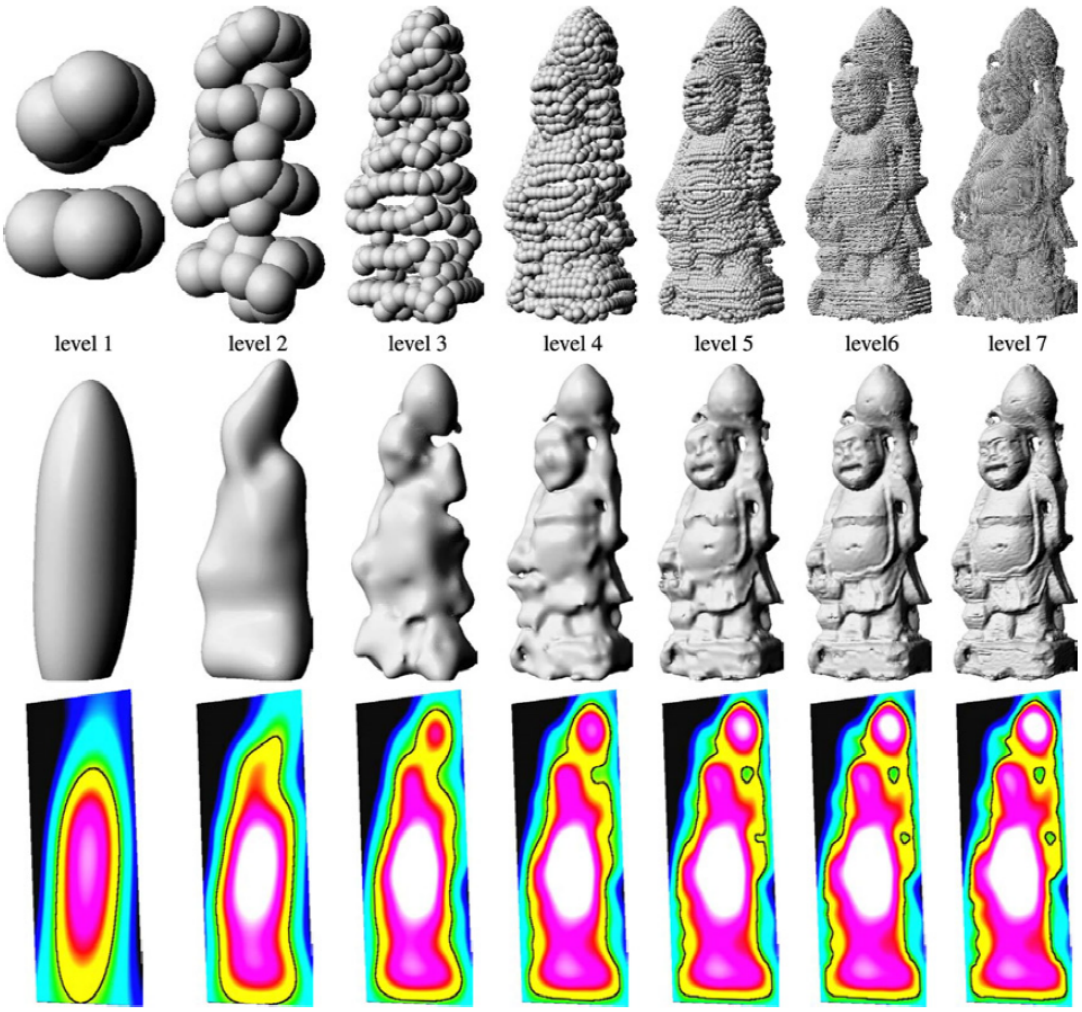


Figure 4.5: Multilevel interpolation of a monk model overview. (Top row) Multiscale hierarchy of points where the radii of the spheres at each level k are proportional to σ^k. (Middle row) Interpolating implicit surfaces polygonized at each level of the hierarchy. (Bottom row) cross-sections of the interpolating; the bold black lines correspond to the zero level sets of the functions. [OBS05].

$\boldsymbol{p}_i^k \in \mathcal{P}^k$. The zero level-set of introduced local approximation $g_i^k(\boldsymbol{x})$ corresponds to the graph of $w = h(u, v)$ where (u, v, w) is a local orthogonal coordinate system with the origin at point $\boldsymbol{p}_i^k$ such that the plane (u, v) is orthogonal to normal $\boldsymbol{n}_i^k$, and the direction of this normal coincides with the positive direction of w and $h(u, v)$ is a quadric

$$h(u, v) \equiv Au^2 + 2Buv + Cv^2, \tag{4.7}$$

where the coefficients A, B, and C are determined via the least square fitting applied to $\mathcal{P}^k$. The geometric idea behind this interpolation approach is illustrated in Figure 4.6.

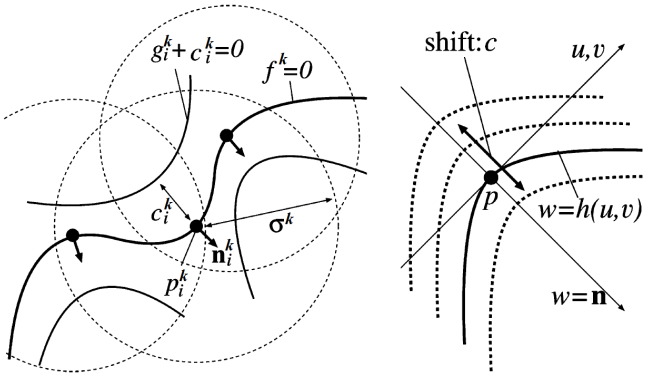


Figure 4.6: Geometric idea behind our approach for scattered point data interpolation [OBS05].

The shifting coefficients c_i^k are determined by solving the following linear system of equations:

$$f^{k-1}(\boldsymbol{p}_i^k) + o^k(\boldsymbol{p}_i^k) = 0. \tag{4.8}$$

The support size σ^k is estimate from the density of the set $\mathcal{P}$. This support size is defined by:

$$\sigma^{k+1} = \frac{\sigma^k}{2}, \qquad \sigma^1 = \alpha D, \tag{4.9}$$

where D corresponds to the length of diagonal of the bounding box of the set $\mathcal{P}$ and parameter α is selected such that a ball with radius σ^1 centered somewhere in the octant always overlays the octant of the bounding box. Parameter $\alpha = 0.75$ is typically used.

At last, the number of subdivision levels L has to be determined. The support size σ^0 is needed for these purposes; the same support size which was used for single-level interpolation. This support size corresponds to three fourths of the average diagonal of the leaf cells of octree. The introduced octree is constructed by using subdivision of bounding box of set $\mathcal{P}$, and the leaf cell contains no more than eight points of set $\mathcal{P}$. The following relation produced good results for determination number of subdivision levels L:

$$L = \left\lceil -\log_2 \frac{\sigma^0}{2\sigma^1} \right\rceil \tag{4.10}$$

4.3 3D Scattered Data Approximation with Adaptive Partition of Unity and CS-RBFs

The method for the approximation of scattered data combines an adaptive partition of unity approximation with least-square adaptive fitting; as proposed by Othake

et al in [OBS04] and [OBS06].

Let us consider that we have given a set of N points $\mathcal{P} = \{\boldsymbol{p}_i\}_{i=1}^N$ scattered along a surface. Moreover, we assume that each point $\boldsymbol{p}_i$ in this set is associated with its inner unit normal $\mathcal{N} = \{\boldsymbol{n}_i\}_{i=1}^N$ defining an orientation of surface and is attributed with a real number $\omega_i \in [0, 1]$ indicating the confidence of $\boldsymbol{p}_i$. We want to construct a function $y = f(\boldsymbol{x})$ such that its zero level-set $f(\boldsymbol{x}) = 0$ approximates $\mathcal{P}$. Further, we have given a set of approximation centers $\mathcal{C} = \{\boldsymbol{\xi}_i\}_{i=1}^M, M \ll N$. The main goal is the constructed function $f(\boldsymbol{x})$ approximating $\mathcal{P}$ in the following form:

$$\sum_{\boldsymbol{\xi}_i \in \mathcal{C}} \Big[g_i(\boldsymbol{x}) + c_i \Big] \phi_{\sigma_i}(\|\boldsymbol{x} - \boldsymbol{\xi}_i\|), \tag{4.11}$$

where $\phi_\sigma(r) = \phi(r/\sigma)$, $\phi(r) = (1-r)_+^4(4r+1)$ is Wendland's CS-RBFs, σ_i is support size and $g_i(\boldsymbol{x})$ and c_i are unknown functions and coefficients to be determined. For each approximation center $\boldsymbol{\xi}_i$ the function $g_i(\boldsymbol{x})$ is constructed as a local quadratic approximation of set $\mathcal{P}$ in $\{\|\boldsymbol{x} - \boldsymbol{\xi}_i\| < \sigma_i\}$, the region of influence of $\boldsymbol{\xi}_i$. Now, the coefficients $\{c_i\}_{i=1}^M$ from interpolation conditions

$$f(\boldsymbol{\xi}_i) = 0, \qquad i = 1, \ldots, M \tag{4.12}$$

are determined.

When the partition of unity (PU) approximation:

$$\sum_{\boldsymbol{\xi}_i \in \mathcal{C}} g_i(\boldsymbol{x}) \Phi_{\sigma_i}(\|\boldsymbol{x} - \boldsymbol{\xi}_i\|), \tag{4.13}$$

where the PU functions are given by

$$\Phi_{\sigma_i}(\|\boldsymbol{x} - \boldsymbol{\xi}_i\|) = \frac{\phi_{\sigma_i}(\|\boldsymbol{x} - \boldsymbol{\xi}_i\|)}{\sum_j \phi_{\sigma_j}(\|\boldsymbol{x} - \boldsymbol{\xi}_j\|)}, \tag{4.14}$$

which belong to the class of the normalized RBFs, is considered, the approximation of set $\mathcal{P}$ attributed with normals $\mathcal{N}$ and confidences $\{\omega_i\}$ is possible to be performed using a PU approximation and normalized RBFs $\Phi_{\sigma_i}(\|\boldsymbol{x} - \boldsymbol{\xi}_i\|)$:

$$\underbrace{\sum_{\boldsymbol{\xi}_i \in \mathcal{C}} g_i(\boldsymbol{x}) \Phi_{\sigma_i}(\|\boldsymbol{x} - \boldsymbol{\xi}_i\|)}_{\substack{\text{adaptive PU} \\ \text{(base approximation)}}} + \underbrace{\sum_{\boldsymbol{\xi}_i \in \mathcal{C}} c_i \Phi_{\sigma_i}(\|\boldsymbol{x} - \boldsymbol{\xi}_i\|)}_{\substack{\text{normalized RBF} \\ \text{(local details)}}} = 0. \tag{4.15}$$

The explanation, how to choose approximation centers $\{\boldsymbol{\xi}_i\}$ and determine their support sizes σ_i, and the construct local approximations $g_i(\boldsymbol{x})$ are described in Section 2 in [OBS06]. Figure 4.7 presents several steps in the construction the approximation centers $\{\boldsymbol{\xi}_i\}$ and their influence radii $\{\sigma_i\}$ for the Stanford bunny model. One of the possibilities, how to determine unknown RBF weights $\{c_i\}$, is then described in Section 3 in [OBS06].

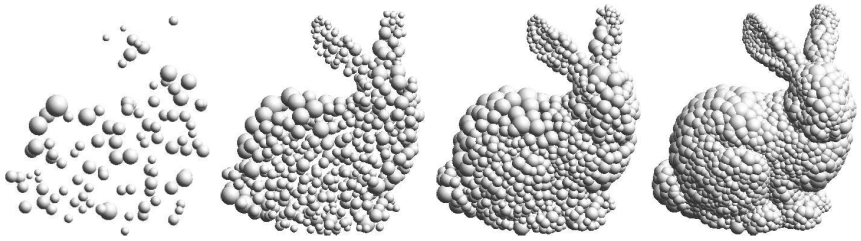


Figure 4.7: Four intermediate stages of the approximation center selection procedure. The number of approximation centers increases from left to right and is equal to 100, 500, 1000, and 2000, respectively [OBS06].

4.4 Morse's Method

Algebraic methods for creating implicit surfaces from scattered surface data are described by Morse et al in [MYC+01]. The authors use the RBF interpolation; see Section 3.2.1, or the RBF interpolation with polynomial reproduction; see Section 3.2.2, in combination with CS-RBFs; see Table 3.2, in this article.

The introduced approach uses the advantages of CS-RBFs for reduction of computational and memory costs. The CS-RBFs have finite radius of support. Therefore, the input interpolated set of points is sorting to $k-$d tree which speed up to find all points within the radius of support. Moreover, the resulting matrix is extremely sparse, see Figure 4.8, thus, the sparse-matrix representation is possible to be used. The authors use the Hartwell-Boeing format which is very similar to CSR format, see Section 2.1.4. It leads to reduction of memory costs.

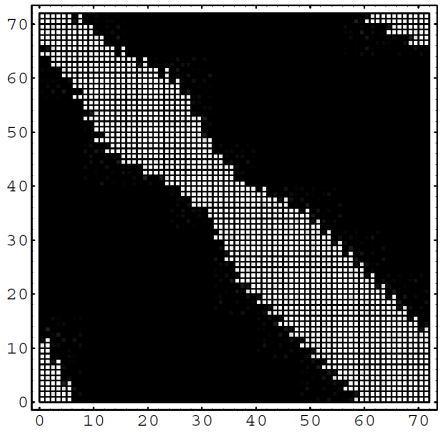


Figure 4.8: Structure of the matrix produced by CS-RBFs. Zero-element is colored by black, non-zero element is white [MYC+01].

4.5 Tobor's Method

Tobor et al in [TRS04] combines the RBF methods and the partition of unity (PU) method. The RBFs are used to solve a set of small local problems, and the PU method combines these local solutions together in this approach.

Let us consider that we have given a set of N pairwise distinct points $\mathcal{P} = \{\boldsymbol{p}_i\}_{i=1}^N \in \Omega \subset \mathbb{R}^d$ where d is dimension of space, and the set of values $\{h_i\}_{i=1}^N$. We want to calculate a function $f : \mathbb{R}^d \to \mathbb{R}$ with $f(\boldsymbol{p}_i) = h_i$ where $i = 1, \ldots, N$.

The reconstruction algorithm described above; contains two steps:

1. The space partitioning step which determines a set of "slightly" overlapping domains $\{\Omega_i\}$ with associated weighting functions $\{W_i\}$ which are needed for PU method.
2. The reconstruction step which computes the set of local functions $\{f_i\}$.

Due to the optimal computational complexity quasi-uniform distribution of the points in the domains $\{\Omega_i\}$ has to be obtained. Therefore, the adaptive subdivision of the domain Ω_i is described in Algorithm 2. In this algorithm each subdomain contains between T_{min} and T_{max} points.

Algorithm 2 Partition ($\mathcal{P}$, Ω_i)

Input: points $\mathcal{P}$, domain Ω_i
Output: set of domains $\{\Omega_j\}$

```
Compute number of points n of the set P
if n > T_max then
    Subdivide Ω_i into overlapping Ω_i^1, ..., Ω_i^m
    Partition (P, Ω_i^1)
    ...
    Partition (P, Ω_i^m)
else if n < T_min then
    while n ∉ [T_min, T_max] do
        if n < T_min then
            Expand Ω_i
        else if n > T_max then
            Reduce Ω_i
    Add Ω_i to {Ω_j}
else
    Add Ω_i to {Ω_j}
```

The example of a hierarchy of domains created by Algorithm 2 is shown in Figure 4.9. The final set of domains $\{\Omega_i\}$ contains all leaves of the tree. Moreover, for each subdomain Ω_i a set of points $\mathcal{P}_i = \{\boldsymbol{p} \in \mathcal{P} | \boldsymbol{p} \in \Omega_i\}$ is constructed.

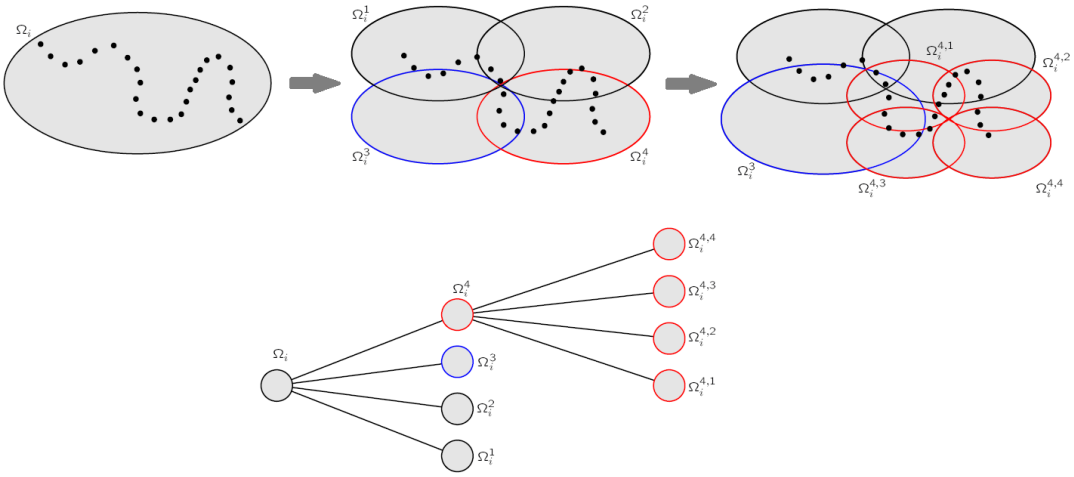


Figure 4.9: Space subdivision (Ω_i and Ω_i^4) and domain expanding (Ω_i^3) in order to adaptively balance the number of points per domain [TRS04].

The set of partition non-negative functions $\{w_i\}$ with limited support and with $\sum w_i = 1$ in the entire domain Ω have to be determined for the final global solution because the global solution is defined as a combination of the local functions weighted by the partition functions. These partition functions are obtained by normalization procedure of any other set of smooth functions $\{W_i\}$. It should be noted that to guarantee the continuity of the global interpolation function, the function W_i has to be continuous at the boundary of the region Ω_i. The examples of such functions are presented in Section 3.3 in [TRS04].

The second step of reconstruction algorithm is simple. An interpolation function f_i is computed for each domain in the set of domains $\{\Omega_i\}$ using the RBF interpolation with a polynomial reproduction (Section 3.2.2) for the set of points $\mathcal{P}_i$.

It should be noted that parameter T_{max} controls time and stability and parameter T_{min} provides a reconstruction without visible artifacts.

4.6 RBF Interpolation on GPU

Cuomo et al in [CGGS13] describes a parallel implicit method based on RBF for the surface reconstruction. This method uses the idea of Carr's method, see Section 4.1, specifically, the set of off-surface points is generated in the same manner. This is shown in Figure 4.10 where black points are original input dataset $\mathcal{X} = \{\boldsymbol{p_i}\}_{i=1}^{N}$, blue points are set of "outside" off-surface points $\mathcal{X}_\delta^+ = \{\boldsymbol{p_i}\}_{i=N+1}^{2N}$ and green points are set of "inside" off-surface points $\mathcal{X}_\delta^- = \{\boldsymbol{p_i}\}_{i=2N+1}^{3N}$.

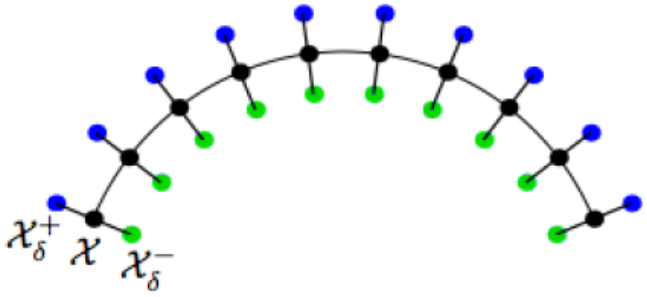


Figure 4.10: Extended interpolation dataset. There is shown set $\mathcal{X}$ (black points), set $\mathcal{X}_\delta^+$ (blue points) and set $\mathcal{X}_\delta^-$ (green points) [CGGS13].

The surface reconstruction algorithm is possible to be briefly described by Algorithm 3. The second step is most computational expensive because it leads to the solution of a linear system of $3N$ equations where N is the size of initial dataset. It means that the problem with memory can become with the growing size of the dataset.

Algorithm 3 Surface reconstruction

Input: Point could $\mathcal{X}$, surface normals $\{\boldsymbol{n}_i\}$, points of evaluation grid $\boldsymbol{x} = \{x_i\}_{i=1}^M$

1: Compute extended dataset: $\mathcal{X}_{ext} = \mathcal{X} \cup \mathcal{X}_\delta^+ \cup \mathcal{X}_\delta^-$ by using $\{\boldsymbol{n}_i\}$
2: Find the interpolant $f(\boldsymbol{x})$ (3.5) on $\mathcal{X}_{ext}$
3: Evaluate $f(\boldsymbol{x})$ (3.5) on $\boldsymbol{x}$
4: Render the surface

Therefore, the parallelization, on distributed memory architectures, becomes necessary as the size of dataset grows. The authors assume the idea of Domain Decomposition Method (DDM), i.e. the domain Ω containing the input dataset is divided into overlapping subdomains Ω_i, and then they try to solve the original problem as the series of subproblems. Moreover, the corresponding empty intersection portions of subdomains Ω_i are denoted $\tilde{\Omega}_i$. The example of the mentioned subdivision is shown in Figure 4.11. The solution of the linear system (3.7) on the whole domain is possible to be obtained by solving the linear sub-system $\boldsymbol{A}_i \boldsymbol{c}_{\Omega_i} = \boldsymbol{h}_{\Omega_i}$ in the individual overlapping subdomains where $\boldsymbol{A}_i$, $\boldsymbol{c}_{\Omega_i}$ and $\boldsymbol{h}_{\Omega_i}$ are the elements corresponding to domain Ω. The additive Schwarz method or restricted additive Schwarz method can be used for determination of this solution by using parallel calculations.

The complexity of calculation of mentioned linear system depends on the choice of RBFs. If basis functions with insignificant global effect are used, e.g. Gaussian function when the matrix $\boldsymbol{A}$ has the following elements:

$$A_{ij} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\|\boldsymbol{p}_i - \boldsymbol{p}_j\|^2}{2\sigma^2}}, \tag{4.16}$$

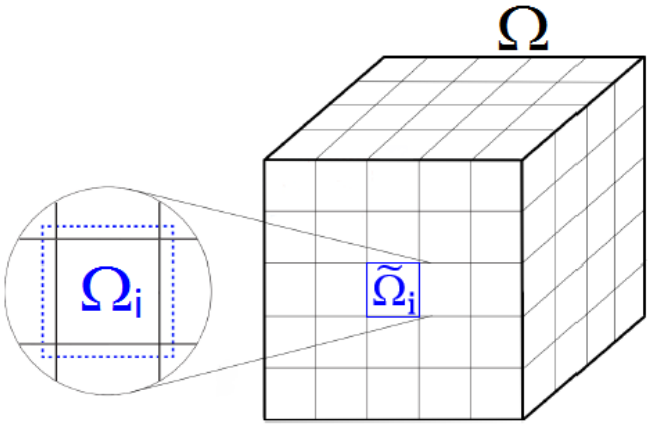


Figure 4.11: Illustration of the Domain Decomposition Method [CGGS13].

then the elements of matrix $\boldsymbol{A}$ corresponding to the interaction of distant points can be neglected.

The authors state that their implementation has been realized using the Portable Extensible Toolkit for Scientific Computation which has been added the GPU support. Moreover, the sparse matrix formats, see Section 2.1, are supported by this toolkit. The pseudo-code for construction of interpolation matrix is presented in Algorithm 4.

Algorithm 4 Interpolation matrix construction

1: **for each** subdomain Ω_i **do**
2: **for each** point $\boldsymbol{p}_i$ in the subdomain Ω_i **do**
3: **for each** point $\boldsymbol{p}_j$ in the truncation area of Ω_i **do**
4: Set $A_{ij} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\|\boldsymbol{p}_i - \boldsymbol{p}_j\|^2}{2\sigma^2}}$

4.7 TVL_1 Shape Approximation

Algorithm for 3D shape reconstruction which combines RBF approximation and non-smooth L_1 Total Variation regularization (TVL_1) for implicit 3D shape modelling is proposed by Funk et al in [FDB15]. This algorithm is based on original RBF approximation, see Section 3.3.1.

Let us consider that we have given a set of N points $\mathcal{P} = \{\boldsymbol{p}_i\}_{i=1}^N$ and we want to find a function $f(\boldsymbol{x})$ such that returns zero on every $i-$th sample $\boldsymbol{p}_i$. Unfortunately, the points lying on zero level of surface do not provide satisfactory information for the determination of search function. Therefore, the surface normals $\boldsymbol{n}_i$ at

every sample are used as constraints for gradient $\nabla f(\boldsymbol{x})$ with respect to $\boldsymbol{x}$. Thus, the goal is to find the function $f(\boldsymbol{x})$ such that satisfies:

$$\begin{aligned} f(\boldsymbol{p}_i) &= 0 \qquad & i = 1, \ldots, N \\ \nabla f(\boldsymbol{p}_i) &= \boldsymbol{n}_i \qquad & i = 1, \ldots, N \end{aligned} \tag{4.17}$$

Using all the information and knowledge that RBF approximation is minimization problem, the convex cost function is defined:

$$\min_{\boldsymbol{c}} \sum_{i=1}^{N} \left(\|f(\boldsymbol{p}_i)\|_2^2 + \|\boldsymbol{n}_i - \nabla f(\boldsymbol{p}_i)\|_2^2 \right), \tag{4.18}$$

where $\boldsymbol{c}$ is vector of unknown weights. Note that only the gradient of basis function $\phi(r) = \phi(\|\boldsymbol{x} - \boldsymbol{p}_j\|_2) = \varphi(\boldsymbol{x}, \boldsymbol{p}_j)$, which is precomputed analytically, is needed for obtaining the gradient ∇f. The equation (4.18) can be written in the following matrix form:

$$\min_{\boldsymbol{c}} \left(\|\boldsymbol{A}\boldsymbol{c}\|_2^2 + \|\boldsymbol{n} - \boldsymbol{A}_\nabla \boldsymbol{c}\|_2^2 \right), \tag{4.19}$$

where $A_{ij} = \varphi(\boldsymbol{p}_i, \boldsymbol{p}_j) \in \mathbb{R}$ is value of the RBF, $A_{\nabla ij} = \nabla\varphi(\boldsymbol{p}_i, \boldsymbol{p}_j) \in \mathbb{R}^3$ represents the derivatives of φ with respect to $\boldsymbol{x}_i$ and vector $\boldsymbol{n} = \left[\boldsymbol{n}_1^T, \ldots, \boldsymbol{n}_N^T\right]^T$ is vector of normals. Thus, the matrices are of sizes $\boldsymbol{A} \in \mathbb{R}^{N \times M}$ and $\boldsymbol{A}_\nabla \in \mathbb{R}^{3N \times M}$. It is obvious that RBFs with local support return zeros for distant points $\boldsymbol{p}_i$, $\boldsymbol{p}_j$, i.e. sparse matrices $\boldsymbol{A}$ and $\boldsymbol{A}_\nabla$ are obtained. The example of such sparse matrices is shown in Figure 4.12 where black dots visualize the non-zero entries of matrix.

Further, the extension of the cost function with total variation (TV) regularization term is performed which supports the piecewise smoothness. It leads to compute the second derivatives with respect to the radius r of the RBF $\phi(r)$. The term of

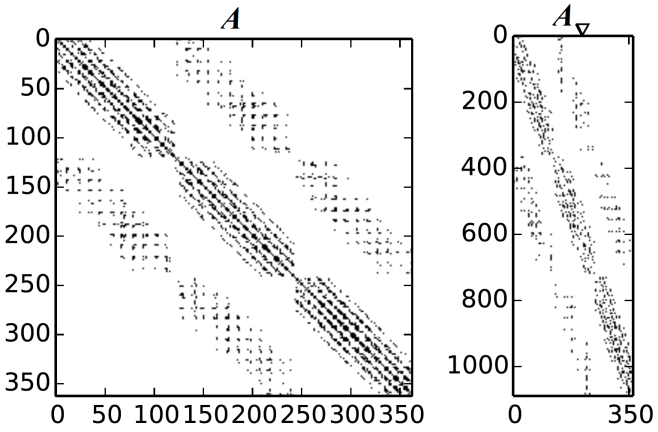


Figure 4.12: Example of sparse matrices $\boldsymbol{A}$ and $\boldsymbol{A}_\nabla$ when CS-RBF is applied [FDB15].

total variation regularization is expressed by:

$$TV(\boldsymbol{x}) = \sum_{i=1}^{M} \partial_{rr}^2 \phi(r) c_i. \tag{4.20}$$

Adding TV regularization term to (4.19) the new definition of cost function is obtained:

$$\min_{\boldsymbol{c}} \Big(\|\boldsymbol{A}\boldsymbol{c}\|_2^2 + \|\boldsymbol{n} - \boldsymbol{A}_{\nabla}\boldsymbol{c}\|_2^2 + \lambda \|\boldsymbol{D}\boldsymbol{c}\|_1 \Big), \tag{4.21}$$

where $D_{ij} = \partial_{rr}^2 \varphi(\boldsymbol{x_i}, \boldsymbol{x_j})$, $\|.\|_2^2$ is square of Euclidean norm, $\|.\|_1$ is L_1 norm and λ is the weight of the regularization term. Note that the weights c_i corresponding to the largest eigenvalue of $\boldsymbol{D}$ are reduced the most while coefficients lying in the kernel of $\boldsymbol{D}$ are not affected at all. Figure 4.13 shows an example when the input data have been perturbed by noise, see Figure 4.13 a, and the shape is reconstructed via simple least squares, see Figure 4.13 b, and TVL_1, see Figure 4.13 c.

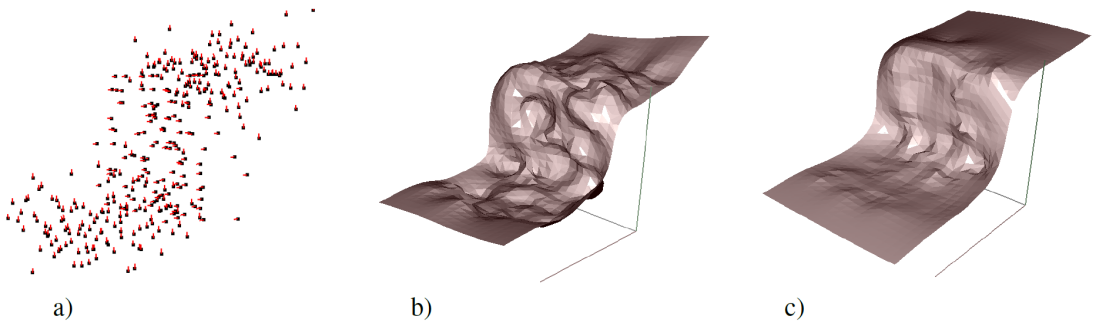


Figure 4.13: a) Noisy 3D samples of the step function. b) Direct least squares. c) TVL_1 regularized approximation [FDB15].

The numerical technique to efficiently solve the TVL_1 approximation is presented in Section 3.3 in [FDB15].

Chapter 5

Proposal of Future Work

In this work, the state of the art for problem of the interpolation and the approximation of large geometric datasets was presented. The explanation is aimed at the description of the meshless interpolation and approximation methods and data structures useful for representation large scattered datasets. Moreover, some improvements, as the new RBF approximation with a polynomial reproduction or the technique for calculation of large datasets, were shown, and it has been identified some problems.

We performed comparisons of different methods of the RBF approximation (Section 3.3) with respect to various criteria, different RBFs and different datasets. Conclusions from these comparisons are summarized in Section 3.5.4.

In this work we showed that the RBF approximation methods have generally problem with the accuracy of calculation on the boundary of an object, and the preservation of sharp edges is also problematic for some types of RBFs.

In the future work, we want to explore possibilities for improvement of RBF methods in respects of problems mentioned above. Further, the error of the RBF methods is influenced by the presence of a noise in the input data, thus, we want to perform research in terms of lower sensitivity to noise. Also improving the computational performance without loss of accuracy and improving the method for large datasets will be further explored. We would also like to implement and test some of the methods described in Chapter 4 to compare their performance and accuracy with our methods.

The another task which offers to deal is research in the area of the determination of the optimal shape parameter which influences radius of support of used RBF. This choice is currently performed on the basis of some experimental results. The question remains whether the relation for shape parameter is possible to be described empirically and ideally be derived adaptively depending on the characteristics (e.g. density) of data in the neighborhood of center of associated

RBF.

To summarize, the main goal of our future work is the development of RBF methods for the reconstruction of large geometric datasets with the following features:

- the better accuracy of calculation on the boundary of an object,
- the preservation of sharp edges,
- the lower sensitivity to noise,
- the better computational performance without loss of accuracy,
- the empirically and adaptively derived relation for shape parameter.

References

[ALP14] K. Anjyo, J. P. Lewis, and F. H. Pighin, "Scattered data interpolation for computer graphics," in *Special Interest Group on Computer Graphics and Interactive Techniques Conference, SIGGRAPH '14, Vancouver, Canada, August 10-14, 2014, Courses*, pp. 27:1–27:69, 2014.

[BG08] N. Bell and M. Garland, "Efficient sparse matrix-vector multiplication on cuda," tech. rep., Nvidia Technical Report NVR-2008-004, Nvidia Corporation, 2008.

[BG09] N. Bell and M. Garland, "Implementing sparse matrix-vector multiplication on throughput-oriented processors," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, p. 18, ACM, 2009.

[CBC+01] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, "Reconstruction and representation of 3d objects with radial basis functions," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2001, Los Angeles, California, USA, August 12-17, 2001*, pp. 67–76, 2001.

[CGGS13] S. Cuomo, A. Galletti, G. Giunta, and A. Starace, "Surface reconstruction from scattered point via RBF interpolation on GPU," in *Proceedings of the 2013 Federated Conference on Computer Science and Information Systems, Kraków, Poland, September 8-11, 2013.*, pp. 433–440, 2013.

[Cyc92] J. M. Cychosz, "Graphics gems iii," ch. Use of Residency Masks and Object Space Partitioning to Eliminate Ray-object Intersection Calculations, pp. 284–287, San Diego, CA, USA: Academic Press Professional, Inc., 1992.

[Dar00] E. Darve, "The fast multipole method: Numerical implementation," *Journal of Computational Physics*, vol. 160, no. 1, pp. 195–240, 2000.

[DT16] F. Dell'Accio and F. D. Tommaso, "Complete hermite-birkhoff interpolation on scattered data by combined shepard operators," *J. Computational Applied Mathematics*, vol. 300, pp. 192–206, 2016.

[DTS02] H. Q. Dinh, G. Turk, and G. G. Slabaugh, "Reconstructing surfaces by volumetric regularization using radial basis functions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 10, pp. 1358–1371, 2002.

[Duc77] J. Duchon, "Splines minimizing rotation-invariant semi-norms in sobolev spaces," in *Constructive theory of functions of several variables*, pp. 85–100, Springer, 1977.

[EPP00] T. Evgeniou, M. Pontil, and T. A. Poggio, "Regularization networks and support vector machines," *Adv. Comput. Math.*, vol. 13, no. 1, pp. 1–50, 2000.

[Fas07] G. E. Fasshauer, *Meshfree Approximation Methods with MATLAB*, vol. 6. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2007.

[FDB15] E. Funk, L. S. Dooley, and A. Börner, "Tvl_1 shape approximation from scattered 3d data," in *VISAPP 2015 - Proceedings of the 10th International Conference on Computer Vision Theory and Applications, Volume 3, Berlin, Germany, 11-14 March, 2015.*, pp. 294–304, 2015.

[Fra79] R. Franke, "A critical comparison of some methods for interpolation of scattered data," Tech. Rep. NPS53-79-003, NAVAL POSTGRADUATE SCHOOL MONTEREY CA, 1979.

[Fra82] R. Franke, "Scattered data interpolation: Tests of some methods," *Mathematics of computation*, vol. 38, no. 157, pp. 181–200, 1982.

[FZ07] G. E. Fasshauer and J. G. Zhang, "On choosing "optimal" shape parameters for RBF approximation," *Numerical Algorithms*, vol. 45, no. 1-4, pp. 345–368, 2007.

[Har71] R. L. Hardy, "Multiquadratic Equations of Topography and Other Irregular Surfaces," *Journal of Geophysical Research*, vol. 76, pp. 1905–1915, 1971.

[HSfY15] Y.-C. Hon, B. Sarler, and D. fang Yun, "Local radial basis function collocation method for solving thermo-driven fluid-flow problems with free surface," *Engineering Analysis with Boundary Elements*, vol. 57, pp. 2 – 8, 2015. {RBF} Collocation Methods.

[IdSPT14] D. Izquierdo, M. C. L. de Silanes, M. C. Parra, and J. J. Torrens, "CS-RBF interpolation of surfaces with vertical faults from scattered data," *Mathematics and Computers in Simulation*, vol. 102, pp. 11–23, 2014.

[Isk04] A. Iske, *Multiresolution Methods in Scattered Data Modelling*, vol. 37 of *Lecture Notes in Computational Science and Engineering*. Springer, 2004.

[JCW+15] G. R. Joldes, H. A. Chowdhury, A. Wittek, B. Doyle, and K. Miller, "Modified moving least squares with polynomial bases for scattered data approximation," *Applied Mathematics and Computation*, vol. 266, pp. 893–902, 2015.

[KHS03] N. Kojekine, I. Hagiwara, and V. V. Savchenko, "Software tools using CSRBFs for processing scattered data," *Computers & Graphics*, vol. 27, no. 2, pp. 311–319, 2003.

[Law13] O. S. Lawlor, "In-memory data compression for sparse matrices," in *Proceedings of the 3rd Workshop on Irregular Applications: Architectures and Algorithms*, p. 6, ACM, 2013.

[LCC13] M. Li, W. Chen, and C. Chen, "The localized RBFs collocation methods for solving high dimensional PDEs," *Engineering Analysis with Boundary Elements*, vol. 37, no. 10, pp. 1300 – 1304, 2013.

[LD08] A. Lagae and P. Dutré, "Compact, fast and robust grids for ray tracing," in *Computer Graphics Forum*, vol. 27, pp. 1235–1244, Wiley Online Library, 2008.

[LK11] S. Laine and T. Karras, "Efficient sparse voxel octrees," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 17, no. 8, pp. 1048–1059, 2011.

[LZ06] E. Langetepe and G. Zachmann, *Geometric data structures for computer graphics*. A K Peters, 2006.

[Mas03] T. Masuda, "Surface curvature estimation from the signed distance field," in *3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on*, pp. 361–368, IEEE, 2003.

[MS04] D. P. Mehta and S. Sahni, eds., *Handbook of Data Structures and Applications*. Chapman and Hall/CRC, 2004.

[MYC+01] B. S. Morse, T. S. Yoo, D. T. Chen, P. Rheingans, and K. R. Subramanian, "Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions," in *2001 International Conference on Shape Modeling and Applications (SMI 2001), 7-11 May 2001, Genoa, Italy*, pp. 89–98, 2001.

[OBA+03] Y. Ohtake, A. G. Belyaev, M. Alexa, G. Turk, and H. Seidel, "Multi-level partition of unity implicits," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 463–470, 2003.

[OBS03] Y. Ohtake, A. G. Belyaev, and H. Seidel, "A multi-scale approach to 3d scattered data interpolation with compactly supported basis functions," in *2003 International Conference on Shape Modeling and Applications (SMI 2003), May 2003, Seoul, Korea*, pp. 153–161, 2003.

[OBS04] Y. Ohtake, A. G. Belyaev, and H. Seidel, "3d scattered data approximation with adaptive compactly supported radial basis functions," in *2004 International Conference on Shape Modeling and Applications (SMI 2004), 7-9 June 2004, Genova, Italy*, pp. 31–39, 2004.

[OBS05] Y. Ohtake, A. G. Belyaev, and H. Seidel, "3d scattered data interpolation and approximation with multilevel compactly supported rbfs," *Graphical Models*, vol. 67, no. 3, pp. 150–165, 2005.

[OBS06] Y. Ohtake, A. G. Belyaev, and H. Seidel, "Sparse surface reconstruction with adaptive partition of unity and radial basis functions," *Graphical Models*, vol. 68, no. 1, pp. 15–24, 2006.

[PRF14] D. W. Pepper, C. Rasmussen, and D. Fyda, "A meshless method using global radial basis functions for creating 3-d wind fields from sparse meteorological data," *Computer Assisted Methods in Engineering and Science*, vol. 21, no. 3-4, pp. 233–243, 2014.

[PS11a] R. Pan and V. Skala, "Continuous global optimization in surface reconstruction from an oriented point cloud," *Computer-Aided Design*, vol. 43, no. 8, pp. 896–901, 2011.

[PS11b] R. Pan and V. Skala, "A two-level approach to implicit surface modeling with compactly supported radial basis functions," *Eng. Comput. (Lond.)*, vol. 27, no. 3, pp. 299–307, 2011.

[Rip99] S. Rippa, "An algorithm for selecting a good value for the parameter c in radial basis function interpolation," *Adv. Comput. Math.*, vol. 11, no. 2-3, pp. 193–210, 1999.

[Sch79] I. Schagen, "Interpolation in two dimensions - a new technique," *IMA Journal of Applied Mathematics*, vol. 23, no. 1, pp. 53–59, 1979.

[Sch11] M. Scheuerer, "An alternative procedure for selecting a good value for the parameter c in rbf-interpolation," *Adv. Comput. Math.*, vol. 34, no. 1, pp. 105–126, 2011.

[SHK09] V. Skala, J. Hrádek, and M. Kuchař, "Hash function for triangular mesh reconstruction," pp. 233–238, 2009.

[Šim09] I. Šimecek, "Sparse matrix computations using the quadtree storage format," in *Proceedings of 11th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2009)*, pp. 168–173, 2009.

[Ska12] V. Skala, "An efficient space partitioning method using binary maps," in *Proceedings of the 11th International Conference on Signal Processing (SIP '12)*, pp. 121–124, WSEAS, 2012.

[Ska13] V. Skala, "Fast Interpolation and Approximation of Scattered Multidimensional and Dynamic Data Using Radial Basis Functions," *WSEAS Transactions on Mathematics*, vol. 12, no. 5, pp. 501–511, 2013.

[Ska15] V. Skala, "Meshless interpolations for computer graphics, visualization and games," in *Eurographics 2015 - Tutorials, Zurich, Switzerland, May 4-8, 2015* (M. Zwicker and C. Soler, eds.), Eurographics Association, 2015.

[SMP+15] N. Sedaghati, T. Mu, L.-N. Pouchet, S. Parthasarathy, and P. Sadayappan, "Automatic selection of sparse matrix representation on gpus," in *Proceedings of the 29th ACM on International Conference on Supercomputing*, pp. 99–108, ACM, 2015.

[SPN13] V. Skala, R. Pan, and O. Nedved, "Simple 3d surface reconstruction using flatbed scanner and 3d print," in *SIGGRAPH Asia 2013, Hong Kong, China, November 19-22, 2013, Poster Proceedings*, p. 7, ACM, 2013.

[SPN14] V. Skala, R. Pan, and O. Nedved, "Making 3d replicas using a flatbed scanner and a 3d printer," in *Computational Science and Its Applications - ICCSA 2014 - 14th International Conference, Guimarães, Portugal, June 30 - July 3, 2014, Proceedings, Part VI*, vol. 8584 of *Lecture Notes in Computer Science*, pp. 76–86, Springer, 2014.

[TGB14] D. F. Trevisan, J. P. Gois, and H. C. Batagelo, "A low-cost-memory CUDA implementation of the conjugate gradient method applied

to globally supported radial basis functions implicits," *J. Comput. Science*, vol. 5, no. 5, pp. 701–708, 2014.

[TO02] G. Turk and J. F. O'Brien, "Modelling with implicit surfaces that interpolate," *ACM Trans. Graph.*, vol. 21, no. 4, pp. 855–873, 2002.

[TRS04] I. Tobor, P. Reuter, and C. Schlick, "Efficient reconstruction of large scattered geometric datasets using the partition of unity and radial basis functions," in *The 12-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2004, WSCG 2004, University of West Bohemia, Campus Bory, Plzen-Bory, Czech Republic, February 2-6, 2004*, pp. 467–474, 2004.

[US05] K. Uhlir and V. Skala, "Reconstruction of damaged images using radial basis functions," *Proceedings of EUSIPCO 2005*, p. 160, 2005.

[Wen95] H. Wendland, "Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree," *Adv. Comput. Math.*, vol. 4, no. 1, pp. 389–396, 1995.

[Wen06] H. Wendland, "Computational aspects of radial basis function approximation," *Studies in Computational Mathematics*, vol. 12, pp. 231–256, 2006.

[ZVS09] J. Zapletal, P. Vaněček, and V. Skala, "RBF-based image restoration utilising auxiliary points," in *Proceedings of the 2009 Computer Graphics International Conference*, pp. 39–43, ACM, 2009.

Publications

[MS16a] Z. Majdisova and V. Skala, "A New Radial Basis Function Approximation with Reproduction," in *Proceedings of the International Conferences on Interfaces and Human Computer Interaction 2016; Game and Entertainment Technologies 2016; and Computer Graphics, Visualization, Computer Vision and Image Processing 2016*, pp. 215–222, IADIS Press, July 2016.

[MS16b] Z. Majdisova and V. Skala, "A Radial Basis Function Approximation for Large Datasets," in *Proceedings of SIGRAD 2016*, pp. 9–14, Linköping University Electronic Press, May 2016.

[SM15] V. Skala and Z. Majdisova, "Fast Algorithm for Finding Maximum Distance with Space Subdivision in E2," in *Image and Graphics* (Y.-J. Zhang, ed.), vol. 9218 of *Lecture Notes in Computer Science*, pp. 261–274, Springer International Publishing, 2015.

[SMS16] V. Skala, Z. Majdisova, and M. Smolik, "Space Subdivision to Speed-up Convex Hull Construction in E3," *Advances in Engineering Software*, vol. 91, pp. 12–22, 2016.

Submitted Publications

[Maj16] Z. Majdisova, "A Comparative Study of Local RBFs for Big Data Approximation," 2016. [In preparation].

[MS] Z. Majdisova and V. Skala, "Radial Basis Function Approximations: Comparison and Applications," *Applied Mathematical Modelling*. [Submitted 05-03-2016].

Appendix A

Project Assignments, Other Activities

A.1 Conferences and Talks

- A New Radial Basis Function Approximation with Reproduction, 10th International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing (CGVCVIP), Funchal, Madeira, Portugal, July, 2016
- A Radial Basis Function Approximation for Large Datasets, SIGRAD 2016 (the Swedish Chapter of Eurographics) conference, Visby, Sweden, May 24, 2016
- Fast Algorithm for Finding Maximum Distance with Space Subdivision in E^2, 8th International Conference on Image and Graphics, Tianjin, China, August 15, 2015
- Data Structures (in Czech), University of West Bohemia, Plzeň, Czech Republic, May, 19, 2015
- Convex Hull and Space Subdivision in E^3 (in Czech), University of West Bohemia, Plzeň, Czech Republic, January, 30, 2015

A.2 Abroad

- CGVCVIP 2016, Funchal, Madeira, Portugal, July 2-4, 2016
- SIGRAD 2016, Visby, Sweden, May 23-24, 2016

- CERC 2015, Zagreb, Croatia, November 13-15, 2015
- ICIG 2015, Tianjin, China, August 13-16, 2015
- Eurographics 2015, Zürich, Switzerland, May 4-8, 2015

A.3 Participation on Scientific Projects

- Advanced Computing and Information Systems. Project code SGS-2016-013.
- Development of Algorithms for Computer Graphics and CAD/CAM Systems. Funded by MSMT CR, project code LH12181.
- Advanced Computing and Information Systems. Project code SGS-2013-029.

A.4 Teaching Activities

2014/2015:

- Programming Strategies (KIV/PRO) - tutor
- Introduction to Computer Graphics (KIV/UPG) - tutor

2015/2016:

- Programming Strategies (KIV/PRO) - tutor
- Introduction to Computer Graphics (KIV/UPG) - tutor

A.5 Other Activities

- Chairing session at WSCG 2016
- Preparation of competitive problems for programming contest PilsProg 2016
- Preparation of competitive problems for programming contest PilsProg 2015
- Chairing session at WSCG 2012