Meshfree Approximation Methods with MATLAB

INTERDISCIPLINARY MATHEMATICAL SCIENCES

Published

Vol. 1: Global Attractors of Nonautonomous Dissipative Dynamical Systems
David N. Cheban

Vol. 2: Stochastic Differential Equations: Theory and Applications
A Volume in Honor of Professor Boris L. Rozovskii
eds. Peter H. Baxendale & Sergey V. Lototsky

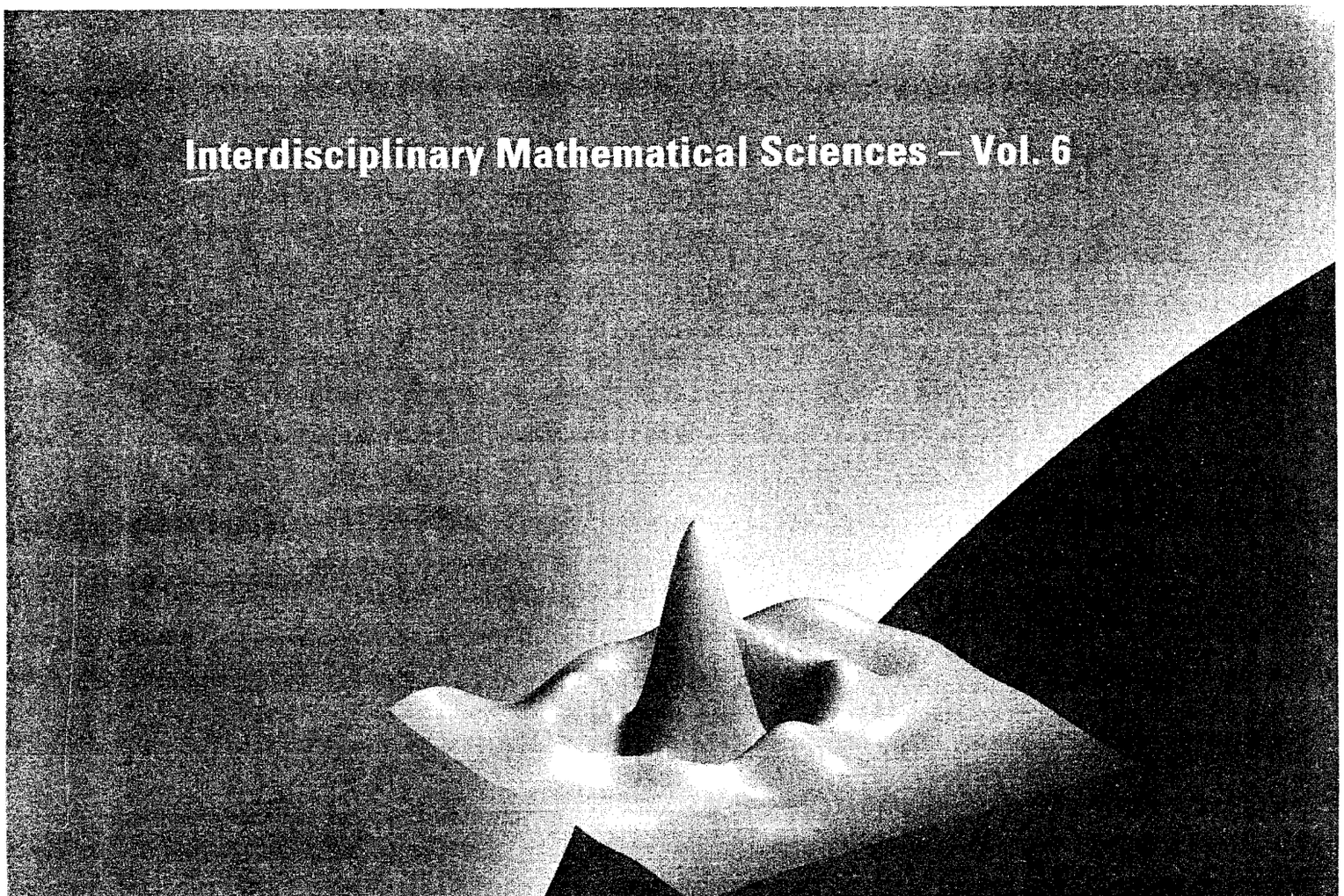
Vol. 3: Amplitude Equations for Stochastic Partial Differential Equations
Dirk Blömker

Vol. 4: Mathematical Theory of Adaptive Control
Vladimir G. Sragovich

Vol. 5: The Hilbert–Huang Transform and Its Applications
Norden E. Huang & Samuel S. P. Shen

Vol. 6: Meshfree Approximation Methods with MATLAB
Gregory E. Fasshauer

Interdisciplinary Mathematical Sciences – Vol. 6



Meshfree Approximation Methods with MATLAB

Gregory E. Fasshauer
Illinois Institute of Technology, USA

World Scientific

Published by

World Scientific Publishing Co. Pte. Ltd.
5 Toh Tuck Link, Singapore 596224
USA office: 27 Warren Street, Suite 401-402, Hackensack, NJ 07601
UK office: 57 Shelton Street, Covent Garden, London WC2H 9HE

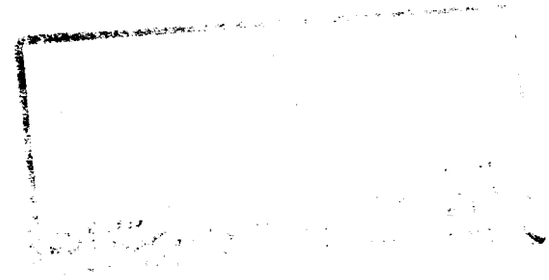
British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

MESHFREE APPROXIMATION METHODS WITH MATLAB
(With CD-ROM)
Interdisciplinary Mathematical Sciences — Vol. 6

ISBN-13 978-981-270-633-1
ISBN-10 981-270-633-X
ISBN-13 978-981-270-634-8 (pbk)
ISBN-10 981-270-634-8 (pbk)

Printed by Mainland Press Pte Ltd

This book is dedicated to
Inge, Conny, Marc and Patrick.



Preface

Traditional numerical methods, such as finite element, finite difference, or finite volume methods, were motivated mostly by early one- and two-dimensional simulations of engineering problems via partial differential equations (PDEs). The discretization involved in all of these methods requires some sort of underlying computational mesh, *e.g.*, a triangulation of the region of interest. Creation of these meshes (and possible re-meshing) becomes a rather difficult task in three dimensions, and virtually impossible for higher-dimensional problems. This is where *meshfree* methods enter the picture. Meshfree methods are often — but by no means have to be — radially symmetric in nature. This is achieved by composing some univariate basic function with a (Euclidean) norm, and therefore turning a problem involving many space dimensions into one that is virtually one-dimensional. Such *radial basis functions* are at the heart of this book. Some people have argued that there are three "big technologies" for the numerical solution of PDEs, namely finite difference, finite element, and spectral methods. While these technologies came into their own right in successive decades, namely finite difference methods in the 1950s, finite element methods in the 1960s, and spectral methods in the 1970s, meshfree methods started to appear in the mathematics literature in the 1980s, and they are now on their way to becoming "big technology" number four. In fact, we will demonstrate in later parts of this book how different types of meshfree methods can be viewed as generalizations of the traditional "big three".

Multivariate meshfree approximation methods are being studied by many researchers. They exist in many flavors and are known under many names, *e.g.*, diffuse element method, element-free Galerkin method, generalized finite element method, *hp*-clouds, meshless local Petrov-Galerkin method, moving least squares method, partition of unity finite element method, radial basis function method, reproducing kernel particle method, smooth particle hydrodynamics method.

In this book we are concerned mostly with the moving least squares (MLS) and radial basis function (RBF) methods. We will consider all different kinds of aspects of these meshfree approximation methods: How to construct them? Are these constructions mathematically justifiable? How accurate are they? Are there ways to implement them efficiently with standard mathematical software packages such

as MATLAB? How do they compare with traditional methods? How do the various flavors of meshfree methods differ from one another, and how are they similar to one another? Is there a general framework that captures all of these methods? What sort of applications are they especially well suited for?

While we do present much of the underlying theory for RBF and MLS approximation methods, the emphasis in this book is not on proofs. For readers who are interested in all the mathematical details and intricacies of the theory we recommend the two excellent recent monographs [Buhmann (2003); Wendland (2005a)]. Instead, our objective is to make the theory accessible to a wide audience that includes graduate students and practitioners in all sorts of science and engineering fields. We want to put the mathematical theory in the context of applications and provide MATLAB implementations which give the reader an easy entry into meshfree approximation methods. The skilled reader should then easily be able to modify the programs provided here for his/her specific purposes.

In a certain sense the present book was inspired by the beautiful little book [Trefethen (2000)]. While the present book is much more expansive (filling more than five hundred pages with forty-seven MATLAB[1] programs, one Maple[2] program, one hundred figures, more than fifty tables, and more than five hundred references), it is our aim to provide the reader with relatively simple MATLAB code that illustrates just about every aspect discussed in the book.

All MATLAB programs printed in the text (as well as a few modifications discussed) are also included on the enclosed CD. The folder `MATLAB` contains M-files and data files of type MAT that have been written and tested with MATLAB 7. For those readers who do not have access to MATLAB 7, the folder `MATLAB6` contains versions of these files that are compatible with the older MATLAB release. The main difference between the two versions is the use of `anonymous` functions in the MATLAB 7 code as compared to `inline` functions in the MATLAB 6 version. Two packages from the MATLAB Central File Exchange [MCFE] are used throughout the book: the function `haltonseq` written by Daniel Dougherty and used to generate sequences of Halton points; the *kd*-tree library (given as a set of MATLAB MEX-files) written by Guy Shechter and used to generate the *kd*-tree data structure underlying our sparse matrices based on compactly supported basis functions. Both of these packages are discussed in Appendix A and need to be downloaded separately. The folder `Maple` on the CD contains the one Maple file mentioned above.

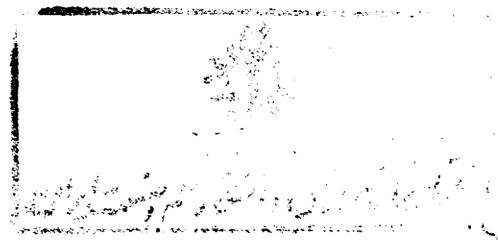
The manuscript for this book and some of its earlier incarnations have been used in graduate level courses and seminars at Northwestern University, Vanderbilt University, and the Illinois Institute of Technology. Special thanks are due to Jon

[1] MATLAB® is a trademark of The MathWorks, Inc. and is used with permission. The MathWorks does not warrant the accuracy of the text or exercises in this book. This book's use or discussion of MATLAB software or related products does not constitute endorsement or sponsorship by The MathWorks of a particular pedagogical approach or particular use of the MATLAB software.

[2] Maple™ is a registered trademark of Waterloo Maple Inc.

Cherrie, John Erickson, Paritosh Mokhasi, Larry Schumaker, and Jack Zhang for reading various portions of the manuscript and/or MATLAB code and providing helpful feedback. Finally, thanks are due to all the people at World Scientific Publishing Co. who helped make this project a success: Rajesh Babu, Ying Oi Chiew, Linda Kwan, Rok Ting Tan, and Yubing Zhai.

Greg Fasshauer
Chicago, IL, January 2007



Contents

Chapter 1

Introduction

Meshfree methods have gained much attention in recent years, not only in the mathematics but also in the engineering community. Thus, much of the work concerned with meshfree approximation methods is interdisciplinary — at the interface between mathematics and numerous application areas (see the partial list below). Moreover, computation with high-dimensional data is an important issue in many areas of science and engineering. Many traditional numerical methods can either not handle such problems at all, or are limited to very special (regular) situations. Meshfree methods are often better suited to cope with changes in the geometry of the domain of interest (*e.g.*, free surfaces and large deformations) than classical discretization techniques such as finite differences, finite elements or finite volumes. Another obvious advantage of meshfree discretizations is — of course — their independence from a mesh. Mesh generation is still the most time consuming part of any mesh-based numerical simulation. Since meshfree discretization techniques are based only on a set of independent points, these costs of mesh generation are eliminated. Meshfree approximation methods can be seen to provide a new generation of numerical tools. Other traditional numerical methods such as the finite element, finite difference or finite volume methods are usually limited to problems involving two or three parameters (space dimensions). However, in many applications the number of parameters can easily range in the hundreds or even thousands. Multivariate approximation methods present one way to address these issues.

Applications of meshfree methods can be found

- in many different areas of science and engineering via *scattered data modeling* (*e.g.*, fitting of potential energy surfaces in chemistry; coupling of engineering models with sets of incompatible parameters; mapping problems in geodesy, geophysics, meteorology);
- in many different areas of science and engineering via *solution of partial differential equations* (*e.g.*, solution of gas dynamics equations, Boltzmann and Fokker-Planck equations in six-dimensional phase space; problems involving moving discontinuities such as cracks and shocks, multi-scale resolution, large material distortions; elasticity studies in plate and shell bending

problems; applications in nanotechnology);
- in *non-uniform sampling* (*e.g.*, medical imaging, tomographic reconstruction);
- in *mathematical finance* (*e.g.*, option pricing);
- in *computer graphics* (*e.g.*, representation of surfaces from point information such as laser range scan data, image warping);
- in *learning theory*, *neural networks* and *data mining* (*e.g.*, kernel approximation, support vector machines);
- in *optimization.*

Since many of these applications either come down to a function approximation problem, or include function approximation as a fundamental component, we will begin our discussion with — and in fact base a large part of the contents of this book on — the multivariate scattered data interpolation problem.

1.1 Motivation: Scattered Data Interpolation in $\mathbb{R}^s$

We will now describe the general process of scattered data fitting, which is one of the fundamental problems in approximation theory and data modeling in general. Our desire to have a well-posed problem formulation will naturally lead to an introductory example based on the use of so-called *distance matrices.* In the next chapters we will generalize this approach by introducing the concept of a radial basis function.

1.1.1 *The Scattered Data Interpolation Problem*

In many scientific disciplines one faces the following problem: We are given a set of data (measurements, and locations at which these measurements were obtained), and we want to find a rule which allows us to deduce information about the process we are studying also at locations different from those at which we obtained our measurements. Thus, we are trying to find a function $\mathcal{P}_f$ which is a "good" fit to the given data. There are many ways to decide what we mean by "good", and the only criterion we will consider now is that we want the function $\mathcal{P}_f$ to exactly match the given measurements at the corresponding locations. This approach is called *interpolation*, and if the locations at which the measurements are taken do not lie on a uniform or regular grid, then the process is called *scattered data interpolation.*

To give a precise definition we assume that the measurement locations (or *data sites*) are labeled $\boldsymbol{x}_j$, $j = 1, \ldots, N$, and the corresponding measurements (or *data values*) are called y_j. We will use $\mathcal{X}$ to denote the set of data sites and assume that $\mathcal{X} \subset \Omega$ for some region Ω in $\mathbb{R}^s$. Throughout this book we will restrict our discussion to scalar-valued data, *i.e.*, $y_j \in \mathbb{R}$. However, much of the following can be generalized easily to problems with vector-valued data. In many of our later

discussions we will assume that the data are obtained by sampling some (unknown) function f at the data sites, *i.e.*, $y_j = f(\boldsymbol{x}_j)$, $j = 1, \ldots, N$. Our notation $\mathcal{P}_f$ for the interpolating function emphasizes the connection between the interpolant and the data function f. We are now ready for a precise formulation of the scattered data interpolation problem.

Problem 1.1 (Scattered Data Interpolation). *Given data* $(\boldsymbol{x}_j, y_j)$, $j = 1, \ldots, N$, *with* $\boldsymbol{x}_j \in \mathbb{R}^s$, $y_j \in \mathbb{R}$, *find a (continuous) function* $\mathcal{P}_f$ *such that* $\mathcal{P}_f(\boldsymbol{x}_j) = y_j$, $j = 1, \ldots, N$.

The fact that we allow $\boldsymbol{x}_j$ to lie in an arbitrary s-dimensional space $\mathbb{R}^s$ means that the formulation of Problem 1.1 allows us to cover many different types of applications. If $s = 1$ the data could, *e.g.*, be a series of measurements taken over a certain time period, thus the "data sites" $\boldsymbol{x}_j$ would correspond to certain time instances. For $s = 2$ we can think of the data being obtained over a planar region, and so $\boldsymbol{x}_j$ corresponds to the two coordinates in the plane. For instance, we might want to produce a map that shows the rainfall in the state we live in based on the data collected at weather stations located throughout the state. For $s = 3$ we might think of a similar situation in space. One possibility is that we could be interested in the temperature distribution inside some solid body. Higher-dimensional examples might not be that intuitive, but a multitude of them exist, *e.g.*, in finance, optimization, economics or statistics, but also in artificial intelligence or learning theory.

A convenient and common approach to solving the scattered data problem is to make the assumption that the function $\mathcal{P}_f$ is a linear combination of certain *basis functions* B_k, *i.e.*,

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{k=1}^{N} c_k B_k(\boldsymbol{x}), \qquad \boldsymbol{x} \in \mathbb{R}^s. \tag{1.1}$$

Solving the interpolation problem under this assumption leads to a system of linear equations of the form

$$A\mathbf{c} = \boldsymbol{y},$$

where the entries of the *interpolation matrix* A are given by $A_{jk} = B_k(\boldsymbol{x}_j)$, $j, k = 1, \ldots, N$, $\mathbf{c} = [c_1, \ldots, c_N]^T$, and $\boldsymbol{y} = [y_1, \ldots, y_N]^T$.

Problem 1.1 will be *well-posed*, *i.e.*, a solution to the problem will exist and be unique, if and only if the matrix A is non-singular.

In the univariate setting it is well known that one can interpolate to arbitrary data at N distinct data sites using a polynomial of degree $N-1$. For the multivariate setting, however, there is the following negative result (see [Mairhuber (1956); Curtis (1959)]).

Theorem 1.1 (Mairhuber-Curtis). *If* $\Omega \subset \mathbb{R}^s$, $s \geq 2$, *contains an interior point, then there exist no Haar spaces of continuous functions except for one-dimensional ones.*

In order to understand this theorem we need

Definition 1.1. Let the finite-dimensional linear function space $\mathcal{B} \subseteq C(\Omega)$ have a basis $\{B_1, \ldots, B_N\}$. Then $\mathcal{B}$ is a *Haar space* on Ω if

$$\det A \neq 0$$

for any set of distinct $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ in Ω. Here A is the matrix with entries $A_{jk} = B_k(\boldsymbol{x}_j)$.

Note that existence of a Haar space guarantees invertibility of the interpolation matrix A, *i.e.*, existence and uniqueness of an interpolant of the form (1.1) to data specified at $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ from the space $\mathcal{B}$. As mentioned above, univariate polynomials of degree $N-1$ form an N-dimensional Haar space for data given at $x_1, \ldots, x_N$.

The Mairhuber-Curtis theorem tells us that if we want to have a well-posed multivariate scattered data interpolation problem we can no longer fix in advance the set of basis functions we plan to use for interpolation of arbitrary scattered data. For example, it is not possible to perform unique interpolation with (multivariate) polynomials of degree N to data given at arbitrary locations in $\mathbb{R}^2$. Instead, the basis should depend on the data locations. We will give a simple example of such an interpolation scheme in the next subsection.

Proof. [of Theorem 1.1] Let $s \geq 2$ and assume that $\mathcal{B}$ is a Haar space with basis $\{B_1, \ldots, B_N\}$ with $N \geq 2$. We need to show that this leads to a contradiction.

We let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ be a set of distinct points in $\Omega \subset \mathbb{R}^s$ and A the matrix with entries $A_{jk} = B_k(\boldsymbol{x}_j)$, $j, k = 1, \ldots, N$. Then, by the definition of a Haar space, we have

$$\det A \neq 0. \tag{1.2}$$

Now, consider a closed path P in Ω connecting only $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. This is possible since — by assumption — Ω contains an interior point. We can exchange the positions of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ by moving them continuously along the path P (without interfering with any of the other $\boldsymbol{x}_j$). This means, however, that rows 1 and 2 of the determinant (1.2) have been exchanged, and so the determinant has changed sign.

Since the determinant is a continuous function of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ we must have had $\det = 0$ at some point along P. This contradicts (1.2). □

1.1.2 *Example: Interpolation with Distance Matrices*

In order to obtain data dependent approximation spaces as suggested by the Mairhuber-Curtis theorem we now consider a simple example. As a "testfunction" we employ the function

$$f_s(\boldsymbol{x}) = 4^s \prod_{d=1}^{s} x_d(1 - x_d), \qquad \boldsymbol{x} = (x_1, \ldots, \boldsymbol{x}_s) \in [0,1]^s.$$

This function is zero on the boundary of the unit cube in $\mathbb{R}^s$ and has a maximum value of one at the center of the cube. A simple MATLAB script defining f_s is given as Program C.1 in Appendix C.

We will use a set of uniformly scattered data sites in the unit cube at which we sample our testfunction f_s. This will be accomplished here (and in many other examples later on) by resorting to the so-called *Halton points*. These are uniformly distributed random points in $(0,1)^s$. A set of 289 Halton points in the unit square in $\mathbb{R}^2$ is shown in Figure 1.1. More details on Halton points are presented in Appendix A. In our computational experiments we generate Halton points using the program `haltonseq.m` written by Daniel Dougherty. This function can be downloaded from the MATLAB Central File Exchange (see [MCFE]).

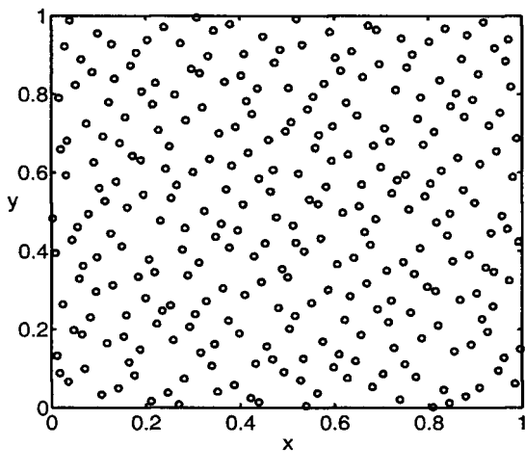


Fig. 1.1 289 Halton points in the unit square in $\mathbb{R}^2$.

As explained in the previous subsection we are interested in constructing a (continuous) function $\mathcal{P}_f$ that interpolates the samples obtained from f_s at the set of Halton points, *i.e.*, such that

$$\mathcal{P}_f(\boldsymbol{x}_j) = f_s(\boldsymbol{x}_j), \qquad \boldsymbol{x}_j \text{ a Halton point.}$$

As pointed out above, if $s = 1$, then this problem is often solved using univariate polynomials or splines. For a small number of data sites polynomials may work satisfactorily. However, if the number of points increases, *i.e.*, the polynomial degree grows, then it is well known that one should use splines (or piecewise polynomials) to avoid oscillations. The simplest solution is to use a continuous piecewise linear spline, *i.e.*, to "connect the dots". It is also well known that one possible basis for the space of piecewise linear splines interpolating data at a given set of points in $[0,1]$ consists of the shifts of the absolute value function to the data sites. In other words, we can construct the piecewise linear spline interpolant by assuming $\mathcal{P}_f$ is of the form

$$\mathcal{P}_f(x) = \sum_{k=1}^{N} c_k |x - x_k|, \qquad x \in [0,1],$$

and then determine the coefficients c_k by satisfying the interpolation conditions

$$\mathcal{P}_f(x_j) = f_1(x_j), \qquad j = 1, \ldots, N.$$

Clearly, the basis functions $B_k = |\cdot - x_k|$ are dependent on the data sites as suggested by the Mairhuber-Curtis theorem. The points x_k to which the basic function $B(x) = |x|$ is shifted are usually referred to as *centers*. While there may be circumstances that suggest choosing these centers different from the data sites one generally picks the centers to coincide with the data sites. This simplifies the analysis of the method, and is sufficient for many applications. Since the functions B_k are (radially) symmetric about their centers x_k this constitutes the first example of *radial basis functions*. We will formally introduce the notion of a radial function in the next chapter.

Of course, one can imagine many other ways to construct an N-dimensional data-dependent basis for the purpose of scattered data interpolation. However, the use of shifts of one single basic function makes the radial basis function approach particularly elegant.

Note that we distinguish between *basis* functions B_k and the *basic* function B. We use this terminology to emphasize that there is one basic function B which generates the basis via shifts to the various centers.

Coming back to the scattered data problem, we find the coefficients c_k by solving the linear system

$$\begin{bmatrix} |x_1 - x_1| & |x_1 - x_2| & \ldots & |x_1 - x_N| \\ |x_2 - x_1| & |x_2 - x_2| & \ldots & |x_2 - x_N| \\ \vdots & \vdots & \ddots & \vdots \\ |x_N - x_1| & |x_N - x_2| & \ldots & |x_N - x_N| \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} f_1(x_1) \\ f_1(x_2) \\ \vdots \\ f_1(x_N) \end{bmatrix}. \tag{1.3}$$

As mentioned earlier, for higher space dimensions s such a data dependent basis is required. Thus, even though the construction of piecewise linear splines in higher space dimensions is a different one (they are closely associated with an underlying computational mesh), the idea just presented suggests a very simple generalization of univariate piecewise linear splines that works for any space dimension.

The matrix in (1.3) above is an example of a *distance matrix*. Such matrices have been studied in geometry and analysis in the context of isometric embeddings of metric spaces for a long time (see, *e.g.*, [Baxter (1991); Blumenthal (1938); Bochner (1941); Micchelli (1986); Schoenberg (1938a); Wells and Williams (1975)] and also Chapter 10). It is known that the distance matrix based on the Euclidean distance between a set of distinct points in $\mathbb{R}^s$ is always non-singular (see Section 9.3 for more details). Therefore, we can solve the scattered data interpolation problem we posed on $[0,1]^s$ by assuming

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{k=1}^{N} c_k \|\boldsymbol{x} - \boldsymbol{x}_k\|_2, \qquad \boldsymbol{x} \in [0,1]^s, \tag{1.4}$$

and then determine the coefficients c_k by solving the linear system

$$\begin{bmatrix} \|\boldsymbol{x}_1 - \boldsymbol{x}_1\|_2 & \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2 & \dots & \|\boldsymbol{x}_1 - \boldsymbol{x}_N\|_2 \\ \|\boldsymbol{x}_2 - \boldsymbol{x}_1\|_2 & \|\boldsymbol{x}_2 - \boldsymbol{x}_2\|_2 & \dots & \|\boldsymbol{x}_2 - \boldsymbol{x}_N\|_2 \\ \vdots & \vdots & \ddots & \vdots \\ \|\boldsymbol{x}_N - \boldsymbol{x}_1\|_2 & \|\boldsymbol{x}_N - \boldsymbol{x}_2\|_2 & \dots & \|\boldsymbol{x}_N - \boldsymbol{x}_N\|_2 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} f_s(\boldsymbol{x}_1) \\ f_s(\boldsymbol{x}_2) \\ \vdots \\ f_s(\boldsymbol{x}_N) \end{bmatrix}.$$

This is precisely the interpolation method we will choose to illustrate with our first MATLAB script `DistanceMatrixFit.m` (see Program 1.2 below) and the supporting figures and tables. A typical basis function for the Euclidean distance matrix fit, $B_k(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}_k\|_2$, is shown in Figure 1.2 for the case $\boldsymbol{x}_k = \mathbf{0}$ and $s = 2$.

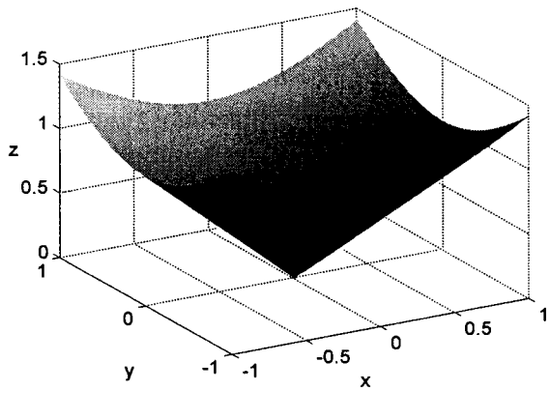


Fig. 1.2 A typical basis function for the Euclidean distance matrix centered at the origin in $\mathbb{R}^2$.

Before we discuss the actual interpolation program we first list a subroutine used in many of our later examples. It is called `DistanceMatrix.m` and we use it to compute the matrix of pairwise Euclidean distances of two (possibly different) sets of points in $\mathbb{R}^s$. In the code these two sets are denoted by `dsites` and `ctrs`. In most of our examples both of these sets will coincide with the set $\mathcal{X}$ of data sites.

Program 1.1. `DistanceMatrix.m`

```
% DM = DistanceMatrix(dsites,ctrs)
% Forms the distance matrix of two sets of points in R^s,
% i.e., DM(i,j) = || datasite_i - center_j ||_2.
% Input
%   dsites: Mxs matrix representing a set of M data sites in R^s
%              (i.e., each row contains one s-dimensional point)
%   ctrs:   Nxs matrix representing a set of N centers in R^s
%              (one center per row)
% Output
%   DM:     MxN matrix whose i,j position contains the Euclidean
%              distance between the i-th data site and j-th center
```

```
1  function DM = DistanceMatrix(dsites,ctrs)
2  [M,s] = size(dsites); [N,s] = size(ctrs);
3  DM = zeros(M,N);
   % Accumulate sum of squares of coordinate differences
   % The ndgrid command produces two MxN matrices:
   %   dr, consisting of N identical columns (each containing
   %       the d-th coordinate of the M data sites)
   %   cc, consisting of M identical rows (each containing
   %       the d-th coordinate of the N centers)
4  for d=1:s
5     [dr,cc] = ndgrid(dsites(:,d),ctrs(:,d));
6     DM = DM + (dr-cc).^2;
7  end
8  DM = sqrt(DM);
```

Note that this subroutine can easily be modified to produce a p-norm distance matrix by making the obvious changes to lines 6 and 8 of the code in Program 1.1. We will come back to this idea in Chapter 10.

Our first main script is Program 1.2. This script can be used to compute the distance matrix interpolant to data sampled from the test function f_s provided by Program C.1. We use Halton points and are able to select the space dimension s and number of points N by editing lines 1 and 2 of the code. The subroutine `MakeSDGrid.m` which we use to compute the equally spaced points in the s-dimensional unit cube on line 6 of `DistanceMatrixFit.m` is provided in Appendix C. These equally spaced points are used as evaluation points and to compute errors. Note that since the distance matrix interpolant is of the form (1.4) its simultaneous evaluation at the entire set of evaluation points amounts to a matrix-vector product of the evaluation matrix `EM` and the coefficients $\boldsymbol{c}$. Here the evaluation matrix has the same structure as the interpolation matrix and can also be computed using the subroutine `Distancematrix.m` (only using evaluation points in place of the data sites, see line 9 of `DistanceMatrixFit.m`). The coefficient vector $\boldsymbol{c}$ is supplied directly as solution of the linear system $A\boldsymbol{c} = \boldsymbol{f}$ (see (1.3) and the MATLAB expression `IM\rhs` on line 10 of the program). The evaluation points are subsequently used for the error computation in lines 11–13 and are also used for plotting purposes in the last part of the program (lines 16–35). Note that for this example we know the function f_s that generated the data, and therefore are able to compute the error in our reconstruction. The subroutines that produce the 2D and 3D plots on lines 24–32 are provided in Appendix C. Note that the use of `reshape` on lines 22–23 and 27–29 corresponds to the use of `meshgrid` for plotting purposes.

Program 1.2. `DistanceMatrixFit.m`

```
% DistanceMatrixFit
% Script that uses Euclidean distance matrices to perform
```

```
% scattered data interpolation for arbitrary space dimensions
% Calls on: DistanceMatrix, MakeSDGrid, testfunction
% Uses:     haltonseq (written by Daniel Dougherty from MATLAB
%                      Central File Exchange)
 1  s = 3;
 2  k = 2; N = (2^k+1)^s;
 3  neval = 10; M = neval^s;
    % Use Halton points as data sites and centers
 4  dsites = haltonseq(N,s);
 5  ctrs = dsites;
    % Create neval^s equally spaced evaluation locations in the
    % s-dimensional unit cube
 6  epoints = MakeSDGrid(s,neval);
    % Create right-hand side vector,
    % i.e., evaluate the test function at the data sites
 7  rhs = testfunction(s,dsites);
    % Compute distance matrix for the data sites and centers
 8  IM = DistanceMatrix(dsites,ctrs);
    % Compute distance matrix for evaluation points and centers
 9  EM = DistanceMatrix(epoints,ctrs);
    % Evaluate the interpolant on evaluation points
    % (evaluation matrix * solution of interpolation system)
10  Pf = EM * (IM\rhs);
    % Compute exact solution,
    % i.e., evaluate test function on evaluation points
11  exact = testfunction(s,epoints);
    % Compute maximum and RMS errors on evaluation grid
12  maxerr = norm(Pf-exact,inf);
13  rms_err = norm(Pf-exact)/sqrt(M);
14  fprintf('RMS error:     %e\n', rms_err)
15  fprintf('Maximum error: %e\n', maxerr)
16  switch s
17     case 1
18        plot(epoints, Pf)
19        figure; plot(epoints, abs(Pf-exact))
20     case 2
21        fview = [-30,30];
22        xe = reshape(epoints(:,2),neval,neval);
23        ye = reshape(epoints(:,1),neval,neval);
24        PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview);
25        PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview);
26     case 3
```

```
27        xe = reshape(epoints(:,2),neval,neval,neval);
38        ye = reshape(epoints(:,1),neval,neval,neval);
29        ze = reshape(epoints(:,3),neval,neval,neval);
30        xslice = .25:.25:1; yslice = 1; zslice = [0,0.5];
31        PlotSlices(xe,ye,ze,Pf,neval,xslice,yslice,zslice);
32a       PlotErrorSlices(xe,ye,ze,Pf,exact,neval,...
32b           xslice,yslice,zslice);
33    otherwise
34        disp('Cannot display plots for s>3')
35 end
```

In Tables 1.1 and 1.2 as well as Figures 1.3 and 1.4 we present some examples computed with Program 1.2. The number M of evaluation points (determined by neval on line 3 of the code) we used for the cases $s = 1, 2, \ldots, 6$, was 1000, 1600, 1000, 256, 1024, and 4096, respectively (*i.e.*, neval = 1000, 40, 10, 4, 4, and 4, respectively). Note that, as the space dimension s increases, more and more of the evaluation points lie on the boundary of the domain, while the data sites (which are given as Halton points) are located in the interior of the domain. The value k listed in Tables 1.1 and 1.2 is the same as the k in line 2 of Program 1.2. The formula for the root-mean-square error (RMS-error) is given by

$$\text{RMS-error} = \sqrt{\frac{1}{M}\sum_{j=1}^{M}\left[\mathcal{P}_f(\boldsymbol{\xi}_j) - f(\boldsymbol{\xi}_j)\right]^2} = \frac{1}{\sqrt{M}}\|\mathcal{P}_f - f\|_2, \tag{1.5}$$

where the $\boldsymbol{\xi}_j$, $j = 1, \ldots, M$ are the *evaluation points*. Formula (1.5) is used on line 13 of Program 1.2.

The basic MATLAB code for the solution of any kind of RBF interpolation problem will be very similar to Program 1.2. Note in particular that the data used — even for the distance matrix interpolation considered here — can also be "real" data. In that case one simply needs to replace lines 4 and 7 of the program by appropriate code that generates the data sites and data values for the right-hand side.

The plots on the left of Figures 1.3 and 1.4 display the graphs of the distance matrix fits for space dimensions $s = 1$, 2, and 3, respectively, while those on the right depict the corresponding errors. For the 1D plots (in Figure 1.3) we used 5 Halton points to interpolate the testfunction f_1. The piecewise linear nature of the interpolant is clearly visible at this resolution. If we use more points then the fit becomes more accurate — see Table 1.1 — but then it is no longer possible to distinguish the piecewise linear nature of the interpolant. The 2D plot (top left of Figure 1.4) interpolates the testfunction f_2 at 289 Halton points. The graph of $\mathcal{P}_f$ is false-colored according to the absolute error (indicated by the color bar at the right of the plot). The bottom plot in Figure 1.4 shows a slice plot of the distance matrix interpolant to f_3 based on 729 Halton points. For this plot the colors represent function values (again indicated by the color bar on the right).

Table 1.1 Distance matrix fit to N Halton points in $[0,1]^s$, $s = 1, 2, 3$.

	1D		2D		3D	
k	N	RMS-error	N	RMS-error	N	RMS-error
1	3	5.896957e-001	9	1.937341e-001	27	9.721476e-002
2	5	3.638027e-001	25	6.336315e-002	125	6.277141e-002
3	9	1.158328e-001	81	2.349093e-002	729	2.759452e-002
4	17	3.981270e-002	289	1.045010e-002		
5	33	1.406188e-002	1089	4.326940e-003		
6	65	5.068541e-003	4225	1.797430e-003		
7	129	1.877013e-003				
8	257	7.264159e-004				
9	513	3.016376e-004				
10	1025	1.381896e-004				
11	2049	6.907386e-005				
12	4097	3.453179e-005				

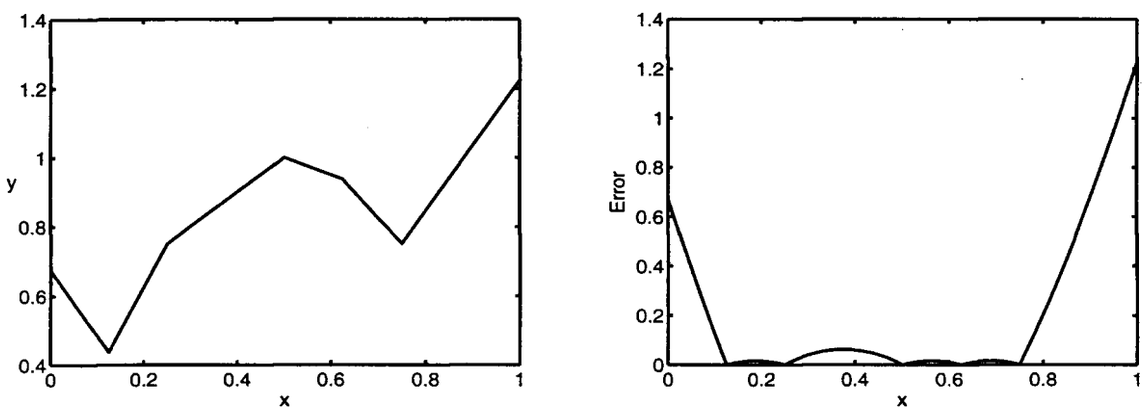


Fig. 1.3 Fit (left) and absolute error (right) for 5 point distance matrix interpolation in 1D.

In the right half of Figures 1.3 and 1.4 we show absolute errors for the distance matrix interpolants displayed in the left column. We use analogous color schemes, *i.e.*, the 2D plot (top part of Figure 1.4) is false-colored according to the absolute error, and so is the 3D plot (bottom) since now the "function value" corresponds to the absolute error. We can see clearly that most of the error is concentrated near the boundary of the domain. In fact, the absolute error is about one order of magnitude larger near the boundary than it is in the interior of the domain. This is no surprise since the data sites are located in the interior. However, even for uniformly spaced data sites (including points on the boundary) the main error in radial basis function interpolation is usually located near the boundary.

From this first simple example we can observe a number of other features. Most of them are characteristic for the radial basis function interpolants we will be studying later on. First, the basis functions employed, $B_k = \| \cdot - \boldsymbol{x}_k \|_2$, are radially sym-

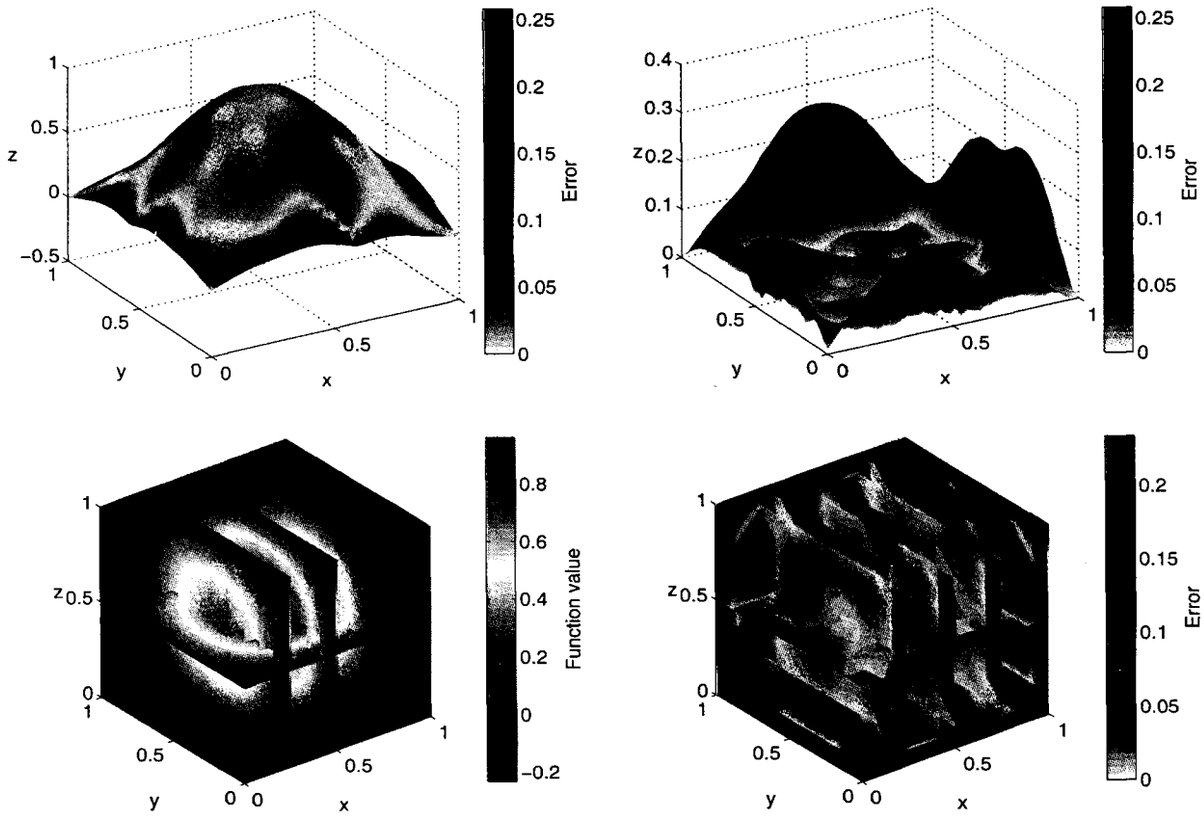


Fig. 1.4 Fits (left) and errors (right) for distance matrix interpolation with 289 points in 2D (top), and 729 points in 3D (bottom).

Table 1.2 Distance matrix fit to N Halton points in $[0,1]^s$, $s = 4, 5, 6$.

	4D		5D		6D	
k	N	RMS-error	N	RMS-error	N	RMS-error
1	81	1.339581e-001	243	9.558350e-002	729	5.097600e-002
2	625	6.817424e-002	3125	3.118905e-002		

metric. Second, as the MATLAB scripts show, the method is extremely simple to implement for any space dimension s. For example, no underlying computational mesh is required to compute the interpolant. The process of mesh generation is a major factor when working in higher space dimensions with polynomial-based methods such as splines or finite elements. All that is required for our method is the pairwise distance between the data sites. Therefore, we have what is known as a *meshfree* (or *meshless*) method.

Third, the accuracy of the method improves if we add more data sites. In fact, it seems that the RMS-error in Tables 1.1 and 1.2 is reduced by a factor of about two from one row to the next. Since we use $(2^k + 1)^s$ uniformly distributed random

data points in row k this indicates a convergence rate of roughly $\mathcal{O}(h)$, where h can be viewed as something like the average distance or meshsize of the set $\mathcal{X}$ of data sites (we will be more precise later on).

Another thing to note is that the simple distance function interpolant used here (as well as many other radial basis function interpolants used later) requires the solution of a system of linear equations with a dense $N \times N$ matrix. This makes it very costly to apply the method in its simple form to large data sets. Moreover, as we will see later, these matrices also tend to be rather ill-conditioned. These are the reasons why we can only present results for relatively small data sets in higher space dimensions using this simple approach.

In the remainder of this book it is our goal to present alternatives to this basic interpolation method that address the problems mentioned above such as limitation to small data sets, ill-conditioning, limited accuracy and limited smoothness of the interpolant.

1.2 Some Historical Remarks

Originally, the motivation for the basic meshfree approximation methods (radial basis function and moving least squares methods) came from applications in geodesy, geophysics, mapping, or meteorology. Later, applications were found in many other areas such as in the numerical solution of PDEs, computer graphics, artificial intelligence, statistical learning theory, neural networks, signal and image processing, sampling theory, statistics (kriging), finance, and optimization. It should be pointed out that meshfree local regression methods have been used independently in statistics for well over 100 years (see, *e.g.*, [Cleveland and Loader (1996)] and the references therein). In fact, the basic moving least squares method (known also as local regression in the statistics literature) can be traced back at least to the work of [Gram (1883); Woolhouse (1870); De Forest (1873); De Forest (1874)].

In the literature on approximation theory and related applications areas some historical landmark contributions have come from

- Donald Shepard, who as an undergraduate student at Harvard University, suggested the use of what are now called *Shepard functions* in the late 1960s (see Chapter 22). The publication [Shepard (1968)] discusses the basic inverse distance weighted Shepard method and some modifications thereof. The method was at the time incorporated into a computer program, SYMAP, for map making.
- Rolland Hardy, who was a geodesist at Iowa State University. He introduced the so-called *multiquadrics* (MQs) in the early 1970s (see, *e.g.*, [Hardy (1971)] or Chapter 8). Hardy's work was primarily concerned with applications in geodesy and mapping.

- Robert L. Harder and Robert N. Desmarais, who were aerospace engineers at MacNeal-Schwendler Corporation (MSC Software), and NASA's Langley Research Center. They introduced the so-called *thin plate splines* (TPSs) in 1972 (see, *e.g.*, [Harder and Desmarais (1972)] or Chapter 8). Their work was concerned mostly with aircraft design.
- Jean Duchon, a mathematician at the Université Joseph Fourier in Grenoble, France. Duchon suggested a variational approach minimizing the integral of $\nabla^2 f$ in $\mathbb{R}^2$ which also leads to the thin plate splines. This work was done in the mid 1970s and is considered to be the foundation of the variational approach to radial basis functions (see [Duchon (1976); Duchon (1977); Duchon (1978); Duchon (1980)]) or Chapter 13).
- Jean Meinguet, a mathematican at Université Catholique de Louvain in Louvain, Belgium. Meinguet introduced what he called *surface splines* in the late 1970s. Surface splines and thin plate splines fall under what we will refer to as *polyharmonic splines* (see, *e.g.*, [Meinguet (1979a); Meinguet (1979b); Meinguet (1979c); Meinguet (1984)] or Chapter 8).
- Peter Lancaster and Kes Šalkauskas, mathematicians at the University of Calgary, Canada. They published [Lancaster and Šalkauskas (1981)] introducing the *moving least squares method* (a generalization of Shepard functions).
- Richard Franke, a mathematician at the Naval Postgraduate School in Monterey, California. In [Franke (1982a)] he compared various scattered data interpolation methods, and concluded MQs and TPSs were the best. Franke also conjectured that the interpolation matrix for MQs is invertible.
- Wolodymyr (Wally) Madych, a mathematician at the University of Connecticut, and Stuart Alan Nelson, a mathematician from Iowa State University. In 1983 they completed their manuscript [Madych and Nelson (1983)] in which they proved Franke's conjecture (and much more) based on a variational approach. However, this manuscript was never published. Other fundamental papers by these two authors are, *e.g.*, [Madych and Nelson (1988); Madych and Nelson (1990a); Madych and Nelson (1992)].
- Charles Micchelli, a mathematician at the IBM Watson Research Center. Micchelli published the paper [Micchelli (1986)]. He also proved Franke's conjecture. His proofs are rooted in the work of [Bochner (1932); Bochner (1933)] and [Schoenberg (1937); Schoenberg (1938a); Schoenberg (1938b)] on positive definite and completely monotone functions. This is also the approach we will follow throughout much of this book.
- Grace Wahba, a statistician at the University of Wisconsin. She studied the use of thin plate splines for statistical purposes in the context of smoothing *noisy data* and data on spheres, and introduced the ANOVA and cross validation approaches to the radial basis function setting(see, *e.g.*, [Wahba

(1979); Wahba (1981); Wahba and Wendelberger (1980)]). One of the first monographs on the subject is [Wahba (1990b)].

- Robert Schaback, a mathematician at the University of Göttingen, Germany. *Compactly supported radial basis functions* (CSRBFs) were introduced in [Schaback (1995a)], and a very popular family of CSRBFs was presented by Holger Wendland (also a mathematician in Göttingen) in his Ph.D. thesis (see also [Wendland (1995)] and Chapter 11). Both of these authors have contributed extensively to the field of radial basis functions. We mention particularly the recent monograph [Wendland (2005a)].

Chapter 2

Radial Basis Function Interpolation in MATLAB

Before we discuss any of the theoretical foundation of radial basis functions we want to get a feel for what they are all about. We saw in the introductory chapter that it is easy to use Euclidean distance matrices to compute a solution to the scattered data interpolation problem. However, we also pointed out a number of limitations to that approach such as the limited accuracy and limited smoothness. It turns out that we can maintain the underlying structure presented by the distance matrix approach and address these limitations by composing the distance function with certain "good" univariate functions.

2.1 Radial (Basis) Functions

As a first example we pick a function well-represented in many branches of mathematics, namely the *Gaussian*

$$\varphi(r) = e^{-(\varepsilon r)^2}, \qquad r \in \mathbb{R}.$$

Our *shape parameter* ε is related to the variance σ^2 of the normal distribution function by $\varepsilon^2 = 1/(2\sigma^2)$. If we compose the Gaussian with the Euclidean distance function $\|\cdot\|_2$ we obtain for any fixed *center* $\boldsymbol{x}_k \in \mathbb{R}^s$ a *multivariate* function

$$\Phi_k(\boldsymbol{x}) = e^{-\varepsilon^2 \|\boldsymbol{x}-\boldsymbol{x}_k\|_2^2}, \qquad \boldsymbol{x} \in \mathbb{R}^s.$$

Obviously, the connection between Φ_k and φ is given by

$$\Phi_k(\boldsymbol{x}) = \varphi(\|\boldsymbol{x} - \boldsymbol{x}_k\|_2).$$

It is this connection that gives rise to the name *radial basis function* (RBF). The following is a formal definition of a radial function.

Definition 2.1. A function $\Phi : \mathbb{R}^s \to \mathbb{R}$ is called *radial* provided there exists a *univariate* function $\varphi : [0, \infty) \to \mathbb{R}$ such that

$$\Phi(\boldsymbol{x}) = \varphi(r), \quad \text{where} \quad r = \|\boldsymbol{x}\|,$$

and $\|\cdot\|$ is some norm on $\mathbb{R}^s$ — usually the Euclidean norm.

Definition 2.1 says that for a radial function Φ

$$\|\boldsymbol{x}_1\| = \|\boldsymbol{x}_2\| \quad \Longrightarrow \quad \Phi(\boldsymbol{x}_1) = \Phi(\boldsymbol{x}_2), \qquad \boldsymbol{x}_1,\ \boldsymbol{x}_2 \in \mathbb{R}^s.$$

In other words, the value of Φ at any point at a certain fixed distance from the origin (or any other fixed center point) is constant. Thus, Φ is radially (or spherically) symmetric about its center. Definition 2.1 shows that the Euclidean distance function we used in the introduction is just a special case of a radial (basis) function. Namely, with $\varphi(r) = r$.

Figure 2.1 shows the graphs of two Gaussian radial basis functions, one with shape parameter $\varepsilon = 1$ (left) and one with $\varepsilon = 3$ (right) (both centered at the origin in $\mathbb{R}^2$). A smaller value of ε (*i.e.*, larger variance) causes the function to become "flatter", while increasing ε leads to a more peaked RBF, and therefore localizes its influence. We will see soon that the choice of ε has a profound influence on both the accuracy and numerical stability of the solution to our interpolation problem.

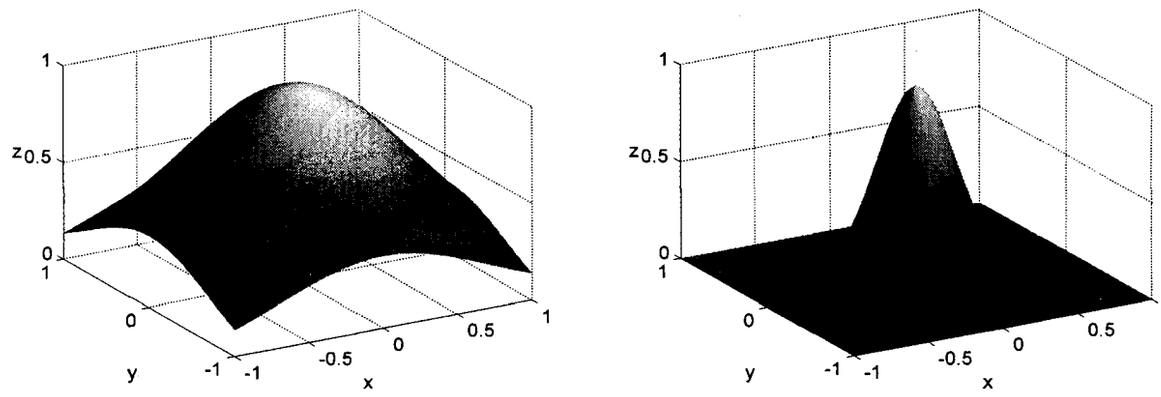


Fig. 2.1 Gaussian with $\varepsilon = 1$ (left) and $\varepsilon = 3$ (right) centered at the origin in $\mathbb{R}^2$.

Definition 2.1 and the discussion leading up to it show again why it makes sense to call φ the *basic* function, and $\Phi_k(\|\cdot\|_2)$ (centered at $\boldsymbol{x}_k$) a radial *basis* function. One single basic function generates all of the basis functions that are used in the expansion (1.1).

Radial function interpolants have the nice property of being invariant under all Euclidean transformations (*i.e.*, translations, rotations, and reflections). By this we mean that it does not matter whether we first compute the RBF interpolant and then apply a Euclidean transformation, or if we first transform the data and then compute the interpolant. This is an immediate consequence of the fact that Euclidean transformations are characterized by orthogonal transformation matrices and are therefore 2-norm-invariant. Invariance under translation, rotation and reflection is often desirable in applications.

Moreover, the application of radial functions to the solution of the scattered data interpolation problem (as well as many other multivariate approximation problems) benefits from the fact that the interpolation problem becomes insensitive to the

dimension s of the space in which the data sites lie. Instead of having to deal with a multivariate function Φ (whose complexity will increase with increasing space dimension s) we can work with the same univariate function φ for all choices of s.

2.2 Radial Basis Function Interpolation

Instead of using simple distance matrices as we did earlier, we now use a radial basis function expansion to solve the scattered data interpolation problem in $\mathbb{R}^s$ by assuming

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{k=1}^{N} c_k \varphi\left(\|\boldsymbol{x} - \boldsymbol{x}_k\|_2\right), \qquad \boldsymbol{x} \in \mathbb{R}^s. \tag{2.1}$$

The coefficients c_k are found by enforcing the interpolation conditions, and thus solving the linear system

$$\begin{bmatrix} \varphi(\|\boldsymbol{x}_1 - \boldsymbol{x}_1\|_2) & \varphi(\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2) & \dots & \varphi(\|\boldsymbol{x}_1 - \boldsymbol{x}_N\|_2) \\ \varphi(\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|_2) & \varphi(\|\boldsymbol{x}_2 - \boldsymbol{x}_2\|_2) & \dots & \varphi(\|\boldsymbol{x}_2 - \boldsymbol{x}_N\|_2) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi(\|\boldsymbol{x}_N - \boldsymbol{x}_1\|_2) & \varphi(\|\boldsymbol{x}_N - \boldsymbol{x}_2\|_2) & \dots & \varphi(\|\boldsymbol{x}_N - \boldsymbol{x}_N\|_2) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} f(\boldsymbol{x}_1) \\ f(\boldsymbol{x}_2) \\ \vdots \\ f(\boldsymbol{x}_N) \end{bmatrix}.$$

As the solution of the scattered data interpolation problem hinges entirely on the solution of this system of linear equations we will devote the next chapter to the question of when (*i.e.*, for what type of basic functions φ) the system matrix is non-singular.

For the numerical example presented below we restrict ourselves to the two-dimensional case $s = 2$. As basic function φ we will use both Gaussians and the linear function $\varphi(r) = r$ which gives rise to the Euclidean distance matrix approach used earlier.

The code of the MATLAB script `RBFInterpolation2D.m` (see Program 2.1) we use to perform RBF interpolation in 2D is very similar to the earlier script `DistanceMatrixFit.m`. It also makes use of the subroutine `DistanceMatrix.m`. While it is easy to write a version of the interpolation script that works for any space dimension s (just as we did in `DistanceMatrixFit.m`) we will stick with a basic 2D version here.

In line 1 we define the Gaussian RBF as a MATLAB `anonymous` function that accepts a matrix argument (namely the output from `DistanceMatrix`) along with its shape parameter. Note that this feature is only available since MATLAB Release 7. For older MATLAB versions we suggest using an `inline` function instead (see the programs in the folder Matlab6 of the enclosed CD). If execution speed is important, then one should explicitly provide the function (either hardcoded directly where needed, or as an `M-file`). This latter approach will always be more efficient than the `inline` or even `anonymous` function approach. However, then the interpolation program is no longer as generic.

We can replace the definition of the Gaussian on line 1 by the definition of the linear function $\varphi(r) = r$ or any other admissible RBF we will encounter later. In lines 2–6 we define a test function that we will sample similarly to the function f_s used in the introductory example. Here (and in many later examples) we use *Franke's function*

$$f(x,y) = \frac{3}{4}e^{-1/4((9x-2)^2+(9y-2)^2)} + \frac{3}{4}e^{-(1/49)(9x+1)^2-(1/10)(9y+1)^2} + \frac{1}{2}e^{-1/4((9x-7)^2+(9y-3)^2)} - \frac{1}{5}e^{-(9x-4)^2-(9y-7)^2} \tag{2.2}$$

which is a standard test function for 2D scattered data fitting. Note that we used (x,y) to denote the two components of $\boldsymbol{x} \in \mathbb{R}^2$. The graph of Franke's function over the unit square is shown in Figure 2.2.

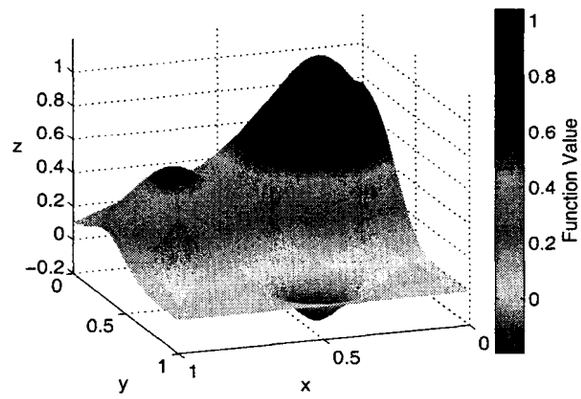


Fig. 2.2 Franke's test function.

For many of our examples we use data locations that have been saved in files named `Data2D_%d%s` where the number of points (`%d`) is taken from the progression $\{(2^k+1)^2\} = \{9, 25, 81, 289, 1089, 4225, \ldots\}$. The characters `u` or `h` (in place of `%s`) are used to denote either uniformly spaced points, or Halton points in the unit square. The set of data points is defined and loaded in lines 7 and 8. As in the earlier example we consider here only the case where the centers for the RBFs coincide with the data locations (line 9).

A grid of evaluation points used to evaluate our interpolant for the purposes of rendering and error computation is defined in lines 10 and 11. The test data (right-hand side of the interpolation equations) are computed on line 12 where the test function is sampled at the data sites.

The main part of the code is given by lines 13–17. Note that this part is very similar to the corresponding segment (lines 7–9) in `DistanceMatrixFit.m`. The only difference is that we now apply the basic function φ to the entire distance matrices in order to obtain the interpolation and evaluation matrices.

Program 2.1. RBFInterpolation2D.m

```
% RBFInterpolation2D
% Script that performs basic 2D RBF interpolation
% Calls on: DistanceMatrix
    % Define the Gaussian RBF and shape parameter
 1  rbf = @(e,r) exp(-(e*r).^2); ep = 21.1;
    % Define Franke's function as testfunction
 2  f1 = @(x,y) 0.75*exp(-((9*x-2).^2+(9*y-2).^2)/4);
 3  f2 = @(x,y) 0.75*exp(-((9*x+1).^2/49+(9*y+1).^2/10));
 4  f3 = @(x,y) 0.5*exp(-((9*x-7).^2+(9*y-3).^2)/4);
 5  f4 = @(x,y) 0.2*exp(-((9*x-4).^2+(9*y-7).^2));
 6  testfunction = @(x,y) f1(x,y)+f2(x,y)+f3(x,y)-f4(x,y);
 7  N = 1089; gridtype = 'h';
    % Load data points
 8  name = sprintf('Data2D_%d%s',N,gridtype); load(name)
 9  ctrs = dsites;
10  neval = 40; grid = linspace(0,1,neval);
11  [xe,ye] = meshgrid(grid); epoints = [xe(:) ye(:)];
    % Evaluate the test function at the data points
12  rhs = testfunction(dsites(:,1),dsites(:,2));
    % Compute distance matrix between the data sites and centers
13  DM_data = DistanceMatrix(dsites,ctrs);
    % Compute interpolation matrix
14  IM = rbf(ep,DM_data);
    % Compute distance matrix between evaluation points and centers
15  DM_eval = DistanceMatrix(epoints,ctrs);
    % Compute evaluation matrix
16  EM = rbf(ep,DM_eval);
    % Compute RBF interpolant
    % (evaluation matrix * solution of interpolation system)
17  Pf = EM * (IM\rhs);
    % Compute exact solution, i.e.,
    % evaluate test function on evaluation points
18  exact = testfunction(epoints(:,1),epoints(:,2));
    % Compute errors on evaluation grid
19  maxerr = norm(Pf-exact,inf);
20  rms_err = norm(Pf-exact)/neval;
21  fprintf('RMS error:     %e\n', rms_err)
22  fprintf('Maximum error: %e\n', maxerr)
23  fview = [160,20]; % for Franke's function
24  PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview);
25  PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview);
```

In Table 2.1 we report the results of a series of experiments in which we compute Gaussian RBF and distance matrix interpolants to increasingly larger sets of data. We use one fixed value of ε for all of the experiments with the Gaussians. This type of approximation is known as *non-stationary* approximation. Its counterpart is known as *stationary* approximation.

Even though we do not perform stationary interpolation in this experiment we take a minute to explain the essential difference between the two approaches. In the stationary setting we would scale the shape parameter ε according to the *fill distance* (or meshsize) h so that we end up using "peaked" basis functions for densely spaced data and "flat" basis functions for coarsely spaced data. We will use the fill distance as a measure of the data distribution. The fill distance is usually defined as

$$h = h_{\mathcal{X},\Omega} = \sup_{\boldsymbol{x}\in\Omega} \min_{\boldsymbol{x}_j\in\mathcal{X}} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2, \tag{2.3}$$

and it indicates how well the data in the set $\mathcal{X}$ fill out the domain Ω. A geometric interpretation of the fill distance is given by the radius of the largest possible empty ball that can be placed among the data locations inside Ω (see Figure 2.3). Sometimes the synonym *covering radius* is used. In our MATLAB code we can estimate the fill distance via

```
hX = max(min(DM_eval'))
```
(2.4)

where `DM_eval` is the matrix consisting of pairwise distances between the evaluation points (placed on a fine uniform grid in Ω) and the data sites $\mathcal{X}$ (*c.f.* line 15 of Program 2.1). Note that we transpose the non-symmetric evaluation matrix. This corresponds to finding — for each evaluation point — the distance to the corresponding closest data site, and then setting $h_{\mathcal{X},\Omega}$ as the worst of those distances. Figure 2.3 illustrates the fill distance for a set of 25 Halton points. Note that in this case the largest "hole" in the data is near the boundary.

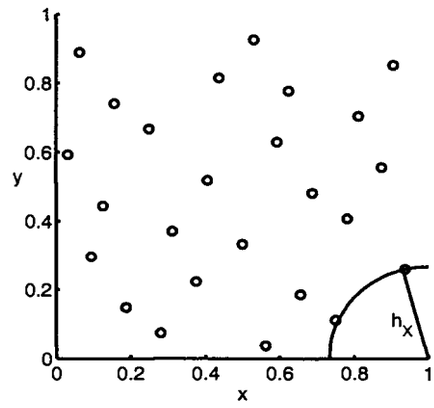


Fig. 2.3 The fill distance for $N = 25$ Halton points ($h_{\mathcal{X},\Omega} \approx 0.2667$).

We will take a closer look at the differences between stationary and non-stationary interpolation in later chapters of this book.

In the following examples we will clearly see the effects the shape parameter has on the condition number of the interpolation matrix (and therefore the numerical stability) of our computations. In order to be able to use our script `RBFInterpolation2D.m` in conjunction with Gaussians to produce a meaningful sequence of non-stationary experiments, *i.e.*, with a fixed value of the shape parameter ε, we are required to take the fairly large value $\varepsilon = 21.1$. Otherwise computation with the relatively densely spaced point set of $N = 4225$ Halton points results in MATLAB warnings of ill-conditioning. This means that — for the non-stationary approach — the basis functions are too localized on the smaller point sets, and the approximation is very poor (see Figure 2.4).

The test results for a non-stationary interpolation experiment using Gaussians and Euclidean distance matrices for Franke's function are shown in Table 2.1. As just pointed out, we note that a fit with Gaussians and a small shape parameter such as $\varepsilon = 1$ would quickly lead to a numerical breakdown. For as few as $N = 25$ data points and $\varepsilon = 1$ MATLAB issues a "matrix close to singular" warning with an estimated reciprocal condition number of `RCOND=3.986027e-020`.

Table 2.1 Non-stationary RBF interpolation to Franke's function using Gaussians ($\varepsilon = 21.1$) and Euclidean distance matrices.

		Gaussian		distance matrix	
k	N	RMS-error	max-error	RMS-error	max-error
1	9	3.647169e-001	1.039682e+000	1.323106e-001	4.578028e-001
2	25	3.203404e-001	9.670980e-001	6.400558e-002	2.767871e-001
3	81	2.152222e-001	8.455161e-001	1.343780e-002	6.733130e-002
4	289	7.431729e-002	7.219253e-001	3.707360e-003	3.057540e-002
5	1089	1.398297e-002	3.857234e-001	1.143589e-003	1.451950e-002
6	4225	4.890709e-004	1.940675e-002	4.002749e-004	8.022336e-003

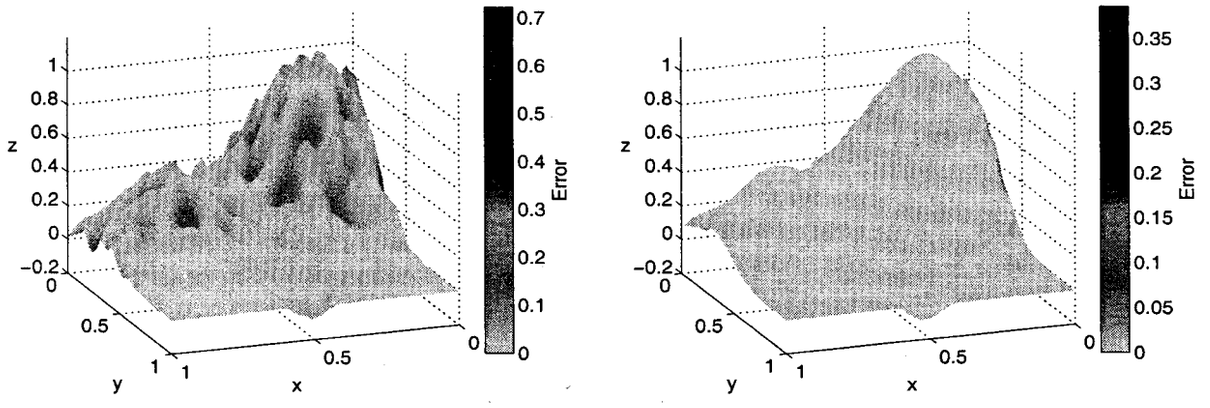


Fig. 2.4 Gaussian RBF interpolant with $\varepsilon = 21.1$ at $N = 289$ (left) and at $N = 1089$ Halton points (right).

If we look at the entries in Table 2.1 then we see that — contrary to what we announced earlier — the results based on the distance matrix fit are more accurate than those obtained with Gaussians. This will change, however, if we try to optimize our choice of the shape parameter ε (see the results of the next experiment in Table 2.2).

In a second experiment we consider the same test function and Gaussian basis functions. Now, however, we want to study the effects of the shape parameter. Therefore, in Figure 2.5 we display both the maximum and RMS errors as a function of the shape parameter ε for four fixed data sets (81, 289, 1089 and 4225 Halton points). These curves reveal some of the problems associated with radial basis function interpolation — especially when working with globally supported basis functions, *i.e.*, dense matrices. We see that the errors decrease with decreasing ε (of course, they also decrease with decreasing fill distance — but that is not what we are concerned with now). However, the error curves are not monotonic. We can identify an optimal value of ε for which both errors are minimal (the minima of the two error curves occur at almost the same place). Moreover, there is a value of ε at which the computational results become unpredictable, and the error curves become erratic. This point is associated with severe ill-conditioning of the system matrix. Since MATLAB issues a warning when attempting to solve an ill-conditioned linear system, we refer to the smallest value of ε for which we do not see a MATLAB warning as the "safe" value of ε (for a given set $\mathcal{X}$ of data sites and basic function φ). The interesting fact about the four plots displayed in Figure 2.5 is that for the smaller data sets ($N = 81$ and $N = 289$) the minimum errors are obtained for a "safe" ε, while for the larger sets ($N = 1089$ and $N = 4225$) the minimum errors are obtained in the "unsafe" range. Therefore, we are computing in a certain "gray zone". We are obtaining highly accurate solutions from severely ill-conditioned linear systems. We will come back later to this interesting feature of radial basis function interpolation (called *uncertainty* or *trade-off principle*). It is conceivable (and in fact possible [Fornberg and Wright (2004)]) to obtain even more accurate results by using a more stable way to evaluate the radial basis function interpolant (see the discussion in Chapters 16 and 17).

In Table 2.2 we list the "best possible" results for stationary Gaussian interpolation. We view "best" in two different ways. For the results presented in columns 3–5 we select for each choice of N the smallest possible value of ε such that MATLAB does not issue a warning. We refer to this case as the "safe" ε case in Table 2.2. Most of these errors are now comparable (or smaller) than those for simple distance matrix interpolation (*c.f.* Table 2.1).

These results, however, do not always represent the smallest achievable error. Therefore, we present (in columns 6–8) results for those "optimal" values of ε which yield the smallest RMS-error. These results are obtained in the "gray zone" mentioned above. For example, if we use $N = 1089$ Halton points and shape parameter $\varepsilon = 6.2$ then MATLAB issues a warning with `RCOND = 2.683527e-020`. However,

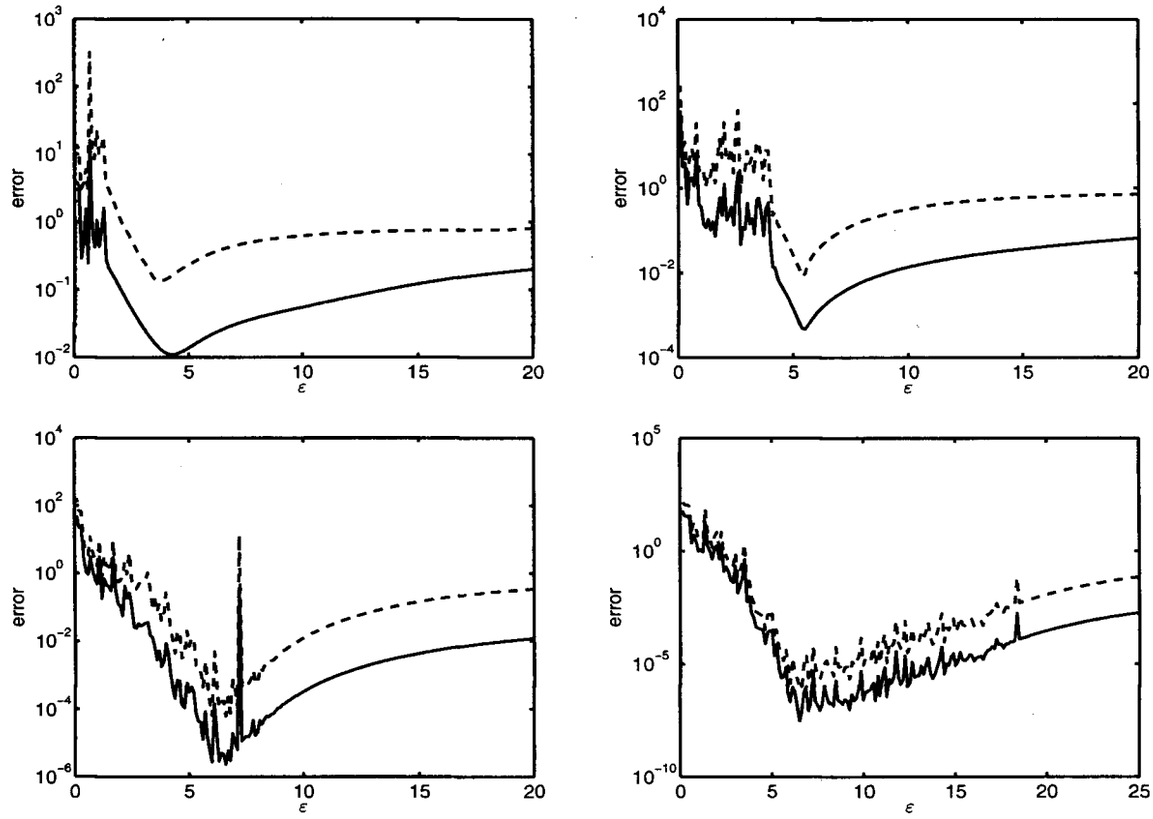


Fig. 2.5 Maximum (dashed/top curve) and RMS (solid/bottom curve) errors vs. ε for 81 (top left), 289 (top right), 1089 (bottom left), and 4225 Halton points (bottom right).

Table 2.2 "Optimal" RBF interpolation to Franke's function using Gaussians.

		smallest "safe" ε			smallest RMS-error		
k	N	ε	RMS-error	max-error	ε	RMS-error	max-error
1	9	0.02	3.658421e-001	1.580259e+000	2.23	1.118026e-001	3.450275e-001
2	25	0.32	3.629342e-001	2.845554e+000	3.64	4.032550e-002	2.996488e-001
3	81	1.64	1.743059e-001	2.398284e+000	4.28	1.090601e-002	1.579465e-001
4	289	4.73	2.785388e-003	5.472502e-002	5.46	4.610079e-004	7.978283e-003
5	1089	10.5	4.945428e-004	1.812246e-002	6.2	2.498848e-006	8.779119e-005
6	4225	21.1	4.890709e-004	1.940675e-002	6.3	4.269292e-008	8.889552e-007

as we can see in Table 2.2 and Figure 2.5, the corresponding errors are now much smaller than those previously obtained. Moreover, the errors decrease at a rate that is faster than the $\mathcal{O}(h)$ we observed earlier for the distance matrix fit example.

Of course, if the data we are trying to fit are not sampled from a known test function then we will not be able to choose an "optimal" shape parameter by monitoring the RMS error. The associated issues of ill-conditioning, preconditioning, optimal shape parameter selection, and alternate stable evaluation methods via a Contour-Padé algorithm are studied later in Chapters 16 and 17.

Chapter 3

Positive Definite Functions

We noted in the previous chapters that the solution of the scattered data interpolation problem with RBFs boils down to the solution of a system of linear equations

$$A\mathbf{c} = \boldsymbol{y},$$

where the system matrix A has entries $\varphi(\|\boldsymbol{x}_j - \boldsymbol{x}_k\|_2)$, $j,k = 1,\ldots,N$. We know from linear algebra that this system will have a unique solution whenever the matrix A is non-singular. While no one has yet succeeded in characterizing the class of all basic functions φ that generate a non-singular system matrix for any set $\mathcal{X} = \{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N\}$ of distinct data sites, the situation is much better if we consider *positive definite matrices.*

In this chapter we present the main theoretical results underlying this approach along with some of their proofs. A series of examples are presented in the next chapter. A comprehensive treatment of the mathematical theory needed for scattered data interpolation with strictly positive definite functions (see Def. 3.2 below) is presented in the recent monograph [Wendland (2005a)].

3.1 Positive Definite Matrices and Functions

Definition 3.1. A real symmetric matrix A is called *positive semi-definite* if its associated quadratic form is non-negative, *i.e.*,

$$\sum_{j=1}^{N}\sum_{k=1}^{N} c_j c_k A_{jk} \geq 0 \tag{3.1}$$

for $\mathbf{c} = [c_1,\ldots,c_N]^T \in \mathbb{R}^N$.

If the quadratic form (3.1) is zero only for $\mathbf{c} \equiv \mathbf{0}$, then A is called *positive definite.*

An important property of positive definite matrices is that all their eigenvalues are positive, and therefore a positive definite matrix is non-singular (but certainly not vice versa).

If we therefore had basis functions B_k in the expansion (1.1) that generate a positive definite interpolation matrix, we would always have a well-posed interpolation problem. To this end we introduce the concept of a *positive definite function* from classical analysis.

Positive definite functions were first considered in classical analysis early in the 20th century. [Mathias (1923)] seems to have been the first to define and study positive definite functions. An overview of the development of positive definite functions up to the mid 1970s can be found in [Stewart (1976)]. However, as we see from the definition below, positive definite functions were — unfortunately — defined in analogy to positive semi-definite matrices. Therefore, in order to meet our goal of having a well-posed interpolation problem, it is necessary to sharpen the classical notion of a positive definite function to that of a *strictly* positive definite one. This concept does not seem to have been studied until [Micchelli (1986)] made the connection between scattered data interpolation and positive definite functions. This leads to an unfortunate difference in terminology used in the context of matrices and functions. Instead of rewriting history we will adhere to this terminology here. We would like to point out that when reading recent articles (especially in the radial basis function literature) dealing with (strictly) positive definite functions one has to be aware of the fact that some authors have tried to "correct" history, and now refer to strictly positive definite functions as positive definite functions.

Definition 3.2. A complex-valued continuous function $\Phi : \mathbb{R}^s \to \mathbb{C}$ is called *positive definite on* $\mathbb{R}^s$ if

$$\sum_{j=1}^{N}\sum_{k=1}^{N} c_j\overline{c_k}\Phi(\boldsymbol{x}_j - \boldsymbol{x}_k) \geq 0 \tag{3.2}$$

for any N pairwise different points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^s$, and $\boldsymbol{c} = [c_1, \ldots, c_N]^T \in \mathbb{C}^N$.

The function Φ is called *strictly positive definite on* $\mathbb{R}^s$ if the quadratic form (3.2) is zero only for $\boldsymbol{c} \equiv \boldsymbol{0}$.

We note that even though we are interested in problems with real data and real coefficients, an extension of the notion of positive definiteness to cover complex coefficients $\boldsymbol{c}$ and complex-valued functions Φ as done in Definition 3.2 will be helpful when deriving some properties of (strictly) positive definite functions later on. Moreover, the celebrated *Bochner's theorem* (see Theorem 3.3) characterizes exactly the positive definite functions of Definition 3.2. In all practical circumstances, however, we will be concerned with real-valued functions only, and a characterization of such functions appears below as Theorem 3.2. It should also be noted that Definition 3.2 implies that only functions whose quadratic form is real are candidates for (strictly) positive definite functions.

Example 3.1. Here, and throughout this book, we will denote the standard inner product of $\boldsymbol{x}$ and $\boldsymbol{y}$ in $\mathbb{R}^s$ by $\boldsymbol{x} \cdot \boldsymbol{y}$. With this notation the function $\Phi(\boldsymbol{x}) = e^{i\boldsymbol{x}\cdot\boldsymbol{y}}$,

for $\boldsymbol{y} \in \mathbb{R}^s$ fixed, is positive definite on $\mathbb{R}^s$ since the quadratic form in Definition 3.2 becomes

$$\begin{aligned}\sum_{j=1}^{N}\sum_{k=1}^{N} c_j\overline{c_k}\Phi(\boldsymbol{x}_j-\boldsymbol{x}_k) &= \sum_{j=1}^{N}\sum_{k=1}^{N} c_j\overline{c_k}e^{i(\boldsymbol{x}_j-\boldsymbol{x}_k)\cdot\boldsymbol{y}}\\ &= \sum_{j=1}^{N} c_j e^{i\boldsymbol{x}_j\cdot\boldsymbol{y}}\sum_{k=1}^{N}\overline{c_k}e^{-i\boldsymbol{x}_k\cdot\boldsymbol{y}}\\ &= \left|\sum_{j=1}^{N} c_j e^{i\boldsymbol{x}_j\cdot\boldsymbol{y}}\right|^2 \geq 0.\end{aligned}$$

Definition 3.2 and the discussion preceding it suggest that we should use strictly positive definite functions as basis functions in (1.1), *i.e.*, $B_k(\boldsymbol{x}) = \Phi(\boldsymbol{x}-\boldsymbol{x}_k)$, or

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{k=1}^{N} c_k\Phi(\boldsymbol{x}-\boldsymbol{x}_k), \qquad \boldsymbol{x}\in\mathbb{R}^s. \tag{3.3}$$

Note that at this point we do not require Φ to be a radial function. In fact, the function $\mathcal{P}_f$ of (3.3) will yield an interpolant that is *translation invariant*, *i.e.*, the interpolant to translated data is the same as the translated interpolant to the original data. In order to obtain invariance also under rotations and reflections we will later specialize to strictly positive definite functions that are also radial on $\mathbb{R}^s$.

We will now discuss some of the most important properties and characterizations of (strictly) positive definite functions. For the sake of completeness we present a list of some basic properties of (strictly) positive definite functions and some examples.

Theorem 3.1. *Some basic properties of positive definite functions are*

(1) *Non-negative finite linear combinations of positive definite functions are positive definite. If $\Phi_1,\ldots,\Phi_n$ are positive definite on $\mathbb{R}^s$ and $c_j \geq 0$, $j=1,\ldots,n$, then*

$$\Phi(\boldsymbol{x}) = \sum_{j=1}^{n} c_j\Phi_j(\boldsymbol{x}), \qquad \boldsymbol{x}\in\mathbb{R}^s,$$

is also positive definite. Moreover, if at least one of the Φ_j is strictly positive definite and the corresponding $c_j > 0$, then Φ is strictly positive definite.

(2) $\Phi(\mathbf{0}) \geq 0$.

(3) $\Phi(-\boldsymbol{x}) = \overline{\Phi(\boldsymbol{x})}$.

(4) *Any positive definite function is bounded. In fact,*

$$|\Phi(\boldsymbol{x})| \leq \Phi(\mathbf{0}).$$

(5) *If Φ is positive definite with $\Phi(\mathbf{0}) = 0$ then $\Phi \equiv 0$.*

(6) *The product of (strictly) positive definite functions is (strictly) positive definite.*

Proof. Properties (1) and (2) follow immediately from Definition 3.2.

To show (3) we let $N = 2$, $\boldsymbol{x}_1 = \mathbf{0}$, $\boldsymbol{x}_2 = \boldsymbol{x}$, and choose $c_1 = 1$ and $c_2 = c$. Then the quadratic form in Definition 3.2 becomes

$$\sum_{j=1}^{2}\sum_{k=1}^{2} c_j\overline{c_k}\Phi(\boldsymbol{x}_j - \boldsymbol{x}_k) = (1 + |c|^2)\Phi(\mathbf{0}) + c\Phi(\boldsymbol{x}) + \overline{c}\Phi(-\boldsymbol{x}) \geq \mathbf{0}$$

for every $c \in \mathbb{C}$. Taking $c = 1$ and $c = i$ (where $i = \sqrt{-1}$), respectively, we can see that both $\Phi(\boldsymbol{x}) + \Phi(-\boldsymbol{x})$ and $i\,(\Phi(\boldsymbol{x}) - \Phi(-\boldsymbol{x}))$ must be real. This, however, is only possible if $\Phi(-\boldsymbol{x}) = \overline{\Phi(\boldsymbol{x})}$.

For the proof of (4) we let $N = 2$, $\boldsymbol{x}_1 = \mathbf{0}$, $\boldsymbol{x}_2 = \boldsymbol{x}$, and choose $c_1 = |\Phi(\boldsymbol{x})|$ and $c_2 = -\overline{\Phi(\boldsymbol{x})}$. Then the quadratic form in Definition 3.2 is

$$\sum_{j=1}^{2}\sum_{k=1}^{2} c_j\overline{c_k}\Phi(\boldsymbol{x}_j - \boldsymbol{x}_k) = 2\Phi(\mathbf{0})|\Phi(\boldsymbol{x})|^2 - \Phi(-\boldsymbol{x})\Phi(\boldsymbol{x})|\Phi(\boldsymbol{x})| - \Phi^2(\boldsymbol{x})|\Phi(\boldsymbol{x})| \geq \mathbf{0}.$$

Since $\Phi(-\boldsymbol{x}) = \overline{\Phi(\boldsymbol{x})}$ by Property (3), this gives

$$2\Phi(\mathbf{0})|\Phi(\boldsymbol{x})|^2 - 2|\Phi(\boldsymbol{x})|^3 \geq \mathbf{0}.$$

If $|\Phi(\boldsymbol{x})| > 0$, we divide by $|\Phi(\boldsymbol{x})|^2$ and the statement follows immediately. In case $|\Phi(\boldsymbol{x})| \equiv 0$ the statement holds trivially.

Property (5) follows immediately from (4), and Property (6) is a consequence of a theorem by Schur in the field of linear algebra which states that the elementwise (or Hadamard) product of positive (semi-)definite matrices is positive (semi-)definite. For more details we refer the reader to [Cheney and Light (1999)] or [Wendland (2005a)]. □

Example 3.2. The cosine function is positive definite on $\mathbb{R}$ since, for $x \in \mathbb{R}$, we have $\cos x = \frac{1}{2}\left(e^{ix} + e^{-ix}\right)$. Now Property (1) and Example 3.1 can be invoked.

Property (3) shows that any real-valued (strictly) positive definite function has to be even. However, it is also possible to characterize real-valued (strictly) positive definite functions using only *real* coefficients (see [Wendland (2005a)] for details), *i.e.*,

Theorem 3.2. *A real-valued continuous function Φ is positive definite on $\mathbb{R}^s$ if and only if it is even and*

$$\sum_{j=1}^{N}\sum_{k=1}^{N} c_j c_k \Phi(\boldsymbol{x}_j - \boldsymbol{x}_k) \geq \mathbf{0} \tag{3.4}$$

for any N pairwise different points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^s$, and $\boldsymbol{c} = [c_1, \ldots, c_N]^T \in \mathbb{R}^N$.

The function Φ is strictly positive definite on $\mathbb{R}^s$ if the quadratic form (3.4) is zero only for $\boldsymbol{c} \equiv \mathbf{0}$.

3.2 Integral Characterizations for (Strictly) Positive Definite Functions

We will now summarize some facts about integral characterizations of positive definite functions. They were established in the 1930s by Bochner and Schoenberg. However, we will also mention the more recent extensions to strictly positive definite and strictly completely/multiply monotone functions that are essential to the application of the theory to the scattered data interpolation problem. A much more detailed discussion of this material is presented in the recent book [Wendland (2005a)]. Some frequently used integral transforms are listed in Appendix B. Integral characterizations of the closely related completely and multiply monotone functions are presented in Chapter 5.

3.2.1 *Bochner's Theorem*

One of the most celebrated results on positive definite functions is their characterization in terms of Fourier transforms established by Bochner in 1932 (for $s = 1$) and 1933 (for general s).

Theorem 3.3 (Bochner). *A (complex-valued) function $\Phi \in C(\mathbb{R}^s)$ is positive definite on $\mathbb{R}^s$ if and only if it is the Fourier transform of a finite non-negative Borel measure μ on $\mathbb{R}^s$, i.e.*

$$\Phi(\boldsymbol{x}) = \hat{\mu}(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^s}} \int_{\mathbb{R}^s} e^{-i\boldsymbol{x}\cdot\boldsymbol{y}} d\mu(\boldsymbol{y}), \qquad \boldsymbol{x} \in \mathbb{R}^s.$$

Proof. There are many proofs of this theorem. Bochner's original proof can be found in [Bochner (1933)]. Other proofs can be found, *e.g.*, in the books [Cuppens (1975)] or [Gel'fand and Vilenkin (1964)]. A proof using the Riesz representation theorem to interpret the Borel measure as a distribution, and then taking advantage of distributional Fourier transforms can be found in the book [Wendland (2005a)].

We will prove only the one (easy) direction. It is this part of the statement that is important for the application to scattered data interpolation. We assume Φ is the Fourier transform of a finite non-negative Borel measure and show Φ is positive definite. Thus,

$$\begin{aligned}
\sum_{j=1}^{N}\sum_{k=1}^{N} c_j\overline{c_k}\Phi(\boldsymbol{x}_j - \boldsymbol{x}_k) &= \frac{1}{\sqrt{(2\pi)^s}} \sum_{j=1}^{N}\sum_{k=1}^{N} \left[c_j\overline{c_k} \int_{\mathbb{R}^s} e^{-i(\boldsymbol{x}_j - \boldsymbol{x}_k)\cdot\boldsymbol{y}} d\mu(\boldsymbol{y}) \right] \\
&= \frac{1}{\sqrt{(2\pi)^s}} \int_{\mathbb{R}^s} \left[\sum_{j=1}^{N} c_j e^{-i\boldsymbol{x}_j\cdot\boldsymbol{y}} \sum_{k=1}^{N} \overline{c_k} e^{i\boldsymbol{x}_k\cdot\boldsymbol{y}} \right] d\mu(\boldsymbol{y}) \\
&= \frac{1}{\sqrt{(2\pi)^s}} \int_{\mathbb{R}^s} \left| \sum_{j=1}^{N} c_j e^{-i\boldsymbol{x}_j\cdot\boldsymbol{y}} \right|^2 d\mu(\boldsymbol{y}) \geq 0.
\end{aligned}$$

The last inequality holds because of the conditions imposed on the measure μ. □

Remark 3.1. We can see from Theorem 3.3 that the function $\Phi(\boldsymbol{x}) = e^{i\boldsymbol{x}\cdot\boldsymbol{y}}$ of Example **3**.1 can be considered as the *fundamental positive definite function* since all other positive definite functions are obtained as (infinite) linear combinations of this function. While Property (1) of Theorem 3.1 implies that linear combinations of Φ will again be positive definite, the remarkable content of Bochner's Theorem is the fact that indeed *all* positive definite functions are generated by Φ.

3.2.2 *Extensions to Strictly Positive Definite Functions*

In order to accomplish our goal of guaranteeing a well-posed interpolation problem we have to extend (if possible) Bochner's characterization to *strictly* positive definite functions.

We begin with a sufficient condition for a function to be strictly positive definite on $\mathbb{R}^s$.

For this result we require the notion of the *carrier* of a (non-negative) Borel measure defined on some topological space X (see also Appendix B). This set is given by

$$X \setminus \bigcup \{O : \ O \text{ is open and } \mu(O) = 0\}.$$

Theorem 3.4. *Let μ be a non-negative finite Borel measure on $\mathbb{R}^s$ whose carrier is a set of nonzero Lebesgue measure. Then the Fourier transform of μ is strictly positive definite on $\mathbb{R}^s$.*

Proof. As in the proof of Bochner's theorem we have

$$\begin{aligned}
\sum_{j=1}^{N}\sum_{k=1}^{N} c_j\overline{c_k}\hat{\mu}(\boldsymbol{x}_j - \boldsymbol{x}_k) &= \frac{1}{\sqrt{(2\pi)^s}}\sum_{j=1}^{N}\sum_{k=1}^{N} c_j\overline{c_k}\left[\int_{\mathbb{R}^s} e^{-i(\boldsymbol{x}_j-\boldsymbol{x}_k)\cdot\boldsymbol{y}} d\mu(\boldsymbol{y})\right] \\
&= \frac{1}{\sqrt{(2\pi)^s}}\int_{\mathbb{R}^s}\left[\sum_{j=1}^{N} c_j e^{-i\boldsymbol{x}_j\cdot\boldsymbol{y}}\sum_{k=1}^{N}\overline{c_k}e^{i\boldsymbol{x}_k\cdot\boldsymbol{y}}\right] d\mu(\boldsymbol{y}) \\
&= \frac{1}{\sqrt{(2\pi)^s}}\int_{\mathbb{R}^s}\left|\sum_{j=1}^{N} c_j e^{-i\boldsymbol{x}_j\cdot\boldsymbol{y}}\right|^2 d\mu(\boldsymbol{y}) \geq 0.
\end{aligned}$$

Now let

$$g(\boldsymbol{y}) = \sum_{j=1}^{N} c_j e^{-i\boldsymbol{x}_j\cdot\boldsymbol{y}},$$

and assume that the points $\boldsymbol{x}_j$ are all distinct and $\boldsymbol{c} \neq \boldsymbol{0}$. In this case the functions $\boldsymbol{y} \mapsto e^{-i\boldsymbol{x}_j\cdot\boldsymbol{y}}$ are linearly independent so that $g \neq 0$. Since g is an entire function its zero set, *i.e.*, $\{\boldsymbol{y} \in \mathbb{R}^s : \ g(\boldsymbol{y}) = 0\}$ can have no accumulation point and therefore it has Lebesgue measure zero (see, *e.g.*, [Cheney and Light (1999)]). Now, the only

remaining way to make the above inequality an equality is if the carrier of μ is contained in the zero set of g, *i.e.*, has Lebesgue measure zero. This, however, is ruled out in the hypothesis of the theorem. □

Work toward an analog of Bochner's theorem, *i.e.*, a complete integral characterization of functions that are strictly positive definite on $\mathbb{R}^s$, is given in [Chang (1996)] for the case $s = 1$.

The following corollary gives us a way to *construct* strictly positive definite functions.

Corollary 3.1. *Let f be a continuous non-negative function in $L_1(\mathbb{R}^s)$ which is not identically zero. Then the Fourier transform of f is strictly positive definite on $\mathbb{R}^s$.*

Proof. This is a special case of the previous theorem in which the measure μ has Lebesgue density f. Thus, we use the measure μ defined for any Borel set B by

$$\mu(B) = \int_B f(\boldsymbol{x}) d\boldsymbol{x}.$$

Then the carrier of μ is equal to the (closed) support of f. However, since f is non-negative and not identically equal to zero, its support has positive Lebesgue measure, and hence the Fourier transform of f is strictly positive definite by the preceding theorem. □

Finally, a criterion to check whether a given function is strictly positive definite is given in [Wendland (2005a)].

Theorem 3.5. *Let Φ be a continuous function in $L_1(\mathbb{R}^s)$. Φ is strictly positive definite if and only if Φ is bounded and its Fourier transform is non-negative and not identically equal to zero.*

Theorem 3.5 is of fundamental importance and we will come back to this theorem several times later on. In fact, the proof of Theorem 3.5 in [Wendland (2005a)] shows that — if $\Phi \not\equiv 0$ (which implies that then also $\hat{\Phi} \not\equiv 0$) — we need to ensure only that $\hat{\Phi}$ be non-negative in order for Φ to be strictly positive definite.

3.3 Positive Definite Radial Functions

We now turn our attention to positive definite *radial* functions. Recall that Definition 3.2 characterizes (strictly) positive definite functions in terms of multivariate functions Φ. However, when we are dealing with radial functions, *i.e.*, $\Phi(\boldsymbol{x}) = \varphi(\|\boldsymbol{x}\|)$, then it will be convenient to also refer to the univariate function φ as a *positive definite radial function.* While this does present a slight abuse of our terminology for positive definite functions this is what is commonly done in the literature.

An immediate consequence of this notational convention is

Lemma 3.1. *If $\Phi = \varphi(\|\cdot\|)$ is (strictly) positive definite and radial on $\mathbb{R}^s$ then Φ is also (strictly) positive definite and radial on $\mathbb{R}^\sigma$ for any $\sigma \leq s$.*

We now return to integral characterizations and begin with a theorem due to Schoenberg (see, *e.g.*, [Schoenberg (1938a)], p.816, or [Wells and Williams (1975)], p.27).

Theorem 3.6. *A continuous function $\varphi : [0,\infty) \to \mathbb{R}$ is positive definite and radial on $\mathbb{R}^s$ if and only if it is the Bessel transform of a finite non-negative Borel measure μ on $[0,\infty)$, i.e.*

$$\varphi(r) = \int_0^\infty \Omega_s(rt) d\mu(t).$$

Here

$$\Omega_s(r) = \begin{cases} \cos r & \text{for } s = 1, \\ \Gamma\left(\frac{s}{2}\right)\left(\frac{2}{r}\right)^{(s-2)/2} J_{(s-2)/2}(r) & \text{for } s \geq 2, \end{cases}$$

and $J_{(s-2)/2}$ is the classical Bessel function of the first kind of order $(s-2)/2$.

As above, now the function $\Phi(x) = \cos(x)$ from Example 3.2 can be viewed as the fundamental positive definite radial function on $\mathbb{R}$. We will see below (in Example 3 of Chapter 4) that the characterization of Theorem 3.6 immediately suggests a class of (even strictly) positive definite radial functions. As for the basic 1D example, the measure μ will simply be a point evaluation measure.

A Fourier transform characterization of strictly positive definite radial functions on $\mathbb{R}^s$ can be found in [Wendland (2005a)]. It is essentially a combination of Theorem 3.5 and the formula in Theorem B.1 of Appendix B for the Fourier transform of a radial function:

Theorem 3.7. *A continuous function $\varphi : [0,\infty) \to \mathbb{R}$ such that $r \mapsto r^{s-1}\varphi(r) \in L_1[0,\infty)$ is strictly positive definite and radial on $\mathbb{R}^s$ if and only if the s-dimensional Fourier transform*

$$\mathcal{F}_s\varphi(r) = \frac{1}{\sqrt{r^{s-2}}} \int_0^\infty \varphi(t) t^{\frac{s}{2}} J_{(s-2)/2}(rt) dt$$

is non-negative and not identically equal to zero.

Since Lemma 3.1 states that any function that is (strictly) positive definite and radial on $\mathbb{R}^s$ is also (strictly) positive definite and radial on $\mathbb{R}^\sigma$ for any $\sigma \leq s$, those functions which are (strictly) positive definite and radial on $\mathbb{R}^s$ *for all* s are of particular interest. The class of functions that are positive definite on $\mathbb{R}^s$ for all s was also characterized by Schoenberg ([Schoenberg (1938a)], pp. 817–821). An extension to the strictly positive definite case can be found in [Micchelli (1986)]:

Theorem 3.8 (Schoenberg). *A continuous function $\varphi : [0, \infty) \to \mathbb{R}$ is strictly positive definite and radial on $\mathbb{R}^s$ for all s if and only if it is of the form*

$$\varphi(r) = \int_0^\infty e^{-r^2 t^2} d\mu(t),$$

where μ is a finite non-negative Borel measure on $[0, \infty)$ not concentrated at the origin.

As suggested for Theorem 3.6 above, letting μ be a point evaluation measure in Theorem 3.8 we obtain that the Gaussian is strictly positive definite and radial on $\mathbb{R}^s$ for all s (*c.f* Example 1 of Chapter 4).

The Schoenberg characterization of (strictly) positive definite radial functions on $\mathbb{R}^s$ for all s (Theorem 3.8) implies that we have a finite non-negative Borel measure μ on $[0, \infty)$ such that

$$\varphi(r) = \int_0^\infty e^{-r^2 t^2} d\mu(t).$$

If we want to find a zero r_0 of φ then we have to solve

$$\varphi(r_0) = \int_0^\infty e^{-r_0^2 t^2} d\mu(t) = 0.$$

Since the exponential function is positive and the measure is non-negative, it follows that μ must be the zero measure. However, then φ is identically equal to zero. Therefore, a non-trivial function φ that is positive definite and radial on $\mathbb{R}^s$ for all s can have no zeros. This implies in particular that

Theorem 3.9. *There are no oscillatory univariate continuous functions that are strictly positive definite and radial on $\mathbb{R}^s$ for all s. Moreover, there are no compactly supported univariate continuous functions that are strictly positive definite and radial on $\mathbb{R}^s$ for all s.*

An equivalent argument for the oscillatory case is given in Theorem 2.3 of [Fornberg *et al.* (2004)].

Chapter 4

Examples of Strictly Positive Definite Radial Functions

We now present a number of functions that are covered by the theory presented thus far. While it is possible to include a *shape parameter* ε for all of the functions presented in the examples below by rescaling $\boldsymbol{x}$ to $\varepsilon\boldsymbol{x}$, we avoid its use in the formulation of all but the Gaussian example to keep the formulas as simple as possible. We do, however, use a shape parameter when plotting some of the basis functions.

Our use of the shape parameter does not always match its "traditional" use. For example, Hardy introduced his inverse multiquadrics (see Example 5 below) in the form $\Phi(\|\boldsymbol{x}\|) = 1/\sqrt{c^2 + \|\boldsymbol{x}\|^2}$ with shape parameter c. It is, of course, straightforward to transform this representation to the one suggested above, *i.e.*, $\Phi(\|\boldsymbol{x}\|) = 1/\sqrt{1 + \varepsilon^2\|\boldsymbol{x}\|^2}$, by setting $c^2 = 1/\varepsilon^2$ and scaling the result by $1/|\varepsilon|$.

Our use of the shape parameter as a factor applied directly to $\boldsymbol{x}$ has the advantage of providing a unified treatment in which a decrease of the shape parameter always has the effect of producing "flat" basis functions, while increasing ε leads to more peaked (or localized) basis functions.

4.1 Example 1: Gaussians

We can now show that the *Gaussian*

$$\Phi(\boldsymbol{x}) = e^{-\varepsilon^2\|\boldsymbol{x}\|^2}, \quad \varepsilon > 0, \tag{4.1}$$

is strictly positive definite (and radial) on $\mathbb{R}^s$ for any s. This is due to the fact that the Fourier transform of a Gaussian is essentially a Gaussian. In fact,

$$\hat{\Phi}(\boldsymbol{\omega}) = \frac{1}{(\sqrt{2}\varepsilon)^s} e^{-\frac{\|\boldsymbol{\omega}\|^2}{4\varepsilon^2}},$$

and this is positive independent of the space dimension s. In particular, for $\varepsilon = \frac{1}{\sqrt{2}}$ we have $\hat{\Phi} = \Phi$. Plots of Gaussian RBFs were presented in Fig. 2.1. Clearly, the Gaussians are infinitely differentiable. Some of its derivatives (as well as those of many other RBFs) are collected in Appendix D.

Another argument to show that Gaussians are strictly positive definite and radial on $\mathbb{R}^s$ for any s that avoids dealing with Fourier transforms will become available later. It will make use of completely monotone functions.

Recall that Property (1) of Theorem 3.1 shows that any finite non-negative linear combination of (strictly) positive definite functions is again (strictly) positive definite. Moreover, we just saw that Gaussians are strictly positive definite and radial on all $\mathbb{R}^s$. Now, the Schoenberg characterization of functions that are (strictly) positive definite and radial on any $\mathbb{R}^s$, Theorem 3.8, states that *all* such functions are given as infinite linear combinations of Gaussians. Therefore, the Gaussians can be viewed as the fundamental member of the family of functions that are strictly positive definite and radial on $\mathbb{R}^s$ for all s.

Since Gaussians play a central role in statistics this is a good place to mention that positive definite functions are — up to a normalization $\Phi(0) = 1$ — identical with characteristic functions of distribution functions in statistics.

4.2 Example 2: Laguerre-Gaussians

In order to obtain a generalization of Gaussians we start with the *generalized Laguerre polynomials* $L_n^{s/2}$ of degree n and order $s/2$ defined by their Rodrigues formula (see, *e.g.*, formula (6.2.1) in [Andrews *et al.* (1999)])

$$L_n^{s/2}(t) = \frac{e^t t^{-s/2}}{n!} \frac{d^n}{dt^n} \left(e^{-t} t^{n+s/2} \right), \quad n = 1, 2, 3, \ldots.$$

An explicit formula for the generalized Laguerre polynomials is

$$L_n^{s/2}(t) = \sum_{k=0}^{n} \frac{(-1)^k}{k!} \binom{n + s/2}{n - k} t^k.$$

We then define the *Laguerre-Gaussians*

$$\Phi(\boldsymbol{x}) = e^{-\|\boldsymbol{x}\|^2} L_n^{s/2}(\|\boldsymbol{x}\|^2), \tag{4.2}$$

and list their Fourier transforms as

$$\hat{\Phi}(\boldsymbol{\omega}) = \frac{e^{-\frac{\|\boldsymbol{\omega}\|^2}{4}}}{\sqrt{2^s}} \sum_{j=0}^{n} \frac{\|\boldsymbol{\omega}\|^{2j}}{j! 4^j} \geq 0. \tag{4.3}$$

Note that the definition of the Laguerre-Gaussians depends on the space dimension s. Therefore they are strictly positive definite and radial on $\mathbb{R}^s$ (and by Lemma 3.1 also on $\mathbb{R}^\sigma$ for any $\sigma \leq s$).

Laguerre-Gaussian functions for some special choices of s and n are listed in Table 4.1. Figure 4.1 shows a Laguerre-Gaussian for $s = 1, n = 2$, and for $s = 2, n = 2$ displayed with a shape parameter $\varepsilon = 3$ and scaled so that $\Phi(\mathbf{0}) = 1$. Moreover, the Laguerre-Gaussians are infinitely smooth for all choices of n and s.

Note that the Laguerre-Gaussians (while being strictly positive definite functions) are not positive. Since the Laguerre-Gaussians are oscillatory functions we

know from Theorem 3.9 that they cannot be strictly positive definite and radial on $\mathbb{R}^s$ for all s. We will encounter these functions later in the context of approximate moving least squares approximation (*c.f.* Chapter 26).

Table 4.1 Laguerre-Gaussians for various choices of s and n.

s	$n=1$	$n=2$
1	$\left(\frac{3}{2} - \lvert x\rvert^2\right) e^{-\lvert x\rvert^2}$	$\left(\frac{15}{8} - \frac{5}{2}\lvert x\rvert^2 + \frac{1}{2}\lvert x\rvert^4\right) e^{-\lvert x\rvert^2}$
2	$(2 - \lVert \boldsymbol{x}\rVert^2) e^{-\lVert \boldsymbol{x}\rVert^2}$	$\left(3 - 3\lVert \boldsymbol{x}\rVert^2 + \frac{1}{2}\lVert \boldsymbol{x}\rVert^4\right) e^{-\lVert \boldsymbol{x}\rVert^2}$
3	$\left(\frac{5}{2} - \lVert \boldsymbol{x}\rVert^2\right) e^{-\lVert \boldsymbol{x}\rVert^2}$	$\left(\frac{35}{8} - \frac{7}{2}\lVert \boldsymbol{x}\rVert^2 + \frac{1}{2}\lVert \boldsymbol{x}\rVert^4\right) e^{-\lVert \boldsymbol{x}\rVert^2}$

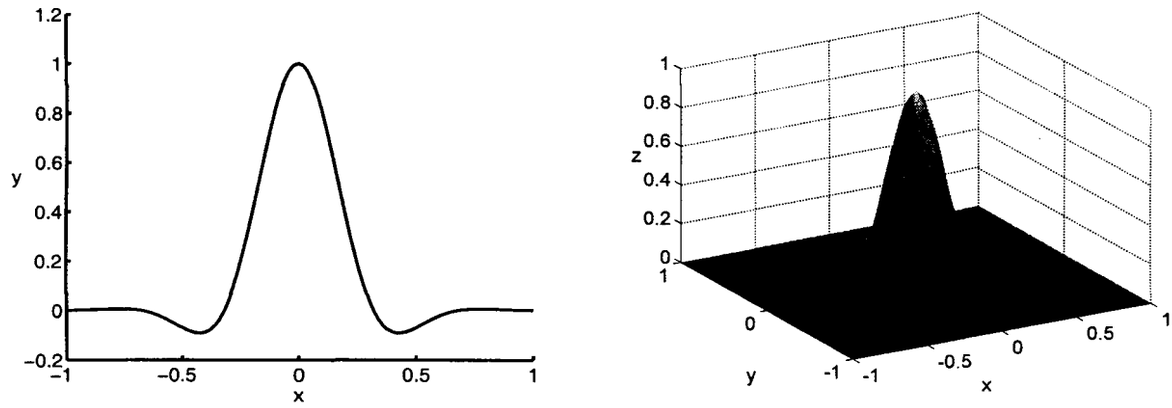


Fig. 4.1 Laguerre-Gaussians with $s = 1, n = 2$ (left) and $s = 2, n = 2$ (right) centered at the origin.

4.3 Example 3: Poisson Radial Functions

Another class of oscillatory functions that are strictly positive definite and radial on $\mathbb{R}^s$ (and all $\mathbb{R}^\sigma$ for $\sigma \leq s$) were recently studied by Fornberg and co-workers (see [Fornberg *et al.* (2004)] and also [Flyer (2006)]). These functions are of the form

$$\Phi(\boldsymbol{x}) = \frac{J_{s/2-1}(\lVert \boldsymbol{x}\rVert)}{\lVert \boldsymbol{x}\rVert^{s/2-1}}, \quad s \geq 2, \tag{4.4}$$

where J_ν is the Bessel function of the first kind of order ν. While these functions are not defined at the origin they can be extended to be infinitely differentiable in all of $\mathbb{R}^s$.

The functions (4.4) were already studied by Schoenberg (see the discussion surrounding Theorem 3.6) who suggested calling them *Poisson functions.* In fact, the functions in (4.4) are (up to the scale factor $2^{(s-2)/2}\Gamma(s/2)$) the functions Ω_s of

Theorem 3.6 and therefore can be viewed as the fundamental member of the family of functions that are strictly positive definite and radial on $\mathbb{R}^s$ for fixed s.

Schoenberg showed that the functions Ω_s are given by

$$\Omega_s(\boldsymbol{x}) = \frac{1}{\omega_{s-1}} \int_{S_{s-1}} e^{i\boldsymbol{x}\cdot\sigma} d\sigma,$$

where ω_{s-1} denotes the area of the unit sphere S^{s-1} in $\mathbb{R}^s$, and $d\sigma$ denotes the usual measure on S^{s-1}.

The Poisson functions are another generalization of Gaussians (the fundamental strictly positive definite radial function on $\mathbb{R}^s$ for all s) since the following limiting relation due to John von Neumann holds (see the discussion in [Schoenberg (1938a)]):

$$\lim_{s\to\infty} \Omega_s(r\sqrt{2s}) = e^{-r^2}.$$

Since the Poisson radial functions are defined in terms of Bessel functions they are also *band-limited*, *i.e.*, their Fourier transform has compact support. In fact, the Fourier transform of Φ in $\mathbb{R}^\sigma$, $\sigma \leq s$, is given by (see [Flyer (2006)])

$$\hat{\Phi}(\boldsymbol{\omega}) = \frac{1}{2^{\sigma-1}\Gamma(\frac{s-\sigma}{2})\pi^\sigma}(1 - \|\boldsymbol{\omega}\|^2)^{(s-\sigma-2)/2}, \qquad -1 < \omega_1, \ldots, \omega_s < 1.$$

Some of these Poisson functions are listed in Table 4.2 and displayed in Figure 4.2 (where a shape parameter $\varepsilon = 10$ was used for the plots).

Table 4.2 Poisson functions for various choices of s.

$s = 2$	$s = 3$	$s = 4$	$s = 5$
$J_0(\|\boldsymbol{x}\|)$	$\sqrt{\frac{2}{\pi}}\frac{\sin(\|\boldsymbol{x}\|)}{\|\boldsymbol{x}\|}$	$\frac{J_1(\|\boldsymbol{x}\|)}{\|\boldsymbol{x}\|}$	$\sqrt{\frac{2}{\pi}}\frac{\sin(\|\boldsymbol{x}\|) - \|\boldsymbol{x}\|\cos(\|\boldsymbol{x}\|)}{\|\boldsymbol{x}\|^3}$

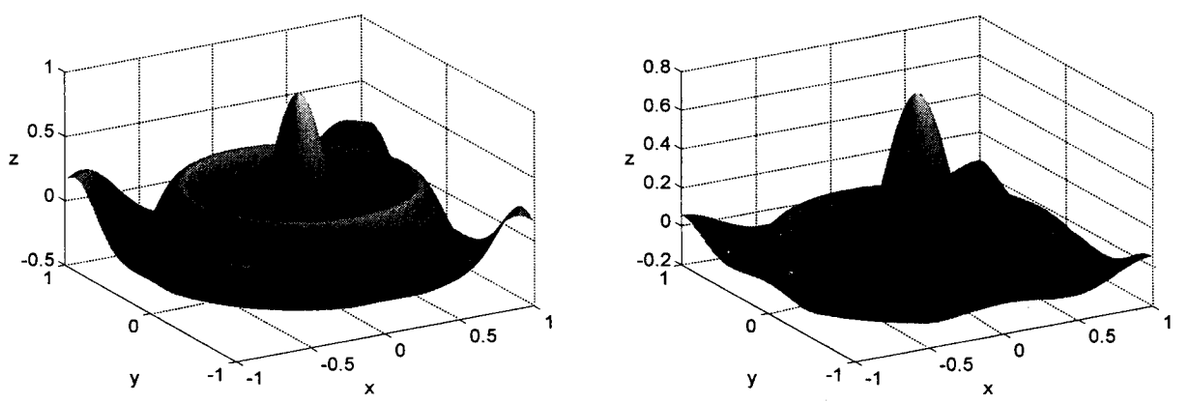


Fig. 4.2 Poisson functions with $s = 2$ (left) and $s = 3$ (right) centered at the origin in $\mathbb{R}^2$.

4.4 Example 4: Matérn Functions

A fourth example of strictly positive definite functions is given by the class of *Matérn functions* which are quite common in the statistics literature (see, *e.g.*, [Matérn (1986)] or [Stein (1999)])

$$\Phi(\boldsymbol{x}) = \frac{K_{\beta-\frac{s}{2}}(\|\boldsymbol{x}\|)\|\boldsymbol{x}\|^{\beta-\frac{s}{2}}}{2^{\beta-1}\Gamma(\beta)}, \qquad \beta > \frac{s}{2}. \tag{4.5}$$

Here K_ν is the *modified Bessel function of the second kind* (sometimes also called modified Bessel function of the third kind, or MacDonald's function) *of order* ν. The Fourier transform of the Matérn functions is given by the *Bessel kernels*

$$\hat{\Phi}(\boldsymbol{\omega}) = \left(1 + \|\boldsymbol{\omega}\|^2\right)^{-\beta} > 0.$$

Therefore the Matérn functions are strictly positive definite on $\mathbb{R}^s$ for all $s < 2\beta$. Schaback calls these functions *Sobolev splines* (see, *e.g.*, [Schaback (1995a)] or his earlier discussion in [Schaback (1993)]) since they are naturally related to Sobolev spaces (see Chapter 13). These functions are also discussed in the relatively early paper [Dix and Ogden (1994)].

Some simple representatives of the family of Matérn functions are listed (up to a dimension-dependent scale factor) in Table 4.3. Note that the scaled functions listed in Table 4.3 do not depend on s. Since the modified Bessel functions are positive, so are the Matérn functions. Two examples are displayed in Figure 4.3. The function on the left is displayed using a shape parameter $\varepsilon = 3$. The plot on the right is scaled so that the value at the origin equals one and uses a shape parameter $\varepsilon = 10$. Note that the function on the left (corresponding to $\beta = \frac{s+1}{2}$) is not differentiable at the origin. The Matérn function for $\beta = \frac{s+3}{2}$ is C^2 smooth, and that for $\beta = \frac{s+5}{2}$ is in $C^4(\mathbb{R}^s)$.

Table 4.3 Matérn functions for various choices of β.

$\beta = \frac{s+1}{2}$	$\beta = \frac{s+3}{2}$	$\beta = \frac{s+5}{2}$
$e^{-\|\boldsymbol{x}\|}$	$(1 + \|\boldsymbol{x}\|)e^{-\|\boldsymbol{x}\|}$	$(3 + 3\|\boldsymbol{x}\| + \|\boldsymbol{x}\|^2)e^{-\|\boldsymbol{x}\|}$

4.5 Example 5: Generalized Inverse Multiquadrics

Since both Φ and $\hat{\Phi}$ in the previous example are positive radial functions we can use the Hankel inversion theorem (see Appendix B) to reverse their roles and see that the so-called *generalized inverse multiquadrics*

$$\Phi(\boldsymbol{x}) = \left(1 + \|\boldsymbol{x}\|^2\right)^{-\beta}, \qquad \beta > \frac{s}{2}, \tag{4.6}$$

are strictly positive definite on $\mathbb{R}^s$ for $s < 2\beta$. Generalized inverse multiquadrics are infinitely differentiable. By using another argument based on completely monotone

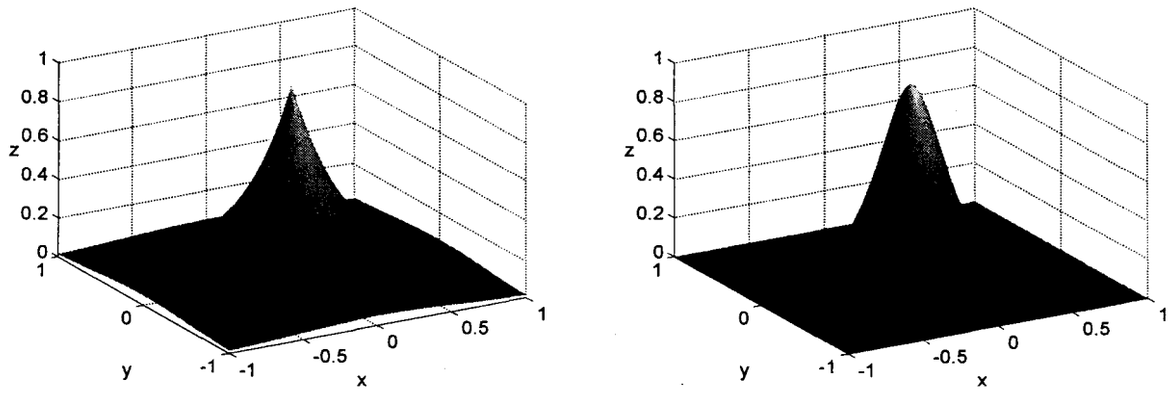


Fig. 4.3 Matérn functions with $\beta = \frac{s+1}{2}$ (left) and $\beta = \frac{s+5}{2}$ (right) centered at the origin in $\mathbb{R}^2$.

functions we will be able to show that in fact we need to require only $\beta > 0$, and therefore the generalized inverse multiquadrics are strictly positive definite on $\mathbb{R}^s$ for any s.

The "original" inverse multiquadric was introduced by Hardy in the early 1970s and corresponds to the value $\beta = 1/2$. The special choice $\beta = 1$ was referred to as *inverse quadratic* in various papers of Fornberg and co-workers (see, *e.g.*, [Fornberg and Wright (2004)]). These two functions are displayed in Figure 4.4 using a shape parameter $\varepsilon = 5$.

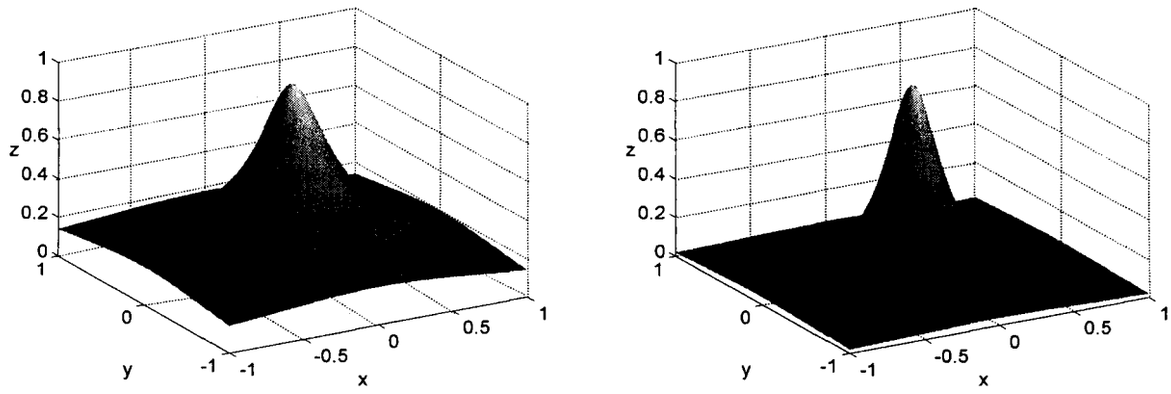


Fig. 4.4 Inverse multiquadric ($\beta = \frac{1}{2}$, left) and inverse quadratic ($\beta = 1$, right) centered at the origin in $\mathbb{R}^2$.

4.6 Example 6: Truncated Power Functions

We now present an example of a family of strictly positive definite functions with *compact support*. Note that due to the observation made in Theorem 3.9 at the end of the previous chapter, they can not be strictly positive definite on $\mathbb{R}^s$ for all s.

The *truncated power functions*

$$\varphi_\ell(r) = (1-r)_+^\ell \tag{4.7}$$

give rise to strictly positive definite and radial functions on $\mathbb{R}^s$ provided ℓ satisfies $\ell \geq \lfloor \frac{s}{2} \rfloor + 1$. Finding the Fourier transform of the truncated power function is rather involved. For details we refer to [Wendland (2005a)]. We will later use a simpler test based on multiply monotone functions to establish the strict positive definiteness of the truncated power functions. In (4.7) we used the *cutoff function* $(\cdot)_+$ which is defined by

$$(x)_+ = \begin{cases} x, & \text{for } x \geq 0, \\ 0, & \text{for } x < 0. \end{cases}$$

The cutoff function can be implemented conveniently in MATLAB using the `max` function, *i.e.*, if `fx` is a vector of function values of f for different choices of x, then `max(fx,0)` computes $(f(x))_+$. We also point out that the expressions of the form $(1-r)_+^\ell$ are to be interpreted as $((1-r)_+)^\ell$, *i.e.*, we first apply the cutoff function, and then the power.

Two different truncated power functions (with $\ell = 2, 4$) are displayed in Figure 4.5. While none of the truncated power functions are differentiable at the origin, the smoothness at the boundary of the support increases with ℓ.

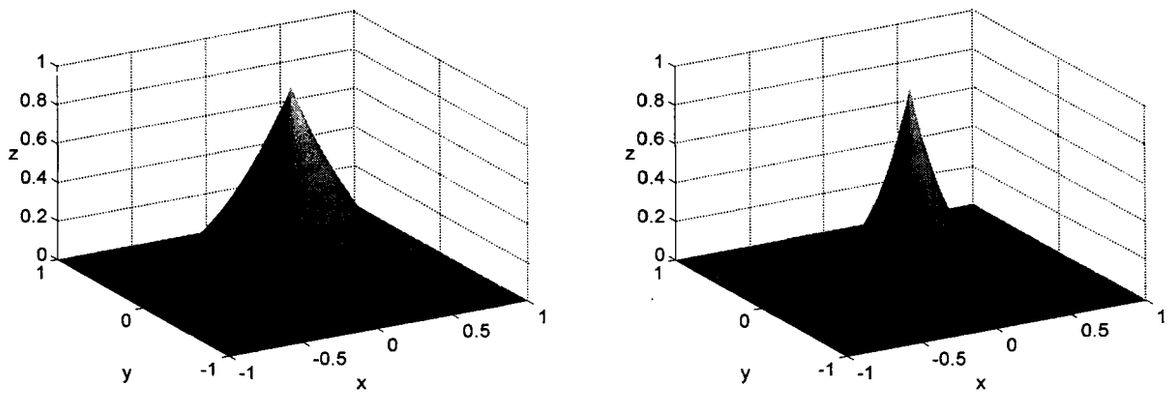


Fig. 4.5 Truncated power function with $\ell = 2$ (left) and $\ell = 4$ (right) centered at the origin in $\mathbb{R}^2$.

4.7 Example 7: Potentials and Whittaker Radial Functions

Let $f \in C[0, \infty)$ be non-negative and not identically equal to zero, and define the function φ by

$$\varphi(r) = \int_0^\infty (1-rt)_+^{k-1} f(t) dt. \tag{4.8}$$

Then $\Phi = \varphi(\|\cdot\|)$ is strictly positive definite and radial on $\mathbb{R}^s$ provided $k \geq \lfloor \frac{s}{2} \rfloor + 2$ (see also Theorem 5.5 below). This can be verified by considering the quadratic form

$$\sum_{j=1}^{N}\sum_{k=1}^{N} c_j c_k \varphi(\|\boldsymbol{x}_j - \boldsymbol{x}_k\|) = \int_0^\infty \sum_{j=1}^{N}\sum_{k=1}^{N} c_j c_k \varphi_{k-1}(t\|\boldsymbol{x}_j - \boldsymbol{x}_k\|) f(t) dt$$

which is non-negative since the truncated power function $\varphi_{k-1}(\|\cdot\|)$ is strictly positive definite by Example 6, and f is non-negative. Since f is also assumed to be not identically equal to zero, the only way for the quadratic form to equal zero is if $\boldsymbol{c} = 0$, and therefore φ is strictly positive definite.

For example, if we take $f(t) = t^\beta$, $\beta \geq 0$, then we get

$$\Phi(\boldsymbol{x}) = \frac{\Gamma(k)\Gamma(\beta+1)}{\Gamma(k+\beta+1)\|\boldsymbol{x}\|^{\beta+1}}. \tag{4.9}$$

While these functions are strictly positive definite and radial they are also singular at the origin and therefore not useful for our purposes. However, these functions are — up to scaling — generalizations of the *Coulomb potential* (for $\beta = 0$), and can therefore be given a physical interpretation.

Another possibility is to take $f(t) = t^\alpha e^{-\beta t}$, $\alpha \geq 0, \beta > 0$. Then we get

$$\Phi(\boldsymbol{x}) = \frac{\|\boldsymbol{x}\|^{(k-\alpha)/2}\Gamma(1+\alpha)\Gamma(k)}{\beta^{1+(k+\alpha)/2}\Gamma(k+\alpha+2)} e^{-\frac{\beta}{2\|\boldsymbol{x}\|}} \times \tag{4.10}$$
$$\left(kM_{(\alpha-k)/2,(k+\alpha+1)/2}\left(\frac{\beta}{\|\boldsymbol{x}\|}\right) + (1+\alpha)M_{1-(k-\alpha)/2,(k+\alpha+1)/2}\left(\frac{\beta}{\|\boldsymbol{x}\|}\right)\right).$$

Here $M_{\mu,\nu}$ is the Whittaker-M function, a confluent hypergeometric function (see, *e.g.*, Chapter 13 of [Abramowitz and Stegun (1972)]). When ν is a half-integer (which is, *e.g.*, the case for integer k and α) formula (4.10) simplifies significantly. Examples for various integer values of k and α are listed in Table 4.4. Note that these functions are not defined at the origin. However, they can be made (only) continuous at the origin. Plots of two of these functions are provided in Figure 4.6. Note that only the functions for $k \geq 3$ are guaranteed to be strictly positive definite and radial on $\mathbb{R}^3$.

Table 4.4 Whittaker radial functions Φ for various choices of k and α.

α	$k = 2$	$k = 3$
0	$\frac{\beta - \|\boldsymbol{x}\| + \|\boldsymbol{x}\|e^{-\frac{\beta}{\|\boldsymbol{x}\|}}}{\beta^2}$	$\frac{\beta^2 - 2\beta\|\boldsymbol{x}\| + 2\|\boldsymbol{x}\|^2 - 2\|\boldsymbol{x}\|^2 e^{-\frac{\beta}{\|\boldsymbol{x}\|}}}{\beta^3}$
1	$\frac{\beta - 2\|\boldsymbol{x}\| + (\beta + 2\|\boldsymbol{x}\|)e^{-\frac{\beta}{\|\boldsymbol{x}\|}}}{\beta^3}$	$\frac{\beta^2 - 4\beta\|\boldsymbol{x}\| + 6\|\boldsymbol{x}\|^2 - (2\beta\|\boldsymbol{x}\| + 6\|\boldsymbol{x}\|^2)e^{-\frac{\beta}{\|\boldsymbol{x}\|}}}{\beta^4}$

Equation (4.8) amounts to another integral transform of f (not listed in Appendix B) with the compactly supported truncated power function as integration kernel. We will take another look at these functions in the context of multiply monotone functions below.

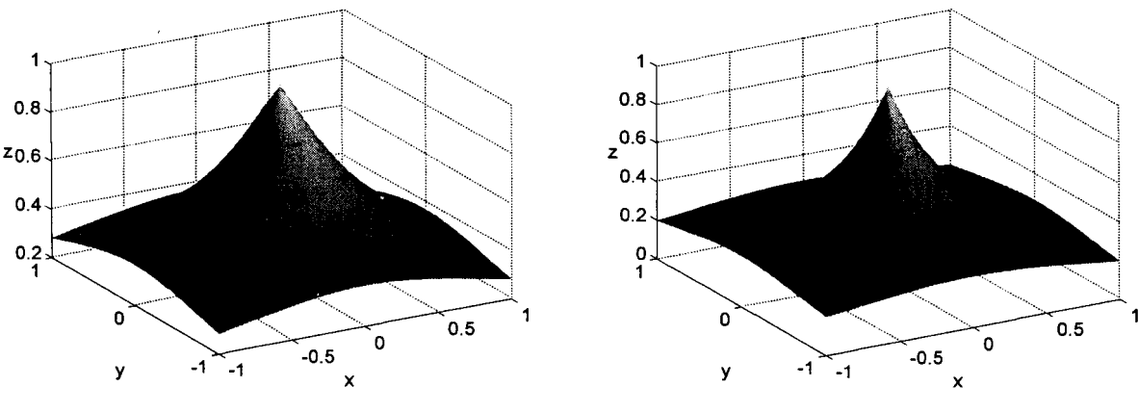


Fig. 4.6 Whittaker radial functions for $\alpha = 0$ and $\beta = 1$ with $k = 2$ (left) and $k = 3$ (right) centered at the origin in $\mathbb{R}^2$.

4.8 Example 8: Integration Against Strictly Positive Definite Kernels

In fact, in [Wendland (2005a)] it is shown that integration of any non-negative function f that is not identically equal to zero against a function $K(t, \cdot)$ that is strictly positive definite on $\mathbb{R}^s$ leads to another function that is strictly positive definite on $\mathbb{R}^s$, *i.e.*,

$$\varphi(r) = \int_0^\infty K(t, r) f(t) dt$$

gives rise to $\Phi = \varphi(\| \cdot \|)$ being strictly positive definite on $\mathbb{R}^s$. By choosing f and K appropriately we can obtain both globally supported and compactly supported functions.

For example, the multiply monotone functions in Williamson's characterization Theorem 5.4 are covered by this general theorem by taking $K(t, r) = (1 - rt)_+^{k-1}$ and f an arbitrary positive function in L_1 so that $d\mu(t) = f(t)dt$. Also, functions that are strictly positive definite and radial on $\mathbb{R}^s$ for all s (or equivalently completely monotone functions) are covered by choosing $K(t, r) = e^{-rt}$.

4.9 Summary

To summarize the theory surveyed thus far we can say that any multivariate (radial) function Φ whose Fourier transform is non-negative can be used to generate a basis for the scattered data interpolation problem by shifting it to the data sites. The function Φ can be positive, oscillatory, or have compact support. However, if Φ has any zeros then it cannot be strictly positive definite on $\mathbb{R}^s$ for all choices of s.

Chapter 5

Completely Monotone and Multiply Monotone Functions

Since Fourier transforms are not always easy to compute, we now present two alternative criteria that allow us to decide whether a function is strictly positive definite and radial on $\mathbb{R}^s$ (one for the case of all s, and one for only limited choices of s).

5.1 Completely Monotone Functions

We begin with the former case. To this end we now introduce a class of functions that is very closely related to positive definite radial functions and leads to a simple characterization of such functions.

Definition 5.1. A function $\varphi : [0,\infty) \to \mathbb{R}$ that is in $C[0,\infty) \cap C^\infty(0,\infty)$ and satisfies

$$(-1)^\ell \varphi^{(\ell)}(r) \geq 0, \qquad r > 0,\ \ell = 0,1,2,\ldots,$$

is called *completely monotone on* $[0,\infty)$.

Example 5.1. The function $\varphi(r) = \varepsilon$, $\varepsilon \geq 0$, is completely monotone on $[0,\infty)$.

Example 5.2. The function $\varphi(r) = e^{-\varepsilon r}$, $\varepsilon \geq 0$, is completely monotone on $[0,\infty)$ since

$$(-1)^\ell \varphi^{(\ell)}(r) = \varepsilon^\ell e^{-\varepsilon r} \geq 0, \qquad \ell = 0,1,2,\ldots.$$

Example 5.3. The function $\varphi(r) = \dfrac{1}{(1+r)^\beta}$, $\beta \geq 0$, is completely monotone on $[0,\infty)$ since

$$(-1)^\ell \varphi^{(\ell)}(r) = (-1)^{2\ell}\beta(\beta+1)\cdots(\beta+\ell-1)(1+r)^{-\beta-\ell} \geq 0, \qquad \ell = 0,1,2,\ldots.$$

Some properties of completely monotone functions that can be found in [Cheney and Light (1999); Feller (1966); Widder (1941)] are:

(1) A non-negative finite linear combination of completely monotone functions is completely monotone.
(2) The product of two completely monotone functions is completely monotone.

(3) If φ is completely monotone and ψ is absolutely monotone (*i.e.*, $\psi^{(\ell)} \geq 0$ for all $\ell \geq 0$), then $\psi \circ \varphi$ is completely monotone.
(4) If φ is completely monotone and ψ is a positive function such that its derivative is completely monotone, then $\varphi \circ \psi$ is completely monotone.

Note that the functions in the second and third example above are, except for a variable substitution $r \mapsto r^2$, similar to the Gaussian and inverse multiquadrics mentioned earlier. In order to see how completely monotone functions are related to strictly positive definite radial functions we require an integral characterization of completely monotone functions.

Theorem 5.1 (Hausdorff-Bernstein-Widder). *A function $\varphi : [0, \infty) \to \mathbb{R}$ is completely monotone on $[0, \infty)$ if and only if it is the Laplace transform of a finite non-negative Borel measure μ on $[0, \infty)$, i.e., φ is of the form*

$$\varphi(r) = \mathcal{L}\mu(r) = \int_0^\infty e^{-rt} d\mu(t).$$

Proof. Widder's proof of this theorem can be found in [Widder (1941)], p. 160, where he reduces the proof of this theorem to another theorem by Hausdorff on completely monotone sequences. A detailed proof can also be found in the books [Cheney and Light (1999); Wendland (2005a)]. □

Theorem 5.1 shows that, in the spirit of our earlier remarks, the function $\varphi(r) = e^{-\varepsilon r}$ can be viewed as the fundamental completely monotone function.

The following connection between positive definite radial and completely monotone functions was first pointed out by Schoenberg in 1938.

Theorem 5.2. *A function φ is completely monotone on $[0, \infty)$ if and only if $\Phi = \varphi(\|\cdot\|^2)$ is positive definite and radial on $\mathbb{R}^s$ for all s.*

Note that the function Φ is now defined via the *square* of the norm. This differs from our definition of radial functions (see Definition 2.1).

Proof. One possibility is to use a change of variables to combine Schoenberg's characterization of functions that are positive definite and radial on any $\mathbb{R}^s$, Theorem 3.8, with the Hausdorff-Bernstein-Widder characterization of completely monotone functions. To get more insight we present an alternative proof of the claim that the completely monotone function φ gives rise to a Φ that is positive definite and radial on any $\mathbb{R}^s$. Details for the other direction can be found, *e.g.*, in [Wendland (2005a)].

The Hausdorff-Bernstein-Widder theorem implies that we can write φ as

$$\varphi(r) = \int_0^\infty c^{-rt} d\mu(t)$$

with a finite non-negative Borel measure μ. Therefore, $\Phi(\boldsymbol{x}) = \varphi(\|\boldsymbol{x}\|^2)$ has the representation

$$\Phi(\boldsymbol{x}) = \int_0^\infty e^{-\|\boldsymbol{x}\|^2 t} d\mu(t).$$

To see that this function is positive definite on any $\mathbb{R}^s$ we consider the quadratic form

$$\sum_{j=1}^N \sum_{k=1}^N c_j c_k \Phi(\boldsymbol{x}_j - \boldsymbol{x}_k) = \int_0^\infty \sum_{j=1}^N \sum_{k=1}^N c_j c_k e^{-t\|\boldsymbol{x}_j - \boldsymbol{x}_k\|^2} d\mu(t).$$

Since we saw earlier that the Gaussians are strictly positive definite and radial on any $\mathbb{R}^s$ it follows that the quadratic form is non-negative. □

We can see from the previous proof that if the measure μ is not concentrated at the origin, then Φ is even strictly positive definite and radial on any $\mathbb{R}^s$. This condition on the measure is equivalent with φ not being constant. With this additional restriction on φ we can apply the notion of a completely monotone function to the scattered data interpolation problem. The following *interpolation theorem* originates in the work of Schoenberg ([Schoenberg (1938a)], p. 823) who showed that complete monotonicity implies strict positive definiteness, thus providing a very simple test for verifying the well-posedness of many scattered data interpolation problems. A proof that the converse also holds can be found in [Wendland (2005a)].

Theorem 5.3. *A function $\varphi : [0, \infty) \to \mathbb{R}$ is completely monotone but not constant if and only if $\varphi(\|\cdot\|^2)$ is strictly positive definite and radial on $\mathbb{R}^s$ for any s.*

Example 5.4. Since we showed above that the functions $\varphi(r) = e^{-\varepsilon r}$, $\varepsilon > 0$, and $\varphi(r) = 1/(1+r)^\beta$, $\beta \geq 0$, are completely monotone on $[0, \infty)$, and since they are also not constant we know from Theorem 5.3 that the Gaussians $\Phi(\boldsymbol{x}) = \varphi(\|\boldsymbol{x}\|^2) = e^{-\varepsilon^2\|\boldsymbol{x}\|^2}$, $\varepsilon > 0$, and inverse multiquadrics $\Phi(\boldsymbol{x}) = \varphi(\|\boldsymbol{x}\|^2) = 1/(1+\|\boldsymbol{x}\|^2)^\beta$, $\beta \geq 0$, are strictly positive definite and radial on $\mathbb{R}^s$ for all s. Not only is the test for complete monotonicity a simpler one than calculation of the Fourier transforms, but we also are able to verify strict positive definiteness of the inverse multiquadrics without any dependence of s on β.

5.2 Multiply Monotone Functions

We can also use monotonicity to test for strict positive definiteness of radial functions on $\mathbb{R}^s$ for some fixed value of s. To this end we introduce the concept of a *multiply monotone function.*

Definition 5.2. A function $\varphi : (0, \infty) \to \mathbb{R}$ which is in $C^{k-2}(0, \infty)$, $k \geq 2$, and for which $(-1)^l \varphi^{(l)}(r)$ is non-negative, non-increasing, and convex for $l = 0, 1, 2, \ldots, k-$

2 is called *k-times monotone on* $(0,\infty)$. In case $k = 1$ we only require $\varphi \in C(0,\infty)$ to be non-negative and non-increasing.

Since convexity of φ means that $\varphi(\frac{r_1+r_2}{2}) \leq \frac{\varphi(r_1)+\varphi(r_2)}{2}$, or simply $\varphi''(r) \geq 0$ if φ'' exists, a multiply monotone function is in essence just a completely monotone function whose monotonicity is "truncated".

Example 5.5. The truncated power function (*c.f.* (4.7))

$$\varphi_\ell(r) = (1-r)_+^\ell$$

is ℓ-times monotone for any ℓ since

$$(-1)^l \varphi_\ell^{(l)}(r) = \ell(\ell-1)\dots(\ell-l+1)(1-r)_+^{\ell-l} \geq 0, \quad l = 0,1,2,\dots,\ell.$$

We saw in Section 4.6 that the truncated power functions lead to radial functions that are strictly positive definite on $\mathbb{R}^s$ provided $\ell \geq \lfloor s/2 \rfloor + 1$.

Example 5.6. If we define the integral operator I by

$$(If)(r) = \int_r^\infty f(t)dt, \qquad r \geq 0, \tag{5.1}$$

and f is ℓ-times monotone, then If is $\ell+1$-times monotone. This follows immediately from the fundamental theorem of calculus. As we will see later, the operator I plays an important role in the construction of compactly supported radial basis functions.

To make the connection to strictly positive definite radial functions we require an integral representation for the class of multiply monotone functions. This was given in [Williamson (1956)] but apparently already known to Schoenberg in 1940.

Theorem 5.4 (Williamson). *A continuous function $\varphi : (0,\infty) \to \mathbb{R}$ is k-times monotone on $(0,\infty)$ if and only if it is of the form*

$$\varphi(r) = \int_0^\infty (1-rt)_+^{k-1} d\mu(t), \tag{5.2}$$

where μ is a non-negative Borel measure on $(0,\infty)$.

Proof. To see that a function of the form (5.2) is indeed multiply monotone we just need to differentiate under the integral (since derivatives up to order $k-2$ of $(1-rt)_+^{k-1}$ are continuous and bounded). The other direction can be found in [Williamson (1956)]. □

Williamson's characterization shows us that — just like the truncated power functions — the Whittaker radial functions (4.10) in Section 4.7 are based on multiply monotone functions.

For $k \to \infty$ the Williamson characterization corresponds to the Hausdorff-Bernstein-Widder characterization Theorem 5.1 of completely monotone functions

(and is equivalent provided we extend Williamson's work to include continuity at the origin).

We can see from Sections 4.6 and 4.7 that multiply monotone functions give rise to positive definite radial functions. Such a connection was first noted in [Askey (1973)] (and in the one-dimensional case by Pólya) using the truncated power functions of Section 4.6.

In the RBF literature the following theorem was stated in [Micchelli (1986)], and then refined in [Buhmann (1993a)]:

Theorem 5.5 (Micchelli). *Let $k = \lfloor s/2 \rfloor + 2$ be a positive integer. If $\varphi : [0, \infty) \to \mathbb{R}$, $\varphi \in C[0, \infty)$, is k-times monotone on $(0, \infty)$ but not constant, then φ is strictly positive definite and radial on $\mathbb{R}^s$ for any s such that $\lfloor s/2 \rfloor \leq k - 2$.*

We would like to mention that several versions of Theorem 5.5 contain misprints in the literature. The correct form should be as stated above (*c.f.* also the generalization for strictly conditionally positive definite functions, Theorem 9.3).

Using Theorem 5.5 we can now verify the strict positive definiteness of the truncated power functions and Whittaker radial functions of Sections 4.6 and 4.7 without the use of Fourier transforms. Again, as for Gaussians and the Poisson radial functions, we can view the truncated power function as the fundamental compactly supported strictly positive definite radial function since it is obtained using the point evaluation measure in Williamson's characterization of a multiply monotone function.

It is interesting to observe a certain lack of symmetry in the theory for completely monotone and multiply monotone functions. First, in the completely monotone case we can use Theorem 5.3 to conclude that if φ is completely monotone and not constant then $\varphi(\cdot^2)$ is strictly positive definite on $\mathbb{R}^s$ for any s. In the multiply monotone case (see Theorem 5.5) the square is missing. Now it is clear that we cannot expect the statement with a square to be true in the multiply monotone case. To see this we consider the truncated power function $\varphi(r) = (1 - r)_+^\ell$ (which we know — according to Example 5.1 above — to be ℓ-times multiply monotone for any ℓ). However, the function $\psi(r) = (1 - r^2)_+^\ell$ is not strictly positive definite and radial on $\mathbb{R}^s$ for any s since it is not even strictly positive definite and radial on $\mathbb{R}$ (and therefore even much less so on any higher-dimensional space). We can see this from the univariate radial Fourier transform of ψ (see Theorem B.1 of Appendix B with $s = 1$)

$$\begin{aligned}
\mathcal{F}_1\psi(r) &= \frac{1}{\sqrt{r^{-1}}} \int_0^\infty (1 - t^2)_+^\ell t^{\frac{1}{2}} J_{-1/2}(rt) dt \\
&= \sqrt{\frac{2}{\pi}} \int_0^1 (1 - t^2)^\ell \cos(rt) dt \\
&= 2^\ell \Gamma(\ell + 1) \frac{J_{\ell+1/2}(r)}{r^{\ell+1/2}}.
\end{aligned}$$

Here we used the compact support of ψ and the fact that $J_{-1/2}(r) = \sqrt{2/\pi r}\cos r$. The function $\mathcal{F}_1\psi$ is oscillatory, and therefore ψ cannot be strictly positive definite (*c.f.* Theorem 3.5). In fact, the Fourier transform $\mathcal{F}_1\psi$ is closely related to the Poisson radial functions of Section 4.3.

Moreover, in the completely monotone case we have an equivalence between completely monotone and strictly positive definite functions that are radial on any $\mathbb{R}^s$ (see Theorem 5.3). Again, we cannot expect such an equivalence to hold in the multiply monotone case, *i.e.*, the converse of Theorem 5.5 cannot be true. This is clear since we have already seen a number of functions that are strictly positive definite and radial, but not monotone at all — namely the oscillatory Laguerre-Gaussians of Section 4.2 and the Poisson radial functions of Section 4.3.

However, it is interesting to combine the Schoenberg Theorem 5.3 and Theorem 5.5 based on Williamson's characterization. If one starts with the strictly positive definite radial Gaussian $\varphi(r) = e^{-\varepsilon^2 r^2}$, then Theorem 5.3 tells us that $\phi(r) = \varphi(\sqrt{r}) = e^{-\varepsilon^2 r}$ is completely monotone. Now, any function that is completely monotone is also multiply monotone of any order, so that we can use Theorem 5.5 and conclude that the function $\phi(r) = e^{-\varepsilon^2 r}$ is also strictly positive definite and radial on $\mathbb{R}^s$ for all s. Of course, now we can repeat the argument and conclude that $\psi(r) = e^{-\varepsilon^2\sqrt{r}}$ is strictly positive definite and radial on $\mathbb{R}^s$ for all s, and so on (see [Wendland (2005c)]). This result was already known to Schoenberg (at least in the non-strict case).

As a final remark in this chapter we mention that we are a long way from having a complete characterization of (radial) functions for which the scattered data interpolation problem has a unique solution. As we will see later, such an (as of now unknown) characterization will involve also functions which are not strictly positive definite. For example, we will mention a result of Micchelli's according to which *conditionally* positive definite functions of order one can be used for the scattered data interpolation problem. Furthermore, all of the results dealt with so far involve radial basis functions that are centered at the given data sites. There are only limited results addressing the situation in which the centers for the basis functions and the data sites may differ.

Chapter 6

Scattered Data Interpolation with Polynomial Precision

6.1 Interpolation with Multivariate Polynomials

As we mentioned in the introduction it is not an easy matter to use polynomials to perform multivariate scattered data interpolation. Only if the data sites are in certain special locations can we guarantee well-posedness of multivariate polynomial interpolation. We now address this problem.

Definition 6.1. We call a set of points $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \mathbb{R}^s$ *m-unisolvent* if the only polynomial of total degree at most m interpolating zero data on $\mathcal{X}$ is the zero polynomial.

This definition guarantees a unique solution for interpolation to given data at a subset of cardinality $M = \binom{m+s}{m}$ of the points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ by a polynomial of degree m. Here M is the dimension of the linear space Π_m^s of polynomials of total degree less than or equal to m in s variables.

For polynomial interpolation at N distinct data sites in $\mathbb{R}^s$ to be a well-posed problem, the polynomial degree needs to be chosen accordingly, *i.e.*, we need $M = N$, and the data sites need to form an m-unisolvent set. This is rather restrictive. For example, this implies that polynomial interpolation at $N = 7$ points in $\mathbb{R}^2$ can not be done in a unique way since we could either attempt to use bivariate quadratic polynomials (for which $M = 6$), or bivariate cubic polynomials (with $M = 10$). There exists no space of bivariate polynomials for which $M = 7$.

We will see in the next chapter that m-unisolvent sets play an important role in the context of conditionally positive definite functions. There, however, even though we will be interested in interpolating N pieces of data, the polynomial degree will be small (usually $m = 1, 2, 3$), and the restrictions imposed on the locations of the data sites by the unisolvency conditions will be rather mild.

A sufficient condition (to be found in [Chui (1988)], Ch. 9) on the points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ to form an m-unisolvent set in $\mathbb{R}^2$ is

Theorem 6.1. *Suppose $\{L_0, \ldots, L_m\}$ is a set of $m+1$ distinct lines in $\mathbb{R}^2$, and that $\mathcal{U} = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_M\}$ is a set of $M = (m+1)(m+2)/2$ distinct points such that the*

first point lies on L_0, the next two points lie on L_1 but not on L_0, and so on, so that the last $m+1$ points lie on L_m but not on any of the previous lines $L_0, \ldots, L_{m-1}$. Then there exists a unique interpolation polynomial of total degree at most m to arbitrary data given at the points in $\mathcal{U}$. Furthermore, if the data sites $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ contain $\mathcal{U}$ as a subset then they form an m-unisolvent set on $\mathbb{R}^2$.

Proof. We use induction on m. For $m = 0$ the result is trivial. Take R to be the matrix arising from polynomial interpolation at the points in $\mathcal{U}$, *i.e.*,

$$R_{jk} = p_k(\boldsymbol{u}_j), \quad j, k = 1, \ldots, M,$$

where the p_k form a basis of Π_m^2. We want to show that the only possible solution to $R\boldsymbol{c} = \boldsymbol{0}$ is $\boldsymbol{c} = \boldsymbol{0}$. This is equivalent to showing that if $p \in \Pi_m^2$ satisfies

$$p(\boldsymbol{u}_i) = 0, \quad i = 1, \ldots, M,$$

then p is the zero polynomial.

For each $i = 1, \ldots, m$, let the equation of the line L_i be given by

$$\alpha_i x + \beta_i y = \gamma_i,$$

where $\boldsymbol{x} = (x, y) \in \mathbb{R}^2$.

Suppose now that p interpolates zero data at all the points $\boldsymbol{u}_i$ as stated above. Since p reduces to a univariate polynomial of degree m on L_m which vanishes at $m+1$ distinct points on L_m, it follows that p vanishes identically on L_m, and so

$$p(x, y) = (\alpha_m x + \beta_m y - \gamma_m) q(x, y),$$

where q is a polynomial of degree $m-1$. But now q satisfies the hypothesis of the theorem with m replaced by $m-1$ and $\mathcal{U}$ replaced by $\widetilde{\mathcal{U}}$ consisting of the first $\binom{m+1}{2}$ points of U. By induction, therefore $q \equiv 0$, and thus $p \equiv 0$. This establishes the uniqueness of the interpolation polynomial. The last statement of the theorem is obvious. □

A similar theorem was already proved in [Chung and Yao (1977)]. Theorem 6.1 can be generalized to $\mathbb{R}^s$ by using hyperplanes. The proof is constructed with the help of an additional induction on s. Chui also gives an explicit expression for the determinant of the matrix associated with (polynomial) interpolation at the set of points $\mathcal{U}$.

Remark 6.1. For later reference we note that $(m-1)$-unisolvency of the points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ is equivalent to the fact that the matrix P with

$$P_{jl} = p_l(\boldsymbol{x}_j), \quad j = 1, \ldots, N, \; l = 1, \ldots, M,$$

has full (column-)rank. For $N = M$ this is the polynomial interpolation matrix.

Example 6.1. As can easily be verified, three collinear points in $\mathbb{R}^2$ are not 1-unisolvent, since a linear interpolant, *i.e.*, a plane through three arbitrary heights at these three collinear points is not uniquely determined. On the other hand, if a set of points in $\mathbb{R}^2$ contains three non-collinear points, then it is 1-unisolvent.

We used the difficulties associated with multivariate polynomial interpolation as one of the motivations for the use of radial basis functions. However, sometimes it is desirable to have an interpolant that exactly reproduces certain types of functions. For example, if the data are constant, or come from a linear function, then it would be nice if our interpolant were also constant or linear, respectively. Unfortunately, the methods we have presented thus far (except for the distance matrix fit in the $s = 1$ case) do not reproduce these simple polynomial functions. Moreover, later on we will be interested in applying our interpolation methods to the numerical solution of partial differential equations, and practitioners (especially of finite element methods) often judge an interpolation method by its ability to pass the so-called *patch test*. An interpolation method passes the standard patch test if it can reproduce linear functions. In engineering applications this translates into exact calculation of constant stress and strain. We will see later that in order to prove error estimates for meshfree approximation methods it is not necessary to be able to reproduce polynomials globally (but local polynomial reproduction is an essential ingredient). Thus, if we are only concerned with the approximation power of a numerical method there is really no need for the standard patch test to hold.

6.2 Example: Reproduction of Linear Functions Using Gaussian RBFs

If we do insist on reproduction of linear functions then the top part of Figure 6.1 shows a Gaussian RBF interpolant ($\varepsilon = 6$) to the bivariate linear function $f(x, y) = (x + y)/2$ based on 1089 uniformly spaced points in the unit square along with the absolute error. Clearly the interpolant is not completely planar — not even to machine precision.

Fortunately, there is a simple remedy for this problem. All we need to do is add the polynomial functions $\boldsymbol{x} \mapsto 1$, $\boldsymbol{x} \mapsto x$, and $\boldsymbol{x} \mapsto y$ to the basis $\{e^{-\varepsilon^2\|\cdot-\boldsymbol{x}_1\|^2}, \ldots, e^{-\varepsilon^2\|\cdot-\boldsymbol{x}_N\|^2}\}$ we have thus far been using to obtain our interpolant. However, now we have $N + 3$ unknowns, namely the coefficients c_k, $k = 1, \ldots, N + 3$, in the expansion

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{k=1}^{N} c_k e^{-\varepsilon^2\|\boldsymbol{x}-\boldsymbol{x}_k\|^2} + c_{N+1} + c_{N+2}x + c_{N+3}y, \qquad \boldsymbol{x} = (x, y) \in \mathbb{R}^2,$$

and we have only N conditions to determine them, namely the interpolation conditions

$$\mathcal{P}_f(\boldsymbol{x}_j) = f(\boldsymbol{x}_j) = (x_j + y_j)/2, \qquad j = 1, \ldots, N.$$

What can we do to obtain a (non-singular) square system? As we will see below, we can add the following three conditions:

$$\sum_{k=1}^{N} c_k = 0,$$

$$\sum_{k=1}^{N} c_k x_k = 0,$$
$$\sum_{k=1}^{N} c_k y_k = 0.$$

How do we have to modify our existing MATLAB program for scattered data interpolation to incorporate these modifications? If we previously dealt with the solution of

$$A\boldsymbol{c} = \boldsymbol{y},$$

with $A_{jk} = e^{-\varepsilon^2 \|\boldsymbol{x}_j - \boldsymbol{x}_k\|^2}$, $j,k = 1,\ldots,N$, $\boldsymbol{c} = [c_1,\ldots,c_N]^T$, and $\boldsymbol{y} = [f(\boldsymbol{x}_1),\ldots,f(\boldsymbol{x}_N)]^T$, then we now have to solve the *augmented system*

$$\begin{bmatrix} A & P \\ P^T & O \end{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{d} \end{bmatrix} = \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{0} \end{bmatrix}, \tag{6.1}$$

where A, $\boldsymbol{c}$, and $\boldsymbol{y}$ are as before, and $P_{jl} = p_l(\boldsymbol{x}_j)$, $j = 1,\ldots,N$, $l = 1,\ldots,3$, with $p_1(\boldsymbol{x}) = 1$, $p_2(\boldsymbol{x}) = x$, and $p_3(\boldsymbol{x}) = y$. Moreover, $\boldsymbol{0}$ is a zero vector of length 3, and O is a zero matrix of size 3×3.

The MATLAB script `RBFInterpolation2Dlinear.m` shows an implementation of this approach for Gaussians (although they can easily be replaced by any other RBF) and test function $f(x,y) = (x+y)/2$. The resulting interpolant using $N = 9$ equally spaced data points and $\varepsilon = 6$ is shown in the bottom part of Figure 6.1. Now, while still not perfectly linear, the error is on the level of machine accuracy.

Program 6.1. `RBFInterpolation2Dlinear.m`

```
% RBFInterpolation2Dlinear
% Script that performs 2D RBF interpolation with reproduction of
% linear functions
% Calls on: DistanceMatrix
    % Define the Gaussian RBF and shape parameter
 1  rbf = @(e,r) exp(-(e*r).^2); ep = 6;
    % Define linear test function
 2  testfunction = @(x,y) (x+y)/2;
    % Number and type of data points
 3  N = 9; gridtype = 'u';
    % Load data points
 4  name = sprintf('Data2D_%d%s',N,gridtype); load(name)
 5  ctrs = dsites;
 6  neval = 40; M = neval^2; grid = linspace(0,1,neval);
 7  [xe,ye] = meshgrid(grid); epoints = [xe(:) ye(:)];
    % Evaluate the test function at the data points.
 8  rhs = testfunction(dsites(:,1),dsites(:,2));
```

```
    % Add zeros for linear (2D) reproduction
 9  rhs = [rhs; zeros(3,1)];
    % Compute distance matrix between the data sites and centers
10  DM_data = DistanceMatrix(dsites,ctrs);
    % Compute interpolation matrix
11  IM = rbf(ep,DM_data);
    % Define 3-column matrix P for linear reproduction
12  PM = [ones(N,1) dsites];
    % Augment interpolation matrix
13  IM = [IM PM; [PM' zeros(3,3)]];
    % Compute distance matrix between evaluation points and centers
14  DM_eval = DistanceMatrix(epoints,ctrs);
    % Compute evaluation matrix
15  EM = rbf(ep,DM_eval);
    % Add column for constant reproduction
16  PM = [ones(M,1) epoints]; EM = [EM PM];
    % Compute RBF interpolant
    % (evaluation matrix * solution of interpolation system)
17  Pf = EM * (IM\rhs);
    % Compute maximum error on evaluation grid
18  exact = testfunction(epoints(:,1),epoints(:,2));
19  maxerr = norm(Pf-exact,inf);
20  rms_err = norm(Pf-exact)/neval;
21  fprintf('RMS error:     %e\n', rms_err)
22  fprintf('Maximum error: %e\n', maxerr)
23  fview = [-30,30];
24  plotsurf(xe,ye,Pf,neval,exact,maxerr,fview);
25  ploterror2D(xe,ye,Pf,exact,maxerr,neval,fview);
```

Note that Program 6.1 is almost the same as Program 2.1. The only difference are lines 9, 12, 13, and 16 that have been added to deal with the augmented problem. In Program 6.1 we also modified the definition of the test function.

6.3 Scattered Data Interpolation with More General Polynomial Precision

As we just saw for a specific example, we may want to modify the assumption on the form (1.1) of the solution to the scattered data interpolation Problem 1.1 by adding certain polynomials to the expansion, *i.e.*, $\mathcal{P}_f$ is now assumed to be of the form

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{k=1}^{N} c_k \varphi(\|\boldsymbol{x} - \boldsymbol{x}_k\|) + \sum_{l=1}^{M} d_l p_l(\boldsymbol{x}), \qquad \boldsymbol{x} \in \mathbb{R}^s, \tag{6.2}$$

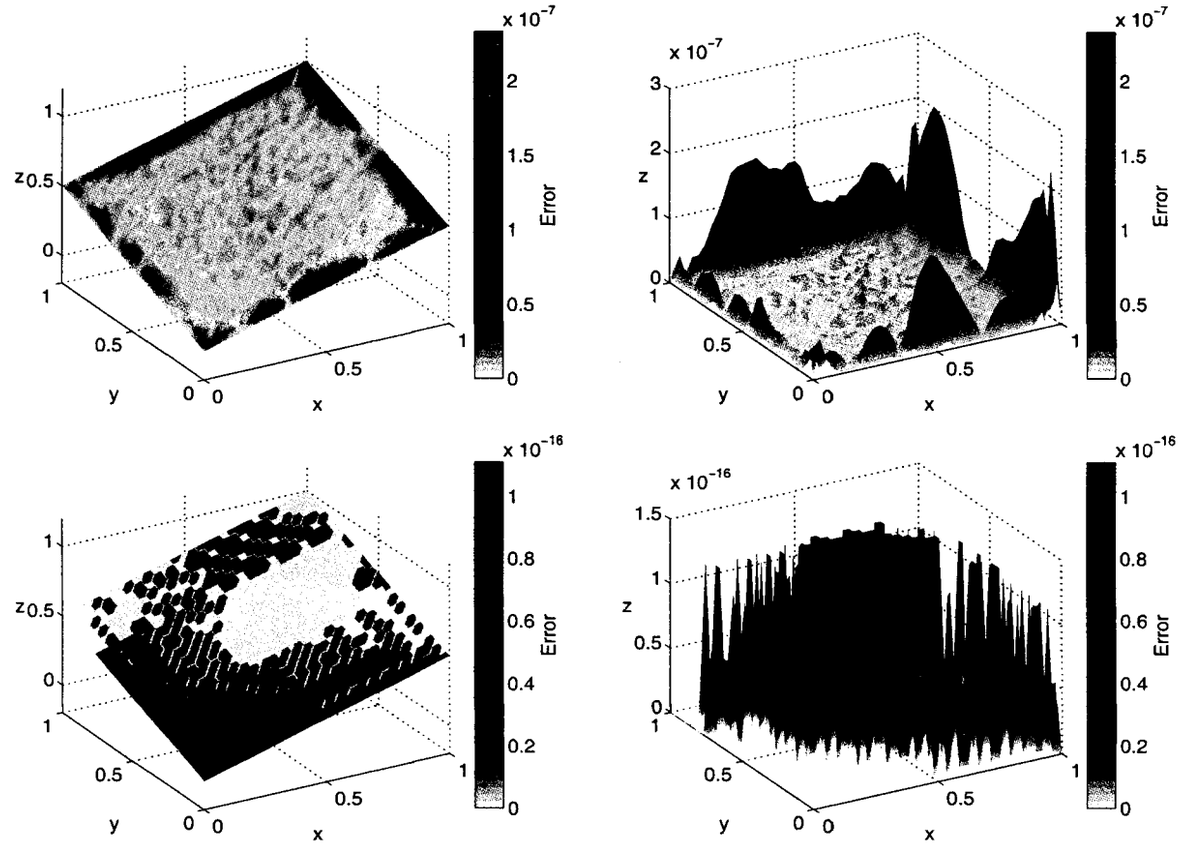


Fig. 6.1 Top: Gaussian interpolant to bivariate linear function with $N = 1089$ (left) and associated abolute error (right). Bottom: Interpolant based on linearly augmented Gaussians to bivariate linear function with $N = 9$ (left) and associated abolute error (right).

where $p_1, \ldots, p_M$ form a basis for the $M = \binom{m-1+s}{m-1}$-dimensional linear space H^s_{m-1} of polynomials of total degree less than or equal to $m-1$ in s variables. It seems awkward to formulate this setup with polynomials in Π^s_{m-1} instead of degree m polynomials. However, in light of our discussion of conditionally positive definite functions in the next chapter this choice is quite natural.

Since enforcing the interpolation conditions $\mathcal{P}_f(\boldsymbol{x}_j) = f(\boldsymbol{x}_j)$, $j = 1, \ldots, N$, leads to a system of N linear equations in the $N + M$ unknowns c_k and d_l one usually adds the M additional conditions

$$\sum_{k=1}^{N} c_k p_l(\boldsymbol{x}_k) = 0, \qquad l = 1, \ldots, M,$$

to ensure a unique solution. The example in the previous section represents the particular case $s = m = 2$.

While the use of polynomials is somewhat arbitrary (any other set of M linearly independent functions could also be used), it is obvious that the addition of polynomials of total degree at most $m-1$ guarantees polynomial precision provided the points in $\mathcal{X}$ form an $(m-1)$-unisolvent set. In other words, if the data come from a

polynomial of total degree less than or equal to $m-1$, then they are fitted exactly by the expansion (6.2).

In general, solving the interpolation problem based on the extended expansion (6.2) now amounts to solving a system of linear equations of the form

$$\begin{bmatrix} A & P \\ P^T & O \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ \boldsymbol{d} \end{bmatrix} = \begin{bmatrix} \boldsymbol{y} \\ \mathbf{0} \end{bmatrix}, \tag{6.3}$$

where the pieces are given by $A_{jk} = \varphi(\|\boldsymbol{x}_j - \boldsymbol{x}_k\|)$, $j,k = 1,\ldots,N$, $P_{jl} = p_l(\boldsymbol{x}_j)$, $j = 1,\ldots,N$, $l = 1,\ldots,M$, $\mathbf{c} = [c_1,\ldots,c_N]^T$, $\boldsymbol{d} = [d_1,\ldots,d_M]^T$, $\boldsymbol{y} = [y_1,\ldots,y_N]^T$, $\mathbf{0}$ is a zero vector of length M, and O is an $M \times M$ zero matrix. Below we will study the invertibility of this matrix in two steps. First for the case $m = 1$ in Theorem 6.2, and then for the case of general m in Theorem 7.2.

Note that we can easily modify the MATLAB program listed above to deal with reproduction of polynomials of other degrees. For example, if we want to reproduce constants then we need to replace lines 9, 12, 13, and 16 by

```
 9  rhs = [rhs; 0];
12  PM = ones(N,1);
13  IM = [IM PM; [PM' 0]];
16  PM = ones(M,1); EM = [EM PM];
```

and for reproduction of bivariate quadratic polynomials we can use

```
 9  rhs = [rhs; zeros(6,1)];
12a PM = [ones(N,1) dsites dsites(:,1).^2 ...
12b       dsites(:,2).^2 dsites(:,1).*dsites(:,2)];
13  IM = [IM PM; [PM' zeros(6,6)]];
16a PM = [ones(M,1) epoints epoints(:,1).^2 ...
16b       epoints(:,2).^2 epoints(:,1).*epoints(:,2)];
16c EM = [EM PM];
```

Of course, these specific examples work only for the case $s = 2$. The generalization to higher dimensions, however, is obvious but more cumbersome.

6.4 Conditionally Positive Definite Matrices and Reproduction of Constant Functions

We now need to investigate whether the augmented system matrix in (6.3) is nonsingular. The special case $m = 1$ (in any space dimension s), *i.e.*, reproduction of constants, is covered by standard results from linear algebra, and we discuss it first.

Definition 6.2. A real symmetric matrix A is called *conditionally positive semi-*

definite of order one if its associated quadratic form is non-negative, *i.e.*

$$\sum_{j=1}^{N}\sum_{k=1}^{N} c_j c_k A_{jk} \geq \mathbf{0} \tag{6.4}$$

for all $\boldsymbol{c} = [c_1, \ldots, c_N]^T \in \mathbb{R}^N$ that satisfy

$$\sum_{j=1}^{N} c_j = 0.$$

If $\boldsymbol{c} \neq \mathbf{0}$ implies strict inequality in (6.4) then A is called *conditionally positive definite of order one.*

In the linear algebra literature the definition usually is formulated using "$\leq$" in (6.4), and then A is referred to as (conditionally or almost) *negative* definite. Obviously, conditionally positive definite matrices of order one exist only for $N > 1$.

We can interpret a matrix A that is conditionally positive definite of order one as one that is positive definite on the space of vectors $\boldsymbol{c}$ such that

$$\sum_{j=1}^{N} c_j = 0.$$

Thus, in this sense, A is positive definite on the space of vectors $\boldsymbol{c}$ "perpendicular" to constant functions.

Now we are ready to formulate and prove

Theorem 6.2. *Let A be a real symmetric $N \times N$ matrix that is conditionally positive definite of order one, and let $P = [1, \ldots, 1]^T$ be an $N \times 1$ matrix (column vector). Then the system of linear equations*

$$\begin{bmatrix} A & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ d \end{bmatrix} = \begin{bmatrix} \boldsymbol{y} \\ 0 \end{bmatrix},$$

is uniquely solvable.

Proof. Assume $[\boldsymbol{c}, d]^T$ is a solution of the homogeneous linear system, *i.e.*, with $\boldsymbol{y} = \mathbf{0}$. We show that $[\boldsymbol{c}, d]^T = \mathbf{0}^T$ is the only possible solution.

Multiplication of the top block of the (homogeneous) linear system by $\boldsymbol{c}^T$ yields

$$\boldsymbol{c}^T A \boldsymbol{c} + d\boldsymbol{c}^T P = 0.$$

From the bottom block of the system we know $P^T \boldsymbol{c} = \boldsymbol{c}^T P = 0$, and therefore

$$\boldsymbol{c}^T A \boldsymbol{c} = 0.$$

Since the matrix A is conditionally positive definite of order one by assumption we get that $\boldsymbol{c} = \mathbf{0}$. Finally, the top block of the homogeneous linear system under consideration states that

$$A\boldsymbol{c} + dP = \mathbf{0},$$

so that $\boldsymbol{c} = \mathbf{0}$ and the fact that P is a vector of ones imply $d = 0$. □

Since Gaussians (and any other strictly positive definite radial function) give rise to positive definite matrices, and since positive definite matrices are also conditionally positive definite of order one, Theorem 6.2 establishes the nonsingularity of the (augmented) radial basis function interpolation matrix for constant reproduction.

In order to cover radial basis function interpolation with reproduction of higher-order polynomials we will now introduce (strictly) conditionally positive definite functions of order m.

Chapter 7

Conditionally Positive Definite Functions

7.1 Conditionally Positive Definite Functions Defined

In analogy to our earlier discussion of interpolation with positive definite functions we will now introduce conditionally positive definite and strictly conditionally positive definite functions of order m. We will realize that these functions provide the natural generalization of RBF interpolation with polynomial reproduction discussed in the previous chapter. Examples of strictly conditionally positive definite (radial) functions are presented in the next chapter.

Definition 7.1. A complex-valued continuous function Φ is called *conditionally positive definite of order m on $\mathbb{R}^s$* if

$$\sum_{j=1}^{N}\sum_{k=1}^{N} c_j \overline{c_k} \Phi(\boldsymbol{x}_j - \boldsymbol{x}_k) \geq 0 \tag{7.1}$$

for any N pairwise distinct points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^s$, and $\mathbf{c} = [c_1, \ldots, c_N]^T \in \mathbb{C}^N$ satisfying

$$\sum_{j=1}^{N} c_j p(\boldsymbol{x}_j) = 0,$$

for any complex-valued polynomial p of degree at most $m-1$. The function Φ is called *strictly conditionally positive definite of order m on $\mathbb{R}^s$* if the quadratic form (7.1) is zero only for $\mathbf{c} \equiv \mathbf{0}$.

An immediate observation is

Lemma 7.1. *A function that is (strictly) conditionally positive definite of order m on $\mathbb{R}^s$ is also (strictly) conditionally positive definite of any higher order. In particular, a (strictly) positive definite function is always (strictly) conditionally positive definite of any order.*

Proof. The first statement follows immediately from Definition 7.1. The second statement is true since the case $m = 0$ yields the class of (strictly) positive definite functions, *i.e.*, (strictly) conditionally positive definite functions of order zero are (strictly) positive definite. □

As for positive definite functions earlier, we can restrict ourselves to real-valued, even functions Φ and real coefficients. A detailed discussion is presented in [Wendland (2005a)].

Theorem 7.1. *A real-valued continuous even function Φ is called* conditionally positive definite of order m on $\mathbb{R}^s$ *if*

$$\sum_{j=1}^{N}\sum_{k=1}^{N} c_j c_k \Phi(\boldsymbol{x}_j - \boldsymbol{x}_k) \geq \mathbf{0} \tag{7.2}$$

for any N pairwise distinct points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^s$, and $\boldsymbol{c} = [c_1, \ldots, c_N]^T \in \mathbb{R}^N$ satisfying

$$\sum_{j=1}^{N} c_j p(\boldsymbol{x}_j) = 0,$$

for any real-valued polynomial p of degree at most $m-1$. The function Φ is called strictly conditionally positive definite of order m on $\mathbb{R}^s$ *if the quadratic form (7.2) is zero only for $\boldsymbol{c} \equiv \mathbf{0}$.*

The matrix A with entries $A_{jk} = \Phi(\boldsymbol{x}_j - \boldsymbol{x}_k)$ corresponding to a real and even strictly conditionally positive definite function Φ of order m can also be interpreted as being positive definite on the space of vectors $\boldsymbol{c}$ such that

$$\sum_{j=1}^{N} c_j p(\boldsymbol{x}_j) = 0, \qquad p \in \Pi^s_{m-1}.$$

Thus, in this sense, A is positive definite on the space of vectors $\boldsymbol{c}$ "perpendicular" to s-variate polynomials of degree at most $m-1$.

We can now generalize the interpolation Theorem 6.2 to the case of general polynomial reproduction:

Theorem 7.2. *If the real-valued even function Φ is strictly conditionally positive definite of order m on $\mathbb{R}^s$ and the points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ form an $(m-1)$-unisolvent set, then the system of linear equations (6.3) is uniquely solvable.*

Proof. The proof is almost identical to the proof of Theorem 6.2. Assume $[\boldsymbol{c}, \boldsymbol{d}]^T$ is a solution of the homogeneous linear system, *i.e.*, with $\boldsymbol{y} = \mathbf{0}$. We show that $[\boldsymbol{c}, \boldsymbol{d}]^T = \mathbf{0}$ is the only possible solution.

Multiplication of the top block by $\boldsymbol{c}^T$ yields

$$\boldsymbol{c}^T A \boldsymbol{c} + \boldsymbol{c}^T P \boldsymbol{d} = 0.$$

From the bottom block of (6.3) we know $P^T \boldsymbol{c} = \mathbf{0}$. This implies $\boldsymbol{c}^T P = \mathbf{0}^T$, and therefore

$$\boldsymbol{c}^T A \boldsymbol{c} = 0. \tag{7.3}$$

Since the function Φ is strictly conditionally positive definite of order m by assumption we know that the quadratic form of A (with coefficients such that $P^T\boldsymbol{c} = \mathbf{0}$) above is zero only for $\boldsymbol{c} = \mathbf{0}$. Therefore (7.3) tells us that $\boldsymbol{c} = \mathbf{0}$. The unisolvency of the data sites, *i.e.*, the linear independence of the columns of P (*c.f.* Remark 6.1), and the fact that $\boldsymbol{c} = \mathbf{0}$ guarantee $\boldsymbol{d} = \mathbf{0}$ from the top block

$$A\boldsymbol{c} + P\boldsymbol{d} = \mathbf{0}$$

of (6.3). □

7.2 Conditionally Positive Definite Functions and Generalized Fourier Transforms

As before, integral characterizations help us identify functions that are strictly conditionally positive definite of order m on $\mathbb{R}^s$. An integral characterization of conditionally positive definite functions of order m, *i.e.*, a generalization of Bochner's theorem, can be found in the paper [Sun (1993b)]. However, since the subject matter is rather complicated, and since it does not really help us solve the scattered data interpolation problem, we do not mention any details here.

The Fourier transform characterization of strictly conditionally positive definite functions of order m on $\mathbb{R}^s$ also makes use of some advanced tools from analysis. However, since this characterization is relevant for our purposes we state the result (due to [Iske (1994)]) and collect some of the most relevant concepts from distribution theory in Appendix B.

This distributional approach originated in the manuscript [Madych and Nelson (1983)]. Many more details can be found in the original papers mentioned above as well as in the book [Wendland (2005a)].

Theorem 7.3. *Suppose the complex-valued function $\Phi \in \mathcal{B}$ possesses a generalized Fourier transform $\hat{\Phi}$ of order m which is continuous on $\mathbb{R}^s \setminus \{\mathbf{0}\}$. Then Φ is strictly conditionally positive definite of order m if and only if $\hat{\Phi}$ is non-negative and non-vanishing.*

Theorem 7.3 states that strictly conditionally positive definite functions on $\mathbb{R}^s$ are characterized by the order of the singularity of their generalized Fourier transform at the origin, provided that this generalized Fourier transform is non-negative and non-zero.

Since integral characterizations similar to Schoenberg's Theorems 3.6 and 3.8 are so complicated in the conditionally positive definite case we do not pursue the concept of a conditionally positive definite radial function here. The interested reader is referred to [Guo *et al.* (1993a)] for details. We will discuss some examples of radial functions via the Fourier transform approach in the next chapter, and in Chapter 9 we will explore the connection between completely and multiply monotone functions and conditionally positive definite radial functions.

Chapter 8

Examples of Conditionally Positive Definite Functions

We now present a number of strictly conditionally positive definite (radial) functions that are covered by the Fourier transform characterization Theorem 7.3. The generalized Fourier transforms for these examples are explicitly computed in [Wendland (2005a)]. We will establish the strict conditional positive definiteness of these functions again in detail in the next chapter with the help of completely monotone functions. Included in the examples below are several of the best known radial basic functions such as the multiquadric due to [Hardy (1971)] and the thin plate spline due to [Duchon (1976)].

8.1 Example 1: Generalized Multiquadrics

The *generalized multiquadrics*

$$\Phi(\boldsymbol{x}) = (1 + \|\boldsymbol{x}\|^2)^\beta, \qquad \boldsymbol{x} \in \mathbb{R}^s,\ \beta \in \mathbb{R} \setminus \mathbb{N}_0, \tag{8.1}$$

have generalized Fourier transforms

$$\hat{\Phi}(\boldsymbol{\omega}) = \frac{2^{1+\beta}}{\Gamma(-\beta)} \|\boldsymbol{\omega}\|^{-\beta-s/2} K_{\beta+s/2}(\|\boldsymbol{\omega}\|), \qquad \boldsymbol{\omega} \neq \mathbf{0},$$

of order $m = \max(0, \lceil \beta \rceil)$, where $\lceil \beta \rceil$ denotes the smallest integer greater than or equal to β. Here the K_ν are again the modified Bessel functions of the second kind of order ν (*c.f.* Section 4.5). Note that we need to exclude positive integer values of β since this would lead to polynomials of even degree (see the related discussion in Example 2 below).

Since the generalized Fourier transforms are positive with a singularity of order m at the origin, Theorem 7.3 tells us that the functions

$$\Phi(\boldsymbol{x}) = (-1)^{\lceil \beta \rceil} (1 + \|\boldsymbol{x}\|^2)^\beta, \qquad 0 < \beta \notin \mathbb{N},$$

are strictly conditionally positive definite of order $m = \lceil \beta \rceil$ (and higher).

For $\beta < 0$ the Fourier transform is a classical one and we are back to the generalized inverse multiquadrics of Section 4.5. These functions are again shown to be strictly conditionally positive definite of order $m = 0$, *i.e.*, strictly positive definite.

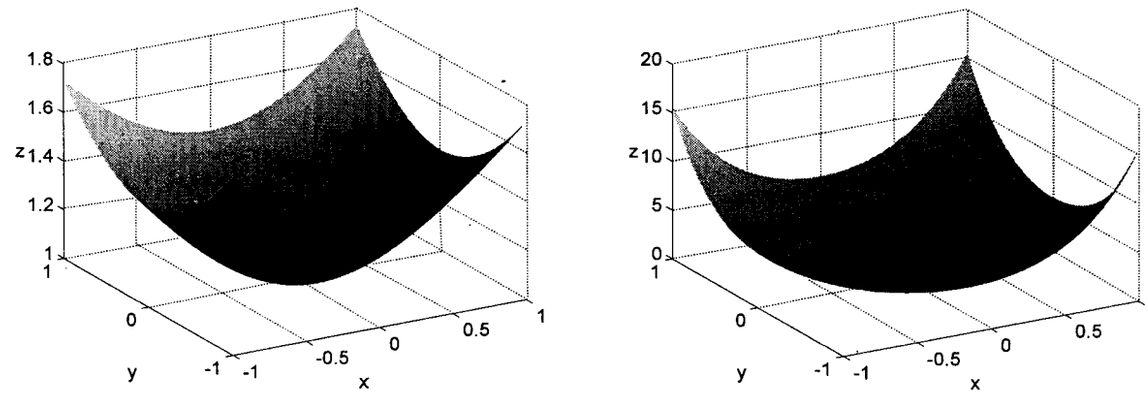


Fig. 8.1 Hardy's multiquadric with $\beta = \frac{1}{2}$ (left) and a generalized multiquadric with $\beta = \frac{5}{2}$ (right) centered at the origin in $\mathbb{R}^2$.

Figure 8.1 shows Hardy's "original" multiquadric (with $\beta = 1/2$, *i.e.*, strictly conditionally positive definite of order 1) and a generalized multiquadric with $\beta = 5/2$ (*i.e.*, strictly conditionally positive definite of order 3). Note that the generalized multiquadrics are no longer "bump" functions (as most of the strictly positive definite functions were), but functions that grow with the distance from the origin.

The arguments above together with Theorem 7.2 show that we can use Hardy's multiquadrics in the form

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{k=1}^{N} c_k \sqrt{1 + \|\boldsymbol{x} - \boldsymbol{x}_k\|^2} + d, \qquad \boldsymbol{x} \in \mathbb{R}^s,$$

together will the constraint

$$\sum_{k=1}^{N} c_k = 0$$

to solve the scattered data interpolation problem. The resulting interpolant will be exact for constant data. As in our earlier discussions we can scale the basis functions with a shape parameter ε by replacing $\|\boldsymbol{x}\|$ by $|\varepsilon|\|\boldsymbol{x}\|$. This does not affect the well-posedness of the interpolation problem. However, a small value of ε gives rise to "flat" basis functions, whereas a large value of ε produces very steep functions. As before, the accuracy of the fit will improve with decreasing ε while the stability will decrease, and the numerical results will become increasingly less reliable. For Figure 8.1 we used the shape parameter $\varepsilon = 1$.

By Theorem 9.7 below we can also solve the scattered data interpolation problem using the simpler expansion

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{k=1}^{N} c_k \sqrt{1 + \|\boldsymbol{x} - \boldsymbol{x}_k\|^2}, \qquad \boldsymbol{x} \in \mathbb{R}^s.$$

This is what Hardy proposed to do in his work in the early 1970s (see, *e.g.*, [Hardy (1971)]).

8.2 Example 2: Radial Powers

The *radial powers*

$$\Phi(\boldsymbol{x}) = \|\boldsymbol{x}\|^\beta, \qquad \boldsymbol{x} \in \mathbb{R}^s,\ 0 < \beta \notin 2\mathbb{N}, \tag{8.2}$$

have generalized Fourier transforms

$$\hat{\Phi}(\boldsymbol{\omega}) = \frac{2^{\beta+s/2}\Gamma(\frac{s+\beta}{2})}{\Gamma(-\beta/2)}\|\boldsymbol{\omega}\|^{-\beta-s}, \qquad \boldsymbol{\omega} \neq \mathbf{0},$$

of order $m = \lceil \beta/2 \rceil$. Therefore, the functions

$$\Phi(\boldsymbol{x}) = (-1)^{\lceil \beta/2 \rceil}\|\boldsymbol{x}\|^\beta, \qquad 0 < \beta \notin 2\mathbb{N},$$

are strictly conditionally positive definite of order $m = \lceil \beta/2 \rceil$ (and higher).

This shows that the basic function $\Phi(\boldsymbol{x}) = \|\boldsymbol{x}\|_2$ used for the distance matrix fits in the introductory chapter are strictly conditionally positive definite of order one. According to Theorem 7.2 we should have used these basic functions together with an appended constant. However, Theorem 9.7 below provides the justification for their use as a pure distance matrix.

In Figure 8.2 we show radial cubics ($\beta = 3$, *i.e.*, strictly conditionally positive definite of order 2) and quintics ($\beta = 5$, *i.e.*, strictly conditionally positive definite of order 3).

Note that we had to exclude even powers in (8.2). This is clear since an even power combined with the square root in the definition of the Euclidean norm results in a polynomial — and we have already decided that polynomials cannot be used for interpolation at arbitrarily scattered multivariate sites.

Note that radial powers are not affected by a scaling of their argument. In other words, radial powers are *shape parameter free*. This has the advantage that the user need not worry about finding a "good" value of ε. On the other hand, we will see below that radial powers will not be able to achieve the spectral convergence rates that are possible with some of the other basic functions such as Gaussians and generalized (inverse) multiquadrics.

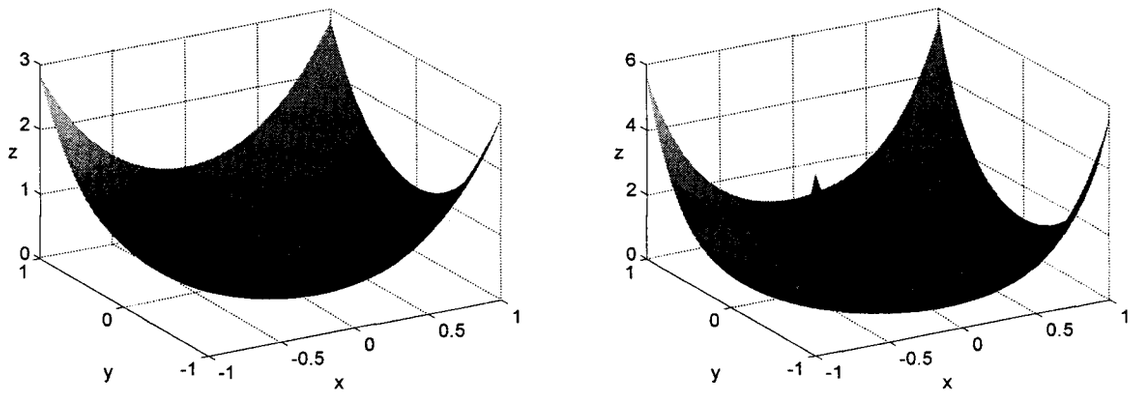


Fig. 8.2 Radial cubic (left) and quintic (right) centered at the origin in $\mathbb{R}^2$.

8.3 Example 3: Thin Plate Splines

In the previous example we had to rule out even powers. However, if the even radial powers are multiplied by a log term, then we are back in business.

Duchon's *thin plate splines* (or Meinguet's *surface splines*)

$$\Phi(\boldsymbol{x}) = \|\boldsymbol{x}\|^{2\beta} \log \|\boldsymbol{x}\|, \qquad \boldsymbol{x} \in \mathbb{R}^s, \ \beta \in \mathbb{N}, \tag{8.3}$$

have generalized Fourier transforms

$$\hat{\Phi}(\boldsymbol{\omega}) = (-1)^{\beta+1} 2^{2\beta-1+s/2} \Gamma(\beta + s/2) \beta! \|\boldsymbol{\omega}\|^{-s-2\beta}$$

of order $m = \beta + 1$. Therefore, the functions

$$\Phi(\boldsymbol{x}) = (-1)^{\beta+1} \|\boldsymbol{x}\|^{2\beta} \log \|\boldsymbol{x}\|, \qquad \beta \in \mathbb{N},$$

are strictly conditionally positive definite of order $m = \beta + 1$. In particular, we can use

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{k=1}^{N} c_k \|\boldsymbol{x} - \boldsymbol{x}_k\|^2 \log \|\boldsymbol{x} - \boldsymbol{x}_k\| + d_1 + d_2 x + d_3 y, \qquad \boldsymbol{x} = (x, y) \in \mathbb{R}^2,$$

together will the constraints

$$\sum_{k=1}^{N} c_k = 0, \qquad \sum_{k=1}^{N} c_k x_k = 0, \qquad \sum_{k=1}^{N} c_k y_k = 0,$$

to solve the scattered data interpolation problem in $\mathbb{R}^2$ provided the data sites are not all collinear. The resulting interpolant will be exact for data coming from a bivariate linear function.

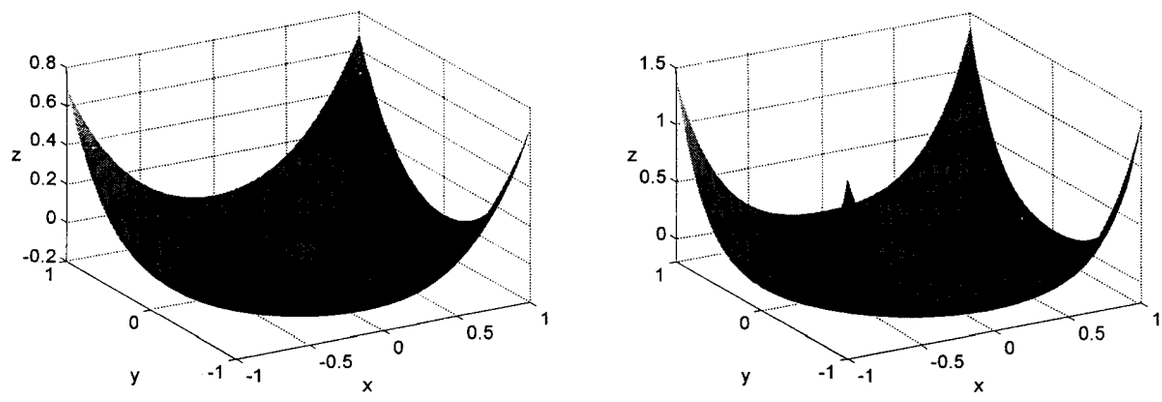


Fig. 8.3 "Classical" thin plate spline (left) and order 3 thin plate spline (right) centered at the origin in $\mathbb{R}^2$.

Figure 8.3 shows the "classical" thin plate spline (with $\beta = 1$, *i.e.*, strictly conditionally positive definite of order 2) and the order 3 spline $\Phi(\boldsymbol{x}) = \|\boldsymbol{x}\|^4 \log \|\boldsymbol{x}\|$. Note that the thin plate spline basic functions are not monotone. Also, both graphs displayed in Figure 8.3 contain a portion with negative function values.

As with radial powers, use of a shape parameter ε in conjunction with thin plate splines is pointless. Finally, we note that the families of radial powers and thin plate splines are often referred to collectively as *polyharmonic splines.*

There is no result that states that interpolation with thin plate splines (or any other strictly conditionally positive definite function of order $m \geq 2$) without the addition of an appropriate degree $m - 1$ polynomial is well-posed. Theorem 9.7 quoted several times before covers only the case $m = 1$.

Chapter 9

Conditionally Positive Definite Radial Functions

As for strictly positive definite radial functions, we will be able to connect strictly conditionally positive definite radial functions to completely monotone and multiply monotone functions, and thus be able to obtain a criterion for checking conditional positive definiteness of radial functions that is easier to use than the generalized Fourier transform in the previous chapters.

9.1 Conditionally Positive Definite Radial Functions and Completely Monotone Functions

In analogy to the discussion in Section 3.3 we now focus on conditionally positive definite functions that are radial on $\mathbb{R}^s$ for all s. The paper [Guo *et al.* (1993a)] by Guo, Hu and Sun contains an integral characterization for such functions. This characterization is too technical to be included here.

Another important result in [Guo *et al.* (1993a)] is a characterization of conditionally positive definite radial functions on $\mathbb{R}^s$ for all s in terms of completely monotone functions.

Theorem 9.1. *Let $\varphi \in C[0,\infty) \cap C^\infty(0,\infty)$. Then the function $\Phi = \varphi(\|\cdot\|^2)$ is conditionally positive definite of order m and radial on $\mathbb{R}^s$ for all s if and only if $(-1)^m\varphi^{(m)}$ is completely monotone on $(0,\infty)$.*

Proof. The fact that complete monotonicity implies conditional positive definiteness was proved in [Micchelli (1986)]. Micchelli also conjectured that the converse holds and gave a simple proof for this in the case $m = 1$. For $m = 0$ this is Schoenberg's characterization of positive definite radial functions on $\mathbb{R}^s$ for all s in terms of completely monotone functions (Theorem 5.2). The remaining part of the theorem is shown in [Guo *et al.* (1993a)]. □

In order to get strict conditional positive definiteness we need to generalize Theorem 5.3, *i.e.*, the fact that φ not be constant. This leads to (see [Wendland (2005a)])

Theorem 9.2. *If φ is as in Theorem 9.1 and not a polynomial of degree at most m, then Φ is strictly conditionally positive definite of order m and radial on $\mathbb{R}^s$ for all s.*

We can now more easily verify the conditional positive definiteness of the functions listed in the previous chapter.

Example 9.1. The functions

$$\varphi(r) = (-1)^{\lceil\beta\rceil}(1+r)^\beta, \qquad 0 < \beta \notin \mathbb{N}$$

imply

$$\varphi^{(\ell)}(r) = (-1)^{\lceil\beta\rceil}\beta(\beta-1)\cdots(\beta-\ell+1)(1+r)^{\beta-\ell}$$

so that

$$(-1)^{\lceil\beta\rceil}\varphi^{(\lceil\beta\rceil)}(r) = \beta(\beta-1)\cdots(\beta-\lceil\beta\rceil+1)(1+r)^{\beta-\lceil\beta\rceil}$$

is completely monotone. Moreover, $m = \lceil\beta\rceil$ is the smallest possible m such that $(-1)^m\varphi^{(m)}$ is completely monotone. Since $\beta \notin \mathbb{N}$ we know that φ is not a polynomial, and therefore the generalized multiquadrics (*c.f.* (8.1))

$$\Phi(\|\boldsymbol{x}\|) = (-1)^{\lceil\beta\rceil}(1+\|\boldsymbol{x}\|^2)^\beta, \qquad \beta > 0,$$

are strictly conditionally positive definite of order $m \geq \lceil\beta\rceil$ and radial on $\mathbb{R}^s$ for all values of s.

Example 9.2. The functions

$$\varphi(r) = (-1)^{\lceil\beta/2\rceil}r^{\beta/2}, \qquad 0 < \beta \notin 2\mathbb{N},$$

imply

$$\varphi^{(\ell)}(r) = (-1)^{\lceil\beta/2\rceil}\frac{\beta}{2}\left(\frac{\beta}{2}-1\right)\cdots\left(\frac{\beta}{2}-\ell+1\right)r^{\beta/2-\ell}$$

so that $(-1)^{\lceil\beta/2\rceil}\varphi^{(\lceil\beta/2\rceil)}$ is completely monotone and $m = \lceil\beta/2\rceil$ is the smallest possible m such that $(-1)^m\varphi^{(m)}$ is completely monotone. Since β is not an even integer φ is not a polynomial, and therefore, the radial powers (*c.f.* (8.2))

$$\Phi(\|\boldsymbol{x}\|) = (-1)^{\lceil\beta/2\rceil}\|\boldsymbol{x}\|^\beta, \qquad \beta > 0,\ \beta \notin 2\mathbb{N},$$

are strictly conditionally positive definite of order $m \geq \lceil\beta/2\rceil$ and radial on $\mathbb{R}^s$ for all s.

Example 9.3. The thin plate splines (*c.f.* (8.3))

$$\Phi(\|\boldsymbol{x}\|) = (-1)^{\beta+1}\|\boldsymbol{x}\|^{2\beta}\log\|\boldsymbol{x}\|, \qquad \beta \in \mathbb{N},$$

are strictly conditionally positive definite of order $m \geq \beta+1$ and radial on $\mathbb{R}^s$ for all s. To see this we observe that

$$2\Phi(\|\boldsymbol{x}\|) = (-1)^{\beta+1}\|\boldsymbol{x}\|^{2\beta}\log(\|\boldsymbol{x}\|^2).$$

Therefore, we let

$$\varphi(r) = (-1)^{\beta+1} r^\beta \log r, \qquad \beta \in \mathbb{N},$$

which is obviously not a polynomial. Differentiating φ we get

$$\varphi^{(\ell)}(r) = (-1)^{\beta+1}\beta(\beta-1)\cdots(\beta-\ell+1)r^{\beta-\ell}\log r + p_\ell(r), \qquad 1 \le \ell \le \beta,$$

with p_ℓ a polynomial of degree $\beta - \ell$. Therefore, taking $\ell = \beta$ we have

$$\varphi^{(\beta)}(r) = (-1)^{\beta+1}\beta! \log r + C$$

and

$$\varphi^{(\beta+1)}(r) = (-1)^{\beta+1}\frac{\beta!}{r},$$

which is completely monotone on $(0, \infty)$.

9.2 Conditionally Positive Definite Radial Functions and Multiply Monotone Functions

Finally, [Micchelli (1986)] proved a more general version of Theorem 5.5 relating conditionally positive definite radial functions of order m on $\mathbb{R}^s$ (for some fixed value of s) and multiply monotone functions. We state a stronger version due to [Buhmann (1993a)] which ensures strict conditional positive definiteness.

Theorem 9.3. *Let $k = \lfloor s/2 \rfloor - m + 2$ be a positive integer, and suppose $\varphi \in C^{m-1}[0,\infty)$ is not a polynomial of degree at most m. If $(-1)^m\varphi^{(m)}$ is k-times monotone on $(0,\infty)$ but not constant, then φ is strictly conditionally positive definite of order m and radial on $\mathbb{R}^s$ for any s such that $\lfloor s/2 \rfloor \le k + m - 2$.*

Just as we showed earlier that compactly supported radial functions cannot be strictly positive definite on $\mathbb{R}^s$ for all s, it is important to note that there are no truly conditionally positive definite functions with compact support. More precisely (see [Wendland (2005a)]),

Theorem 9.4. *Assume that the complex-valued function $\Phi \in C(\mathbb{R}^s)$ has compact support. If Φ is strictly conditionally positive definite of (minimal) order m, then m is necessarily zero, i.e., Φ is already strictly positive definite.*

Proof. The hypotheses on Φ ensure that it is integrable, and therefore it possesses a classical Fourier transform $\hat{\Phi}$ which is continuous. For integrable functions the generalized Fourier transform coincides with the classical Fourier transform. Theorem 7.3 ensures that $\hat{\Phi}$ is non-negative on $\mathbb{R}^s \setminus \{0\}$ and not identically equal to zero. By continuity we also get $\hat{\Phi}(0) \ge 0$, and Theorem 3.5 shows that Φ is strictly positive definite. □

Theorem 9.3 together with Theorem 9.4 implies that if we perform m-fold anti-differentiation on a non-constant k-times monotone function, then we obtain a function that is strictly positive definite and radial on $\mathbb{R}^s$ for $\lfloor s/2 \rfloor \leq k + m - 2$.

Example 9.4. The function $\varphi_k(r) = (1-r)_+^k$ is k-times monotone (see Example 5.5 in Section 5.2). To avoid the integration constant for the compactly supported truncated power function we compute the anti-derivative via the integral operator I of Example 5.6 in Section 5.2, *i.e.*,

$$I\varphi_k(r) = \int_r^\infty \varphi_k(t)dt = \int_r^\infty (1-t)_+^k dt = \frac{(-1)^k}{k+1}(1-r)_+^{k+1}.$$

If we apply m-fold anti-differentiation we get

$$I^m\varphi_k(r) = II^{m-1}\varphi_k(r) = \frac{(-1)^{mk}}{(k+1)(k+2)\cdots(k+m)}(1-r)_+^{k+m}.$$

Therefore, by Theorem 9.3, the function

$$\varphi(r) = (1-r)_+^{k+m}$$

is strictly conditionally positive definite of order m and radial on $\mathbb{R}^s$ for $\lfloor s/2 \rfloor \leq k+m-2$, and by Theorem 9.4 it is even strictly positive definite and radial on $\mathbb{R}^s$. This was also observed in Example 6 of Chapter 4. In fact, we saw there that φ is strictly positive definite and radial on $\mathbb{R}^s$ for $\lfloor s/2 \rfloor \leq k+m-1$.

We see that we can construct strictly positive definite compactly supported radial functions by anti-differentiating the truncated power function. This is essentially the approach taken by Wendland to construct his popular compactly supported radial basis functions. We provide more details of his construction in Chapter 11.

9.3 Some Special Properties of Conditionally Positive Definite Functions of Order One

Since an $N \times N$ matrix that is conditionally positive definite of order one is positive definite on a subspace of dimension $N-1$ it has the interesting property that at least $N-1$ of its eigenvalues are positive. This follows immediately from the Courant-Fischer theorem of linear algebra (see *e.g.*, p. 550 of [Meyer (2000)]):

Theorem 9.5 (Courant-Fischer). *Let A be a real symmetric $N \times N$ matrix with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$, then*

$$\lambda_k = \max_{\dim \mathcal{V}=k} \min_{\substack{x \in \mathcal{V} \\ \|x\|=1}} x^T A x$$

and

$$\lambda_k = \min_{\dim \mathcal{V}=N-k+1} \max_{\substack{x \in \mathcal{V} \\ \|x\|=1}} x^T A x.$$

With an additional assumption on A we can make an even stronger statement.

Theorem 9.6. *An $N \times N$ matrix A which is conditionally positive definite of order one and has a non-positive trace possesses one negative and $N-1$ positive eigenvalues.*

Proof. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$ denote the eigenvalues of A. From the Courant-Fischer theorem we get

$$\lambda_{N-1} = \max_{\dim \mathcal{V}=N-1} \min_{\substack{x \in \mathcal{V} \\ \|x\|=1}} x^T A x \geq \min_{c:\ \sum c_k=0 \atop \|c\|=1} c^T A c > 0,$$

so that A has at least $N-1$ positive eigenvalues. But since $\operatorname{tr}(A) = \sum_{k=1}^{N} \lambda_k \leq 0$, A also must have at least one negative eigenvalue. □

Note that the additional hypothesis of Theorem 9.6 is satisfied for the interpolation matrix resulting from (the negative) of RBFs such as Hardy's multiquadric or the linear radial function $\varphi(r) = r$ since its diagonal elements correspond to the value of the basic function at the origin.

Moreover, we will now use Theorem 9.6 to conclude that we can use radial functions that are strictly conditionally positive definite of order one (such as the multiquadric, $0 < \beta < 1$, and the norm basic function) *without* appending the constant term to solve the scattered data interpolation problem. This was first proved by [Micchelli (1986)] and motivated by Hardy's earlier work with multiquadrics and Franke's conjecture that the matrix A is non-singular in this case (see [Franke (1982a)]).

Theorem 9.7 (Interpolation). *Suppose Φ is strictly conditionally positive definite of order one and that $\Phi(0) \leq 0$. Then for any distinct points $x_1, \ldots, x_N \in \mathbb{R}^s$ the matrix A with entries $A_{jk} = \Phi(x_j - x_k)$ has $N-1$ positive and one negative eigenvalue, and is therefore non-singular.*

Proof. Clearly, the matrix A is conditionally positive definite of order one. Moreover, the trace of A is given by $\operatorname{tr}(A) = N\Phi(0) \leq 0$. Therefore, Theorem 9.6 applies and the statement follows. □

As mentioned above, this theorem covers the generalized multiquadrics $\Phi(x) = -(1+\|x\|)^\beta$ with $0 < \beta < 1$ (which includes the Hardy multiquadric). The theorem also covers the radial powers $\Phi(x) = -\|x\|^\beta$ for $0 < \beta < 2$ (including the Euclidean distance function).

Another special property of a conditionally positive definite function of order one is

Lemma 9.1. *If C is an arbitrary real constant and the real even function Φ is (strictly) conditionally positive definite of order one, then $\Phi + C$ is also (strictly) conditionally positive definite of order one.*

Proof. Simply consider

$$\sum_{j=1}^{N}\sum_{k=1}^{N} c_j c_k [\Phi(\boldsymbol{x}_j - \boldsymbol{x}_k) + C] = \sum_{j=1}^{N}\sum_{k=1}^{N} c_j c_k \Phi(\boldsymbol{x}_j - \boldsymbol{x}_k) + \sum_{j=1}^{N}\sum_{k=1}^{N} c_j c_k C.$$

The second term on the right is zero since Φ is conditionally positive definite of order one, *i.e.*, $\sum_{j=1}^{N} c_j = 0$, and thus the statement follows. □

Chapter 10

Miscellaneous Theory: Other Norms and Scattered Data Fitting on Manifolds

10.1 Conditionally Positive Definite Functions and p-Norms

In Chapter 1 we used interpolation with distance matrices as a multivariate generalization of the piecewise linear approach. Our choice of the distance matrix approach was motivated by the fact that the associated basis functions, $\Phi_j(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}_j\|$ would satisfy the dependence on the data sites imposed on a multivariate interpolation method by the Mairhuber-Curtis theorem. We made the (natural?) choice of using the Euclidean (2-norm) distance function, and then showed in subsequent chapters that the function $\Phi(\boldsymbol{x}) = -\|\boldsymbol{x}\|_2$ is strictly conditionally positive definite of order one and radial on $\mathbb{R}^s$, and thus our distance matrix approach was indeed well-posed via Micchelli's Theorem 9.7.

We now briefly consider solving the scattered data interpolation problem with radial functions based on other p-norms. These norms are defined as usual as

$$\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^{s} |x_i|^p \right)^{1/p}, \qquad \boldsymbol{x} \in \mathbb{R}^s, \ 1 \leq p < \infty.$$

The content of this section is mostly the subject of the paper [Baxter (1991)].

If we consider only distance matrices, *i.e.*, interpolation matrices generated by the basic function $\Phi(\boldsymbol{x}) = \|\boldsymbol{x}\|_p$, then it was shown in [Dyn *et al.* (1989)] that the choice $p = 1$ leads to a singular matrix already for very simple sets of distinct interpolation points. For example, if $\mathcal{X} = \{(0,0), (1,0), (1,1), (0,1)\}$ then the 1-norm distance matrix is given by

$$\begin{bmatrix} 0 & 1 & 2 & 1 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 1 \\ 1 & 2 & 1 & 0 \end{bmatrix},$$

and it is easy to verify that this matrix is singular. This result has discouraged people from using 1-norm radial basis functions.

However, if we use, *e.g.*, N Halton points, then we have never encountered a singular 1-norm distance matrix in all of our numerical experiments. In fact, the

matrix seems to have $N-1$ negative and one positive eigenvalue (just as predicted by Theorem 9.7 for the 2-norm case).

Figure 10.2 shows various interpolants to the linear function $f(x,y) = (x+y)/2$ on the unit square. The interpolant is false colored according to the maximum error. In the top row of the figure we used a 1-norm distance matrix based on 1089 Halton points. The MATLAB code for generating a p-norm distance matrix fit is virtually identical to our earlier code in Programs 1.1 and 1.2. The only change required is the replacement of lines 6 and 8 of Program 1.1 by

```
6  DM = DM + abs(dr-cc).^p;
8  DM = DM.^(1/p);
```

We can also use this modification of Program 1.1 to produce more general RBF interpolants (see the example with p-norm Gaussians in the bottom row of Figure 10.2 below).

Similar to the 1-norm result from [Dyn *et al.* (1989)] quoted above it was shown in [Baxter (1991)] that for $p > 2$ we cannot in general guarantee non-singular distance matrices, either. On the other hand, a number of numerical experiments showed the p-norm matrices to be non-singular provided uniformly spaced or Halton points in $[0,1]^2$ were used. The second row of Figure 10.2 shows distance matrix interpolants to $f(x,y) = (x+y)/2$ on the unit square using a p-norm distance matrix for $p = 10$ and $p = 100$ based on 25 uniformly spaced points.

These examples show that certainly not all is lost when using p-norm radial basis functions. The situation is similar as with the use of Kansa's method for the collocation solution of elliptic PDEs (see Chapter 38). There do exist configurations of data points for which the interpolation matrix becomes singular. However, these configurations may be rare, and therefore the use of p-norm radial basis functions may be justified in many cases. We point out that we used norms for $p > 2$ even though the Baxter result mentioned above guarantees existence of data sets $\mathcal{X}$ for which the interpolation matrix will be singular. For our examples the interpolation matrix was far from singular. Using 25 uniformly spaced data sites the matrices again exhibited 24 negative and one positive eigenvalue. This use of p-norm radial basis functions certainly deserves further investigation.

The case $1 < p < 2$, however, is much better understood. In [Baxter (1991)] we find

Theorem 10.1. *Suppose $1 < p < 2$ and let A be the p-norm distance matrix with entries*

$$A_{jk} = \|\boldsymbol{x}_j - \boldsymbol{x}_k\|_p, \qquad j,k = 1,\ldots,N.$$

Then the matrix $-A$ is conditionally positive definite of order one. Moreover, it is strictly conditionally positive definite of order one if $N \geq 2$ and the points $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N$ are distinct.

This theorem is derived from a much earlier theorem by Schoenberg relating conditionally positive definite matrices of order one and Euclidean distance matrices. When Schoenberg first studied conditionally positive definite matrices of order one this was in connection with isometric embeddings. Based on earlier work by Karl Menger [Menger (1928)] Schoenberg derived the following result characterizing certain conditionally positive definite matrices as Euclidean distance matrices (see [Schoenberg (1937)]).

Theorem 10.2 (Schoenberg-Menger). *Let A be a real symmetric $N \times N$ matrix with all diagonal entries zero and all other elements positive. Then $-A$ is conditionally positive semi-definite of order one if and only if there exist N points $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N \in \mathbb{R}^N$ for which*

$$A_{jk} = \|\boldsymbol{y}_j - \boldsymbol{y}_k\|_2^2.$$

These points are the vertices of a simplex in $\mathbb{R}^N$.

In the third row of Figure 10.2 we display the interpolants to the test function $f(x, y) = (x+y)/2$ on $[0, 1]^2$ using distance matrix interpolation based on 25 equally spaced points and p-norms with $p = 1.001$ and $p = 2$. Since we use a plain distance interpolant, *i.e.*, $\Phi(\boldsymbol{x}) = \|\boldsymbol{x}\|_p$ it is remarkable that the error using the $p = 1.001$-norm is about two orders of magnitude smaller than the next best p-norm distance matrix fit among our experiments (which we obtained for $p = 100$, *c.f.* Figure 10.2).

The use of different p-norms for different applications has not been studied carefully in the literature.

Two other results regarding interpolation with p-norm radial basis functions can also be found in the literature. In [Wendland (2005a)] we find a reference to [Zastavnyi (1993)] according to which — for space dimensions $s \geq 3$ — the only function that is positive definite and p-norm radial on $\mathbb{R}^s$ is the zero function. Again, somewhat discouraging news. However, there is also good news. The following rather powerful theorem comes from [Baxter (1991)]. Baxter calls the matrix A of Theorem 10.2 an *almost negative definite* matrix (*c.f.* the remarks following Definition 6.2).

Theorem 10.3. *Let $-A$ be an $N \times N$ matrix that is conditionally positive semi-definite of order one with all diagonal entries zero, and let $\varphi(\cdot^2)$ be a function that is conditionally positive definite of order one and radial on $\mathbb{R}^s$. Then the matrix defined by*

$$B_{jk} = -\varphi(A_{jk}), \qquad j, k = 1, \ldots, N,$$

is conditionally positive semi-definite of order one. Moreover, if $N \geq 2$ and no off-diagonal elements of A vanish, then B is strictly conditionally positive definite of order one whenever $\varphi(\cdot^2)$ is strictly conditionally positive definite of order one.

Proof. By Schoenberg's Theorem 10.2 we can write $A_{jk} = \|\boldsymbol{y}_j - \boldsymbol{y}_k\|_2^2$ for appropriate points $\boldsymbol{y}_j \in \mathbb{R}^N$. By assumption $\varphi(\cdot^2)$ is conditionally positive definite of order one and radial, and therefore B is conditionally positive semi-definite of order one. Moreover, if $A_{jk} \neq 0$ for all off-diagonal elements, then $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N$ are distinct, and therefore B is strictly conditionally positive definite of order one provided $\varphi(\cdot^2)$ is strictly conditionally positive definite of order one. □

Since Baxter also shows that if A is a 1-norm distance matrix, then $-A$ is a conditionally positive semi-definite matrix of order one, Theorem 10.3 guarantees that we can use many "standard" radial basic functions in conjunction with the 1-norm for RBF interpolation. For example, the use of 1-norm Gaussians is justified by Theorem 10.3. In the literature one can also find an analog of Bochner's theorem for positive definite 1-norm radial functions due to [Cambanis *at al.* (1983)] (see also [Wendland (2005a)]).

Figure 10.1 shows p-norm Gaussians $\Phi(\boldsymbol{x}) = e^{-\varepsilon^2 \|\boldsymbol{x}\|_p^2}$ for $p = 1$ and $p = 10$. A shape parameter $\varepsilon = 3$ was used. Interpolants to the function $f(x, y) = (x+y)/2$ at 25 equally spaced points in $[0, 1]^2$ using these basic functions with $\varepsilon = 1$ are shown in the bottom row of Figure 10.2.

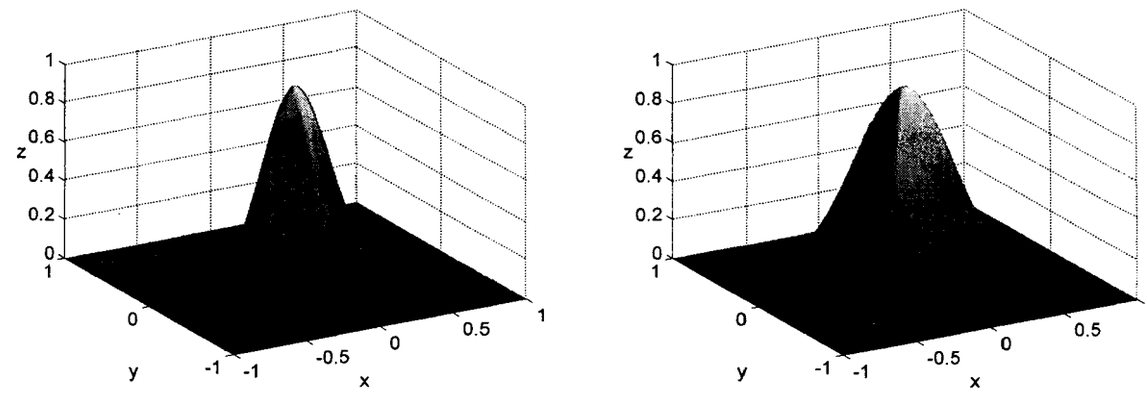


Fig. 10.1 p-norm Gaussians for $p = 1$ (left) and $p = 10$ (right) centered at the origin in $\mathbb{R}^2$.

Another, closely related theorem by Baxter is

Theorem 10.4. *Suppose $\varphi(\cdot^2)$ and $\psi(\cdot^2)$ are functions that are conditionally positive definite of order one and radial on $\mathbb{R}^s$ with $\psi(0) = 0$. Then $\varphi \circ \psi(\cdot^2)$ is also conditionally positive definite of order one and radial on $\mathbb{R}^s$. Indeed, if $\varphi(\cdot^2)$ is strictly conditionally positive definite of order one and radial and ψ vanishes only at zero, then $\varphi \circ \psi(\cdot^2)$ is strictly conditionally positive definite of order one and radial.*

This theorem is a generalization of a classical result in linear algebra by Schur (see, *e.g.*, [Horn and Johnson (1991); Micchelli (1986)], where Schur's theorem was extended to cover strictness).

10.2 Scattered Data Fitting on Manifolds

There exists a sizeable body of literature on the topic of scattered data interpolation on manifolds, especially the sphere S^{s-1} in $\mathbb{R}^s$. We will not mention any specific results here. Instead we refer the reader to the book [Freeden *et al.* (1998)], the survey papers [Cheney (1995a); Fasshauer and Schumaker (1998)], as well as many original papers such as [Baxter and Hubbert (2001); Bingham (1973); Fasshauer (1995a); Fasshauer (1999b); Hubbert and Morton (2004a); Hubbert and Morton (2004b); Levesley *et al.* (1999); Menegatto (1994b); Narcowich and Ward (2002); Ragozin and Levesley (1996); Ron and Sun (1996); Schoenberg (1942); Schreiner (1997); Wahba (1981); Wahba (1982); Xu and Cheney (1992b)].

Radial basis functions on more general Riemannian manifolds are studied in, *e.g.*, [Dyn *et al.* (1997); Dyn *et al.* (1999); Levesley and Ragozin (2002); Narcowich (1995); Narcowich *et al.* (2003); Schimming and Belger (1991)].

There is also a "poor man's solution" to interpolation on manifolds, especially the sphere. One can use the Euclidean radial basis function methods discussed thus far, and simply restrict their evaluation to the manifold. This approach is outlined in Section 6 of [Fasshauer and Schumaker (1998)].

We will discuss another, related, interpolation problem later. Namely, interpolation to point cloud data in $\mathbb{R}^3$. In this case, the underlying manifold is unknown, and another approach needs to be taken. See Chapter 30 for details.

10.3 Remarks

Many of the results given in the previous chapters can be generalized to vector-valued or even matrix-valued functions. Some results along these lines can be found in [Lowitzsch (2002); Lowitzsch (2005); Myers (1992); Narcowich and Ward (1994a); Schaback (1995a)].

We point out that the approach to solving the interpolation problems taken in the previous chapters always assumes that the centers, *i.e.*, the points $\boldsymbol{x}_k$, $k = 1, \ldots, N$, at which the basis functions are centered, coincide with the data sites. This is a fairly severe restriction, and it has been shown in examples in the context of least squares approximation of scattered data (see, *e.g.*, [Franke *et al.* (1994); Franke *et al.* (1995)] or [Fasshauer (1995a)]) that better results can be achieved if the centers are chosen different from the data sites. Theoretical results in this direction are very limited, and are reported in [Quak *et al.* (1993)] and in [Sun (1993a)].

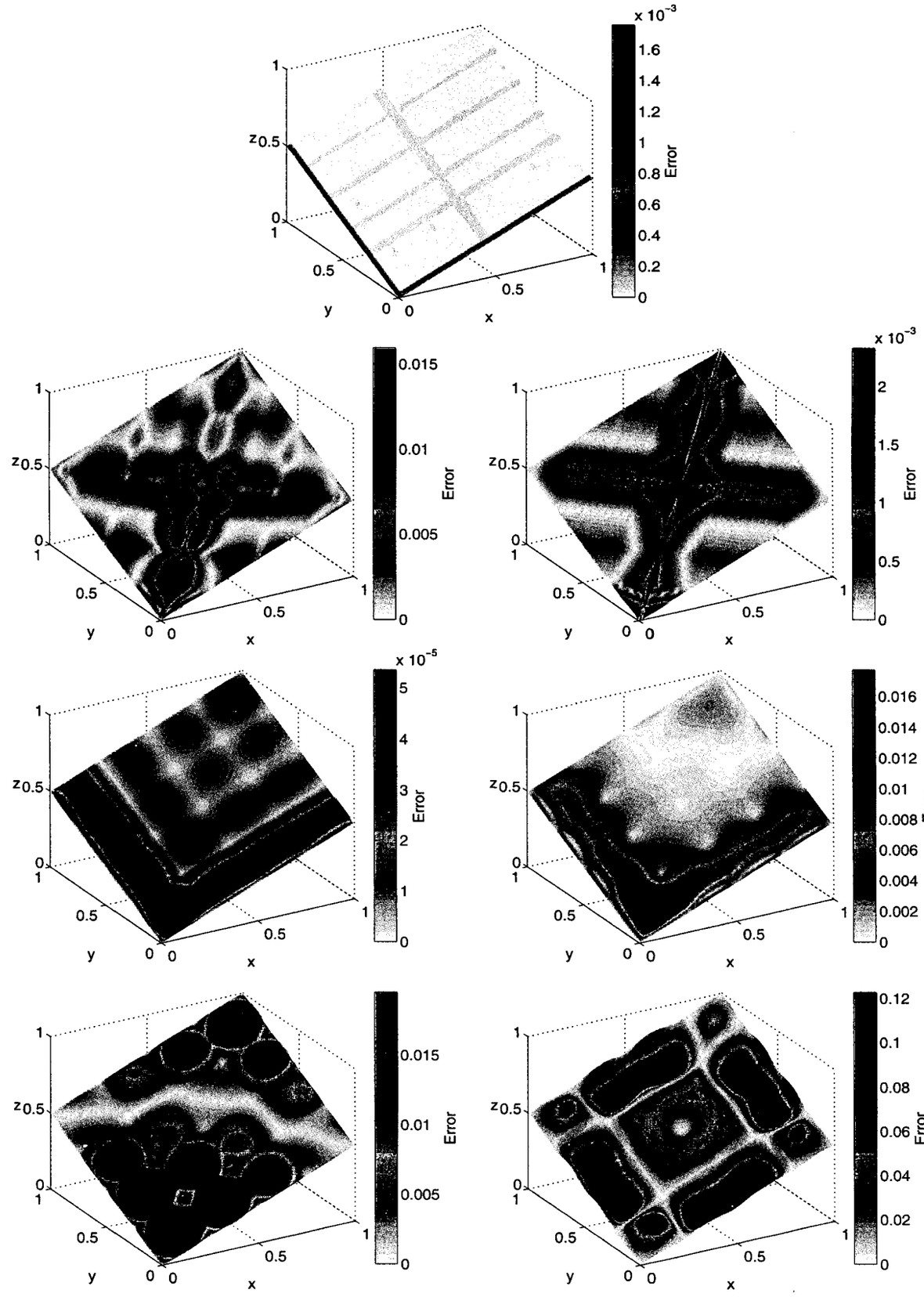


Fig. 10.2 p-norm distance matrix fits to $f(x,y) = (x+y)/2$ on a 5×5 grid in $[0,1]^2$ unless noted otherwise. Top: $p = 1$ (1089 Halton points). 2nd row: $p = 10$ (left), $p = 100$ (right). 3rd row: $p = 1.001$ (left), $p = 2$ (right). Bottom: p-norm Gaussian fits for $p = 1$ (left) and $p = 10$ (right).

Chapter 11

Compactly Supported Radial Basis Functions

As we saw earlier (see Theorem 9.4), compactly supported functions Φ that are truly strictly conditionally positive definite of order $m > 0$ do not exist. The compact support automatically ensures that Φ is strictly positive definite. Another observation (see Theorem 3.9) was that compactly supported radial functions can be strictly positive definite on $\mathbb{R}^s$ only for a fixed maximal s-value. It is not possible for a function to be strictly positive definite and radial on $\mathbb{R}^s$ for all s and also have a compact support. Therefore we focus our attention on the characterization and construction of functions that are compactly supported, strictly positive definite and radial on $\mathbb{R}^s$ for some fixed s.

According to our earlier work (Bochner's theorem and generalizations thereof), a function is strictly positive definite and radial on $\mathbb{R}^s$ if its s-variate Fourier transform is non-negative. Theorem B.1 in the Appendix gives the Fourier transform of the radial function $\Phi = \varphi(\|\cdot\|)$ as another radial function

$$\hat{\Phi}(\boldsymbol{x}) = \mathcal{F}_s\varphi(\|\boldsymbol{x}\|) = \|\boldsymbol{x}\|^{-(s-2)/2} \int_0^\infty \varphi(t) t^{s/2} J_{(s-2)/2}(t\|\boldsymbol{x}\|) dt,$$

where J_ν is the Bessel function of the first kind of order ν.

11.1 Operators for Radial Functions and Dimension Walks

A certain integral operator and its inverse differential operator were defined in [Schaback and Wu (1996)]. In that paper an entire calculus was developed for how these operators act on radial functions. In fact, according to [Gneiting (2002)], these operators can be traced back to [Matheron (1965)] who called the integral operator *montée* and the differential operator *descente* motivated by an application related to mining.

In the following we define these operators and show how they facilitate the construction of compactly supported radial functions.

Definition 11.1.

(1) Let φ be such that $t \mapsto t\varphi(t) \in L_1[0,\infty)$. Then we define the *integral operator* $\mathcal{I}$ via

$$(\mathcal{I}\varphi)(r) = \int_r^\infty t\varphi(t)dt, \qquad r \geq 0.$$

(2) For even $\varphi \in C^2(\mathbb{R})$ we define the *differential operator* $\mathcal{D}$ via

$$(\mathcal{D}\varphi)(r) = -\frac{1}{r}\varphi'(r), \qquad r \geq 0.$$

In both cases the resulting functions are to be interpreted as even functions using even extensions.

Note that the integral operator $\mathcal{I}$ differs from the operator I introduced earlier (see (5.1)) by a factor t in the integrand.

The most important properties of the montée and descente operators are (see, *e.g.*, [Schaback and Wu (1996)] or [Wendland (1995)]):

Theorem 11.1.

(1) *Both $\mathcal{D}$ and $\mathcal{I}$ preserve compact support, i.e., if φ has compact support, then so do $\mathcal{D}\varphi$ and $\mathcal{I}\varphi$.*
(2) *If $\varphi \in C(\mathbb{R})$ and $t \mapsto t\phi(t) \in L_1[0,\infty)$, then $\mathcal{D}\mathcal{I}\varphi = \varphi$.*
(3) *If $\varphi \in C^2(\mathbb{R})$ ($\varphi \not\equiv 1$) is even and $\varphi' \in L_1[0,\infty)$, then $\mathcal{I}\mathcal{D}\varphi = \varphi$.*
(4) *If $t \mapsto t^{s-1}\varphi(t) \in L_1[0,\infty)$ and $s \geq 3$, then $\mathcal{F}_s(\varphi) = \mathcal{F}_{s-2}(\mathcal{I}\varphi)$.*
(5) *If $\varphi \in C^2(\mathbb{R})$ is even and $t \mapsto t^s\varphi'(t) \in L_1[0,\infty)$, then $\mathcal{F}_s(\varphi) = \mathcal{F}_{s+2}(\mathcal{D}\varphi)$.*

The operators $\mathcal{I}$ and $\mathcal{D}$ allow us to express s-variate Fourier transforms as $(s-2)$- or $(s+2)$-variate Fourier transforms, respectively. In particular, a direct consequence of the above properties and the characterization of strictly positive definite radial functions (Theorem 3.6) is

Theorem 11.2.

(1) *Suppose $\varphi \in C(\mathbb{R})$. If $t \mapsto t^{s-1}\varphi(t) \in L_1[0,\infty)$ and $s \geq 3$, then φ is strictly positive definite and radial on $\mathbb{R}^s$ if and only if $\mathcal{I}\varphi$ is strictly positive definite and radial on $\mathbb{R}^{s-2}$.*
(2) *If $\varphi \in C^2(\mathbb{R})$ is even and $t \mapsto t^s\varphi'(t) \in L_1[0,\infty)$, then φ is strictly positive definite and radial on $\mathbb{R}^s$ if and only if $\mathcal{D}\varphi$ is strictly positive definite and radial on $\mathbb{R}^{s+2}$.*

This allows us to construct new strictly positive definite radial functions from given ones by a "dimension-walk" technique that steps through multivariate Euclidean space in even increments. The examples presented in the following sections illustrate this technique.

11.2 Wendland's Compactly Supported Functions

Probably the most popular family of compactly supported radial functions presently in use was constructed in [Wendland (1995)]. Wendland starts with the truncated power function $\varphi_\ell(r) = (1-r)_+^\ell$ (which we know to be strictly positive definite and radial on $\mathbb{R}^s$ for $\ell \geq \lfloor \frac{s}{2} \rfloor + 1$ according to Section 4.6), and then he walks through dimensions by repeatedly applying the operator $\mathcal{I}$.

Definition 11.2. With $\varphi_\ell(r) = (1-r)_+^\ell$ we define

$$\varphi_{s,k} = \mathcal{I}^k \varphi_{\lfloor s/2 \rfloor + k + 1}.$$

It turns out that the functions $\varphi_{s,k}$ are all supported on $[0,1]$ and have a polynomial representation there. More precisely,

Theorem 11.3. *The functions $\varphi_{s,k}$ are strictly positive definite and radial on $\mathbb{R}^s$ and are of the form*

$$\varphi_{s,k}(r) = \begin{cases} p_{s,k}(r), & r \in [0,1], \\ 0, & r > 1, \end{cases}$$

with a univariate polynomial $p_{s,k}$ of degree $\lfloor s/2 \rfloor + 3k + 1$. Moreover, $\varphi_{s,k} \in C^{2k}(\mathbb{R})$ are unique up to a constant factor, and the polynomial degree is minimal for given space dimension s and smoothness $2k$.

This theorem states that any other compactly supported C^{2k} polynomial function that is strictly positive definite and radial on $\mathbb{R}^s$ will not have a smaller polynomial degree. Our other examples below (Wu's functions, Gneiting's functions) illustrate this fact. The strict positive definiteness of Wendland's functions $\varphi_{s,k}$ starting with non-integer values of ℓ in Definition 11.2 was established in [Gneiting (1999)]. Note, however, that then the functions are no longer guaranteed to be polynomials on their support.

Wendland gave recursive formulas for the functions $\varphi_{s,k}$ for all s, k. We instead list the explicit formulas of [Fasshauer (1999a)].

Theorem 11.4. *The functions $\varphi_{s,k}$, $k = 0, 1, 2, 3$, have the form*

$$\begin{aligned}
\varphi_{s,0}(r) &= (1-r)_+^\ell, \\
\varphi_{s,1}(r) &\doteq (1-r)_+^{\ell+1} \left[(\ell+1)r + 1\right], \\
\varphi_{s,2}(r) &\doteq (1-r)_+^{\ell+2} \left[(\ell^2 + 4\ell + 3)r^2 + (3\ell + 6)r + 3\right], \\
\varphi_{s,3}(r) &\doteq (1-r)_+^{\ell+3} \left[(\ell^3 + 9\ell^2 + 23\ell + 15)r^3 + (6\ell^2 + 36\ell + 45)r^2 \right. \\
&\quad \left. + (15\ell + 45)r + 15\right],
\end{aligned}$$

where $\ell = \lfloor s/2 \rfloor + k + 1$, and the symbol $\doteq$ denotes equality up to a multiplicative positive constant.

Proof. The case $k = 0$ follows directly from the definition. Application of the definition for the case $k = 1$ yields

$$\begin{aligned}\varphi_{s,1}(r) = (\mathcal{I}\varphi_\ell)(r) &= \int_r^\infty t\varphi_\ell(t)dt \\ &= \int_r^\infty t(1-t)_+^\ell dt \\ &= \int_r^1 t(1-t)^\ell dt \\ &= \frac{1}{(\ell+1)(\ell+2)}(1-r)^{\ell+1}\left[(\ell+1)r+1\right],\end{aligned}$$

where the compact support of φ_ℓ reduces the improper integral to a definite integral which can be evaluated using integration by parts. The other two cases are obtained similarly by repeated application of $\mathcal{I}$. □

Example 11.1. For $s = 3$ we list some of the most commonly used functions in Table 11.1. These functions are strictly positive definite and radial on $\mathbb{R}^s$ for $s \leq 3$. We also list their degree of smoothness $2k$. The functions were determined using the formulas from Theorem 11.4, *i.e.*, for $k = 1, 2, 3$ they match Definition 11.2 only up to a positive constant factor.

For the MATLAB implementation in the next chapter it is better to express the compactly supported functions in a shifted form since we will be using a matrix version of $1-\varepsilon r$ in place of the code used earlier in `DistanceMatrix.m` for r. Thus we also list the appropriate functions $\widetilde{\varphi}_{s,k} = \varphi_{s,k}(1-\cdot)$ so that $\widetilde{\varphi}_{s,k}(1-\varepsilon r) = \varphi_{s,k}(\varepsilon r)$.

For clarification purposes we reiterate that expressions of the form $(x)_+^\ell$ are to be interpreted as $((x)_+)^\ell$, *i.e.*, we first apply the cutoff function, and then the power.

Table 11.1 Wendland's compactly supported radial functions $\varphi_{s,k}$ and $\widetilde{\varphi}_{s,k} = \varphi_{s,k}(1-\cdot)$ for various choices of k and $s = 3$.

k	$\varphi_{3,k}(r)$	$\widetilde{\varphi}_{3,k}(r)$	smoothness
0	$(1-r)_+^2$	r_+^2	C^0
1	$(1-r)_+^4\,(4r+1)$	$r_+^4\,(5-4r)$	C^2
2	$(1-r)_+^6\,(35r^2+18r+3)$	$r_+^6\,(56-88r+35r^2)$	C^4
3	$(1-r)_+^8\,(32r^3+25r^2+8r+1)$	$r_+^8\,(66-154r+121r^2-32r^3)$	C^6

11.3 Wu's Compactly Supported Functions

In [Wu (1995b)] we find another way to construct strictly positive definite radial functions with compact support. Wu starts with the function

$$\psi(r) = (1-r^2)_+^\ell, \qquad \ell \in \mathbb{N},$$

which in itself is not positive definite (see the discussion at the end of Chapter 5). However, Wu then uses convolution to construct another function that is strictly positive definite and radial on $\mathbb{R}$, *i.e.*,

$$\begin{aligned}\psi_\ell(r) &= (\psi * \psi)(2r) \\ &= \int_{-\infty}^{\infty} (1-t^2)_+^\ell (1-(2r-t)^2)_+^\ell dt \\ &= \int_{-1}^{1} (1-t^2)_+^\ell (1-(2r-t)^2)_+^\ell dt.\end{aligned}$$

This function is strictly positive definite since its Fourier transform is essentially the square of the Fourier transform of ψ and therefore non-negative. Just like the Wendland functions, this function is a polynomial on its support. In fact, the degree of the polynomial is $4\ell+1$, and $\psi_\ell \in C^{2\ell}(\mathbb{R})$.

Now, a family of strictly positive definite radial functions is constructed by a dimension walk using the $\mathcal{D}$ operator.

Definition 11.3. With $\psi_\ell(r) = ((1-\cdot^2)_+^\ell * (1-\cdot^2)_+^\ell)(2r)$ we define

$$\psi_{k,\ell} = \mathcal{D}^k \psi_\ell.$$

The functions $\psi_{k,\ell}$ are strictly positive definite and radial on $\mathbb{R}^s$ for $s \le 2k+1$, are polynomials of degree $4\ell-2k+1$ on their support and in $C^{2(\ell-k)}$ in the interior of the support. On the boundary the smoothness increases to $C^{2\ell-k}$.

Example 11.2. For $\ell = 3$ we can compute the four functions

$$\psi_{k,3}(r) = \mathcal{D}^k \psi_3(r) = \mathcal{D}^k((1-\cdot^2)_+^3 * (1-\cdot^2)_+^3)(2r), \qquad k = 0,1,2,3.$$

They are listed in Table 11.2 along with their smoothness. The maximal space dimension s for which these functions are strictly positive definite and radial on $\mathbb{R}^s$ is also listed. Just as with the Wendland functions, the functions in Table 11.2 match the definition only up to a positive multiplicative constant. Again, we also list the functions $\widetilde{\psi}_{k,\ell} = \psi_{k,\ell}(1-\cdot)$ used in our MATLABimplementation in Chapter 12. This representation of the Wu functions is given in Table 11.3.

Table 11.2 Wu's compactly supported radial functions $\psi_{k,\ell}$ for various choices of k and $\ell = 3$.

k	$\psi_{k,3}(r)$	smoothness	s
0	$(1-r)_+^7(5+35r+101r^2+147r^3+101r^4+35r^5+5r^6)$	C^6	1
1	$(1-r)_+^6(6+36r+82r^2+72r^3+30r^4+5r^5)$	C^4	3
2	$(1-r)_+^5(8+40r+48r^2+25r^3+5r^4)$	C^2	5
3	$(1-r)_+^4(16+29r+20r^2+5r^3)$	C^0	7

Table 11.3 Shifted version $\widetilde{\psi}_{k,\ell}$ of Wu's compactly supported radial functions $\psi_{k,\ell}$ for various choices of k and $\ell = 3$.

k	$\widetilde{\psi}_{k,3}(r)$	smoothness	s
0	$r_+^7(429 - 1287r + 1573r^2 - 1001r^3 + 351r^4 - 65r^5 + 5r^6)$	C^6	1
1	$r_+^6(231 - 561r + 528r^2 - 242r^3 + 55r^4 - 5r^5)$	C^4	3
2	$r_+^5(126 - 231r + 153r^2 - 45r^3 + 5r^4)$	C^2	5
3	$r_+^4(70 - 84r + 35r^2 - 5r^3)$	C^0	7

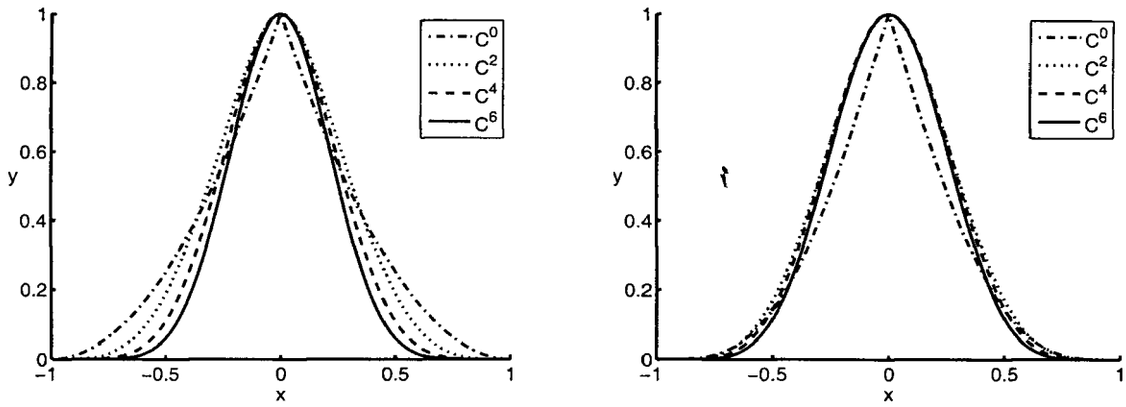


Fig. 11.1 Plot of Wendland's functions from Example 11.1 (left) and Wu's functions from Example 11.2 (right).

As predicted by Theorem 11.3, for a prescribed smoothness the polynomial degree of Wendland's functions is lower than that of Wu's functions. For example, both Wendland's function $\varphi_{3,2}$ and Wu's function $\psi_{1,3}$ are C^4 smooth and strictly positive definite and radial on $\mathbb{R}^3$. However, the polynomial degree of Wendland's function is 8, whereas that of Wu's function is 11. Another comparable function is Gneiting's oscillatory function σ_2 (see Table 11.5), which is a C^4 polynomial of degree 9 that is strictly positive definite and radial on $\mathbb{R}^3$.

While the two families of strictly positive definite compactly supported functions discussed above are both constructed via dimension walk, Wendland uses integration (and thus obtains a family of increasingly smoother functions), whereas Wu needs to start with a function of sufficient smoothness, and then obtains successively less smooth functions (via differentiation).

11.4 Oscillatory Compactly Supported Functions

Other strictly positive definite compactly supported radial functions have been proposed by Gneiting (see, *e.g.*, [Gneiting (2002)]). He showed that a family of oscillatory compactly supported functions can be constructed using the so-called *turning*

bands operator of [Matheron (1973)]. Starting with a function φ_s that is strictly positive definite and radial on $\mathbb{R}^s$ for $s \geq 3$ the turning bands operator produces

$$\varphi_{s-2}(r) = \varphi_s(r) + \frac{r\varphi_s'(r)}{s-2} \tag{11.1}$$

which is strictly positive definite and radial on $\mathbb{R}^{s-2}$.

Example 11.3. One such family of functions is generated is we start with the Wendland functions $\varphi_{s+2,1}(r) = (1-r)_+^{\ell+1}[(\ell+1)r+1]$ (ℓ non-integer allowed). Application of the turning bands operator results in the functions

$$\tau_{s,\ell}(r) = (1-r)_+^{\ell}\left(1+\ell r - \frac{(\ell+1)(\ell+2+s)}{s}r^2\right),$$

which are strictly positive definite and radial on $\mathbb{R}^s$ provided $\ell \geq \frac{s+5}{2}$ (see [Gneiting (2002)]). Some specific functions from this family are listed in Table 11.4. All of the functions are in $C^2(\mathbb{R})$. If we want smoother functions, then we need to start with a smoother Wendland family as described below in Example 11.4.

Table 11.4 Gneiting's compactly supported radial functions $\tau_{s,\ell}$ for various choices of ℓ and $s = 2$.

ℓ	$\tau_{2,\ell}(r)$	smoothness
7/2	$(1-r)_+^{7/2}\left(1+\frac{7}{2}r-\frac{135}{8}r^2\right)$	C^2
5	$(1-r)_+^{5}\left(1+5r-27r^2\right)$	C^2
15/2	$(1-r)_+^{15/2}\left(1+\frac{15}{2}r-\frac{391}{8}r^2\right)$	C^2
12	$(1-r)_+^{12}\left(1+12r-104r^2\right)$	C^2

The functions of Table 11.4 are shown in the left plot of Figure 11.2 with ℓ increasing from the outside in (as viewed near the origin).

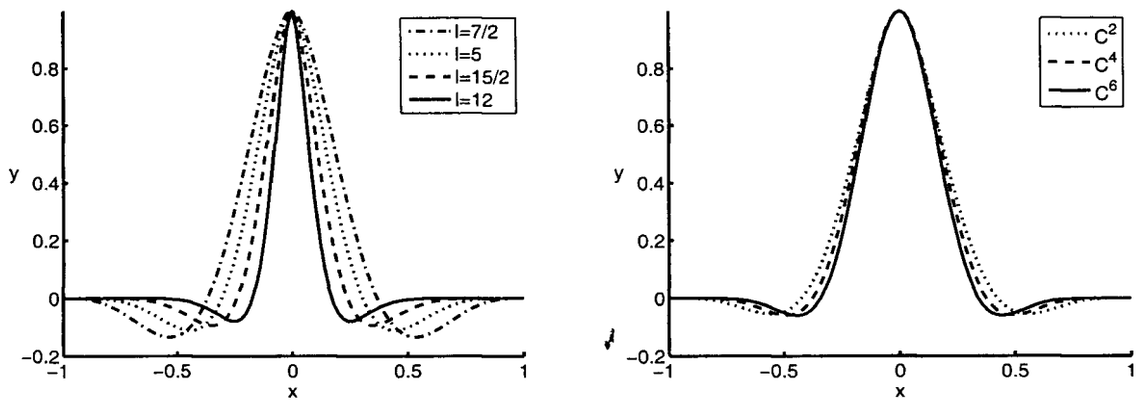


Fig. 11.2 Oscillatory functions of Table 11.4 (left) and Table 11.5 (right).

Example 11.4. Alternatively, we can obtain a set of oscillatory functions that are strictly positive definite and radial on $\mathbb{R}^3$ by applying the turning bands operator to the Wendland functions $\varphi_{5,k}$ that are strictly positive definite and radial on $\mathbb{R}^5$ for different choices of k. Then the resulting functions σ_k will have the same degree of smoothness $2k$ as the original functions and they will be strictly positive definite and radial on $\mathbb{R}^3$. The results for $k = 1, 2, 3$ are listed in Table 11.5 and displayed in the right plot of Figure 11.2.

Table 11.5 Oscillatory compactly supported functions that are strictly positive definite and radial on $\mathbb{R}^3$ parametrized by smoothness.

k	$\sigma_k(r)$	smoothness
1	$(1-r)_+^4\,(1+4r-15r^2)$	C^2
2	$(1-r)_+^6\,(3+18r+3r^2-192r^3)$	C^4
3	$(1-r)_+^8\,(15+120r+210r^2-840r^3-3465r^4)$	C^6

Gneiting also suggests the construction of strictly positive definite radial functions by taking the product of the (appropriately scaled) Poisson functions Ω_s (see either Theorem 3.6 or Section 4.3) with a certain compactly supported non-negative function (see [Gneiting (2002)] for more details). By Property (6) of Theorem 3.1 the resulting function will be strictly positive definite.

11.5 Other Compactly Supported Radial Basis Functions

There are many other ways in which one can construct compactly supported functions that are strictly positive definite and radial on $\mathbb{R}^s$. In [Schaback (1995a)] several such possibilities are described.

Example 11.5. *Euclid's hat* functions are constructed in analogy to B-splines. It is well known that the univariate function $\beta(r) = (1-|r|)_+$ is a second-order B-spline with knots at $-1, 0, 1$, and it is obtained as the convolution of the characteristic function of the interval $[-1/2, 1/2]$ with itself. Euclid's hat functions are now obtained by convolving the characteristic function of the s-dimensional Euclidean unit ball with itself. The resulting functions can be written for $r \in [0,1]$ in the form

$$\varphi_{2k+1}(2r) = \begin{cases} \frac{2\pi\varphi_{2k-1}(2r) - r(1-r^2)^k}{2k+1} & k = 1, 2, 3, \ldots, \\ 2(1-r) & k = 0, \end{cases}$$

for odd space dimensions $s = 2k+1$, and as

$$\varphi_{2k+2}(2r) = \begin{cases} \frac{2\pi\varphi_{2k}(2r) - r\sqrt{(1-r^2)}(1-r^2)^k}{2k+2} & k = 1, 2, 3, \ldots, \\ 2(\arccos r - r\sqrt{1-r^2}) & k = 0, \end{cases}$$

for even space dimensions $s = 2k$. Note that these functions are zero outside the interval $[0, 2]$.

We have listed several of these functions in Table 11.6 where we have employed a substitution $2r \longrightarrow r$ and a normalization factor such that the functions all have a value of one at the origin. The functions are also displayed in the left plot of Figure 11.3.

Table 11.6 Euclid's hat functions (defined for $0 \leq r \leq 2$) for different values of s.

s	$\varphi_s(r)$	smoothness
1	$1 - \frac{r}{2}$	C^0
2	$\frac{1}{2\pi}\left(4\arccos\left(\frac{r}{2}\right) - r\sqrt{4-r^2}\right)$	C^0
3	$1 - \frac{1}{32\pi}\left((4+16\pi)r - r^3\right)$	C^0
4	$\frac{2}{\pi}\arccos\left(\frac{r}{2}\right) - \frac{1}{32\pi}\sqrt{4-r^2}\left(20r + r^3\right)$	C^0
5	$1 - \frac{1}{64\pi^2}\left((12+8\pi+32\pi^2)r - (3+2\pi)r^3\right)$	C^0

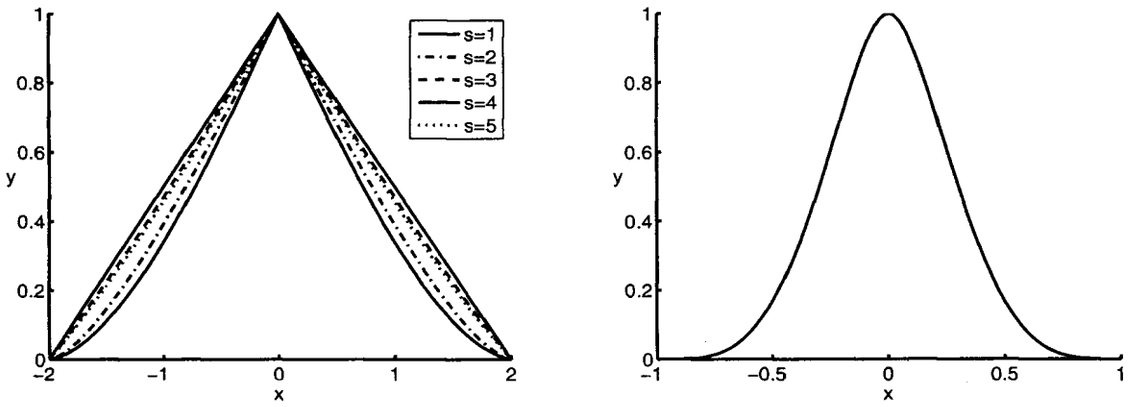


Fig. 11.3 Euclid's hat functions (left) of Table 11.6 and Buhmann's function of Example 11.6 (right).

Another construction described in [Schaback (1995a)] is the radialization of the s-fold tensor product of univariate B-splines of even order $2m$ with uniform knots. These functions do not seem to have a simple representation that lends itself to numerical computations. As can be seen from its radialized Fourier transform, the radialized B-spline itself is not strictly positive definite and radial on any $\mathbb{R}^s$ with $s > 1$. For $s = 1$ only the B-splines of even order are strictly positive definite (see, *e.g.*, [Schölkopf and Smola (2002)]).

The last family of compactly supported strictly positive definite radial functions we would like to mention is due to [Buhmann (1998)]. Buhmann's functions contain

a logarithmic term in addition to a polynomial. His functions have the general form

$$\varphi(r) = \int_0^\infty (1 - r^2/t)_+^\lambda t^\alpha (1 - t^\delta)_+^\rho dt.$$

Here $0 < \delta \leq \frac{1}{2}$, $\rho \geq 1$, and in order to obtain functions that are strictly positive definite and radial on $\mathbb{R}^s$ for $s \leq 3$ the constraints for the remaining parameters are $\lambda \geq 0$, and $-1 < \alpha \leq \frac{\lambda-1}{2}$.

Example 11.6. An example with $\alpha = \delta = \frac{1}{2}$, $\rho = 1$ and $\lambda = 2$ is listed in [Buhmann (2000)]:

$$\varphi(r) \doteq 12r^4 \log r - 21r^4 + 32r^3 - 12r^2 + 1, \qquad 0 \leq r \leq 1.$$

This function is in $C^2(\mathbb{R})$ and strictly positive definite and radial on $\mathbb{R}^s$ for $s \leq 3$. It is displayed in the right plot of Figure 11.3.

While it is stated in [Buhmann (2000)] that the construction there encompasses both Wendland's and Wu's functions, an even more general theorem that shows that integration of a positive function $f \in L_1[0, \infty)$ against a strictly positive definite kernel K results in a strictly positive definite function can be found in [Wendland (2005a)] (see also Section 4.8). More specifically,

$$\varphi(r) = \int_0^\infty K(t, r) f(t) dt$$

is strictly positive definite. Buhmann's construction then corresponds to choosing $f(t) = t^\alpha (1 - t^\delta)_+^\rho$ and $K(t, r) = (1 - r^2/t)_+^\lambda$.

Chapter 12

Interpolation with Compactly Supported RBFs in MATLAB

We now have an alternative way to construct an RBF interpolant to scattered data in $\mathbb{R}^s$. If we use the compactly supported radial functions of the previous chapter then the main difference to our previous interpolants is that now the interpolation matrix can be made *sparse* by scaling the support of the basic function appropriately. To achieve this we use — as we did earlier — the basic functions $\varphi_\varepsilon(r) = \varphi(\varepsilon r)$. Thus, a large value of ε corresponds to a small support. In other words, if the support of φ is the interval $[0, 1]$, then the support radius ρ of φ_ε is given by $\rho = 1/\varepsilon$ so that $\varphi_\varepsilon(r) = 0$ for $r > \rho = 1/\varepsilon$.

Since we know that the interpolation matrix will be a sparse matrix, we want to write MATLAB code to efficiently assemble the matrix. Once we have defined a sparse matrix, MATLAB will automatically use state-of-the-art sparse matrix techniques to solve the linear system. Obviously, we do not want to compute the matrix entries for all pairs of points since we know all of the entries for far away points will be zero. Therefore, an efficient data structure is needed. We use *kd*-trees (implemented in a set of MATLAB MEX-files written by Guy Shechter that can be downloaded from the MATLAB Central File Exchange, see [MCFE]). Some information on *kd*-trees is provided in Appendix A. Data structures for the use with meshfree approximation methods are also discussed in [Wendland (2005a)].

12.1 Assembly of the Sparse Interpolation Matrix

We have structured the scattered data interpolation program in the compactly supported case analogous to the code for the global interpolants, *i.e.*, first construct a distance matrix, and then apply the anonymous function `rbf` to obtain the interpolation/evaluation matrix (as on lines 13–14 and 15–16 of Program 2.1). However, it turns out that it is easier to deal with the compact support if we compute the "distance matrix" corresponding to the $(1 - \varepsilon r)_+$ term since otherwise those entries of the distance matrix that are zero (since the mutual distance between two identical points is zero) would be "lost" in the sparse representation of the matrix.

The MATLAB code `DistanceMatrixCSRBF.m` (Program 12.1) contains two simi-

lar blocks that will be used depending on whether we have more centers than data sites or vice versa. For example, if there are more data sites than centers (cf. lines 7–16), then we build a *kd*-tree for the data sites and find — for each center $\boldsymbol{x}_j$ — those data sites within the support of the basis function centered at $\boldsymbol{x}_j$, *i.e.*, we construct the (sparse) matrix column by column. In the other case (cf. lines 18–27) we start with a tree for the centers and build the matrix row by row. This is accomplished by determining — for each data site $\boldsymbol{x}_i$ — all centers whose associated basis function covers data site $\boldsymbol{x}_i$.

The functions `kdtree` and `kdrangequery` are provided by the *kd*-tree library mentioned above. The call in line 7 (respectively 18) of Program 12.1 generates the *kd*-tree of all the centers (data sites), and with the call to `kdrangequery` in line 9 (respectively 20) we find all centers (data sites) that lie within a distance `support` of the jth center point (data site). The actual distances are returned in the vector `dist` and the indices into the list of all data sites are provided in `idx`. The distances for these points only are stored in the matrix `DM`. For maximum efficiency (in order to avoid dynamic memory allocation) it is important to have a good estimate of the number of nonzero entries in the matrix for the allocation statement in lines 4 and 5. The version of the code presented here has the best performance for larger problems since `sparse` is only invoked once.

Program 12.1. `DistanceMatrixCSRBF.m`

```
% DM = DistanceMatrixCSRBF(dsites,ctrs,ep)
% Forms the distance matrix of two sets of points in R^s
% for compactly supported radial basis functions, i.e.,
%      DM(i,j) = || datasite_i - center_j ||_2.
% The CSRBF used with this code must be given in shifted form
% rbf2(u) = rbf(r), u=1-e*r.
% For example, the Wendland C2
% rbf = @(e,r) max(1-e*r,0).^4.*(4*e*r+1);
% becomes
% rbf2 = @(u) u.^4.*(4*u+5);
% Input
%   dsites: Nxs matrix representing a set of N data sites
%              in R^s (i.e., each row contains one
%              s-dimensional point)
%   ctrs:   Mxs matrix representing a set of M centers for
%              RBFs in R^s (also one center per row)
%   ep:        determines size of support of basis function.
%              Small ep yields wide function,
%              i.e., supportsize = 1/ep
% Output
%   DM:     NxM SPARSE matrix that contains the Euclidean
```

```
%              u-distance (u=1-e*r) between the i-th data
%              site and the j-th center in the i,j position
% Uses:     k-D tree package by Guy Shechter from
%              MATLAB Central File Exchange
 1  function DM = DistanceMatrixCSRBF(dsites,ctrs,ep)
 2  N = size(dsites,1);  M = size(ctrs,1);
    % Build k-D tree for data sites
    % For each center (basis function), find the data sites
    % in its support along with u-distance
 3  support = 1/ep;
 4  nzmax = 25*N; rowidx = zeros(1,nzmax); colidx = zeros(1,nzmax);
 5  validx = zeros(1,nzmax); istart = 1; iend = 0;
 6  if M > N  % faster if more centers than data sites
 7     [tmp,tmp,Tree] = kdtree(ctrs,[]);
 8     for i = 1:N
 9        [pts,dist,idx] = kdrangequery(Tree,dsites(i,:),support);
10        newentries = length(idx);
11        iend = iend + newentries;
12        rowidx(istart:iend) = repmat(i,1,newentries);
13        colidx(istart:iend) = idx';
14        validx(istart:iend) = 1-ep*dist';
15        istart = istart + newentries;
16     end
17  else
18     [tmp,tmp,Tree] = kdtree(dsites,[]);
19     for j = 1:M
20        [pts,dist,idx] = kdrangequery(Tree,ctrs(j,:),support);
21        newentries = length(idx);
22        iend = iend + newentries;
23        rowidx(istart:iend) = idx';
24        colidx(istart:iend) = repmat(j,1,newentries);
25        validx(istart:iend) = 1-ep*dist';
26        istart = istart + newentries;
27     end
28  end
29  idx = find(rowidx);
30  DM = sparse(rowidx(idx),colidx(idx),validx(idx),N,M);
    % Free the k-D Tree from memory.
31  kdtree([],[],Tree);
```

The reason for coding `DistanceMatrixCSRBF.m` in two different ways is so that we will be able to speed up the program when dealing with non-square (evaluation) matrices (for example in the context of MLS approximation (*c.f.* Chapter 24).

One could also implement the distance matrix routine for sparse matrices as follows:

```
1  function DM = DistanceMatrixCSRBF(dsites,ctrs,ep)
2  N = size(dsites,1);  M = size(ctrs,1);
   % Build k-D tree for data sites
   % For each center (basis function), find the data sites
   % in its support along with u-distance
3  support = 1/ep; nzmax = 25*N; DM = spalloc(N,M,nzmax);
4  [tmp,tmp,Tree] = kdtree(dsites,[]);
5  for j = 1:M
6     [pts,dist,idx] = kdrangequery(Tree,ctrs(j,:),support);
7     DM(idx,j) = 1-ep*dist;
8  end
   %  Free the k-D Tree from memory.
9  kdtree([],[],Tree);
```

This code is certainly easier to follow, but not as efficient as the one listed in Program 12.1. Note that we listed only one version of the code here. Clearly, the alternative version can be added analogously to the previous program.

The interpolation program is virtually identical to Program 2.1. The only changes are to replace lines 13 and 15 by the corresponding lines

```
13  DM_data = DistanceMatrixCSRBF(dsites,ctrs,ep);
15  DM_eval = DistanceMatrixCSRBF(epoints,ctrs,ep);
```

and to define the RBF in shifted form, *i.e.*, instead of representing, *e.g.*, the C^2 Wendland function $\varphi_{3,1}$ on line 1 by

```
1  rbf = @(e,r) max(1-e*r,0).^4.*(4*e*r+1); ep=0.7;
```

we now write

```
1  rbf = @(e,r) r.^4.*(5*spones(r)-4*r); ep=0.7;
```

Note the use of the sparse matrix of ones `spones`. Had we used `5-4*r` instead, then a *full* matrix would have been generated (with many additional — and unwanted — ones).

In order to speed up the solution of the (symmetric positive definite) sparse linear system we could use the preconditioned conjugate gradient algorithm (`pcg` in MATLAB) instead of the basic backslash `\` (or matrix left division `mldivide`) operation, *i.e.*, we could replace line 17 of Program 2.1 by

```
17  c = pcg(IM,rhs); Pf = EM * c;
```

Note, however, that the `\` operator also employs state-of-the-art *direct* sparse solvers by first applying a minimum degree preordering.

12.2 Numerical Experiments with CSRBFs

We now present two sets of interpolation experiments with compactly supported RBFs. In Table 12.2 we use the *non-stationary* approach to interpolation, *i.e.*, the support size remains fixed for increasingly denser sets $\mathcal{X}$ of data sites. In this approach we will be able to observe convergence. However, the matrices become increasingly denser, and therefore the non-stationary approach is very inefficient. In Table 12.1, on the other hand, we use the *stationary* approach, *i.e.*, we scale the support size of the basis functions proportionally to the fill distance $h_{\mathcal{X},\Omega}$ (defined in (2.3)). Now the "bandwidth" of the interpolation matrix A is constant. This theoretically results in $\mathcal{O}(N)$ computational complexity, *i.e.*, a very efficient interpolation method. The stationary interpolation method is also numerically stable, but there will be essentially no convergence (see Table 12.1).

We use Wendland's compactly supported function $\varphi_{3,1}(r) = (1-r)_+^4 (4r+1)$ to interpolate Franke's function (2.2) on grids of equally spaced points in the unit square $[0,1]^2$. In the stationary case (Table 12.1) the support of the basis function starts out with an initial scale parameter $\varepsilon = 0.7$ which is subsequently multiplied by a factor of two whenever the fill distance is halved, *i.e.*, when we repeat the experiment on the next finer grid. This corresponds to keeping a constant number of roughly 25 data sites within the support of any basis function. Therefore, the "bandwidth" of the interpolation matrix A is kept constant (at 25), so that A is very sparse for finer grids. We can observe nice convergence for the first few iterations, but once an RMS-error of approximately 5×10^{-3} is reached, there is not much further improvement. This behavior is not yet fully understood. However, it is similar to what happens in the *approximate approximation* method of Maz'ya (see, *e.g.*, [Maz'ya and Schmidt (2001)] and our discussion in Chapter 26). The rate listed in the table is the exponent of the observed RMS-convergence rate $\mathcal{O}(h^{\text{rate}})$. It is computed using the formula

$$\text{rate}_k = \frac{\ln(e_{k-1}/e_k)}{\ln(h_{k-1}/h_k)}, \qquad k = 2, 3, \ldots, \tag{12.1}$$

where e_k is the error for experiment number k, and h_k is the fill distance of the kth computational mesh. Note, that for uniformly spaced points the ratio of fill distances of two consecutive meshes will always be two, while for random points (such as Halton points) we estimate the fill distance via (2.4). The % nonzero column indicates the sparsity of the interpolation matrices, and the time is measured in seconds. Errors are computed on an evaluation grid of 40×40 equally spaced points in $[0,1]^2$.

In the non-stationary case (Table 12.2) we use basis functions without adjusting their support size, *i.e.*, $\varepsilon = 0.7$ is kept fixed for all experiments. We have convergence — although it is not obvious what the rate might be. However, the matrices become increasingly dense and computation requires lots of system memory. Therefore, we left out the solution for the $N = 16641$ and $N = 66049$ cases in Table 12.2. The time

Table 12.1 Stationary interpolation at N equally spaced points in $[0,1]^2$ (constant 25 points in support) with Wendland's function $\varphi(r) = (1-r)_+^4(4r+1)$.

N	RMS-error	rate	% nonzero	time
9	1.562729e-001		100	0.23
25	2.690350e-002	2.5382	57.8	0.31
81	1.027881e-002	1.3881	23.2	0.33
289	6.589552e-003	0.6414	7.47	0.41
1089	3.891263e-003	0.7599	2.13	0.63
4225	3.726913e-003	0.0623	0.57	1.23
16641	2.638296e-003	0.4984	0.15	3.75
66049	2.467867e-003	0.0963	0.04	15.48

Table 12.2 Non-stationary interpolation at N equally spaced points in $[0,1]^2$ ($\varepsilon = 0.7$ fixed) with Wendland's function $\varphi(r) = (1-r)_+^4(4r+1)$.

N	RMS-error	rate	time
9	1.562729e-001		0.03
25	2.807706e-002	2.4766	0.04
81	4.853006e-003	2.5324	0.12
289	2.006041e-004	4.5965	0.45
1089	1.288000e-005	3.9611	2.75
4225	1.382497e-006	3.2198	47.92

comparison between the entries in Table 12.1 and Table 12.2 is not a straightforward one since we used the (dense) code Program 2.1 to do the experiments for Table 12.2 since there is no sparseness to be exploited and the *kd*-trees actually introduce additional overhead.

The interplay between computational efficiency and non-convergence in the stationary case and convergence and computational inefficiency in the non-stationary case is again a *trade-off principle* similar to the interplay between accuracy and ill-conditioning for globally supported RBFs (*c.f.* Chapter 2). These trade-off principles were explained theoretically as well as illustrated with numerical experiments in [Schaback (1997b)], and we will consider them in Chapter 16.

For comparison purposes we repeat the experiments with the oscillatory basic function

$$\varphi(r) = \sigma_2(r) = (1-r)_+^6\left(3 + 18r + 3r^2 - 192r^3\right),$$

which is also C^4 smooth and strictly positive definite and radial on $\mathbb{R}^s$ for $s \leq 3$ (see Table 11.5). The results are listed in Table 12.3 for the stationary case and in Table 12.4 for the non-stationary case. Note that the function is implemented as

```
rbf = @(e,r) -r.^6.*(168*spones(r)-552*r+573*r.^2-192*r.^3);
```

Table 12.3 Stationary interpolation at N equally spaced points in $[0,1]^2$ (constant 25 points in support) with the oscillatory function $\varphi(r) = (1-r)_+^6 \left(3 + 18r + 3r^2 - 192r^3\right)$.

N	RMS-error	rate	% nonzero	time
9	1.655969e-001		100	0.28
25	3.941226e-002	2.0710	57.8	0.34
81	2.978973e-002	0.4038	23.2	0.36
289	2.914215e-002	0.0317	7.47	0.42
1089	3.063424e-002	−0.0720	2.13	0.64
4225	3.094308e-002	−0.0145	0.57	1.31
16641	3.089882e-002	0.0021	0.15	4.13
66049	3.086639e-002	0.0015	0.04	16.81

Table 12.4 Non-stationary interpolation at N equally spaced points in $[0,1]^2$ ($\varepsilon = 0.7$ fixed) with the oscillatory function $\varphi(r) = (1-r)_+^6 \left(3 + 18r + 3r^2 - 192r^3\right)$.

N	RMS-error	rate	time
9	1.655969e-001		0.03
25	3.097850e-002	2.4183	0.06
81	4.612941e-003	2.7475	0.20
289	1.305297e-004	5.1432	0.72
1089	4.780575e-006	4.7711	4.06
4225	2.687479e-007	4.1529	55.09

in the sparse setting and as

```
rbf = @(e,r) max(1-e*r,0).^6.*(3+18*e*r+3*(e*r).^2-192*(e*r).^3);
```

for the dense code.

While the performance of the oscillatory functions for the stationary experiment is even more disappointing than that of Wendland's functions, the situation is reversed in the non-stationary case. In fact, the errors obtained with the oscillatory basis functions are almost as good as those achieved with "optimally" scaled Gaussians (*c.f.* Table 2.2).

In order to overcome the problems due to the trade-off principle that are apparent in both the stationary and non-stationary approach to interpolation with compactly supported radial functions we will later consider using a multilevel stationary scheme (see Chapter 32).

Chapter 13

Reproducing Kernel Hilbert Spaces and Native Spaces for Strictly Positive Definite Functions

In the next few chapters we will present some of the theoretical work on error bounds for approximation and interpolation with radial basis functions. Since the discussion for strictly positive definite functions will already be technical enough, we focus on this case, and only mention a few results for the conditionally positive definite case. The following discussion follows mostly the presentation in [Wendland (2005a)] where the interested reader can find many more details.

13.1 Reproducing Kernel Hilbert Spaces

Our first set of error bounds will come rather naturally once we associate with each (strictly positive definite) radial basic function a certain space of functions called its *native space*. We will then be able to establish a connection to reproducing kernel Hilbert spaces, which in turn will give us the desired error bounds as well as certain optimality results for radial basis function interpolation (see Chapter 18).

Reproducing kernels are a classical concept in analysis introduced by Nachman Aronszajn (see [Aronszajn (1950)]). We begin with

Definition 13.1. Let $\mathcal{H}$ be a real Hilbert space of functions $f : \Omega(\subseteq \mathbb{R}^s) \to \mathbb{R}$ with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. A function $K : \Omega \times \Omega \to \mathbb{R}$ is called *reproducing kernel for* $\mathcal{H}$ if

(1) $K(\cdot, \boldsymbol{x}) \in \mathcal{H}$ for all $\boldsymbol{x} \in \Omega$,
(2) $f(\boldsymbol{x}) = \langle f, K(\cdot, \boldsymbol{x}) \rangle_{\mathcal{H}}$ for all $f \in \mathcal{H}$ and all $\boldsymbol{x} \in \Omega$.

The name *reproducing kernel* is inspired by the reproducing property (2) in Definition 13.1. It is known that the reproducing kernel of a Hilbert space is unique, and that existence of a reproducing kernel is equivalent to the fact that the point evaluation functionals $\delta_{\boldsymbol{x}}$ are bounded linear functionals on Ω, *i.e.*, there exists a positive constant $M = M_{\boldsymbol{x}}$ such that

$$|\delta_{\boldsymbol{x}} f| = |f(\boldsymbol{x})| \leq M \|f\|_{\mathcal{H}}$$

for all $f \in \mathcal{H}$ and all $\boldsymbol{x} \in \Omega$. This latter fact is due to the Riesz representation theorem.

Other properties of reproducing kernels are given by

Theorem 13.1. *Suppose $\mathcal{H}$ is a Hilbert space of functions $f : \Omega \to \mathbb{R}$ with reproducing kernel K. Then we have*

(1) $K(\boldsymbol{x}, \boldsymbol{y}) = \langle K(\cdot, \boldsymbol{y}), K(\cdot, \boldsymbol{x}) \rangle_{\mathcal{H}}$ *for* $\boldsymbol{x}, \boldsymbol{y} \in \Omega$.
(2) $K(\boldsymbol{x}, \boldsymbol{y}) = K(\boldsymbol{y}, \boldsymbol{x})$ *for* $\boldsymbol{x}, \boldsymbol{y} \in \Omega$.
(3) *Convergence in Hilbert space norm implies pointwise convergence, i.e., if we have* $\|f - f_n\|_{\mathcal{H}} \to 0$ *for* $n \to \infty$ *then* $|f(\boldsymbol{x}) - f_n(\boldsymbol{x})| \to 0$ *for all* $\boldsymbol{x} \in \Omega$.

Proof. By Property (1) of Definition 13.1 $K(\cdot, \boldsymbol{y}) \in \mathcal{H}$ for every $\boldsymbol{y} \in \Omega$. Then the reproducing property (2) of the definition gives us

$$K(\boldsymbol{x}, \boldsymbol{y}) = \langle K(\cdot, \boldsymbol{y}), K(\cdot, \boldsymbol{x}) \rangle_{\mathcal{H}}$$

for all $\boldsymbol{x}, \boldsymbol{y} \in \Omega$. This establishes (1). Property (2) follows from (1) by the symmetry of the Hilbert space inner product. For (3) we use the reproducing property of K along with the Cauchy-Schwarz inequality:

$$|f(\boldsymbol{x}) - f_n(\boldsymbol{x})| = |\langle f - f_n, K(\cdot, \boldsymbol{x}) \rangle_{\mathcal{H}}| \leq \|f - f_n\|_{\mathcal{H}} \|K(\cdot, \boldsymbol{x})\|_{\mathcal{H}}. \qquad \square$$

Now it is interesting for us that the reproducing kernel K is known to be positive definite. Here we use a slight generalization of the notion of a positive definite function to a positive definite kernel. Essentially, we replace $\Phi(\boldsymbol{x}_j - \boldsymbol{x}_k)$ in Definition 3.2 by $K(\boldsymbol{x}_j, \boldsymbol{x}_k)$. At this point we remind the reader that the space of bounded linear functionals on $\mathcal{H}$ is known as its *dual*, and denoted by $\mathcal{H}^*$.

Theorem 13.2. *Suppose $\mathcal{H}$ is a reproducing kernel Hilbert function space with reproducing kernel $K : \Omega \times \Omega \to \mathbb{R}$. Then K is positive definite. Moreover, K is strictly positive definite if and only if the point evaluation functionals $\delta_{\boldsymbol{x}}$ are linearly independent in $\mathcal{H}^*$.*

Proof. Since the kernel is real-valued we can restrict ourselves to a quadratic form with real coefficients. For distinct points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ and nonzero $\boldsymbol{c} \in \mathbb{R}^N$ we have

$$\begin{aligned}
\sum_{j=1}^{N} \sum_{k=1}^{N} c_j c_k K(\boldsymbol{x}_j, \boldsymbol{x}_k) &= \sum_{j=1}^{N} \sum_{k=1}^{N} c_j c_k \langle K(\cdot, \boldsymbol{x}_j), K(\cdot, \boldsymbol{x}_k) \rangle_{\mathcal{H}} \\
&= \langle \sum_{j=1}^{N} c_j K(\cdot, \boldsymbol{x}_j), \sum_{k=1}^{n} c_k K(\cdot, \boldsymbol{x}_k) \rangle_{\mathcal{H}} \\
&= \| \sum_{j=1}^{N} c_j K(\cdot, \boldsymbol{x}_j) \|_{\mathcal{H}}^2 \geq 0.
\end{aligned}$$

Thus K is positive definite. To establish the second claim we assume K is not strictly positive definite and show that the point evaluation functionals must be

linearly dependent. If K is not strictly positive definite then there exist distinct points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ and nonzero coefficients c_j such that

$$\sum_{j=1}^{N} \sum_{k=1}^{N} c_j c_k K(\boldsymbol{x}_j, \boldsymbol{x}_k) = 0.$$

The first part of the proof therefore implies

$$\sum_{j=1}^{N} c_j K(\cdot, \boldsymbol{x}_j) = 0.$$

Now we take the Hilbert space inner product with an arbitrary function $f \in \mathcal{H}$ and use the reproducing property of K to obtain

$$\begin{aligned} 0 &= \langle f, \sum_{j=1}^{N} c_j K(\cdot, \boldsymbol{x}_j) \rangle_{\mathcal{H}} \\ &= \sum_{j=1}^{N} c_j \langle f, K(\cdot, \boldsymbol{x}_j) \rangle_{\mathcal{H}} \\ &= \sum_{j=1}^{N} c_j f(\boldsymbol{x}_j) \\ &= \sum_{j=1}^{N} c_j \delta_{\boldsymbol{x}_j}(f). \end{aligned}$$

This, however, implies the linear dependence of the point evaluation functionals $\delta_{\boldsymbol{x}_j}(f) = f(\boldsymbol{x}_j)$, $j = 1, \ldots, N$, since the coefficients c_j were assumed to be not all zero. An analogous argument can be used to establish the converse. □

This theorem provides one direction of the connection between strictly positive definite functions and reproducing kernels. However, we are also interested in the other direction. Since the RBFs we have built our interpolation methods from are strictly positive definite functions, we want to know how to construct a reproducing kernel Hilbert space associated with those strictly positive definite basic functions.

13.2 Native Spaces for Strictly Positive Definite Functions

In this section we will show that every strictly positive definite radial basic function can indeed be associated with a reproducing kernel Hilbert space — its *native space.*

First, we note that Definition 13.1 tells us that $\mathcal{H}$ contains all functions of the form

$$f = \sum_{j=1}^{N} c_j K(\cdot, \boldsymbol{x}_j)$$

provided $\boldsymbol{x}_j \in \Omega$. As a consequence of Theorem 13.1 we have that

$$\begin{aligned}\|f\|_{\mathcal{H}}^2 = \langle f, f\rangle_{\mathcal{H}} &= \langle \sum_{j=1}^{N} c_j K(\cdot, \boldsymbol{x}_j), \sum_{k=1}^{N} c_k K(\cdot, \boldsymbol{x}_k)\rangle_{\mathcal{H}} \\ &= \sum_{j=1}^{N}\sum_{k=1}^{N} c_j c_k \langle K(\cdot, \boldsymbol{x}_j), K(\cdot, \boldsymbol{x}_k)\rangle_{\mathcal{H}} \\ &= \sum_{j=1}^{N}\sum_{k=1}^{N} c_j c_k K(\boldsymbol{x}_j, \boldsymbol{x}_k).\end{aligned}$$

Therefore, we *define* the (possibly infinite-dimensional) space

$$H_K(\Omega) = \operatorname{span}\{K(\cdot, \boldsymbol{y}) : \ \boldsymbol{y} \in \Omega\} \tag{13.1}$$

with an associated bilinear form $\langle \cdot, \cdot \rangle_K$ given by

$$\langle \sum_{j=1}^{N_K} c_j K(\cdot, \boldsymbol{x}_j), \sum_{k=1}^{N_K} d_k K(\cdot, \boldsymbol{y}_k)\rangle_K = \sum_{j=1}^{N_K}\sum_{k=1}^{N_K} c_j d_k K(\boldsymbol{x}_j, \boldsymbol{y}_k),$$

where $N_K = \infty$ is also allowed.

Theorem 13.3. *If $K : \Omega \times \Omega \to \mathbb{R}$ is a symmetric strictly positive definite kernel, then the bilinear form $\langle \cdot, \cdot \rangle_K$ defines an inner product on $H_K(\Omega)$. Furthermore, $H_K(\Omega)$ is a pre-Hilbert space with reproducing kernel K.*

Proof. $\langle \cdot, \cdot \rangle_K$ is obviously bilinear and symmetric. We just need to show that $\langle f, f\rangle_K > 0$ for nonzero $f \in H_K(\Omega)$. Any such f can be written in the form

$$f = \sum_{j=1}^{N_K} c_j K(\cdot, \boldsymbol{x}_j), \qquad \boldsymbol{x}_j \in \Omega.$$

Then

$$\langle f, f\rangle_K = \sum_{j=1}^{N_K}\sum_{k=1}^{N_K} c_j c_k K(\boldsymbol{x}_j, \boldsymbol{x}_k) > 0$$

since K is strictly positive definite. The reproducing property follows from

$$\langle f, K(\cdot, \boldsymbol{x})\rangle_K = \sum_{j=1}^{N_K} c_j K(\boldsymbol{x}, \boldsymbol{x}_j) = f(\boldsymbol{x}).$$

□

Since we just showed that $H_K(\Omega)$ is a pre-Hilbert space, *i.e.*, need not be complete, we now define the *native space* $\mathcal{N}_K(\Omega)$ of K to be the completion of $H_K(\Omega)$ with respect to the K-norm $\|\cdot\|_K$ so that $\|f\|_K = \|f\|_{\mathcal{N}_K(\Omega)}$ for all $f \in H_K(\Omega)$. The technical details concerned with this construction are discussed in [Wendland (2005a)].

In the special case when we are dealing with strictly positive definite (translation invariant) functions $\Phi(\boldsymbol{x} - \boldsymbol{y}) = K(\boldsymbol{x}, \boldsymbol{y})$ and when $\Omega = \mathbb{R}^s$ we get a characterization of native spaces in terms of Fourier transforms.

Theorem 13.4. *Suppose $\Phi \in C(\mathbb{R}^s) \cap L_1(\mathbb{R}^s)$ is a real-valued strictly positive definite function. Define*

$$\mathcal{G} = \{f \in L_2(\mathbb{R}^s) \cap C(\mathbb{R}^s) : \frac{\hat{f}}{\sqrt{\hat{\Phi}}} \in L_2(\mathbb{R}^s)\}$$

and eguip this space with the bilinear form

$$\langle f, g\rangle_{\mathcal{G}} = \frac{1}{\sqrt{(2\pi)^s}}\langle \frac{\hat{f}}{\sqrt{\hat{\Phi}}}, \frac{\hat{g}}{\sqrt{\hat{\Phi}}}\rangle_{L_2(\mathbb{R}^s)} = \frac{1}{\sqrt{(2\pi)^s}} \int_{\mathbb{R}^s} \frac{\hat{f}(\omega)\overline{\hat{g}(\omega)}}{\hat{\Phi}(\omega)} d\omega.$$

Then $\mathcal{G}$ is a real Hilbert space with inner product $\langle \cdot, \cdot \rangle_{\mathcal{G}}$ and reproducing kernel $\Phi(\cdot - \cdot)$. Hence, $\mathcal{G}$ is the native space of Φ on $\mathbb{R}^s$, i.e., $\mathcal{G} = \mathcal{N}_\Phi(\mathbb{R}^s)$ and both inner products coincide. In particular, every $f \in \mathcal{N}_\Phi(\mathbb{R}^s)$ can be recovered from its Fourier transform $\hat{f} \in L_1(\mathbb{R}^s) \cap L_2(\mathbb{R}^s)$.

Another characterization of the native space is given in terms of the eigenfunctions of a linear operator associated with the reproducing kernel. This operator, $T_\Phi : L_2(\Omega) \to L_2(\Omega)$, is given by

$$T_\Phi(v)(\boldsymbol{x}) = \int_\Omega \Phi(\boldsymbol{x}, \boldsymbol{y}) v(\boldsymbol{y}) d\boldsymbol{y}, \qquad v \in L_2(\Omega), \quad \boldsymbol{x} \in \Omega.$$

For the eigenvalues λ_k, $k = 1, 2, \ldots$, and eigenfunctions ϕ_k of this operator Mercer's theorem [Riesz and Sz.-Nagy (1955)] states

Theorem 13.5 (Mercer). *Let $\Phi(\cdot, \cdot)$ be a continuous positive definite kernel that satisfies*

$$\int_\Omega \Phi(\boldsymbol{x}, \boldsymbol{y}) v(\boldsymbol{x}) v(\boldsymbol{y}) d\boldsymbol{x} d\boldsymbol{y} \geq 0, \qquad \text{for all } v \in L_2(\Omega), \ \boldsymbol{x}, \boldsymbol{y} \in \Omega. \tag{13.2}$$

Then Φ can be represented by

$$\Phi(\boldsymbol{x}, \boldsymbol{y}) = \sum_{k=1}^{\infty} \lambda_k \phi_k(\boldsymbol{x}) \phi_k(\boldsymbol{y}), \tag{13.3}$$

where λ_k are the (non-negative) eigenvalues *and ϕ_k are the (L_2-orthonormal)* eigenfunctions *of T_Φ. Moreover, this representation is absolutely and uniformly convergent.*

We can interpret condition (13.2) as a type of *integral positive definiteness*. As usual, the eigenvalues and eigenfunctions satisfy $T_\Phi \phi_k = \lambda_k \phi_k$ or

$$\int_\Omega \Phi(\boldsymbol{x}, \boldsymbol{y}) \phi_k(\boldsymbol{y}) d\boldsymbol{y} = \lambda_k \phi_k(\boldsymbol{x}), \qquad k = 1, 2, \ldots.$$

In general, Mercer's theorem allows us to construct a reproducing kernel Hilbert space $\mathcal{H}$ by representing the functions in $\mathcal{H}$ as infinite linear combinations of the eigenfunctions, *i.e.*,

$$\mathcal{H} = \left\{ f : \ f = \sum_{k=1}^{\infty} c_k \phi_k \right\}.$$

It is clear that the kernel Φ itself is in $\mathcal{H}$ since it has the eigenfunction expansion (13.3). The inner product for $\mathcal{H}$ is given by

$$\langle f, g\rangle_{\mathcal{H}} = \langle \sum_{j=1}^{\infty} c_j \phi_j, \sum_{k=1}^{\infty} d_k \phi_k \rangle_{\mathcal{H}} = \sum_{k=1}^{\infty} \frac{c_k d_k}{\lambda_k},$$

where we used the $\mathcal{H}$-orthogonality

$$\langle \phi_j, \phi_k \rangle_{\mathcal{H}} = \frac{\delta_{jk}}{\sqrt{\lambda_j}\sqrt{\lambda_k}}$$

of the eigenfunctions.

We note that Φ is indeed the reproducing kernel of $\mathcal{H}$ since the eigenfunction expansion (13.3) of Φ and the orthogonality of the eigenfunctions imply

$$\begin{aligned}\langle f, \Phi(\cdot, \boldsymbol{x})\rangle_{\mathcal{H}} &= \langle \sum_{j=1}^{\infty} c_j \phi_j, \sum_{k=1}^{\infty} \lambda_k \phi_k \phi_k(\boldsymbol{x}) \rangle_{\mathcal{H}} \\ &= \sum_{k=1}^{\infty} \frac{c_k \lambda_k \phi_k(\boldsymbol{x})}{\lambda_k} \\ &= \sum_{k=1}^{\infty} c_k \phi_k(\boldsymbol{x}) = f(\boldsymbol{x}).\end{aligned}$$

Finally, one has (*c.f.* [Wendland (2005a)]) that the native space $\mathcal{N}_\Phi(\Omega)$ is given by

$$\mathcal{N}_\Phi(\Omega) = \left\{ f \in L_2(\Omega) : \sum_{k=1}^{\infty} \frac{1}{\lambda_k} |\langle f, \phi_k \rangle_{L_2(\Omega)}|^2 < \infty \right\}$$

and the native space inner product can be written as

$$\langle f, g\rangle_{\mathcal{N}_\Phi} = \sum_{k=1}^{\infty} \frac{1}{\lambda_k} \langle f, \phi_k \rangle_{L_2(\Omega)} \langle g, \phi_k \rangle_{L_2(\Omega)}, \qquad f, g \in \mathcal{N}_\Phi(\Omega).$$

Since $\mathcal{N}_\Phi(\Omega)$ is a subspace of $L_2(\Omega)$ this corresponds to the identification $c_k = \langle f, \phi_k \rangle_{L_2(\Omega)}$ of the generalized Fourier coefficients in the discussion above.

13.3 Examples of Native Spaces for Popular Radial Basic Functions

Theorem 13.4 shows that native spaces of translation invariant functions can be viewed as a generalization of standard *Sobolev spaces*. Indeed, for $m > s/2$ the Sobolev space W_2^m can be defined as (see, *e.g.*, [Adams (1975)])

$$W_2^m(\mathbb{R}^s) = \{ f \in L_2(\mathbb{R}^s) \cap C(\mathbb{R}^s) : \hat{f}(\cdot)(1 + \|\cdot\|_2^2)^{m/2} \in L_2(\mathbb{R}^s) \}. \tag{13.4}$$

One also frequently sees the definition

$$W_2^m(\Omega) = \{ f \in L_2(\Omega) \cap C(\Omega) : D^{\boldsymbol{\alpha}} f \in L_2(\Omega) \text{ for all } |\boldsymbol{\alpha}| \le m, \ \boldsymbol{\alpha} \in \mathbb{N}^s \}, \tag{13.5}$$

which applies whenever $\Omega \subset \mathbb{R}^s$ is a bounded domain. This interpretation will make clear the connection between the natives spaces of Sobolev splines and those of polyharmonic splines to be discussed below. The norm of $W_2^m(\mathbb{R}^s)$ is usually given by

$$\|f\|_{W_2^m(\mathbb{R}^s)} = \left\{ \sum_{|\alpha| \le m} \|D^\alpha f\|^2_{L_2(\mathbb{R}^s)} \right\}^{1/2}.$$

According to (13.4), any strictly positive definite function Φ whose Fourier transform decays only algebraically has a Sobolev space as its native space. In particular, the Matérn functions

$$\Phi_\beta(\boldsymbol{x}) = \frac{K_{\beta-\frac{s}{2}}(\|\boldsymbol{x}\|)\|\boldsymbol{x}\|^{\beta-\frac{s}{2}}}{2^{\beta-1}\Gamma(\beta)}, \qquad \beta > \frac{s}{2},$$

of Section 4.4 with Fourier transform

$$\hat{\Phi}_\beta(\boldsymbol{\omega}) = \left(1 + \|\boldsymbol{\omega}\|^2\right)^{-\beta}$$

can immediately be seen to have native space $\mathcal{N}_{\Phi_\beta}(\mathbb{R}^s) = W_2^\beta(\mathbb{R}^s)$ with $\beta > s/2$ (which is why some people refer to the Matérn functions as Sobolev splines).

Wendland's compactly supported functions $\Phi_{s,k} = \varphi_{s,k}(\|\cdot\|_2)$ of Chapter 11 can be shown to have native spaces $\mathcal{N}_{\Phi_{s,k}}(\mathbb{R}^s) = W_2^{s/2+k+1/2}(\mathbb{R}^s)$ (where the restriction $s \ge 3$ is required for the special case $k = 0$).

Native spaces for strictly conditionally positive definite functions can also be constructed. However, since this is more technical, we limited the discussion above to strictly positive definite functions, and refer the interested reader to the book [Wendland (2005a)] or the papers [Schaback (1999a); Schaback (2000a)]. With the extension of the theory to strictly conditionally positive definite functions the native spaces of the radial powers and thin plate (or surface) splines of Sections 8.2 and 8.3 can be shown to be the so-called *Beppo-Levi spaces* of order k

$$\mathrm{BL}_k(\mathbb{R}^s) = \{f \in C(\mathbb{R}^s) : \ D^\alpha f \in L_2(\mathbb{R}^s) \text{ for all } |\alpha| = k, \ \alpha \in \mathbb{N}^s\},$$

where D^α denotes a *generalized derivative* of order α (defined in the same spirit as the generalized Fourier transform, see Appendix B). In fact, the intersection of all Beppo-Levi spaces $\mathrm{BL}_k(\mathbb{R}^s)$ of order $k \le m$ yields the Sobolev space $W_2^m(\mathbb{R}^s)$. In the literature the Beppo-Levi spaces $\mathrm{BL}_k(\mathbb{R}^s)$ are sometimes referred to as *homogeneous Sobolev spaces of order k*. Alternatively, the Beppo-Levi spaces on $\mathbb{R}^s$ are defined as

$$\mathrm{BL}_k(\mathbb{R}^s) = \{f \in C(\mathbb{R}^s) : \ \hat{f}(\cdot)\|\cdot\|_2^m \in L_2(\mathbb{R}^s)\},$$

and the formulas given in Chapter 8 for the Fourier transforms of radial powers and thin plate splines show immediately that their native spaces are Beppo-Levi spaces. The semi-norm on BL_k is given by

$$|f|_{\mathrm{BL}_k} = \left\{ \sum_{|\alpha|=k} \frac{k!}{\alpha_1! \ldots \alpha_d!} \|D^\alpha f\|^2_{L_2(\mathbb{R}^s)} \right\}^{1/2}, \tag{13.6}$$

and its kernel is the polynomial space Π^s_{k-1}. For more details see [Wendland (2005a)]. Beppo-Levi spaces were already studied in the early papers [Duchon (1976); Duchon (1977); Duchon (1978); Duchon (1980)].

The native spaces for Gaussians and (inverse) multiquadrics are rather small. For example, according to Theorem 13.4, for Gaussians the Fourier transform of $f \in \mathcal{N}_\Phi(\Omega)$ must decay faster than the Fourier transform of the Gaussian (which is itself a Gaussian). It is known that, even though the native space of Gaussians is small, it does contain the important class of so-called *band-limited functions*, *i.e.*, functions whose Fourier transform is compactly supported. These functions play an important role in *sampling theory* where Shannon's famous sampling theorem [Shannon (1949)] states that any band-limited function can be completely recovered from its discrete samples provided the function is sampled at a sampling rate at least twice its bandwidth. The content of this theorem was already known much earlier (see [Whittaker (1915)]).

Theorem 13.6 (Shannon Sampling). *Suppose $f \in C(\mathbb{R}^s) \cap L_1(\mathbb{R}^s)$ such that its Fourier transform vanishes outside the cube $Q = \left[-\frac{1}{2}, \frac{1}{2}\right]^s$. Then f can be uniquely reconstructed from its values on $\mathbb{Z}^s$, i.e.,*

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{\xi} \in \mathbb{Z}^s} f(\boldsymbol{\xi}) \operatorname{sinc}(\boldsymbol{x} - \boldsymbol{\xi}), \qquad \boldsymbol{x} \in \mathbb{R}^s.$$

Here the sine function is defined for any $\boldsymbol{x} = (x_1, \ldots, x_s) \in \mathbb{R}^s$ as $\operatorname{sine} \boldsymbol{x} = \prod_{d=1}^s \frac{\sin(\pi x_d)}{\pi x_d}$. For more details on Shannon's sampling theorem see, *e.g.*, Chapter 29 in the book [Cheney and Light (1999)] or the paper [Unser (2000)].

Chapter 14

The Power Function and Native Space Error Estimates

14.1 Fill Distance and Approximation Orders

Our goal in this section is to provide error estimates for scattered data interpolation with strictly (conditionally) positive definite functions. As in the previous chapter we will provide most of the details for the strictly positive definite case, and only mention the extension to the conditionally positive definite case in the end. In their final form we will want our estimates to depend on some kind of measure of the data distribution. The measure that is usually used in approximation theory is the so-called *fill distance*

$$h = h_{\mathcal{X},\Omega} = \sup_{\boldsymbol{x}\in\Omega} \min_{\boldsymbol{x}_j\in\mathcal{X}} \|\boldsymbol{x} - \boldsymbol{x}_j\|_2$$

already introduced in (2.3) in Chapter 2. The fill distance indicates how well the data fill out the domain Ω, and it therefore denotes the radius of the largest empty ball that can be placed among the data locations. We will be interested in whether the error

$$\|f - \mathcal{P}_f^{(h)}\|_\infty$$

tends to zero as $h \to 0$, and if so, how fast. Here $\{\mathcal{P}^{(h)}\}_h$ presents a sequence of interpolation (or, more generally, projection) operators that vary with the fill distance h. For example, $\mathcal{P}^{(h)}$ could denote interpolation to data given at $(2^n+1)^s$, $n = 1, 2, \ldots$, equally spaced points in the unit cube in $\mathbb{R}^s$ (with $h = 2^{-n}$) as we used in some of our earlier examples. Of course, the definition of the fill distance also covers scattered data such as sets of Halton points. In fact, since Halton points are quasi-uniformly distributed (see Appendix A) we can assume $h \approx 2^{-n}$ for a set of $(2^n + 1)^s$ Halton points in $\mathbb{R}^s$. This explains the specific sizes of the point sets we used in earlier examples.

Since we want to employ the machinery of reproducing kernel Hilbert spaces presented in the previous chapter we will concentrate on error estimates for functions $f \in \mathcal{N}_\Phi$. In the next chapter we will also mention some more general estimates.

The term that is often used to measure the speed of convergence to zero is *approximation order*. We say that the approximation operator $\mathcal{P}^{(h)}$ has L_p*-approximation*

order k if

$$\|f - \mathcal{P}_f^{(h)}\|_p = \mathcal{O}(h^k) \quad \text{for } h \to 0.$$

Moreover, if we can also show that $\|f - \mathcal{P}_f^{(h)}\|_p \neq o(h^k)$, then $\mathcal{P}^{(h)}$ has *exact L_p-approximation order k*. We will concentrate mostly on the case $p = \infty$ (*i.e.*, pointwise estimates), but approximation order in other norms can also be studied.

In order to keep the following discussion as transparent as possible we will restrict ourselves to strictly positive definite functions. With (considerably) more technical details the following can also be formulated for strictly conditionally positive definite functions (see [Wendland (2005a)] for details).

14.2 Lagrange Form of the Interpolant and Cardinal Basis Functions

The key idea for the following discussion is to express the interpolant in *Lagrange form*, *i.e.*, using so-called *cardinal basis functions*. For radial basis function approximation this idea is due to [Wu and Schaback (1993)]. In the previous chapters we established that, for any strictly positive definite function Φ, the linear system

$$A\boldsymbol{c} = \boldsymbol{y}$$

with $A_{ij} = \Phi(\boldsymbol{x}_i - \boldsymbol{x}_j)$, $i, j = 1, \ldots, N$, $\boldsymbol{c} = [c_1, \ldots, c_N]^T$, and $\boldsymbol{y} = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)]^T$ has a unique solution. In the following we will consider the more general situation where Φ is a strictly positive definite kernel, *i.e.*, the entries of A are given by $A_{ij} = \Phi(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The uniqueness result holds in this case also.

In order to obtain the cardinal basis functions u_j^*, $j = 1, \ldots, N$, with the property $u_j^*(\boldsymbol{x}_i) = \delta_{ij}$, *i.e.*,

$$u_j^*(\boldsymbol{x}_i) = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j, \end{cases}$$

we consider the linear system

$$A\boldsymbol{u}^*(\boldsymbol{x}) = \boldsymbol{b}(\boldsymbol{x}), \tag{14.1}$$

where the matrix A is as above (and therefore invertible), $\boldsymbol{u}^* = [u_1^*, \ldots, u_N^*]^T$, and $\boldsymbol{b} = [\Phi(\cdot, \boldsymbol{x}_1), \ldots, \Phi(\cdot, \boldsymbol{x}_N)]^T$. Thus,

Theorem 14.1. *Suppose Φ is a strictly positive definite kernel on $\mathbb{R}^s$. Then, for any distinct points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, there exist functions $u_j^* \in \operatorname{span}\{\Phi(\cdot, \boldsymbol{x}_j), j = 1, \ldots, N\}$ such that $u_j^*(\boldsymbol{x}_i) = \delta_{ij}$.*

Therefore, we can write the interpolant $\mathcal{P}_f$ to f at $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ in the cardinal form

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{j=1}^{N} f(\boldsymbol{x}_j) u_j^*(\boldsymbol{x}), \qquad \boldsymbol{x} \in \mathbb{R}^s.$$

It is of interest to note that the cardinal functions do not depend on the data values of the interpolation problem. Once the data sites are fixed and the basic function is chosen with an appropriate shape parameter (whose optimal value will depend on the data), then the cardinal functions are determined by the linear system (14.1). We have plotted various cardinal functions based on the Gaussian basic function with shape parameter $\varepsilon = 5$ in Figures 14.1–14.3. The dependence on the data locations is clearly apparent when comparing the different data distributions (uniformly spaced in Figure 14.1, tensor-product Chebyshev in Figure 14.2, and Halton points in Figure 14.3). The data sets can be seen in Figure 14.5 below.

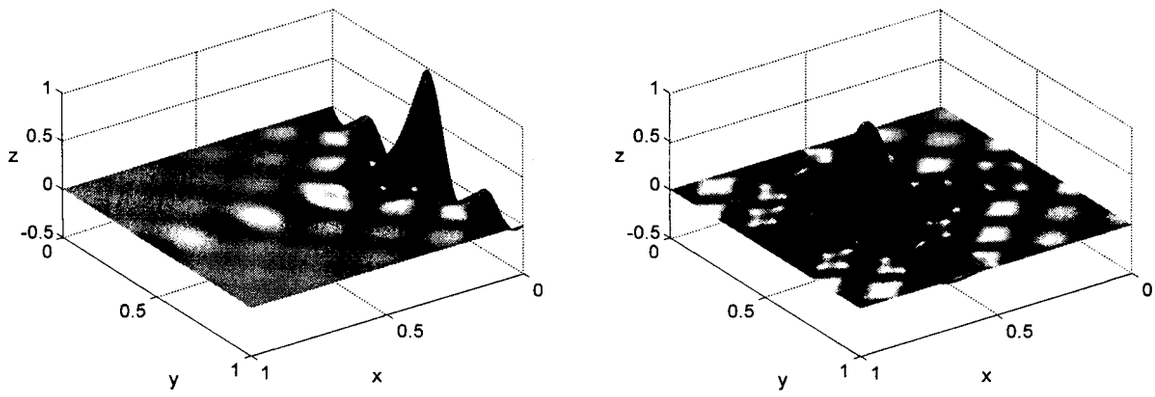


Fig. 14.1 Cardinal functions for Gaussian interpolation (with $\varepsilon = 5$) on 81 uniformly spaced points in $[0,1]^2$. Centered at an edge point (left) and at an interior point (right).

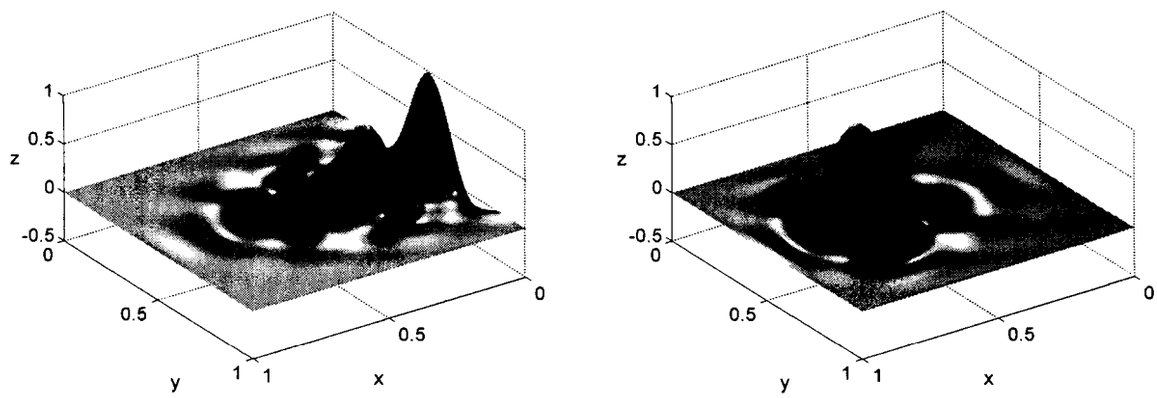


Fig. 14.2 Cardinal functions for Gaussian interpolation (with $\varepsilon = 5$) on 81 tensor-product Chebyshev points in $[0,1]^2$. Centered at an edge point (left) and at an interior point (right).

Basic functions that grow with increasing distance from the center point (such as multiquadrics) are sometimes criticized for being "counter-intuitive" for scattered data approximation. However, as Figure 14.4 shows, the cardinal functions are just as localized as those for the Gaussian basic functions, and thus the function space

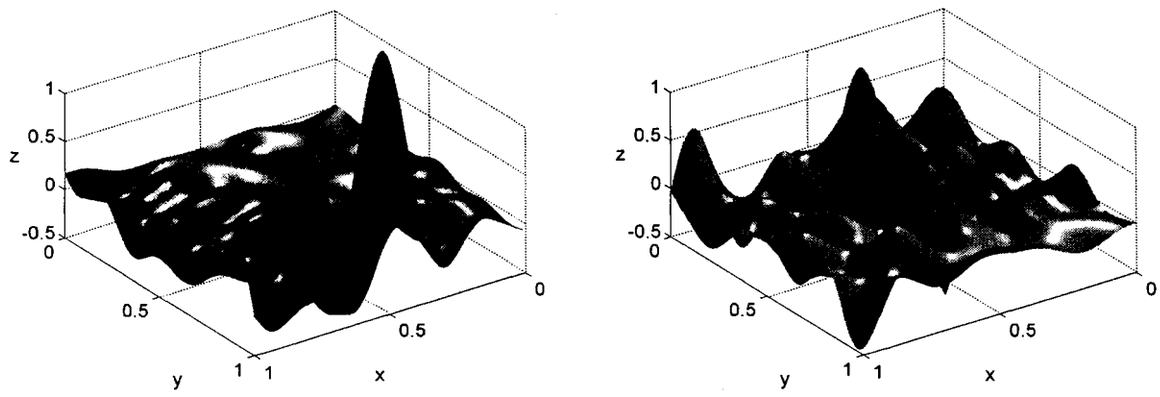


Fig. 14.3 Cardinal functions for Gaussian interpolation (with $\varepsilon = 5$) on 81 Halton points in $[0,1]^2$. Centered at an edge point (left) and at an interior point (right).

spanned by multiquadrics is a "good" local space.

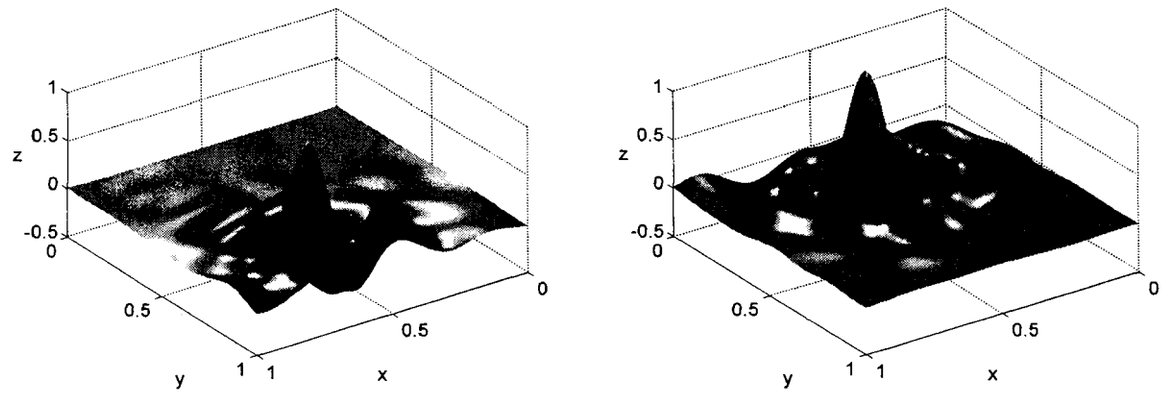


Fig. 14.4 Cardinal functions for multiquadric interpolation (with $\varepsilon = 5$) on 81 Halton points in $[0,1]^2$. Centered at an edge point (left) and at an interior point (right).

The MATLAB program `RBFCardinalFunction.m` used to produce the plots in Figures 14.1–14.3 is provided in Program 14.1. Note that we use the pseudo-inverse (via the MATLABcommand `pinv`) to stably compute the inverse of the interpolation matrix (see line 13 of Program 14.1). A specific cardinal function is then chosen in line 15.

Program 14.1. `RBFCardinalFunction.m`

```
% RBFCardinalFunction
% Computes and plots cardinal function for 2D RBF interpolation
% Calls on: DistanceMatrix
 1  rbf = @(e,r) exp(-(e*r).^2); ep = 5;
 2  N = 81; gridtype = 'u';
 3  neval = 80; M = neval^2;
```

```
   % Load data points
 4 name = sprintf('Data2D_%d%s',N,gridtype); load(name)
 5 ctrs = dsites;   % centers coincide with data sites
 6 grid = linspace(0,1,neval); [xe,ye] = meshgrid(grid);
 7 epoints = [xe(:) ye(:)];
   % Compute distance matrix between evaluation points and centers
 8 DM_eval = DistanceMatrix(epoints,ctrs);
   % Compute distance matrix between the data sites and centers
 9 DM_data = DistanceMatrix(dsites,ctrs);
   % Compute interpolation matrix
10 IM = rbf(ep,DM_data);
   % Compute evaluation matrix
11 EM = rbf(ep,DM_eval);
   % Compute cardinal functions at evaluation points
12 invIM = pinv(IM);
   % centered at datasite(50)
13 for j=1:M
14    cardvec = (invIM*EM(j,:)')';
15    cardfun(j) = cardvec(50);
16 end
17 figure
18 RBFplot = surf(xe,ye,reshape(cardfun,neval,neval));
19 set(RBFplot,'FaceColor','interp','EdgeColor','none')
20 colormap autumn; view([145 45]); camlight; lighting gouraud
```

14.3 The Power Function

Another important ingredient needed for our error estimates is the so-called *power function*. To this end, we consider a domain $\Omega \subseteq \mathbb{R}^s$. Then for any strictly positive definite kernel $\Phi \in C(\Omega \times \Omega)$, any set of distinct points $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subseteq \Omega$, and any vector $\boldsymbol{u} \in \mathbb{R}^N$, we define the quadratic form

$$Q(\boldsymbol{u}) = \Phi(\boldsymbol{x}, \boldsymbol{x}) - 2\sum_{j=1}^{N} u_j \Phi(\boldsymbol{x}, \boldsymbol{x}_j) + \sum_{i=1}^{N}\sum_{j=1}^{N} u_i u_j \Phi(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

Then

Definition 14.1. Suppose $\Omega \subseteq \mathbb{R}^s$ and $\Phi \in C(\Omega \times \Omega)$ is strictly positive definite on $\mathbb{R}^s$. For any distinct points $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subseteq \Omega$ the *power function* is defined by

$$[P_{\Phi,\mathcal{X}}(\boldsymbol{x})]^2 = Q(\boldsymbol{u}^*(\boldsymbol{x})),$$

where $\boldsymbol{u}^*$ is the vector of cardinal functions from Theorem 14.1.

Using the definition of the native space norm from the previous chapter we can rewrite the quadratic form $Q(\boldsymbol{u})$ as

$$\begin{aligned} Q(\boldsymbol{u}) &= \Phi(\boldsymbol{x},\boldsymbol{x}) - 2\sum_{j=1}^{N} u_j \Phi(\boldsymbol{x},\boldsymbol{x}_j) + \sum_{i=1}^{N}\sum_{j=1}^{N} u_i u_j \Phi(\boldsymbol{x}_i,\boldsymbol{x}_j) \\ &= \langle \Phi(\cdot,\boldsymbol{x}), \Phi(\cdot,\boldsymbol{x})\rangle_{\mathcal{N}_\Phi(\Omega)} - 2\sum_{j=1}^{N} u_j \langle \Phi(\cdot,\boldsymbol{x}), \Phi(\cdot,\boldsymbol{x}_j)\rangle_{\mathcal{N}_\Phi(\Omega)} \\ &\quad + \sum_{i=1}^{N}\sum_{j=1}^{N} u_i u_j \langle \Phi(\cdot,\boldsymbol{x}_i), \Phi(\cdot,\boldsymbol{x}_j)\rangle_{\mathcal{N}_\Phi(\Omega)} \\ &= \langle \Phi(\cdot,\boldsymbol{x}) - \sum_{j=1}^{N} u_j \Phi(\cdot,\boldsymbol{x}_j), \Phi(\cdot,\boldsymbol{x}) - \sum_{j=1}^{N} u_j \Phi(\cdot,\boldsymbol{x}_j)\rangle_{\mathcal{N}_\Phi(\Omega)} \\ &= \left\| \Phi(\cdot,\boldsymbol{x}) - \sum_{j=1}^{N} u_j \Phi(\cdot,\boldsymbol{x}_j) \right\|^2_{\mathcal{N}_\Phi(\Omega)}. \end{aligned} \tag{14.2}$$

The name *power function* was chosen by Schaback based on its connection to the power function of a statistical decision function (originally introduced in [Neyman and Pearson (1936)]). In the paper [Wu and Schaback (1993)] the power function was referred to as *kriging function.* This terminology comes from geostatistics (see, *e.g.*, [Myers (1992)]).

Using the linear system notation employed earlier, *i.e.*, $A_{ij} = \Phi(\boldsymbol{x}_i,\boldsymbol{x}_j)$, $i,j = 1,\ldots,N$, $\boldsymbol{u} = [u_1,\ldots,u_N]^T$, and $\boldsymbol{b} = [\Phi(\cdot,\boldsymbol{x}_1),\ldots,\Phi(\cdot,\boldsymbol{x}_N)]^T$, we note that we can also rewrite the quadratic form $Q(\boldsymbol{u})$ as

$$Q(\boldsymbol{u}) = \Phi(\boldsymbol{x},\boldsymbol{x}) - 2\boldsymbol{u}^T\boldsymbol{b}(\boldsymbol{x}) + \boldsymbol{u}^T A\boldsymbol{u}. \tag{14.3}$$

This suggests two alternative representations of the power function. Using the matrix-vector notation for $Q(\boldsymbol{u})$, the power function is given as

$$P_{\Phi,\mathcal{X}}(\boldsymbol{x}) = \sqrt{Q(\boldsymbol{u}^*(\boldsymbol{x}))} = \sqrt{\Phi(\boldsymbol{x},\boldsymbol{x}) - 2(\boldsymbol{u}^*(\boldsymbol{x}))^T\boldsymbol{b}(\boldsymbol{x}) + (\boldsymbol{u}^*(\boldsymbol{x}))^T A\boldsymbol{u}^*(\boldsymbol{x})}.$$

However, by the definition of the cardinal functions $A\boldsymbol{u}^*(\boldsymbol{x}) = \boldsymbol{b}(\boldsymbol{x})$, and therefore we have the two new variants

$$\begin{aligned} P_{\Phi,\mathcal{X}}(\boldsymbol{x}) &= \sqrt{\Phi(\boldsymbol{x},\boldsymbol{x}) - (\boldsymbol{u}^*(\boldsymbol{x}))^T\boldsymbol{b}(\boldsymbol{x})} \\ &= \sqrt{\Phi(\boldsymbol{x},\boldsymbol{x}) - (\boldsymbol{u}^*(\boldsymbol{x}))^T A\boldsymbol{u}^*(\boldsymbol{x})}. \end{aligned}$$

These formulas can be used for the numerical evaluation of the power function at $\boldsymbol{x}$. To this end one has to first find the value of the cardinal functions $\boldsymbol{u}^*(\boldsymbol{x})$ by solving the system $A\boldsymbol{u}^*(\boldsymbol{x}) = \boldsymbol{b}(\boldsymbol{x})$. This results in

$$P_{\Phi,\mathcal{X}}(\boldsymbol{x}) = \sqrt{\Phi(\boldsymbol{x},\boldsymbol{x}) - (\boldsymbol{b}(\boldsymbol{x}))^T A^{-1}\boldsymbol{b}(\boldsymbol{x})}. \tag{14.4}$$

Since A is a positive definite matrix whenever Φ is a strictly positive definite kernel we see that the power function satisfies the bounds

$$0 \leq P_{\Phi,\mathcal{X}}(\boldsymbol{x}) \leq \sqrt{\Phi(\boldsymbol{x},\boldsymbol{x})}.$$

Plots of the power function for the Gaussian with $\varepsilon = 6$ on three different point sets with $N = 81$ in the unit square are provided in Figure 14.5. The sets of data points are displayed on the left, while the plots of the power function are displayed in the right column. Dependence of the power function on the data locations is clearly visible. In fact, this connection was used in a recent paper [De Marchi *et al.* (2005)] to iteratively obtain an optimal set of data locations that are independent of the data values.

At this point the power function is mostly a theoretical tool that helps us better understand error estimates since we can decouple the effects due to the data function f from those due to the basic function Φ and the data locations $\mathcal{X}$ (see the following Theorem 14.2).

The power function is defined in an analogous way for strictly conditionally positive definite functions.

14.4 Generic Error Estimates for Functions in $\mathcal{N}_\Phi(\Omega)$

Now we can give a first generic error estimate.

Theorem 14.2. *Let $\Omega \subseteq \mathbb{R}^s$, $\Phi \in C(\Omega \times \Omega)$ be strictly positive definite on $\mathbb{R}^s$, and suppose that the points $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ are distinct. Denote the interpolant to $f \in \mathcal{N}_\Phi(\Omega)$ on $\mathcal{X}$ by $\mathcal{P}_f$. Then for every $\boldsymbol{x} \in \Omega$*

$$|f(\boldsymbol{x}) - \mathcal{P}_f(\boldsymbol{x})| \leq P_{\Phi,\mathcal{X}}(\boldsymbol{x}) \|f\|_{\mathcal{N}_\Phi(\Omega)}.$$

Proof. Since f is assumed to lie in the native space of Φ the reproducing property of Φ yields

$$f(\boldsymbol{x}) = \langle f, \Phi(\cdot, \boldsymbol{x}) \rangle_{\mathcal{N}_\Phi(\Omega)}.$$

We express the interpolant in its cardinal form and apply the reproducing property of Φ. This gives us

$$\begin{aligned} \mathcal{P}_f(\boldsymbol{x}) &= \sum_{j=1}^{N} f(\boldsymbol{x}_j) u_j^*(\boldsymbol{x}) \\ &= \sum_{j=1}^{N} u_j^*(\boldsymbol{x}) \langle f, \Phi(\cdot, \boldsymbol{x}_j) \rangle_{\mathcal{N}_\Phi(\Omega)} \\ &= \langle f, \sum_{j=1}^{N} u_j^*(\boldsymbol{x}) \Phi(\cdot, \boldsymbol{x}_j) \rangle_{\mathcal{N}_\Phi(\Omega)}. \end{aligned}$$

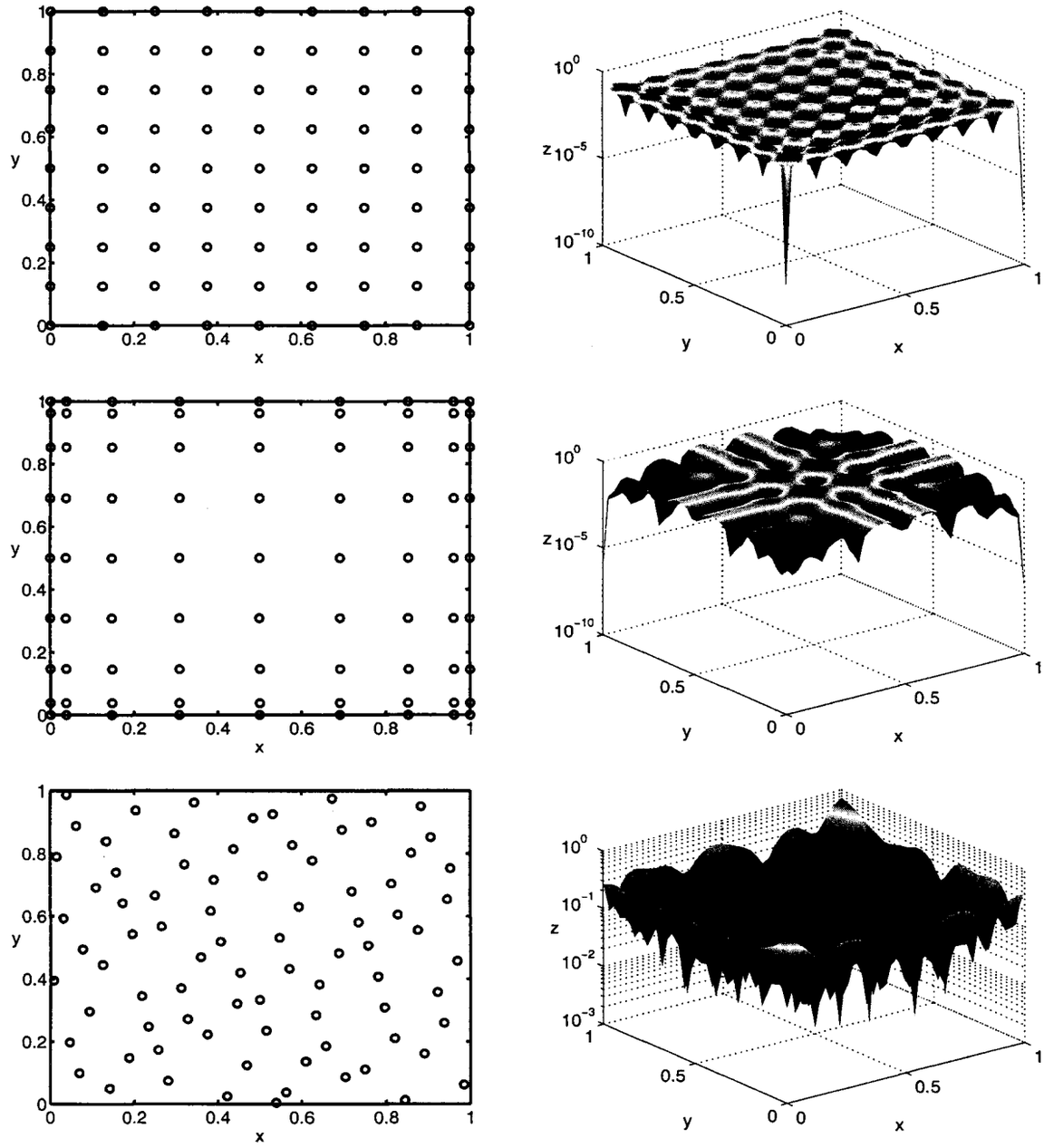


Fig. 14.5 Data sites and power function for Gaussian interpolant with $\varepsilon = 6$ based on $N = 81$ uniformly distributed points (top), tensor-product Chebyshev points (middle), and Halton points (bottom).

Now all that remains to be done is to combine the two formulas just derived and apply the Cauchy-Schwarz inequality. Thus,

$$|f(\boldsymbol{x}) - \mathcal{P}_f(\boldsymbol{x})| = \left| \langle f, \Phi(\cdot, \boldsymbol{x}) - \sum_{j=1}^{N} u_j^*(\boldsymbol{x}) \Phi(\cdot, \boldsymbol{x}_j) \rangle_{\mathcal{N}_\Phi(\Omega)} \right|$$

$$\leq \|f\|_{\mathcal{N}_\Phi(\Omega)} \left\| \Phi(\cdot, \boldsymbol{x}) - \sum_{j=1}^{N} u_j^*(\boldsymbol{x}) \Phi(\cdot, \boldsymbol{x}_j) \right\|_{\mathcal{N}_\Phi(\Omega)}$$
$$= \|f\|_{\mathcal{N}_\Phi(\Omega)} P_{\Phi,\mathcal{X}}(\boldsymbol{x}),$$

where we have applied (14.2) and the definition of the power function. □

One of the main benefits of Theorem 14.2 is that we are now able to estimate the interpolation error by considering two independent phenomena:

- the smoothness of the data (measured in terms of the native space norm of f — which is independent of the data locations, but does depend on Φ),
- and the contribution due to the use of the specific kernel (*i.e.*, basic function) Φ and the distribution of the data (measured in terms of the power function — independent of the actual data values).

This is analogous to the standard error estimate for polynomial interpolation cited in most numerical analysis texts. Note, however, that, for any given basic function Φ, a change of the shape parameter ε will have an effect on both terms in the error bound in Theorem 14.2 since the native space norm of f varies with ε.

14.5 Error Estimates in Terms of the Fill Distance

The next step is to refine this error bound by expressing the influence of the data locations in terms of the fill distance. And then, of course, the bound needs to be specialized to various choices of basic functions Φ.

The most common strategy for obtaining error bounds in numerical analysis is to take advantage of the polynomial precision of a method (at least locally), and then to apply a Taylor expansion. With this in mind we observe

Theorem 14.3. *Let $\Omega \subseteq \mathbb{R}^s$, and suppose $\Phi \in C(\Omega \times \Omega)$ is strictly positive definite on $\mathbb{R}^s$. Let $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ be a set of distinct points in Ω, and define the quadratic form $Q(\boldsymbol{u})$ as in (14.2). The minimum of $Q(\boldsymbol{u})$ is given for the vector $\boldsymbol{u} = \boldsymbol{u}^*(\boldsymbol{x})$ from Theorem 14.1, i.e.,*

$$Q(\boldsymbol{u}^*(\boldsymbol{x})) \leq Q(\boldsymbol{u}) \qquad \text{for all } \boldsymbol{u} \in \mathbb{R}^N.$$

Proof. We showed above (see (14.3)) that

$$Q(\boldsymbol{u}) = \Phi(\boldsymbol{x}, \boldsymbol{x}) - 2\boldsymbol{u}^T \boldsymbol{b}(\boldsymbol{x}) + \boldsymbol{u}^T A \boldsymbol{u}.$$

The minimum of this quadratic form is given by the solution of the linear system

$$A\boldsymbol{u} = \boldsymbol{b}(\boldsymbol{x}).$$

This, however, yields the cardinal functions $\boldsymbol{u} = \boldsymbol{u}^*(\boldsymbol{x})$. □

In the proof below we will use a special coefficient vector $\tilde{\boldsymbol{u}}$ which provides the polynomial precision desired for the proof of the refined error estimate. Its existence is guaranteed by the following theorem on *local polynomial reproduction* proved in [Wendland (2005a)]. This theorem requires the notion of a domain that satisfies an interior cone condition.

Definition 14.2. A region $\Omega \subseteq \mathbb{R}^s$ satisfies an *interior cone condition* if there exists an angle $\theta \in (0, \pi/2)$ and a radius $r > 0$ such that for every $\boldsymbol{x} \in \Omega$ there exists a unit vector $\boldsymbol{\xi}(\boldsymbol{x})$ such that the cone

$$C = \{\boldsymbol{x} + \lambda \boldsymbol{y} : \ \boldsymbol{y} \in \mathbb{R}^s, \|\boldsymbol{y}\|_2 = 1, \ \boldsymbol{y}^T \boldsymbol{\xi}(\boldsymbol{x}) \geq \cos\theta, \ \lambda \in [0, r]\}$$

is contained in Ω.

A consequence of the interior cone condition is the fact that a domain that satisfies this condition contains balls of a controllable radius. In particular, this will be important when bounding the remainder of the Taylor expansions below. For more details see [Wendland (2005a)].

Existence of an approximation scheme with local polynomial precision is guaranteed by

Theorem 14.4. *Suppose $\Omega \subseteq \mathbb{R}^s$ is bounded and satisfies an interior cone condition, and let ℓ be a non-negative integer. Then there exist positive constants h_0, c_1, and c_2 such that for all $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subseteq \Omega$ with $h_{\mathcal{X},\Omega} \leq h_0$ and every $\boldsymbol{x} \in \Omega$ there exist numbers $\tilde{u}_1(\boldsymbol{x}), \ldots, \tilde{u}_N(\boldsymbol{x})$ with*

(1) $\sum_{j=1}^{N} \tilde{u}_j(\boldsymbol{x}) p(\boldsymbol{x}_j) = p(\boldsymbol{x})$ *for all polynomials* $p \in \Pi_\ell^s$,

(2) $\sum_{j=1}^{N} |\tilde{u}_j(\boldsymbol{x})| \leq c_1$,

(3) $\tilde{u}_j(\boldsymbol{x}) = 0$ *if* $\|\boldsymbol{x} - \boldsymbol{x}_j\|_2 \geq c_2 h_{\mathcal{X},\Omega}$.

Property (1) yields the polynomial precision, and property (3) shows that the scheme is local. The bound in property (2) is essential for controlling the growth of error estimates and the quantity on the left-hand side of (2) is known as the *Lebesgue constant* at $\boldsymbol{x}$.

In the following theorem and its proof we will make repeated use of multi-index notation and multivariate Taylor expansions. For $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_s) \in \mathbb{N}_0^s$ with $|\boldsymbol{\beta}| = \sum_{i=1}^{s} \beta_i$ we define the differential operator $D^{\boldsymbol{\beta}}$ as

$$D^{\boldsymbol{\beta}} = \frac{\partial^{|\boldsymbol{\beta}|}}{(\partial x_1)^{\beta_1} \cdots (\partial x_s)^{\beta_s}},$$

and the notation $D_2^{\boldsymbol{\beta}} \Phi(\boldsymbol{w}, \cdot)$ used below indicates that the operator is applied to $\Phi(\boldsymbol{w}, \cdot)$ viewed as a function of the second variable.

The multivariate Taylor expansion of the function $\Phi(\boldsymbol{w}, \cdot)$ centered at $\boldsymbol{w}$ is given by

$$\Phi(\boldsymbol{w}, \boldsymbol{z}) = \sum_{|\boldsymbol{\beta}|<2k} \frac{D_2^{\boldsymbol{\beta}} \Phi(\boldsymbol{w}, \boldsymbol{w})}{\boldsymbol{\beta}!} (\boldsymbol{z} - \boldsymbol{w})^{\boldsymbol{\beta}} + R(\boldsymbol{w}, \boldsymbol{z})$$

with remainder

$$R(\boldsymbol{w}, \boldsymbol{z}) = \sum_{|\boldsymbol{\beta}|=2k} \frac{D_2^{\boldsymbol{\beta}} \Phi(\boldsymbol{w}, \boldsymbol{\xi}_{\boldsymbol{w},\boldsymbol{z}})}{\boldsymbol{\beta}!} (\boldsymbol{z} - \boldsymbol{w})^{\boldsymbol{\beta}},$$

where $\boldsymbol{\xi}_{\boldsymbol{w},\boldsymbol{z}}$ lies somewhere on the line segment connecting $\boldsymbol{w}$ and $\boldsymbol{z}$.

The generic error estimate of Theorem 14.2 can now be formulated in terms of the fill distance.

Theorem 14.5. *Suppose $\Omega \subseteq \mathbb{R}^s$ is bounded and satisfies an interior cone condition. Suppose $\Phi \in C^{2k}(\Omega \times \Omega)$ is symmetric and strictly positive definite. Denote the interpolant to $f \in \mathcal{N}_\Phi(\Omega)$ on the set $\mathcal{X}$ by $\mathcal{P}_f$. Then there exist positive constants h_0 and C (independent of $\boldsymbol{x}$, f and Φ) such that*

$$|f(\boldsymbol{x}) - \mathcal{P}_f(\boldsymbol{x})| \leq C h_{\mathcal{X},\Omega}^k \sqrt{C_\Phi(\boldsymbol{x})} \|f\|_{\mathcal{N}_\Phi(\Omega)},$$

provided $h_{\mathcal{X},\Omega} \leq h_0$. Here

$$C_\Phi(\boldsymbol{x}) = \max_{|\boldsymbol{\beta}|=2k} \max_{\boldsymbol{w},\boldsymbol{z} \in \Omega \cap B(\boldsymbol{x}, c_2 h_{\mathcal{X},\Omega})} |D_2^{\boldsymbol{\beta}} \Phi(\boldsymbol{w}, \boldsymbol{z})|$$

with $B(\boldsymbol{x}, c_2 h_{\mathcal{X},\Omega})$ denoting the ball of radius $c_2 h_{\mathcal{X},\Omega}$ centered at $\boldsymbol{x}$.

Proof. By Theorem 14.2 we know

$$|f(\boldsymbol{x}) - \mathcal{P}_f(\boldsymbol{x})| \leq P_{\Phi,\mathcal{X}}(\boldsymbol{x}) \|f\|_{\mathcal{N}_\Phi(\Omega)}.$$

Therefore, we now derive the bound

$$P_{\Phi,\mathcal{X}}(\boldsymbol{x}) \leq C h_{\mathcal{X},\Omega}^k \sqrt{C_\Phi(\boldsymbol{x})}$$

for the power function in terms of the fill distance.

We know that the power function is defined by

$$[P_{\Phi,\mathcal{X}}(\boldsymbol{x})]^2 = Q(\boldsymbol{u}^*(\boldsymbol{x})).$$

Moreover, we know from Theorem 14.3 that the quadratic form $Q(\boldsymbol{u})$ is minimized by $\boldsymbol{u} = \boldsymbol{u}^*(\boldsymbol{x})$. Therefore, any other coefficient vector $\boldsymbol{u}$ will yield an upper bound on the power function. We take $\boldsymbol{u} = \tilde{\boldsymbol{u}}(\boldsymbol{x})$ from Theorem 14.4 so that we are ensured to have polynomial precision of degree $\ell \geq 2k - 1$.

For this specific choice of coefficients we have

$$[P_{\Phi,\mathcal{X}}(\boldsymbol{x})]^2 \leq Q(\boldsymbol{u}) = \Phi(\boldsymbol{x}, \boldsymbol{x}) - 2 \sum_j u_j \Phi(\boldsymbol{x}, \boldsymbol{x}_j) + \sum_i \sum_j u_i u_j \Phi(\boldsymbol{x}_i, \boldsymbol{x}_j),$$

where the sums are over those indices j with $u_j \neq 0$. Now we apply the Taylor expansion centered at $\boldsymbol{x}$ to $\Phi(\boldsymbol{x}, \cdot)$ and centered at $\boldsymbol{x}_i$ to $\Phi(\boldsymbol{x}_i, \cdot)$, and evaluate both functions at $\boldsymbol{x}_j$. This yields

$$Q(\boldsymbol{u}) = \Phi(\boldsymbol{x}, \boldsymbol{x}) - 2\sum_j u_j \left[\sum_{|\boldsymbol{\beta}|<2k} \frac{D_2^{\boldsymbol{\beta}}\Phi(\boldsymbol{x}, \boldsymbol{x})}{\boldsymbol{\beta}!}(\boldsymbol{x}_j - \boldsymbol{x})^{\boldsymbol{\beta}} + R(\boldsymbol{x}, \boldsymbol{x}_j) \right] + \sum_i \sum_j u_i u_j \left[\sum_{|\boldsymbol{\beta}|<2k} \frac{D_2^{\boldsymbol{\beta}}\Phi(\boldsymbol{x}_i, \boldsymbol{x}_i)}{\boldsymbol{\beta}!}(\boldsymbol{x}_j - \boldsymbol{x}_i)^{\boldsymbol{\beta}} + R(\boldsymbol{x}_i, \boldsymbol{x}_j) \right].$$

Next, we identify $p(\boldsymbol{z}) = (\boldsymbol{z} - \boldsymbol{x})^{\boldsymbol{\beta}}$ so that $p(\boldsymbol{x}) = 0$ unless $\boldsymbol{\beta} = \boldsymbol{0}$. Therefore the polynomial precision property of the coefficient vector $\boldsymbol{u}$ simplifies this expression to

$$Q(\boldsymbol{u}) = \Phi(\boldsymbol{x}, \boldsymbol{x}) - 2\Phi(\boldsymbol{x}, \boldsymbol{x}) - 2\sum_j u_j R(\boldsymbol{x}, \boldsymbol{x}_j) + \sum_i u_i \sum_{|\boldsymbol{\beta}|<2k} \frac{D_2^{\boldsymbol{\beta}}\Phi(\boldsymbol{x}_i, \boldsymbol{x}_i)}{\boldsymbol{\beta}!}(\boldsymbol{x} - \boldsymbol{x}_i)^{\boldsymbol{\beta}} + \sum_i \sum_j u_i u_j R(\boldsymbol{x}_i, \boldsymbol{x}_j). \quad (14.5)$$

Now we can apply the Taylor expansion again and make the observation that

$$\sum_{|\boldsymbol{\beta}|<2k} \frac{D_2^{\boldsymbol{\beta}}\Phi(\boldsymbol{x}_i, \boldsymbol{x}_i)}{\boldsymbol{\beta}!}(\boldsymbol{x} - \boldsymbol{x}_i)^{\boldsymbol{\beta}} = \Phi(\boldsymbol{x}_i, \boldsymbol{x}) - R(\boldsymbol{x}_i, \boldsymbol{x}). \quad (14.6)$$

If we use (14.6) and rearrange the terms in (14.5) we get

$$Q(\boldsymbol{u}) = -\Phi(\boldsymbol{x}, \boldsymbol{x}) - \sum_j u_j \left[2R(\boldsymbol{x}, \boldsymbol{x}_j) - \sum_i u_i R(\boldsymbol{x}_i, \boldsymbol{x}_j) \right] + \sum_i u_i \left[\Phi(\boldsymbol{x}_i, \boldsymbol{x}) - R(\boldsymbol{x}_i, \boldsymbol{x}) \right]. \quad (14.7)$$

One final Taylor expansion we need is (using the symmetry of Φ)

$$\Phi(\boldsymbol{x}_i, \boldsymbol{x}) = \Phi(\boldsymbol{x}, \boldsymbol{x}_i) = \sum_{|\boldsymbol{\beta}|<2k} \frac{D_2^{\boldsymbol{\beta}}\Phi(\boldsymbol{x}, \boldsymbol{x})}{\boldsymbol{\beta}!}(\boldsymbol{x}_i - \boldsymbol{x})^{\boldsymbol{\beta}} + R(\boldsymbol{x}, \boldsymbol{x}_i). \quad (14.8)$$

If we insert (14.8) into (14.7) and once more take advantage of the polynomial precision property of the coefficient vector $\boldsymbol{u}$ we are left with

$$Q(\boldsymbol{u}) = -\sum_j u_j \left[R(\boldsymbol{x}, \boldsymbol{x}_j) + R(\boldsymbol{x}_j, \boldsymbol{x}) - \sum_i u_i R(\boldsymbol{x}_i, \boldsymbol{x}_j) \right].$$

Now Theorem 14.4 allows us to bound $\sum_j |u_j| \leq c_1$. Moreover, $\|\boldsymbol{x} - \boldsymbol{x}_j\|_2 \leq c_2 h_{\mathcal{X},\Omega}$ and $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 \leq 2c_2 h_{\mathcal{X},\Omega}$. Therefore, all three remainder terms can be bounded by an expression of the form $C h_{\mathcal{X},\Omega}^{2k} C_\Phi(\boldsymbol{x})$. Here we made use of the interior cone property of Ω enabling us to define the term $C_\Phi(\boldsymbol{x})$. Combining these bounds and taking the square root yields the stated bound for the power function. □

Theorem 14.5 says that interpolation with a C^{2k} smooth kernel Φ has approximation order k. Thus, for infinitely smooth strictly positive definite functions such as the Gaussians, Laguerre-Gaussians, Poisson radial functions, and the generalized inverse multiquadrics we see that the approximation order k is arbitrarily high. For strictly positive definite functions with limited smoothness such as the Matérn functions, the Whittaker radial functions, as well as all of the compactly supported functions, the approximation order is limited by the smoothness of the basic function.

The estimate in Theorem 14.5 is still generic. It does not fully account for the particular basic function Φ being used for the interpolation since the factor $C_\Phi(\boldsymbol{x})$ still depends on Φ. Moreover, we point out that the term $C_\Phi(\boldsymbol{x})$ may include a hidden dependence on $h_{\mathcal{X},\Omega}$. For most basic functions it will be possible to use $C_\Phi(\boldsymbol{x})$ to "squeeze out" additional powers of h. This is the reason for splitting the constant in front of the h-power into a generic C and a $C_\Phi(\boldsymbol{x})$.

The statement of Theorem 14.5 can be generalized for strictly conditionally positive definite functions and also to cover the error for approximating the derivatives of f by derivatives of $\mathcal{P}_f$. We state this general theorem without comment (*c.f.* [Wendland (2005a)] for details).

Theorem 14.6. *Suppose $\Omega \subseteq \mathbb{R}^s$ is open and bounded and satisfies an interior cone condition. Suppose $\Phi \in C^{2k}(\Omega \times \Omega)$ is symmetric and strictly conditionally positive definite of order m on $\mathbb{R}^s$. Denote the interpolant to $f \in \mathcal{N}_\Phi(\Omega)$ on the $(m-1)$-unisolvent set $\mathcal{X}$ by $\mathcal{P}_f$. Fix $\boldsymbol{\alpha} \in \mathbb{N}_0^s$ with $|\boldsymbol{\alpha}| \leq k$. Then there exist positive constants h_0 and C (independent of $\boldsymbol{x}$, f and Φ) such that*

$$|D^{\boldsymbol{\alpha}} f(\boldsymbol{x}) - D^{\boldsymbol{\alpha}} \mathcal{P}_f(\boldsymbol{x})| \leq C h_{\mathcal{X},\Omega}^{k-|\boldsymbol{\alpha}|} \sqrt{C_\Phi(\boldsymbol{x})} |f|_{\mathcal{N}_\Phi(\Omega)},$$

provided $h_{\mathcal{X},\Omega} \leq h_0$. Here

$$C_\Phi(\boldsymbol{x}) = \max_{\substack{\boldsymbol{\beta},\boldsymbol{\gamma} \in \mathbb{N}_0^s \\ |\boldsymbol{\beta}|+|\boldsymbol{\gamma}|=2k}} \max_{\boldsymbol{w},\boldsymbol{z} \in \Omega \cap B(\boldsymbol{x}, c_2 h_{\mathcal{X},\Omega})} |D_1^{\boldsymbol{\beta}} D_2^{\boldsymbol{\gamma}} \Phi(\boldsymbol{w}, \boldsymbol{z})|.$$

Note that for conditionally positive definite functions we have only a native space semi-norm instead of a norm.

Chapter 15

Refined and Improved Error Bounds

15.1 Native Space Error Bounds for Specific Basis Functions

For the first part of this chapter we discuss the non-stationary setting. The additional refinement of the error estimate of Theorem 14.6 for specific functions Φ is rather technical (for details see, *e.g.*, the book [Wendland (2005a)]). A large body of literature exists on this topic such as, *e.g.*, [Buhmann and Dyn (1991); Light (1996); Light and Wayne (1995); Light and Wayne (1998); Madych (1992); Madych and Nelson (1992); Narcowich and Ward (2004); Narcowich *et al.* (2003); Narcowich *et al.* (2005); Schaback (1995b); Schaback (1996); Wendland (1998); Wendland (1997); Wu and Schaback (1993); Yoon (2003)]). We now list some of the results that can be obtained.

15.1.1 *Infinitely Smooth Basis Functions*

As mentioned before, an application of Theorem 14.6 to infinitely smooth functions such as Gaussians or generalized (inverse) multiquadrics immediately yields arbitrarily high algebraic convergence rates, *i.e.*, for every $\ell \in \mathbb{N}$ and $|\boldsymbol{\alpha}| \leq \ell$ we have

$$|D^{\boldsymbol{\alpha}} f(\boldsymbol{x}) - D^{\boldsymbol{\alpha}} \mathcal{P}_f(\boldsymbol{x})| \leq C_\ell h^{\ell - |\boldsymbol{\alpha}|} |f|_{\mathcal{N}_\Phi(\Omega)}. \tag{15.1}$$

whenever $f \in \mathcal{N}_\Phi(\Omega)$. A considerable amount of work has gone into investigating the dependence of the constant C_ℓ on ℓ (see, *e.g.*, [Wendland (2001b)]).

Using different proof techniques (see, *e.g.*, [Madych and Nelson (1988)] or [Wendland (2005a)] for details) it is possible to derive more precise error bounds for Gaussians and (inverse) multiquadrics. The resulting theorem from [Wendland (2005a)] is

Theorem 15.1. *Let Ω be a cube in $\mathbb{R}^s$. Suppose that $\Phi = \varphi(\|\cdot\|)$ is a strictly conditionally positive definite radial function such that $\psi = \varphi(\sqrt{\cdot})$ satisfies $|\psi^{(\ell)}(r)| \leq \ell! M^\ell$ for all integers $\ell \geq \ell_0$ and all $r \geq 0$, where M is a fixed positive constant. Then there exists a constant c such that for any $f \in \mathcal{N}_\Phi(\Omega)$*

$$\|f - \mathcal{P}_f\|_{L_\infty(\Omega)} \leq e^{\frac{-c}{h_{\mathcal{X},\Omega}}} |f|_{\mathcal{N}_\Phi(\Omega)}, \tag{15.2}$$

for all data sites $\mathcal{X}$ with sufficiently small fill distance $h_{\mathcal{X},\Omega}$.

Moreover, if ψ satisfies even $|\psi^{(\ell)}(r)| \leq M^\ell$, then

$$\|f - \mathcal{P}_f\|_{L_\infty(\Omega)} \leq e^{\frac{-c|\log h_{\mathcal{X},\Omega}|}{h_{\mathcal{X},\Omega}}} \|f\|_{\mathcal{N}_\Phi(\Omega)} \tag{15.3}$$

provided $h_{\mathcal{X},\Omega}$ is sufficiently small.

Example 15.1. For Gaussians $\Phi(\boldsymbol{x}) = e^{-\varepsilon^2\|\boldsymbol{x}\|^2}$, $\varepsilon > 0$ fixed, we have $\psi(r) = e^{-\varepsilon^2 r}$, so that $\psi^{(\ell)}(r) = (-1)^\ell \varepsilon^{2\ell} e^{-\varepsilon^2 r}$ for $\ell \geq \ell_0 = 0$. Thus, $M = \varepsilon^2$, and the error bound (15.3) applies. This kind of exponential approximation order is usually referred to as *spectral* (or even super-spectral) approximation order. We emphasize that this nice property holds only in the non-stationary setting and for data functions f that are in the native space of the Gaussians such as band-limited functions.

Example 15.2. For generalized (inverse) multiquadrics $\Phi(\boldsymbol{x}) = (1+\|\boldsymbol{x}\|^2)^\beta$, $\beta < 0$, or $0 < \beta \notin \mathbb{N}$, we have $\psi(r) = (1+r)^\beta$. In this case one can show that $|\psi^\ell(r)| \leq \ell! M^\ell$ whenever $\ell \geq \lceil\beta\rceil$. Here $M = 1+|\beta+1|$. Therefore, the error estimate (15.2) applies, *i.e.*, in the non-stationary setting generalized (inverse) multiquadrics have spectral approximation order.

Example 15.3. For Laguerre-Gaussians $\Phi(\boldsymbol{x}) = L_n^{s/2}(\|\varepsilon\boldsymbol{x}\|^2)e^{-\varepsilon^2\|\boldsymbol{x}\|^2}$, $\varepsilon > 0$ fixed, we have $\psi(r) = L_n^{s/2}(\varepsilon^2 r)e^{-\varepsilon^2 r}$ and the derivatives $\psi^{(\ell)}$ will be bounded by $\psi^{(\ell)}(0) = p_n(\ell)\varepsilon^{2\ell}$, where p_n is a polynomial of degree n. Thus, the approximation power of Laguerre-Gaussians falls between (15.3) and (15.2) and Laguerre-Gaussians have at least spectral approximation power.

15.1.2 *Basis Functions with Finite Smoothness*

For functions with finite smoothness (such as the Matérn functions, radial powers, thin plate splines, and Wendland's compactly supported functions) it is possible to bound the constant $C_\Phi(\boldsymbol{x})$ by some additional powers of h, and thereby to improve the order predicted by Theorem 14.6. In particular, for C^k functions the factor $C_\Phi(\boldsymbol{x})$ can be expressed as

$$C_\Phi(\boldsymbol{x}) = \max_{|\boldsymbol{\beta}|=2k} \|D^{\boldsymbol{\beta}}\Phi\|_{L_\infty(B(0,2ch_{\mathcal{X},\Omega}))}$$

independent of $\boldsymbol{x}$ (see [Wendland (2005a)]). Therefore, this results in the following error estimates (see, *e.g.*, [Wendland (2005a)], or the much earlier [Wu and Schaback (1993)] where other proof techniques were used).

Example 15.4. For the Matérn functions $\Phi(\boldsymbol{x}) = \frac{K_{\beta-\frac{s}{2}}(\|\boldsymbol{x}\|)\|\boldsymbol{x}\|^{\beta-\frac{s}{2}}}{2^{\beta-1}\Gamma(\beta)}$, $\beta > \frac{s}{2}$, we get

$$|D^{\boldsymbol{\alpha}} f(\boldsymbol{x}) - D^{\boldsymbol{\alpha}}\mathcal{P}_f(\boldsymbol{x})| \leq C h_{\mathcal{X},\Omega}^{\beta-\frac{s}{2}-|\boldsymbol{\alpha}|} |f|_{\mathcal{N}_\Phi(\Omega)}. \tag{15.4}$$

provided $|\boldsymbol{\alpha}| \leq \beta - \lceil\frac{s+1}{2}\rceil$, $h_{\mathcal{X},\Omega}$ is sufficiently small, and $f \in \mathcal{N}_\Phi(\Omega)$.

Example 15.5. For the compactly supported Wendland functions $\Phi_{s,k}(\boldsymbol{x}) = \varphi_{s,k}(\|\boldsymbol{x}\|)$ this first refinement leads to

$$|D^{\boldsymbol{\alpha}} f(\boldsymbol{x}) - D^{\boldsymbol{\alpha}} \mathcal{P}_f(\boldsymbol{x})| \leq C h_{\mathcal{X},\Omega}^{k+\frac{1}{2}-|\boldsymbol{\alpha}|} \|f\|_{\mathcal{N}_\Phi(\Omega)}. \tag{15.5}$$

provided $|\boldsymbol{\alpha}| \leq k$, $h_{\mathcal{X},\Omega}$ is sufficiently small, and $f \in \mathcal{N}_\Phi(\Omega)$.

Example 15.6. For the radial powers $\Phi(\boldsymbol{x}) = (-1)^{\lceil \beta/2 \rceil} \|\boldsymbol{x}\|^\beta$, $0 < \beta \notin 2\mathbb{N}$, we get

$$|D^{\boldsymbol{\alpha}} f(\boldsymbol{x}) - D^{\boldsymbol{\alpha}} \mathcal{P}_f(\boldsymbol{x})| \leq C h_{\mathcal{X},\Omega}^{\frac{\beta}{2}-|\boldsymbol{\alpha}|} |f|_{\mathcal{N}_\Phi(\Omega)}. \tag{15.6}$$

provided $|\boldsymbol{\alpha}| \leq \frac{\lceil \beta \rceil - 1}{2}$, $h_{\mathcal{X},\Omega}$ is sufficiently small, and $f \in \mathcal{N}_\Phi(\Omega)$.

Example 15.7. For thin plate splines $\Phi(\boldsymbol{x}) = (-1)^{k+1} \|\boldsymbol{x}\|^{2k} \log \|\boldsymbol{x}\|$, we get

$$|D^{\boldsymbol{\alpha}} f(\boldsymbol{x}) - D^{\boldsymbol{\alpha}} \mathcal{P}_f(\boldsymbol{x})| \leq C h_{\mathcal{X},\Omega}^{k-|\boldsymbol{\alpha}|} |f|_{\mathcal{N}_\Phi(\Omega)}. \tag{15.7}$$

provided $|\alpha| \leq k-1$, $h_{\mathcal{X},\Omega}$ is sufficiently small, and $f \in \mathcal{N}_\Phi(\Omega)$.

15.2 Improvements for Native Space Error Bounds

Radial powers and thin plate splines can be interpreted as a generalization of univariate natural splines. Therefore, we know that the approximation order estimates obtained via the native space approach are not optimal. For example, for interpolation with univariate piecewise linear splines (*i.e.*, $\Phi(\boldsymbol{x}) = \|\boldsymbol{x}\|$ in $\boldsymbol{x} \in \mathbb{R}$) we know the approximation order to be $\mathcal{O}(h)$, whereas the estimate (15.6) yields only approximation order $\mathcal{O}(h^{1/2})$. Similarly, for thin plate splines $\Phi(\boldsymbol{x}) = \|\boldsymbol{x}\|^2 \log \|\boldsymbol{x}\|$ one would expect order $\mathcal{O}(h^2)$ in the case of pure function approximation. However, the estimate (15.7) yields only $\mathcal{O}(h)$. These two examples suggest that it should be possible to "double" the approximation orders obtained thus far.

One can improve the estimates for functions with finite smoothness (*i.e.*, Matérn functions, Wendland functions, radial powers, and thin plate splines) by either (or both) of the following two ideas:

- by requiring the data function f to be even smoother than what the native space prescribes, *i.e.*, by building certain boundary conditions into the native space;
- by using weaker norms to measure the error.

The idea to localize the data by adding boundary conditions was introduced in the paper [Light and Wayne (1998)]. This "trick" allows us to "square" the approximation order, and thus reach the expected approximation orders. The second idea can already be found in the early paper [Duchon (1978)].

After applying both of these techniques the final approximation order estimate for interpolation with the compactly supported functions $\Phi_{s,k}$ is (see [Wendland (1997)])

$$\|f - \mathcal{P}_f\|_{L_2(\Omega)} \leq C h^{2k+1+s} \|f\|_{W_2^{2k+1+s}(\mathbb{R}^s)}, \tag{15.8}$$

where f is assumed to lie in the subspace $W_2^{2k+1+s}(\mathbb{R}^s)$ of $\mathcal{N}_\Phi(\mathbb{R}^s) = W_2^{k+\frac{s+1}{2}}$. For example, for the popular basic function $\varphi_{3,1}(r) = (1-r)_+^4(4r+1)$ we have

$$\|f - \mathcal{P}_f\|_{L_2(\Omega)} \leq Ch^6 \|f\|_{W_2^6(\mathbb{R}^s)}.$$

Note that the numerical experiments in Table 12.2 produced RMS-convergence rates only as high as 4.5.

For radial powers and thin plate splines one obtains L_2-error estimates of order $\mathcal{O}(h^{\beta+s})$ and $\mathcal{O}(h^{2k+s})$, respectively. These estimates are optimal, *i.e.*, exact approximation orders, as shown in [Bejancu (1999)].

More work on improved error bounds can be found in, *e.g.*, [Johnson (2004a)] or [Schaback (1999b)].

15.3 Error Bounds for Functions Outside the Native Space

The error bounds mentioned so far were all valid under the assumption that the function f providing the data came from (a subspace of) the native space of the RBF employed in the interpolation. We now mention a few recent results that provide error bounds for interpolation of functions f *not* in the native space of the basic function. In particular, the case when f lies in some Sobolev space is of great interest. A rather general theorem was recently given in [Narcowich *et al.* (2005)]. In this theorem Narcowich, Ward and Wendland provide Sobolev bounds for functions with many zeros. However, since the interpolation error function is just such a function, these bounds have a direct application to our situation. We point out that this theorem again applies to the non-stationary setting.

Theorem 15.2. *Let k be a positive integer, $0 < \sigma \leq 1$, $1 \leq p < \infty$, $1 \leq q \leq \infty$ and let $\boldsymbol{\alpha}$ be a multi-index satisfying $k > |\boldsymbol{\alpha}| + s/p$ or, for $p = 1$, $k \geq |\boldsymbol{\alpha}| + s$. Let $\mathcal{X} \subset \Omega$ be a discrete set with fill distance $h = h_{\mathcal{X},\Omega}$ where Ω is a compact set with Lipschitz boundary which satisfies an interior cone condition. If $u \in W_p^{k+\sigma}(\Omega)$ satisfies $u|_{\mathcal{X}} = 0$, then*

$$|u|_{W_q^{|\boldsymbol{\alpha}|}(\Omega)} \leq ch^{k+\sigma-|\boldsymbol{\alpha}|-s(1/p-1/q)_+} |u|_{W_p^{k+\sigma}(\Omega)},$$

where c is a constant independent of u and h, and $(x)_+$ is the cutoff function.

Suppose we have an interpolation process $\mathcal{P} : W_p^{k+\sigma}(\Omega) \to V$ that maps Sobolev functions to a finite-dimensional subspace V of $W_p^{k+\sigma}(\Omega)$ with the additional property $|\mathcal{P}_f|_{W_p^{k+\sigma}(\Omega)} \leq |f|_{W_p^{k+\sigma}(\Omega)}$, then Theorem 15.2 immediately yields the error estimate

$$|f - \mathcal{P}_f|_{W_q^{|\boldsymbol{\alpha}|}(\Omega)} \leq ch^{k+\sigma-|\boldsymbol{\alpha}|-s(1/p-1/q)_+} |f|_{W_p^{k+\sigma}(\Omega)}.$$

The additional property $|\mathcal{P}_f|_{W_p^{k+\sigma}(\Omega)} \leq |f|_{W_p^{k+\sigma}(\Omega)}$ is certainly satisfied provided the native space of the basic function is a Sobolev space. Thus, Theorem 15.2 provides an alternative to the power function approach discussed in the previous

chapter if we base $\mathcal{P}$ on linear combinations of shifts of the basic function Φ. This new approach has the advantage that the term $C_\Phi(\boldsymbol{x})$ which may depend on both Φ and $\mathcal{X}$ no longer needs to be dealt with.

In particular, the authors of [Narcowich *et al.* (2005)] show that if the Fourier transform of Φ satisfies

$$c_1(1+\|\boldsymbol{\omega}\|_2^2)^{-\tau} \leq \hat{\Phi}(\boldsymbol{\omega}) \leq c_2(1+\|\boldsymbol{\omega}\|_2^2)^{-\tau}, \qquad \|\boldsymbol{\omega}\| \to \infty,\ \boldsymbol{\omega} \in \mathbb{R}^s, \tag{15.9}$$

then the above error estimate holds with $\tau = k + \sigma$ and $p = 2$ provided the fill distance is sufficiently small. Examples of basic functions with an appropriately decaying Fourier transform are provided by the families of Wendland or Matérn functions. In addition, Narcowich, Ward and Wendland show that analogous error bounds hold for radial powers and thin plate splines (whose native spaces are Beppo-Levi spaces).

For functions f *outside* the native space of a basic function Φ whose Fourier transform satisfies (15.9) Narcowich, Ward and Wendland prove

Theorem 15.3. *Let k and n be integers with $0 \leq n < k \leq \tau$ and $k > s/2$, and let $f \in C^k(\overline{\Omega})$. Also suppose that $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \Omega$ satisfies $\operatorname{diam}(\mathcal{X}) \leq 1$ with sufficiently small fill distance. Then for any $1 \leq q \leq \infty$ we have*

$$|f - \mathcal{P}_f|_{W_q^n(\Omega)} \leq c\rho_{\mathcal{X}}^{\tau-k} h^{k-n-s(1/2-1/q)_+} \|f\|_{C^k(\overline{\Omega})},$$

where $p_{\mathcal{X}} = \frac{h}{q_{\mathcal{X}}}$ is the mesh ratio for $\mathcal{X}$ and $q_{\mathcal{X}}$ is the separation distance $\frac{1}{2}\min_{i\neq j}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$.

We remind the reader that the fill distance corresponds to the radius of the largest possible empty ball that can be placed between the points in $\mathcal{X}$. The separation distance (*c.f.* Chapter 16), on the other hand, can be interpreted as the radius of the largest ball that can be placed around every point in $\mathcal{X}$ such that no two balls overlap. Thus, the mesh ratio is a measure of the non-uniformity of the distribution of the points in $\mathcal{X}$.

Similar results were obtained earlier in [Brownlee and Light (2004)] (for radial powers and thin plate splines only), and in [Yoon (2003)] (for shifted surface splines, see below).

Example 15.8. If we consider polyharmonic splines, then the decay condition (15.9) for the Fourier transform is satisfied with $\tau = 2\beta$ for thin plate splines and with $\tau = \beta$ for radial powers. If we take $k = \tau$, $n = 0$, and $q = \infty$ in Theorem 15.3 then we arrive at the bound

$$|f - \mathcal{P}_f|_{L_\infty} \leq ch^{2\beta - s/2}\|f\|_{C^{2\beta}(\overline{\Omega})}$$

for thin plate splines $\Phi(\boldsymbol{x}) = \|\boldsymbol{x}\|^{2\beta}\log(\|\boldsymbol{x}\|)$, and

$$|f - \mathcal{P}_f|_{L_\infty} \leq ch^{\beta - s/2}\|f\|_{C^{\beta}(\overline{\Omega})}$$

for radial powers $\Phi(\boldsymbol{x}) = \|\boldsymbol{x}\|^\beta$. These bounds immediately correspond to the "optimal" native space bounds obtained earlier only after the improvements discussed

in the previous subsection. For data functions f with less smoothness the approximation order is reduced accordingly.

Lower bounds on the approximation order for approximation by polyharmonic splines were recently provided in [Maiorov (2005)]. Maiorov studies for any $1 \leq p, q \leq \infty$ and $\beta/s > (1/p - 1/q)_+$ the error E of L_q-approximation of W_p^β functions by polyharmonic splines. More precisely,

$$E(W_p^\beta([0,1]^s), \mathcal{R}_N(\varphi_\beta, \beta), L_q([0,1]^s)) \geq cN^{-\beta/s},$$

where $\mathcal{R}_N(\varphi_\beta, \beta)$ denotes the linear space formed by all possible linear combinations of N polyharmonic (or thin-plate type) splines

$$\varphi_\beta(r) = \begin{cases} r^{2\beta-s} & \text{if } s \text{ is odd} \\ r^{2\beta-s} \log r & \text{if } s \text{ is even,} \end{cases} \qquad \beta > \frac{s}{2},$$

and multivariate polynomials of degree at most $\beta - \mathbf{1}$. Note that these bounds are in terms of the number N of data sites instead of the usual fill distance h.

For the special cases $p = q = \infty$ and $p = 2$, $1 \leq q \leq 2$ the above lower bound is shown to be asymptotically exact.

15.4 Error Bounds for Stationary Approximation

The stationary setting is a natural approach for use with local basis functions. The main motivation comes from the computational point of view. We are interested in maintaining sparse interpolation matrices as the density of the data increases. This can be achieved by scaling the basis functions proportional to the data density. In principle we can take any of our basic functions and apply a scaling of the variable, *i.e.*, we replace $\boldsymbol{x}$ by $\varepsilon\boldsymbol{x}$, $\varepsilon > 0$. As mentioned several times earlier, this scaling results in "peaked" or "narrow" basis functions for large values of ε, and "flat" basis functions for $\varepsilon \to 0$. We will now discuss what happens if we choose ε inversely proportional to the fill distance, *i.e.*,

$$\varepsilon_h = \frac{\varepsilon_0}{h_{\mathcal{X},\Omega}} \tag{15.10}$$

for some fixed base scale ε_0 and study the approximation error based on the RBF interpolant

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \varphi_{\varepsilon_h}(\|\boldsymbol{x} - \boldsymbol{x}_j\|),$$

where

$$\varphi_{\varepsilon_h} = \varphi(\varepsilon_h \cdot).$$

Example 15.9. A rather disappointing fact is that *Gaussians do not provide any positive approximation order*, *i.e.*, the approximation process is *saturated*. This was

studied by [Buhmann (1989a)] on infinite lattices. However, for quasi-interpolation the *approximate approximation* approach of Maz'ya shows that it is possible to choose ε_0 in such a way that the level at which the saturation occurs can be controlled (see, *e.g.*, [Maz'ya and Schmidt (1996)]). Therefore, Gaussians may very well be used for stationary interpolation provided an appropriate initial shape parameter is chosen. We will illustrate this behavior in the next chapter. The same kind of argument also applies to the Laguerre-Gaussians of Section 4.2.

Example 15.10. Basis functions with compact support such as the Wendland functions also do not provide any positive approximation order in the stationary case. This can be seen by looking at the power function for the scaled basic function $\Phi_{\varepsilon_h} = \Phi(\varepsilon_h \cdot)$ which is of the form $P_{\Phi_{\varepsilon_h},\mathcal{X}}(\boldsymbol{x}) = P_{\Phi,\mathcal{X}_{\varepsilon_h}}(\varepsilon_h \boldsymbol{x})$ where $\mathcal{X}_{\varepsilon_h} = \{\varepsilon_h \boldsymbol{x}_1, \ldots, \varepsilon_h \boldsymbol{x}_N\}$ denotes the scaled data set. Moreover, the fill distances of the sets $\mathcal{X}_{\varepsilon_h}$ and $\mathcal{X}$ satisfy $h_{\mathcal{X}_{\varepsilon_h},\Omega} = \varepsilon_h h_{\mathcal{X},\frac{\Omega}{\varepsilon_h}}$. Therefore, the power function (which can be bounded in terms of the fill distance, *c.f.* the proof of Theorem 14.5) satisfies

$$P_{\Phi_{\varepsilon_h},\mathcal{X}}(\boldsymbol{x}) \leq C \left(\varepsilon_h h_{\mathcal{X},\frac{\Omega}{\varepsilon_h}}\right)^{\alpha}$$

for some $\alpha > 0$. This, however, does not go to zero if ε_h is chosen as in (15.10).

If, on the other hand, we work in the approximate approximation regime, then we can obtain good convergence in many cases (see the next chapter for some numerical experiments).

Example 15.11. Stationary interpolation with (inverse) multiquadrics, radial powers and thin plate splines presents no difficulties. In fact, [Schaback (1995c)] shows that the native space error bound for thin plate splines and radial powers is invariant under a stationary scaling. Therefore, the non-stationary bound of Theorem 15.3 applies in the stationary case also. The advantage of scaling thin plate splines or radial powers comes from the added stability one can gain by preventing the separation distance from becoming too small (see Chapter 16 and the work of Iske on local polyharmonic spline approximation, *e.g.*, [Iske (2004)]).

Yoon provides error estimates for stationary approximation of rough functions (*i.e.*, functions that are not in the native space of the basic function) by so-called *shifted surface splines*. Shifted surface splines are of the form

$$\Phi(\boldsymbol{x}) = \begin{cases} (-\mathrm{I})^{\lceil \beta - s/2 \rceil} (1 + \|\boldsymbol{x}\|^2)^{\beta - s/2}, & s \text{ odd}, \\ (-1)^{\beta - s/2 + 1} (1 + \|\boldsymbol{x}\|^2)^{\beta - s/2} \log(1 + \|\boldsymbol{x}\|^2)^{1/2}, & s \text{ even}, \end{cases}$$

where $s/2 < \beta \in \mathbb{N}$. These functions include all of the (inverse) multiquadrics, radial powers and thin plate splines.

Yoon has the following theorem (see [Yoon (2003)] for the L_p case, and [Yoon (2001)] for L_∞ bounds only).

Theorem 15.4. *Let Φ be a shifted surface spline with shape parameter ε inversely proportional to the fill distance $h_{\mathcal{X},\Omega}$. Then there exists a positive constant C (independent of $\mathcal{X}$) such that for every f in the Sobolev space $W_2^\beta(\Omega) \cap W_\infty^\beta(\Omega)$ we*

have

$$\|f - \mathcal{P}_f\|_{L_p(\Omega)} \leq Ch^{\gamma_p}|f|_{W_2^\beta(\mathbb{R}^s)}, \qquad 1 \leq p \leq \infty,$$

with

$$\gamma_p = \min\{\beta, \beta - s/2 + s/p\}.$$

Furthermore, if $f \in W_2^\alpha(\Omega) \cap W_\infty^\alpha(\Omega)$ *with* $\max\{0, s/2 - s/p\} < \alpha < \beta$, *then*

$$\|f - \mathcal{P}_f\|_{L_p(\Omega)} = o(h^{\gamma_p - \beta + \alpha}).$$

Yoon's estimates address the question of how well the infinitely smooth (inverse) multiquadrics approximate functions that are less smooth than those in their native space. For example, Theorem 15.4 states that L_2-approximation to functions in $W_2^2(\Omega)$, $\Omega \subseteq \mathbb{R}^s$, by multiquadrics $\Phi_\varepsilon(\boldsymbol{x}) = \sqrt{1 + \|\varepsilon \boldsymbol{x}\|^2}$ is of the order $\mathcal{O}(h^2)$. However, we emphasize once more that this refers to stationary approximation of rough functions, *i.e.*, ε is scaled inversely proportional to the fill distance and f need not lie in the native space of Φ, whereas the spectral order given in (15.2) corresponds to approximation of functions in the native space in the non-stationary case with fixed ε.

For thin plate splines and radial powers the approximation orders in Theorem 15.4 are equivalent to those of Theorem 15.3 and the results of Brownlee and Light mentioned above. This is to be expected due to the invariance of these basic functions with respect to scaling.

The second part of Yoon's result is a step toward *exact approximation orders* as is the work of [Maiorov (2005)] and [Bejancu (1999)] mentioned above.

15.5 Convergence with Respect to the Shape Parameter

None of the error bounds discussed thus far have taken into account the possibility of varying the shape parameter ε for a fixed data set $\mathcal{X}$. However, in the literature the infinitely smooth basic functions such as the Gaussians and (inverse) multiquadrics are usually formulated including the shape parameter ε (or another parameter equivalent to it) and one may wonder how a change in this shape parameter affects the convergence properties of the RBF interpolant. In fact, quite a bit of work has been spent on the quest for the "optimal" shape parameter (see, *e.g.*, [Carlson and Foley (1991); Foley (1994); Hagan and Kansa (1994); Kansa and Carlson (1992); Rippa (1999); Tarwater (1985); Wertz *et al.* (2006)]).

Convergence of the infinitely smooth Gaussians and (inverse) multiquadrics with respect to the shape parameter was studied early on in [Madych (1991)]. Madych showed that for these basic functions there exists a positive constant $\lambda < 1$ such that

$$|f(\boldsymbol{x}) - \mathcal{P}_f(\boldsymbol{x})| \leq C\lambda^{1/(\varepsilon h_{\mathcal{X},\Omega})} \tag{15.11}$$

provided f is in the native space of Φ. This estimate shows that taking either the shape parameter ε or the fill distance $h_{\mathcal{X},\Omega}$ to zero results in exponential convergence.

15.6 Polynomial Interpolation as the Limit of RBF Interpolation

Recently, a number of authors (see, *e.g.*, [Driscoll and Fornberg (2002); Fornberg and Flyer (2005); Fornberg and Wright (2004); Larsson and Fornberg (2005); Schaback (2005); Schaback (2006b)]) have studied the limiting case as $\varepsilon \to 0$ of scaled radial basis function interpolation with infinitely smooth basic functions such as Gaussians and generalized (inverse) multiquadrics. It turns out that there is an interesting connection to polynomial interpolation.

In [Driscoll and Fornberg (2002)] univariate ($s = 1$) interpolation with ε-scaled infinitely smooth radial basic functions is studied. Driscoll and Fornberg show that the RBF interpolant

$$\mathcal{P}_f(x) = \sum_{j=1}^{N} c_j \varphi(\|\varepsilon(x - x_j)\|), \qquad x \in [a, b] \subset \mathbb{R},$$

to function values at N distinct data sites tends to the Lagrange interpolating polynomial of f as $\varepsilon \to 0$.

The multivariate case is more complicated. However, the limiting RBF interpolant (provided it exists) is given by a low-degree multivariate polynomial (see [Larsson and Fornberg (2005); Schaback (2005); Schaback (2006b)]). For example, if the data sites are located such that they guarantee a unique polynomial interpolant, then the limiting RBF interpolant is given by this polynomial. If polynomial interpolation is not unique, then the RBF limit depends on the choice of basic function. However, these statements require the RBFs to satisfy an (unproven) condition on certain coefficient matrices $A_{p,J}$. In [Larsson and Fornberg (2005)] the authors also provide an explanation for the error behavior for small values of the shape parameter, and for the existence of an optimal (positive) value of ε giving rise to a global minimum of the error function. For the special case of scaled Gaussians Schaback [Schaback (2005)] shows that the RBF interpolant converges to the de Boor and Ron least polynomial interpolant (see [de Boor and Ron (1990); de Boor and Ron (1992a)] and also [de Boor (2006)]) as $\varepsilon \to 0$.

In [Fornberg and Wright (2004)] the authors describe a so-called *Contour-Padé* algorithm that makes it possible (for data sets of relatively modest size) to compute the RBF interpolant for all values of the shape parameter ε including the limiting case $\varepsilon \to 0$. We present some numerical result obtained with Grady Wright's MATLAB toolbox in Chapter 17.

We will later exploit the connection between RBF and polynomial interpolants to design numerical solvers for partial differential equations.

Chapter 16

Stability and Trade-Off Principles

16.1 Stability and Conditioning of Radial Basis Function Interpolants

A standard criterion for measuring the numerical stability of an approximation method is its condition number. In particular, for radial basis function interpolation we need to look at the condition number of the interpolation matrix A with entries $A_{ij} = \Phi(\boldsymbol{x}_i - \boldsymbol{x}_j)$. For any matrix A its ℓ_2-*condition number* is given by

$$\text{cond}(A) = \|A\|_2 \|A^{-1}\|_2 = \frac{\sigma_{\max}}{\sigma_{\min}},$$

where $\sigma_{\max}$ and $\sigma_{\min}$ are the largest and smallest singular values of A. If we concentrate on positive definite matrices A, then the condition number of A can also take be computed as the ratio

$$\frac{\lambda_{\max}}{\lambda_{\min}}$$

of the largest and smallest eigenvalues.

What do we know about these eigenvalues? First, Gershgorin's theorem (see, *e.g.*, [Meyer (2000)]) says that

$$|\lambda_{\max} - A_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{N} |A_{ij}|$$

for some $i \in \{1, \ldots, N\}$. Therefore,

$$\lambda_{\max} \leq N \max_{i,j=1,\ldots,N} |A_{ij}| = N \max_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}} |\Phi(\boldsymbol{x}_i - \boldsymbol{x}_j)|,$$

which, since Φ is strictly positive definite, becomes

$$\lambda_{\max} \leq N\Phi(\mathbf{0})$$

by the properties of positive definite functions (Property (4) in Theorem 3.1). Now, as long as the data are not too wildly distributed, N will grow as $h_{\mathcal{X},\Omega}^{-s}$ which is acceptable. Therefore, the main work in establishing a bound for the condition number of A lies in finding lower bounds for $\lambda_{\min}$ (or correspondingly upper bounds

for the norm of the inverse $\|A^{-1}\|_2$). This is the subject of several papers by Ball, Narcowich, Sivakumar and Ward [Ball *et al.* (1992); Narcowich *et al.* (1994); Narcowich and Ward (1991a); Narcowich and Ward (1991b); Narcowich and Ward (1992)] who make use of a result from [Ball (1992)] on eigenvalues of distance matrices. Ball's result follows from the *Rayleigh quotient* (or the Courant-Fischer Theorem 9.5), which gives the smallest eigenvalue of a symmetric positive definite matrix as

$$\lambda_{\min} = \min_{\boldsymbol{c} \in \mathbb{R}^N \setminus \mathbf{0}} \frac{\boldsymbol{c}^T A \boldsymbol{c}}{\boldsymbol{c}^T \boldsymbol{c}}.$$

This can be used to prove the following bound for the norm of the inverse of A which covers also the case of conditional positive (negative) definiteness of order one.

Lemma 16.1. *Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ be distinct points in $\mathbb{R}^s$ and let $\Phi : \mathbb{R}^s \to \mathbb{R}$ be either strictly positive definite, or strictly conditionally negative definite of order one with $\Phi(\mathbf{0}) \leq 0$. Also, let A be the interpolation matrix with entries $A_{ij} = \Phi(\boldsymbol{x}_i - \boldsymbol{x}_j)$. If the inequality*

$$\sum_{i=1}^{N} \sum_{j=1}^{N} c_i c_j A_{ij} \geq \theta \|\boldsymbol{c}\|_2^2$$

is satisfied whenever the components of $\boldsymbol{c}$ satisfy $\sum_{j=1}^{N} c_j = 0$, then

$$\|A^{-1}\|_2 \leq \theta^{-1}.$$

Note that for positive definite matrices the Rayleigh quotient implies $\theta = \lambda_{\min}$ which shows why lower bounds on the smallest eigenvalue correspond to upper bounds on the norm of the inverse of A. In order to obtain the bound for conditionally negative definite matrices the Courant-Fischer theorem 9.5 needs to be employed.

The bound in Lemma 16.1 is too generic to give us any information for specific basic functions Φ. This extension was accomplished in some of the other papers mentioned above. Narcowich and Ward establish bounds on the norm of the inverse of A in terms of the *separation distance* of the data sites

$$q_{\mathcal{X}} = \frac{1}{2} \min_{i \neq j} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2.$$

We can picture $q_{\mathcal{X}}$ as the radius of the largest ball that can be placed around every point in $\mathcal{X}$ such that no two balls overlap (see Figure 16.1). The separation distance is sometimes also referred to as the *packing radius*. In our MATLAB code we can compute the separation distance via

```
qX = min(min(DM_data+eye(size(DM_data))))/2
```

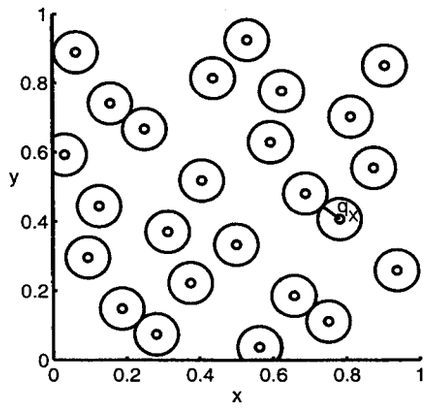


Fig. 16.1 The separation distance for $N = 25$ Halton points ($q_{\mathcal{X}} \approx 0.0597$).

where `DM_data` is the matrix of pairwise distances among the data sites $\mathcal{X}$. The identity matrix is added only to avoid counting the distance of a point $\boldsymbol{x}_j$ to itself as a potential minimum.

The derivation of these bounds is rather technical, and for details we refer to either the original papers by Narcowich, Ward and co-workers listed above, the more recent paper [Schaback (2002)], or the book [Wendland (2005a)]. By providing matching lower bounds for $\|A^{-1}\|_2$ (*i.e.*, *upper* bounds for $\lambda_{\min}$) Schaback showed that the upper bounds on the norm of the inverse obtained by Narcowich, Ward and others are near optimal (see [Schaback (1994b)]).

We now list several (lower) bounds for $\lambda_{\min}$ as derived in [Wendland (2005a)]. In the examples below the explicit (space-dependent) constants

$$M_s = 12\left(\frac{\pi\Gamma^2(\frac{s+2}{2})}{9}\right)^{1/(s+1)} \leq 6.38s \quad \text{and} \quad C_s = \frac{1}{2\Gamma(\frac{s+2}{2})}\left(\frac{M_s}{\sqrt{8}}\right)^s$$

are used. The upper bound for M_s can be obtained using Stirling's formula (see [Wendland (2005a)]).

Since the bounds in the literature for Gaussians and multiquadrics also include the influence of the shape parameter ε we present the basic functions in their scaled version here.

Example 16.1. For Gaussians $\Phi(\boldsymbol{x}) = e^{-\varepsilon^2\|\boldsymbol{x}\|^2}$ one obtains

$$\lambda_{\min} \geq C_s(\sqrt{2}\varepsilon)^{-s}e^{-40.71s^2/(q_{\mathcal{X}}\varepsilon)^2}q_{\mathcal{X}}^{-s}.$$

We see that, for a fixed shape parameter ε, the lower bound for $\lambda_{\min}$ goes exponentially to zero as the separation distance $q_{\mathcal{X}}$ decreases. Since we observed above that the condition number of the interpolation matrix A depends on the ratio of its largest and smallest eigenvalues and the growth of $\lambda_{\max}$ is of order N we see that the condition number grows exponentially with decreasing separation distance. This shows that, if one adds more interpolation points in order to improve the accuracy of the interpolant (within the same domain Ω), then the problem becomes

increasingly ill-conditioned. Of course one would always expect this to happen, but here the ill-conditioning grows primarily due to the decrease in the separation distance $q_{\mathcal{X}}$, and not to the increase in the number N of data points. We will come back to this observation when we discuss a possible change of basis in Section 34.4.

On the other hand, if one keeps the number of points (or at least the separation distance) fixed and instead decreases the value of ε, then the condition number of A suffers in almost the same exponential manner. Of course, an increase in ε can be used to improve the condition number of A (however, as we saw earlier, at the expense of accuracy of the fit).

Example 16.2. For scaled generalized (inverse) multiquadrics $\Phi(\boldsymbol{x}) = \left(1 + \|\varepsilon \boldsymbol{x}\|^2\right)^{\beta}$, $\beta \in \mathbb{R} \setminus \mathbb{N}_0$ one obtains

$$\lambda_{\min} \geq C(s, \beta, \varepsilon) q_{\mathcal{X}}^{\beta - \frac{s}{2} + \frac{1}{2}} e^{-2M_s/(q_{\mathcal{X}}\varepsilon)}$$

with another explicitly known constant $C(s, \beta, \varepsilon)$.

The same comments as in the previous example apply.

Example 16.3. For thin plate splines $\Phi(\boldsymbol{x}) = (-1)^{\beta+1} \|\boldsymbol{x}\|^{2\beta} \log \|\boldsymbol{x}\|$, $\beta \in \mathbb{N}$, one obtains

$$\lambda_{\min} \geq C_s c_\beta (2M_s)^{-s-2\beta} q_{\mathcal{X}}^{2\beta}$$

with another explicitly known constant c_β.

In this case the lower bound also goes to zero with decreasing separation distance. However the decay is only of polynomial order.

Example 16.4. For the radial powers $\Phi(\boldsymbol{x}) = (-1)^{\lceil \beta/2 \rceil} \|\boldsymbol{x}\|^{\beta}$, $0 < \beta \notin 2\mathbb{N}$, one obtains

$$\lambda_{\min} \geq C_s c_\beta (2M_s)^{-s-\beta} q_{\mathcal{X}}^{\beta}$$

with another explicitly known constant c_β (different from c_β in **Example** 3). Again, the decay is of polynomial order.

Example 16.5. For the compactly supported functions of Wendland $\Phi_{s,k}(\boldsymbol{x}) = \varphi_{s,k}(\|\boldsymbol{x}\|)$ one obtains

$$\lambda_{\min} \geq C(s, k) q_{\mathcal{X}}^{2k+1}$$

with a constant $C(s, k)$ depending on s and k. The lower bound goes to zero with the separation distance at a polynomial rate.

16.2 Trade-Off Principle I: Accuracy vs. Stability

The observations made in **Examples** 16.1 and 16.2 above set up the first *trade-off principle.* This principle states that if we use the standard approach to the RBF interpolation problem (*i.e.*, solution of the linear system (6.3)) then there is

a conflict between theoretically achievable accuracy and numerical stability. For example, the error bounds for non-stationary interpolation using infinitely smooth basis functions show that the error decreases (exponentially) as the fill distance decreases. For well-distributed data a decrease in the fill distance also implies a decrease of the separation distance. But now the condition estimates of the previous subsection imply that the condition number of A grows exponentially. This leads to numerical instabilities which make it virtually impossible to obtain the highly accurate results promised by the theoretical error bounds.

Similarly, if we use the shape parameter to (exponentially) increase accuracy as guaranteed by Madych's error bound (15.11), then the condition number again grows exponentially. This is to be expected since for small values of ε the basic functions more and more resemble a constant function, and therefore the rows (as well as columns) of the matrix A become more and more alike, so that the matrix becomes almost singular — even for well separated data sites.

In the literature this phenomenon has been referred to as *trade-off* or (*uncertainty*) *principle* (see, *e.g.*, the papers [Schaback (1995b); Schaback (1995c)]).

Schaback looked at the power function $P_{\Phi,\mathcal{X}}$ and showed that it can always be bounded from above by a function F_Φ depending on the fill distance. On the other hand, he showed that the Rayleigh quotient can always be bounded from below by a function G_Φ depending on the separation distance. Furthermore, Schaback showed that

$$G_\Phi(q_\mathcal{X}) \leq F_\Phi(h_{\mathcal{X},\Omega}),$$

and therefore, for well-distributed data (with $q_\mathcal{X} \approx h_{\mathcal{X},\Omega}$), a small error bound (*i.e.*, small $F_\Phi(h_{\mathcal{X},\Omega})$) will necessarily result in a small lower bound (*i.e.*, small $G_\Phi(q_\mathcal{X})$) for the Rayleigh quotient, and therefore for the smallest eigenvalue. This however implies a large condition number.

We have seen evidence of the first trade-off principle in various numerical experiments. This trade-off has led a number of people to search for an "optimal" value of the shape parameter, *i.e.*, a value that yields maximal accuracy, while still maintaining numerical stability.

In particular, multiquadrics have attracted the best part of this attention. For example, in his original work on (inverse) multiquadric interpolation in $\mathbb{R}^2$ Hardy [Hardy (1971)] suggested using $\varepsilon = 1/(0.815d)$, where $d = \frac{1}{N}\sum_{i=1}^{N} d_i$, and d_i is the distance from $\boldsymbol{x}_i$ to its nearest neighbor. Later, in [Franke (1982a)], one can find the recommended value $\varepsilon = \frac{0.8\sqrt{N}}{D}$, where D is the diameter of the smallest circle containing all data points. Another strategy for finding a good value for ε is based on the observation that such a value seems to be similar for multiquadrics and inverse multiquadrics (see [Foley (1994)]). Other studies were reported in [Carlson and Foley (1992); Carlson and Natarajan (1994)]. We will consider a more recent algorithm proposed in [Rippa (1999)] in the next chapter.

16.3 Trade-Off Principle II: Accuracy and Stability vs. Problem Size

More recently, Fornberg and co-workers have investigated the dependence of the stability on the values of the shape parameter ε in a series of papers (*e.g.*, [Driscoll and Fornberg (2002); Fornberg and Wright (2004); Larsson and Fornberg (2005); Platte and Driscoll (2005)]). They suggest a way to stably compute very accurate generalized (inverse) multiquadric and Gaussian interpolants with extreme values of $\varepsilon \to 0$ by using a complex Contour-Padé integration algorithm. Thus, this approach allows us to overcome the first trade-off principle mentioned in the previous section. However, there is another kind of trade-off associated with the Contour-Padé approach. Namely it is limited to only rather small data sets (roughly $N = 20$ for $s = 1$ and $N = 80$ for $s = 2$).

In spite of these limitations the Contour-Padé algorithm has been used to gain a number of theoretical insights such as the connection between RBF interpolation and polynomial interpolation mentioned in Section 15.6. We present some numerical experiments based on the Contour-Padé approach in the next chapter.

16.4 Trade-Off Principle III: Accuracy vs. Efficiency

There is also a trade-off principle for compactly supported functions. This was explained theoretically as well as illustrated with numerical experiments in [Schaback (1997b)]. The consequences are as follows. In the case of stationary interpolation, *i.e.*, if we scale the support size of the basis functions proportional to the fill distance $h_{\mathcal{X},\Omega}$, the "bandwidth" of the interpolation matrix A is kept constant. This means we can apply numerical algorithms (*e.g.*, the conjugate gradient method) for the solution of the interpolation system that can be performed with $\mathcal{O}(N)$ computational complexity. The method is numerically stable, but there will be essentially no convergence (see our earlier numerical experiments in Table 12.1). In the non-stationary case, *i.e.*, with fixed support size, the bandwidth of A increases as $h_{\mathcal{X},\Omega}$ decreases. This results in convergence (*i.e.*, the error decreases) as we showed with our experiments in Table 12.2 and the error bounds in Section 15.1.2. However, the interpolation matrices will become more and more densely populated as well as ill-conditioned. Therefore, this approach is not very efficient.

Chapter 17

Numerical Evidence for Approximation Order Results

17.1 Interpolation for $\varepsilon \to 0$

We begin by considering the choice of the shape parameter for a fixed data set. This is probably the situation that will arise most frequently in practical situations. In other words, we assume we are given a set of data $(\boldsymbol{x}_j, f_j)$, $j = 1, \ldots, N$, with data sites $\boldsymbol{x}_j \in \mathbb{R}^s$ (with $s = 1$ or $s = 2$ for the purpose of our experiments), and function values $f_j = f(\boldsymbol{x}_j) \in \mathbb{R}$. Our goal is to use an RBF interpolant

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{j=1}^{N} \mathrm{c}_j \varphi(\|\boldsymbol{x} - \boldsymbol{x}_j\|)$$

to match these data exactly, *i.e.*, to satisfy $\mathcal{P}_f(\boldsymbol{x}_i) = f(\boldsymbol{x}_i)$, $i = 1, \ldots, N$. The two most important questions now seem to be:

- Which basic function φ should we use?
- How should we scale the basis functions $\varphi_j = \varphi(\| \cdot - \boldsymbol{x}_j \|)$?

The error bounds we reviewed in previous chapters give us some insight into the first issue. If we know that the data come from a very smooth function, then application of one of the smoother basic functions is called for. Otherwise, there is not much to be gained from doing so. In fact, these functions may add too much smoothness to the interpolant. A first attempt at providing guidelines for the selection of appropriate basic functions (or kernels) can be found in [Schaback and Wendland (2006)]. We will not pursue this issue any further.

Instead we want to focus our attention on the second question, *i.e.*, the choice of the shape parameter ε. A number of strategies can be used to guide us in making a decision. We will assume throughout that a (fixed) basic function has been chosen, and that we will use only one value to scale all basis functions uniformly. Clearly, one can also follow other strategies such as using a shape parameter that varies with j, or even basic functions that vary with j. While some work has been done in these directions (see, *e.g.*, [Bozzini *et al.* (2002); Kansa and Carlson (1992); Schaback and Wendland (2000b); Fornberg and Zuev (2006)]), not much concrete can be said in these cases.

We now discuss four strategies for choosing a "good" value of ε.

17.1.1 *Choosing a Good Shape Parameter via Trial and Error*

The simplest strategy is to perform a series of interpolation experiments with varying shape parameter, and then to pick the "best" one. This strategy can be used if we know the function f that generated the data, and therefore can calculate some sort of error for the interpolant. Of course, if we already know f, then the exercise of finding an interpolant $\mathcal{P}_f$ may be mostly pointless. However, this is the strategy we used for the "academic" examples in Chapter 2.

If we do not have any knowledge of f, then it becomes very difficult to decide what "best" means. One (non-optimal) criterion we used in Chapter 2 was based on the trade-off principle, *i.e.*, the fact that for small ε the error improves while the condition number grows. We then defined "best" to be the smallest ε for which MATLAB did not issue a near-singular warning.

In many cases selection of an optimal shape parameter via *trial and error* will end up being a rather subjective process. However, this may presently be the approach taken by most practitioners.

For comparison with the other selection methods featured below we present three one-dimensional test cases for which we know the data function f. We use

$$F_1(x) = \text{sine}(x),$$
$$F_2(x) = \frac{3}{4}\left(e^{-(9x-2)^2/4} + e^{-(9x+1)^2/49}\right) + \frac{1}{2}e^{-(9x-7)^2/4} - \frac{1}{10}e^{-(9x-4)^2},$$
$$F_3(x) = \left(1 - |x - \frac{1}{2}|\right)^5 \left(1 + 5|x - \frac{1}{2}| - 27|x - \frac{1}{2}|^2\right).$$

The first of these functions is the classical band limited function (and thus in the native space of Gaussians). The second function is a one-dimensional variant of Franke's function, and the third function is one of Gneiting's C^2 oscillatory compactly supported RBFs shifted to the point $(1/2, 1/2)$ (see Table 11.4).

For these functions we list maximum errors and optimal shape parameters ε in Table 17.1. Maximum errors for a large range of ε values and the different values of N used in Table 17.1 are displayed in Figure 17.1. The optimal ε values listed in Table 17.1 corresponds to the lowest point for each of the curves in the figure. Clearly, the optimal value of the shape parameter is strongly dependent on the function that generated the test data.

17.1.2 *The Power Function as Indicator for a Good Shape Parameter*

Another strategy is suggested by the error analysis of Chapter 14. We showed there in Theorem 14.2 that

$$|f(\boldsymbol{x}) - \mathcal{P}_f(\boldsymbol{x})| \leq P_{\Phi,\mathcal{X}}(\boldsymbol{x})\|f\|_{\mathcal{N}_\Phi(\Omega)},$$

Table 17.1 Optimal ε values based on Gaussian interpolation to various test functions in 1D for various choices of N uniform points.

	F_1		F_2		F_3	
N	max-error	optimal ε	max-error	optimal ε	max-error	optimal ε
3	2.1403e-003	1.12	4.9722e-001	2.20	3.7087e-002	5.68
5	2.3260e-005	0.68	6.9380e-002	5.44	2.5253e-002	5.20
9	4.8764e-009	0.64	1.7555e-002	5.20	2.5360e-003	8.84
17	1.8922e-010	1.00	2.1928e-004	5.80	1.4380e-003	9.52
33	1.5250e-010	2.04	1.6536e-007	6.08	3.4189e-004	13.24
65	4.1307e-010	2.04	3.6260e-009	7.48	8.6431e-005	21.84

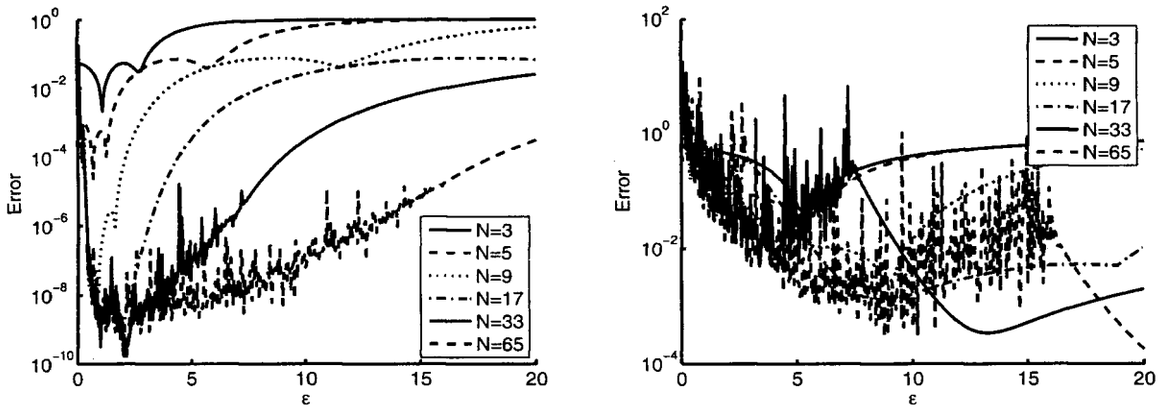


Fig. 17.1 Optimal ε curves based on Gaussian interpolation in 1D for various choices of N uniform points. Data sampled from sine function F_1 (left) and C^2 oscillatory function F_3 (right).

where $P_{\Phi,\mathcal{X}}$ denotes the power function. This estimate decouples the interpolation error into a component independent of the data function f and one depending on f. Once we have decided on a basic function Φ and a data set $\mathcal{X}$ we can use the power function based on scaled versions of Φ to optimize the error component that is independent of f. While this approach has the advantage over the previously mentioned trial and error approach that it is objective and does not depend on any knowledge of the data function, unfortunately, this approach will not be an optimal one since the second component of the error bound also depends on the basic function via the native space norm (which changes when Φ is scaled).

We said earlier (see (14.4)) that the power function can be computed via

$$P_{\Phi,\mathcal{X}}(\boldsymbol{x}) = \sqrt{\Phi(\boldsymbol{x},\boldsymbol{x}) - (\boldsymbol{b}(\boldsymbol{x}))^T A^{-1}\boldsymbol{b}(\boldsymbol{x})},$$

where A is the interpolation matrix and $\boldsymbol{b} = [\Phi(\cdot,\boldsymbol{x}_1),\ldots,\Phi(\cdot,\boldsymbol{x}_N)]^T$. This formula is implemented on lines 15–18 in the MATLAB program `Powerfunction2D.m`. We compute the inverse of A using the function `pinv` which is based on the singular value decomposition of A and therefore guarantees maximum stability. Also, due to roundoff some of the arguments of the `sqrt` function on line 18 come out negative.

This explains the use of the `real` command. The vectors $\boldsymbol{b}(\boldsymbol{x})$ are just rows of the evaluation matrix if $\boldsymbol{x}$ is taken from the grid of evaluation points we used earlier for error computations and plotting purposes. Except for the loop over the shape parameter ε (lines 12–20) the rest of the program is similar to earlier code.

Program 17.1. `Powerfunction2D.m`

```
% Powerfunction2D
% Script that finds "optimal" shape parameter by computing the power
% function for the 2D RBF interpolation approach with varying epsilon
% Calls on: DistanceMatrix
 1  rbf = @(e,r) exp(-(e*r).^2);   % Define the Gaussian RBF
    % Parameters for shape parameter loop below
 2  mine = 0; maxe = 20;
 3  ne = 500; ep = linspace(mine,maxe,ne);
    % Number and type of data points
 4  N = 81; gridtype = 'u';
    % Resolution of grid for power function norm computation
 5  neval = 20; M = neval^2;
    % Load data points
 6  name = sprintf('Data2D_%d%s',N,gridtype); load(name)
 7  ctrs = dsites;   % centers coincide with data sites
 8  grid = linspace(0,1,neval); [xe,ye] = meshgrid(grid);
 9  epoints = [xe(:) ye(:)];
    % Compute distance matrix between evaluation points and centers
10  DM_eval = DistanceMatrix(epoints,ctrs);
    % Compute distance matrix between the data sites and centers
11  DM_data = DistanceMatrix(dsites,ctrs);
12  for i=1:length(ep)
       % Compute interpolation matrix
13     IM = rbf(ep(i),DM_data);
       % Compute evaluation matrix
14     EM = rbf(ep(i),DM_eval);
       % Compute power function at evaluation points
15     invIM = inv(IM); phi0 = rbf(ep(i),0);
16     for j=1:M
17        powfun(j) = real(sqrt(phi0-(invIM*EM(j,:)')'*EM(j,:)'));
18     end
       % Compute max. norm of power function on evaluation grid
19     maxPF(i) = max(powfun);
20  end
21  fprintf('Smallest maximum norm: %e\n', min(maxPF))
22  fprintf('at epsilon = %f\n',ep(maxPF==min(maxPF)))
```

```
23  fprintf('with cond(A) = %e\n', ...
        condest(rbf(ep(find(maxPF==min(maxPF))),DM_data)))
    % Plot power function norm
24  figure; semilogy(ep,maxPF,'b');
```

In Figure 17.2 we show plots of the maximum norms of the power function vs. ε for a one-dimensional experiment (left) and a 2D experiment (right). Each plot shows several curves corresponding to different choices of N (set on line 4 of `Powerfuntion2D.m`). The optimal ε values along with the corresponding condition numbers of the interpolation matrix (computed using the `condest` command) are listed in Table 17.2. The graphs of the maximum norm of the power function can all be included in a single plot by adding another loop to the program which varies N. The program for the one-dimensional case is almost identical to the one printed and therefore omitted.

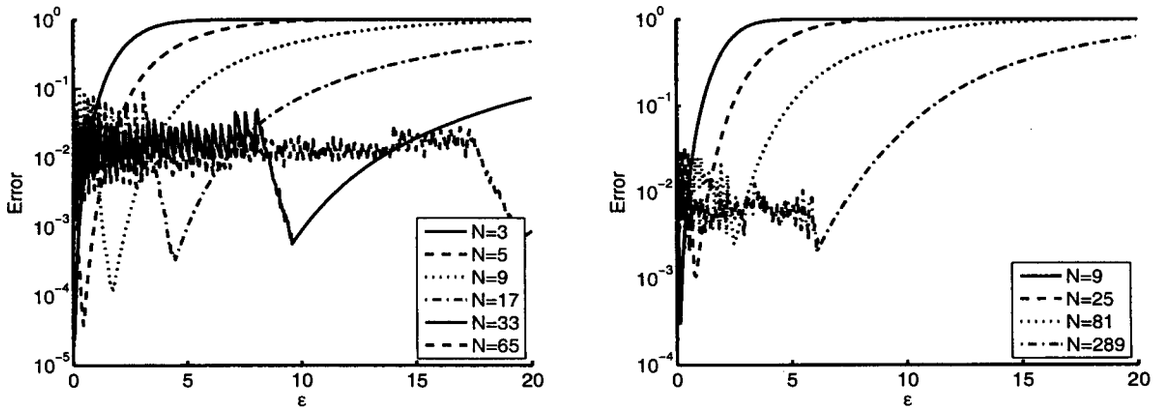


Fig. 17.2 Optimal ε curves based on power functions for Gaussians in 1D (left) and 2D (right) for various choices of N uniform points.

Clearly, even for the small data sets considered here, the numerical instability, *i.e.*, large condition number of the interpolation matrix A, plays a significant role.

Table 17.2 Optimal ε values based on power functions for Gaussians in 1D and 2D for various choices of N uniform points.

1D			2D		
N	optimal ε	cond(A)	N	optimal ε	cond(A)
3	0.04	1.8749e+007	9	0.16	5.3534e+009
5	0.44	5.7658e+007	25	0.84	1.0211e+011
9	1.72	6.5682e+008	81	0.04	2.0734e+019
17	4.48	6.1306e+009	289	0.56	1.2194e+020
33	9.60	5.4579e+010			
65	19.52	1.2440e+011			

17.1.3 *Choosing a Good Shape Parameter via Cross Validation*

A third strategy for finding an "optimal" shape parameter is to use a *cross validation* approach. In [Rippa (1999)] an algorithm is described that corresponds to a variant of cross validation known as *"leave-one-out" cross validation.* This method is rather popular in the statistics literature where it is also known as PRESS (Predictive REsidual Sum of Squares) provided the 2-norm is used. In this algorithm an "optimal" value of ε is selected by minimizing the (least squares) error for a fit to the data based on an interpolant for which one of the centers was "left out". A major advantage over the previous method is that now the dependence of the error on the data function is also taken into account. Therefore, the predicted "optimal" shape parameter is closer to the one we found via the trial and error approach (for which we had to assume knowledge of the exact solution).

A similar strategy was proposed earlier in [Golberg *et al.* (1996)] for the solution of elliptic partial differential equations via the dual reciprocity method based on multiquadric interpolation.

Specifically, if $\mathcal{P}_f^{[k]}$ is the radial basis function interpolant to the data $\{f_1, \ldots, f_{k-1}, f_{k+1}, \ldots, f_N\}$, *i.e.*,

$$\mathcal{P}_f^{[k]}(\boldsymbol{x}) = \sum_{\substack{j=1 \\ j \neq k}}^{N} c_j^{[k]} \varphi(\|\boldsymbol{x} - \boldsymbol{x}_j\|),$$

such that

$$\mathcal{P}_f^{[k]}(\boldsymbol{x}_i) = f_i, \qquad i = 1, \ldots, k-1, k+1, \ldots, N,$$

and if E_k is the error

$$E_k = f_k - \mathcal{P}_f^{[k]}(\boldsymbol{x}_k)$$

at the one point $\boldsymbol{x}_k$ not used to determine the interpolant, then the quality of the fit is determined by the norm of the vector of errors $E = [E_1, \ldots, E_N]^T$ obtained by removing in turn one of the data points and comparing the resulting fit with the (known) value at the removed point. In [Rippa (1999)] the author presented examples based on use of the ℓ_1 and ℓ_2 norms. We will mostly use the maximum norm (see line 14 in the code below).

By adding a loop over ε we can compare the error norms for different values of the shape parameter, and choose that value of ε that yields the minimal error norm as the optimal one.

While a naive implementation of the leave-one-out algorithm is rather expensive (on the order of N^4 operations), Rippa showed that the computation of the error components can be simplified to a single formula

$$E_k = \frac{c_k}{A_{kk}^{-1}}, \tag{17.1}$$

where c_k is the kth coefficient in the interpolant $\mathcal{P}_f$ based on the *full* data set, and A_{kk}^{-1} is the kth diagonal element of the inverse of the corresponding interpolation

matrix. Since both c_k and A^{-1} need to be computed only once for each value of ε this results in $\mathcal{O}(N^3)$ computational complexity. Note that all entries in the error vector E can be computed in a single statement in MATLAB if we vectorize the component formula (17.1) (see line 13 in Program 17.2). The sine function used on line 5 is not a standard MATLAB function (it is part of the Signal Processing Toolbox). Therefore we provide it in Program C.2 in Appendix C.

Program 17.2. LOOCV2D.m

```
% LOOCV2D
% Script that performs leave-one-out cross-validation
% (Rippa's method) to find a good epsilon for 2D RBF interpolation
% Calls on: DistanceMatrix
 1  rbf = @(e,r) exp(-(e*r).^2);   % Gaussian RBF
    % Parameters for shape parameter loop below
 2  mine = 0; maxe = 20; ne = 500;
 3  ep = linspace(mine,maxe,ne);
    % Number and type of data points
 4  N = 81; gridtype = 'u';
    % Define test function
 5  testfunction = @(x,y) sinc(x).*sinc(y);
    % Load data points
 6  name = sprintf('Data2D_%d%s',N,gridtype); load(name)
 7  ctrs = dsites;   % centers coincide with data sites
    % Create right-hand side vector, i.e.,
    % evaluate the test function at the data points.
 8  rhs = testfunction(dsites(:,1),dsites(:,2));
    % Compute distance matrix between the data sites and centers
 9  DM_data = DistanceMatrix(dsites,ctrs);
10  for i=1:length(ep)
        % Compute interpolation matrix
11      IM = rbf(ep(i),DM_data);
        % Compute error function (i.e., "cost" of epsilon)
12      invIM = pinv(IM);
13      EF = (invIM*rhs)./diag(invIM);
        % Compute maximum norm of EF
14      maxEF(i) = norm(EF(:),inf);
15  end
16  fprintf('Smallest maximum norm: %e\n', min(maxEF))
17  fprintf('at epsilon = %f\n',ep(maxEF==min(maxEF)))
    % Plot cost function norm
18  figure; semilogy(ep,maxEF,'b');
```

In Figure 17.3 we show plots of the predicted maximum errors vs. ε for a one-

dimensional experiment (left) and a 2D experiment (right) based on data sampled from the sine function F_1. Each plot shows several curves corresponding to different choices of N (set on line 4 of `LOOCV2D.m`). The optimal ε values are listed in Table 17.3.

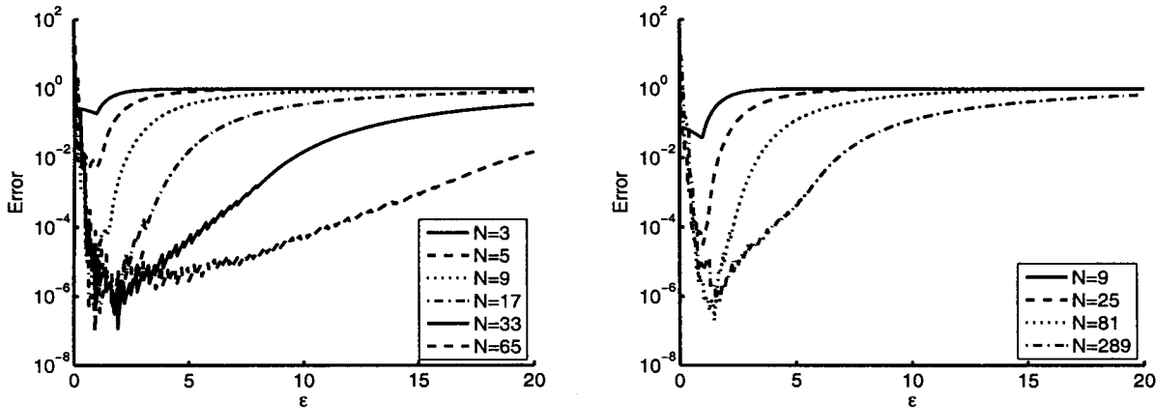


Fig. 17.3 Optimal ε curves based on leave-one-out cross validation for interpolation to the sine function with Gaussians in 1D (left) and 2D (right) for various choices of N uniform points.

Table 17.3 Optimal ε values based on leave-one-out cross validation for interpolation to the sinc function with Gaussians in 1D and 2D for various choices of N uniform points.

1D		2D	
N	optimal ε	N	optimal ε
3	0.96	9	0.96
5	1.00	25	1.00
9	0.80	81	1.48
17	0.92	289	1.60
33	1.92		
65	1.76		

The graphs in Figure 17.4 show side-by-side the optimal ε curves for the trial and error approach and for the leave-one-out cross validation approach in the case of 1D Gaussian interpolation to data sampled from the test function F_2. The similarity of the curves is clearly apparent. Thus, the leave-one-out cross validation approach can be recommended as a good method for selecting an"optimal" shape parameter ε since for this method no knowledge of the exact error is needed. Another pair of comparison plots is given by the Gaussian interpolants to the sine function F_1 in Figure 17.1 (left) and Figure 17.3 (left).

Similar conclusions hold for other basic functions, other test functions, other data distributions, and other space dimensions. For example, Figure 17.5 shows the

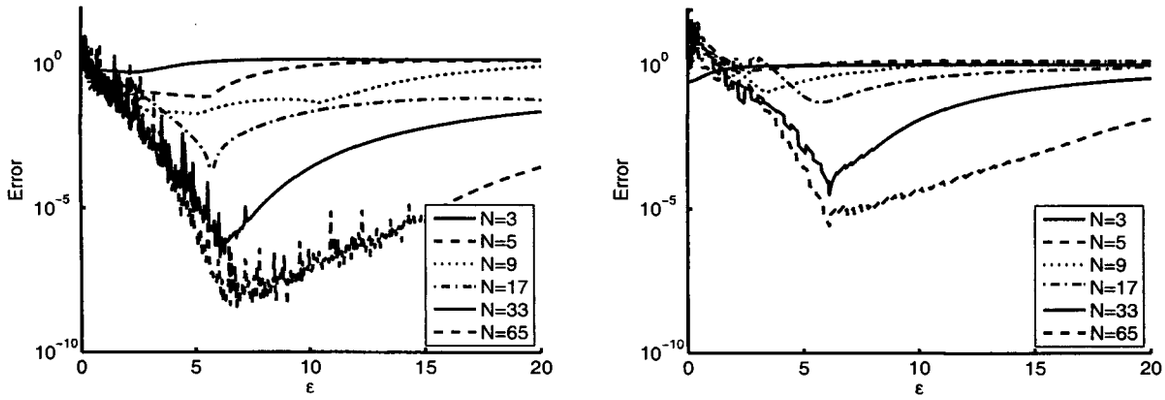


Fig. 17.4 Optimal ε curves based on interpolation to the test function F_2 with Gaussians for various choices of N uniform points. Trial and error approach (left), leave-one-out cross validation (right).

optimal ε curves for interpolation to the 1D Franke function F_2 with Wendland's C^2 function $\varphi_{3,1}$ on uniformly spaced points, and on Chebyshev points. Note that for this configuration all computations are stable, and the optimal scale parameter is quite small, *i.e.*, the support radius of the compactly supported basic function is chosen to be very large. In other words, the best results for compactly supported functions are obtained with dense matrices.

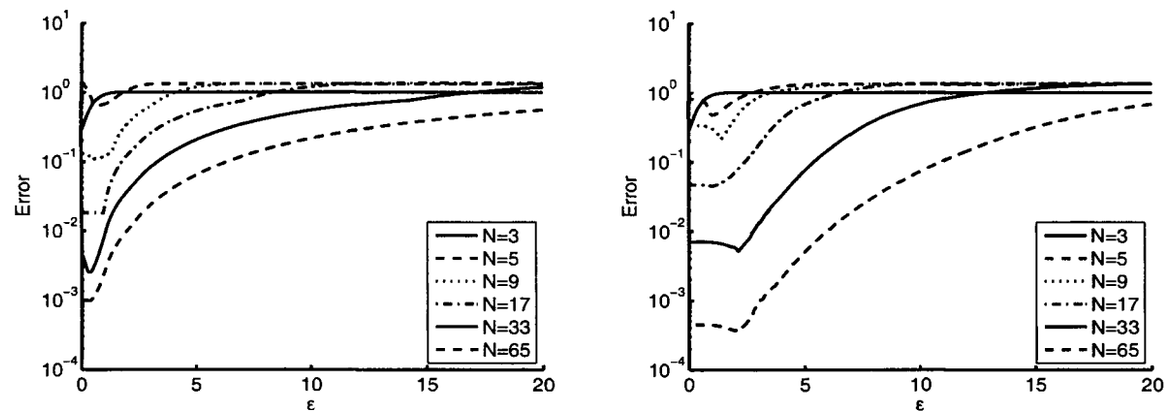


Fig. 17.5 Optimal ε curves based on leave-one-out cross validation for interpolation to 1D Franke's function with Wendland's function $\varphi_{3,1}$ for various choices of N uniform points (left) and Chebyshev points (right).

If we are not interested in the ε-curves displayed above, but only want to find a good value of the shape parameter as quickly as possible, then we can use the MATLAB function `fminbnd` to find the minimum of the cost function for ε. First, we implement the cost function in the subroutine `CostEpsilon.m` displayed in Program 17.3. The commands are the same as those on lines 11–14 in Program 17.2.

Program 17.3. CostEpsilon.m

```
% ceps = CostEpsilon(ep,r,rbf,rhs)
% Implements cost function for optimization of shape parameter
% via Rippa's LOOCV algorithm
% Example of usage in LOOCV2Dmin.m
 1  function ceps = CostEpsilon(ep,r,rbf,rhs)
 2  A = rbf(ep,r);
 3  invA = pinv(A);
 4  EF = (invA*rhs)./diag(invA);
 5  ceps = norm(EF(:),inf);
```

In order to demonstrate the use of the CostEpsilon function we use a modification of Program 17.2 which we list as Program 17.4.

Program 17.4. LOOCV2Dmin.m

```
% LOOCV2Dmin
% Script that performs leave-one-out cross-validation
% (Rippa's method) to find a good epsilon for 2D RBF interpolation
% with the help of MATLAB's fminbnd
% Calls on: DistanceMatrix
% Requires: CostEpsilon
 1  rbf = @(e,r) exp(-(e*r).^2);   % Gaussian RBF
    % Parameters for shape parameter optimization below
 2  mine = 0; maxe = 20;
    % Number and type of data points
 3  N = 81; gridtype = 'u';
    % Define test function
 4  testfunction = @(x,y) sinc(x).*sinc(y);
    % Load data points
 5  name = sprintf('Data2D_%d%s',N,gridtype); load(name)
 6  ctrs = dsites;   % centers coincide with data sites
    % Create right-hand side vector, i.e.,
    % evaluate the test function at the data points.
 7  rhs = testfunction(dsites(:,1),dsites(:,2));
    % Compute distance matrix between the data sites and centers
 8  DM_data = DistanceMatrix(dsites,ctrs);
 9a [ep,fval] = fminbnd(@(ep) CostEpsilon(ep,DM_data,rbf,rhs),...
 9b                     mine,maxe);
10  fprintf('Smallest maximum norm: %e\n', fval)
11  fprintf('at epsilon = %f\n', ep)
```

17.1.4 *The Contour-Padé Algorithm*

The Contour-Padé algorithm was the subject of Grady Wright's Ph.D. thesis [Wright (2003)] and was reported in [Fornberg and Wright (2004)]. The aim of the Contour-Padé algorithm is to come up with a method that allows the computation and evaluation of RBF interpolants for infinitely smooth basic functions when the shape parameter ε tends to zero (including the limiting case).

The starting point is to consider evaluation of the RBF interpolant

$$\mathcal{P}_f(\boldsymbol{x},\varepsilon) = \sum_{j=1}^{N} c_j \varphi_\varepsilon(\|\boldsymbol{x}-\boldsymbol{x}_j\|)$$

for a fixed evaluation point $\boldsymbol{x}$ as an analytic function of ε.

The key idea is to represent $\mathcal{P}_f(\boldsymbol{x},\varepsilon)$ by a *Laurent series* in ε, and approximate the "negative part" of the series by a Padé approximant, *i.e.*,

$$\mathcal{P}_f(\boldsymbol{x},\varepsilon) \approx r(\varepsilon) + \sum_{k=0}^{\infty} d_k \varepsilon^k,$$

where $r(\varepsilon)$ is the rational Padé approximant.

We then rewrite the interpolant in cardinal form, *i.e.*, as

$$\begin{aligned}\mathcal{P}_f(\boldsymbol{x},\varepsilon) &= \sum_{j=1}^{N} c_j \varphi_\varepsilon(\|\boldsymbol{x}-\boldsymbol{x}_j\|)\\ &= \boldsymbol{b}^T(\boldsymbol{x},\varepsilon)\boldsymbol{c}\\ &= \boldsymbol{b}^T(\boldsymbol{x},\varepsilon)A^{-1}(\varepsilon)\boldsymbol{f}\\ &= (\boldsymbol{u}^*(\boldsymbol{x},\varepsilon))^T\,\boldsymbol{f}\end{aligned}$$

where $\boldsymbol{b}(\boldsymbol{x},\varepsilon)_j = \varphi_\varepsilon(\|\boldsymbol{x}-\boldsymbol{x}_j\|)$, $A(\varepsilon)_{i,j} = \varphi_\varepsilon(\|\boldsymbol{x}_i-\boldsymbol{x}_j\|)$, $\boldsymbol{c} = [c_1,\ldots,c_N]^T$, $\boldsymbol{f} = [f_1,\ldots,f_N]^T$, and

$$\boldsymbol{u}^*(\boldsymbol{x},\varepsilon) = A^{-1}(\varepsilon)\boldsymbol{b}(\boldsymbol{x},\varepsilon)$$

denotes the vector of values of the cardinal functions at $\boldsymbol{x}$ (*c.f.* Chapter 14).

It is now the goal to stably compute the vector $\boldsymbol{u}^*(\varepsilon)$ for all values of $\varepsilon \geq 0$ without explicitly forming the inverse $A(\varepsilon)^{-1}$ and without computing the matrix vector product $A(\varepsilon)^{-1}\boldsymbol{b}(\varepsilon)$. Here the vectors $\boldsymbol{u}^*(\varepsilon)$ and $\boldsymbol{b}(\varepsilon)$ are obtained by evaluating the vector functions $\boldsymbol{u}^*(\cdot,\varepsilon)$ and $\boldsymbol{b}(\cdot,\varepsilon)$ on an appropriate evaluation grid.

The solution proposed by Wright and Fornberg is to use Cauchy's integral theorem to integrate around a circle in the complex ε-plane. The residuals (*i.e.*, coefficients in the Laurent expansion) are obtained using the (inverse) fast Fourier transform. The terms with negative powers of ε are then approximated using a rational Padé approximant. The integration contour (usually a circle) has to lie between the region of instability near $\varepsilon = 0$ and possible branch point singularities that lie somewhere in the complex plane depending on the choice of φ. Details of the method can be found in [Fornberg and Wright (2004)].

In Figure 17.6 we show optimal ε curves for interpolation to the 1D and 2D sine function F_2 using Gaussians at equally spaced points. These curves should be compared with the optimal ε curves obtained for the same problem via trial and error (see Figure 17.1 and Table 17.1) and via leave-one-out cross validation (see Figure 17.3 and Table 17.3).

The main drawback of the Contour-Padé algorithm is the fact that if N becomes too large then the region of ill-conditioning around the origin in the complex ε-plane and the branch point singularities will overlap. This, however, implies that the method can only be used with limited success. Moreover, as the graphs in Figure 17.6 and the entries in Table 17.4 show, the value of N that has to be considered "large" is unfortunately rather small. For the one-dimensional case the results for $N = 17$ already are affected by instabilities, and in the two-dimensional experiment $N = 81$ causes problems.

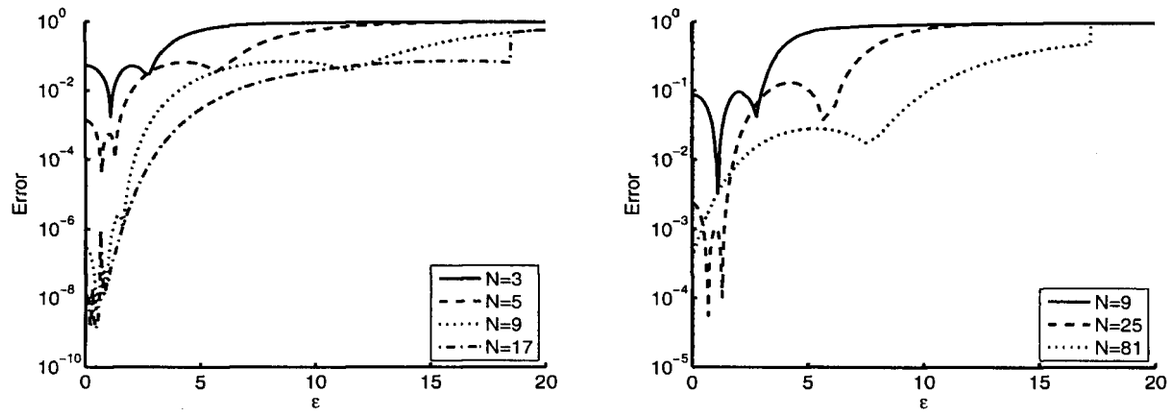


Fig. 17.6 Optimal ε curves based on Contour-Padé for interpolation to the sinc function with Gaussians in 1D (left) and 2D (right) for various choices of N uniform points.

Table 17.4 Optimal ε values based on Contour-Padé for interpolation to the sinc function with Gaussians in 1D and 2D for various choices of N uniform points.

1D				2D			
N	max-error	ε	cond(A)	N	max-error	ε	cond(A)
3	1.7605e-003	1.10	3.3386e+001	9	3.3875e-003	1.10	1.1146e+003
5	4.0380e-005	0.70	1.3852e+006	25	5.5542e-005	0.70	1.9187e+012
9	3.9703e-009	0.45	7.7731e+016	81	3.6528e-004	0.00	∞
17	1.2726e-009	0.45	1.7327e+018				

17.1.5 *Summary*

All strategies pursued in this chapter have shown that even though the bound (15.11) by Madych seems to indicate that the interpolation error for functions in

the native space of the basic function goes to zero exponentially as $\varepsilon \to 0$, this does not seem to be true in practice. Especially those optimal ε curves that were computed reliably with the Contour-Padé algorithm all have a global minimum for some positive value of ε. In many cases this optimal ε value (or an ε close to the optimal value) can be found using the leave-one-out cross validation algorithm of Program 17.2. From now on we will frequently use leave-one-out cross validation to find an optimal shape parameter for our numerical experiments.

17.2 Non-stationary Interpolation

In order to illustrate the spectral convergence predicted for infinitely smooth basic functions such as Gaussians and generalized (inverse) multiquadrics we need to work in a setting for which neither the instability due to large problem size or small shape parameter have a significant effect on our experiments. Otherwise, if we simply take an "optimal" value of ε (determined via trial and error for a large $N = 4225$ problem in the "gray zone", c.*f.* Chapter 2) then the spectral convergence will only be visible for a limited number of experiments (see Table 17.5).

Table 17.5 2D non-stationary interpolation ($\varepsilon = 6.3$) to Franke's function with Gaussians on uniformly spaced and Halton points.

	uniform		Halton	
N	RMS-error	rate	RMS-error	rate
9	3.195983e-001		2.734756e-001	
25	5.008591e-002	2.6738	8.831682e-002	2.3004
81	9.029664e-003	2.4717	2.401868e-002	1.7582
289	2.263880e-004	5.3178	1.589117e-003	5.0969
1089	3.323287e-008	12.7339	1.595051e-006	10.8015
4225	1.868286e-008	0.8309	9.510404e-008	4.8203

Even for a band-limited function (see Table 17.6) the situation is not better; in fact worse, for the value of ε used.

In Figures 17.7–17.8 we are able to verify (at least to some extent) the convergence estimates for non-stationary RBF interpolants. We obtain the data for all experiments by sampling the sine function $f(x) = \sin(\pi x)/(\pi x)$ at N uniformly spaced points in the interval $[0, 1]$ where N runs from 1 to 100. Each plot shows six maximum error curves (corresponding to shape parameters $\varepsilon = 1, 6, 11, 16, 21, 26$) versus the number N of data points on a loglog scale. The errors are evaluated on a grid of 250 equally spaced points. In order to compare these curves with the theoretical bounds from Chapter 15 we have plotted comparison curves corresponding to the theoretical bounds. For Gaussians the comparison curve is given by the graph of $h \mapsto e^{-|\log h|/h}$ corresponding to super-spectral convergence with $h = 1/(N-1)$,

Table 17.6 2D non-stationary interpolation ($\varepsilon = 6.3$) to the sinc function with Gaussians on uniformly spaced and Halton points.

	uniform		Halton	
N	RMS-error	rate	RMS-error	rate
9	3.302644e-001		2.823150e-001	
25	3.271035e-002	3.3358	1.282572e-001	1.6058
81	1.293184e-002	1.3388	3.407580e-002	1.7898
289	3.786113e-004	5.0941	1.990217e-003	5.3309
1089	3.476835e-008	13.4107	2.286014e-006	10.5905
4225	3.775365e-008	−0.1188	9.868530e-008	5.3724

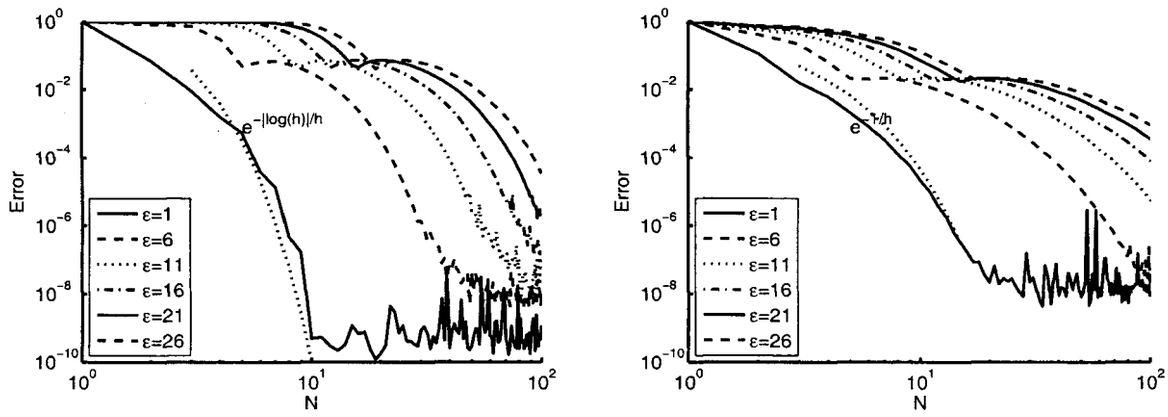


Fig. 17.7 Maximum errors for non-stationary interpolation to the sine function with Gaussians (left) and inverse multiquadrics (right) based on N uniformly spaced points in $[0, 1]$ and $\varepsilon = 1, 6, 11, 16, 21, 26$.

and for inverse multiquadrics we have spectral convergence with $h \mapsto e^{-1/h}$. We can see that for a certain range of problems these rates are indeed obtained (see Figure 17.7).

In the case of functions with finite smoothness (such as the compactly supported functions of Wendland) we can only expect algebraic convergence rates. Figure 17.8 shows two more sets of maximum error curves. These plots are based on Wendland's C^2 function $\varphi_{3,1}(r) = (1-r)_+^4(4r+1)$ and the C^6 function $\varphi_{3,3}(r) = (1-r)_+^8(32r^3+25r^2+8r+1)$. While the error bound (15.5) predicts only $\mathcal{O}(h^{3/2})$ and $\mathcal{O}(h^{7/2})$ approximation order, respectively. We see that an extra factor of $h^{s/2}$ is indeed possible in practice. This extra factor has also been captured in some of the theoretical work on improved error bounds (*c.f.* Section 15.2).

For less smooth data functions we no longer have spectral convergence for the infinitely smooth functions, while the orders remain unchanged for the basic functions with finite smoothness (as long as the data function lies in the native space of the basic function). This is illustrated in Figure 17.9 where we compare Gaussians and C^2 Wendland functions for the C^2 test function

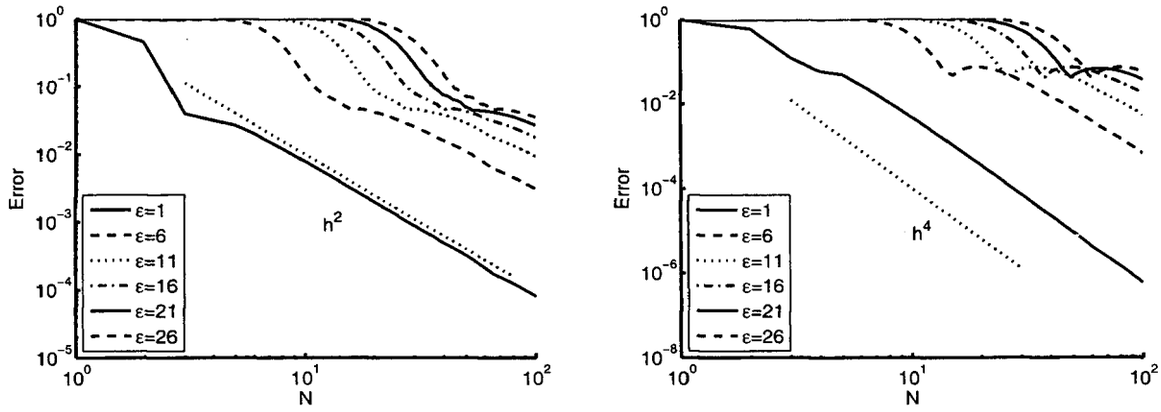


Fig. 17.8 Maximum errors for non-stationary interpolation to the sine function with C^2 (left) and C^6 (right) Wendland function based on N uniformly spaced points in $[0,1]$ and $\varepsilon = 1, 6, 11, 16, 21, 26$.

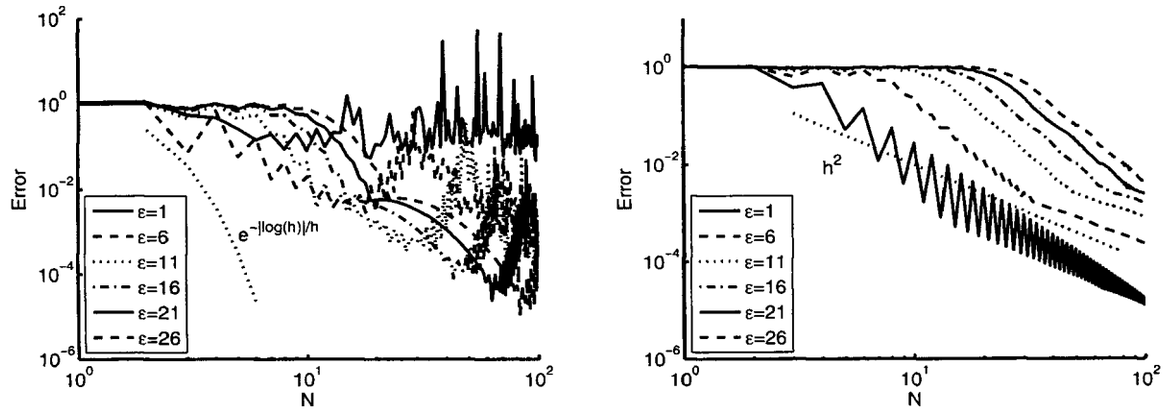


Fig. 17.9 Maximum errors for non-stationary interpolation to a C^2 function with Gaussians (left) and C^2 Wendland function (right) based on N uniformly spaced points in $[0,1]$ and $\varepsilon = 1, 6, 11, 16, 21, 26$.

$(1-|x-1/2|)_+^5(1+5|x-1/2|-27(x-1/2)^2)$ (*c.f.* the oscillatory functions of Table 11.4). It is interesting to note that for a certain range of N the rate of convergence for the C^2 Wendland function is even better than predicted.

17.3 Stationary Interpolation

We begin with an illustration of the fact that for radial powers and thin plate splines there is no difference in convergence behavior between the stationary and non-stationary regime. Figure 17.10 shows this phenomenon for the norm radial function $\Phi(\boldsymbol{x}) = \|\boldsymbol{x}\|$ in the case of interpolation to data sampled from the C^2 function $f(x) = |x-1/2|^3$ at uniformly spaced points in $[0,1]$. Moreover, the left plot in Figure 17.10 (illustrating the non-stationary setting) shows that the shape

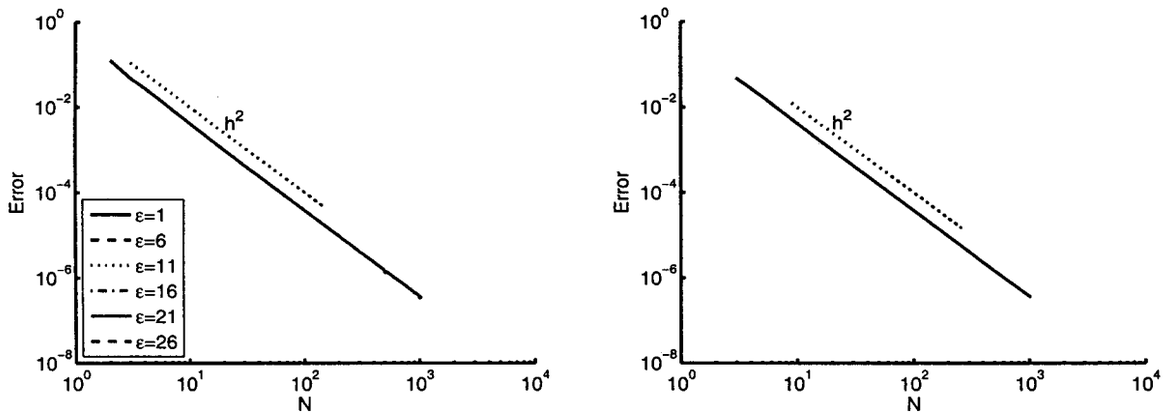


Fig. 17.10 Maximum errors for non-stationary (left) and stationary (right) interpolation to a C^2 function with the norm basic function based on N uniformly spaced points in $[0, 1]$.

parameter has no effect for the norm basic function and other polyharmonic splines.

Note that Figure 17.10 suggests that the norm basic function has $\mathcal{O}(h^2)$ approximation order, while the bound from Theorem 15.3 with $\tau = k = \beta = 1$, $n = 0$, $s = 1$ and $q = \infty$ yields only $\mathcal{O}(h^{1/2})$. Since the norm basic function is strictly conditionally positive definite of order one we can use the same RBF expansion as for strictly positive definite functions, *i.e.*, without appending a constant (*c.f.* Theorem 9.7).

The discrepancy between the theoretical bounds of Theorem 15.3 (or Theorem 15.4 as well as the native space bounds of Examples 15.6 and 15.7 of Chapter 15) and those observed in numerical experiments is similar for radial cubics and thin plate splines (which are both strictly conditionally positive definite of order two). For cubics Theorem 15.3 with $\tau = \beta = 3$, $k = 2$, $n = 0$, $s = 2$ and $q = \infty$ predicts $\mathcal{O}(h^2)$ since the mesh ratio provides another power of h for uniformly distributed data. The left plot of Figure 17.11, however, suggests $\mathcal{O}(h^3)$ or better L_∞ approximation order based on interpolation to the 2D analog of the oscillatory C^2 test function F_3, *i.e.*, $f(\boldsymbol{x}) = (1 - \|\boldsymbol{x} - (1/2, 1/2)\|)_+^5(1 + 5\|\boldsymbol{x} - (1/2, 1/2)\| - 27\|\boldsymbol{x} - (1/2, 1/2)\|^2)$. The predicted rate for thin plate splines is $\mathcal{O}(h^{3/2})$ (since $\tau = 2\beta = 2$, $k = 2$, $n = 0$, $s = 2$ and $q = \infty$) while the plot on the right of Figure 17.11 indicates at least $\mathcal{O}(h^2)$ convergence.

For Gaussian basis functions we noted earlier that we should not expect any convergence in the stationary setting. However, if the initial shape parameter is chosen small enough (but not too small), then we can observe the approximate approximation phenomenon, *i.e.*, there is convergence up to a certain point, and then saturation occurs. This is depicted in Figure 17.12. In the left plot we used the Gaussian basic function with different initial shape parameters ($\varepsilon = 0.8, 1.0, 1.2, 1.4, 1.6, 1.8$) to interpolate data sampled from the oscillatory C^2 function used in the previous illustration at uniformly spaced points in the unit square. The plot on the right corresponds to Gaussian interpolation of data sampled from the 2D sine function $f(x, y) = \text{sinc}(x)\text{sinc}(y)$ with initial $\varepsilon = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6$.

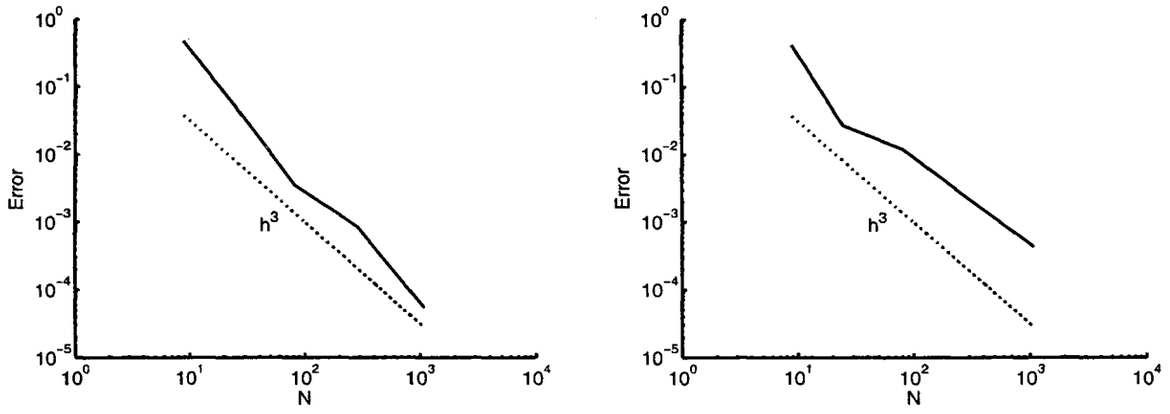


Fig. 17.11 Maximum errors for stationary interpolation to a C^2 function with the cubic radial basic function (left) and thin plate spline basic function (right) based on N uniformly spaced points in $[0, 1]^2$.

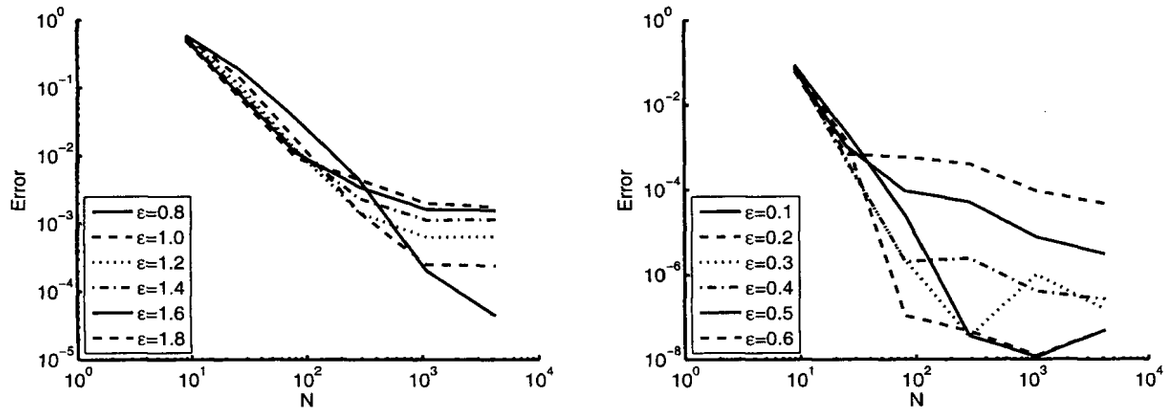


Fig. 17.12 Maximum errors for stationary interpolation to the C^2 oscillatory function (left) and to the sine function (right) with Gaussians based on N uniformly spaced points in $[0, 1]^2$ using various initial ε values.

If we consider the range of N values used in the experiments ($N = 9, 25, 81, 289, 1089, 4225$), then we see that stationary interpolation with Gaussians does converge for the smaller values of N (at at rate better than $\mathcal{O}(h^2)$). However, the larger the value of the initial ε is taken, the sooner does the saturation occur. It is also apparent that in the case of interpolation to the sine function small initial values of the shape parameter lead to severe ill-conditioning and subsequent instabilities especially for the tests with larger values of N. We also point out that the range of values of ε for which we can observe convergence depends on the data function f.

We will come back to the approximate approximation phenomenon in the context of quasi-interpolation and approximate moving least squares approximation in Chapters 26 and 27.

Chapter 18

The Optimality of RBF Interpolation

In this chapter we will see that within the native Hilbert spaces associated with strictly positive definite (and strictly conditionally positive definite) radial functions the radial basis function interpolant provides the *best approximation* to a given data function. This optimality of interpolants in Hilbert space is the subject of the theory of *optimal recovery* described in the late 1950s by Michael Golomb and Hans Weinberger in their paper [Golomb and Weinberger (1959)].

18.1 The Connection to Optimal Recovery

In [Golomb and Weinberger (1959)] the authors studied the following general problem:

Problem 18.1. *Given the values* $f_1 = \lambda_1(f), \ldots, f_N = \lambda_N(f) \in \mathbb{R}$, *where* $\{\lambda_1, \ldots, \lambda_N\}$ *is a linearly independent set of linear functionals (called* information functionals *yielding the* information *about* f*), how does one "best" approximate the value* $\lambda(f)$ *(called a* feature *of* f*) where* λ *is a given linear functional and* f *is unknown? Moreover, what is the total range of values for* $\lambda(f)$*?*

This is a very general problem formulation that allows not only for interpolation of function values, but also for other types of data (such as values of derivatives and integrals of f, such as averages or moments of f, etc.), as well as methods of approximation other than interpolation.

The kind of problem described above is known in the literature as an *optimal recovery problem.* Besides the seminal work by Golomb and Weinberger, optimal recovery was also studied in detail by Micchelli, Rivlin and Winograd [Micchelli *et al.* (1976); Micchelli and Rivlin (1977); Micchelli and Rivlin (1980); Micchelli and Rivlin (1985)].

In a Hilbert space setting the solution to this optimal recovery problem is shown to be the *minimum-norm interpolant.* More precisely, given a Hilbert space $\mathcal{H}$ and data $f_1 = \lambda_1(f), \ldots, f_N = \lambda_N(f) \in \mathbb{R}$ with $\{\lambda_1, \ldots, \lambda_N\} \subseteq \mathcal{H}^*$ (the dual of $\mathcal{H}$), the

minimum-norm interpolant is that function $g^* \in \mathcal{H}$ that satisfies

$$\lambda_j(g^*) = f_j, \qquad j = 1, \ldots, N,$$

and for which

$$\|g^*\|_{\mathcal{H}} = \min_{\substack{g \in \mathcal{H} \\ \lambda_j(g) = f_j, j = 1, \ldots, N}} \|g\|_{\mathcal{H}}.$$

It turns out that the radial basis function interpolant with basic function Φ satisfies these criteria if $\mathcal{H}$ is taken as the associated native space $\mathcal{N}_\Phi(\Omega)$.

We will present three optimality results:

- The radial basis function interpolant for any strictly conditionally positive definite function Φ is the minimum norm interpolant from $\mathcal{N}_\Phi(\Omega)$.
- The radial basis function interpolant provides the best approximation to f in the native space norm.
- The (cardinal form of the) radial basis function interpolant is more accurate (as measured by the pointwise error) than any other linear combination of the data.

18.2 Orthogonality in Reproducing Kernel Hilbert Spaces

The proofs of the first two "optimality theorems" require the following two lemmas. These lemmas and their corollary can also be generalized to cover the strictly conditionally positive definite case. However, to keep our discussion transparent, we present only the details of the strictly positive definite case.

Lemma 18.1. *Assume Φ is a symmetric strictly positive definite kernel on $\mathbb{R}^s$ and let $\mathcal{P}_f$ be the interpolant to $f \in \mathcal{N}_\Phi(\Omega)$ at the data sites $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subseteq \Omega$. Then*

$$\langle \mathcal{P}_f, \mathcal{P}_f - g \rangle_{\mathcal{N}_\Phi(\Omega)} = 0$$

for all interpolants $g \in \mathcal{N}_\Phi(\Omega)$, i.e., with $g(\boldsymbol{x}_j) = f(\boldsymbol{x}_j)$, $j = 1, \ldots, N$.

Proof. The interpolant $\mathcal{P}_f$ is of the form

$$\mathcal{P}_f = \sum_{j=1}^{N} c_j \Phi(\cdot, \boldsymbol{x}_j),$$

where the coefficients c_j are determined by the interpolation conditions $\mathcal{P}_f(\boldsymbol{x}_i) = f(\boldsymbol{x}_i)$, $i = \mathbf{1}, \ldots, N$. Using this representation, the symmetry of the kernel Φ and its reproducing property we have

$$\langle \mathcal{P}_f, \mathcal{P}_f - g \rangle_{\mathcal{N}_\Phi(\Omega)} = \langle \sum_{j=1}^{N} c_j \Phi(\cdot, \boldsymbol{x}_j), \mathcal{P}_f - g \rangle_{\mathcal{N}_\Phi(\Omega)}$$

$$
\begin{aligned}
&= \sum_{j=1}^{N} c_j \langle \Phi(\cdot, \boldsymbol{x}_j), \mathcal{P}_f - g \rangle_{\mathcal{N}_\Phi(\Omega)} \\
&= \sum_{j=1}^{N} c_j \langle \mathcal{P}_f - g, \Phi(\cdot, \boldsymbol{x}_j) \rangle_{\mathcal{N}_\Phi(\Omega)} \\
&= \sum_{j=1}^{N} c_j (\mathcal{P}_f - g)(\boldsymbol{x}_j) \\
&= 0
\end{aligned}
$$

since both $\mathcal{P}_f$ and g interpolate f on $\mathcal{X}$. □

For the next result, recall the definition of the space $H_\Phi(\mathcal{X})$ as the linear span

$$H_\Phi(\mathcal{X}) = \operatorname{span}\{\Phi(\cdot, \boldsymbol{x}_j) : \boldsymbol{x}_j \in \mathcal{X}\}$$

(*c.f.* (13.1)). Clearly, $H_\Phi(\mathcal{X})$ is a subspace of the native space $\mathcal{N}_\Phi(\Omega)$.

Lemma 18.2. *Assume Φ is a strictly positive definite kernel on $\mathbb{R}^s$ and let $\mathcal{P}_f$ be the interpolant to $f \in \mathcal{N}_\Phi(\Omega)$ on $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subseteq \Omega$. Then*

$$\langle f - \mathcal{P}_f, h \rangle_{\mathcal{N}_\Phi(\Omega)} = 0$$

for all $h \in H_\Phi(\mathcal{X})$.

Proof. Any $h \in H_\Phi(\mathcal{X})$ can be written in the form

$$h = \sum_{j=1}^{N} c_j \Phi(\cdot, \boldsymbol{x}_j)$$

with appropriate coefficients c_j. Using this representation of h as well as the reproducing property of Φ we have

$$
\begin{aligned}
\langle f - \mathcal{P}_f, h \rangle_{\mathcal{N}_\Phi(\Omega)} &= \langle f - \mathcal{P}_f, \sum_{j=1}^{N} c_j \Phi(\cdot, \boldsymbol{x}_j) \rangle_{\mathcal{N}_\Phi(\Omega)} \\
&= \sum_{j=1}^{N} c_j \langle f - \mathcal{P}_f, \Phi(\cdot, \boldsymbol{x}_j) \rangle_{\mathcal{N}_\Phi(\Omega)} \\
&= \sum_{j=1}^{N} c_j (f - \mathcal{P}_f)(\boldsymbol{x}_j).
\end{aligned}
$$

This last expression, however, is zero since $\mathcal{P}_f$ interpolates f on $\mathcal{X}$, *i.e.*, $(f - \mathcal{P}_f)(\boldsymbol{x}_j) = 0$, $j = 1, \ldots, N$. □

The following Pythagorean theorem (or "energy splitting" theorem) is an immediate consequence of Lemma 18.2. It says that the native space "energy" of f can be split into the "energy" of the interpolant $\mathcal{P}_f$ and that of the residual $f - \mathcal{P}_f$, which — according to Lemma 18.2 — is orthogonal to the interpolant.

Corollary 18.1. *The orthogonality property of Lemma 18.2 implies the* energy split

$$\|f\|^2_{\mathcal{N}_\Phi(\Omega)} = \|f - \mathcal{P}_f\|^2_{\mathcal{N}_\Phi(\Omega)} + \|\mathcal{P}_f\|^2_{\mathcal{N}_\Phi(\Omega)}.$$

Proof. The statement follows from

$$\begin{aligned}\|f\|^2_{\mathcal{N}_\Phi(\Omega)} &= \|f - \mathcal{P}_f + \mathcal{P}_f\|^2_{\mathcal{N}_\Phi(\Omega)} \\ &= \langle (f - \mathcal{P}_f) + \mathcal{P}_f, (f - \mathcal{P}_f) + \mathcal{P}_f \rangle_{\mathcal{N}_\Phi(\Omega)} \\ &= \|f - \mathcal{P}_f\|^2_{\mathcal{N}_\Phi(\Omega)} + 2\langle f - \mathcal{P}_f, \mathcal{P}_f \rangle_{\mathcal{N}_\Phi(\Omega)} + \|\mathcal{P}_f\|^2_{\mathcal{N}_\Phi(\Omega)}\end{aligned}$$

and the fact that $\langle f - \mathcal{P}_f, \mathcal{P}_f \rangle_{\mathcal{N}_\Phi(\Omega)} = 0$ by Lemma 18.2 since $\mathcal{P}_f \in H_\Phi(\mathcal{X})$. □

The above energy split is the fundamental idea behind a number of Krylov-type iterative algorithms for approximately solving the interpolation problem when very large data sets are involved (see, *e.g.*, our discussion in Chapter 33 or the papers [Faul and Powell (1999); Faul and Powell (2000)] or [Schaback and Wendland (2000a)]).

18.3 Optimality Theorem I

The following theorem presents the first optimality property formulated for the general case of strictly conditionally positive definite kernels. It is taken from [Wendland (2005a)].

Theorem 18.1 (Optimality I). *Suppose $\Phi \in C(\Omega \times \Omega)$ is a strictly conditionally positive definite kernel with respect to the finite-dimensional space $P \subseteq C(\Omega)$ and that $\mathcal{X}$ is P-unisolvent. If the values $f_1, \ldots, f_N$ are given, then the interpolant $\mathcal{P}_f$ is the minimum-(semi)norm interpolant to $\{f_j\}_{j=1}^N$, i.e.,*

$$|\mathcal{P}_f|_{\mathcal{N}_\Phi(\Omega)} = \min_{\substack{g \in \mathcal{N}_\Phi(\Omega) \\ g(x_j) = f_j, j = 1, \ldots, N}} |g|_{\mathcal{N}_\Phi(\Omega)}.$$

Proof. We consider only the strictly positive definite case. Consider an arbitrary interpolant $g \in \mathcal{N}_\Phi(\Omega)$ to $f_1, \ldots, f_N$. Then Lemma 18.1 gives us

$$\langle \mathcal{P}_f, \mathcal{P}_f - g \rangle_{\mathcal{N}_\Phi(\Omega)} = 0.$$

This orthogonality relation gives us

$$\begin{aligned}|\mathcal{P}_f|^2_{\mathcal{N}_\Phi(\Omega)} &= \langle \mathcal{P}_f, \mathcal{P}_f - g + g \rangle_{\mathcal{N}_\Phi(\Omega)} \\ &= \langle \mathcal{P}_f, \mathcal{P}_f - g \rangle_{\mathcal{N}_\Phi(\Omega)} + \langle \mathcal{P}_f, g \rangle_{\mathcal{N}_\Phi(\Omega)} \\ &= \langle \mathcal{P}_f, g \rangle_{\mathcal{N}_\Phi(\Omega)},\end{aligned}$$

and the Cauchy-Schwarz inequality yields

$$|\mathcal{P}_f|^2_{\mathcal{N}_\Phi(\Omega)} \leq |\mathcal{P}_f|_{\mathcal{N}_\Phi(\Omega)} |g|_{\mathcal{N}_\Phi(\Omega)},$$

so that the statement follows. □

As in our earlier use of conditionally positive definite functions, the space P mentioned in Theorem 18.1 is usually taken as the space Π^s_{m-1} of multivariate polynomials. Also, if Φ is strictly positive definite then the semi-norms in Theorem 18.1 become norms.

Example 18.1. We said earlier that the native space of thin plate splines $\phi(r) = r^2 \log r$, $r = \|\boldsymbol{x}\|_2$ with $\boldsymbol{x} = (x, y) \in \mathbb{R}^2$ is given by the Beppo-Levi space $\mathrm{BL}_2(\mathbb{R}^2)$. Now, the corresponding semi-norm in the Beppo-Levi space $\mathrm{BL}_2(\mathbb{R}^2)$ is (*c.f.* (13.6))

$$|f|^2_{\mathrm{BL}_2(\mathbb{R}^2)} = \int_{\mathbb{R}^2} \left(\left| \frac{\partial^2 f}{\partial x^2}(\boldsymbol{x}) \right|^2 + 2 \left| \frac{\partial^2 f}{\partial x \partial y}(\boldsymbol{x}) \right|^2 + \left| \frac{\partial^2 f}{\partial y^2}(\boldsymbol{x}) \right|^2 \right) d\boldsymbol{x},$$

which is the bending energy of a thin plate. By Theorem 18.1 the thin plate spline interpolant minimizes this energy. This explains the name of these functions.

18.4 Optimality Theorem II

Another nice property of the radial basis function interpolant is the fact that it is at the same time the best Hilbert-space approximation to the given data, and thus not just any projection of f but the *orthogonal projection.* Again, we formulate the theorem for the strictly conditionally positive definite case and provide details only for the strictly positive definite case.

Theorem 18.2 (Optimality II). *Let*

$$H_\Phi(\mathcal{X}) = \{h = \sum_{j=1}^{N} c_j \Phi(\cdot, \boldsymbol{x}_j) + p \ : \ p \in P \text{ and } \sum_{j=1}^{N} c_j q(\boldsymbol{x}_j) = 0 \text{ for all } q \in P \text{ and } \boldsymbol{x}_j \in \mathcal{X}\},$$

where $\Phi \in C(\Omega \times \Omega)$ is a strictly conditionally positive definite kernel with respect to the finite-dimensional space $P \subseteq C(\Omega)$ and $\mathcal{X}$ is P-unisolvent. If only the values $f_1 = f(\boldsymbol{x}_1), \ldots, f_N = f(\boldsymbol{x}_N)$ are given, then the interpolant $\mathcal{P}_f$ is the best approximation to f from $H_\Phi(\mathcal{X})$ in $\mathcal{N}_\Phi(\Omega)$, i.e.,

$$|f - \mathcal{P}_f|_{\mathcal{N}_\Phi(\Omega)} \leq |f - h|_{\mathcal{N}_\Phi(\Omega)}$$

for all $h \in H_\Phi(\mathcal{X})$.

Proof. We consider only the strictly positive definite case. As explained in Section 13.2, the native space $\mathcal{N}_\Phi(\Omega)$ is the completion of $H_\Phi(\Omega)$ with respect to the $\|\cdot\|_\Phi$-norm so that $\|f\|_\Phi = \|f\|_{\mathcal{N}_\Phi(\Omega)}$ for all $f \in H_\Phi(\Omega)$. Also, $\mathcal{X} \subseteq \Omega$. Therefore, we can characterize the best approximation g^* to f from $H_\Phi(\mathcal{X})$ by

$$\langle f - g^*, h \rangle_{\mathcal{N}_\Phi(\Omega)} = 0 \qquad \text{for all } h \in H_\Phi(\mathcal{X}).$$

However, Lemma 18.2 shows that $g^* = \mathcal{P}_f$ satisfies this relation. □

These optimality properties of radial basis function interpolants play an important role in applications such as in the design of support vector machines in statistical learning theory or the numerical solution of partial differential equations.

The optimality results above imply that one could also start with some Hilbert space $\mathcal{H}$ with norm $\|\cdot\|_{\mathcal{H}}$ and ask to find the minimum norm interpolant (*i.e.*, Hilbert space best approximation) to some given data. In this way any given space defines a set of *optimal basis functions*, generated by the reproducing kernel of $\mathcal{H}$. This is how Duchon approached the subject in his papers [Duchon (1976); Duchon (1977); Duchon (1978); Duchon (1980)]. More recently, Kybic, Blu and Unser [Kybic *et al.* (2002a); Kybic *et al.* (2002b)] take this point of view and explain from a sampling theory point of view how the thin plate splines can be interpreted as fundamental solutions of the differential operator defining the semi-norm in the Beppo-Levi space $\mathrm{BL}_2(\mathbb{R}^2)$, and thus radial basis functions can be viewed as *Green's functions.*

18.5 Optimality Theorem III

The third optimality result is in the context of quasi-interpolation, *i.e.*,

Theorem 18.3 (Optimality III). *Suppose $\Phi \in C(\Omega \times \Omega)$ is a strictly conditionally positive definite kernel with respect to the finite-dimensional space $P \subseteq C(\Omega)$. Suppose $\mathcal{X}$ is P-unisolvent and $\boldsymbol{x} \in \Omega$ is fixed. Let $u_j^*(\boldsymbol{x})$, $j = 1, \ldots, N$, be the values at $\boldsymbol{x}$ of the cardinal basis functions for interpolation with Φ. Then*

$$\left| f(\boldsymbol{x}) - \sum_{j=1}^{N} f(\boldsymbol{x}_j) u_j^*(\boldsymbol{x}) \right| \leq \left| f(\boldsymbol{x}) - \sum_{j=1}^{N} f(\boldsymbol{x}_j) u_j \right|$$

for all choices of $u_1, \ldots, u_N \in \mathbb{R}$ with $\sum_{j=1}^{N} u_j p(\boldsymbol{x}_j) = p(\boldsymbol{x})$ for any $p \in P$.

Theorem 18.3 is proved in [Wendland (2005a)]. It says in particular that the minimum norm interpolant $\mathcal{P}_f$ is also more accurate (in the pointwise sense) than any linear combination of the given data values that reproduce P.

Chapter 19

Least Squares RBF Approximation with MATLAB

Up to now we have looked only at interpolation. However, many times it makes more sense to approximate the given data by a least squares fit. This is especially true if the data are contaminated with noise, or if there are so many data points that efficiency considerations force us to approximate from a space spanned by fewer basis functions than data points.

19.1 Optimal Recovery Revisited

As we saw in Chapter 18 we can interpret radial basis function interpolation as a constrained optimization problem, *i.e.*, the RBF interpolant automatically minimizes the native space norm among all interpolants in the native space. We now take this point of view again, but start with a more general formulation. Let us assume we are seeking a function $\mathcal{P}_f$ of the form

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{j=1}^{M} c_j \Phi(\boldsymbol{x}, \boldsymbol{x}_j), \qquad \boldsymbol{x} \in \mathbb{R}^s,$$

where the number M of basis functions is in general less than or equal the number N of data sites. We then want to determine the coefficients $\boldsymbol{c} = [c_1, \ldots, c_M]^T$ so that we minimize the quadratic form

$$\frac{1}{2}\boldsymbol{c}^T Q \boldsymbol{c} \tag{19.1}$$

with some symmetric positive definite matrix Q subject to the linear constraints

$$A\boldsymbol{c} = \boldsymbol{f} \tag{19.2}$$

where A is an $N \times M$ matrix with full rank, and the right-hand side $\boldsymbol{f} = [f_1, \ldots, f_N]^T$ is given. Such a constrained quadratic minimization problem can be converted to a system of linear equations by introducing *Lagrange multipliers* $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_N]^T$, *i.e.*, we consider finding the minimum of

$$\frac{1}{2}\boldsymbol{c}^T Q \boldsymbol{c} - \boldsymbol{\lambda}^T [A\boldsymbol{c} - \boldsymbol{f}] \tag{19.3}$$

with respect to $\boldsymbol{c}$ and $\boldsymbol{\lambda}$. Since Q is assumed to be a positive definite matrix, it is well known that the functional to be minimized is convex, and thus has a unique minimum. Therefore, the standard necessary condition for such a minimum (which is obtained by differentiating with respect to $\boldsymbol{c}$ and $\boldsymbol{\lambda}$ and finding the zeros of those derivatives) is also sufficient. This leads to

$$\begin{aligned} Q\boldsymbol{c} - A^T\boldsymbol{\lambda} &= \boldsymbol{0} \\ A\boldsymbol{c} - \boldsymbol{f} &= \boldsymbol{0} \end{aligned}$$

or, in matrix form,

$$\begin{bmatrix} Q & -A^T \\ A & O \end{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{f} \end{bmatrix}.$$

By applying (block) Gaussian elimination to this block matrix (Q is invertible since it is assumed to be positive definite) we get

$$\boldsymbol{\lambda} = \left(AQ^{-1}A^T\right)^{-1} \boldsymbol{f} \tag{19.4}$$

$$\boldsymbol{c} = Q^{-1}A^T \left(AQ^{-1}A^T\right)^{-1} \boldsymbol{f}. \tag{19.5}$$

In particular, if the quadratic form represents the native space norm of the interpolant $\mathcal{P}_f = \sum_{j=1}^{M} c_j \Phi(\cdot, \boldsymbol{x}_j)$, *i.e.*,

$$\|\mathcal{P}_f\|^2_{\mathcal{N}_\Phi(\Omega)} = \sum_{i=1}^{M}\sum_{j=1}^{M} c_i c_j \Phi(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{c}^T Q \boldsymbol{c}$$

with $Q_{ij} = \Phi(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\boldsymbol{c} = [c_1, \ldots, c_M]^T$, and the linear side conditions are the interpolation conditions

$$A\boldsymbol{c} = \boldsymbol{f} \qquad \Longleftrightarrow \qquad \mathcal{P}_f(\boldsymbol{x}_i) = f_i, \quad i = 1, \ldots, M,$$

with $A = A^T = Q$ (symmetric), the same $\boldsymbol{c}$ as above and data vector $\boldsymbol{f} = [f_1, \ldots, f_M]^T$, then we see that the Lagrange multipliers (19.4) become

$$\boldsymbol{\lambda} = A^{-1}\boldsymbol{f}$$

and the coefficients are given by

$$\boldsymbol{c} = \boldsymbol{\lambda}$$

via (19.5). Therefore, as we saw earlier, the minimum norm interpolant is obtained by solving the interpolation equations alone.

19.2 Regularized Least Squares Approximation

Since we took the more general point of view that $\mathcal{P}_f$ is generated by M basis functions, and N linear constraints are specified, the above formulation also covers both over- and under-determined least squares fitting where the quadratic form

$\boldsymbol{c}^T Q\boldsymbol{c}$ represents an added *smoothing* (or *regularization*) term. This term is not required to obtain a unique solution of the system $A\boldsymbol{c} = \boldsymbol{f}$ in the over-determined case ($N \geq M$), but in the under-determined case such a constraint is needed (*c.f.* the solution of under-determined linear systems via singular value decomposition in the numerical linear algebra literature (*e.g.*, [Trefethen and Bau (1997)])).

Usually the regularized least squares approximation problem is formulated as minimization of

$$\begin{aligned} &\frac{1}{2}\boldsymbol{c}^T Q\boldsymbol{c} + \omega \sum_{j=1}^{N} (\mathcal{P}_f(\boldsymbol{x}_j) - f_j)^2 \\ \Longleftrightarrow\ &\tfrac{1}{2}\boldsymbol{c}^T Q\boldsymbol{c} + \omega(A\boldsymbol{c} - \boldsymbol{f})^T(A\boldsymbol{c} - \boldsymbol{f}). \end{aligned} \tag{19.6}$$

The quadratic form $\boldsymbol{c}^T Q\boldsymbol{c}$ controls the smoothness of the fitting function and the least squares term measures the closeness to the data. The parameter ω controls the tradeoff between these two terms with a large value of ω shifting the balance toward increased pointwise accuracy.

The formulation (19.6) is used in *regularization theory* (see, *e.g.*, [Evgeniou *et al.* (2000); Girosi (1998)]). The same formulation is also used in *penalized least squares* fitting (see, *e.g.*, [von Golitschek and Schumaker (1990)]), the literature on *smoothing splines* [Reinsch (1967); Schoenberg (1964)], and in papers by Wahba on thin plate splines (*e.g.*, [Kimeldorf and Wahba (1971); Wahba (1979); Wahba (1990b); Wahba and Luo (1997); Wahba and Wendelberger (1980)]). In fact, the idea of smoothing a data fitting process by this kind of formulation seems to go back to at least [Whittaker (1923)]. In practice a penalized least squares formulation is especially useful if the data f_i cannot be completely trusted, *i.e.*, they are contaminated by noise. The problem of minimizing (19.6) is also known as *ridge regression* in the statistics literature. The regularization parameter ω is usually chosen using generalized cross validation.

If we restrict ourselves to working with square symmetric systems, *i.e.*, $A = A^T$, and assume the smoothness functional is given by the native space norm, *i.e.*, $Q = A$, then we obtain the minimizer of the unconstrained quadratic functional (19.6) by solving the linear system

$$\left(A + \frac{1}{2\omega} I\right) \boldsymbol{c} = \boldsymbol{f} \tag{19.7}$$

which is the result of setting the derivative of (19.6) with respect to $\boldsymbol{c}$ equal to zero. Thus, ridge regression corresponds to a diagonal stabilization/regularization of the usual interpolation system $A\boldsymbol{c} = \boldsymbol{f}$. This approach is especially useful for smoothing of noisy data. We present an implementation of this method and some numerical examples below in Section 19.4.

19.3 Least Squares Approximation When RBF Centers Differ from Data Sites

We are now interested in the more general setting where we still sample the given function f on the set $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ of data sites, but now introduce a second set $\Xi = \{\boldsymbol{\xi}_i\}_{i=1}^M$ at which we center the basis functions. Usually we will have $M \leq N$, and the case $M = N$ with $\Xi = \mathcal{X}$ recovers the traditional interpolation setting discussed in earlier chapters. Therefore, we can let the RBF approximant be of the form

$$\mathcal{Q}_f(\boldsymbol{x}) = \sum_{j=1}^{M} c_j \Phi(\boldsymbol{x}, \boldsymbol{\xi}_j), \quad \boldsymbol{x} \in \mathbb{R}^s. \tag{19.8}$$

The coefficients c_j can be found as the least squares solution of $A\boldsymbol{c} = \boldsymbol{f}$, *i.e.*, by minimizing $\|\mathcal{Q}_f - f\|_2^2$, where the ℓ_2-norm

$$\|f\|_2^2 = \sum_{i=1}^{N} [f(\boldsymbol{x}_i)]^2, \qquad \boldsymbol{x}_i \in \mathcal{X},$$

is induced by the discrete inner product

$$\langle f, g \rangle = \sum_{i=1}^{N} f(\boldsymbol{x}_i) g(\boldsymbol{x}_i), \qquad \boldsymbol{x}_i \in \mathcal{X}. \tag{19.9}$$

This approximation problem has a unique solution if the (rectangular) *collocation matrix* A with entries

$$A_{jk} = \Phi(\boldsymbol{x}_j, \boldsymbol{\xi}_k), \quad j = 1, \ldots, N, \; k = 1, \ldots, M,$$

has full rank.

If the centers in Ξ are chosen to form a subset of the data locations $\mathcal{X}$, then A does have full rank provided the radial basis functions are selected according to our previous chapters on interpolation. This is true, since in this case A will have an $M \times M$ square submatrix which is non-singular (by virtue of being an *interpolation matrix*).

The over-determined linear system $A\boldsymbol{c} = \boldsymbol{f}$ which arises in the solution of the least squares problem can be solved using standard algorithms from numerical linear algebra such as QR or singular value decomposition. Therefore the MATLAB code for RBF least squares approximation is almost identical to that for interpolation.

Program 19.1 presents an example for least squares approximation in 2D. Now we define two sets of points: the data points (defined in lines 3 and 8), and the centers (defined in lines 4, 6 and 7). Note that we first load the centers since our data files `Data2D_1089h` and `Data2D_81u` contain a variable `dsites` which we want to use for our data sites. Loading the data sites first, and then the centers would lead to unwanted overwriting of the values in `dsites`. The solution of the least squares problem is computed on line 16 using backslash matrix left division (\ or `mldivide`) which automatically produces a least squares solution. The subroutines `PlotSurf` and `PlotError2D` are provided in Appendix C.

Program 19.1. RBFApproximation2D.m

```
% RBFApproximation2D
% Script that performs basic 2D RBF least squares approximation
% Calls on: DistanceMatrix, PlotSurf, PlotError2D
 1  rbf = @(e,r) exp(-(e*r).^2); ep = 1;
 2  testfunction = @(x,y) sinc(x).*sinc(y);
 3  N = 1089; gridtype = 'h';
 4  M = 81; grid2type = 'u';
 5  neval = 40;
    % Load centers
 6  name = sprintf('Data2D_%d%s',M,grid2type); load(name)
 7  ctrs = dsites;
    % Load data points
 8  name = sprintf('Data2D_%d%s',N,gridtype); load(name)
    % Compute distance matrix between data sites and centers
 9  DM_data = DistanceMatrix(dsites,ctrs);
    % Build collocation matrix
10  CM = rbf(ep,DM_data);
    % Create right-hand side vector, i.e.,
    % evaluate the test function at the data points.
11  rhs = testfunction(dsites(:,1),dsites(:,2));
    % Create neval-by-neval equally spaced evaluation
    % locations in the unit square
12  grid = linspace(0,1,neval); [xe,ye] = meshgrid(grid);
13  epoints = [xe(:) ye(:)];
    % Compute distance matrix between evaluation points and centers
14  DM_eval = DistanceMatrix(epoints,ctrs);
15  EM = rbf(ep,DM_eval);
    % Compute RBF least squares approximation
16  Pf = EM * (CM\rhs);
    % Compute exact solution, i.e., evaluate test
    % function on evaluation points
17  exact = testfunction(epoints(:,1),epoints(:,2));
    % Compute maximum error on evaluation grid
18  maxerr = norm(Pf-exact,inf);
    % Plots
19  figure; fview = [100,30]; % viewing angles for plot
20  caption = sprintf('%d data sites and %d centers',N,M);
21  title(caption);
22  plot(dsites(:,1),dsites(:,2),'bo',ctrs(:,1),ctrs(:,2),'r+');
23  PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview);
24  PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview);
```

Output from `RBFApproximation2D.m` is presented in Figure 19.1 and the top part of Figure 19.2.

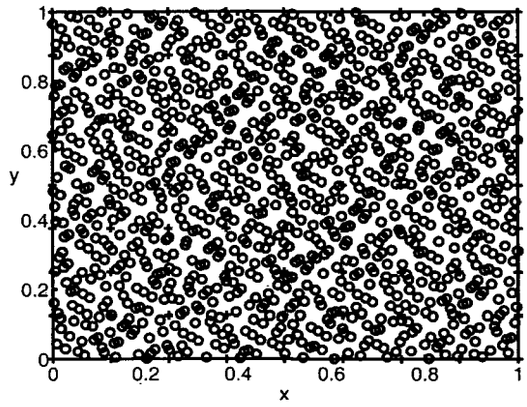


Fig. 19.1 1089 Halton data sites (o) and 81 uniform centers (+).

If $\varepsilon = 1$, then the coliocation matrix is rank deficient with MATLAB reporting a numerical rank of 58. In order to have a full numerical rank for this problem ε needs to be at least 2.2 (in which case the maximum error deteriorates to 5.255591e-004 instead of 2.173460e-007 for $\varepsilon = 1$, *c.f.* the top part of Figure 19.2). There is not much theory available for the case of differing centers and data sites. We present what is known in the next chapter. Some care needs to be taken when computing least squares solutions based on sets of differing centers and data sites.

19.4 Least Squares Smoothing of Noisy Data

We present two strategies for dealing with noisy data, *i.e.*, data that we consider to be not reliable due to, *e.g.*, measurement or transmission errors. This situation arises frequently in practice. We simulate a set of noisy data by sampling Franke's test function at a set $\mathcal{X}$ of data sites, and then adding uniformly distributed random noise of various strengths. For this experiment we use thin plate splines since their native space norm corresponds to the bending energy of a thin plate and thus they have a tendency to produce "visually pleasing" smooth and tight surfaces.

Since the thin plate splines have a singularity at the origin a little extra care needs to be taken with their implementation. The MATLAB script `tps.m` we use for our implementation of this basic function is included in Appendix C as Program C.4.

Our first strategy is to compute a straightforward least squares approximation to the (large) set of data using a (small) set of basis functions as we did in the previous section. In the statistics literature this approach is known as *regression splines*. We will not address the question of how to choose the centers for the basis functions at this point.

We use a modification of program `RBFApproximation2D.m` that allows us to use

thin plate splines with the added linear polynomial term. These changes can be found on lines 1, 15, 16, 19 and 24 of Program 19.2. Also, we now replace the sinc test function by Franke's function (2.2). The noise is added to the right-hand side of the linear system on line 18. This modification adds 3% noise to the data.

Program 19.2. RBFApproximation2Dlinear.m

```
% RBFApproximation2Dlinear
% Script that performs 2D RBF least squares approximation with
% linear reproduction for noisy data
% Calls on: tps, DistanceMatrix
 1  rbf = @tps; ep = 1;   % defined in tps.m (see Appendix C)
    % Define Franke's function as testfunction
 2  f1 = @(x,y) 0.75*exp(-((9*x-2).^2+(9*y-2).^2)/4);
 3  f2 = @(x,y) 0.75*exp(-((9*x+1).^2/49+(9*y+1).^2/10));
 4  f3 = @(x,y) 0.5*exp(-((9*x-7).^2+(9*y-3).^2)/4);
 5  f4 = @(x,y) 0.2*exp(-((9*x-4).^2+(9*y-7).^2));
 6  testfunction = @(x,y) f1(x,y)+f2(x,y)+f3(x,y)-f4(x,y);
 7  N = 1089; gridtype = 'h';
 8  M = 81; grid2type = 'u';
 9  neval = 40;
    % Load centers
10  name = sprintf('Data2D_%d%s',M,grid2type); load(name)
11  ctrs = dsites;
    % Load data points
12  name = sprintf('Data2D_%d%s',N,gridtype); load(name)
    % Compute distance matrix between data sites and centers
13  DM_data = DistanceMatrix(dsites,ctrs);
14  CM = rbf(ep,DM_data);   % Collocation matrix
    % Add extra columns and rows for linear reproduction
15  PM = [ones(N,1) dsites]; PtM = [ones(M,1) ctrs]';
16  CM = [CM PM; [PtM zeros(3,3)]];
    % Create right-hand side vector and add noise
17  rhs = testfunction(dsites(:,1),dsites(:,2));
18  rhs = rhs + 0.03*randn(size(rhs));
    % Add zeros for linear (2D) reproduction
19  rhs = [rhs; zeros(3,1)];
    % Create neval-by-neval equally spaced evaluation locations
    % in the unit square
20  grid = linspace(0,1,neval); [xe,ye] = meshgrid(grid);
21  epoints = [xe(:) ye(:)];
    % Compute distance matrix between evaluation points and centers
22  DM_eval = DistanceMatrix(epoints,ctrs);
```

```
23  EM = rbf(ep,DM_eval);   % Evaluation matrix
    % Add columns for linear reproduction
24  PM = [ones(neval^2,1) epoints]; EM = [EM PM];
    % Compute RBF least squares approximation
25  Pf = EM * (CM\rhs);
    % Compute exact solution, i.e.,
    % evaluate test function on evaluation points
26  exact = testfunction(epoints(:,1),epoints(:,2));
    % Compute maximum error on evaluation grid
27  maxerr = norm(Pf-exact,inf);
    % Plots
28  figure; fview = [160,20]; % viewing angles for plot
29  caption = sprintf('%d data sites and %d centers',N,M);
30  title(caption);
31  plot(dsites(:,1),dsites(:,2),'bo',ctrs(:,1),ctrs(:,2),'r+');
32  PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview);
33  PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview);
```

Program `RBFApproxiation2Dlinear.m` was used to produce the right plot in the bottom part of Figure 19.2 and the entries in line 2 of Table 19.1. Clearly, this simple least squares approach performs much better than straightforward interpolation to the noisy data (see the left plot the bottom part of Figure 19.2 and line 1 of Table 19.1). Moreover, this least squares approximation is also much cheaper to compute. However, as we pointed out earlier, it is not clear how to choose the smaller set of RBF centers, and what is even more unsettling, there is not much mathematical theory (see the next section) to guarantee if (or when) this approach is well-posed, *i.e.*, the collocation matrix has full rank.

Table 19.1 Errors and condition numbers for various least squares approximants to noisy data.

method	ω	RMS-error	max-error	cond(A)
Interpolation	oo	2.482624e-002	9.914837e-002	1.502900e+007
LSQ approximation	NA	9.665743e-003	5.490050e-002	NA
Ridge regression	1000	1.713843e-002	7.580288e-002	2.537652e+006
Ridge regression	100	1.078358e-002	4.215865e-002	3.839384e+005
Ridge regression	10	9.173961e-003	3.349371e-002	4.571167e+004
Ridge regression	1	2.764272e-002	1.041350e-001	9.317936e+003

Another strategy for smoothing of noisy data is the ridge regression method explained earlier (see (19.7)). This method is popular in the statistics and neural network community.

The nature of the MATLAB program for ridge regression is very similar to that for RBF interpolation. We present a version for ridge regression with thin plate splines

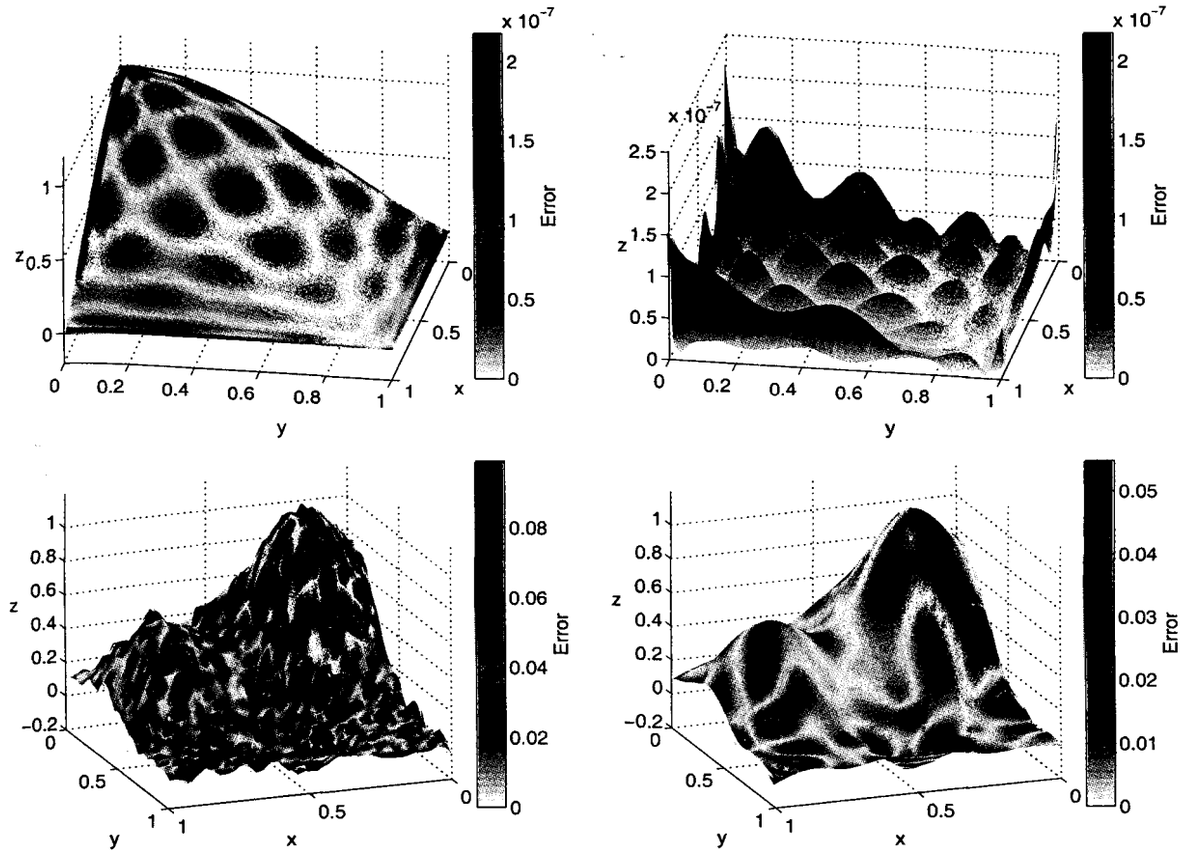


Fig. 19.2 Top: Least squares approximation (left) to 1089 data points sampled from 2D sinc function with 81 Gaussian basis functions with $\varepsilon = 1$ and maximum error (right) false-colored by magnitude of error. Bottom: Thin plate spline interpolant (left) to 1089 noisy data points sampled from Franke's function, and least squares approximation with 81 uniformly spaced thin plate spline basis functions (right) false-colored by magnitude of error.

(including the linear term in the basis expansion) for smoothing of noisy data. The smoothing parameter ω of (19.7) is defined on line 7 of Program 19.3, and the diagonal stabilization of the (interpolation) matrix A is performed on line 17. Note that the stabilization only affects the A part of the matrix, and not the extra rows and columns added for polynomial precision.

Program 19.3. TPS_RidgeRegression2D.m

```
% TPS_RidgeRegression2D
% Script that performs 2D TPS-RBF approximation with reproduction of
% linear functions and smoothing via ridge regression
% Calls on: tps, DistanceMatrix
    % Use TPS (defined in tps.m, see Appendix C)
 1  rbf = @tps; ep = 1;
    % Define Franke's function as testfunction
 2  fl = @(x,y) 0.75*exp(-((9*x-2).^2+(9*y-2).^2)/4);
```

```
 3  f2 = @(x,y) 0.75*exp(-((9*x+1).^2/49+(9*y+1).^2/10));
 4  f3 = @(x,y) 0.5*exp(-((9*x-7).^2+(9*y-3).^2)/4);
 5  f4 = @(x,y) 0.2*exp(-((9*x-4).^2+(9*y-7).^2));
 6  testfunction = @(x,y) f1(x,y)+f2(x,y)+f3(x,y)-f4(x,y);
 7  omega = 100;   % Smoothing parameter
 8  N = 1089; gridtype = 'h';
 9  neval = 40;
    % Load data points
10  name = sprintf('Data2D_%d%s',N,gridtype); load(name)
11  ctrs = dsites;
    % Compute distance matrix between data sites and centers
12  DM_data = DistanceMatrix(dsites,ctrs);
    % Create noisy right-hand side vector
13  rhs = testfunction(dsites(:,1),dsites(:,2));
14  rhs = rhs + 0.03*randn(size(rhs));
    % Add zeros for 2D linear reproduction
15  rhs = [rhs; zeros(3,1)];
    % Compute interpolation matrix and add diagonal regularization
16  IM = rbf(ep,DM_data);
17  IM = IM + eye(size(IM))/(2*omega);
    % Add extra columns and rows for linear reproduction
18  PM = [ones(N,1) dsites]; IM = [IM PM; [PM' zeros(3,3)]];
19  fprintf('Condition number estimate: %e\n',condest(IM))
    % Create neval-by-neval equally spaced evaluation locations
    % in the unit square
20  grid = linspace(0,1,neval); [xe,ye] = meshgrid(grid);
21  epoints = [xe(:) ye(:)];
    % Compute distance matrix between evaluation points and centers
22  DM_eval = DistanceMatrix(epoints,ctrs);
    % Compute evaluation matrix and add columns for linear precision
23  EM = rbf(ep,DM_eval);
24  PM = [ones(neval^2,1) epoints]; EM = [EM PM];
    % Compute RBF interpolant
25  Pf = EM * (IM\rhs);
    % Compute exact solution, i.e.,
    % evaluate test function on evaluation points
26  exact = testfunction(epoints(:,1),epoints(:,2));
    % Compute maximum error on evaluation grid
27  maxerr = norm(Pf-exact,inf);
    % Plots
28  fview = [160,20]; % viewing angles for plot
29  PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview);
```

```
30 PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview);
```

The results for our examples computed with Program 19.3 are shown in Figure 19.3 as well as in lines 3–6 of Table 19.1. These results illustrate very nicely the smoothing effect obtained by varying ω. A very large value of ω emphasizes the fitting component of the functional to be minimized in (19.6) resulting in a rather rough surface, while a small value of ω gives preference to the smoothing term. The "optimal" value of ω lies somewhere in the middle. In practice one would usually use cross validation to obtain the optimal value of ω.

Besides the visual smoothing of the approximating surface, a small value of ω also has a stabilizing effect on the collocation matrix. The diagonal of the matrix becomes more and more dominant. The condition estimates in Table 19.1 also verify this behavior.

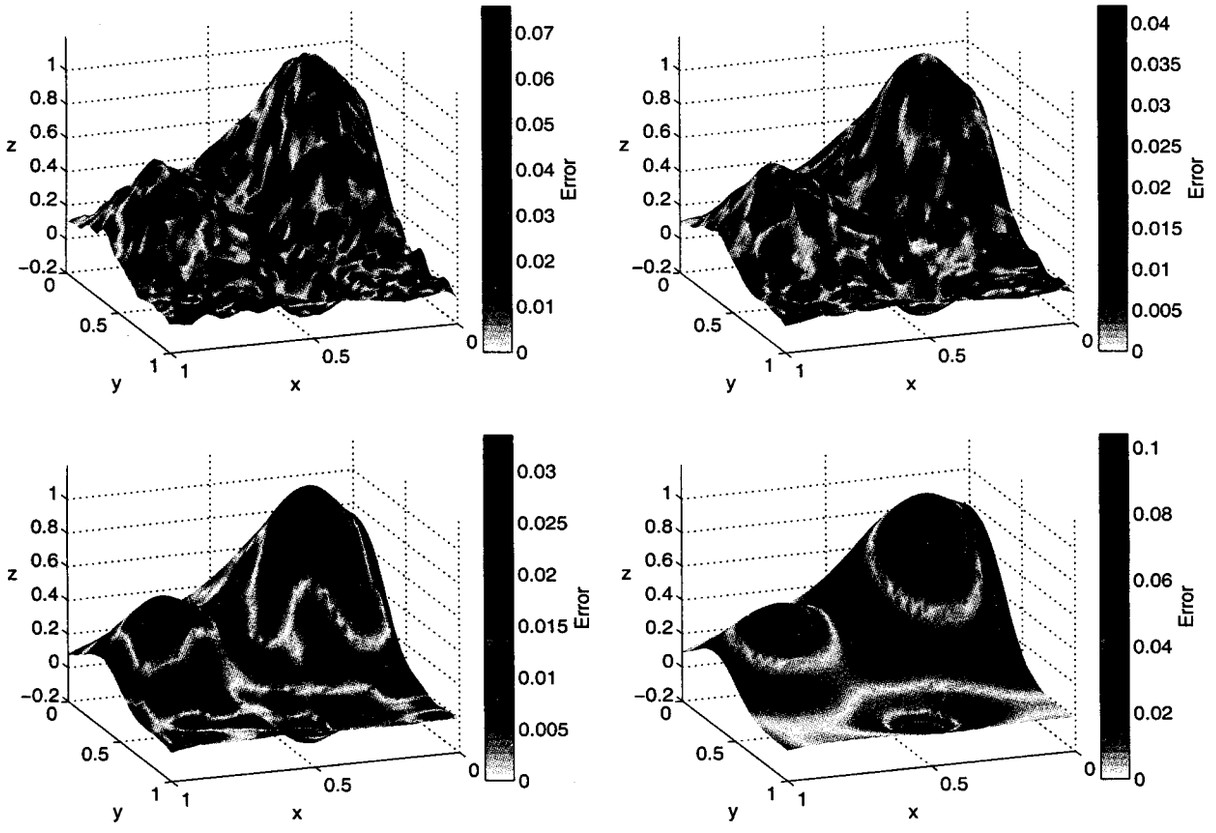


Fig. 19.3 Thin plate spline ridge regression to 1089 noisy data points sampled from Franke's function with $\omega = 1000$ (top left), $\omega = 100$ (top right), $\omega = 10$ (bottom left), and $\omega = 1$ (bottom right).

Chapter 20

Theory for Least Squares Approximation

In this chapter we give a brief account of the theoretical results known for least squares approximation with radial basis functions. These results include extensions of the RBF interpolation theory to cover well-posedness for the situation in which centers Ξ and data sites $\mathcal{X}$ differ. We also present some recent error estimates for least squares approximation.

20.1 Well-Posedness of RBF Least Squares Approximation

The results mentioned here are due to Quak, Sivakumar and Ward [Sivakumar and Ward (1993); Quak *et al.* (1993)]. The first paper deals with discrete, the second with continuous least squares approximation. In both papers the authors do not discuss the collocation matrix A we used in the previous chapter, but rather base their results on the non-singularity of the coefficient matrix obtained from a system of normal equations.

In the discrete setting they use the inner product (19.9) which induces the ℓ_2 norm, and then discuss non-singularity of the *Gram matrix* G that occurs in the following system of *normal equations*

$$G\boldsymbol{c} = \boldsymbol{w}, \tag{20.1}$$

where the entries of G are the ℓ_2 inner products of the radial basis functions, *i.e.*,

$$G_{jk} = \langle \Phi(\cdot, \boldsymbol{\xi}_j), \Phi(\cdot, \boldsymbol{\xi}_k) \rangle = \sum_{i=1}^{N} \Phi(\boldsymbol{x}_i, \boldsymbol{\xi}_j)\Phi(\boldsymbol{x}_i, \boldsymbol{\xi}_k), \quad j,k = 1, \ldots, M,$$

and the right-hand side vector $\boldsymbol{w}$ in (20.1) is given by

$$\boldsymbol{w}_j = \langle \Phi(\cdot, \boldsymbol{\xi}_j), \boldsymbol{f} \rangle = \sum_{i=1}^{N} \Phi(\boldsymbol{x}_i, \boldsymbol{\xi}_j) f(\boldsymbol{x}_i), \quad j = 1, \ldots, M.$$

Note that in the interpolation case with $M = N$ and $\Xi = \mathcal{X}$ (*i.e.*, coinciding centers and data sites) we have

$$\langle \Phi(\cdot, \boldsymbol{x}_j), \Phi(\cdot, \boldsymbol{x}_k) \rangle = \langle \Phi(\cdot, \boldsymbol{x}_j), \Phi(\cdot, \boldsymbol{x}_k) \rangle_{\mathcal{N}_\Phi(\Omega)} = \Phi(\boldsymbol{x}_j, \boldsymbol{x}_k)$$

so that G is just the interpolation matrix A. This provides yet another way of saying that the interpolation matrix A is also the system matrix for the normal equations in the case of best approximation with respect to the native space norm — a fact already mentioned earlier in Chapter 18 on optimal recovery.

In both papers, [Sivakumar and Ward (1993)] as well as [Quak *et al.* (1993)], even the formulation of the main theorems is very technical. We therefore just try to give a feel for their results.

Essentially, the authors show that the Gram matrix for certain radial basis functions (norm, (inverse) multiquadrics, and Gaussians) is non-singular if the centers $\Xi = \{\boldsymbol{\xi}_k, k = 1, \ldots, M\}$ are sufficiently well distributed and the data points $\mathcal{X} = \{\boldsymbol{x}_j, j = 1, \ldots, N\}$ are fairly evenly clustered about the centers with the diameter of the clusters being relatively small compared to the separation distance of the data points. Figure 20.1 illustrates this clustering idea.

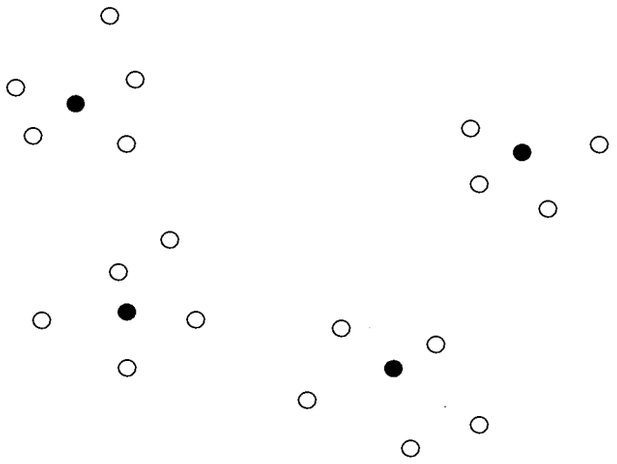


Fig. 20.1 Clusters of data points ○ around well separated centers ●.

One of the key ingredients in the proof of the non-singularity of G is to set up an interpolation matrix B for which the basis functions are centered at certain representatives of the clusters of knots about the data sites. One then splits the matrix B (which is non-symmetric in general) into a part that is symmetric and one that is anti-symmetric, a standard strategy in linear algebra, *i.e.*, $B = B_1 + B_2$ where B_1 and B_2 are defined by

$$B_1 = \frac{B + B^T}{2} \quad \text{(symmetric)},$$

$$B_2 = \frac{B - B^T}{2} \quad \text{(anti-symmetric)}.$$

Then, lower estimates for the norm of these two parts are found and used to conclude that, under certain restrictions, G is non-singular.

As a by-product of this argumentation the authors obtain a proof for the non-singularity of *interpolation* matrices for the case in which the centers of the basis

functions are chosen different from the data sites, namely as small perturbations thereof.

20.2 Error Bounds for Least Squares Approximation

In the case of basis functions centered at the points of an infinite lattice de Boor, DeVore and Ron [de Boor *et al.*(1994b); Ron (1992)] discussed L_2-approximation orders for radial basis functions.

More recently, [Ward (2004)] provided error bounds for least squares approximation at scattered centers and in finite domains. His work is closely linked to the results in [Narcowich *et al.* (2005)] discussed earlier in Chapter 15.

A typical result is that the continuous least squares error for approximation based on the compactly supported Wendland functions is

$$\min_{\mathcal{Q}_f \in H_\Phi(\mathcal{X})} \|f - \mathcal{Q}_f\|_{L_2(\Omega)} \leq C_{k,s,\Omega,\Phi} h^\tau \|f\|_{W_2^\tau(\Omega)}.$$

Here $\Omega \subset \mathbb{R}^s$ is a bounded Lipschitz domain which satisfies an interior cone condition, $\mathcal{X} \subset \Omega$ is the set of centers, $H_\Phi(\mathcal{X}) = \text{span}\{\Phi(\cdot - \boldsymbol{x}_j),\ \boldsymbol{x}_j \in \mathcal{X}\}$, k is such that $\tau = k + \sigma$ with $0 < \sigma \leq 1$, and $\tau > s/2$ measures the decay of the Fourier transform of Φ, *i.e.*,

$$c_1(1 + \|\boldsymbol{\omega}\|_2^2)^{-\tau} \leq \hat{\Phi}(\boldsymbol{\omega}) \leq c_2(1 + \|\boldsymbol{\omega}\|_2^2)^{-\tau}, \qquad \|\boldsymbol{\omega}\| \to \infty,\ \boldsymbol{\omega} \in \mathbb{R}^s,$$

with positive constants c_1 and c_2 so that the native space of Φ is given by the Sobolev space $W_2^\tau(\mathbb{R}^s)$. This error bound is of the same order as the one for interpolation (*c.f.* Theorem 15.3). We refer the reader to [Ward (2004)] for more details.

Chapter 21

Adaptive Least Squares Approximation

In this chapter we mention some strategies for solving the least squares problem in an adaptive fashion.

21.1 Adaptive Least Squares using Knot Insertion

A classical technique used to improve the quality of a given initial approximation based on a linear combination of certain basis functions is to adaptively increase the number of basis functions used for the fit. In other words, one refines the space from which the approximation is computed. Since every radial basis function is associated with one particular center (or *knot*), this can be achieved by adding new knots to the existing ones. This idea was explored for multiquadrics on $\mathbb{R}^2$ in [Franke *et al.* (1994); Franke *et al.* (1995)], and for radial basis functions on spheres in [Fasshauer (1995a)].

We will now describe an algorithm that adaptively adds knots to a radial basis function approximant in order to improve the ℓ_2 error.

Let us assume we are given a large number, N, of data and we want to fit them with a radial basis expansion to within a given tolerance. The idea is to start with very few initial knots, and then to repeatedly insert a knot at that data location whose ℓ_2 error component is largest. This is done as long as the least squares error exceeds a given tolerance. The following algorithm may be used.

Algorithm 21.1. Knot insertion

(1) Let data sites $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, data f_i, $i = 1, \ldots, N$, and a tolerance *tol* be given.
(2) Choose M initial knots $\Xi = \{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_M\}$.
(3) Calculate the least squares fit

$$\mathcal{Q}_f(\boldsymbol{x}) = \sum_{j=1}^{M} c_j \Phi(\boldsymbol{x}, \boldsymbol{\xi}_j)$$

with its associated error

$$e = \sum_{i=1}^{N} [f_i - \mathcal{Q}_f(\boldsymbol{x}_i)]^2.$$

While $e > tol$ **do**

(4) "Weight" each data point $\boldsymbol{x}_i$, $i = 1, \ldots, N$, according to its error component, *i.e.*, let

$$w_i = |f_i - \mathcal{Q}_f(\boldsymbol{x}_i)|, \quad i = 1, \ldots, N.$$

(5) Find the data point $\boldsymbol{x}_\nu \notin \Xi$ with maximum weight w_ν and insert it as a knot, *i.e.*,

$$\Xi = \Xi \cup \{\boldsymbol{x}_\nu\} \quad \text{and} \quad M = M + 1.$$

(6) Recalculate fit and associated error.

A MATLAB implementation of the knot insertion algorithm is provided in `RBFKnotInsert2D.m` (Program 21.1). This program is a little more technical than previous ones since we need to avoid adding the same point multiple times. This would lead to a singular system. In MATLAB we can easily check this with the command `ismember` (see line 28). We also take advantage of the `sort` command to help us find (possibly several) knots with largest error contribution. The addition of these data sites to the set of centers is accomplished on lines 26–32. Evaluation of the approximant (see lines 34–36) is not required until all of the knots have been inserted.

Program 21.1. `RBFKnotInsert2D.m`

```
% RBFKnotInsert2D
% Script that performs 2D RBF least squares approximation
% via knot insertion
% Calls on: DistanceMatrix
 1  rbf = @(e,r) exp(-(e*r).^2); ep = 5.5;
    % Define Franke's function as testfunction
 2  f1 = @(x,y) 0.75*exp(-((9*x-2).^2+(9*y-2).^2)/4);
 3  f2 = @(x,y) 0.75*exp(-((9*x+1).^2/49+(9*y+1).^2/10));
 4  f3 = @(x,y) 0.5*exp(-((9*x-7).^2+(9*y-3).^2)/4);
 5  f4 = @(x,y) 0.2*exp(-((9*x-4).^2+(9*y-7).^2));
 6  testfunction = @(x,y) f1(x,y)+f2(x,y)+f3(x,y)-f4(x,y);
 7  N = 289; gridtype = 'h';
 8  M = 1;  % Number of initial centers
 9  neval = 40;
10  grid = linspace(0,1,neval); [xe,ye] = meshgrid(grid);
11  epoints = [xe(:) ye(:)];
12  tol = 1e-5;  % Tolerance; stopping criterion
```

```
   % Load data points
13 name = sprintf('Data2D_%d%s',N,gridtype); load(name)
   % Take first M "data sites" as centers
14 ctrs = dsites(1:M,:);
   % Compute exact solution, i.e., evaluate test function
   % on evaluation points
15 exact = testfunction(epoints(:,1),epoints(:,2));
   % Create right-hand side vector, i.e.,
   % evaluate the test function at the data points.
16 rhs = testfunction(dsites(:,1),dsites(:,2));
17 rms_res = 999999;
18 while (rms_res > tol)
      % Compute least squares fit
19    DM_data = DistanceMatrix(dsites,ctrs);
20    CM = rbf(ep,DM_data);
21    coef = CM\rhs;
      % Compute residual
22    residual = abs(CM*coef - rhs);
23    [sresidual,idx] = sort(residual);
24    lres = length(residual);
25    rms_res = norm(residual)/sqrt(lres);
      % Add point(s)
26    if (rms_res > tol)
27       addpoint = idx(lres);   % This is the point we add
         % If already used, try next point
28       while any(ismember(ctrs,dsites(addpoint,:),'rows'))
29          lres = lres-1;  addpoint = idx(lres);
30       end
31       ctrs = [ctrs; dsites(addpoint,:)];
32    end
33 end
   % Compute evaluation matrix
34 DM_eval = DistanceMatrix(epoints,ctrs);
35 EM = rbf(ep,DM_eval);
36 Pf = EM*coef;   % Compute RBF least squares approximation
37 maxerr = max(abs(Pf - exact));  rms_err = norm(Pf-exact)/neval;
38 fprintf('RMS error:     %e\n', rms_err)
39 figure; % Plot data sites and centers
40 plot(dsites(:,1),dsites(:,2),'bo',ctrs(:,1),ctrs(:,2),'r+');
41 PlotSurf(xe,ye,Pf,neval,exact,maxerr,[160,20]);
```

We point out that we have to solve one linear least squares problem in each iteration. We do this using the standard MATLAB backslash (or `mldivide`) QR-

based solver (see line 21). The size of these problems increases at each step which means that addition of new knots becomes increasingly more expensive. This is usually not such a big deal. Both [Franke *et al.* (1994); Franke *et al.* (1995)] and [Fasshauer (1995a)] found that the desired accuracy was usually achieved with fairly few additional knots and thus the algorithm is quite fast.

If the initial knots are chosen to lie at data sites (as we did in our MATLAB implementation), then the collocation matrix A in the knot insertion algorithm will always have full rank. This is guaranteed since we only add data sites as new knots, and we make sure in step (5) of the algorithm that no multiple knots are created (which would obviously lead to a rank deficiency).

Instead of deciding which point to add based on residuals one could also pick the new point by looking at the power function, since the dependence of the approximation error on the data sites is encoded in the power function. This strategy is used to build so-called *greedy* adaptive algorithms that *interpolate* successively more and more data (see [Schaback and Wendland (2000a); Schaback and Wendland (2000b)] or Chapter 33). The power function is also employed in [De Marchi *et al.* (2005)] to compute an optimal set of RBF centers independent of the specific data values.

21.2 Adaptive Least Squares using Knot Removal

The idea of knot removal was primarily motivated by the need for data reduction, but it can also be used for the purpose of adaptive approximation (for a survey of knot removal see, *e.g.*, [Lyche (1992)]). The basic idea is to start with a good fit (*e.g.*, an interpolation to the data), and then successively reduce the number of knots used (and therefore basis functions) until a certain given tolerance is reached.

Specifically, this means we will start with an initial fit and then use some kind of weighting strategy for the knots, so that we can repeatedly remove those contributing least to the accuracy of the fit. The following algorithm was suggested in [Fasshauer (1995a)] for adaptive least squares approximation on spheres and performs this task.

Algorithm 21.2. Knot removal

(1) Let data points $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, data f_i, $i = 1, \ldots, N$, and a tolerance *tol* be given.
(2) Choose M initial knots $\Xi = \{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_M\}$.
(3) Calculate an initial fit

$$\mathcal{Q}_f(\boldsymbol{x}) = \sum_{j=1}^{M} c_j \Phi(\boldsymbol{x}, \boldsymbol{\xi}_j)$$

with its associated least squares error

$$e = \sum_{i=1}^{N} [f_i - \mathcal{Q}_f(\boldsymbol{x}_i)]^2.$$

While $e < tol$ **do**

(4) "Weight" each knot $\boldsymbol{\xi}_j$, $j = 1, \ldots, M$, according to its least squares error, *i.e.*, form

$$\Xi^* = \Xi \setminus \{\boldsymbol{\xi}_j\},$$

and calculate the weights

$$w_j = \sum_{i=1}^{N} \left[f_i - \mathcal{Q}_f^*(\boldsymbol{x}_i)\right]^2,$$

where

$$\mathcal{Q}_f^*(\boldsymbol{x}) = \sum_{j=1}^{M-1} c_j \Phi(\boldsymbol{x}, \boldsymbol{\xi}_j^*)$$

is the approximation based on the reduced set of knots Ξ^*.

(5) Find the knot $\boldsymbol{\xi}_\mu$ with lowest weight $w_\mu < tol$ and permanently remove it, *i.e.*,

$$\Xi = \Xi \setminus \{\boldsymbol{\xi}_\mu\} \quad \text{and} \quad M = M - 1.$$

(6) Recalculate fit and associated error.

We present a MATLAB implementation of a knot removal algorithm that is slightly more efficient. Its weighting strategy is based on the leave-one-out cross validation algorithm (see [Rippa (1999)] and Chapter 17). The code is given in `RBFKnotRemoval2D.m` (Program 21.2). This program is similar to the knot insertion program. In fact, it is a little simpler since we do not have to worry about multiple knots.

Program 21.2. `RBFKnotRemove2D.m`

```
% RBFKnotRemove2D
% Script that performs 2D RBF least squares approximation
% via knot removal
% Calls on: DistanceMatrix
 1  rbf = @(e,r) exp(-(e*r).^2); ep = 5.5;
    % Define Franke's function as testfunction
 2  f1 = @(x,y) 0.75*exp(-((9*x-2).^2+(9*y-2).^2)/4);
 3  f2 = @(x,y) 0.75*exp(-((9*x+1).^2/49+(9*y+1).^2/10));
 4  f3 = @(x,y) 0.5*exp(-((9*x-7).^2+(9*y-3).^2)/4);
 5  f4 = @(x,y) 0.2*exp(-((9*x-4).^2+(9*y-7).^2));
 6  testfunction = @(x,y) f1(x,y)+f2(x,y)+f3(x,y)-f4(x,y);
```

```
 7  N = 289; gridtype = 'h';
 8  M = 289;  % Number of initial centers
 9  neval = 40;
10  grid = linspace(0,1,neval); [xe,ye] = meshgrid(grid);
11  epoints = [xe(:) ye(:)];
12  tol = 5e-1;  % Tolerance; stopping criterion
    % Load data points
13  name = sprintf('Data2D_%d%s',N,gridtype); load(name)
    % Take first M "data sites" as centers
14  ctrs = dsites(1:M,:);
    % Compute exact solution, i.e., evaluate test function
    % on evaluation points
15  exact = testfunction(epoints(:,1),epoints(:,2));
    % Create right-hand side vector, i.e.,
    % evaluate the test function at the data points.
16  rhs = testfunction(dsites(:,1),dsites(:,2));
17  minres = 0;
18  while (minres < tol)
       % Compute collocation matrix
19     DM_data = DistanceMatrix(dsites,ctrs);
20     CM = rbf(ep,DM_data);
       % Compute residual
21     invCM = pinv(CM);  EF = (invCM*rhs)./diag(invCM);
22     residual = abs(EF);
23     [sresidual,idx] = sort(residual);  minres = residual(1);
       % Remove point
24     if (minres < tol)
25        ctrs = [ctrs(1:idx(1)-1,:); ctrs(idx(1)+1:M,:)];
26        M = M-1;
27     end
28  end
    % Evaluate final least squares fit
29  DM_data = DistanceMatrix(dsites,ctrs);
30  CM = rbf(ep,DM_data);
31  DM_eval = DistanceMatrix(epoints,ctrs);
32  EM = rbf(ep,DM_eval);
33  Pf = EM*(CM\rhs);
34  maxerr = max(abs(Pf - exact));  rms_err = norm(Pf-exact)/neval;
35  fprintf('RMS error:     %e\n', rms_err)
36  figure;   % Plot data sites and centers
37  plot(dsites(:,1),dsites(:,2),'bo',ctrs(:,1),ctrs(:,2),'r+');
38  caption = sprintf('%d data sites and %d centers', N, M);
```

```
39  title(caption);
40  PlotSurf(xe,ye,Pf,neval,exact,maxerr,[160,20]);
```

Again we would like to comment on the algorithm. As far as computational times are concerned, Algorithm 21.2 as listed above is *much* slower than the MATLAB implementation Program 21.2 based on the LOOCV idea since the weight for every knot is determined by the solution of a least squares problem, *i.e.*, in every iteration one needs to solve M least squares problems. The MATLAB program runs considerably faster, but usually it is still slower than the knot insertion algorithm. This is clear since with the knot removal strategy one starts with large problems that get successively smaller, whereas with knot insertion one begins with small problems that can be solved quickly.

The only way the knot removal approach will be beneficial is when the number of evaluations of the constructed approximant is much larger than its actual computation. This is so since, for comparable tolerances, one would expect knot removal to result in fewer knots than knot insertion. However, our examples show that this is not necessarily true.

If the initial knots are chosen at the data sites then, again, there will be no problems with the collocation matrix becoming rank deficient.

In [Fasshauer (1995a); Fasshauer (1995b)] some other alternatives to this knot removal strategy were considered. One of them is the removal of certain groups of knots at one time in order to speed up the process. Another is based on choosing the weights based on the size of the coefficients c_j in the expansion of $\mathcal{Q}_f$, *i.e.*, to remove that knot whose associated coefficient is smallest.

A further variation of the adaptive algorithms was considered in both [Franke *et al.* (1994)] and in [Fasshauer (1995a)]. Instead of treating only the coefficients of the expansion of $\mathcal{Q}_f$ as parameters in the minimization process, one can also include the knot locations themselves and possibly a (variable) shape parameter. This however, leads to *nonlinear* least squares problems. We will not discuss this topic further here.

Buhmann, Derrien, and Le Méhauté [Buhmann *et al.* (1995); Le Méhauté (1995)] also discuss knot removal. Their approach is based on an *a priori estimate* for the error made when removing a certain knot. These estimates depend on the specific choice of radial basis function, and only cover the inverse multiquadric type, *i.e.*,

$$\varphi(r) = (1 + r^2)^{-\beta}, \quad 0 < \beta \leq s/2.$$

Iske (see [Iske (1999a); Iske (2004)]) suggests an alternative knot removal strategy for least squares approximation. His removal heuristics are based on so-called *thinning algorithms*. In particular, in each iteration a point is removed if it belongs to a pair of points in Ξ with minimal separation distance. The thinning phase of the algorithm is then enhanced by an exchange phase in which points can be "swapped back in" if this process reduces the fill-distance of Ξ. This strategy maintains a relatively stable mesh ratio.

21.3 Some Numerical Examples

For the following examples we consider Franke's test function (2.2). All final fits are evaluated on a grid of 40×40 equally spaced points in the unit square. RMS and maximum errors are also computed on this grid. The programs `RBFKnotInsert2D.m` (Program 21.1) and `RBFKnotRemove2D.m` (Program 21.2) were used to compute the results.

In the top and middle parts of Figure 21.1 we compare the knot insertion and knot removal algorithms. In both cases we use a reference set of 289 Halton data sites and Gaussian basic functions scaled with a shape parameter $\varepsilon = 5.5$. Using the knot insertion algorithm with *tol* = 1e-004 as in Program 21.1 we select a subset of 154 data sites as centers for the basis functions. These points are displayed on the left side of the top part of Figure 21.1 along with the fit on the right. The knot insertion algorithm is initialized with a single random knot in the unit square.

Table 21.1 Total of knots selected, errors and runtimes for adaptive least squares approximants.

Method	N	M	RMS-error	max-error	time
Knot insert	289	154	1.334611e-003	2.871526e-002	2.66
Knot remove	289	153	1.424598e-003	3.961593e-002	35.09
Knot insert	4225	163	1.198695e-004	1.137886e-003	48.75

In the middle part of Figure 21.1 we show the results for the same configuration, *i.e.*, Gaussians with $\varepsilon = 5.5$ and $N = 289$ initial knots, for the knot removal algorithm. This time we take *tol* = 0.5, and the knot removal algorithm begins with an interpolant to all 289 data values. In Table 21.1 we can compare the errors and runtimes for the two algorithms. It is clear that the knot insertion algorithm is much more efficient for this example.

Another advantage of the knot insertion algorithm is revealed in the bottom part of Figure 21.1 and line 3 of Table 21.1. We still use Gaussians with shape parameter $\varepsilon = 5.5$. However, now we take $N = 4225$ Halton points as our data set. This provides many more candidates as centers for basis functions. The chosen centers are displayed on the left of the bottom part of Figure 21.1, and it is clear that this center selection is closely adapted to the features of the data function, namely the peaks and valley. The rest of the knots are located along the boundary of the domain. Moreover, we see that it is possible to obtain a much more accurate fit with roughly the same number of basis functions. While the runtime for this example is considerably longer than that for the other knot insertion example it is comparable to the time required for the knot removal algorithm with only 289 data sites. Running the knot removal algorithm with 4225 data sites would be prohibitively expensive.

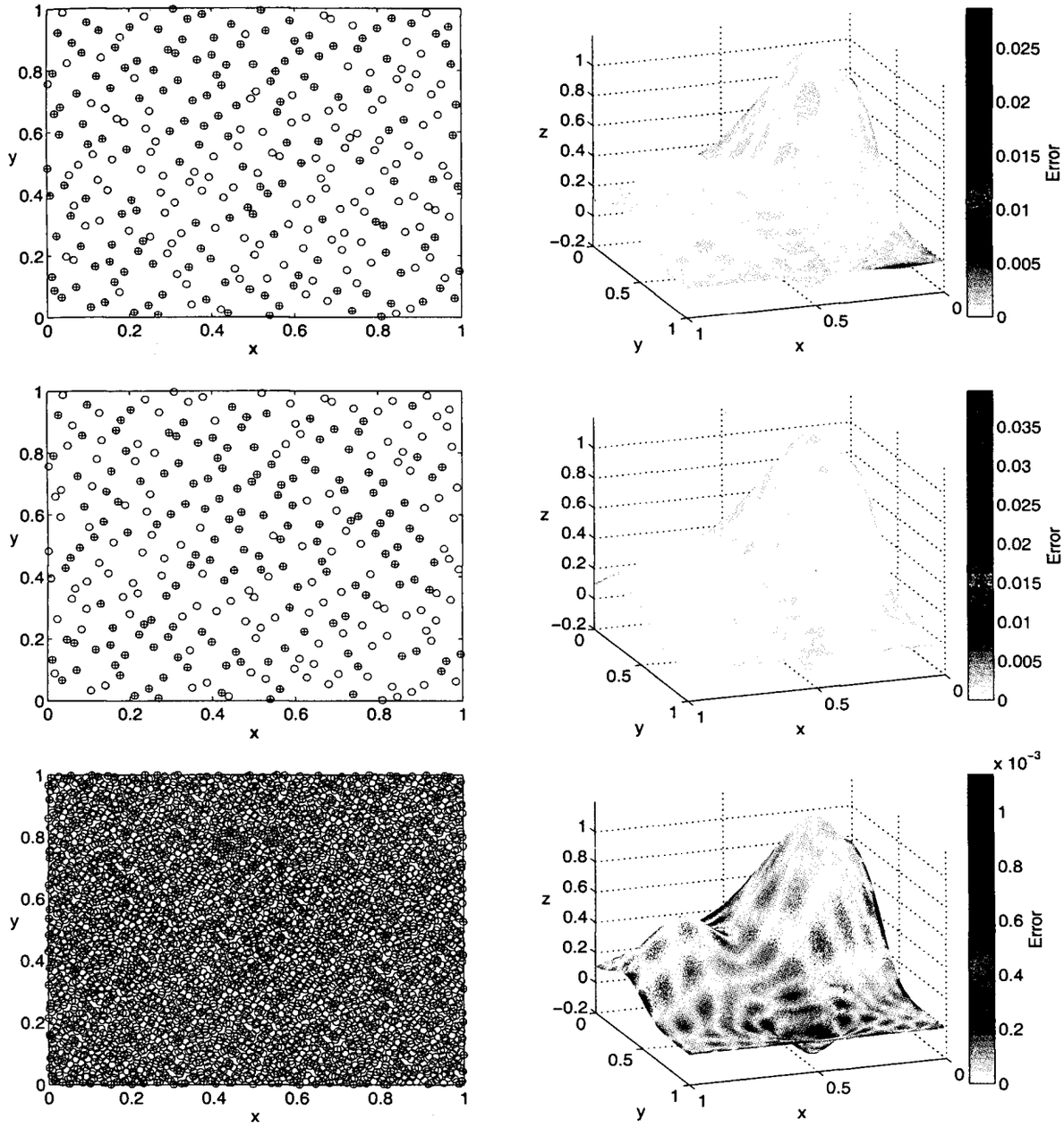


Fig. 21.1 Top: 289 data sites and 154 knots (left) for least squares approximation to Franke's function (right) using knot insertion algorithm with tol = 1e-004. Middle: 289 data sites and 153 knots (left) for least squares approximation to Franke's function (right) using knot removal algorithm with $tol = 0.5$. Bottom: 4225 data sites and 163 knots (left) for least squares approximation to Franke's function (right) using knot insertion algorithm with tol = 1e-004.

Chapter 22

Moving Least Squares Approximation

An alternative to radial basis function interpolation and approximation is the so-called *moving least squares* (MLS) method. As we will see below, in this method the approximation $\mathcal{P}_f$ to f is obtained by solving many (small) linear systems, instead of via solution of a single — but large — linear system as we did in the previous chapters.

22.1 Discrete Weighted Least Squares Approximation

In order to motivate the moving least squares method we begin by discussing discrete weighted least squares approximation from the space of multivariate polynomials. Thus, we consider data $(\boldsymbol{x}_i, f(\boldsymbol{x}_i))$, $i = 1, \ldots, N$, where $\boldsymbol{x}_i \in \Omega \subset \mathbb{R}^s$ and $f(\boldsymbol{x}_i) \in \mathbb{R}$ with arbitrary $s \geq 1$. The approximation space is of the form

$$\mathcal{U} = \text{span}\{p_1, \ldots, p_m\}, \qquad m < N,$$

with multivariate polynomials $p_m \in \Pi_d^s$ of degree at most d.

We intend to find the best discrete weighted least squares approximation from $\mathcal{U}$ to some given function f, *i.e.*, we need to determine the coefficients c_j in

$$u(\boldsymbol{x}) = \sum_{j=1}^{m} c_j p_j(\boldsymbol{x}), \qquad \boldsymbol{x} \in \mathbb{R}^s,$$

such that

$$\|f - u\|_{2,w} \rightarrow \min.$$

Here the norm is defined via the discrete (pseudo) inner product

$$\langle f, g \rangle_w = \sum_{i=1}^{N} f(\boldsymbol{x}_i) g(\boldsymbol{x}_i) w(\boldsymbol{x}_i)$$

with scalar weights $w_i = w(\boldsymbol{x}_i)$, $i = 1, \ldots, N$. The induced norm is then of the form

$$\|f\|_{2,w}^2 = \sum_{i=1}^{N} [f(\boldsymbol{x}_i)]^2 w(\boldsymbol{x}_i).$$

It is well known that the best approximation u from $\mathcal{U}$ to f is characterized by

$$\begin{aligned} f - u \perp_w \mathcal{U} &\iff \langle f - u, p_k \rangle_w = 0, \quad k = 1, \ldots, m, \\ &\iff \langle f - \sum_{j=1}^{m} c_j p_j, p_k \rangle_w = 0, \quad k = 1, \ldots, m, \\ &\iff \sum_{j=1}^{m} c_j \langle p_j, p_k \rangle_w = \langle f, p_k \rangle_w, \quad k = 1, \ldots, m, \\ &\iff G\boldsymbol{c} = \boldsymbol{f_p}. \end{aligned} \tag{22.1}$$

Here the Gram matrix G has entries $G_{jk} = \langle p_j, p_k \rangle_w$ and the right-hand side vector is $\boldsymbol{f_p} = [\langle f, p_1 \rangle_w, \ldots, \langle f, p_m \rangle_w]^T$. We refer to (22.1) as the *normal equations* associated with this problem.

Another way to think of this problem would be as a pure linear algebra problem. To this end, define the $N \times m$ matrix A with entries $A_{ij} = p_j(\boldsymbol{x}_i)$, and the vectors $\boldsymbol{c} = [c_1, \ldots, c_m]^T$ and $\boldsymbol{f} = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)]^T$. With this notation we seek a solution of the (overdetermined, since $N > m$) linear system $A\boldsymbol{c} = \boldsymbol{f}$. The standard weighted least squares solution is given by the solution of the normal equations $A^T W A\boldsymbol{c} = A^T W \boldsymbol{f}$, where W is the diagonal weighting matrix $W = \text{diag}(w_1, \ldots, w_N)$. This, however, is exactly what is written in (22.1), *i.e.*, the matrix G is of the form $G = A^T W A$, and for the right-hand side vector we have $\boldsymbol{f_p} = A^T W \boldsymbol{f}$.

22.2 Standard Interpretation of MLS Approximation

Several equivalent formulations exist for the moving least squares approximation scheme. In order to make a connection with the discussion of the discrete weighted least squares approximation just presented we start with the standard formulation of MLS approximation. The Backus-Gilbert formulation to be presented in the following section will have a closer connection to previous chapters since it corresponds to a linearly constrained quadratic minimization problem.

The general moving least squares method first appeared in the approximation theory literature in the paper [Lancaster and Šalkauskas (1981)] whose authors also pointed out the connection to the earlier more specialized work [Shepard (1968); McLain (1974)]. We now present a description of MLS approximation that is similar to the discussion in Lancaster and Šalkauskas' original paper and most closely resembles what is found in much of the other literature on MLS approximation.

We consider the following approximation problem. Assume we are given data values $f(\boldsymbol{x}_i)$, $i = 1, \ldots, N$, on some set $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \mathbb{R}^s$ of distinct data sites, where f is some (smooth) function, as well as an approximation space $\mathcal{U} = \text{span}\{u_1, \ldots, u_m\}$ with $m < N$. In addition, we define a weighted ℓ_2 inner product

$$\langle f, g \rangle_{w_{\boldsymbol{y}}} = \sum_{i=1}^{N} f(\boldsymbol{x}_i) g(\boldsymbol{x}_i) w(\boldsymbol{x}_i, \boldsymbol{y}), \qquad \boldsymbol{y} \in \mathbb{R}^s \text{ fixed}, \tag{22.2}$$

where now the weight functions $w_i = w(\boldsymbol{x}_i, \cdot)$, $i = 1, \ldots, N$, vary with the point $\boldsymbol{y}$. Note that the definition of this inner product naturally introduces a second variable, $\boldsymbol{y}$, into the discussion of the problem. This two-variable formulation of MLS approximation will be essential to understanding the connection between the various formulations.

As in the previous sections we wish to find the best approximation u from $\mathcal{U}$ to f. However, we focus our interest on best approximation *at the point* $\boldsymbol{y}$, *i.e.*, with respect to the norm induced by (22.2). In order to keep the discussion as simple as possible we will restrict our discussion to the multivariate polynomial case, *i.e.*, $\mathcal{U} = \mathbb{H}_d^s$ with basis $\{p_1, \ldots, p_m\}$. As always, the space Π_d^s of s-variate polynomials of degree d has dimension $m = \binom{s+d}{d}$. We emphasize, however, that everything that is said below also goes through for a more general linear approximation space $\mathcal{U}$.

Since we just introduced the second variable $\boldsymbol{y}$ into our formulation we will now look for the best approximation u in the form

$$u(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j=1}^{m} c_j(\boldsymbol{y}) p_j(\boldsymbol{x} - \boldsymbol{y}), \qquad \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^s. \tag{22.3}$$

We can think of $\boldsymbol{x}$ as the *global variable* and $\boldsymbol{y}$ as the *local variable*. Thus, expressing the polynomial basis functions in this form is reminiscent of a Taylor expansion. This shift to the local evaluation point $\boldsymbol{y}$ also adds stability to numerical computations.

For the purpose of final evaluation of our approximation we identify the global and the local variable, *i.e.*, we have

$$\mathcal{P}_f(\boldsymbol{x}) = u(\boldsymbol{x}, \boldsymbol{x}) = \sum_{j=1}^{m} c_j(\boldsymbol{x}) p_j(\boldsymbol{0}), \qquad \boldsymbol{x} \in \mathbb{R}^s. \tag{22.4}$$

Since for the polynomial approximation space Π_d^s with standard monomial basis we have $p_1(\boldsymbol{x}) \equiv 1$, and $p_j(\boldsymbol{0}) = \boldsymbol{0}$ for $j > 1$ we get the standard MLS approximation in the final form

$$\mathcal{P}_f(\boldsymbol{x}) = c_1(\boldsymbol{x}), \qquad \boldsymbol{x} \in \mathbb{R}^s. \tag{22.5}$$

Note, however, that $\boldsymbol{x}$ has been identified with the fixed local point $\boldsymbol{y}$, and therefore in general we still need to recompute the coefficient c_1 every time the evaluation point changes. Examples for some common choices of s and d will be provided in the next chapter.

As in the standard least squares case, the coefficients $c_j(\boldsymbol{y})$ in (22.3) are found by (locally) minimizing the weighted least squares error $\|f - u(\cdot, \boldsymbol{y})\|_{w_{\boldsymbol{y}}}$, *i.e.*,

$$\sum_{i=1}^{N} [f(\boldsymbol{x}_i) - u(\boldsymbol{x}_i, \boldsymbol{y})]^2 \, w(\boldsymbol{x}_i, \boldsymbol{y}) \tag{22.6}$$

is minimized over the coefficients in the expansion (22.3) of $u(\cdot, \boldsymbol{y})$. Note, however, that due to the definition of the inner product (22.2) whose weights "move" with

the local point $\boldsymbol{y}$, the coefficients c_j in (22.3) depend also on $\boldsymbol{y}$. As a consequence one has to solve the *normal equations*

$$\sum_{j=1}^{m} c_j(\boldsymbol{y})\langle p_j(\cdot-\boldsymbol{y}), p_k(\cdot-\boldsymbol{y})\rangle_{w_{\boldsymbol{y}}} = \langle f, p_k(\cdot-\boldsymbol{y})\rangle_{w_{\boldsymbol{y}}}, \qquad k=1,\ldots,m, \tag{22.7}$$

anew each time the point $\boldsymbol{y}$ is changed. In matrix notation (22.7) becomes

$$G(\boldsymbol{y})\mathrm{c}(\boldsymbol{y}) = \boldsymbol{f_p}(\boldsymbol{y}). \tag{22.8}$$

Here the positive definite Gram matrix $G(\boldsymbol{y})$ has entries

$$\begin{aligned} G(\boldsymbol{y})_{jk} &= \langle p_j(\cdot-\boldsymbol{y}), p_k(\cdot-\boldsymbol{y})\rangle_{w_{\boldsymbol{y}}} \\ &= \sum_{i=1}^{N} p_j(\boldsymbol{x}_i-\boldsymbol{y})p_k(\boldsymbol{x}_i-\boldsymbol{y})w(\boldsymbol{x}_i,\boldsymbol{y}), \end{aligned} \tag{22.9}$$

and the coefficient vector is of the form $\mathrm{c}(\boldsymbol{y}) = [c_1(\boldsymbol{y}),\ldots,c_m(\boldsymbol{y})]^T$. On the right-hand side of (22.8) we have the vector $\boldsymbol{f_p}(\boldsymbol{y}) = [\langle f, p_1(\cdot-\boldsymbol{y})\rangle_{w_{\boldsymbol{y}}},\ldots,\langle f, p_m(\cdot-\boldsymbol{y})\rangle_{w_{\boldsymbol{y}}}]^T$ of projections of the data onto the basis functions.

Several comments are called for. First, to ensure invertibility of the Gram matrix we need to impose a small restriction on the set $\mathcal{X}$ of data sites. Namely, $\mathcal{X}$ needs to be d-unisolvent (*c.f.* Definition 6.1). In this case the Gram matrix is symmetric and positive definite since the polynomial basis is linearly independent and the weights are positive. Second, the fact that the coefficients c_j depend on the point $\boldsymbol{y}$, and thus for every evaluation of $\mathcal{P}_f$ a Gram system (with different matrix $G(\boldsymbol{y})$) needs to be solved, initially scared people away from the moving least squares approach. However, for small values of m, *i.e.*, small polynomial degree d and small space dimensions s, it is possible to solve the Gram system analytically, and thus avoid solving linear systems altogether. We follow this approach and present some examples with explicit formulas in Chapter 23 and use them for our numerical experiments later. Moreover, if one chooses to use compactly supported weight functions, then only a few terms are "active" in the sum defining the entries of $G(\boldsymbol{y})$ (*c.f.* (22.9)).

22.3 The Backus-Gilbert Approach to MLS Approximation

The connection between the standard moving least squares formulation and Backus-Gilbert theory was pointed out in [Bos and Šalkauskas (1989)]. Mathematically, in the Backus-Gilbert approach one considers a *quasi-interpolant* of the form

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{i=1}^{N} f(\boldsymbol{x}_i)\Psi_i(\boldsymbol{x}), \tag{22.10}$$

where $\boldsymbol{f} = [f(\boldsymbol{x}_1),\ldots,f(\boldsymbol{x}_N)]^T$ represents the given data.

Quasi-interpolation is a generalization of the interpolation idea. If we use a linear function space $\text{span}\{\Phi_1, \ldots, \Phi_N\}$ to approximate given data $\{f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)\}$, then we saw earlier that we can determine coefficients $c_1, \ldots, c_N$ such that

$$u(\boldsymbol{x}) = \sum_{i=1}^{N} c_i \Phi_i(\boldsymbol{x})$$

interpolates the data, *i.e.*, $u(\boldsymbol{x}_i) = f(\boldsymbol{x}_i)$, $i = 1, \ldots, N$. In particular, if the basis functions Φ_i are cardinal functions, *i.e.*, $\Phi_i(\boldsymbol{x}_j) = \delta_{ij}$, $i, j = 1, \ldots, N$, then the coefficients are given by the data, *i.e.*, $c_i = f(\boldsymbol{x}_i)$, $i = 1, \ldots, N$.

Now, for a general quasi-interpolant we take *generating functions* Ψ_i, $i = 1, \ldots, N$ (which can be the same as the basis functions Φ_i) and form the expansion (22.10). This expansion will in general no longer interpolate the data, but it will represent some form of approximation. In order to ensure that a quasi-interpolant achieves a certain approximation power one usually requires that the generating functions reproduce polynomials of a certain degree. The same approach will be followed here, also (cf. (22.13)). The major advantage of quasi-interpolation over interpolation is the fact that we no longer have to solve a (potentially very large) system of linear equations to determine the coefficients c_j. Instead, they are given directly by the data. We will now discuss a scheme that tells us how to choose "good" generating functions Ψ_i.

As before, we consider the more general formulation

$$u(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{N} f(\boldsymbol{x}_i) \Psi_i(\boldsymbol{x}, \boldsymbol{y}),$$

with a global variable $\boldsymbol{x}$ and a local variable $\boldsymbol{y}$. To obtain the Backus-Gilbert approximant we identify $\boldsymbol{x}$ and $\boldsymbol{y}$, *i.e.*,

$$\mathcal{P}f(\boldsymbol{x}) = u(\boldsymbol{x}, \boldsymbol{x}) = \sum_{i=1}^{N} f(\boldsymbol{x}_i) K(\boldsymbol{x}_i, \boldsymbol{x}), \tag{22.11}$$

where we now introduced the notation $K(\boldsymbol{x}_i, \boldsymbol{x}) = \Psi_i(\boldsymbol{x}, \boldsymbol{x})$ with a kernel K.

From the discussion above and from Theorem 18.3 we know that the quasi-interpolant that minimizes the point-wise error is given if the generating functions $\Psi_i(\cdot, \boldsymbol{y})$ are cardinal functions (for fixed $\boldsymbol{y}$).

In the Backus-Gilbert formulation of the moving least squares method one does not attempt to minimize the pointwise error, but instead seeks — for a fixed reference point $\boldsymbol{y}$ — to find the *values* of the generating functions $\Psi_i(\boldsymbol{x}, \boldsymbol{y})$ at the fixed point $\boldsymbol{x}$ as the minimizers of

$$\frac{1}{2} \sum_{i=1}^{N} \Psi_i^2(\boldsymbol{x}, \boldsymbol{y}) \frac{1}{w(\boldsymbol{x}_i, \boldsymbol{y})} \tag{22.12}$$

subject to the polynomial reproduction constraints

$$\sum_{i=1}^{N} p(\boldsymbol{x}_i - \boldsymbol{y}) \Psi_i(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{x} - \boldsymbol{y}), \quad \text{for all } p \in \Pi_d^s, \tag{22.13}$$

where Π_d^s is the space of s-variate polynomials of total degree at most d with dimension $m = \binom{d+s}{d}$. If we again express the basis polynomials by $p_1, \ldots, p_m$, then we can reformulate (22.13) in matrix-vector form as

$$A(\boldsymbol{y})\Psi(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{p}(\boldsymbol{x} - \boldsymbol{y}), \tag{22.14}$$

where $A_{ji}(\boldsymbol{y}) = p_j(\boldsymbol{x}_i - \boldsymbol{y})$, $j = 1, \ldots, m$, $i = 1, \ldots, N$, $\Psi(\boldsymbol{x}, \boldsymbol{y}) = [\Psi_1(\boldsymbol{x}, \boldsymbol{y}), \ldots, \Psi_N(\boldsymbol{x}, \boldsymbol{y})]^T$ is the vector of values of the generating functions, and $\boldsymbol{p} = [p_1, \ldots, p_m]^T$ is the vector of basis polynomials. The corresponding matrix-vector formulation of (22.12) is

$$\frac{1}{2}\Psi^T(\boldsymbol{x}, \boldsymbol{y})Q(\boldsymbol{y})\Psi(\boldsymbol{x}, \boldsymbol{y}), \tag{22.15}$$

where

$$Q(\boldsymbol{y}) = \operatorname{diag}\left(\frac{1}{w(\boldsymbol{x}_1, \boldsymbol{y})}, \ldots, \frac{1}{w(\boldsymbol{x}_N, \boldsymbol{y})}\right), \tag{22.16}$$

and the $w(\boldsymbol{x}_i, \cdot)$ are positive *weight functions* (and thus for any fixed $\boldsymbol{y}$ the matrix $Q(\boldsymbol{y})$ is positive definite).

In the above formulation there is no explicit emphasis on nearness of fit as this is implicitly obtained by the quasi-interpolation "ansatz" and the closeness of the generating functions to the pointwise optimal delta functions. This is achieved by the above problem formulation if the $w(\boldsymbol{x}_i, \cdot)$ are weight functions that decrease with distance from the origin. The strictly positive definite radial functions used earlier are candidates for these weight functions. However, strict positive definiteness is not required at this point, so that, *e.g.*, (radial or tensor product) B-splines can also be used. As mentioned earlier, the polynomial reproduction constraint is added so that the quasi-interpolant will achieve a desired approximation order. This will become clear in Chapter 25.

In pure linear algebra notation the Backus-Gilbert approach corresponds to finding the minimum norm solution of an underdetermined linear system, *i.e.*, we want to solve the polynomial reproduction constraints

$$A(\boldsymbol{y})\Psi(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{p}(\boldsymbol{x} - \boldsymbol{y})$$

with $m \times N$ ($m < N$) system matrix. The norm of the solution vector is a weighted norm that varies with the (fixed) reference point $\boldsymbol{y}$ and is measured as in (22.15). In other words, the Backus-Gilbert formulation guarantees that we find the "best" system of generating functions with local polynomial reproduction properties, where "best" is measured in terms of the norm (22.15).

The quadratic form (22.12) (or equivalently (22.15)) can also be interpreted as a smoothness functional. Its use is also motivated by practical applications. In the Backus-Gilbert theory, which was developed in the context of geophysics (see [Backus and Gilbert (1968)]), it is desired that the generating functions Ψ_i are as close as possible to the ideal cardinal functions (*i.e.*, delta functions). Therefore, one needs to minimize their "spread". The polynomial reproduction constraints

are a generalization of an original normalization condition which corresponds to reproduction of constants only.

For any combination of a fixed (evaluation) point $\boldsymbol{x}$ and a fixed (reference) point $\boldsymbol{y}$ the combination of (22.12) and (22.13) (or equivalently (22.15) and (22.14)) present just another constrained quadratic minimization problem of the form discussed in previous chapters.

According to our earlier work we use Lagrange multipliers $\boldsymbol{\lambda}(\boldsymbol{x},\boldsymbol{y}) = [\lambda_1(\boldsymbol{x},\boldsymbol{y}),\ldots,\lambda_m(\boldsymbol{x},\boldsymbol{y})]^T$ (depending on $\boldsymbol{x}$ and $\boldsymbol{y}$), and then know that (*c.f.* (19.4) and (19.5))

$$\boldsymbol{\lambda}(\boldsymbol{x},\boldsymbol{y}) = \left(A(\boldsymbol{y})Q^{-1}(\boldsymbol{y})A^T(\boldsymbol{y})\right)^{-1}\boldsymbol{p}(\boldsymbol{x}-\boldsymbol{y}) \tag{22.17}$$

$$\Psi(\boldsymbol{x},\boldsymbol{y}) = Q^{-1}(\boldsymbol{y})A^T(\boldsymbol{y})\boldsymbol{\lambda}(\boldsymbol{x},\boldsymbol{y}). \tag{22.18}$$

Equation (22.18) tells us how to compute the generating functions for (22.11), *i.e.*, if we write (22.18) componentwise then

$$\Psi_i(\boldsymbol{x},\boldsymbol{y}) = w(\boldsymbol{x}_i,\boldsymbol{y})\sum_{j=1}^{m}\lambda_j(\boldsymbol{x},\boldsymbol{y})p_j(\boldsymbol{x}_i-\boldsymbol{y}), \qquad i=1,\ldots,N. \tag{22.19}$$

Therefore, once the values of the Lagrange multipliers $\lambda_j(\boldsymbol{x},\boldsymbol{y})$, $j=1,\ldots,N$, have been determined we have explicit formulas for the values of the generating functions. In particular, we get

$$\begin{aligned}\mathcal{P}_f(\boldsymbol{x}) = u(\boldsymbol{x},\boldsymbol{x}) &= \sum_{i=1}^{N} f(\boldsymbol{x}_i)\Psi_i(\boldsymbol{x},\boldsymbol{x}) \\ &= \sum_{i=1}^{N} f(\boldsymbol{x}_i)w(\boldsymbol{x}_i,\boldsymbol{x})\sum_{j=1}^{m}\lambda_j(\boldsymbol{x},\boldsymbol{x})p_j(\boldsymbol{x}_i-\boldsymbol{x}), \qquad i=1,\ldots,N, \\ &= \sum_{i=1}^{N} f(\boldsymbol{x}_i)K(\boldsymbol{x}_i,\boldsymbol{x})\end{aligned}$$

with kernels $K(\boldsymbol{x}_i,\boldsymbol{x}) = w(\boldsymbol{x}_i,\boldsymbol{x})\boldsymbol{\lambda}^T(\boldsymbol{x},\boldsymbol{x})\boldsymbol{p}(\boldsymbol{x}_i-\boldsymbol{x})$.

In general, finding the Lagrange multipliers involves solving a (small) linear system that changes as soon as the reference point $\boldsymbol{y}$ changes (see (22.17)). Using equation (22.17), the Lagrange multipliers are obtained as the solution of a Gram system

$$G(\boldsymbol{y})\boldsymbol{\lambda}(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{p}(\boldsymbol{x}-\boldsymbol{y}), \tag{22.20}$$

where the entries of the $m\times m$ matrix $G(\boldsymbol{y})$ are the weighted ℓ_2 inner products

$$G_{jk}(\boldsymbol{y}) = \langle p_j(\cdot-\boldsymbol{y}), p_k(\cdot-\boldsymbol{y})\rangle_{w_{\boldsymbol{y}}} = \sum_{i=1}^{N} p_j(\boldsymbol{x}_i-\boldsymbol{y})p_k(\boldsymbol{x}_i-\boldsymbol{y})w(\boldsymbol{x}_i,\boldsymbol{y}). \tag{22.21}$$

Note that this is identical to the matrix $G(\boldsymbol{y})$ needed for the determination of the coefficients $\boldsymbol{c}(\boldsymbol{y})$ in the standard MLS approach (*c.f.* (22.8) and (22.9)). Equation (22.20) shows us that — for a fixed reference point $\boldsymbol{y}$ — the Lagrange multipliers are polynomials, *i.e.*, $\lambda_j(\cdot,\boldsymbol{y})$, $j=1,\ldots,m$, are polynomials.

22.4 Equivalence of the Two Formulations of MLS Approximation

We now show that the two formulations of moving least squares approximation described in the previous two sections are equivalent, *i.e.*, we show that $\mathcal{P}_f(\boldsymbol{x})$ computed via (22.4) and (22.11) are the same. The approximant in the standard moving least squares formulation (22.4) establishes $u(\boldsymbol{x}, \boldsymbol{y})$ in the form

$$u(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j=1}^{m} c_j(\boldsymbol{y}) p_j(\boldsymbol{x} - \boldsymbol{y}) = \boldsymbol{p}^T(\boldsymbol{x} - \boldsymbol{y})\boldsymbol{c}(\boldsymbol{y}),$$

where $\boldsymbol{p} = [p_1, \ldots, p_m]^T$ and $\boldsymbol{c} = [c_1, \ldots, c_m]^T$.

In (22.8) we wrote the normal equations for this approach as

$$G(\boldsymbol{y})\boldsymbol{c}(\boldsymbol{y}) = \boldsymbol{f}_{\boldsymbol{p}}(\boldsymbol{y}).$$

Note that the right-hand side vector $\boldsymbol{f}_{\boldsymbol{p}}(\boldsymbol{y})$ can be written as

$$\begin{aligned} \boldsymbol{f}_{\boldsymbol{p}}(\boldsymbol{y}) &= [\langle f, p_1(\cdot - \boldsymbol{y})\rangle_{w_{\boldsymbol{y}}}, \ldots, \langle f, p_m(\cdot - \boldsymbol{y})\rangle_{w_{\boldsymbol{y}}}]^T \\ &= A(\boldsymbol{y})Q^{-1}(\boldsymbol{y})\boldsymbol{f} \end{aligned} \tag{22.22}$$

with the matrix $Q^{-1}(\boldsymbol{y})$ used in the Backus-Gilbert formulation. This implies

$$\boldsymbol{c}(\boldsymbol{y}) = G^{-1}(\boldsymbol{y})A(\boldsymbol{y})Q^{-1}(\boldsymbol{y})\boldsymbol{f}.$$

Thus, using the standard approach, we get

$$u(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{p}^T(\boldsymbol{x} - \boldsymbol{y})\boldsymbol{c}(\boldsymbol{y}) = \boldsymbol{p}^T(\boldsymbol{x} - \boldsymbol{y})G^{-1}(\boldsymbol{y})A(\boldsymbol{y})Q^{-1}(\boldsymbol{y})\boldsymbol{f}. \tag{22.23}$$

For the Backus-Gilbert approach, on the other hand, the "ansatz" is of the form (22.11)

$$u(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{N} f(\boldsymbol{x}_i)\Psi_i(\boldsymbol{x}, \boldsymbol{y}) = \Psi^T(\boldsymbol{x}, \boldsymbol{y})\boldsymbol{f},$$

where as before $\Psi(\boldsymbol{x}, \boldsymbol{y}) = [\Psi_1(\boldsymbol{x}, \boldsymbol{y}), \ldots, \Psi_N(\boldsymbol{x}, \boldsymbol{y})]^T$ and $\boldsymbol{f} = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)]^T$. For this approach we derived (see (22.17) and (22.18))

$$\begin{aligned} \boldsymbol{\lambda}(\boldsymbol{x}, \boldsymbol{y}) &= G^{-1}(\boldsymbol{y})\boldsymbol{p}(\boldsymbol{x} - \boldsymbol{y}) \\ \Psi(\boldsymbol{x}, \boldsymbol{y}) &= Q^{-1}(\boldsymbol{y})A^T(\boldsymbol{y})\boldsymbol{\lambda}(\boldsymbol{x}, \boldsymbol{y}), \end{aligned}$$

where $G(\boldsymbol{y}) = A(\boldsymbol{y})Q^{-1}(\boldsymbol{y})A^T(\boldsymbol{y})$ (see (22.21) or (22.9)). Therefore, we now obtain for the Backus-Gilbert approximant

$$u(\boldsymbol{x}, \boldsymbol{y}) = \Psi^T(\boldsymbol{x}, \boldsymbol{y})\boldsymbol{f} = \left[Q^{-1}(\boldsymbol{y})A^T(\boldsymbol{y})G^{-1}(\boldsymbol{y})\boldsymbol{p}(\boldsymbol{x} - \boldsymbol{y})\right]^T \boldsymbol{f}$$

which, by the symmetry of $Q(\boldsymbol{y})$ and $G(\boldsymbol{y})$, is the same as (22.23). Clearly, we also have equivalence when we evaluate either representation at $\boldsymbol{y} = \boldsymbol{x}$, *i.e.*, consider $\mathcal{P}_f(\boldsymbol{x}) = u(\boldsymbol{x}, \boldsymbol{x})$.

The equivalence of the two approaches shows that the moving least squares approximant has all of the following properties:

- It reproduces any polynomial of degree at most d in s variables exactly.

- It produces the best locally weighted least squares fit.
- Viewed as a quasi-interpolant, the generating functions Ψ_i are as close as possible to the optimal cardinal basis functions among all functions that produce polynomials of the desired degree in the sense that (22.12) is minimized.
- Since polynomials are infinitely smooth, either of the representations of $\mathcal{P}_f$ shows that its smoothness is determined by the smoothness of the weight function(s) $w_i = w(\boldsymbol{x}_i, \cdot)$.

In particular, the standard moving least squares method will reproduce the polynomial basis functions $p_1, \ldots, p_m$ even though this is not explicitly enforced by the minimization (solution of the normal equations). Moreover, the more general "ansatz" with (non-polynomial) approximation space $\mathcal{U}$ allows us to build moving least squares approximations that also reproduce any other function that is included in $\mathcal{U}$. This can be very beneficial for the solution of partial differential equations with known singularities (see, *e.g.*, the papers [Babuška and Melenk (1997); Belytschko *et al.* (1996)]).

22.5 Duality and Bi-Orthogonal Bases

From the Backus-Gilbert formulation we know that

$$G(\boldsymbol{y})\boldsymbol{\lambda}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{p}(\boldsymbol{x} - \boldsymbol{y}) \quad \Longleftrightarrow \quad \boldsymbol{\lambda}(\boldsymbol{x}, \boldsymbol{y}) = G^{-1}(\boldsymbol{y})\boldsymbol{p}(\boldsymbol{x} - \boldsymbol{y}). \tag{22.24}$$

By taking multiple right-hand sides $\boldsymbol{p}(\boldsymbol{x} - \boldsymbol{y})$ with $\boldsymbol{x} \in \mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ we get

$$[\boldsymbol{\lambda}(\boldsymbol{x}_1, \boldsymbol{y}), \ldots, \boldsymbol{\lambda}(\boldsymbol{x}_N, \boldsymbol{y})] = G^{-1}(\boldsymbol{y})[\boldsymbol{p}(\boldsymbol{x}_1 - \boldsymbol{y}), \ldots, \boldsymbol{p}(\boldsymbol{x}_N - \boldsymbol{y})]$$

or

$$\Lambda(\boldsymbol{y}) = G^{-1}(\boldsymbol{y})A(\boldsymbol{y}), \tag{22.25}$$

where $A(\boldsymbol{y})$ is the polynomial matrix (22.14) from above and the $m \times N$ matrix $\Lambda(\boldsymbol{y})$ has entries $\Lambda_{ji}(\boldsymbol{y}) = \lambda_j(\boldsymbol{x}_i, \boldsymbol{y})$.

The standard MLS formulation, on the other hand, gives us (see (22.8) and (22.22))

$$G(\boldsymbol{y})\boldsymbol{c}(\boldsymbol{y}) = \boldsymbol{f}_{\boldsymbol{p}}(\boldsymbol{y}) \quad \Longleftrightarrow \quad \boldsymbol{c}(\boldsymbol{y}) = G^{-1}(\boldsymbol{y})A(\boldsymbol{y})Q^{-1}(\boldsymbol{y})\boldsymbol{f}. \tag{22.26}$$

By combining (22.25) with (22.26) we get

$$\boldsymbol{c}(\boldsymbol{y}) = G^{-1}(\boldsymbol{y})A(\boldsymbol{y})Q^{-1}(\boldsymbol{y})\boldsymbol{f} = \Lambda(\boldsymbol{y})Q^{-1}(\boldsymbol{y})\boldsymbol{f} = \boldsymbol{f}_{\boldsymbol{\lambda}}(\boldsymbol{y}),$$

where $\boldsymbol{f}_{\boldsymbol{\lambda}}(\boldsymbol{y})$ is defined analogously to $\boldsymbol{f}_{\boldsymbol{p}}(\boldsymbol{y})$ (*c.f.* (22.22)). Componentwise this gives us

$$c_j(\boldsymbol{y}) = \langle \boldsymbol{f}, \lambda_j(\cdot, \boldsymbol{y}) \rangle_{w_{\boldsymbol{y}}}, \qquad j = 1, \ldots, m,$$

and therefore,

$$u(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j=1}^{m} \langle f, \lambda_j(\cdot, \boldsymbol{y}) \rangle_{w_{\boldsymbol{y}}} p_j(\boldsymbol{x} - \boldsymbol{y}). \tag{22.27}$$

It is also possible to formulate the moving least squares method by using the Lagrange multipliers of the Backus-Gilbert approach as basis functions for the approximation space $\mathcal{U}$. Then, using the same argumentation as above, we end up with

$$u(\boldsymbol{x},\boldsymbol{y}) = \sum_{j=1}^{m} d_j(\boldsymbol{y})\lambda_j(\boldsymbol{x},\boldsymbol{y}) \tag{22.28}$$

with

$$d_j(\boldsymbol{y}) = \langle f, p_j(\cdot - \boldsymbol{y})\rangle_{w_{\boldsymbol{y}}}, \qquad j = 1,\ldots,m.$$

We can verify this by applying (22.20) and (22.22) to (22.23), *i.e.*,

$$u(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{p}^T(\boldsymbol{x}-\boldsymbol{y})G^{-1}(\boldsymbol{y})A(\boldsymbol{y})Q^{-1}(\boldsymbol{y})\boldsymbol{f}.$$

We then obtain

$$u(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{\lambda}^T(\boldsymbol{x},\boldsymbol{y})\boldsymbol{f}_{\boldsymbol{p}}(\boldsymbol{y}),$$

which corresponds to (22.28).

The calculations just presented show that the Lagrange multipliers form a basis that is *dual* to the polynomial basis. In particular, if we recall the MLS approximant in its standard representation

$$u(\boldsymbol{x},\boldsymbol{y}) = \sum_{j=1}^{m} \langle f, \lambda_j(\cdot,\boldsymbol{y})\rangle_{w_{\boldsymbol{y}}} p_j(\boldsymbol{x}-\boldsymbol{y})$$

and let $f = p_k(\cdot - \boldsymbol{y})$, then the polynomial reproduction property of the method ensures

$$\sum_{j=1}^{m} \langle p_k(\cdot - \boldsymbol{y}), \lambda_j(\cdot,\boldsymbol{y})\rangle_{w_{\boldsymbol{y}}} p_j(\boldsymbol{x}-\boldsymbol{y}) = p_k(\boldsymbol{x}-\boldsymbol{y}).$$

This, however, implies

$$\langle p_k(\cdot - \boldsymbol{y}), \lambda_j(\cdot,\boldsymbol{y})\rangle_{w_{\boldsymbol{y}}} = \delta_{jk}, \qquad j,k = 1,\ldots,m. \tag{22.29}$$

Therefore we have two sets of basis functions that are *bi-orthogonal* with respect to the special weighted inner product $\langle\cdot,\cdot\rangle_{w_{\boldsymbol{y}}}$ on the set $\mathcal{X}$. We will illustrate this duality in Chapter 24.

Earlier we derived the representation (*c.f.* (22.18))

$$\Psi(\boldsymbol{x},\boldsymbol{y}) = Q^{-1}(\boldsymbol{y})A^T(\boldsymbol{y})\boldsymbol{\lambda}(\boldsymbol{x},\boldsymbol{y})$$

for the generating functions in the Backus-Gilbert formulation. Since the Lagrange multipliers are given by $\boldsymbol{\lambda}(\boldsymbol{x},\boldsymbol{y}) = G^{-1}(\boldsymbol{y})\boldsymbol{p}(\boldsymbol{x}-\boldsymbol{y})$ (see (22.20)) we get

$$\Psi(\boldsymbol{x},\boldsymbol{y}) = Q^{-1}(\boldsymbol{y})A^T(\boldsymbol{y})G^{-1}(\boldsymbol{y})\boldsymbol{p}(\boldsymbol{x}-\boldsymbol{y}) = Q^{-1}(\boldsymbol{y})\Lambda^T(\boldsymbol{y})\boldsymbol{p}(\boldsymbol{x}-\boldsymbol{y})$$

due to (22.25). Thus, the Backus-Gilbert representation is given also by the dual representation

$$u(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{N} f(\boldsymbol{x}_i)w(\boldsymbol{x}_i,\boldsymbol{y})\boldsymbol{\lambda}^T(\boldsymbol{x}_i,\boldsymbol{y})\boldsymbol{p}(\boldsymbol{x}-\boldsymbol{y}). \tag{22.30}$$

By also considering the two dual expansions (22.28) and (22.30) we now have four alternative representations for the moving least squares quasi-interpolant. This is summarized in the following theorem.

Theorem 22.1. *Let $f : \Omega \to \mathbb{R}$ be some function whose values on the d-unisolvent set of points $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^N \subset \mathbb{R}^s$ are given as data. Let $p_1, \ldots, p_m$ be a basis for Π_d^s with $p_1(\boldsymbol{x}) \equiv 1$, let $\{w(\boldsymbol{x}_i, \cdot), \ldots, w(\boldsymbol{x}_N, \cdot)\}$ be a set of positive weight functions, and let λ_j, $j = 1, \ldots, m$, be the Lagrange multipliers defined by (22.17). Furthermore, consider the generating functions*

$$\Psi_i(\boldsymbol{x}, \boldsymbol{y}) = w(\boldsymbol{x}_i, \boldsymbol{y}) \sum_{j=1}^m \lambda_j(\boldsymbol{x}, \boldsymbol{y}) p_j(\boldsymbol{x}_i - \boldsymbol{y}), \qquad i = 1, \ldots, N,$$

or in dual form

$$\Psi_i(\boldsymbol{x}, \boldsymbol{y}) = w(\boldsymbol{x}_i, \boldsymbol{y}) \sum_{j=1}^m \lambda_j(\boldsymbol{x}_i, \boldsymbol{y}) p_j(\boldsymbol{x} - \boldsymbol{y}), \qquad i = 1, \ldots, N.$$

The best local least squares approximation to f on $\mathcal{X}$ in the sense of (22.6) is given by $\mathcal{P}_f(\boldsymbol{x}) = u(\boldsymbol{x}, \boldsymbol{x})$, where

$$\begin{aligned} u(\boldsymbol{x}, \boldsymbol{y}) &= \sum_{j=1}^m \langle f, \lambda_j(\cdot, \boldsymbol{y}) \rangle_{w_{\boldsymbol{y}}} p_j(\boldsymbol{x} - \boldsymbol{y}) \\ &= \sum_{j=1}^m \langle f, p_j(\cdot - \boldsymbol{y}) \rangle_{w_{\boldsymbol{y}}} \lambda_j(\boldsymbol{x}, \boldsymbol{y}) \\ &= \sum_{i=1}^N f(\boldsymbol{x}_i) \Psi_i(\boldsymbol{x}, \boldsymbol{y}). \end{aligned}$$

This results in the four representations

$$\begin{aligned} \mathcal{P}_f(\boldsymbol{x}) &= \sum_{j=1}^m \langle f, \lambda_j(\cdot, \boldsymbol{x}) \rangle_{w_{\boldsymbol{x}}} p_j(\boldsymbol{0}) = \langle f, \lambda_1(\cdot, \boldsymbol{x}) \rangle_{w_{\boldsymbol{x}}} = c_1(\boldsymbol{x}) \\ &= \sum_{j=1}^m \langle f, p_j(\cdot - \boldsymbol{x}) \rangle_{w_{\boldsymbol{x}}} \lambda_j(\boldsymbol{x}, \boldsymbol{x}) \\ &= \sum_{i=1}^N f(\boldsymbol{x}_i) w(\boldsymbol{x}_i, \boldsymbol{x}) \sum_{j=1}^m \lambda_j(\boldsymbol{x}, \boldsymbol{x}) p_j(\boldsymbol{x}_i - \boldsymbol{x}) \\ &= \sum_{i=1}^N f(\boldsymbol{x}_i) w(\boldsymbol{x}_i, \boldsymbol{x}) \sum_{j=1}^m \lambda_j(\boldsymbol{x}_i, \boldsymbol{x}) p_j(\boldsymbol{0}) = \sum_{i=1}^N f(\boldsymbol{x}_i) w(\boldsymbol{x}_i, \boldsymbol{x}) \lambda_1(\boldsymbol{x}_i, \boldsymbol{x}). \end{aligned}$$

Note that the first two expansions for $\mathcal{P}_f(\boldsymbol{x})$ in Theorem 22.1 can be viewed as generalizations of (finite) eigenfunction or Fourier series expansions.

22.6 Standard MLS Approximation as a Constrained Quadratic Optimization Problem

Finally, it is also possible to formulate the standard MLS approach as a constrained quadratic minimization problem. To this end we (artificially) introduce the quadratic functional

$$\begin{aligned} \boldsymbol{c}^T(\boldsymbol{y})G(\boldsymbol{y})\boldsymbol{c}(\boldsymbol{y}) &= \sum_{j=1}^{m}\sum_{k=1}^{m} c_j(\boldsymbol{y})c_k(\boldsymbol{y})G_{jk}(\boldsymbol{y}) \\ &= \sum_{j=1}^{m}\sum_{k=1}^{m} c_j(\boldsymbol{y})c_k(\boldsymbol{y})\langle p_j(\cdot-\boldsymbol{y}), p_k(\cdot-\boldsymbol{y})\rangle_{w_{\boldsymbol{y}}} \end{aligned}$$

which should be interpreted as the ($\boldsymbol{y}$-dependent) native space norm of the approximant $u(\boldsymbol{x},\boldsymbol{y}) = \sum_{j=1}^{m} c_j(\boldsymbol{y})p_j(\boldsymbol{x}-\boldsymbol{y})$. The Gram system (22.8) can be written in matrix-vector form as

$$G(\boldsymbol{y})\boldsymbol{c}(\boldsymbol{y}) = A(\boldsymbol{y})Q^{-1}(\boldsymbol{y})\boldsymbol{f}$$

where $Q(\boldsymbol{y})$ is the diagonal matrix of weight functions (22.16) and $A(\boldsymbol{y})$ is the matrix of polynomials (22.14) used earlier.

Minimization of the quadratic form subject to the linear side conditions is equivalent to minimization of (for fixed $\boldsymbol{y}$)

$$\frac{1}{2}\boldsymbol{c}^T(\boldsymbol{y})G(\boldsymbol{y})\boldsymbol{c}(\boldsymbol{y}) - \boldsymbol{\mu}^T(\boldsymbol{y})\left[G(\boldsymbol{y})\boldsymbol{c}(\boldsymbol{y}) - A(\boldsymbol{y})Q^{-1}(\boldsymbol{y})\boldsymbol{f}\right], \tag{22.31}$$

where $\boldsymbol{\mu}(\boldsymbol{y})$ is a vector of Lagrange multipliers.

The solution of the linear system resulting from the minimization problem (22.31) gives us

$$\begin{aligned} \boldsymbol{\mu}(\boldsymbol{y}) &= \left(G(\boldsymbol{y})G^{-1}(\boldsymbol{y})G^T(\boldsymbol{y})\right)^{-1} A(\boldsymbol{y})Q^{-1}(\boldsymbol{y})\boldsymbol{f} = G^{-T}(\boldsymbol{y})A(\boldsymbol{y})Q^{-1}(\boldsymbol{y})\boldsymbol{f} \\ \boldsymbol{c}(\boldsymbol{y}) &= G^{-1}(\boldsymbol{y})G^T(\boldsymbol{y})\boldsymbol{\mu}(\boldsymbol{y}) = \boldsymbol{\mu}(\boldsymbol{y}) \end{aligned}$$

so that — as in the case of radial basis function interpolation — the quadratic functional $\boldsymbol{c}^T(\boldsymbol{y})G(\boldsymbol{y})\boldsymbol{c}(\boldsymbol{y})$ is automatically minimized by solving only the Gram system $G(\boldsymbol{y})\boldsymbol{c}(\boldsymbol{y}) = \boldsymbol{f}_{\boldsymbol{p}}(\boldsymbol{y})$.

22.7 Remarks

In the statistics literature the moving least squares approach is known as *local (polynomial) regression*. Good sources of information are the book [Fan and Gijbels (1996)] and the review article [Cleveland and Loader (1996)] according to which the basic ideas of local regression can be traced back at least to work done in the late 19th century by [Gram (1883); Woolhouse (1870); De Forest (1873); De Forest (1874)]. In particular, in the statistics literature one learns that the use

of least squares approximation is justified when the data $f_1, \ldots, f_N$ are normally distributed, whereas, if the noise in the data is not Gaussian, then other criteria should be used. See, *e.g.*, the survey article [Cleveland and Loader (1996)] for more details.

We close by establishing a connection between the polynomial reproduction constraints and certain moment conditions. Recall the polynomial reproduction constraints (22.14) in the Backus-Gilbert formulation

$$A(\boldsymbol{y})\Psi(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{p}(\boldsymbol{x} - \boldsymbol{y}).$$

By setting $\boldsymbol{y} = \boldsymbol{x}$ we get

$$A(\boldsymbol{x})\Psi(\boldsymbol{x}, \boldsymbol{x}) = \boldsymbol{p}(0). \tag{22.32}$$

Since we defined the kernels $K(\boldsymbol{x}_i, \boldsymbol{x}) = \Psi_i(\boldsymbol{x}, \boldsymbol{x})$ earlier, and since we assume that the polynomials basis is such that $p_1(\boldsymbol{x}) \equiv 1$ and $p_j(0) = 0$ for $j > 1$ we get from (22.32)

$$\sum_{i=1}^{N} p_k(\boldsymbol{x}_i - \boldsymbol{x}) K(\boldsymbol{x}_i, \boldsymbol{x}) = \delta_{1k}, \qquad k = 1, \ldots, m.$$

This comprises a set of *discrete moment conditions* for the kernel K. Since there are only m conditions for the N kernel values $K(\boldsymbol{x}_i, \boldsymbol{x})$ at a fixed point $\boldsymbol{x}$, the kernel is not uniquely determined by these moment conditions. If, however, we add the least norm constraint from the Backus-Gilbert formulation, *i.e.*, we minimize

$$\frac{1}{2} \sum_{i=1}^{N} K^2(\boldsymbol{x}_i, \boldsymbol{x}) \frac{1}{w(\boldsymbol{x}_i, \boldsymbol{x})} = \frac{1}{2} \| K(\cdot, \boldsymbol{x}) \|^2_{1/w_{\boldsymbol{x}}},$$

then we get the standard minimum norm solution for underdetermined least squares problems. We also point out that we derived earlier (see Theorem 22.1) that the kernel $K(\cdot, \boldsymbol{x})$ is of the form

$$K(\cdot, \boldsymbol{x}) = \lambda_1(\cdot, \boldsymbol{x}) w(\cdot, \boldsymbol{x})$$

with polynomial term $\lambda_1(\cdot, \boldsymbol{x})$ and weight function $w(\cdot, \boldsymbol{x})$. Thus, we have a unique solution once the weight function $w(\cdot, \boldsymbol{x})$ has been chosen.

The interpretation of MLS approximation with the help of moment matrices is used in the engineering literature (see, *e.g.*, [Li and Liu (2002)]), and also plays an essential role when connecting moving least squares approximation to the more efficient concept of *approximate approximation* [Maz'ya and Schmidt (2001)]. For a discussion of approximate moving least squares approximation see [Fasshauer (2002c); Fasshauer (2002d); Fasshauer (2003); Fasshauer (2004)] or Chapter 26.

Chapter 23

Examples of MLS Generating Functions

23.1 Shepard's Method

The moving least squares method in the case $d = 0$ (and therefore $m = 1$) with $p_1(\boldsymbol{x}) \equiv 1$ is usually referred to as *Shepard's method* [Shepard (1968)]. In the statistics literature Shepard's method is also known as a *kernel method* (see, *e.g.*, the papers from the 1950s and 60s [Rosenblatt (1956); Parzen (1962); Nadaraya (1964); Watson (1964)]). Using our notation we have

$$\mathcal{P}_f(\boldsymbol{x}) = c_1(\boldsymbol{x}).$$

The Gram "matrix" (22.9) consists of only one element

$$G(\boldsymbol{x}) = \langle p_1(\cdot - \boldsymbol{x}), p_1(\cdot - \boldsymbol{x}) \rangle_{w_{\boldsymbol{x}}} = \sum_{i=1}^{N} w(\boldsymbol{x}_i, \boldsymbol{x})$$

so that

$$G(\boldsymbol{x})\boldsymbol{c}(\boldsymbol{x}) = \boldsymbol{f_p}(\boldsymbol{x})$$

implies

$$c_1(\boldsymbol{x}) = \frac{\sum_{i=1}^{N} f(\boldsymbol{x}_i) w(\boldsymbol{x}_i, \boldsymbol{x})}{\sum_{j=1}^{N} w(\boldsymbol{x}_j, \boldsymbol{x})}. \tag{23.1}$$

If we compare (23.1) with the Backus-Gilbert quasi-interpolation formulation (22.11) we immediately see that the generating functions Ψ_i are given by

$$\Psi_i(\boldsymbol{x}, \boldsymbol{x}) = \frac{w(\boldsymbol{x}_i, \boldsymbol{x})}{\sum_{j=1}^{N} w(\boldsymbol{x}_j, \boldsymbol{x})}. \tag{23.2}$$

Since

$$\sum_{i=1}^{N} \Psi_i(\boldsymbol{x}, \boldsymbol{x}) = \sum_{i=1}^{N} \frac{w(\boldsymbol{x}_i, \boldsymbol{x})}{\sum_{j=1}^{N} w(\boldsymbol{x}_j, \boldsymbol{x})} \equiv 1 \tag{23.3}$$

independent of $\boldsymbol{x}$, Shepard's method is also known as a *partition of unity* method.

The weight functions can take many forms. In practice one usually takes a single basic weight function w which is then shifted to the data sites. Often the basic weight function is also a radial function (in either the Euclidean or maximum norm). We will use radial functions as basic weights in our numerical experiments below, *i.e.*,

$$w(\boldsymbol{x}_i, \boldsymbol{x}) = w(\|\boldsymbol{x} - \boldsymbol{x}_i\|)$$

where w is one of our (strictly positive definite) radial basic functions. Both compactly supported and globally supported functions will be used.

The dual Shepard basis is defined by (see (22.24))

$$G(\boldsymbol{y})\boldsymbol{\lambda}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{p}(\boldsymbol{x} - \boldsymbol{y})$$

so that

$$\lambda_1(\boldsymbol{x}, \boldsymbol{x}) = \frac{1}{\sum_{i=1}^{N} w(\boldsymbol{x}_i, \boldsymbol{x})},$$

and (*c.f.* (22.28))

$$\mathcal{P}_f(\boldsymbol{x}) = d_1(\boldsymbol{x})\lambda_1(\boldsymbol{x}, \boldsymbol{x}) \tag{23.4}$$

with

$$d_1(\boldsymbol{x}) = \langle f, p_1(\cdot - \boldsymbol{x})\rangle_{w_{\boldsymbol{x}}} = \sum_{i=1}^{N} f(\boldsymbol{x}_i) w(\boldsymbol{x}_i, \boldsymbol{x}).$$

The explicit dual representation of the Shepard approximant is, of course, identical to (23.1).

The generating functions (22.19) are defined as

$$\Psi_i(\boldsymbol{x}, \boldsymbol{x}) = w(\boldsymbol{x}_i, \boldsymbol{x})\lambda_1(\boldsymbol{x}, \boldsymbol{x}) p_1(\boldsymbol{x}_i - \boldsymbol{x}) = \frac{w(\boldsymbol{x}_i, \boldsymbol{x})}{\sum_{j=1}^{N} w(\boldsymbol{x}_j, \boldsymbol{x})},$$

which matches our earlier expression (23.2). This once more gives rise to the well-known quasi-interpolation formula for Shepard's method

$$\begin{aligned} \mathcal{P}_f(\boldsymbol{x}) &= \sum_{i=1}^{N} f(\boldsymbol{x}_i)\Psi_i(\boldsymbol{x}, \boldsymbol{x}) \\ &= \sum_{i=1}^{N} f(\boldsymbol{x}_i) \frac{w(\boldsymbol{x}_i, \boldsymbol{x})}{\sum_{j=1}^{N} w(\boldsymbol{x}_j, \boldsymbol{x})}. \end{aligned}$$

We now also have bi-orthogonality of the basis and dual basis, *i.e.*,

$$\langle \lambda_1(\cdot, \boldsymbol{x}), p_1(\cdot - \boldsymbol{x})\rangle_{w_{\boldsymbol{x}}} = 1.$$

Indeed

$$\langle \lambda_1(\cdot, \boldsymbol{x}), p_1(\cdot - \boldsymbol{x}) \rangle_{w_{\boldsymbol{x}}} = \sum_{i=1}^{N} \lambda_1(\boldsymbol{x}_i, \boldsymbol{x}) w(\boldsymbol{x}_i, \boldsymbol{x}) = \sum_{i=1}^{N} \frac{w(\boldsymbol{x}_i, \boldsymbol{x})}{\sum_{j=1}^{N} w(\boldsymbol{x}_j, \boldsymbol{x})} = 1.$$

23.2 MLS Approximation with Nontrivial Polynomial Reproduction

While it is always possible to simply solve the local Gram systems (22.8) or (22.20) and therefore implicitly compute the generating functions Ψ_i, $i = 1, \ldots, N$, it is often of interest to have explicit formulas for Ψ_i. We saw in the previous section that in the case of reproduction of constants we arrive at Shepard's method which is valid independent of the space dimension. However, if the degree d of polynomial reproduction is nontrivial (*i.e.*, $d > 0$), then the size $m = \binom{d+s}{d}$ of the Gram systems, and therefore the resulting formulas for the generating functions will depend on the space dimension s.

We now present three examples with explicit formulas for the MLS generating functions.

To simplify the notation in the following examples we define the *moments*

$$\mu_{\boldsymbol{\alpha}} = \sum_{i=1}^{N} (\boldsymbol{x}_i - \boldsymbol{x})^{\boldsymbol{\alpha}} w(\boldsymbol{x}_i, \boldsymbol{x}), \qquad \boldsymbol{x} \in \mathbb{R}^s, \quad |\boldsymbol{\alpha}| \leq 2d,$$

with $\boldsymbol{\alpha}$ a multi-index. These moments arise as entries in the Gram matrix $G(\boldsymbol{y})$ if we use a monomial basis and identify the reference point $\boldsymbol{y}$ with the evaluation point $\boldsymbol{x}$.

Example 23.1. We take $s = 1$, $d = 1$, and therefore $m = 2$. The set of data sites is given by $\mathcal{X} = \{x_1, \ldots, x_N\}$. We choose the standard monomial basis

$$\mathcal{U} = \operatorname{span}\{p_1(x) = 1,\ p_2(x) = x\}.$$

Then the Gram matrix (22.20) is of the form

$$\begin{aligned} G(x) &= \begin{bmatrix} \langle p_1(\cdot - x), p_1(\cdot - x) \rangle_{w_x} & \langle p_1(\cdot - x), p_2(\cdot - x) \rangle_{w_x} \\ \langle p_2(\cdot - x), p_1(\cdot - x) \rangle_{w_x} & \langle p_2(\cdot - x), p_2(\cdot - x) \rangle_{w_x} \end{bmatrix} \\ &= \begin{bmatrix} \sum_{i=1}^{N} w(x_i, x) & \sum_{i=1}^{N} (x_i - x) w(x_i, x) \\ \sum_{i=1}^{N} (x_i - x) w(x_i, x) & \sum_{i=1}^{N} (x_i - x)^2 w(x_i, x) \end{bmatrix} \\ &= \begin{bmatrix} \mu_0 & \mu_1 \\ \mu_1 & \mu_2 \end{bmatrix}, \end{aligned}$$

and the right-hand side of the Gram system is given by

$$\boldsymbol{p}(x-x) = \begin{bmatrix} p_1(0) \\ p_2(0) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Therefore, solution of the Gram system via Cramer's rule immediately yields

$$\lambda_1(x,x) = \frac{\mu_2}{\mu_0\mu_2 - \mu_1^2},$$
$$\lambda_2(x,x) = \frac{\mu_1}{\mu_1^2 - \mu_0\mu_2},$$

and according to (22.19) the generating functions for the Backus-Gilbert quasi-interpolant (MLS approximant) (22.11) are given by

$$\Psi_i(x,x) = w(x_i,x)\left[\lambda_1(x,x) + \lambda_2(x,x)(x_i - x)\right], \qquad i = 1,\ldots,N,$$

where the $w(x_i,\cdot)$ are arbitrary (positive) weight functions.

Example 23.2. We remain in the univariate case ($s = 1$) but increase the degree of polynomial reproduction to $d = 2$ so that $m = 3$. Again we take the standard monomial basis, *i.e.*,

$$\mathcal{U} = \text{span}\{p_1(x) = 1,\ p_2(x) = x,\ p_3(x) = x^2\}.$$

The 3×3 Gram system

$$\begin{bmatrix} \mu_0 & \mu_1 & \mu_2 \\ \mu_1 & \mu_2 & \mu_3 \\ \mu_2 & \mu_3 & \mu_4 \end{bmatrix} \begin{bmatrix} \lambda_1(x,x) \\ \lambda_2(x,x) \\ \lambda_3(x,x) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

can then be solved analytically (*e.g.*, using Maple), and one obtains the Lagrange multipliers

$$\lambda_1(x,x) = \frac{\mu_2\mu_4 - \mu_3^2}{D},$$
$$\lambda_2(x,x) = \frac{\mu_2\mu_3 - \mu_1\mu_4}{D},$$
$$\lambda_3(x,x) = \frac{\mu_1\mu_3 - \mu_2^2}{D},$$

where $D = 2\mu_1\mu_2\mu_3 - \mu_0\mu_3^2 - \mu_2^3 - \mu_1^2\mu_4 + \mu_0\mu_2\mu_4$. Now the generating functions are given by

$$\Psi_i(x,x) = w(x_i,x)\left[\lambda_1(x,x) + \lambda_2(x,x)(x_i - x) + \lambda_3(x,x)(x_i - x)^2\right], \quad i = 1,\ldots,N.$$

Example 23.3. Reproduction of linear polynomials in 2D also leads to a 3×3 Gram system (since $s = 2$, $d = 1$, and therefore $m = 3$). Now the monomial basis is given by

$$\mathcal{U} = \text{span}\{p_1(x,y) = 1,\ p_2(x,y) = x,\ p_3(x,y) = y\},$$

where $\boldsymbol{x} = (x, y) \in \mathbb{R}^2$, and the Gram system looks like

$$\begin{bmatrix} \mu_{00} & \mu_{10} & \mu_{01} \\ \mu_{10} & \mu_{20} & \mu_{11} \\ \mu_{01} & \mu_{11} & \mu_{02} \end{bmatrix} \begin{bmatrix} \lambda_1(\boldsymbol{x}, \boldsymbol{x}) \\ \lambda_2(\boldsymbol{x}, \boldsymbol{x}) \\ \lambda_3(\boldsymbol{x}, \boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

The Lagrange multipliers in this case turn out to be

$$\lambda_1(\boldsymbol{x}, \boldsymbol{x}) = \frac{1}{D} \left[\mu_{11}^2 - \mu_{20}\mu_{02} \right],$$

$$\lambda_2(\boldsymbol{x}, \boldsymbol{x}) = \frac{1}{D} \left[\mu_{10}\mu_{02} - \mu_{01}\mu_{11} \right],$$

$$\lambda_3(\boldsymbol{x}, \boldsymbol{x}) = \frac{1}{D} \left[\mu_{20}\mu_{01} - \mu_{10}\mu_{11} \right],$$

with $D = \mu_{10}^2\mu_{02} + \mu_{20}\mu_{01}^2 - \mu_{00}\mu_{20}\mu_{02} - 2\mu_{10}\mu_{01}\mu_{11} + \mu_{00}\mu_{11}^2$. The generating functions Ψ_i, $i = 1, \ldots, N$, for this example are of the form

$$\Psi_i(\boldsymbol{x}, \boldsymbol{x}) = w(\boldsymbol{x}_i, \boldsymbol{x}) \left[\lambda_1(\boldsymbol{x}, \boldsymbol{x}) + \lambda_2(\boldsymbol{x}, \boldsymbol{x})(x_i - x) + \lambda_3(\boldsymbol{x}, \boldsymbol{x})(y_i - y) \right].$$

While we can use a symbolic manipulation program such as Maple to solve the Gram system analytically for other choices of d and s, it is clear that the expressions for the generating functions quickly become very unwieldy. It may be tolerable to continue this approach for a 4×4 Gram system corresponding to reproduction of linear polynomials in 3D (or cubic polynomials in 1D), but already the case of quadratic reproduction in 2D (with $m = 6$) is much too complex to print here, and reproduction of quadratics in 3D (or cubics in 2D) requires even 10 Lagrange multipliers.

Chapter 24

MLS Approximation with MATLAB

24.1 Approximation with Shepard's Method

In this section we investigate approximation with Shepard's method

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{i=1}^{N} f(\boldsymbol{x}_i) \frac{w(\boldsymbol{x}_i, \boldsymbol{x})}{\sum_{j=1}^{N} w(\boldsymbol{x}_j, \boldsymbol{x})}, \qquad \boldsymbol{x} \in \mathbb{R}^s.$$

We will look at three sets of experiments. The first two sets will employ global Gaussian weights, *i.e.*, $w(\boldsymbol{x}_i, \boldsymbol{x}) = e^{-\varepsilon^2 \|\boldsymbol{x}-\boldsymbol{x}_i\|^2}$, in $\mathbb{R}^2$, while the third set of experiments is based on Wendland's compactly supported C^2 weights $w(\boldsymbol{x}_i, \boldsymbol{x}) = (1 - \varepsilon\|\boldsymbol{x} - \boldsymbol{x}_i\|)_+^4 (4\varepsilon\|\boldsymbol{x} - \boldsymbol{x}_i\| + 1)$ in $\mathbb{R}^s$ for $s = 1, 2, \ldots, 6$. Note that the Wendland weights are strictly positive definite basic functions only for $s \leq 3$ so that it would not be advisable to use them for RBF interpolation in higher dimensions. For MLS approximation, however, strict positive definiteness is not required. Here we only ask that the weights be positive on their support.

In our 2D experiments we take Franke's function (2.2) as our test function and sample it at uniformly spaced points in the unit square. In Table 24.1 we list the RMS-errors and computed convergence rates for both stationary and non-stationary approximation. Clearly, non-stationary approximation does not converge. This behavior is *exactly opposite* the convergence behavior of Gaussians in the RBF interpolation setting. There we have convergence in the non-stationary setting, but not in the stationary setting (*c.f.* Table 17.5 and Figure 17.12). The initial shape parameter for the stationary setting (and the fixed value in the non-stationary setting) is $\varepsilon = 3$. This value is subsequently multiplied by a factor of two for each halving of the fill-distance. We can see that Shepard's method seems to have a stationary approximation order of at least $\mathcal{O}(h)$. This will be verified theoretically in the next chapter.

The first two stationary Shepard approximations (based on 9 and 25 points, respectively) are displayed in the top part of Figure 24.1. It is apparent that the Shepard method has a smoothing effect. In order to emphasize this feature we provide Shepard approximations to noisy data (with 3% noise added to the

Table 24.1 2D stationary and non-stationary Shepard approximation with Gaussian weight function.

	stationary		non-stationary	
N	RMS-error	rate	RMS-error	rate
9	1.835110e-001		1.835110e-001	
25	5.885159e-002	1.6407	1.303771e-001	0.4932
81	2.299502e-002	1.3558	1.311538e-001	−0.0086
289	6.726166e-003	1.7735	1.315894e-001	−0.0048
1089	2.113604e-003	1.6701	1.320564e-001	−0.0051
4225	8.065893e-004	1.3898	1.323576e-001	−0.0033

Franke data) in the bottom part of Figure 24.1. On the left we display the Shepard approximation based on Gaussian weights with $\varepsilon = 48$ (corresponding to the entry in line 5 of Table 24.1). The RMS-error for this approximation is 1.820897e-002. The plot on the right of the bottom part of Figure 24.1 is the result of reducing ε to 12, and thus increasing the smoothing effect since "wider" weight functions are used, *i.e.*, more neighboring data are incorporated into the local regression fit. The corresponding RMS-error is 2.481218e-002. Note that even though the RMS-error is larger for the plot on the right, the surface is visually smoother. These examples should be compared to the data smoothing experiments in Chapter 19.

The MATLAB code for the 2D experiments is `Shepard2D.m` listed as **Program 24.1**. It is a little simpler than the RBF interpolation code used earlier since we do not need to assemble an interpolation matrix and no linear system needs to be solved. The evaluation matrix is assembled in two steps. First, on line 14, we compute the standard RBF evaluation matrix. In order to implement the Shepard scaling we note that all of the entries in row i of the evaluation matrix are scaled with the same sum, $\sum_{j=1}^{N} w(\boldsymbol{x}_j, \boldsymbol{x}_i)$. Therefore, on line 16, we perform the extra Shepard scaling with the help of `repmat` and a vector of `ones` which gives us all the necessary sums in the denominator. For the example with noisy data we add

```
f = f + 0.03*randn(size(f));
```

after line 11.

Program 24.1. `Shepard2D.m`

```
% Shepard2D
% Script that performs 2D Shepard approximation with global weights
% Calls on: DistanceMatrix, PlotSurf, PlotError2D
 1  rbf = @(e,r) exp(-(e*r).^2); ep = 5.5;
    % Define Franke's function as testfunction
 2  f1 = @(x,y) 0.75*exp(-((9*x-2).^2+(9*y-2).^2)/4);
 3  f2 = @(x,y) 0.75*exp(-((9*x+1).^2/49+(9*y+1).^2/10));
 4  f3 = @(x,y) 0.5*exp(-((9*x-7).^2+(9*y-3).^2)/4);
```

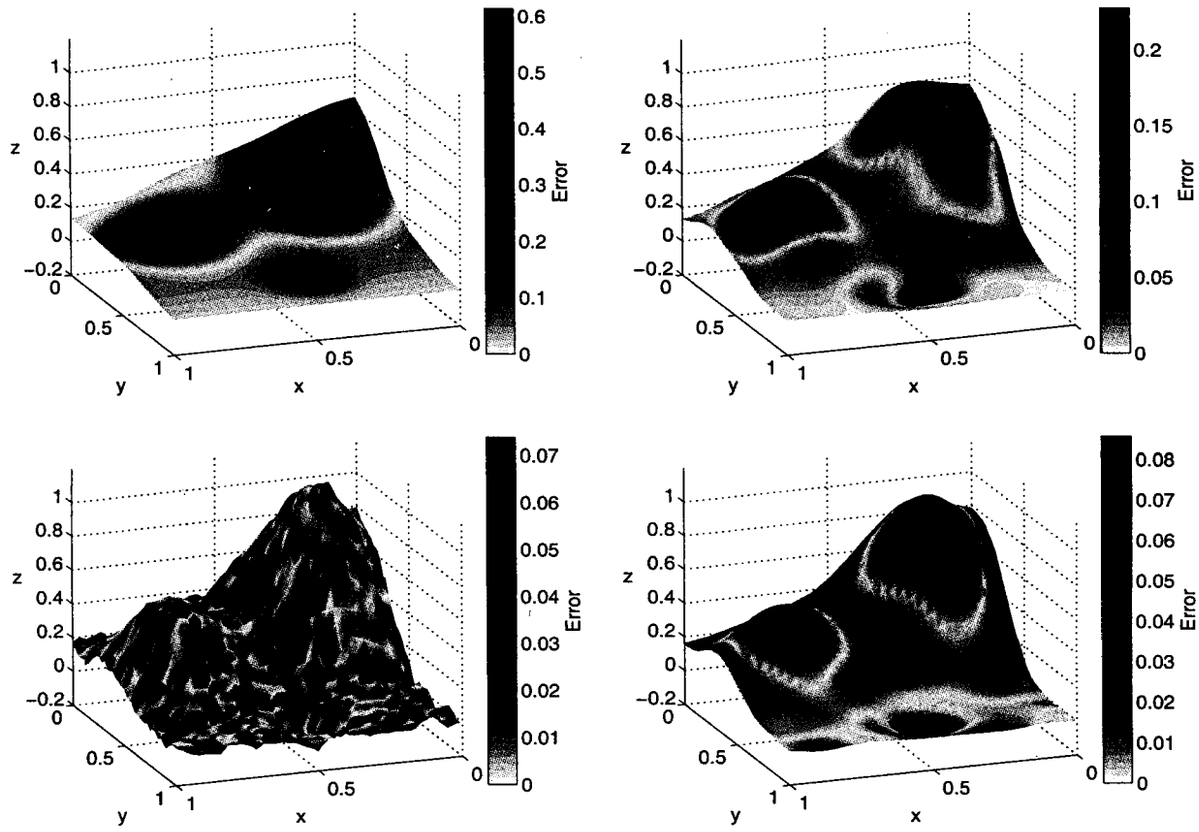


Fig. 24.1 Top: Shepard approximation with stationary Gaussian weights to Franke's function using 9 (left) and 25 (right) uniformly spaced points in $[0,1]^2$. Bottom: Shepard approximation with Gaussian weights to noisy Franke's function using $\varepsilon = 48$ (left) and $\varepsilon = 12$ (right) on 1089 uniformly spaced points in $[0,1]^2$.

```
 5  f4 = @(x,y) 0.2*exp(-((9*x-4).^2+(9*y-7).^2));
 6  testfunction = @(x,y) f1(x,y)+f2(x,y)+f3(x,y)-f4(x,y);
 7  N = 1089; gridtype = 'u';
 8  neval = 40;
    % Load data points
 9  name = sprintf('Data2D_%d%s',N,gridtype); load(name)
10  ctrs = dsites;
    % Create vector of function (data) values
11  f = testfunction(dsites(:,1),dsites(:,2));
    % Create neval-by-neval equally spaced evaluation locations
    % in the unit square
12  grid = linspace(0,1,neval); [xe,ye] = meshgrid(grid);
13  epoints = [xe(:) ye(:)];
    % Compute distance matrix between evaluation points and centers
14  DM_eval = DistanceMatrix(epoints,ctrs);
    % Compute evaluation matrix
15  EM = rbf(ep,DM_eval);
```

```
16  EM = EM./repmat(EM*ones(N,1),1,N);  % Shepard normalization
    % Compute quasi-interpolant
17  Pf = EM*f;
    % Compute exact solution, i.e.,
    % evaluate test function on evaluation points
18  exact = testfunction(epoints(:,1),epoints(:,2));
    % Compute errors on evaluation grid
19  maxerr = norm(Pf-exact,inf);
20  rms_err = norm(Pf-exact)/neval;
21  fprintf('RMS error:     %e\n', rms_err)
22  fprintf('Maximum error: %e\n', maxerr)
    % Plot interpolant
23  PlotSurf(xe,ye,Pf,neval,exact,maxerr,[160,20]);
    % Plot absolute error
24  PlotError2D(xe,ye,Pf,exact,maxerr,neval,[160,20]);
```

The next group of experiments (reported in Tables 24.2 and 24.3) should be compared to the distance matrix fits of Chapter 1. The data are again sampled from the test function

$$f_s(\boldsymbol{x}) = 4^s \prod_{d=1}^{s} x_d(1-x_d), \qquad \boldsymbol{x} = (x_1, \ldots, x_s) \in [0,1]^s,$$

which is parametrized by the space dimension s and coded in the MATLAB subroutine `testfunction.m` (see Program C.1).

Since we are using the compactly supported weights $w(\boldsymbol{x}_i, \boldsymbol{x}) = (1 - \varepsilon\|\boldsymbol{x} - \boldsymbol{x}_i\|)_+^4 (4\varepsilon\|\boldsymbol{x} - \boldsymbol{x}_i\| + 1)$ we can keep the evaluation matrix sparse and are therefore able to deal with much larger data sets. Again, we employ a stationary approximation scheme, *i.e.*, the scale parameter ε for the support of the weight functions is (inversely) proportional to the fill-distance. In fact, we take $\varepsilon = 2^k + 1$ (see line 4 of Program 24.2), where k also determines the number $N = (2^k + 1)^s$ of data points used (*c.f.* line 3). These points are taken to be Halton points in $[0,1]^s$ (*c.f.* line 6).

In Tables 24.2 and 24.3 we list RMS-errors (computed on various grids of equally spaced points, *i.e.*, mostly on the boundary of the unit cube in higher dimensions) for a series of approximation problems in $\mathbb{R}^s$ for $s = 1, 2, \ldots, 6$. Again, the approximation order for Shepard's method seems to be roughly $\mathcal{O}(h)$ independent of the space dimension s.

The MATLAB code `Shepard_CS.m` is listed as Program 24.2. This time the evaluation matrix is a sparse matrix which we also compute in two steps. The standard RBF evaluation matrix is computed as for our earlier CSRBF computations on lines 10 and 11 using the subroutine `DistanceMatrixCSRBF.m` (see Chapter 12). This time the Shepard scaling is performed on line 12 with the help of a diagonal matrix stored in the `spdiags` format.

Table 24.2 Shepard fit in 1D–3D with compactly supported weight function.

	1D		2D		3D	
k	N	RMS-error	N	RMS-error	N	RMS-error
1	3	2.844398e-001	9	2.388314e-001	27	1.912827e-001
2	5	1.665656e-001	25	1.271941e-001	125	1.249589e-001
3	9	8.796073e-002	81	7.107988e-002	729	6.898182e-002
4	17	3.970488e-002	289	3.944228e-002	4913	3.718816e-002
5	33	1.738869e-002	1089	3.109722e-002	35937	2.838763e-002
6	65	7.535727e-003	4225	1.888361e-002	274625	9.837855e-003
7	129	3.418925e-003	16641	2.831294e-002		
8	257	1.615694e-003	66049	2.667914e-003		
9	513	7.872903e-004	263169	2.352736e-003		
10	1025	3.884881e-004				
11	2049	1.940611e-004				
12	4097	9.699922e-005				

Table 24.3 Shepard fit in 4D–6D with compactly supported weight function.

	4D		5D		6D	
k	N	RMS-error	N	RMS-error	N	RMS-error
1	81	1.310311e-001	243	8.936253e-002	729	6.372675e-002
2	625	8.943469e-002	3125	6.344921e-002	15625	3.910649e-002
3	6561	4.599027e-002	59049	3.492958e-002	531441	2.234389e-002
4	83521	2.523581e-002				

Program 24.2. Shepard_CS.m

```
% Shepard_CS
% Script that performs Shepard approximation for arbitrary
% space dimensions s using sparse matrices
% Calls on: DistanceMatrixCSRBF, MakeSDGrid, testfunction
% Uses:     haltonseq (written by Daniel Dougherty from
%           MATLAB Central File Exchange)
    % Wendland C2 weight function
 1  rbf = @(e,r) r.^4.*(5*spones(r)-4*r);
 2  s = 6;   % Space dimension s
    % Number of Halton data points
 3  k = 3; N = (2^k+1)^s;
 4  ep = 2^k+1; % Scale parameter for basis function
 5  neval = 4; M = neval^s;
    % Compute data sites as Halton points
 6  dsites = haltonseq(N,s);
 7  ctrs = dsites;
    % Create vector of function (data) values
```

```
 8  f = testfunction(s,dsites);
    % Create neval^s equally spaced evaluation locations in
    % the s-dimensional unit cube
 9  epoints = MakeSDGrid(s,neval);
    % Compute evaluation matrix, i.e.,
    % matrix of values of generating functions
10  DM_eval = DistanceMatrixCSRBF(epoints,ctrs,ep);
11  EM = rbf(ep,DM_eval);
    % Shepard scaling
12  EM = spdiags(1./(EM*ones(N,1)),0,M,M)*EM;
    % Compute quasi-interpolant
13  Pf = EM*f;
    % Compute exact solution, i.e.,
    % evaluate test function on evaluation points
14  exact = testfunction(s,epoints);
    % Compute errors on evaluation grid
15  maxerr = norm(Pf-exact,inf);
16  rms_err = norm(Pf-exact)/sqrt(M);
17  fprintf('RMS error:     %e\n', rms_err)
18  fprintf('Maximum error: %e\n', maxerr)
```

24.2 MLS Approximation with Linear Reproduction

In general one would expect a moving least squares method that reproduces linear polynomials to be more accurate than one that reproduces only constants such as Shepard's method. We illustrate this in Table 24.4 where we again use the C^2 compactly supported weight functions $w(\boldsymbol{x}_i, \boldsymbol{x}) = (1 - \varepsilon\|\boldsymbol{x} - \boldsymbol{x}_i\|)_+^4 (4\varepsilon\|\boldsymbol{x} - \boldsymbol{x}_i\| + 1)$ in a stationary approximation setting with a base $\varepsilon = 3$ for $N = 9$ data points (and then scaled inversely proportional to the fill-distance) to approximate data sampled from Franke's function at uniformly spaced points in $[0, 1]^2$. We can see that the MLS method with linear reproduction has a numerical approximation order of $\mathcal{O}(h^2)$. This will be justified theoretically in the next chapter.

Instead of solving a Gram system for each evaluation point as suggested in (22.8) or (22.20) we use the explicit formulas for the Lagrange multipliers and generating functions given for the 2D case in Example 23.3. These formulas are implemented in MATLAB in the subroutine `LinearScaling2D_CS.m` and listed in Program 24.3. In particular, this subroutine is written to deal with compactly supported weight functions and thus uses sparse matrices. As in the assembly of sparse interpolation matrices we make use of *kd*-trees.

We build a *kd*-tree of all of the centers for the weight functions and find — for each evaluation point — those centers whose support overlaps the evaluation point. The input to `LinearScaling2D_CS.m` are `epoints` (an $N \times s$ matrix representing a set of N data sites in $\mathbb{R}^s$), `ctrs` (an $M \times s$ matrix representing a set of M centers

Table 24.4 2D stationary MLS approximation to Franke's function at uniformly spaced points in $[0,1]^2$ with linear precision using compactly supported weight function.

N	RMS-error	rate
9	1.789573e-001	
25	7.089522e-002	1.3359
81	2.691693e-002	1.3972
289	7.516377e-003	1.8404
1089	1.944252e-003	1.9508
4225	4.903575e-004	1.9873
16641	1.228639e-004	1.9968
66049	3.072866e-005	1.9994
263169	7.658656e-006	2.0044
1050625	1.921486e-006	1.9949

for the weight functions in $\mathbb{R}^s$), `rbf` (an anonymous or inline function defining the RBF weight function), and `ep` (the scale parameter that determines the size of the support of the weight functions). As always, wide functions result from a small value of `ep`, *i.e.*, the size of the support of the weight function is given by `1/ep`. The output of `LinearScaling2D_CS.m` is an $N \times M$ sparse matrix `Phi` that contains the MLS generating functions centered at the points given by `center` and evaluated at the evaluation points in `epoints`. Note that the compactly supported basic function needs to be supplied in its standard (unshifted) form, *e.g.*, as

```
rbf = @(e,r) max(spones(r)-e*r,0).^4.*(4*e*r+spones(r));
```

for the C^2 Wendland function $\varphi_{3,1}(r) = (1-r)_+^4(4r+1)$.

For each evaluation point (see the loop over `i` from line 10 to line 33) we compute the six different moments (entries of the Gram matrix G, *c.f.* Example 23.3) required for the computation of the Lagrange multipliers on lines 14–20. Then the values of the Lagrange multipliers and of the generating functions at the ith evaluation point are computed on lines 21–24. The final sparse matrix `Phi` is assembled on line 35.

Program 24.3. `LinearScaling2D_CS.m`

```
% Phi = LinearScaling2D_CS(epoints,ctrs,rbf,ep)
% Forms a sparse matrix of scaled generating functions for MLS
% approximation with linear reproduction.
% Uses:          k-D tree package by Guy Shechter from
%                MATLAB Central File Exchange
 1  function Phi = LinearScaling2D_CS(epoints,ctrs,rbf,ep)
 2  [N,s] = size(epoints); [M,s] = size(ctrs);
 3  alpha = [0 0; 1 0; 0 1; 1 1; 2 0; 0 2];
    % Build k-D tree for centers
```

```
 4  [tmp,tmp,Tree] = kdtree(ctrs,[]);
    % For each eval. point, find centers whose support overlap it
 5  support = 1/ep; mu = zeros(6);
    % Modify the following line for optimum performance
 6  veclength = round(support*N*M/4);
 7  rowidx = zeros(1,veclength); colidx = zeros(1,veclength);
 8  validx = zeros(1,veclength);
 9  istart = 1; iend = 0;
10  for i = 1:N
11     [pts,dist,idx] = kdrangequery(Tree,epoints(i,:),support);
12     newlen = length(idx);
       % Vector of basis functions
13     Phi_i = rbf(ep,dist');
       % Compute all 6 moments for i-th evaluation point
14     for j=1:6
15        x_to_alpha = 1;
16        for coord=1:s
17a          x_to_alpha = x_to_alpha .*(ctrs(idx,coord)-...
17b            repmat(epoints(i,coord),newlen,1)).^alpha(j,coord);
18        end
19        mu(j) = Phi_i*x_to_alpha;
20     end
21     L1=(mu(4)^2-mu(5)*mu(6)); L2=(mu(2)*mu(6)-mu(3)*mu(4));
22     L3=(mu(5)*mu(3)-mu(2)*mu(4));
23a    scaling = L1*repmat(1,newlen,1) + ...
23b          L2*(ctrs(idx,1)-repmat(epoints(i,1),newlen,1))+...
23c          L3*(ctrs(idx,2)-repmat(epoints(i,2),newlen,1));
24a    denom = mu(2)^2*mu(6)+mu(5)*mu(3)^2-mu(1)*mu(5)*mu(6)-...
24b            2*mu(2)*mu(3)*mu(4)+mu(1)*mu(4)^2;
25     if (denom ~= 0)
26        scaling = scaling/denom;
27        iend = iend + newlen;
28        rowidx(istart:iend) = repmat(i,1,newlen);
29        colidx(istart:iend) = idx';
30        validx(istart:iend) = Phi_i.*scaling';
31        istart = istart + newlen;
32     end
33  end
34  filled = find(rowidx); % only those actually filled
35  Phi = sparse(rowidx(filled),colidx(filled),validx(filled),N,M);
    % Free memory
36  clear rowidx colidx validx; kdtree([],[],Tree);
```

With all the difficult coding relegated to the subroutine `LinearScaling2D_CS.m` the main program `LinearMLS2D_CS.m` is rather simple. It is listed in Program 24.4. The version of the code listed here is adapted to read a data file that contains both data sites (in the variable `dsites`) and function values (in the variable `rhs`). For the example displayed in Figure 24.2 we used an actual set of 1:250,000-scale Digital Elevation Model (DEM) data from the U.S. Geological Survey, namely the data set `Dubuque-E` which contains elevation data from a region in northeastern Iowa, northwestern Illinois, and southwestern Wisconsin. The data set is available on the worldwide web at `http://edcsgs9.cr.usgs.gov/glis/hyper/guide/1_dgr_demfig/-states/IL.html`. The original data set contains 1201×1201 uniformly spaced measurements ranging in height from 178 meters to 426 meters. We converted the DEM data format with the utility `DEM2XYZN` that can be downloaded from `http://data.geocomm.com/dem/dem2xyzn/`, selected only the northeastern quadrant of $601 \times 601 = 361201$ elevation values, and scaled the x and y coordinates that originally defined a square with a side length of about 35 miles to the unit square. The resulting data set `Data2D_DubuqueNE` is included on the enclosed CD. An MLS approximation with linear reproduction based on the C^2 compactly supported Wendland weight used earlier was evaluated on a grid of 60×60 equally spaced points and is displayed in Figure 24.2. We use a shape parameter of $\varepsilon = 30$ to determine the support size of the weight functions.

Program 24.4. `LinearMLS2D_CS.m`

```
% LinearMLS2D_CS
% Script that performs MLS approximation with linear reproduction
% using sparse matrices
% Calls on: LinearScaling2D_CS
 1  rbf = @(e,r) max(spones(r)-e*r,0).^4.*(4*e*r+spones(r));
 2  ep = 30; neval = 60;
    % Load data points and rhs
 3  load('Data2D_DubuqueNE');
 4  ctrs = dsites;
    % Create neval-by-neval equally spaced evaluation locations
    % in the unit square
 5  grid = linspace(0,1,neval); [xe,ye] = meshgrid(grid);
 6  epoints = [xe(:) ye(:)];
    % Compute evaluation matrix
 7  EM = LinearScaling2D_CS(epoints,ctrs,rbf,ep);
    % Compute MLS approximation   (rhs read from data file)
 8  Pf = EM*rhs;
 9  figure   % Plot MLS fit
10  surfplot = surf(xe,ye,reshape(Pf,neval,neval));
11  set(surfplot,'FaceColor','interp','EdgeColor','none')
12  view([15,35]); camlight; lighting gouraud; colormap summer
```

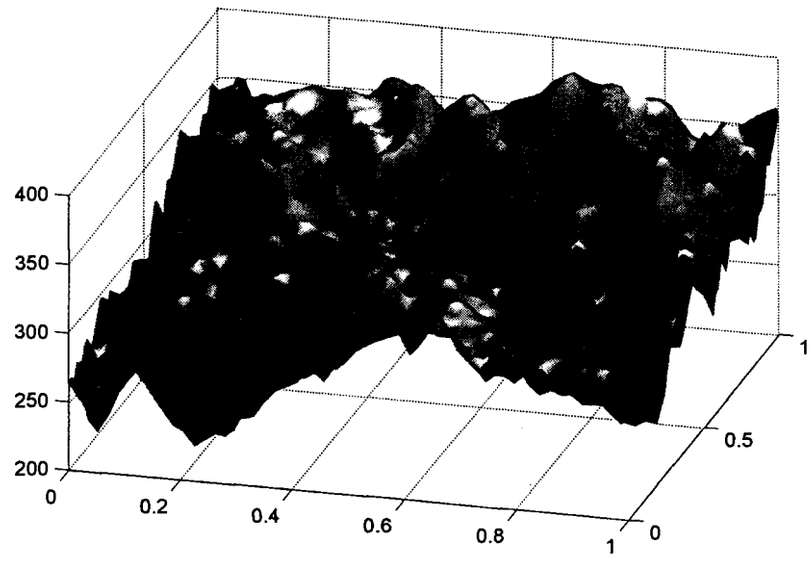


Fig. 24.2 MLS approximation using compactly supported weight for elevation data in northwestern Illinois.

A much simpler (but also much slower) implementation of MLS approximation with linear reproduction is given as Program 24.5. In this implementation we solve the local least squares problem for each evaluation point $\boldsymbol{x}$, *i.e.*, we obtain the MLS approximation at the point $\boldsymbol{x}$ as

$$\mathcal{P}f(\boldsymbol{x}) = u(\boldsymbol{x}, \boldsymbol{x}) = \sum_{j=1}^{m} c_j(\boldsymbol{x}) p_j(\boldsymbol{x} - \boldsymbol{x}) = c_1(\boldsymbol{x}),$$

where the coefficients are found by solving the Gram system (*c.f.* 22.8)

$$G(\boldsymbol{x})c(\boldsymbol{x}) = \boldsymbol{f_p}(\boldsymbol{x}).$$

The solution of these systems (for each evaluation point $\boldsymbol{x}$) is computed on line 18 of the program, with the Gram matrix built as

$$G(\boldsymbol{x}) = P^T(\boldsymbol{x})W(\boldsymbol{x})P(\boldsymbol{x}),$$

and the polynomial matrix P with entries $P_{ij}(\boldsymbol{x}) = p_j(\boldsymbol{x}_i - \boldsymbol{x})$ and diagonal weight matrix computed on lines 16 and 17, respectively. The subroutine `LinearScaling2D_CS.m` is not required by this program. Compare this program with Program 24.1.

Program 24.5. `LinearMLS2D_GramSolve.m`

```
% LinearMLS2D_GramSolve
% Script that performs MLS approximation with linear reproduction
% by solving local Gram systems
% Calls on: DistanceMatrix, PlotSurf, PlotError2D
```

```
 1  rbf = @(e,r) exp(-(e*r).^2); ep = 5.5;
    % Define Franke's function as testfunction
 2  f1 = @(x,y) 0.75*exp(-((9*x-2).^2+(9*y-2).^2)/4);
 3  f2 = @(x,y) 0.75*exp(-((9*x+1).^2/49+(9*y+1).^2/10));
 4  f3 = @(x,y) 0.5*exp(-((9*x-7).^2+(9*y-3).^2)/4);
 5  f4 = @(x,y) 0.2*exp(-((9*x-4).^2+(9*y-7).^2));
 6  testfunction = @(x,y) f1(x,y)+f2(x,y)+f3(x,y)-f4(x,y);
 7  N = 1089; gridtype = 'u';
 8  neval = 40;
    % Load data points
 9  name = sprintf('Data2D_%d%s',N,gridtype); load(name)
10  ctrs = dsites;
    % Create vector of function (data) values
11  f = testfunction(dsites(:,1),dsites(:,2));
    % Create neval-by-neval equally spaced evaluation locations
    % in the unit square
12  grid = linspace(0,1,neval); [xe,ye] = meshgrid(grid);
13  epoints = [xe(:) ye(:)];
    % Compute MLS approximation with shifted basis polynomials
14  Pf = zeros(neval^2,1);
15  for j=1:neval^2
16     P = [ones(N,1) dsites-repmat(epoints(j,:),N,1)];
17     W = diag(rbf(ep,DistanceMatrix(epoints(j,:),ctrs)));
18     c = (P'*W*P)\(P'*W*f);
19     Pf(j) = c(1);
20  end
    % Compute exact solution, i.e.,
    % evaluate test function on evaluation points
21  exact = testfunction(epoints(:,1),epoints(:,2));
    % Compute errors on evaluation grid
22  maxerr = norm(Pf-exact,inf);
23  rms_err = norm(Pf-exact)/neval;
    % Plot interpolant
24  PlotSurf(xe,ye,Pf,neval,exact,maxerr,[160,20]);
    % Plot absolute error
25  PlotError2D(xe,ye,Pf,exact,maxerr,neval,[160,20]);
```

Alternatively, we could have coded lines 14–20 with an unshifted polynomial basis (*c.f.* the discussion in Chapter 22), in which case we would have

```
14  P = [ones(N,1) dsites];
15  Pf = zeros(neval^2,1);
16  for j=1:neval^2
```

```
17      W = diag(rbf(ep,DistanceMatrix(epoints(j,:),ctrs)));
18      c = (P'*W*P)\(P'*W*f);
19      Pf(j) = [1 epoints(j,:)]*c;
20  end
```

in Program 24.5. This requires setting up the matrix P only once. However, the computations are numerically not as stable. Also, additional computation of the approximant on line 19 is required via a dot product.

24.3 Plots of Basis-Dual Basis Pairs

We now provide some plots of the polynomial basis functions p for the standard MLS approach, the dual basis functions λ (Lagrange multipliers), and the generating functions Ψ from the Backus-Gilbert approach for a one-dimensional example with $\mathcal{X}$ being the set of 11 equally spaced points in $[0, 1]$. We take $m = 3$, *i.e.*, we consider the case that ensures reproduction of quadratic polynomials. The weight function is taken to be the standard Gaussian radial function scaled with a shape parameter $\varepsilon = 5$. In the only exception in this book, these plots were created with Maple using the program `MLSDualBases.mws` included in Appendix C.

The three basis polynomials $p_1(x) = 1$, $p_2(x) = x$, and $p_3(x) = x^2$ are shown in Figure 24.3, whereas the dual basis functions λ_1, λ_2, and λ_3 are displayed in Figure 24.4.

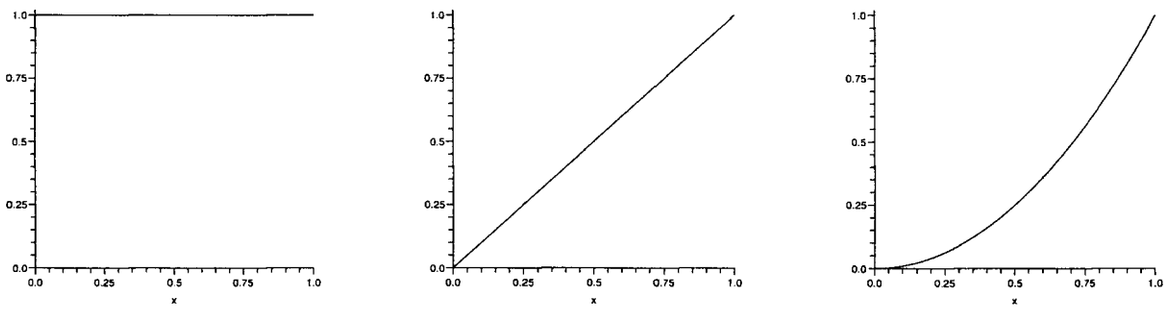


Fig. 24.3 Plot of the three polynomial basis functions for moving least squares approximation with quadratic reproduction on $[0, 1]$.

In Figure 24.5 we plot three MLS generating functions (solid) together with the corresponding generating functions from the approximate moving least squares approach (dashed) described in Chapter 26. The generating functions for the approximate MLS approach are Laguerre-Gaussians (*c.f.* Section 4.2). While the standard MLS generating functions reflect the fact that the data is given on a finite interval, the generating functions for approximate MLS approximation are all just shifts of

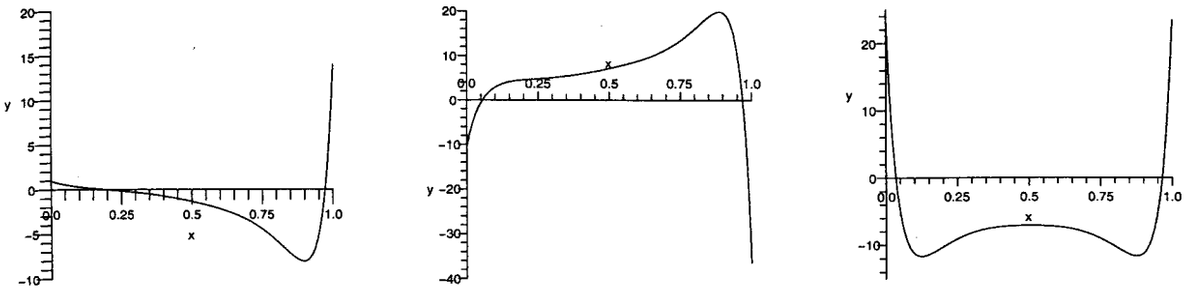


Fig. 24.4 Plot of the three dual basis functions for moving least squares approximation with quadratic reproduction for 11 equally spaced points in $[0, 1]$.

the one function

$$\Psi(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{\sqrt{\mathcal{D}\pi}} \left(\frac{3}{2} - \frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{\mathcal{D}h^2} \right) e^{-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{\mathcal{D}h^2}}$$

to the center points $\boldsymbol{y}$. Here we identify the scale parameter $\mathcal{D}$ with our shape parameter ε for the weight function via $\varepsilon = \frac{1}{\sqrt{\mathcal{D}}h}$. For this example with 11 points in $[0, 1]$ we have $h = 1/10$, so that $\varepsilon = 5$ corresponds to a value of $\mathcal{D} = 4$.

In the center of the interval, where the influence of the boundary is minimal, the two types of generating functions are almost identical (see the right plot in Figure 24.5).

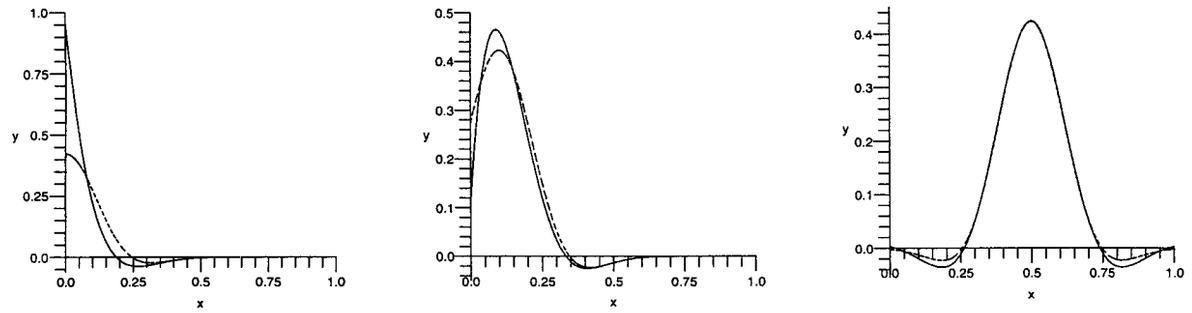


Fig. 24.5 Standard MLS generating functions (solid) and approximate MLS generating functions (dashed) centered at three of the 11 equally spaced points in $[0, 1]$.

If the data points are no longer equally spaced, the Lagrange functions and generating functions are also less uniform. Figures 24.6 and 24.7 illustrate this dependence on the data distribution for 11 Halton points in $[0, 1]$.

Finally, we provide plots of MLS generating functions for the case of reproduction of linear polynomials in 2D (see Figure 24.8). These plots were created with the MATLAB program `LinearMLS2D_CS.m` (see Program 24.4) by plotting column j of the evaluation matrix `EM` corresponding to the values of the jth generating function. We used the C^2 Wendland weights $w(\boldsymbol{x}_i, \boldsymbol{x}) = (1 - \varepsilon\|\boldsymbol{x} - \boldsymbol{x}_i\|)_+^4 (4\varepsilon\|\boldsymbol{x} - \boldsymbol{x}_i\| + 1)$ with $\varepsilon = 5$.

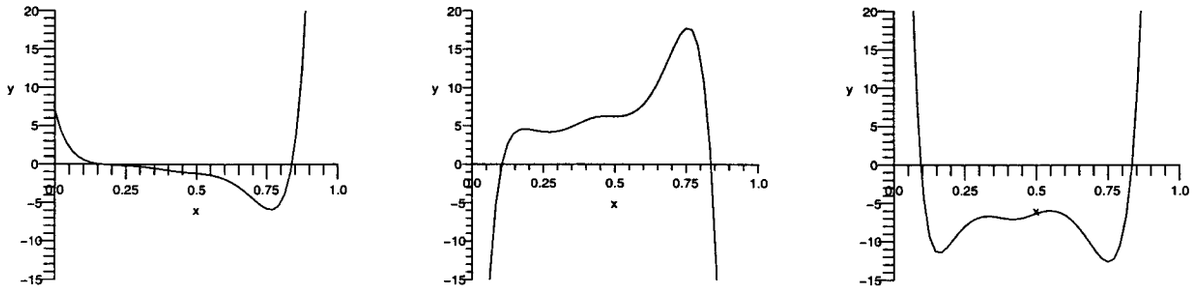


Fig. 24.6 Plot of the three dual basis functions for moving least squares approximation with quadratic reproduction for 11 Halton points in $[0, 1]$.

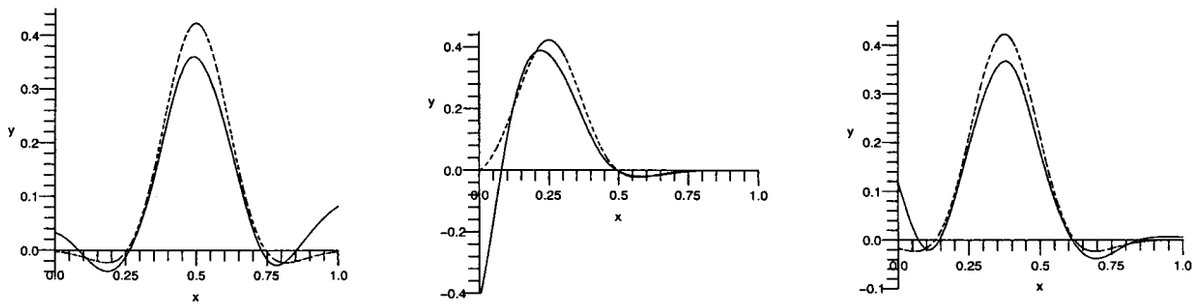


Fig. 24.7 Standard MLS generating functions (solid) and approximate MLS generating functions (dashed) centered at three of the 11 Halton points in $[0, 1]$.

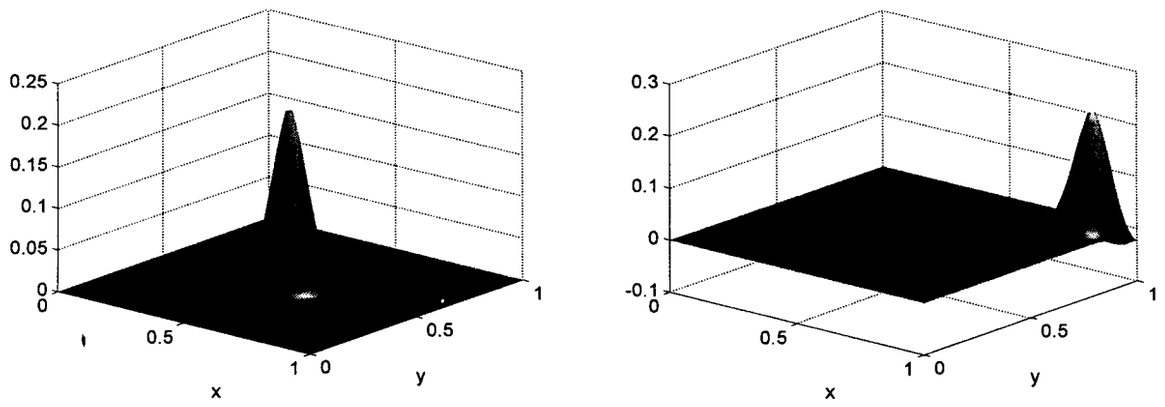


Fig. 24.8 MLS generating functions for linear reproduction centered at two of 289 uniformly spaced data sites in $[0, 1]^2$.

Chapter 25

Error Bounds for Moving Least Squares Approximation

25.1 Approximation Order of Moving Least Squares

Since the moving least squares approximants can be written as quasi-interpolants we can use standard techniques to derive their point-wise error estimates. The standard argument proceeds as follows. Let f be a given (smooth) function that generates the data, *i.e.*, $f_1 = f(\boldsymbol{x}_1), \ldots, f_N = f(\boldsymbol{x}_N)$, and let p be an arbitrary polynomial. Moreover, assume that the moving least squares approximant is given in the form

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{i=1}^{N} f(\boldsymbol{x}_i)\Psi_i(\boldsymbol{x}, \boldsymbol{x})$$

with the generating functions Ψ_i satisfying the polynomial reproduction property

$$\sum_{i=1}^{N} p(\boldsymbol{x}_i)\Psi_i(\boldsymbol{x}, \boldsymbol{x}) = p(\boldsymbol{x}), \quad \text{for all } p \in \Pi_d^s,$$

as described in Chapter 22. Then, due to the polynomial reproduction property of the generating functions, we get

$$\begin{aligned} |f(\boldsymbol{x}) - \mathcal{P}_f(\boldsymbol{x})| &\leq |f(\boldsymbol{x}) - p(\boldsymbol{x})| + |p(\boldsymbol{x}) - \sum_{i=1}^{N} f(\boldsymbol{x}_i)\Psi_i(\boldsymbol{x}, \boldsymbol{x})| \\ &= |f(\boldsymbol{x}) - p(\boldsymbol{x})| + |\sum_{i=1}^{N} p(\boldsymbol{x}_i)\Psi_i(\boldsymbol{x}, \boldsymbol{x}) - \sum_{i=1}^{N} f(\boldsymbol{x}_i)\Psi_i(\boldsymbol{x}, \boldsymbol{x})|. \end{aligned}$$

Combination of the two sum and the definition of the discrete maximum norm yield

$$\begin{aligned} |f(\boldsymbol{x}) - \mathcal{P}_f(\boldsymbol{x})| &\leq |f(\boldsymbol{x}) - p(\boldsymbol{x})| + \sum_{i=1}^{N} |p(\boldsymbol{x}_i) - f(\boldsymbol{x}_i)||\Psi_i(\boldsymbol{x}, \boldsymbol{x})| \\ &\leq \|f - p\|_\infty \left[1 + \sum_{i=1}^{N} |\Psi_i(\boldsymbol{x}, \boldsymbol{x})|\right]. \end{aligned} \tag{25.1}$$

We see that in order to refine the error estimate we now have to answer two questions:

- How well do polynomials approximate f? This can be achieved with standard Taylor expansions.
- Are the generating functions bounded? The expression $\sum_{i=1}^{N} |\Psi_i(\boldsymbol{x}, \boldsymbol{x})|$ is known as the (value of the) *Lebesgue function*, and finding a bound for the Lebesgue function is the main task that remains.

By taking the polynomial p above to be the Taylor polynomial of total degree d for f at $\boldsymbol{x}$, the remainder term immediately yields an estimate of the form

$$\|f - p\|_\infty \leq C_1 h^{d+1} \max_{\boldsymbol{\xi} \in \Omega} |D^{\boldsymbol{\alpha}} f(\boldsymbol{\xi})|, \qquad |\boldsymbol{\alpha}| = d + 1. \tag{25.2}$$

Thus, if we can establish a uniform bound for the Lebesgue function, then (25.1) and (25.2) will result in

$$|f(\boldsymbol{x}) - \mathcal{P}_f(\boldsymbol{x})| \leq C h^{d+1} \max_{\boldsymbol{\xi} \in \Omega} |D^{\boldsymbol{\alpha}} f(\boldsymbol{\xi})|, \qquad |\boldsymbol{\alpha}| = d + 1,$$

which shows that moving least squares approximation with polynomial reproduction of degree d has approximation order $\mathcal{O}(h^{d+1})$.

For Shepard's method, *i.e.*, moving least squares approximation with constant reproduction (*i.e.*, $m = 1$ or $d = 0$), we saw above that the generating functions are of the form

$$\Psi_i(\boldsymbol{x}, \boldsymbol{x}) = \frac{w(\boldsymbol{x}_i, \boldsymbol{x})}{\sum_{j=1}^{N} w(\boldsymbol{x}_j, \boldsymbol{x})}$$

and form a partition of unity (see (23.3)). Therefore the Lebesgue function admits the uniform bound

$$\sum_{i=1}^{N} |\Psi_i(\boldsymbol{x}, \boldsymbol{x})| = 1.$$

This shows that Shepard's method has approximation order $\mathcal{O}(h)$.

Bounding the Lebesgue function in the general case is more involved and is the subject of the papers [Levin (1998); Wendland (2001a)]. As indicated above, this results in approximation order $\mathcal{O}(h^{d+1})$ for a moving least squares method that reproduces polynomials of degree d. In both papers the authors assumed that the weight function is compactly supported, and that the support size is scaled proportional to the fill distance. The following theorem paraphrases the results of [Levin (1998); Wendland (2001a)].

Theorem 25.1. *Let $\Omega \subset \mathbb{R}^s$. If $f \in C^{d+1}(\Omega)$, $\{\boldsymbol{x}_i : i = 1, \ldots, N\} \subset \Omega$ are quasi-uniformly distributed with fill distance h, the weight functions $w_i(\boldsymbol{x}) = w(\boldsymbol{x}_i, \boldsymbol{x})$ are compactly supported with support size $\rho_i = ch$ ($c = $ const.), and if polynomials in Π_d^s are reproduced according to (22.13), then the scale of MLS approximations*

$$\mathcal{P}_f^{(h)}(\boldsymbol{x}) = \sum_{i=1}^{N} f(\boldsymbol{x}_i) \Psi\left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{\rho_i}\right), \tag{25.3}$$

with generating functions $\Psi(\boldsymbol{x} - \boldsymbol{x}_i) = \Psi_i(\boldsymbol{x}, \boldsymbol{x})$ *determined via (22.17)-(22.19) satisfies*

$$\|f - \mathcal{P}_f^{(h)}\|_\infty \leq Ch^{d+1} \max_{\boldsymbol{\xi}\in\Omega} |D^{\boldsymbol{\alpha}} f(\boldsymbol{\xi})|, \qquad |\boldsymbol{\alpha}| = d+1.$$

It should be possible to arrive at similar estimates if the weight function only decays fast enough (see, *e.g.*, the survey [de Boor (1993)]).

Aside from this constraint on the weight function (which essentially corresponds to a stationary approximation scheme), the choice of weight function w does not play a role in determining the approximation order of the moving least squares method. As noted earlier, it only determines the smoothness of $\mathcal{P}_f$. For example, in the paper [Cleveland and Loader (1996)] from the statistics literature on local regression the authors state that often "the choice [of weight function] is not too critical", and the use of the so-called *tri-cube*

$$w_i(\boldsymbol{x}) = (1 - \|\boldsymbol{x} - \boldsymbol{x}_i\|^3)_+^3, \qquad \boldsymbol{x} \in \mathbb{R}^s,$$

is suggested. Of course, many other weight functions such as (radial) B-splines or any of the (bell-shaped) radial basis functions studied earlier can also be used. If the weight function is compactly supported, then the generating functions Ψ_i will be so, too. This leads to computationally efficient methods since the Gram matrix $G(\boldsymbol{x})$ will be sparse.

An interesting question is also the size of the support of the different local weight functions. Obviously, a fixed support size for all weight functions is possible. However, this will cause serious problems as soon as the data are not uniformly distributed. Therefore, in the arguments in [Levin (1998); Wendland (2001a)] the assumption is made that the data are at least quasi-uniformly distributed. Another choice for the support size of the individual weight functions is based on the number of nearest neighbors, *i.e.*, the support size is chosen so that each of the local weight functions contains the same number of centers in its support. A third possibility is suggested in [Schaback (2000b)] where the author proposes to use another moving least squares approximation based on (equally spaced) auxiliary points to determine a smooth function δ such that at each evaluation point $\boldsymbol{x}$ the radius of the support of the weight function is given by $\delta(\boldsymbol{x})$. However, convergence estimates for these latter two choices do not exist.

Sobolev error estimates are provided for moving least squares approximation with compactly supported weight functions in [Armentano (2001)]. The rates obtained in that paper are not in terms of the fill distance but instead in terms of the support size ρ of the weight function. Moreover, it is assumed that for general s and $m = \binom{s+d}{d}$ the local Lagrange functions are bounded. As mentioned above, this is the hard part, and in [Armentano (2001)] such bounds are only provided in the case $s = 2$ with $d = 1$ or $d = 2$. However, if combined with the general bounds for the Lebesgue function provided by Wendland, the paper [Armentano (2001)] yields the following estimates:

$$|f(\boldsymbol{x}) - \mathcal{P}_f(\boldsymbol{x})| \leq C\rho^{d+1} \max_{\boldsymbol{\xi}\in\Omega} |D^{\boldsymbol{\alpha}} f(\boldsymbol{\xi})|, \qquad |\boldsymbol{\alpha}| = d+1,$$

but also

$$|\nabla(f - \mathcal{P}_f)(\boldsymbol{x})| \leq C\rho^d \max_{\boldsymbol{\xi}\in\Omega} |D^{\boldsymbol{\alpha}} f(\boldsymbol{\xi})|, \qquad |\boldsymbol{\alpha}| = d + 1.$$

In the weaker (local) L_2-norm we have

$$\|f - \mathcal{P}_f\|_{L_2(B_j\cap\Omega)} \leq C\rho^{d+1} |f|_{W_2^{d+1}(B_j\cap\Omega)}$$

and

$$\|\nabla(f - \mathcal{P}_f)\|_{L_2(B_j\cap\Omega)} \leq C\rho^{d} |f|_{W_2^{d+1}(B_j\cap\Omega)},$$

where the balls B_j provide a finite cover of the domain Ω, *i.e.*, $\Omega \subseteq \bigcup_j B_j$, and the number of overlapping balls is bounded. Here $W_2^d(\Omega)$ is the Sobolev space of functions whose derivatives up to order d are in L_2 (*c.f.* Chapter 13).

We close this chapter by pointing out that early error estimates for some special cases were provided in [Farwig (1987); Farwig (1991)].

Chapter 26

Approximate Moving Least Squares Approximation

26.1 High-order Shepard Methods via Moment Conditions

While we mentioned earlier that the weight function does not have an effect on the approximation order results for Shepard's method (cf. Theorem 25.1), wee now present some heuristic considerations for obtaining higher-order Shepard methods. This will result in certain conditions on the moments of the weight function w.

Recall that using our shifted monomial representation we obtain $\mathcal{P}_f(\boldsymbol{x}) = c_1(\boldsymbol{x})$ (see (22.5)) for any degree d. Therefore, if we can find weight functions $w(\boldsymbol{x}_i, \cdot)$ such that the first row (and first column) of the Gram matrix $G(\boldsymbol{x})$ becomes $[\sum_i w(\boldsymbol{x}_i, \boldsymbol{x}), 0, \ldots, 0]^T = [\langle 1, 1\rangle_{w_{\boldsymbol{x}}}, 0, \ldots, 0]^T$, then $c_1(\boldsymbol{x}) = \langle f, 1\rangle_{w_{\boldsymbol{x}}} / \langle 1, 1\rangle_{w_{\boldsymbol{x}}}$ via (22.8). Thus,

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{i=1}^{N} f(\boldsymbol{x}_i) \frac{w(\boldsymbol{x}_i, \boldsymbol{x})}{\sum_{j=1}^{N} w(\boldsymbol{x}_j, \boldsymbol{x})},$$

but now — by construction — the method has approximation order $\mathcal{O}(h^{d+1})$ (instead of the mere $\mathcal{O}(h)$ of Shepard's method which permits the use of *arbitrary* weights).

We therefore use the *discrete moments* (*c.f.* Section 23.2)

$$\mu_{\boldsymbol{\alpha}} = \sum_{i=1}^{N} (\boldsymbol{x}_i - \boldsymbol{x})^{\boldsymbol{\alpha}} w(\boldsymbol{x}_i, \boldsymbol{x}), \qquad \boldsymbol{x} \in \mathbb{R}^s, \qquad 0 \leq |\boldsymbol{\alpha}| \leq d, \tag{26.1}$$

where $\boldsymbol{\alpha}$ is a multi-index, and then demand

$$\mu_{\boldsymbol{\alpha}} = \delta_{\boldsymbol{\alpha}\mathbf{0}}, \qquad 0 \leq |\boldsymbol{\alpha}| \leq d. \tag{26.2}$$

As mentioned in Chapter 23 these moments provide the entries of the Gram matrix $G(\boldsymbol{x})$. The condition $\mu_{\mathbf{0}} = 1$ ensures that the weights $w(\boldsymbol{x}_i, \cdot)$ form a *partition of unity*, and we end up with a quasi-interpolant of the form

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{i=1}^{N} f(\boldsymbol{x}_i) w(\boldsymbol{x}_i, \boldsymbol{x}).$$

We can summarize our heuristics in

Proposition 26.1. *Let $\Omega \subset \mathbb{R}^s$. If $f \in C^{d+1}(\Omega)$, the data sites $\{\boldsymbol{x}_i : i = 1, \ldots, N\} \subset \Omega$ are quasi-uniformly distributed with fill distance h, the weight functions $w(\boldsymbol{x}_i, \cdot) = w(\boldsymbol{x}_i - \boldsymbol{x})$ are compactly supported with support size $\rho_i = ch$ ($c = \text{const.}$), and the discrete moment conditions*

$$\mu_{\boldsymbol{\alpha}} = \delta_{\boldsymbol{\alpha}\mathbf{0}}, \qquad 0 \leq |\boldsymbol{\alpha}| \leq d,$$

are satisfied, then

$$\mathcal{P}_f^{(h)}(\boldsymbol{x}) = \sum_{i=1}^{N} f(\boldsymbol{x}_i) w\left(\frac{\boldsymbol{x}_i - \boldsymbol{x}}{\rho_i}\right)$$

has approximation order $\mathcal{O}(h^{d+1})$.

The discrete moment conditions in Proposition 26.1 lead us to the following interpretation of the weight function w:

$$w(\boldsymbol{x}, \boldsymbol{y}) = w_0(\boldsymbol{x}, \boldsymbol{y}) q(\boldsymbol{x} - \boldsymbol{y}), \qquad q \in \Pi_d^s, \tag{26.3}$$

where w_0 is a new (arbitrary) weight function, and q is a polynomial of degree d orthonormal in the sense of the inner product (22.2) with respect to w_0. This view is also taken by Liu and co-workers (see, *e.g.*, [Li and Liu (2002)]) where q is called the *correction function.*

The discrete moment conditions in Proposition 26.1 are difficult to satisfy analytically, as is the construction of discrete multivariate orthogonal polynomials as needed for (26.3). In order to obtain quasi-interpolants for arbitrary space dimensions and scattered data sites we consider the concept of approximate approximation in the next section. There one satisfies continuous moment conditions while ensuring that the discrete moment conditions are almost satisfied.

26.2 Approximate Approximation

We now give the main results on approximate approximation relevant to our work. The concept of *approximate approximation* was first introduced by Maz'ya in the early 1990s (see [Maz'ya (1991); Maz'ya (1994)]). To keep the discussion transparent we restrict ourselves to results for regular center distributions. However, irregularly spaced data are also allowed by the theory (see, *e.g.*, [Maz'ya and Schmidt (2001); Lanzara *et al.* (2006)]) and have been investigated in practice (see, *e.g.*, [Fasshauer (2004)] or [Lanzara *et al.* (2006)]). In [Maz'ya and Schmidt (2001)] the authors present a quasi-interpolation scheme of the form

$$\mathcal{M}_f^{(h)}(\boldsymbol{x}) = \mathcal{D}^{-s/2} \sum_{\boldsymbol{\nu} \in \mathbb{Z}^s} f(\boldsymbol{x}_{\boldsymbol{\nu}}) \Psi\left(\frac{\boldsymbol{x} - \boldsymbol{x}_{\boldsymbol{\nu}}}{\sqrt{\mathcal{D}} h}\right), \tag{26.4}$$

where the data sites $\boldsymbol{x}_{\boldsymbol{\nu}} = h\boldsymbol{\nu}$ are required to lie on a regular s-dimensional grid with gridsize h. The parameter $\mathcal{D}$ scales the support of the generating function Ψ. As we

will see below, the parameter $\mathcal{D}$ can be chosen to make a so-called *saturation error* so small that it does not affect numerical computations. The generating function is required to satisfy the *continuous moment conditions*

$$\int_{\mathbb{R}^s} \boldsymbol{y}^{\boldsymbol{\alpha}} \Psi(\boldsymbol{y}) d\boldsymbol{y} = \delta_{\boldsymbol{\alpha}\mathbf{0}}, \qquad 0 \leq |\boldsymbol{\alpha}| \leq d. \tag{26.5}$$

This is the continuous analog of (26.2).

The following approximate approximation result is due to Maz'ya and Schmidt (see, *e.g.*, [Maz'ya and Schmidt (2001)]).

Theorem 26.1. *Let $f \in C^{d+1}(\mathbb{R}^s)$, $\{\boldsymbol{x}_\nu : \nu \in \mathbb{Z}^s\} \subset \mathbb{R}^s$ and let Ψ be a continuous generating function which satisfies the moment conditions (26.5) along with the decay requirement*

$$|\Psi(\boldsymbol{x})| \leq c_K (1 + \|\boldsymbol{x}\|^2)^{-K/2}, \qquad \boldsymbol{x} \in \mathbb{R}^s,$$

where c_K is some constant, $K > d + s + 1$ and d is the desired degree of polynomial reproduction. Then

$$\|f - \mathcal{M}_f^{(h)}\|_\infty = \mathcal{O}(h^{d+1}) + E_0(\Psi, \mathcal{D}). \tag{26.6}$$

Using Poisson's summation formula one can bound the *saturation error* E_0 by (see Lemma 2.1 in [Maz'ya and Schmidt (1996)])

$$E_0(\Psi, \mathcal{D}) \leq \sum_{\nu \in \mathbb{Z}^s \setminus \{0\}} \mathcal{F}\Psi(\sqrt{\mathcal{D}}\nu). \tag{26.7}$$

For this result the Fourier transform of Ψ is defined via

$$\mathcal{F}\Psi(\omega) = \int_{\mathbb{R}^s} \Psi(\boldsymbol{x}) e^{-2\pi i \boldsymbol{x}\cdot\omega} d\boldsymbol{x}$$

with $\boldsymbol{x} \cdot \omega$ the standard Euclidean inner product of $\boldsymbol{x}$ and ω in $\mathbb{R}^s$. The saturation error can be interpreted as the discrepancy between the continuous and discrete moment conditions, and its influence can be controlled by the choice of the parameter $\mathcal{D}$ in (26.4).

If we use radial generating functions, then we can use the formula for the Fourier transform of a radial function (see Theorem B.1 in Appendix B) to compute the leading term of (26.7), and therefore obtain an estimate for $\mathcal{D}$ for any desired saturation error. If $\mathcal{D}$ is chosen large enough, then the saturation error will be smaller than the machine accuracy for any given computer, and therefore not noticeable in numerical computations. This means, that even though — theoretically — the quasi-interpolation scheme (26.4) fails to converge, it does converge for all practical numerical purposes. Moreover, we can have an arbitrarily high rate of convergence, $d+1$, by picking a generating function Ψ with sufficiently many vanishing moments.

We point out that the error bound (26.6) is formulated for gridded data on an unbounded domain. An analogous result also holds for finite grids (see [Ivanov *et al.* (1999)]); and similar results for scattered data on bounded domains can be found in, *e.g.*, [Lanzara *et al.* (2006); Maz'ya and Schmidt (2001)]. However, in this case the function f (defining the "data") is required to be compactly supported on the finite domain, or appropriately mollified.

26.3 Construction of Generating Functions for Approximate MLS Approximation

In order to construct the generating functions for the approximate MLS approach we assume the generating function Ψ to be radial and of the form

$$\Psi(\boldsymbol{x}) = \psi_0(\|\boldsymbol{x}\|^2)q(\|\boldsymbol{x}\|^2).$$

Therefore, the continuous moment conditions become (*c.f.* (26.5))

$$\int_{\mathbb{R}^s} \|\boldsymbol{x}\|^{2k} q(\|\boldsymbol{x}\|^2)\psi_0(\|\boldsymbol{x}\|^2)dx = \delta_{k0}, \qquad 0 \le k \le d, \tag{26.8}$$

and we need to determine the univariate polynomial q of degree d accordingly to get approximation order $\mathcal{O}(h^{2d+2})$.

By using s-dimensional spherical coordinates we can rewrite the integral in (26.8) as

$$\begin{aligned}
&\int_0^\infty \int_0^{2\pi} \int_0^\pi \dots \int_0^\pi r^{2k} q(r^2)\psi_0(r^2) r^{s-1} \sin^{s-2}\phi_1 \dots \sin\phi_{s-2} d\phi_1 \dots d\phi_{s-2} d\theta dr \\
&\quad = 2\pi \int_0^\pi \sin^{s-2}\phi_1 d\phi_1 \dots \int_0^\pi \sin\phi_{s-2} d\phi_{s-2} \int_0^\infty r^{2k} q(r^2)\psi_0(r^2) r^{s-1} dr \\
&\quad = 2\pi \prod_{m=1}^{s-2} \int_0^\pi \sin^m \phi d\phi \int_0^\infty r^{2k} q(r^2)\psi_0(r^2) r^{s-1} dr.
\end{aligned} \tag{26.9}$$

The substitution $y = r^2$ converts the last integral to

$$\int_0^\infty r^{2k} q(r^2)\psi_0(r^2) r^{s-1} dr = \frac{1}{2} \int_0^\infty y^k q(y)\psi_0(y) y^{(s-2)/2} dy, \tag{26.10}$$

and one can show that

$$\pi \prod_{m=1}^{s-2} \int_0^\pi \sin^m \phi d\phi = \frac{\pi^{s/2}}{\Gamma(s/2)}. \tag{26.11}$$

Combining (26.9), (26.10) and (26.11) gives us

$$\int_{\mathbb{R}^s} \|\boldsymbol{x}\|^{2k} q(\|\boldsymbol{x}\|^2)\psi_0(\|\boldsymbol{x}\|^2)dx = \frac{\pi^{s/2}}{\Gamma(s/2)} \int_0^\infty y^{k-1} q(y)\psi_0(y) y^{s/2} dy,$$

and therefore we now have obtained a set of one-dimensional orthogonality conditions

$$\frac{\pi^{s/2}}{\Gamma(s/2)} \int_0^\infty y^{k-1} q(y)\psi_0(y) y^{s/2} dy = \delta_{k0}, \qquad 0 \le k \le d \tag{26.12}$$

that can be used to determine the generating function

$$\Psi(\boldsymbol{x}) = \psi_0(\|\boldsymbol{x}\|^2)q(\|\boldsymbol{x}\|^2)$$

once we have chosen an initial weight ψ_0. Moreover, we know that this construction ensures Ψ to have approximate approximation order $\mathcal{O}(h^{2d+2})$.

Thus, the strategy for constructing generating functions for higher-order approximate MLS approxmation is as follows:

(1) Pick an arbitrary (univariate) weight function ψ_0.
(2) Compute the coefficients of $q \in \Pi_d^1$ via the (univariate) moment conditions (26.12).
(3) This leads to the (multivariate) radial generating function $\Psi(\boldsymbol{x}) = \psi_0(\|\boldsymbol{x}\|^2)q(\|\boldsymbol{x}\|^2)$ to be used in the quasi-interpolant (26.4).

Example 26.1. Probably the most aesthetic example is given by $\psi_0(y) = e^{-y}$ so that the basic generating function (corresponding to $d = 0$, *i.e.*, with approximate approximation order $\mathcal{O}(h^2)$ is found by assuming that the polynomial q is a constant, *i.e.*, $q(y) \equiv a_0$. Then (26.12) becomes

$$\int_0^\infty a_0 y^{(s-2)/2} e^{-y} dy = \frac{\Gamma(s/2)}{\pi^{s/2}},$$

which leads to $a_0 = \pi^{-s/2}$, so that we have

$$\Psi(\boldsymbol{x}) = \frac{1}{\sqrt{\pi^s}} e^{-\|\boldsymbol{x}\|^2}$$

— an appropriately scaled Gaussian.

If we take $d = 1$ (to obtain approximate approximation order $\mathcal{O}(h^4)$) and assume q to be a linear polynomial of the form $q(y) = a_0 + a_1 y$, then there are two moment conditions, namely

$$\int_0^\infty (a_0 + a_1 y) y^{(s-2)/2} e^{-y} dy = \frac{\Gamma(s/2)}{\pi^{s/2}},$$

$$\int_0^\infty (a_0 + a_1 y) y^{s/2} e^{-y} dy = 0,$$

or

$$a_0 \Gamma(s/2) + a_1 \Gamma((s+2)/2) = \frac{\Gamma(s/2)}{\pi^{s/2}},$$

$$a_0 \Gamma((s+2)/2) + a_1 \Gamma((s+4)/2) = 0.$$

We can solve this system of linear equations and obtain

$$a_0 = \frac{s+2}{2\sqrt{\pi^s}}, \qquad a_1 = \frac{-1}{\sqrt{\pi^s}}.$$

In the special case $s = 2$ this yields $a_0 = \frac{2}{\pi}$ and $a_1 = -\frac{1}{\pi}$, so that

$$\Psi(\boldsymbol{x}) = \frac{1}{\pi} \left(2 - \|\boldsymbol{x}\|^2\right) e^{-\|\boldsymbol{x}\|^2}.$$

In general, the polynomials q turn out to be (univariate) generalized Laguerre polynomials $L_d^{s/2}$ of degree d (*c.f.* Section 4.2) which are known to be orthogonal on the interval $[0, \infty)$ with respect to the weight function $y^{s/2} e^{-y}$. Therefore the generating functions for the approximate MLS approximation method are the Laguerre-Gaussians

$$\Psi(\boldsymbol{x}) = \frac{1}{\sqrt{\pi^s}} e^{-\|\boldsymbol{x}\|^2} L_d^{s/2}(\|\boldsymbol{x}\|^2).$$

In particular, Table 4.1 contains specific examples for $s = 1, 2, 3$ and $d = 1$ and 2 except for the scale factor $1/\sqrt{\pi^s}$. Figure 4.1 shows plots of the generating functions in the cases $s = 1$, $d = 2$, and $s = 2$, $d = 2$.

Example 26.2. If we use the function $\psi_0(y) = \left(1-\sqrt{y}\right)_+^4 (4\sqrt{y}+1)$ as initial weight, then we can perform calculations analogous to those above. For $d = 0$ and $s = 2$ we get

$$\Psi(\boldsymbol{x}) = \frac{7}{\pi}\psi_0(\|\boldsymbol{x}\|^2) = \frac{7}{\pi}(1-\|\boldsymbol{x}\|)_+^4 (4\|\boldsymbol{x}\|+1), \qquad \boldsymbol{x} \in \mathbb{R}^2, \tag{26.13}$$

with approximation order $\mathcal{O}(h^2)$. Except for the factor $7/\pi$ this generating function corresponds to Wendland's compactly supported C^2 radial basic function (*c.f.* Table 11.1).

Using the same function ψ_0 for $d = 1$ and $s = 2$ we obtain

$$\Psi(\boldsymbol{x}) = \psi_0(\|\boldsymbol{x}\|^2)\frac{7}{\pi}\left(\frac{504}{229} - \frac{1980}{229}\|\boldsymbol{x}\|^2\right) \tag{26.14}$$

with approximation order $\mathcal{O}(h^4)$. This function is displayed in the left plot of Figure 26.1. See Examples 27.2 and 27.4 for more special cases based on this initial weight function.

Example 26.3. Many other choices for the initial weight function ψ_0 are possible. For example, we can take $\psi_0(y) = (1-y)^\alpha(1+y)^\beta$, the weight function for univariate Jacobi polynomials (which are orthogonal on $[-1,1]$). Since the integral defining the orthogonality relations contains an extra factor of $y^{(2k+s-2)/2}$, the moment conditions (26.12) do not yield Jacobi polynomials. However, the resulting generating functions for approximate MLS approximation can still be rather simple. In Table 26.1 we list several such examples for $s = 2$, $\beta = 0$ and various combinations of d and α. Note that these functions are also compactly supported, *i.e.*, they are defined to be zero for $\|\boldsymbol{x}\| > 1$.

Table 26.1 Approximate MLS generating functions Ψ based on $\psi_0(y) = (1-y)^\alpha$, $y \in [-1,1]$ for various choices of d and α.

d	$\alpha = 2$	$\alpha = 5/2$
0	$\frac{3}{\pi}(1-\|\boldsymbol{x}\|^2)^2$	$\frac{7}{2\pi}(1-\|\boldsymbol{x}\|^2)^{5/2}$
1	$\frac{4}{\pi}(2-5\|\boldsymbol{x}\|^2)(1-\|\boldsymbol{x}\|^2)^2$	$\frac{9}{4\pi}(4-11\|\boldsymbol{x}\|^2)(1-\|\boldsymbol{x}\|^2)^{5/2}$
2	$\frac{15}{\pi}(1-6\|\boldsymbol{x}\|^2+7\|\boldsymbol{x}\|^4)(1-\|\boldsymbol{x}\|^2)^2$	$\frac{33}{16\pi}(8-52\|\boldsymbol{x}\|^2+65\|\boldsymbol{x}\|^4)(1-\|\boldsymbol{x}\|^2)^{5/2}$

The function $\Psi(\boldsymbol{x}) = \frac{4}{\pi}\left(2-5\|\boldsymbol{x}\|^2\right)(1-\|\boldsymbol{x}\|^2)^2$ is displayed in the right plot of Figure 26.1.

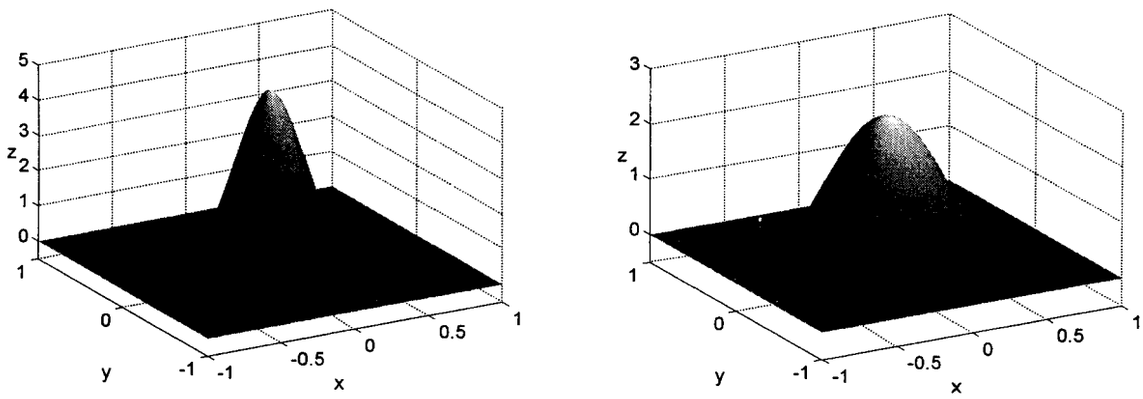


Fig. 26.1 Compactly supported generating functions for approximate linear reproduction. $\Psi(\boldsymbol{x}) = \frac{7}{\pi}\left(\frac{664}{229} - \frac{1980}{229}\|\boldsymbol{x}\|^2\right)(1-\|\boldsymbol{x}\|)_+^4\,(4\|\boldsymbol{x}\|+1)$ (left) and $\Psi(\boldsymbol{x}) = \frac{4}{\pi}\left(2-5\|\boldsymbol{x}\|^2\right)(1-\|\boldsymbol{x}\|^2)^2$ (right) centered at the origin in $\mathbb{R}^2$.

Chapter 27

Numerical Experiments for Approximate MLS Approximation

In this chapter we present a series of experiments for approximate MLS approximation with both globally supported Laguerre-Gaussian generating functions as well as with compactly supported generating functions based on the initial weight $\psi_0(y) = \left(1-\sqrt{y}\right)_+^4 (4\sqrt{y}+1)$ as in Example 26.2 of the previous chapter.

27.1 Univariate Experiments

Example 27.1. We begin with univariate globally supported Laguerre-Gaussians. These functions are listed in Table 4.1 except for the scaling factor $1/\sqrt{\pi}$ required for the ID case. In the left plot of Figure 27.1 we illustrate the effect the scaling parameter $\mathcal{D}$ has on the convergence behavior for Gaussian generating functions. We use a mollified univariate Franke-like function of the form

$$f(x) = 15e^{-\frac{1}{1-(2x-1)^2}}\left(\frac{3}{4}e^{-\frac{(9x-2)^2}{4}} + \frac{3}{4}e^{-\frac{(9x+1)^2}{49}} + \frac{1}{2}e^{-\frac{(9x-7)^2}{4}} - \frac{1}{5}e^{-(9x-4)^2}\right)$$

as test function. For each choice of $\mathcal{D} \in \{0.4, 0.8, 1.2, 1.6, 2.0\}$ we use a sequence of grids of $N = 2^k + 1$ (with $k = 1, \ldots, 14$) equally spaced points in $[0, 1]$ at which we sample the test function. The approximant is computed via

$$\mathcal{P}_f(x) = \frac{1}{\sqrt{\pi\mathcal{D}}} \sum_{i=1}^{N} f(x_i) e^{-\frac{(x-x_i)^2}{\mathcal{D}h^2}}, \qquad x \in [0, 1],$$

where $h = 1/(N-1)$. This corresponds to our usual shape parameter ε having a value of

$$\varepsilon = \frac{1}{\sqrt{\mathcal{D}}h} = \frac{N-1}{\sqrt{\mathcal{D}}} = \frac{2^k}{\sqrt{\mathcal{D}}},$$

i.e., we are in the regime of stationary approximation. The effect of $\mathcal{D}$ is clearly visible in the figure. A value of $\mathcal{D} \geq 2$ exhibits an approximation order of $\mathcal{O}(h^2)$ throughout the range of our experiments, while smaller values allow the saturation error to creep in at earlier stages.

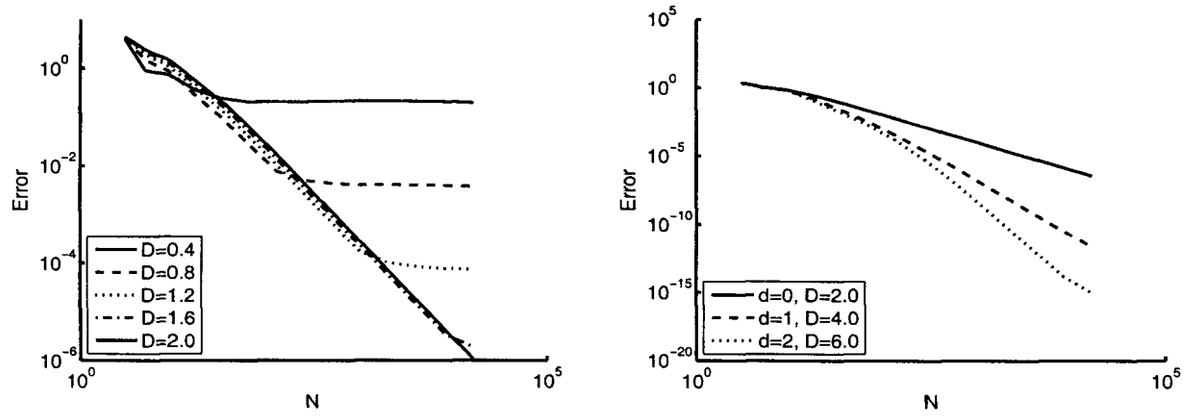


Fig. 27.1 Convergence of 1D approximate MLS approximation. The left plot shows the effect of various choices of $\mathcal{D}$ on the convergence behavior of Gaussians. The right plot illustrates the convergence of Laguerre-Gaussians for various values of d.

In the right plot of Figure 27.1 we compare the approximation orders achievable with the Laguerre-Gaussians of orders $d = 0, 1, 2$ in 1D. The respective values of $\mathcal{D}$ are $\mathcal{D} = 2, 4, 6$. The steepest sections of the curves correspond to approximate approximation orders of $\mathcal{O}(h^{2.0})$, $\mathcal{O}(h^{4.0})$, and $\mathcal{O}(h^{5.99})$, respectively — a perfect match with the rates predicted by the theory. Notice that for the second-order Laguerre-Gaussian we have convergence all the way to machine accuracy.

The MATLAB program `ApproxMLSApprox1D.m` (see Program 27.1) was used to generate the right plot in Figure 27.1. We define the three different Laguerre-Gaussian generating functions as members of a MATLAB cell array `rbf` and place the corresponding values of $\mathcal{D}$ to be used with each of the functions in the vector `D` (see lines 1–4). The univariate Franke-like test function is defined in lines 5–10. This function is mollified so that it goes to zero smoothly at the boundaries of the interval. The program contains two for-loops. The first is over the three different generating functions (corresponding to approximate constant, linear and quadratic reproduction, respectively). The inner loop performs a series of experiments with an increasing number N of data. Here we perform 14 iterations with N ranging from $N = 3$ to $N = 16385$.

For applications of approximate MLS approximation we limit ourselves to uniformly spaced data since there are presently no robust methods for dealing with nonuniform data (see [Lanzara *et al.* (2006); Maz'ya and Schmidt (2001)] for a theoretical approach to non-uniform data, and [Fasshauer (2004); Lanzara *et al.* (2006)] for some numerical experiments). All we need in order to compute the approximant is the evaluation matrix `EM` computed on line 23, which is then multiplied by the function values `f` and scaled by the factor $\mathcal{D}^{-s/2}$ on line 24. The commands needed to produce the plot are included on lines 15, 27 and 29–31.

Program 27.1. ApproxMLSApprox1D.m

```
% ApproxMLSApprox1D
% Script that performs 1D approximate MLS approximation
% Calls on: DistanceMatrix
    % Laguerre-Gaussians for 1D
 1  rbf{1} = @(e,r) exp(-(e*r).^2)/sqrt(pi);
 2  rbf{2} = @(e,r) exp(-(e*r).^2)/sqrt(pi).*(1.5-(e*r).^2);
 3a rbf{3} = @(e,r) exp(-(e*r).^2)/sqrt(pi).*...
 3b          (1.875-2.5*(e*r).^2+0.5*(e*r).^4);
 4  D = [2, 4, 6];   % Scale parameters for generating functions
    % Define Franke-like function as testfunction
 5  f1 = @(x) 0.75*exp(-(9*x-2).^2/4);
 6  f2 = @(x) 0.75*exp(-(9*x+1).^2/49);
 7  f3 = @(x) 0.5*exp(-(9*x-7).^2/4);
 8  f4 = @(x) 0.2*exp(-(9*x-4).^2);
 9  moll = @(x) 15*exp(-1./(1-4*(x-0.5).^2));
10  testfunction = @(x) moll(x).*(f1(x)+f2(x)+f3(x)-f4(x));
11  maxlevel = 14;   % number of iterations
12  M = 200;   % to create M evaluation points in unit interval
13  xe = linspace(0,1,M); epoints = xe(:);
14  exact = testfunction(epoints);
15  figure; hold on; cword = cellstr(['r- ';'g--';'b: ']);
16  for i=1:length(D)
17     for k=1:maxlevel
18        N(k) = (2^k+1); ep = (N(k)-1)/sqrt(D(i));
19        name = sprintf('DataID_%du', N(k));  load(name);
20        ctrs = dsites;
          % Create vector of function values
21        f = testfunction(dsites);
          % Compute evaluation matrix
22        DM = DistanceMatrix(epoints,ctrs);
23        EM = rbf{i}(ep,DM);
          % Compute approximate MLS approximation
24        Pf = EM*f/sqrt(D(i));
          % Compute RMS error on evaluation grid
25        rms_err(k) = norm(Pf-exact)/sqrt(M);
26     end
27     plot(N,rms_err,cword{i});
28  end
29  set(gca,'XScale','log','YScale','log','Fontsize',14)
30  legend('d=0, D=2.0','d=1, D=4.0','d=2, D=6.0',3);
31  xlabel('N');  ylabel('Error');  hold off
```

Example 27.2. In the second set of experiments we use compactly supported generating functions with initial weight $\psi_0(y) = \left(1-\sqrt{y}\right)_+^4 (4\sqrt{y}+1)$. In the univariate case these functions are for $d = 0, 1, 2$

$$\begin{aligned}
\Psi(x) &= \frac{3}{2}\left(1-|x|\right)_+^4 (4|x|+1)\,, \\
\Psi(x) &= \frac{1}{3}\left(7-35|x|^2\right)\left(1-|x|\right)_+^4 (4|x|+1)\,, \\
\Psi(x) &= \frac{105}{11993}\left(350-3960|x|^2+7293|x|^4\right)\left(1-|x|\right)_+^4 (4|x|+1)\,.
\end{aligned} \tag{27.1}$$

These functions can be computed as in Section 26.3 of the previous chapter. An approximate approximation order of $\mathcal{O}(h^2)$ for the first function in (27.1) is illustrated in the left plot of Figure 27.2. Note that a rather large value of $\mathcal{D}$ (namely $\mathcal{D} \approx 500$) is required to make the saturation error so small that it no longer affects our experiments. Since the shape parameter ε determines the support radius of our generating functions, and since $\varepsilon = 1/\sqrt{D}h = (N-1)/\sqrt{\mathcal{D}}$, we see that the evaluation matrix will be completely dense until N grows above approximately 25. Furthermore, for such a large value of $\mathcal{D}$ there is a visible smoothing effect (very slow convergence) during the first few iterations with $N = 3, 5, 9, 17, 33, 65$.

In the right plot of Figure 27.2 we compare the approximation orders achievable with the univariate compactly supported generating functions (27.1) of orders $d = 0, 1, 2$. The respective values of $\mathcal{D}$ required to prevent the saturation error from corrupting our experiments are $\mathcal{D} = 500, 5000, 20000$, respectively. Note that $\mathcal{D} = 20000$ implies that the evaluation matrix will be dense (and therefore computationally inefficient) until N reaches about 150. Thus, it is not until the very last iterations with $N = 8193$ and $N = 16385$ that we get to take real advantage of the compact support of the generating function, *i.e.*, have sparse sums. The steepest sections of the curves correspond to approximate approximation orders of $\mathcal{O}(h^{2.0})$, $\mathcal{O}(h^{3.91})$, and $\mathcal{O}(h^{5.45})$, respectively.

The MATLAB code for the compactly supported experiments can be written in two ways. One possibility would be to rewrite the generating functions in shifted form as explained in Chapter 12, and then use the sparse code `DistanceMatrixCSRBF.m`. However, as just explained, we cannot take much advantage of the sparsity usually associated with compactly supported functions. Therefore, we use essentially the same code as in Program 27.1. The only changes needed are the substitution of the definition of the compactly supported generating functions for the Laguerre-Gaussians along with appropriate values for `D`.

Our experiments seem to suggest that there is no point in using compactly supported generating functions for univariate approximate MLS approximation. The results using Laguerre-Gaussians are far more accurate, and due to the large value of $\mathcal{D}$ required for the compactly supported functions they do not offer an advantage in terms of computational complexity, either.

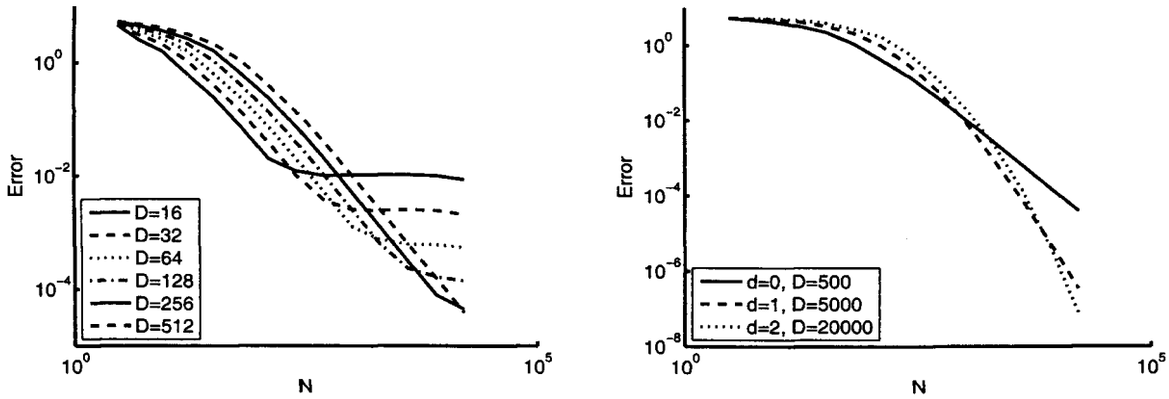


Fig. 27.2 Convergence of 1D approximate MLS approximation with compactly supported generating functions. The left plot shows the effect of various choices of $\mathcal{D}$ on the convergence behavior for the first function in (27.1). The right plot illustrates the convergence for the three functions in (27.1).

27.2 Bivariate Experiments

Example 27.3. This example is similar to the second part of Example 27.1. Now we use bivariate Laguerre-Gaussians with scale factor $1/\pi$. The test data are sampled from a bivariate mollified Franke function, *i.e.*, we multiply Franke's function (2.2) by the mollifier

$$g(x,y) = 15e^{-\frac{1}{1-(2x-1)^2}} e^{-\frac{1}{1-(2y-1)^2}}.$$

The data sites are uniform grids of $(2^k+1)^2$ points (with $k = 1,\ldots,5$) in the unit square. The values for the scale parameter $\mathcal{D}$ used are $\mathcal{D} = 1, 2, 2.5$. The steepest sections of the error curves correspond to approximate approximation orders of $\mathcal{O}(h^{1.83})$, $\mathcal{O}(h^{2.80})$, and $\mathcal{O}(h^{3.00})$, respectively. Note that these rates do not match the theoretically predicted orders. We will see that we can achieve the theoretically predicted orders by using more data sites. This, however, will require special evaluation techniques to deal with the large sums efficiently (see the next chapter).

Example 27.4. The bivariate compactly supported generating functions with initial weight $\psi_0(y) = \left(1-\sqrt{y}\right)_+^4 (4\sqrt{y}+1)$ providing approximate reproduction of constants, linear and quadratic polynomials, respectively, are (*c.f.* Example 26.2)

$$\Psi(\boldsymbol{x}) = \frac{7}{\pi}\left(1-\|\boldsymbol{x}\|\right)_+^4 (4\|\boldsymbol{x}\|+1),$$

$$\Psi(\boldsymbol{x}) = \frac{252}{229\pi}\left(14-55\|\boldsymbol{x}\|^2\right)\left(1-\|\boldsymbol{x}\|\right)_+^4 (4\|\boldsymbol{x}\|+1),$$

$$\Psi(\boldsymbol{x}) = \frac{693}{112901\pi}\left(4179-37050\|\boldsymbol{x}\|^2+59605\|\boldsymbol{x}\|^4\right)\left(1-\|\boldsymbol{x}\|\right)_+^4 (4\|\boldsymbol{x}\|+1).$$

The values we use for the scale parameter are $\mathcal{D} = 20, 40, 80$, respectively. The steepest sections of the error curves correspond to approximate approximation or-

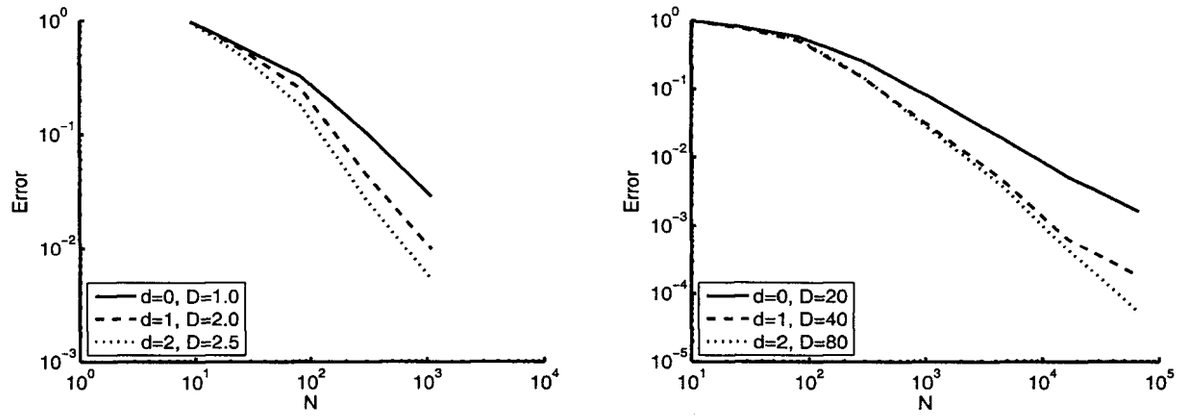


Fig. 27.3 Convergence of 2D approximate MLS approximation with Laguerre-Gaussians (left) and compactly supported (right) generating functions for various values of d.

ders of $\mathcal{O}(h^{1.96})$, $\mathcal{O}(h^{3.03})$, and $\mathcal{O}(h^{3.26})$, respectively. Again, we do not obtain a match with the theoretically predicted orders, even though we used up to $N = 66049$ data points.

The bivariate case provides a more level playing field for the compactly supported generating functions. We can take advantage of the compact support and obtain more accurate results at a reasonable cost. However, another alternative way of obtaining highly accurate multivariate approximate MLS approximations is presented in the next chapter.

Chapter 28

Fast Fourier Transforms

28.1 NFFT

In the recent papers [Kunis *et al.* (2002); Nieslony *et al.* (2004); Potts and Steidl (2003)] use of the *fast Fourier transform for non-uniformly spaced points* was suggested as an efficient way to solve and evaluate radial basis function problems. The C++ software package NFFT by the authors is available for free download [Kunis and Potts (2002)]. A discussion of the actual NFFT software would go beyond the scope of this book. Instead, we briefly describe how to use NFFTs and FFTs to simultaneously evaluate expansions of the form

$$\mathcal{P}_f(\boldsymbol{y}_j) = \sum_{k=1}^{N} c_k \Phi(\boldsymbol{y}_j - \boldsymbol{x}_k) \tag{28.1}$$

at many evaluation points $\boldsymbol{y}_j$, $j = 1, \ldots, M$. Note that this covers not only approximate MLS approximations, but also the evaluation of other quasi-interpolants as well as the evaluation of RBF interpolants.

Direct summation of (28.1) requires $\mathcal{O}(MN)$ operations, while it can be shown that use of the NFFT reduces the cost to $\mathcal{O}(M + N)$ operations. Therefore, as is always the case with fast Fourier transforms, use of the algorithm will pay off for sufficiently many evaluations.

In their papers Nieslony, Potts and Steidl distinguish between basic functions Φ that are singular and those that are non-singular. Singular basic functions are C^∞ everywhere except at the origin and include examples such as

$$\frac{1}{r}, \quad \frac{1}{r^2}, \quad \log r, \quad r^2 \log r,$$

where $r = \|\cdot\|$. Non-singular basic functions are smooth everywhere such as Gaussians and (inverse) multiquadrics. We will restrict our discussion to this latter class.

The basic idea for the following algorithm is remarkably simple. It relies on the fact that the exponential $e^{-\alpha(\boldsymbol{y}_j - \boldsymbol{x}_k)}$ can be written as $e^{-\alpha \boldsymbol{y}_j} e^{\alpha \boldsymbol{x}_k}$. Moreover, the method applies to arbitrary basic functions (which is in strong contrast to the fast

multipole type methods discussed in Chapter 35. One starts out by approximating the (arbitrary, but smooth) basic function Φ using standard Fourier series, *i.e.*,

$$\Phi(\boldsymbol{x}) \approx \sum_{\boldsymbol{\ell} \in I_n} b_{\boldsymbol{\ell}} e^{2\pi i \boldsymbol{\ell} \cdot \boldsymbol{x}} \tag{28.2}$$

with $\boldsymbol{\ell}$ a multi-index in the index set $I_n = \left[-\frac{n}{2}, \frac{n}{2}\right)^s$. The coefficients $b_{\boldsymbol{\ell}}$ are found by the discrete inverse Fourier transform

$$b_{\boldsymbol{\ell}} = \frac{1}{n^s} \sum_{\boldsymbol{k} \in I_n} \Phi\left(\frac{\boldsymbol{k}}{n}\right) e^{-2\pi i \boldsymbol{k} \cdot \boldsymbol{\ell}/n}. \tag{28.3}$$

Numerically, this task is accomplished with software for the standard (inverse) FFT (*e.g.*, [FFTW]).

Remark 28.1. Note that this definition of the Fourier transform (as well as the one below) is different from the one used throughout the rest of this book. However, in order to stay closer to the software packages, we adopt the notation used there.

Using the representation (28.2) of the basic function Φ we can rewrite (28.1) as

$$\begin{aligned} \mathcal{P}_f(\boldsymbol{y}_j) &\approx \sum_{k=1}^{N} c_k \sum_{\boldsymbol{\ell} \in I_n} b_{\boldsymbol{\ell}} e^{2\pi i \boldsymbol{\ell} \cdot (\boldsymbol{y}_j - \boldsymbol{x}_k)} \\ &= \sum_{\boldsymbol{\ell} \in I_n} b_{\boldsymbol{\ell}} \sum_{k=1}^{N} c_k e^{2\pi i \boldsymbol{\ell} \cdot (\boldsymbol{y}_j - \boldsymbol{x}_k)} \end{aligned}$$

Now, the exponential is split using the above mentioned property, *i.e.*,

$$\mathcal{P}_f(\boldsymbol{y}_j) \approx \sum_{\boldsymbol{\ell} \in I_n} b_{\boldsymbol{\ell}} \sum_{k=1}^{N} c_k e^{-2\pi i \boldsymbol{\ell} \cdot \boldsymbol{x}_k} e^{2\pi i \boldsymbol{\ell} \cdot \boldsymbol{y}_j}.$$

This, however, can be viewed as a fast Fourier transform at the non-uniformly spaced points $\boldsymbol{y}_j$, *i.e.*,

$$\mathcal{P}_f(\boldsymbol{y}_j) \approx \sum_{\boldsymbol{\ell} \in I_n} d_{\boldsymbol{\ell}} e^{2\pi i \boldsymbol{\ell} \cdot \boldsymbol{y}_j}.$$

where the coefficients $d_{\boldsymbol{\ell}} = b_{\boldsymbol{\ell}} a_{\boldsymbol{\ell}}$ with

$$a_{\boldsymbol{\ell}} = \sum_{k=1}^{N} c_k e^{-2\pi i \boldsymbol{\ell} \cdot \boldsymbol{x}_k},$$

which in turn is nothing but an inverse discrete Fourier transform at the non-uniformly spaced points $\boldsymbol{x}_k$. These latter two transforms are dealt with numerically using the NFFT software.

Together, for the case of non-singular basic functions Φ, we have the following algorithm.

Algorithm 28.1. Fast Fourier transform evaluation

For $\boldsymbol{\ell} \in I_n$

Compute the coefficients

$$b_{\boldsymbol{\ell}} = \frac{1}{n^s} \sum_{\boldsymbol{k} \in I_n} \Phi\left(\frac{\boldsymbol{k}}{n}\right) e^{-2\pi i \boldsymbol{k} \cdot \boldsymbol{\ell}/n}$$

by inverse FFT.

Compute the coefficients

$$a_{\boldsymbol{\ell}} = \sum_{k=1}^{N} c_k e^{-2\pi i \boldsymbol{\ell} \cdot \boldsymbol{x}_k}$$

by inverse NFFT.

Compute the coefficients $d_{\boldsymbol{\ell}} = a_{\boldsymbol{\ell}} b_{\boldsymbol{\ell}}$.

end

For $1 \leq j \leq M$

Compute the values

$$\mathcal{P}_f(\boldsymbol{y}_j) \approx \sum_{\boldsymbol{\ell} \in I_n} d_{\boldsymbol{\ell}} e^{2\pi i \boldsymbol{\ell} \cdot \boldsymbol{y}_j}$$

by NFFT.

end

In the papers [Kunis *et al.* (2002); Nieslony *et al.* (2004); Potts and Steidl (2003)] the authors suggest a special boundary regularization in case the basic function does not decay fast enough, *i.e.*, the basic function is large near the boundary of the domain. However, for our experiments with Laguerre-Gaussians reported in the next section this is not an issue.

Of course, this method will only provide an approximation of the expansion (28.1) and error estimates are provided in the literature (see, *e.g.*, [Nieslony *et al.* (2004)]).

While we only illustrate the use of (N)FFTs for the evaluation of radial sums it should be clear that this method can also be coupled with any other algorithm that is based on evaluation of fast summation at non-uniform points (such as the preconditioned GMRES algorithm of Section 34.3, the "greedy" algorithm of Section 33.1, or the Faul-Powell algorithm of Section 33.2).

28.2 Approximate MLS Approximation via Non-uniform Fast Fourier Transforms

A few examples that illustrate the use of fast Fourier transforms for the evaluation of approximate moving least squares approximations (quasi-interpolants) are

taken from the paper [Fasshauer and Zhang (2004)]. We deviate from our usual policy of providing only results of experiments based on MATLAB code and include Figures 28.1–28.3, which were obtained with C++ software incorporating the NFFT library.

We use the following mollified Franke-type function in the unit cube $[0,1]^s$ in space dimensions $s = 1, 2, 3$:

$$\begin{aligned} f(x_1,x_2,x_3) = \frac{3}{4}\Big[&\exp(-\frac{(9x_1-2)^2}{4} - \frac{(9x_2-2)^2}{4} - \frac{(9x_3-2)^2}{4}) \\ &+ \exp(-\frac{(9x_1+1)^2}{49} - \frac{(9x_2+1)^2}{10} - \frac{(9x_3+1)^2}{29})\Big] \\ &+ \frac{1}{2}\exp(-\frac{(9x_1-7)^2}{4} - (9x_2-3)^2 - \frac{(9x_3-5)^2}{2}) \\ &- \frac{1}{5}\exp(-(9x_1-4)^2 - (9x_2-7)^2 - (9x_3-5)^2), \end{aligned}$$

$$g(x_1,x_2,x_3) = 15 f(x_1,x_2,x_3) \prod_{i=1}^{3} \exp(\frac{-1}{1-(2x_i-1)^2}),$$

where x_1, x_2, x_3 are used according to the space dimension s. The data sites are $N = (2^k+1)^s$ equally spaced points in the unit cube, while the errors are computed at M evaluation points randomly distributed in $[0,1]^s$ with $M = 32768$ for $s = 1$, $M = 262144$ for $s = 2$, and $M = 2146689$ for $s = 3$.

In our experiments we use $n = 4N^{1/s}$ in (28.3) for all computations except for the very last experiments in 2D and 3D. We do not have an automated strategy for choosing n. However, the values just mentioned yield satisfactory results and go along with the values suggested by Theorems 3.1 and 3.4 of [Nieslony *et al.* (2004)]. In all experiments displayed in Figures 28.1–28.3, the scale parameter $\mathcal{D}$ is taken to be 3.0.

The left plots in Figures 28.1–28.3 show the maximum error versus the number of centers N on a logarithmic scale for the three types of Laguerre-Gaussian generating functions of Table 4.1. This illustrates that the approximation does converge well (almost reaching the rates predicted by the theory) as we increase the number of data sites.

The presence of the saturation error is clearly visible in Figure 28.1. The plots on the right compare the cost of direct summation versus that for NFFT summation, and show that the efficiency is greatly improved by the use of the NFFT. Due to their long duration some of the computational times for the direct summation were omitted.

The experiments presented in this section show that it is not unreasonable to approximate large data sets with globally supported generating functions. In fact, the accuracy achieved with the global functions is far superior to that which we obtained with the compactly supported functions for similar problems in the previous chapter.

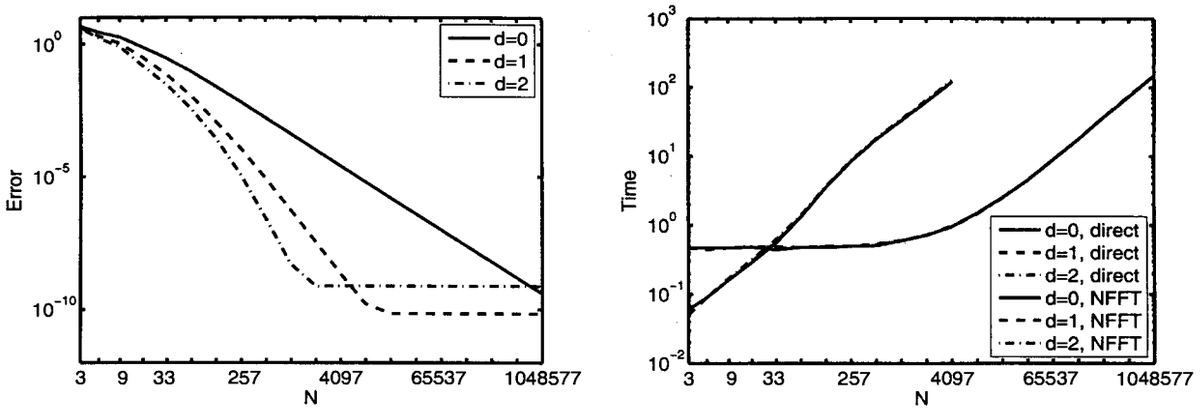


Fig. 28.1 Convergence and execution times for 1D example (Gaussian, linear and quadratic Laguerre-Gaussian generating functions).

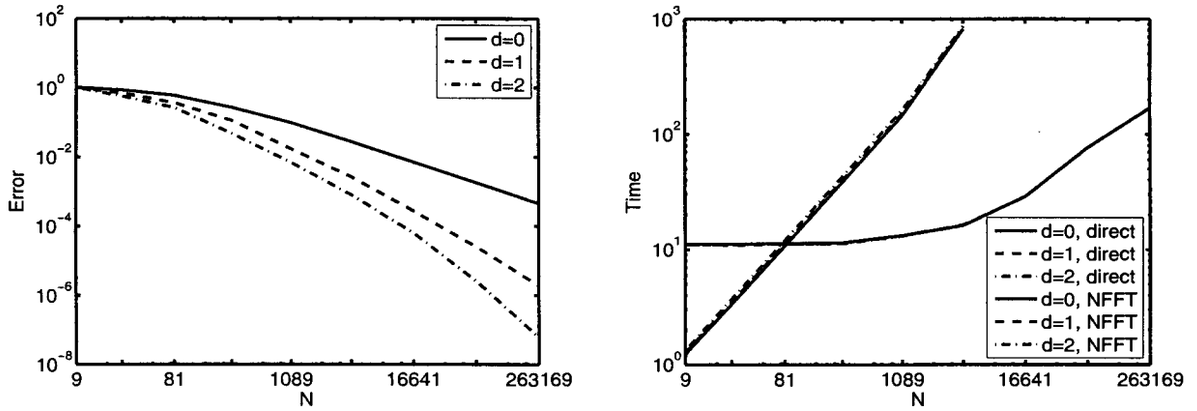


Fig. 28.2 Convergence and execution times for 2D example (Gaussian, linear and quadratic Laguerre-Gaussian generating functions).

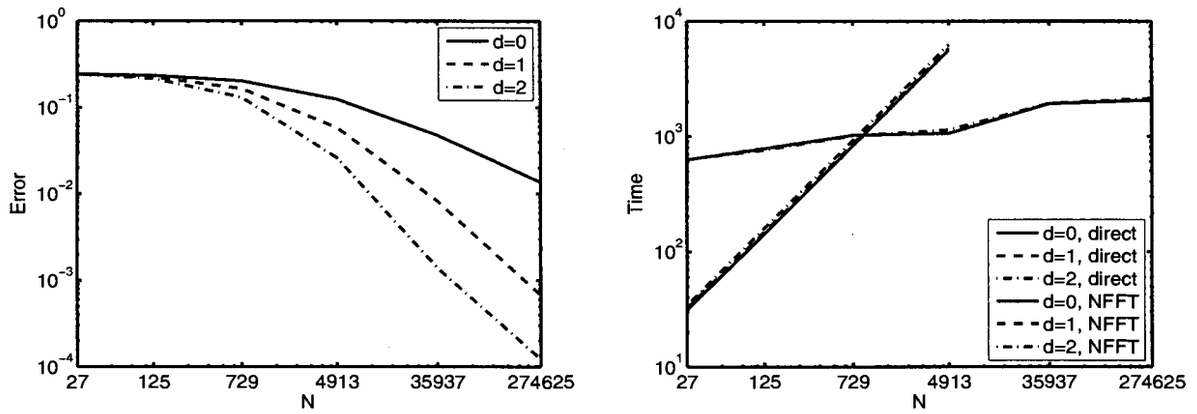


Fig. 28.3 Convergence and (predicted) execution times for 3D example (Gaussian, linear and quadratic Laguerre-Gaussian generating functions).

Chapter 29

Partition of Unity Methods

Another possibility for fast computation with meshfree approximation methods is the *partition of unity* method. This approach offers a simple way to decompose a large problem into many small problems while at the same time ensuring that the accuracy obtained for the local fits is carried over to the global fit.

29.1 Theory

The partition of unity method was suggested in [Babuška and Melenk (1997); Melenk and Babuška (1996)] in the mid 1990s in the context of meshfree Galerkin methods for the solution of partial differential equations (see Chapters 44 and 45 for a discussion of an RBF-based Galerkin approach). In the scattered data fitting context the paper [Franke (1977)] already contains a similar algorithm. We base the presentation in this section on the paper [Wendland (2002a)].

The basic idea for the partition of unity method is to start with a partition of the open and bounded domain $\Omega \subseteq \mathbb{R}^s$ into M subdomains Ω_j such that $\bigcup_{j=1}^{M} \Omega_j \supseteq \Omega$ with some mild overlap among the subdomains. Associated with these subdomains we choose a *partition of unity*, *i.e.*, a family of compactly supported, non-negative, continuous functions w_j supported on the closure of Ω_j such that at every point $\boldsymbol{x}$ in Ω we have

$$\sum_{j=1}^{M} w_j(\boldsymbol{x}) = 1. \tag{29.1}$$

Now, for every subdomain Ω_j we construct a local approximation u_j (*e.g.*, a radial basis function interpolant), and then form the global approximant to the data on the entire domain Ω via

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{j=1}^{M} u_j(\boldsymbol{x}) w_j(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega. \tag{29.2}$$

Note that if the local fits interpolate at a given data point $\boldsymbol{x}_\ell$, *i.e.*, $u_j(\boldsymbol{x}_\ell) = f(\boldsymbol{x}_\ell)$,

then the global approximant also interpolates at this point:

$$\begin{aligned}\mathcal{P}_f(\boldsymbol{x}_\ell) &= \sum_{j=1}^{M} u_j(\boldsymbol{x}_\ell) w_j(\boldsymbol{x}_\ell) \\ &= \sum_{j=1}^{M} f(\boldsymbol{x}_\ell) w_j(\boldsymbol{x}_\ell) = f(\boldsymbol{x}_\ell).\end{aligned}$$

The last equality holds due to the partition of unity property (29.1). Technically, the second sum will most likely not run over the full set of indices, $1, \ldots, M$, since $\boldsymbol{x}_\ell$ will lie only in the support of some of the w_j. Of course, this does not change the result.

In order to be able to formulate error bounds we need some technical conditions. We require the partition of unity functions to be *k-stable*, *i.e.*, we require that each $w_j \in C^k(\mathbb{R}^s)$ satisfies for every multi-index $\boldsymbol{\alpha}$ with $|\boldsymbol{\alpha}| \leq k$ the inequality

$$\|D^{\boldsymbol{\alpha}} w_j\|_{L_\infty(\Omega_j)} \leq C_{\boldsymbol{\alpha}} / \delta_j^{|\boldsymbol{\alpha}|},$$

where $C_{\boldsymbol{\alpha}}$ is some positive constant, and $\delta_j = \operatorname{diam}(\Omega_j)$.

In order to understand the following approximation theorem from [Wendland (2002a)] we need to define the space $C^k_\beta(\mathbb{R}^s)$ of all C^k functions f whose derivatives of order $|\boldsymbol{\alpha}| = k$ satisfy $D^{\boldsymbol{\alpha}} f(\boldsymbol{x}) = \mathcal{O}(\|\boldsymbol{x}\|_2^\beta)$ for $\|\boldsymbol{x}\|_2 \to 0$.

Theorem 29.1. *Suppose $\Omega \subseteq \mathbb{R}^s$ is open and bounded, and let $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subseteq \Omega$. Let $\Phi \in C^k_\beta(\mathbb{R}^s)$ be strictly conditionally positive definite of order m. Let $\{\Omega_j\}$ be a regular covering for $(\Omega, \mathcal{X})$ and let $\{w_j\}$ be k-stable for $\{\Omega_j\}$. Then the error between $f \in \mathcal{N}_\Phi(\Omega)$ and its partition of unity interpolant (29.2) with $u_j \in \operatorname{span}\{\Phi(\cdot, \boldsymbol{x}) : \boldsymbol{x} \in \mathcal{X} \cap \Omega_j\} + \Pi^s_{m-1}$ can be bounded by*

$$|D^{\boldsymbol{\alpha}} f(\boldsymbol{x}) - D^{\boldsymbol{\alpha}} \mathcal{P}_f(\boldsymbol{x})| \leq C h_{\mathcal{X},\Omega}^{\frac{k+\beta}{2} - |\boldsymbol{\alpha}|} |f|_{\mathcal{N}_\Phi(\Omega)},$$

for all $\boldsymbol{x} \in \Omega$ and all $|\boldsymbol{\alpha}| \leq k/2$.

The regularity assumptions on the subdomains Ω_j are:

- For every $\boldsymbol{x} \in \Omega$ the number of subdomains Ω_j with $\boldsymbol{x} \in \Omega_j$ is bounded by a global constant K.
- Every subdomain Ω_j satisfies an interior cone condition (*c.f.* Definition 14.2).
- The local fill distances $h_{\mathcal{X}_j,\Omega_j}$ are uniformly bounded by the global fill distance $h_{\mathcal{X},\Omega}$, where $\mathcal{X}_j = \mathcal{X} \cap \Omega_j$.

If we compare this with the global error estimates from Chapter 15 we see that the partition of unity preserves the local approximation order for the global fit. Thus, we can efficiently compute large RBF interpolants by solving many small RBF interpolation problems (in parallel if we wish) and then glue them together with the global partition of unity $\{w_j\}$.

A simple way to obtain a partition of unity is via a Shepard approximant (*c.f.* Chapter 23). Therefore, we can think of the partition of unity method as a Shepard

method with higher-order data. Namely, the "data" are now given by the local approximations u_j instead of just the values $f(\boldsymbol{x}_j)$. The benefits of this kind of approach seem to have first been realized in [Franke (1977)].

29.2 Partition of Unity Approximation with MATLAB

The MATLAB program for the partition of unity approximation based on local RBF interpolants is again rather similar to our earlier programs. The main difference is that we now also have to create the subdomains Ω_j (which we will do as overlapping circles) and the associated partition of unity $\{w_j\}$ (for which we use a Shepard method based on Wendland's compactly supported RBFs) .

The compactly supported radial weight functions on line 1 of Program 29.1 and the RBF on line 2 are used in the shifted form $\widetilde{\varphi}_{s,k} = \varphi_{s,k}(1 - \cdot)$ (cf. Table 11.1). Note that we use the *kd*-tree routines to build two trees: a data tree, and an evaluation tree. Inside the loop over all partition of unity cells (lines 27–38) we first use kdrangequery to find all data sites in cell j, build the local interpolation matrix based on these points, and then repeat the process for the evaluation points. Note that the contributions to the final global fit are accumulated cell by cell (see line 36).

Program 29.1. PU2D_CS.m

```
% PU2D_CS
% Script that performs partition of unity approximation using
% sparse matrices
% Calls on: DistanceMatrixCSRBF
% Uses:     k-D tree package by Guy Shechter
%           from MATLAB Central File Exchange
    % Weight function for global Shepard partition of unity weighting
 1  wf = @(e,r) r.^4.*(5*spones(r)-4*r);
    % RBF basis function for local RBF interpolation
 2  rbf = @(e,r) r.^4.*(5*spones(r)-4*r);
 3  ep = 0.1; % Parameter for local basis functions
    % Define Franke's function as testfunction
 4  f1 = @(x,y) 0.75*exp(-((9*x-2).^2+(9*y-2).^2)/4);
 5  f2 = @(x,y) 0.75*exp(-((9*x+1).^2/49+(9*y+1).^2/10));
 6  f3 = @(x,y) 0.5*exp(-((9*x-7).^2+(9*y-3).^2)/4);
 7  f4 = @(x,y) 0.2*exp(-((9*x-4).^2+(9*y-7).^2));
 8  testfunction = @(x,y) f1(x,y)+f2(x,y)+f3(x,y)-f4(x,y);
 9  N = 1089; gridtype = 'h';
    % Parameter for npu-by-npu grid of PU cells in unit square
10  npu = 16;
    % Parameter for neval-by-neval evaluation grid in unit square
```

```
11  neval = 40;
    % Load data points
12  name = sprintf('Data2D_%d%s',N,gridtype); load(name)
13  ctrs = dsites;
14  rhs = testfunction(dsites(:,1),dsites(:,2));
15  wep = npu; % Parameter for weight function
    % Create neval-by-neval equally spaced evaluation locations
    % in the unit square
16  grid = linspace(0,1,neval); [xe,ye] = meshgrid(grid);
17  epoints = [xe(:) ye(:)];
    % Create npu-by-npu equally spaced centers of PU cells in the
    % unit square
18  pugrid = linspace(0,1,npu); [xpu,ypu] = meshgrid(pugrid);
19  cellctrs = [xpu(:) ypu(:)];
20  cellradius = 1/wep;
    % Compute Shepard evaluation matrix
21  DM_eval = DistanceMatrixCSRBF(epoints,cellctrs,wep);
22  SEM = wf(ep,DM_eval);
23  SEM = spdiags(1./(SEM*ones(npu^2,1)),0,neval^2,neval^2)*SEM;
    % Build k-D trees for data sites and evaluation points
24  [tmp,tmp,datatree] = kdtree(dsites,[]);
25  [tmp,tmp,evaltree] = kdtree(epoints,[]);
26  Pf = zeros(neval^2,1);  % initialize
27  for j=1:npu^2
    % Find data sites in cell j
28a    [pts,dist,idx] = kdrangequery(datatree,...
28b                     cellctrs(j,:),cellradius);
29     if (length(idx) > 0)
    % Build local interpolation matrix for cell j
30a       DM_data = DistanceMatrixCSRBF(dsites(idx,:),...
30b                           ctrs(idx,:),ep);
31        IM = rbf(ep,DM_data);
    % Find evaluation points in cell j
32a       [epts,edist,eidx] = kdrangequery(evaltree,...
32b                             cellctrs(j,:),cellradius);
    % Compute local evaluation matrix
33a       DM_eval = DistanceMatrixCSRBF(epoints(eidx,:),...
33b                           ctrs(idx,:),ep);
34        EM = rbf(ep,DM_eval);
    % Compute local RBF interpolant
35        localfit = EM * (IM\rhs(idx));
    % Accumulate global fit
```

```
36          Pf(eidx) = Pf(eidx) + localfit.*SEM(eidx,j);
37        end
38    end
      % Compute exact solution
39    exact = testfunction(epoints(:,1),epoints(:,2));
      % Compute maximum error on evaluation grid
40    maxerr = norm(Pf-exact,inf);
41    rms_err = norm(Pf-exact)/neval;
42    fprintf('RMS error:     %e\n', rms_err)
43    fprintf('Maximum error: %e\n', maxerr)
      % Plot interpolant
44    fview = [160,20];
45    PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview);
      % Plot maximum error
46    PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview);
```

In Tables 29.1 and 29.2 we illustrate how the local convergence order is maintained globally for a partition of unity based on Wendland's C^2 compactly supported RBFs. In both tables the local approximations are computed via either the compactly supported C^2 functions of Wendland, or via Gaussians (that are globally supported on the local subdomains). The data were sampled from Franke's function at various sets of uniformly spaced points (in Table 29.1) and Halton points (in Table 29.2) in the unit square. The M local subdomains were given by circles centered at equally spaced points in the unit square. We can see that (especially on the uniformly spaced data sites) the partition of unity method reflects the approximation behavior of the local methods. For the compactly supported Wendland functions we obtain $\mathcal{O}(h^2)$ throughout our series of experiments (with a theoretically predicted local order of $\mathcal{O}(h^{3/2})$), whereas for the Gaussians we obtain an approximation behavior in places vaguely suggestive of exponential convergence. For these experiments the fill distances for the sets of Halton points were not estimated via (2.4). Instead, we assumed that the fill distance decreases by a factor of two from one iteration to the next, as it does in the case of uniformly distributed points.

A relatively large uniform value of the shape parameter ε was used for the Gaussians on the uniform data sets in Table 29.1 to obtain the exponential convergence results. Use of the same value of ε on the Halton data sets results in RMS-errors for the Gaussians that are *worse* than those for local interpolants based on compactly supported RBFs (see Table 29.2). For the local interpolants based on the Wendland functions we used a large support radius of $\rho = 1/\varepsilon = 10$ in accordance to the observations made in Chapter 17. The main reason for the relatively poor quality of the local interpolants based on Gaussians in the Halton setting is that all of the computations with the Wendland functions are performed with well-conditioned interpolation matrices (in spite of the large support radius, *i.e.*, flat basis functions).

Note that the RMS-error for local Gaussian interpolation in the last row of

Table 29.1 2D partition of unity approximation on uniform points with Wendland's C^2 compactly supported functions for partition of unity and local Wendland and Gaussian interpolants.

		Wendland			Gaussian		
N	M	RMS-error	rate	ε	RMS-error	rate	ε
9	1	3.475816e-001		0.1	3.703960e-001		6
25	4	1.301854e-001	1.4168	0.1	1.229046e-001	1.5915	6
81	16	8.089165e-003	4.0084	0.1	2.413852e-002	2.3481	6
289	64	1.183369e-003	2.7731	0.1	4.551325e-003	2.4070	6
1089	256	2.716542e-004	2.1231	0.1	5.393771e-004	3.0769	6
4225	1024	6.795949e-005	1.9990	0.1	1.112507e-005	5.5994	6
16641	4096	1.697195e-005	2.0015	0.1	2.031757e-006	2.4530	6
66049	16384	4.241184e-006	2.0006	0.1	1.798274e-007	3.4980	6
263169	65536	9.778231e-007	2.1168	0.1	2.657751e-008	2.7583	6
1050625	262144	2.557991e-007	1.9346	0.1	1.844820e-009	3.8487	6

Table 29.2 2D partition of unity approximation on Halton points with Wendland's C^2 compactly supported functions for partition of unity and local Wendland and Gaussian interpolants.

		Wendland			Gaussian		
N	M	RMS-error	rate	ε	RMS-error	rate	ε
9	1	3.325975e-001		0.1	3.384117e-001		6
25	4	1.355396e-001	1.2951	0.1	1.383321e-001	1.2906	6
81	16	1.035963e-002	3.7097	0.1	3.640953e-002	1.9257	6
289	64	2.569458e-003	2.0114	0.1	1.030984e-002	1.8203	6
1089	256	5.860966e-004	2.1323	0.1	3.543463e-003	1.5408	6
4225	1024	2.703318e-004	1.1164	0.1	1.045103e-003	1.7615	6
16641	4096	7.701234e-005	1.8116	0.1	3.896345e-004	1.4235	6
66049	16384	4.492321e-005	0.7776	0.1	5.012220e-005	2.9586	6
263169	65536	1.589134e-005	1.4992	0.1	1.631609e-005	1.6192	6
1050625	262144	1.032629e-006	3.9438	0.1	2.000238e-006	3.0281	6

Table 29.1 presents the best approximation of Franke's test function reported in this book. However, in Table 17.5 we needed only $N = 4225$ uniformly spaced points for a global Gaussian interpolant with $\varepsilon = 6.3$ to achieve an RMS-error of 7.371879e-009. Of course, it may be possible to obtain even better approximations with other RBFs. We do not claim that Gaussians are the "best" RBFs. However, we do recommend the partition of unity approach for the solution of large interpolation or approximation problems since it is relatively simple to implement and its execution is quite efficient.

Chapter 30

Approximation of Point Cloud Data in 3D

30.1 A General Approach via Implicit Surfaces

A common problem in computer graphics and computer aided design (CAD) is the reconstruction of a three-dimensional surface defined in terms of *point cloud data*, *i.e.*, as a set of unorganized, irregular points in 3D. For example, this could be laser range data obtained for the purpose of computer modeling of a complicated 3D object. Such applications also arise, *e.g.*, in computer graphics or in medical imaging. An approach to obtaining a surface that fits the given 3D point cloud data that has recently become rather popular (see, *e.g.*, [Carr *et al.* (1997); Carr *et al.* (2001); Morse *et al.* (2001); Ohtake *et al.* (2003a); Ohtake *et al.* (2003b); Turk and O'Brien (2002); Wendland (2002b)]) is based on the use of *implicit surfaces* defined in terms of some meshfree approximation method such as an RBF interpolant or an MLS approximant.

More precisely, given data of the form $\{\boldsymbol{x}_i = (x_i, y_i, z_i) \in \mathbb{R}^3,\ i = 1, \ldots, N\}$ assumed to come from some two-dimensional manifold $\mathcal{M}$ (*i.e.*, a surface in $\mathbb{R}^3$), we seek another surface $\mathcal{M}^*$ that is a reasonable approximation to $\mathcal{M}$. For the implicit surface approach we think of $\mathcal{M}$ as the surface of all points (x, y, z) that satisfy the implicit equation

$$f(x, y, z) = 0$$

for some function f. Thus, the function f implicitly defines the surface $\mathcal{M}$. In other words, the equation $f(x, y, z) = 0$ defines the zero iso-surface of the trivariate function f and therefore this iso-surface coincides with $\mathcal{M}$.

As so often before, we will construct the surface $\mathcal{M}^*$ via interpolation. Obviously, if we only specify the interpolant to be zero at the data points, then we will obtain a zero interpolant, and will not be able to extract a meaningful iso-surface. Therefore, the key to finding an approximation to the trivariate function f from the given data points $\boldsymbol{x}_i$, $i = 1, \ldots, N$, is to add an extra set of *off-surface points* to the data so that we can then compute a (solid) three-dimensional interpolant $\mathcal{P}_f$ to the total set of points, *i.e.*, the surface points plus the auxiliary off-surface points. This will result in **a** nontrivial interpolant, and we will then be able to extract its

zero iso-surface. To illustrate this technique we discuss how this idea works in the 2D setting in the next section.

The addition of off-surface points results in a problem of the same type as the scattered data approximation problems discussed in earlier chapters. In particular, if the data sets are large, it is advisable to use either a local radial basis interpolant, a moving-least squares approach, or a partition of unity interpolant. However, there are also implicit point cloud interpolants based on fast evaluation algorithms with global RBFs (see, *e.g.*, [Carr *et al.* (2001)]).

The surface reconstruction problem consists of three sub-problems:

(1) Construct the extra off-surface points.
(2) Find the trivariate meshfree approximant to the augmented data set.
(3) Render the iso-surface (zero-contour) of the fit computed in step (2).

In order to keep the discussion as simple as possible we assume that, in addition to the point cloud data, we are also given a set of surface normals $\boldsymbol{n}_i = (n_i^x, n_i^y, n_i^z)$ to the surface $\mathcal{M}$ at the points $\boldsymbol{x}_i = (x_i, y_i, z_i)$. If these normals are not explicitly given, there are techniques available that can be used to estimate the normals (see, *e.g.*, the discussion in [Wendland (2002b)]). Once we have the (oriented) surface normals, we construct the extra off-surface points by marching a small distance along the surface normal, *i.e.*, we obtain for each data point (x_i, y_i, z_i) two additional off-surface points. One point lies "outside" the manifold $\mathcal{M}$ and is given by

$$(x_{N+i}, y_{N+i}, z_{N+i}) = \boldsymbol{x}_i + \delta \boldsymbol{n}_i = (x_i + \delta n_i^x, y_i + \delta n_i^y, y_i + \delta n_i^z),$$

and the other point lies "inside" $\mathcal{M}$ and is given by

$$(x_{2N+i}, y_{2N+i}, z_{2N+i}) = \boldsymbol{x}_i - \delta \boldsymbol{n}_i = (x_i - \delta n_i^x, y_i - \delta n_i^y, y_i - \delta n_i^z).$$

Here δ is a small step size (whose specific magnitude can be rather critical for a good surface fit, see [Carr *et al.* (2001)]). In particular, if δ is chosen too large, then this can easily result in self-intersecting inner or outer auxiliary surfaces. In our simple MATLAB implementation in the next section we uniformly take δ to be 1% of the maximum dimension of the bounding box of the data as suggested in [Wendland (2002b)].

Once we have created the auxiliary data, the interpolant is computed by determining a function $\mathcal{P}_f$ whose zero contour interpolates the given point cloud data, and whose "inner" and "outer" offset contours interpolate the augmented data, *i.e.*,

$$\begin{aligned} \mathcal{P}_f(\boldsymbol{x}_i) &= 0, \qquad i = 1, \ldots, N, \\ \mathcal{P}_f(\boldsymbol{x}_i) &= 1, \qquad i = N+1, \ldots, 2N, \\ \mathcal{P}_f(\boldsymbol{x}_i) &= -1, \qquad i = 2N+1, \ldots, 3N. \end{aligned}$$

The values of ± 1 for the auxiliary data are arbitrary. Their precise value is not as critical as the choice of δ.

For the third step we also use a very simple solution, namely we just render the resulting approximating surface $\mathcal{M}^*$ as the zero contour of the 3D interpolant. In

MATLAB this can be accomplished with the command `isosurface` (or `contour` for 2D problems). This provides a rough picture of the implicit surface interpolant, but may also lead to some rendering artifacts that can be avoided with more sophisticated rendering procedures. For more serious applications one usually employs some variation of a *ray tracing* or *marching cube* algorithm (see the discussion in the references listed above).

The implicit surface representation has the advantage that the surface normals of the approximating surface $\mathcal{M}^*$ can be explicitly and analytically calculated as the gradients of the trivariate function $\mathcal{P}_f$, *i.e.*, $\boldsymbol{n}(\boldsymbol{x}) = \nabla \mathcal{P}_f(\boldsymbol{x})$.

Since in practice the data (*e.g.*, obtained by laser range scanners) is subject to measurement errors it is often beneficial if an additional smoothing procedure is employed. One can either use the ridge regression approach suggested in Chapter 19, or use a moving least squares approximation instead of an RBF interpolant. Another implicit smoothing technique was suggested in [Beatson and Bui (2003)]. Noisy data can also be dealt with by using a multilevel technique such as suggested in [Ohtake *et al.* (2003b)]. We discuss multilevel interpolation and approximation algorithms in Chapter 32.

30.2 An Illustration in 2D

Since the 3D point cloud interpolation problem requires an interpolant to points viewed as samples of a trivariate function whose graph is a 4D hypersurface, the visualization of the individual steps of the construction of the final iso-surface is problematic. We therefore illustrate the process with an analogous two-dimensional problem, *i.e.*, we assume we are given points (taken from a closed curve $\mathcal{C}$) in the plane, and it is our goal to find an interpolating curve $\mathcal{C}^*$. Below we present both MATLAB code and several figures.

Program 30.1. `PointCloud2D.m`

```
% PointCloud2D
% Script that fits a curve to 2D point cloud
% Calls on: DistanceMatrix
% Uses:     haltonseq (written by Daniel Dougherty
%                      from MATLAB Central File Exchange)
    % Gaussian RBF
 1  rbf = @(e,r) exp(-(e*r).^2); ep = 3.5;
 2  N = 81;   % number of data points
 3  neval = 40; % to create neval-by-neval evaluation grid
 4  t = 2*pi*haltonseq(N,1); dsites = [cos(t) sin(t)];
 5  x = (2+sin(t)).*cos(t); y = (2+cos(t)).*sin(t);
 6  nx = (2+cos(t)).*cos(t)-sin(t).^2;
```

```
 7  ny = (2+sin(t)).*sin(t)-cos(t).^2;
 8  dsites = [x y]; normals = [nx ny];
    % Produce auxiliary points along normals "inside" and "outside"
 9  bmin = min(dsites,[],1);  bmax = max(dsites,[],1);
10  bdim = max(bmax-bmin);
    % Distance along normal at which to place new points
11  delta = bdim/100;
    % Create new points
12  dsites(N+1:2*N,:) = dsites(1:N,:) + delta*normals;
13  dsites(2*N+1:3*N,:) = dsites(1:N,:) - delta*normals;
    % "original" points have rhs=0,
    % "inside" points have rhs=-1, "outside" points have rhs=1
14  rhs = [zeros(N,1); ones(N,1); -ones(N,1)];
    % Let centers coincide with data sites
15  ctrs = dsites;
    % Compute new bounding box
16  bmin = min(dsites,[],1);  bmax = max(dsites,[],1);
    % Create neval-by-neval equally spaced evaluation locations
    % in bounding box
17  xgrid = linspace(bmin(1),bmax(1),neval);
18  ygrid = linspace(bmin(2),bmax(2),neval);
19  [xe,ye] = meshgrid(xgrid,ygrid);
20  epoints = [xe(:) ye(:)];
21  DM_eval = DistanceMatrix(epoints,ctrs);
22  EM = rbf(ep,DM_eval);
23  DM_data = DistanceMatrix(dsites,ctrs);
24  IM = rbf(ep,DM_data);
25  Pf = EM * (IM\rhs);
    % Plot extended data with 2D-fit Pf
26  figure; hold on; view([-30,30])
27  plot3(dsites(:,1),dsites(:,2),rhs,'r.','markersize',20);
28  mesh(xe,ye,reshape(Pf,neval,neval));
29  axis tight; hold off
    % Plot data sites with interpolant (zero contour of 2D-fit Pf)
30  figure; hold on
31  plot(dsites(1:N,1),dsites(1:N,2),'bo');
32  contour(xe,ye,reshape(Pf,neval,neval),[0 0],'r');
33  hold off
```

In the MATLAB program `PointCloud2D.m` (see Program 30.1) we create test data on lines 4–8 by sampling a parametric curve (in polar coordinates) at irregular parameter values `t`. These points are displayed in the left plot of Figure 30.1.

Since we use a known representation of the curve to generate the point cloud it is also possible to obtain an exact normal vector associated with each data point (see lines 6–8).

The strategy for creation of the auxiliary points is the same as above, *i.e.*, we add data points "inside" and "outside" the given point set. This is done by placing these points along the normal vector at each original data point (see lines 12 and 13). The distance along the normal at which the auxiliary points are placed is taken to be 1% of the size of the maximum dimension of the bounding box of the original data (see lines 9–11).

Next, the problem is turned into a full 2D interpolation problem (whose solution has a 3D graph) by adding function values (of the unknown bivariate function f whose zero-level iso-curve will be the desired interpolating curve) at the extended data set. We assign a value of 0 to each original data point, and a value of 1 or -1 to "outside" or "inside" points, respectively. This is done on line 14 of the code and the resulting data is displayed in the right plot of Figure 30.1 (*c.f.* also line 27 of the code).

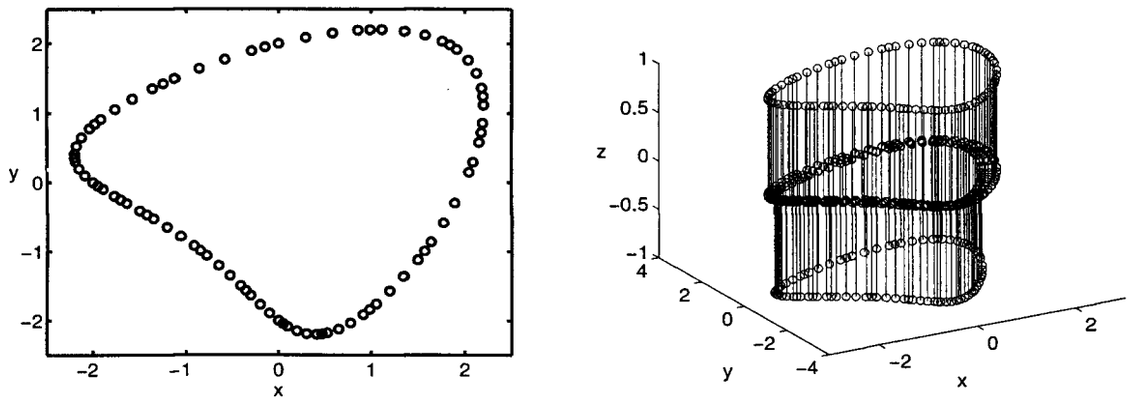


Fig. 30.1 Point cloud data (left) and extended "inner" and "outer" data (right) for implicit curve with 81 non-uniform data points.

Now we can solve the problem just like any of our 2D interpolation problems discussed earlier. In fact, in Program 30.1 we use straightforward RBF interpolation with Gaussian RBFs (see lines 1 and 19–25).

Finally, the zero contour of the resulting surface is extracted on line 32 using the `contour` command. In the left plot of Figure 30.2 we display a surface plot of the bivariate RBF interpolant to the extended data set (obtained via the `mesh` command on line 28), and in the right plot we show the final interpolating curve along with the original data.

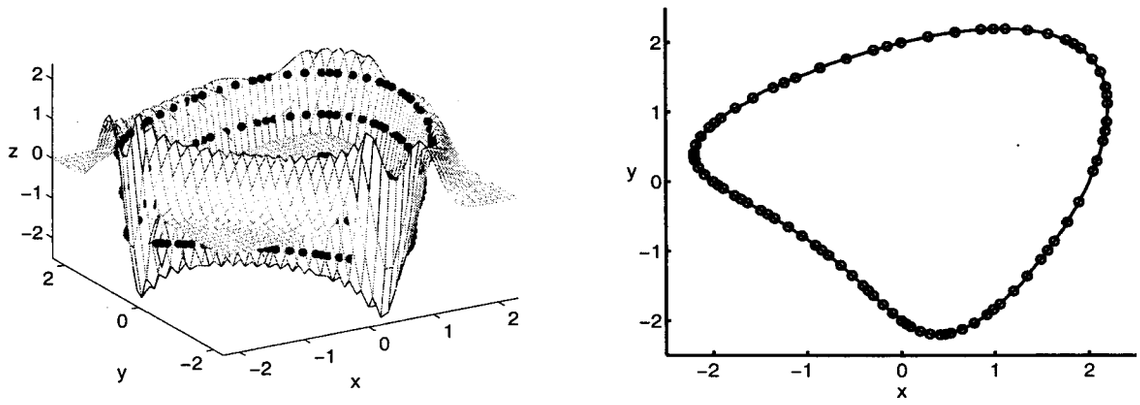


Fig. 30.2 Surface fit (left) and zero contour for 81 non-uniform data points.

30.3 A Simplistic Implementation in 3D via Partition of Unity Approximation in MATLAB

In the MATLAB program `PointCloud3D_PUCS` (see Program 30.2) we present a fairly simple implementation of the partition of unity approach to interpolation of point cloud data in 3D. The partition of unity is created with a Shepard approximant as in the previous chapter. As in Program 29.1 we use Wendland's C^2 compactly supported function as both the Shepard weights and for the local RBF interpolants.

The data sets used in our examples correspond to various resolutions of the Stanford bunny available on the world-wide web at `http://graphics.stanford.edu/~data/3Dscanrep/`. Data sets consisting of 35947, 8171, 1889, and 453 points are included in the file `bunny.tar.gz`. The normals for this kind of `PLY` data can be computed with the utility `normalsply` from the package `ply.tar.gz` provided by Greg Turk, and available on the world-wide web at `http://www.cc.gatech.edu/projects/large_models/ply.html`. Results obtained with the `PointCloud3D_PUCS` for the 453 and 8171 point cloud sets are displayed in Figure 30.3. Processed data files are included on the enclosed CD.

Many parts of the MATLAB code for `PointCloud3D_PUCS` are similar to Program 29.1. The bunny data set including point normals is loaded on line 6, and the bounding box for the point cloud and its maximum dimension are computed on lines 7–8. The off-surface points are added in lines 10–14. Then the right-hand side for the augmented (3D) interpolation problem is defined on line 15 (assigning a value of 0 for the on-surface data points, and a value of ± 1 for the "outside" and "inside" off-surface points), and we recompute the bounding box for the augmented data on line 16.

We have found that a reasonable value for the radius of the partition of unity subdomains seems to be given by the maximal dimension of the bounding box divided by the cube root of the number, M, of subdomains, *i.e.*, $\mathrm{diam}(\Omega_j) = 1/w_\varepsilon$ with $w_\varepsilon = \sqrt[3]{M}/\mathrm{bdim}$ (see lines 9 and 28).

The main part of the program (lines 29–46) is almost identical to lines 21–38 of Program 29.1. On lines 47–55 we add the code that creates and displays the zero-contour iso-surface for the 3D (solid) interpolant `Pf` along with the point cloud data.

Program 30.2. `PointCloud3D_PUCS.m`

```
% PointCloud3D_PUCS
% Script that fits a surface to 3D point cloud using partition of
% unity approximation with sparse matrices
% Calls on: CSEvalMatrix
% Uses:     k-D tree package by Guy Shechter
%           from MATLAB Central File Exchange
    % Weight function for global Shepard partition of unity weighting
 1  wf = @(e,r) r.^4.*(5*spones(r)-4*r);
    % The RBF basis function for local RBF interpolation
 2  rbf = @(e,r) r.^4.*(5*spones(r)-4*r);
 3  ep = 1; % Parameter for basis function
    % Parameter for npu-by-npu-by-npu grid of PU cells
 4  npu = 8;
    % Parameter for npu-by-npu-by-npu grid of PU cells
 5  neval = 25;
    % Load data points and compute bounding box
 6  load('Data3D_Bunny3');  N = size(dsites,1);
 7  bmin = min(dsites,[],1);  bmax = max(dsites,[],1);
 8  bdim = max(bmax-bmin);
 9  wep = npu/bdim;
    % Add auxiliary points along normals "inside" and "outside"
    % Find points with nonzero normal vectors and count them
10  withnormals = find(normals(:,1)|normals(:,2)|normals(:,3));
11  addpoints = length(withnormals);
    % Distance along normal at which to place new points
12  delta = bdim/100;
    % Create new points
13a dsites(N+1:N+addpoints,:) = ...
13b      dsites(withnormals,:) + delta*normals(withnormals,:);
14a dsites(N+addpoints+1:N+2*addpoints,:) = ...
14b      dsites(withnormals,:) - delta*normals(withnormals,:);
    % Interpolant is implicit surface, i.e.,
    % "original" points have rhs=0, "inside" rhs=-1, "outside" rhs=1
15  rhs = [zeros(N,1); ones(addpoints,1); -ones(addpoints,1)];
    % Compute new bounding box
16  bmin = min(dsites,[],1);  bmax = max(dsites,[],1);
```

```
17  ctrs = dsites;
    % Create neval-by-neval-by-neval equally spaced evaluation
    % locations in bounding box
18  xgrid = linspace(bmin(1),bmax(1),neval);
19  ygrid = linspace(bmin(2),bmax(2),neval);
20  zgrid = linspace(bmin(3),bmax(3),neval);
21  [xe,ye,ze] = meshgrid(xgrid,ygrid,zgrid);
22  epoints = [xe(:) ye(:) ze(:)];
    % Create npu-by-npu-by-npu equally spaced centers of PU cells
    % in bounding box
23  puxgrid = linspace(bmin(1),bmax(1),npu);
24  puygrid = linspace(bmin(2),bmax(2),npu);
25  puzgrid = linspace(bmin(3),bmax(3),npu);
26  [xpu,ypu,zpu] = meshgrid(puxgrid,puygrid,puzgrid);
27  cellctrs = [xpu(:) ypu(:) zpu(:)];
28  cellradius = 1/wep;
    % Compute Shepard evaluation matrix
29  DM_eval = DistanceMatrixCSRBF(epoints,cellctrs,wep);
30  SEM = wf(wep,DM_eval);
31  SEM = spdiags(1./(SEM*ones(npu^3,1)),0,neval^3,neval^3)*SEM;
    % Build k-D trees for data sites and evaluation points
32  [tmp,tmp,datatree] = kdtree(dsites,[]);
33  [tmp,tmp,evaltree] = kdtree(epoints,[]);
34  Pf = zeros(neval^3,1); % initialize
35  for j=1:npu^3
       % Find data sites in cell j
36a    [pts,dist,idx] = kdrangequery(datatree,...
36b                     cellctrs(j,:),cellradius);
37     if (length(idx) > 0)
          % Build local interpolation matrix for cell j
38a       DM_data = DistanceMatrixCSRBF(dsites(idx,:),...
38b                 ctrs(idx,:),ep);
39        IM = rbf(ep,DM_data);
          % Find evaluation points in cell j
40a       [epts,edist,eidx] = kdrangequery(evaltree,...
40b                           cellctrs(j,:),cellradius);
          % Compute local evaluation matrix
41a       DM_eval = DistanceMatrixCSRBF(epoints(eidx,:),...
41b                 ctrs(idx,:),ep);
42        EM = rbf(ep,DM_eval);
          % Compute local RBF interpolant
43        localfit = EM * (IM\rhs(idx));
```

```
          % Accumulate global fit
44        Pf(eidx) = Pf(eidx) + localfit.*SEM(eidx,j);
45     end
46  end
    % Plot data sites with interpolant (zero contour of 3D-fit Pf)
47  figure; hold on
48  plot3(dsites(1:N,1),dsites(1:N,2),dsites(1:N,3),'bo');
49a pfit = patch(isosurface(xe,ye,ze,...
49b                       reshape(Pf,neval,neval,neval),0));
50  isonormals(xe,ye,ze,reshape(Pf,neval,neval,neval),pfit)
51a set(pfit,'FaceLighting','gouraud','FaceColor',...
51b     'red','EdgeColor','none');
52  light('Position',[0 0 1],'Style','infinite');
53  daspect([1 1 1]); view([0,90]);
54  axis([bmin(1) bmax(1) bmin(2) bmax(2) bmin(3) bmax(3)]);
55  axis off;  hold off
```

In Figure 30.3 we display partition of unity fits based on local RBF interpolants built with compactly supported Wendland's C^2 basis functions. The point cloud data sets consist of 453 points (top plots in Figure 30.3) and 8171 points (bottom plots in Figure 30.3). The augmented data sets for the 3D interpolants are almost three times as large (since not every data point has a normal vector associated with it). On the left we show the fitted surface along with the data points, and on the right the fit is displayed by itself. For the plots in the bottom part of Figure 30.3 the simple iso-surface plot in MATLAB does a surprisingly good job. In other situations, however, it causes some artifacts such as the extra surface fragment near the bunny's ear in the top part of Figure 30.3.

For the top plots in Figure 30.3 we used $8^3 = 256$ subdomains and $25^3 = 15625$ evaluation points in the bounding box of the 3D data, and for those on the bottom of Figure 30.3 we used $32^3 = 32768$ subdomains and $50^3 = 125000$ evaluation points.

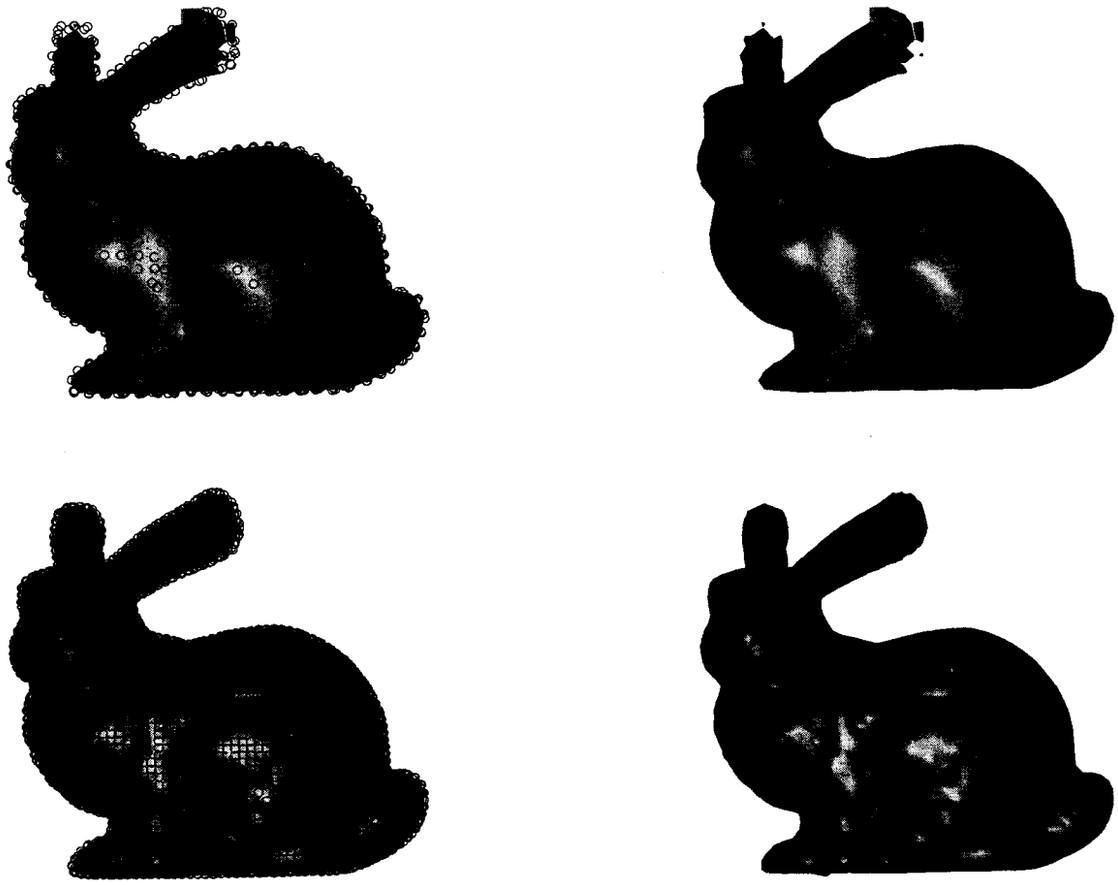


Fig. 30.3 Partition of unity implicit surface interpolant to Stanford bunny with 453 (top) and 8171 (bottom) data points.

Chapter 31

Fixed Level Residual Iteration

In the next few chapters we will look at various versions of residual iteration. The basic idea of using an iterative algorithm in which one takes advantage of the residual of an initial approximation to obtain a more accurate solution is well-known in many branches of mathematics.

31.1 Iterative Refinement

For example, in numerical linear algebra this process is known as *iterative refinement* (see, *e.g.*, [Kincaid and Cheney (2002)]). We might be interested in solving a system of linear equations $A\boldsymbol{x} = \boldsymbol{b}$, and obtain a (numerical) solution $\boldsymbol{x}_0$ by applying an algorithm such as Gaussian elimination. We can then compute the residual $\boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{x}_0$, and realize that it is related to the error, $\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}_0$, via the relation

$$A\boldsymbol{e} = A\boldsymbol{x} - A\boldsymbol{x}_0 = \boldsymbol{r}. \tag{31.1}$$

Thus, by adding the (numerical) solution $\boldsymbol{e}_0$ of equation (31.1) to the initial solution $\boldsymbol{x}_0$ one expects to improve the initial approximation to $\boldsymbol{x}_1 = \boldsymbol{x}_0 + \boldsymbol{e}_0$ (since the true solution $\boldsymbol{x} = \boldsymbol{x}_0 + \boldsymbol{e}$). Of course, this procedure can be repeated iteratively. This leads to the algorithm

Algorithm 31.1. Iterative refinement

(1) Compute an approximate solution $\boldsymbol{x}_0$ of $A\boldsymbol{x} = \boldsymbol{b}$.
(2) For $k = 1, 2, \ldots$ do
   (a) Compute the residual $\boldsymbol{r}_k = \boldsymbol{b} - A\boldsymbol{x}_{k-1}$.
   (b) Solve $A\boldsymbol{e}_k = \boldsymbol{r}_k$.
   (c) Update $\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \boldsymbol{e}_k$.

We can rewrite the last statement in the algorithm as

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + B(\boldsymbol{b} - A\boldsymbol{x}_{k-1}), \tag{31.2}$$

where B is an *approximate (or numerical) inverse* of A characterized by the property that $\|I - BA\| < 1$ for some matrix norm. This condition allows us to express the

exact inverse of A by a Neumann series, *i.e.*,

$$A^{-1} = (BA)^{-1}B = \left[\sum_{j=0}^{\infty}(I - BA)^j\right] B.$$

Therefore the exact solution of $A\boldsymbol{x} = \boldsymbol{b}$ can be written in the form

$$\boldsymbol{x} = A^{-1}\boldsymbol{b} = \left[\sum_{j=0}^{\infty}(I - BA)^j\right] B\boldsymbol{b}. \tag{31.3}$$

For the iterative refinement algorithm, on the other hand, we have from (31.2) and the fact that $\boldsymbol{x}_0 = B\boldsymbol{b}$, that

$$\begin{aligned}\boldsymbol{x}_k &= \boldsymbol{x}_{k-1} + B(\boldsymbol{b} - A\boldsymbol{x}_{k-1}) \\ &= (I - BA)\boldsymbol{x}_{k-1} + B\boldsymbol{b} \\ &= (I - BA)\boldsymbol{x}_{k-1} + \boldsymbol{x}_0. \end{aligned} \tag{31.4}$$

We can recursively substitute this relation back in for $\boldsymbol{x}_{k-1}$, $\boldsymbol{x}_{k-2}$, etc., and obtain

$$\boldsymbol{x}_k = \left[\sum_{j=0}^{k}(I - BA)^j\right] \boldsymbol{x}_0 = \left[\sum_{j=0}^{k}(I - BA)^j\right] B\boldsymbol{b}. \tag{31.5}$$

It is now easy to see that the iterates $\boldsymbol{x}_k$ of the refinement algorithm converge to the exact solution $\boldsymbol{x}$. We simply look at the difference $\boldsymbol{x} - \boldsymbol{x}_k$ at level k, *i.e.*, from (31.3) and (31.5) we obtain

$$\boldsymbol{x} - \boldsymbol{x}_k = \left[\sum_{j=k+1}^{\infty}(I - BA)^j\right] B\boldsymbol{b},$$

whose norm goes to zero for $k \to \infty$ since $\|I - BA\| < 1$ by the assumption made on the approximate inverse B.

We now apply these ideas to RBF interpolation and MLS approximation. In this and the following chapters we will consider three different scenarios:

- Fixed level iteration, *i.e.*, the iterative refinement algorithm is performed on a fixed set of data points $\mathcal{X}$.
- Multilevel iteration, *i.e.*, we work with a nested sequence of data sets $\mathcal{X}_0 \subseteq \mathcal{X}_1 \subseteq \cdots \subseteq \mathcal{X}$.
- Adaptive iteration, *i.e.*, residual iteration is performed on adaptively chosen subsets of $\mathcal{X}$, *e.g.*, by starting with some small subset of $\mathcal{X}$ and then adding one point at a time from the remainder of $\mathcal{X}$ that is determined to be "optimal".

Note that the third approach is similar to the adaptive knot insertion algorithm of Chapter 21.

31.2 Fixed Level Iteration

The simplest setting for a meshfree residual iteration algorithm arises when we fix the data sites $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ throughout the iterative procedure. For this setting the iterative refinement algorithm from linear algebra can be adapted in a straightforward way. We first discuss residual iteration for quasi-interpolants (or approximate MLS approximants) based on (radial) generating functions Ψ_j, $j = 1, \ldots, N$.

If we keep the same set of generating functions for all steps of the iteration then we are performing *non-stationary approximation* and we obtain

Algorithm 31.2. Fixed level residual iteration based on quasi-interpolation

(1) Compute an initial approximation $\mathcal{P}_f^{(0)}$ to the data $\{(\boldsymbol{x}_j, f(\boldsymbol{x}_j)),\ j = 1, \ldots, N\}$ of the form $\mathcal{P}_f^{(0)}(\boldsymbol{x}) = \sum_{j=1}^{N} f(\boldsymbol{x}_j)\Psi_j(\boldsymbol{x})$.

(2) For $k = 1, 2, \ldots$ do

(a) Compute the residuals $r_k(\boldsymbol{x}_j) = f(\boldsymbol{x}_j) - \mathcal{P}_f^{(k-1)}(\boldsymbol{x}_j)$ for all $j = 1, \ldots, N$.

(b) Compute the correction $u(\boldsymbol{x}) = \sum_{j=1}^{N} r_k(\boldsymbol{x}_j)\Psi_j(\boldsymbol{x})$.

(c) Update $\mathcal{P}_f^{(k)}(\boldsymbol{x}) = \mathcal{P}_f^{(k-1)}(\boldsymbol{x}) + u(\boldsymbol{x})$.

As for the iterative refinement algorithm, we can rewrite the last line of the algorithm as

$$\mathcal{P}_f^{(k)}(\boldsymbol{x}) = \mathcal{P}_f^{(k-1)}(\boldsymbol{x}) + \sum_{j=1}^{N} \left[f(\boldsymbol{x}_j) - \mathcal{P}_f^{(k-1)}(\boldsymbol{x}_j) \right] \Psi_j(\boldsymbol{x}).$$

Now we restrict the evaluation of the approximation to the data sites only. Thus, we have

$$\mathcal{P}_f^{(k)}(\boldsymbol{x}_i) = \mathcal{P}_f^{(k-1)}(\boldsymbol{x}_i) + \sum_{j=1}^{N} \left[f(\boldsymbol{x}_j) - \mathcal{P}_f^{(k-1)}(\boldsymbol{x}_j) \right] \Psi_j(\boldsymbol{x}_i), \qquad i = 1, \ldots, N. \tag{31.6}$$

Next we collect all of these N equations into one single matrix-vector equation by introducing the vectors $\boldsymbol{f} = [f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_N)]^T$ and $\Psi = [\Psi_1, \Psi_2, \ldots, \Psi_N]^T$. This allows us to rewrite the initial approximant in matrix-vector form

$$\mathcal{P}_f^{(0)}(\boldsymbol{x}) = \Psi^T(\boldsymbol{x})\boldsymbol{f}. \tag{31.7}$$

Moreover, evaluation of the vector Ψ of generating functions at the data sites $\boldsymbol{x}_i$, $i = 1, \ldots, N$ gives rise to a matrix A with rows $\Psi^T(\boldsymbol{x}_i)$, $i = 1, \ldots, N$. Therefore, (31.6) now becomes

$$\boldsymbol{\mathcal{P}_f}^{(k)} = \boldsymbol{\mathcal{P}_f}^{(k-1)} + A(\boldsymbol{f} - \boldsymbol{\mathcal{P}_f}^{(k-1)}),$$

where we interpret $\mathcal{P}_f{}^{(k)}$ as a vector of values of the approximant at the data sites, *i.e.*, $\mathcal{P}_f{}^{(k)} = [\mathcal{P}_f^{(k)}(\boldsymbol{x}_1), \ldots, \mathcal{P}_f^{(k)}(\boldsymbol{x}_N)]^T$.

Next we follow analogous steps as in our discussion of iterative refinement above. Thus

$$\begin{aligned} \mathcal{P}_f{}^{(k)} &= \mathcal{P}_f{}^{(k-1)} + A(\boldsymbol{f} - \mathcal{P}_f{}^{(k-1)}) \\ &= (I - A)\mathcal{P}_f{}^{(k-1)} + A\boldsymbol{f} \\ &= (I - A)\mathcal{P}_f{}^{(k-1)} + \mathcal{P}_f{}^{(0)}, \end{aligned} \tag{31.8}$$

since (31.7) implies that on the data sites we have $\mathcal{P}_f{}^{(0)} = A\boldsymbol{f}$. Now we can again recursively substitute back in and obtain

$$\mathcal{P}_f{}^{(k)} = \left[\sum_{j=0}^{k}(I - A)^j\right] \mathcal{P}_f{}^{(0)} = \left[\sum_{j=0}^{k}(I - A)^j\right] A\boldsymbol{f}. \tag{31.9}$$

Note that here we have to deal only with the matrix A since the computation of the correction in the algorithm does not require the solution of a linear system.

As before, the sum $\sum_{j=0}^{k}(I - A)^j$ can be seen as a truncated Neumann series expansion for the inverse of the matrix A. If we demand that $\|I - A\| < 1$, then the matrix $\left(\sum_{j=0}^{k}(I - A)^j\right)$ is an approximate inverse of A which converges to A^{-1} since $\|I - A\|^k \to 0$ for $k \to \infty$. More details (such as sufficient conditions under which $\|I - A\| < 1$) are given in [Fasshauer and Zhang (2006)].

In order to establish a connection between iterated (approximate) MLS approximation and RBF interpolation we assume the matrix A to be positive definite and generated by radial basis functions $\Phi_j = \varphi(\|\cdot - \boldsymbol{x}_j\|)$ as in our discussions in earlier chapters. Then A corresponds to an RBF interpolation matrix, and we see that the iterated (approximate) MLS approximation converges to the RBF interpolant provided the same function spaces are used, *i.e.*, $\text{span}\{\Psi_j,\ j = 1, \ldots, N\} = \text{span}\{\Phi_j,\ j = 1, \ldots, N\}$.

In particular, we have established

Theorem 31.1. *Assume $\Psi_1, \ldots, \Psi_N$ are strictly positive definite (radial) generating functions for approximate MLS approximation as discussed in Chapter 26. Then the residual iteration fit of Algorithm 31.2 based on approximate MLS approximation with these generating functions converges to the RBF interpolant based on the same basis functions $\Psi_1, \ldots, \Psi_N$ provided the matrix A with entries $A_{ij} = \Psi_j(\boldsymbol{x}_i)$ satisfies $\|I - A\| < 1$.*

A sufficient condition for A to satisfy $\|I - A\| < 1$ was given in [Fasshauer and Zhang (2006)]. As long as the maximum row sum of A is small enough, *i.e.*,

$$\max_{i=1,2,\ldots,N}\left\{\sum_{j=1}^{N}|A_{i,j}|\right\} < 2,$$

we have convergence of the residual iteration algorithm. This condition is closely related to the Lebesgue function of the RBF interpolant. For example, it is not hard to see that Shepard generating functions satisfy this condition since each row sum is equal to one due to the partition of unity property of the Shepard functions. For other types of functions the condition can be satisfied by an appropriate scaling of the basic function with a sufficiently small shape parameter. However, if ε is taken too small, then the algorithm converges very slowly. A series of experiments analyzing the behavior of the algorithm are presented in [Fasshauer and Zhang (2006)] and also in Section 31.4 below.

The question of whether the approximate MLS generating functions are strictly positive definite has been irrelevant up to this point. However, in order to make the connection between AMLS approximation and RBF interpolation as stated in Theorem 31.1 it is important to find AMLS generating functions that satisfy this additional condition. Of course, any (appropriately normalized) strictly positive definite function can serve as a second-order accurate AMLS generating function. However, it is an open question for which of these functions their higher-order generating functions computed according to our discussion in Chapter 26 are also strictly positive definite.

The family of Laguerre-Gaussians (4.2) provides one example of generating/basis functions that can be used to illustrate Theorem 31.1 (see the numerical experiments below) since their Fourier transforms are positive (see (4.3)).

31.3 Modifications of the Basic Fixed Level Iteration Algorithm

If we start from the interpolation end, then the interpolation conditions $\mathcal{P}_f(\boldsymbol{x}_i) = f(\boldsymbol{x}_i)$ tell us that we need to solve the linear system $A\mathbf{c} = \boldsymbol{f}$ in order to find the coefficients of the RBF expansion

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \Phi_j(\boldsymbol{x}).$$

Following the same iterative procedure as above (*c.f.* (31.4)) this leads to

$$\mathbf{c}_k = \mathbf{c}_{k-1} + B(\boldsymbol{f} - A\mathbf{c}_{k-1}) \tag{31.10}$$

$$= \sum_{j=0}^{k} (I - BA)^j B\boldsymbol{f}, \tag{31.11}$$

where B is an approximate inverse of A as in Section 31.1 and we let $\boldsymbol{c}_0 = B\boldsymbol{f}$. Here $\mathbf{c}_k$ is the k-th step approximation to the coefficient vector $\mathbf{c} = [c_1, \ldots, c_N]^T$.

Equation (31.10) can also be rewritten as

$$\mathbf{c}_k = (I - BA)\mathbf{c}_{k-1} + B\boldsymbol{f},$$

and therefore corresponds to a standard stationary iteration for the solution of linear systems (see, *e.g.*, p. 620 of [Meyer (2000)]). The splitting matrices such that $A = M - N$ are $M = B^{-1}$, $N = B^{-1} - A$, and $H = M^{-1}N = (I - BA)$.

On the other hand, (31.11) gives us an interpretation of the residual iteration as a Krylov subspace method with the Krylov subspaces generated by the matrix $I - BA$ and the vector $B\boldsymbol{f}$.

In the quasi-interpolation formulation the corresponding formulas are given by (31.9), *i.e.*,

$$\mathcal{P}_{\boldsymbol{f}}{}^{(k)} = \left[\sum_{j=0}^{k}(I - A)^j\right] A\boldsymbol{f} \tag{31.12}$$

and can also be interpreted as a Krylov subspace iteration with the Krylov subspaces generated by the matrix $I - A$ and the vector $A\boldsymbol{f}$. Note, however, that in (31.11) we are computing the coefficients of the RBF interpolant, while in (31.12) we are directly computing an approximation to the interpolant.

A natural problem associated with Krylov subspace methods is the determination of coefficients (search directions) α_j such that $\sum_{j=0}^{k}\alpha_j(I - A)^j A\boldsymbol{f}$ converges faster than the generic method with $\alpha_j = 1$ discussed above. Some related work is discussed in the context of the Faul-Powell algorithm in Section 33.2.

We conclude our discussion of modifications of the basic fixed level residual iteration algorithm by noting that the usual stationary approximation method cannot be applied within the fixed level iteration paradigm since we do not have a change in data density that can be used as a guide to re-scale the basis functions. However, it is possible to generalize the non-stationary algorithm to a more general setting in which we change the approximation space from one step to the next. As in the non-stationary setting we can only apply this strategy with approximation methods since an interpolation method will immediately lead to a zero initial residual. For example, one could devise an algorithm in which we use cross-validation at each iteration step to determine the optimal shape parameter (or support size) for the next residual correction. Such an algorithm would also fit into the category of adaptive iterations as discussed below.

31.4 Iterated Approximate MLS Approximation in MATLAB

We now illustrate the fixed level residual iteration algorithm with some MATLAB experiments based on the iteration of approximate MLS approximants with Gaussian generating functions. To obtain some test data we use Franke's function (2.2) on 289 Halton points in the unit square.

In our earlier discussion of approximate MLS approximation we limited ourselves mostly to the case of uniformly spaced data. This was due to the fact, that for non-uniformly spaced data one needs to scale the generating functions individually according to the local variation in the data density in order to maintain the approximate approximation orders stated in Theorem 26.1. Now the convergence result of Theorem 31.1 shows that we no longer need to feel bound by those limitations.

Iteration will automatically improve the approximate MLS fit also on non-uniform data. On the other hand, this observation suggests that the use of a uniform shape parameter for RBF interpolation is most likely not the ideal strategy to obtain highly accurate RBF fits. While a few experiments of RBF interpolation with varying shape parameters exist in the literature (see, *e.g.*, [Kansa and Carlson (1992); Bozzini *et al.* (2002); Fornberg and Zuev (2006)]), the theory for this case is only rudimentary [Bozzini *et al.* (2002)].

The MATLAB code for our examples is provided in Program 31.1. Since we are iterating the approximate MLS approximation we define the scale of the generating functions in terms of the parameter $\mathcal{D}$ (see line 2). However, since the RBF (Gaussian) is defined with the parameter ε we convert $\mathcal{D}$ to ε based on the formula $\varepsilon = 1/(\sqrt{\mathcal{D}}h)$. We approximate h (even for non-uniform Halton points) by $h = 1/(\sqrt{N} - 1)$, where N is the number of data points (in 2D).

In contrast to previous programs we now require two sets of evaluation points. The usual `epoints` that we employ for error computation and plotting along with another set `respoints`, the points at which we evaluate the residuals during the iterative procedure. These points coincide with the data points (see line 13). The iteration on lines 23–28 is equivalent to the formulation in Algorithm 31.2 above.

Program 31.1. `Iterated_MLSApproxApprox2D.m`

```
% Iterated_MLSApproxApprox2D
% Script that performs iterated approximate MLS approximation
% Calls on: DistanceMatrix
 1  rbf = @(e,r) exp(-(e*r).^2);
 2  D = 64/9;  % Parameter for basis function
    % Define Franke's function as testfunction
 3  f1 = @(x,y) 0.75*exp(-((9*x-2).^2+(9*y-2).^2)/4);
 4  f2 = @(x,y) 0.75*exp(-((9*x+1).^2/49+(9*y+1).^2/10));
 5  f3 = @(x,y) 0.5*exp(-((9*x-7).^2+(9*y-3).^2)/4);
 6  f4 = @(x,y) 0.2*exp(-((9*x-4).^2+(9*y-7).^2));
 7  testfunction = @(x,y) f1(x,y)+f2(x,y)+f3(x,y)-f4(x,y);
 8  neval = 40;
 9  N = 289; gridtype = 'h';
    % Convert D to epsilon for use with basis function definition
10  h = 1/(sqrt(N)-1); ep = 1/(sqrt(D)*h);
    % Number of levels for multilevel iteration
11  maxlevel = 10000;
    % Load data points
12  name = sprintf('Data2D_%d%s',N,gridtype); load(name)
13  respoints = dsites; ctrs = dsites;
    % Create neval-by-neval equally spaced evaluation locations
    % in the unit square
```

```
14  grid = linspace(0,1,neval); [xe,ye] = meshgrid(grid);
15  epoints = [xe(:) ye(:)];
    % Compute exact solution
16  exact = testfunction(epoints(:,1),epoints(:,2));
    % Compute evaluation matrix directly based on the distances
    % between the evaluation points and centers
17  DM = DistanceMatrix(epoints,ctrs);
18  EM = rbf(ep,DM)/(pi*D);
    % Compute - for all levels - evaluation matrices for
    % residuals directly based on the distances between the
    % next finer points (respoints) and centers
19  DM = DistanceMatrix(respoints,ctrs);
20  RM = rbf(ep,DM)/(pi*D);
21  Pf = zeros(neval^2,1);  %  initialize
    % Create vector of function values (initial residual),
22  rhs = testfunction(dsites(:,1),dsites(:,2));
23  for level=1:maxlevel
       % Evaluate on evaluation points
       % (for error computation and plotting)
24     Pf = Pf + EM*rhs;
       % Compute new residual
25     rhs = rhs - RM*rhs;
       % Compute errors on evaluation grid
26     maxerr(level) = norm(Pf-exact,inf);
27     rms_err(level) = norm(Pf-exact)/neval;
28  end
29  figure; semilogy(1:maxlevel,maxerr,'b',1:maxlevel,rms_err,'r');
```

According to the experiments shown in Figure 2.5 the optimal shape parameter for Gaussian interpolation to Franke's function on 289 Halton points is close to $\varepsilon = 6$. The corresponding value of $\mathcal{D}$ for the Gaussian as an approximate MLS generating function is $\mathcal{D} = 64/9$ (since $\varepsilon = 1/(\sqrt{\mathcal{D}}h)$, and we approximate h for the non-uniform Halton points by the value we would have for uniform points, *i.e.*, $h = 1/(\sqrt{N} - 1)$. For this value of the shape parameter we see the convergence behavior of the approximate MLS residual iteration in the left plot of Figure 31.1. The final maximum error after 10000 iterations is 9.921772e-002 (top/solid curve), and the RMS error is 3.939342e-003 (lower/dashed curve). For comparison, the errors for the corresponding RBF interpolant (which is the theoretical limit of the residual iteration) are 3.238735e-002 for the maximum error and 1.074443e-003 for the RMS error. These errors are included as horizontal straight lines in the left plot of Figure 31.1.

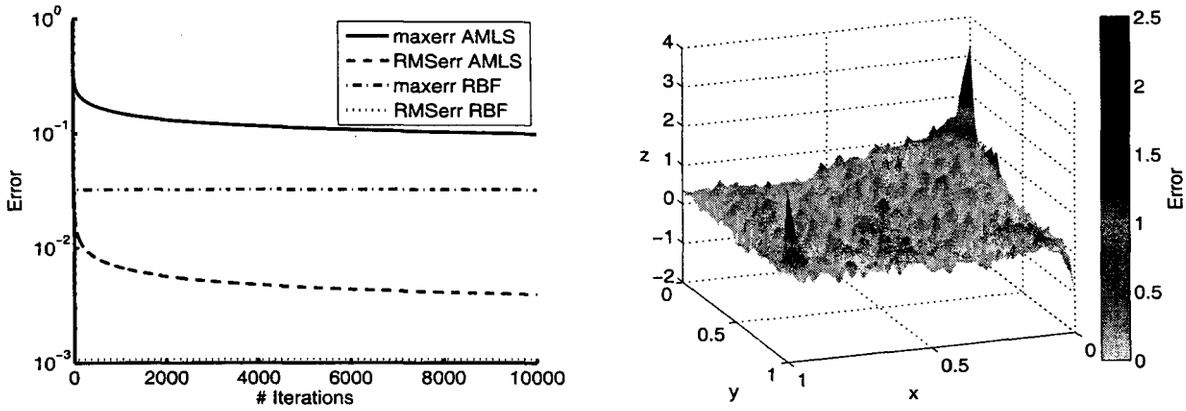


Fig. 31.1 Convergence for iterated MLS approximation based on Gaussian generating functions with $\mathcal{D} = 64/9$ ($\varepsilon = 6$) to data sampled from Franke's function at 289 Halton points (left), and fit for an RBF interpolant based on Gaussians with $\varepsilon = 1$ (right).

An advantage of the residual iteration is that it allows us to compute radial basis approximations also for values of the shape parameter for which the interpolation matrix is very ill-conditioned. For example, the right plot of Figure 31.1 shows an RBF interpolant to the 289 Halton samples of Franke's function based on Gaussians with shape parameter $\varepsilon = 1$. The reciprocal condition number estimate provided by MATLAB for the interpolation matrix for this problem is `RCOND = 2.132739e-020`.

In these ill-conditioned cases convergence of the residual iteration to the limit is rather slow, but the approximations can be computed very stably. In the left plot of Figure 31.2 we show the convergence behavior for the residual iteration with approximate MLS approximants based on Gaussian generating functions with $\mathcal{D} = 256$ (corresponding to $\varepsilon = 1$). Note that the approximation errors for the iterative scheme quickly drop below the maximum error of 2.507017e+000 and RMS error of 2.186992e-001 of the mostly meaningless interpolant. The corresponding fit for the iterative method is displayed in the right plot of the figure. While this fit is not very accurate, it is still much more reliable than the fit consisting of mostly numerical noise shown in the right plot of Figure 31.1. The final errors after 10000 iterations are 2.933448e-001 for the maximum error and 8.470775e-002 for the RMS error.

It is of interest to note that the residual iteration shows the most dramatic error improvement during the first few iterations. Thus, only a few iterations of approximate MLS approximation are required to obtain a reasonable (and stably computable) approximation to the RBF interpolant. Moreover, we emphasize again that while our discussion of approximate MLS approximation was mostly limited to the case of uniform data (at least for most practical purposes), this limitation no longer exists for the residual iteration algorithm.

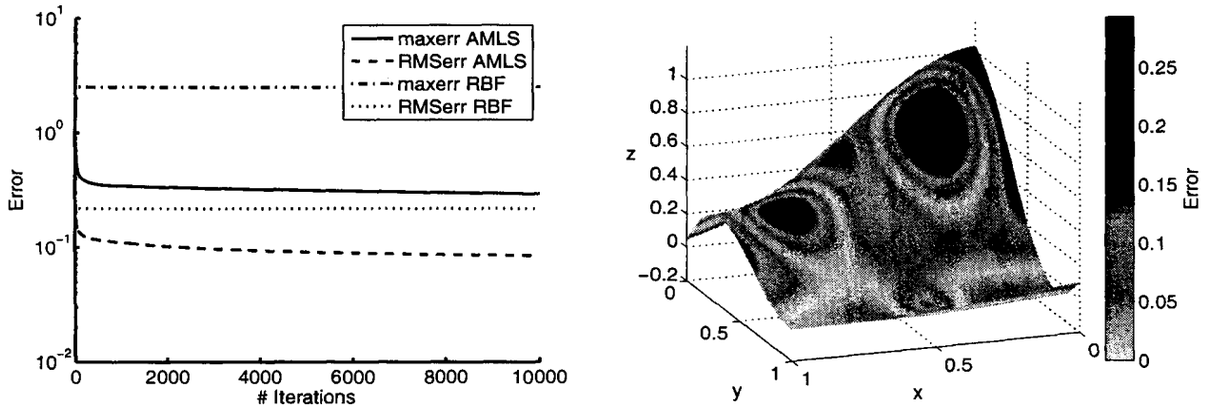


Fig. 31.2 Convergence and final fit for iterated approximate MLS approximation based on 289 Halton points and Gaussians with $\mathcal{D} = 256$ ($\varepsilon = 1$).

31.5 Iterated Shepard Approximation

The use of other approximation methods within the fixed level residual iteration algorithm such as regular MLS approximation or RBF least squares approximation is also possible. We point out, however, that the use of RBF interpolation does not make any sense in the context of fixed level residual iteration since the residuals on the data are automatically zero if we perform interpolation of the data.

If we want to use regular MLS approximation instead of approximate MLS approximation in the residual iteration, then, for Shepard's method, this means replacing line 18 in Program 31.1 by

```
18a EM = rbf(ep,DM);
18b EM = EM./repmat(EM*ones(N,1),1,N);     % Shepard normalization
```

and line 20 by

```
20a RM = rbf(ep,DM);
20b RM = RM./repmat(RM*ones(N,1),1,N);     % Shepard normalization
```

Note that we can now no longer claim that the limit of the iterated Shepard approximant is given by the RBF interpolant based on the Shepard weights as basis functions. In fact, the iterated Shepard approximant becomes more accurate than the RBF interpolant. For example, if we take Gaussian weight functions with $\varepsilon = 6$ or $\varepsilon = 1$ as above, then the corresponding convergence behavior for the iterated Shepard approximant is displayed in Figure 31.3. Moreover, for the example with $\varepsilon = 6$, 45 iterations of the Shepard approximant result in a smaller maximum error than the RBF interpolant, while 3515 iterations are required to push the RMS error for the Shepard iteration below 1.074443e-003. After 10000 iterations with

Shepard approximants the maximum error is 8.760946e-003 and the RMS error is 7.889609e-004. For the $\varepsilon = 1$ example the final errors (after 10000 iterations) are 2.664303e-001 for the maximum error and 8.169080e-002 for the RMS error. This example is very similar to the AMLS example displayed in Figure 31.2. Again, the most significant part of the error improvement occurs during the first 10–20 iterations.

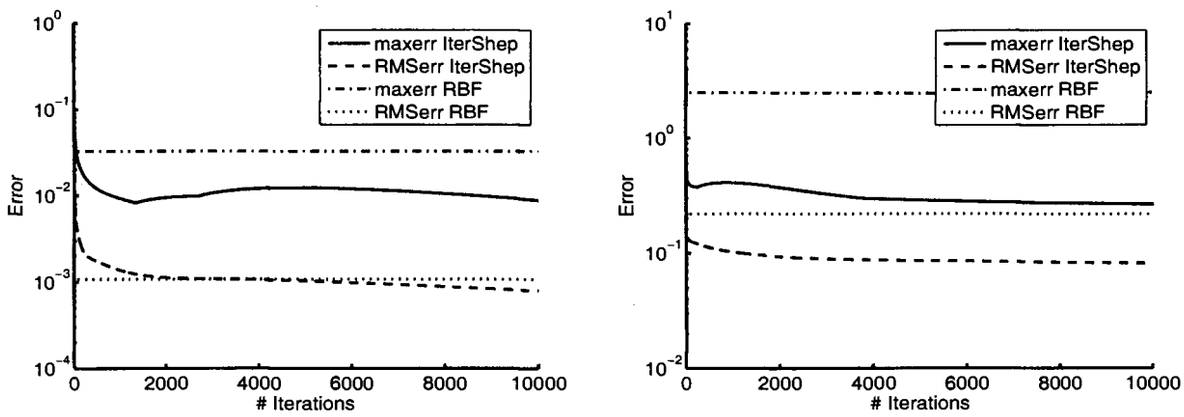


Fig. 31.3 Convergence for iterated Shepard approximation based on 289 Halton points and Gaussian weights with $\varepsilon = 6$ (left), and $\varepsilon = 1$ (right).

For other data sets and other values of ε residual iteration may converge faster. For example, in Figure 31.4 we used 1089 data points (again taken from Franke's function) and a value of $\varepsilon = 16$. On the left we show the convergence behavior for iterated approximate MLS with Gaussian generating functions, and on the right for iterated Shepard approximation with Gaussian weights. Both graphs contain the errors for RBF interpolation with Gaussian basis functions for comparison. We note that the approximate MLS iteration approaches the RBF interpolant faster than in the previous examples. Moreover, both errors for iterated Shepard approximation are smaller than those for the interpolant after only three iterations. In fact, after 10 iterations the maximum error for the iterated Shepard approximation is one order of magnitude smaller than that for the RBF interpolant.

A detailed study of the dependence of the convergence of the fixed-level residual iteration algorithm on the shape parameter ε and more examples are provided in [Fasshauer and Zhang (2006)].

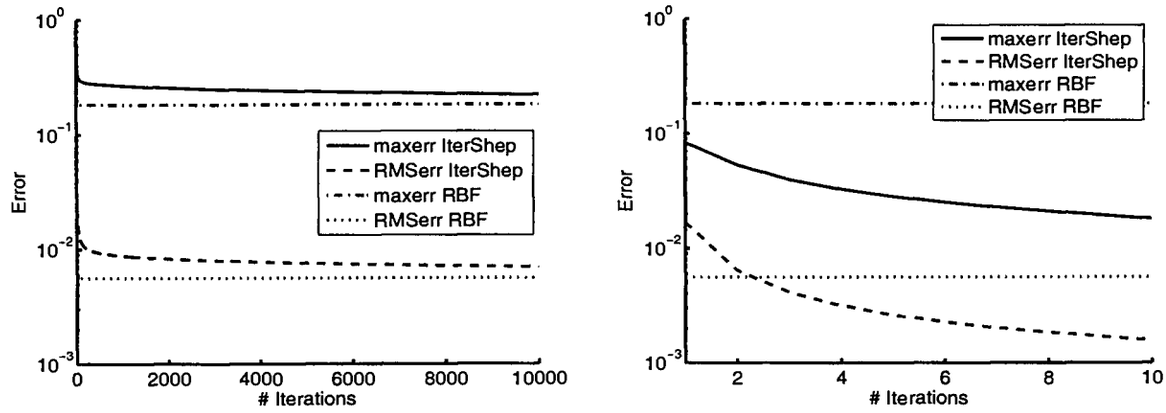


Fig. 31.4 Convergence for iterated approximate MLS approximation (left) and Shepard approximation (right) based on 1089 Halton points and Gaussian weights with $\varepsilon = 16$.

Chapter 32

Multilevel Iteration

As we saw in Chapters 12 and 16 that there is a trade-off principle for interpolation with compactly supported radial functions. Namely, using a non-stationary approach we can obtain good approximation results at the cost of increasing computational complexity, while with a stationary approach one has an efficient approximation method. However, stationary approximation with compactly supported radial functions is saturated, *i.e.*, it provides only limited convergence.

32.1 Stationary Multilevel Interpolation

In order to combine the advantages of both approaches for interpolation with compactly supported radial functions described above, Schaback suggested the use of a *multilevel* stationary scheme. This scheme was implemented first in [Floater and Iske (1996b)] and later studied by a number of other researchers (see, *e.g.*, [Chen *et al.* (2002); Fasshauer and Jerome (1999); Hales and Levesley (2002); Hartmann (1998); Iske (2001); Narcowich *et al.* (1999); Wendland (1999a)]).

In contrast to the fixed level iteration of the previous chapter we will now use a nested sequence $\mathcal{X}_1 \subset \cdots \subset \mathcal{X}_K = \mathcal{X} \subset \mathbb{R}^s$ of point sets with increasingly greater data density, *i.e.*, smaller fill distance $h_{\mathcal{X},\Omega}$. The basic idea of the stationary multilevel interpolation algorithm is to scale the size of the support of the basis functions with the fill distance, but to interpolate to residuals on progressively refined sets of centers. This method has all of the combined benefits of the interpolation methods for compactly supported RBFs referred to earlier: it is computationally efficient (can be performed in $\mathcal{O}(N)$ operations), well-conditioned, and appears to be convergent.

An algorithm for multilevel interpolation is as follows:

Algorithm 32.1. Stationary multilevel interpolation

(1) Create nested point sets $\mathcal{X}_1 \subset \cdots \subset \mathcal{X}_K = \mathcal{X} \subset \mathbb{R}^s$, and initialize $\mathcal{P}_f(\boldsymbol{x}) = 0$.
(2) For $k = 1, 2, \ldots, K$ do
  (a) Solve $u(\boldsymbol{x}) = f(\boldsymbol{x}) - \mathcal{P}_f(\boldsymbol{x})$ on $\mathcal{X}_k$.

(b) Update $\mathcal{P}_f(\boldsymbol{x}) \leftarrow \mathcal{P}_f(\boldsymbol{x}) + u(\boldsymbol{x})$.

The representation of the update u at step k is a radial basis function expansion of the form

$$u(\boldsymbol{x}) = \sum_{\boldsymbol{x}_j \in \mathcal{X}_k} c_j^{(k)} \varphi\left(\frac{\|\boldsymbol{x} - \boldsymbol{x}_j\|}{\rho_k}\right)$$

with φ a (compactly supported) basic function and its support scale $\rho_k \simeq h_{\mathcal{X}_k,\Omega}$. This requires the solution of a linear system whose size is determined by the number of points in $\mathcal{X}_k$.

Unfortunately, so far there are only limited theoretical results concerning the convergence of this multilevel algorithm. In [Narcowich *et al.* (1999)] the authors show that a related algorithm (in which additional boundary conditions are imposed) converges at least linearly. Hartmann analyzes the multilevel algorithm in his Ph.D. thesis [Hartmann (1998)]. He shows at least linear convergence for multilevel interpolation on a regular lattice for various radial basis functions. Similar results are obtained in [Hales and Levesley (2002)] for (globally supported) polyharmonic splines, *i.e.*, thin plate splines and radial powers. In this context linear convergence is to be interpreted as an improvement of the form

$$\|f - \mathcal{P}_f^{(k)}\| \le C\|f - \mathcal{P}_f^{(k-1)}\|, \qquad k = 1, \ldots, K,$$

where $\mathcal{P}_f^{(k)}$ denotes the interpolant at level k, and $C < 1$ is a positive (level-independent) constant.

The main difficulty in proving the convergence of the multilevel algorithm is the fact that the approximation space changes from one level to the next. The approximation spaces are not nested (as they usually are for wavelets). This means that the native space norm changes from one level to the next. Hales and Levesley avoid this problem by scaling the (uniformly spaced) data instead of the basis functions. Then the fact that polyharmonic splines are in a certain sense homogeneous (see Section 34.4) simplifies the analysis. This fact was also used in [Wendland (2005a)] to prove linear convergence for multilevel (scattered data) interpolation based on thin plate splines.

Another approach to multilevel interpolation was recently suggested by Opfer (see [Opfer (2004); Opfer (2006)]). He constructs so-called *multiscale kernels* that have the information from different resolution levels built into a single function. These kernels are built as tensor products of scaling functions that form a wavelet-like multiresolution analysis. Opfer provides error bounds for interpolation with these kernels analogous to those in Theorem 15.3. He also demonstrates how multiscale kernels can be used for scattered data interpolation, and for image decomposition and compression.

32.2 A MATLAB Implementation of Stationary Multilevel Interpolation

The MATLAB program `ML_CSRBF3D.m` for stationary multilevel interpolation with compactly supported RBFs is displayed as Program 32.1. In this program we use the compactly supported RBF $\varphi(r) = (1-r)_+^6(35r^2+18r+3)$ of Wendland. This function is C^4 and strictly positive definite and radial on $\mathbb{R}^3$. The program shows a number of differences compared to our previous codes. On line 2 we define the maximum number of levels K we wish to use, and then define the corresponding scale parameters ε for the basic function. Since we load data sets consisting of $(2^k+1)^3$ uniformly spaced data points in the unit cube (where k runs from 1 to K, see lines 13–17), the fill distance $h_{\mathcal{X}_k,\Omega}$ changes by a factor of two from one level to the next. Therefore, in order to guarantee a stationary interpolation scheme, the scale parameter ε needs to change by a factor of two from one level to the next, also. This is achieved in line 2 of the code where we also use an additional factor of 0.7 to uniformly scale all values of ε. Note that here ε corresponds to a support radius $\rho = 1/\varepsilon$.

In each iteration we solve one interpolation problem (see line 18–22). The expansion coefficients `coef` are stored in one component of a MATLAB cell array. Similarly, we need to keep the centers for all levels in memory (again using a cell array `ctrs`, see line 17). The right-hand side for the interpolation problem is given by the data (values of the test function on the initial data set) in the first iteration, and in later iterations by the residual, *i.e.*, the difference between the data (values of the test function on the present grid) and the values of the fit on the present grid. These values are computed at the end of the previous iteration and stored in the vector `Rf` (see lines 28–32). This extra evaluation (in addition to the evaluation on a separate evaluation grid for error monitoring and plotting purposes, see lines 34–39) adds to the complexity (and inefficiency) of the present code. Moreover, the evaluation of the residual on the present grid requires us to keep evaluation matrices for all levels in memory (in the cell array `RM`, see lines 24–27). Note that the evaluation points (`respoints`) for the residuals change with every iteration and need not be kept in storage. An alternative approach would be to evaluate all residuals on a common (fine) grid, *e.g.*, the evaluation grid. This, however, would make (the evaluation of) the initial iterates rather expensive.

Finally, on lines 42–48 we keep track of the RMS error, and compute the rate of convergence from one level to the next. The commands that generate plots of the data sites, interpolant and error are given on lines 49–54.

Program 32.1. `ML_CSRBF3D.m`

```
% ML_CSRBF3D
% Script that performs multilevel RBF Interpolation using
% sparse matrices
```

```
% Calls on: DistancMatrixCSRBF
    % Wendland C4
 1  rbf = @(e,r) r.^6.*(35*r.^2-88*r+56*spones(r));
    % Number of levels with epsilons for stationary interpolation
 2  K = 4; ep = 0.7*2.^[0:K-1];
 3  testfunction = @(x,y,z) 64*x.*(1-x).*y.*(1-y).*z.*(1-z);
 4  gridtype = 'u'; % Type of data points: 'u'=uniform
 5  neval = 10; M = neval^3;
 6  grid=linspace(0,1,neval); [xe,ye,ze]=meshgrid(grid);
 7  epoints=[xe(:) ye(:) ze(:)];
 8  exact = testfunction(epoints(:,1),epoints(:,2),epoints(:,3));
 9  isomin = 0.1; isomax = 1; isostep = .1;
10  xslice = .25:.25:1; yslice = 1; zslice = [0,0.5];
11  Rf_old = zeros(27,1);    % initialize
12  for k=1:K
13     N1 = (2^k+1)^3;    N2 = (2^(k+1)+1)^3;
14     name1 = sprintf('Data3D_%d%s',N1,gridtype);
15     name2 = sprintf('Data3D_%d%s',N2,gridtype);
16     load(name2);     respoints = dsites;
17     load(name1);     ctrs{k} = dsites;
      % Compute right-hand side (= residual)
18     Tf = testfunction(dsites(:,1),dsites(:,2),dsites(:,3));
19     rhs = Tf - Rf_old;
20     DM_data = DistanceMatrixCSRBF(dsites,ctrs{k},ep(k));
21     IM = rbf(ep(k),DM_data);
       % Compute coefficients for RBF interpolant to detail level
22     coef{k} = IM\rhs;
23     if (k < K)
          % Compute - for all levels - evaluation matrices for
          % residuals directly
24        for j=1:k
25           DM_res = DistanceMatrixCSRBF(respoints,ctrs{j},ep(j));
26           RM{j} = rbf(ep(j),DM_res);
27        end
          % Evaluate RBF interpolant (sum of all previous fits
          % evaluated on current grid)
28        Rf = zeros(N2,1);
29        for j=1:k
30           Rf = Rf + RM{j}*coef{j};
31        end
32        Rf_old = Rf;
33     end
```

```
34      DM_eval = DistanceMatrixCSRBF(epoints,ctrs{k},ep(k));
35      EM = rbf(ep(k),DM_eval);
36      Pf = EM*coef{k};
37      if (k > 1)
38         Pf = Pf_old + Pf;
39      end
40      Pf_old = Pf;
41      maxerr = norm(Pf-exact,inf);
42      rms_err = norm(Pf-exact)/sqrt(M);
43      fprintf('RMS error:     %e\n', rms_err)
44      if (k > 1)
45         rms_rate = log(rms_err_old/rms_err)/log(2);
46         fprintf('RMS rate:      %f\n', rms_rate)
47      end
48      rms_err_old = rms_err;
49      figure
50      plot3(dsites(:,1),dsites(:,2),dsites(:,3),'bo');
51a     Plotlsosurf(xe,ye,ze,Pf,neval,exact,maxerr,isomin,...
51b         isostep,isomax);
52      PlotSlices(xe,ye,ze,Pf,neval,xslice,yslice,zslice);
53a     PlotErrorSlices(xe,ye,ze,Pf,exact,neval,...
53b         xslice,yslice,zslice);
54  end
```

In Figure 32.1 we display four data sets used in the 3D multilevel experiment. The corresponding 3D multilevel interpolants are shown as iso-surfaces in Figure 32.2. Note that the test function $f(x,y,z) = 64x(1-x)y(1-y)z(1-z)$ is a three-dimensional "bump" function, and therefore only the outermost iso-surface (corresponding to the function value 0.1) is visible. Therefore, we also display three-dimensional slice plots of the absolute error in Figure 32.3. Both the iso-surfaces and the slice plots are color coded according to the absolute error.

In Table 32.1 we list the corresponding RMS errors and observed convergence rates for the 3D multilevel experiment.

Table 32.1 3D stationary multilevel interpolation with $\varphi(r) = (1-r)_+^6(35r^2+18r+3)$.

mesh	RMS-error	rate	% nonzero	time
$3 \times 3 \times 3$	1.005315e-001		92.32	0.16
$5 \times 5 \times 5$	2.764907e-002	1.8623	36.99	0.56
$9 \times 9 \times 9$	2.626864e-003	3.3958	8.88	13.45
$17 \times 17 \times 17$	5.706061e-004	2.2028	1.57	73.50

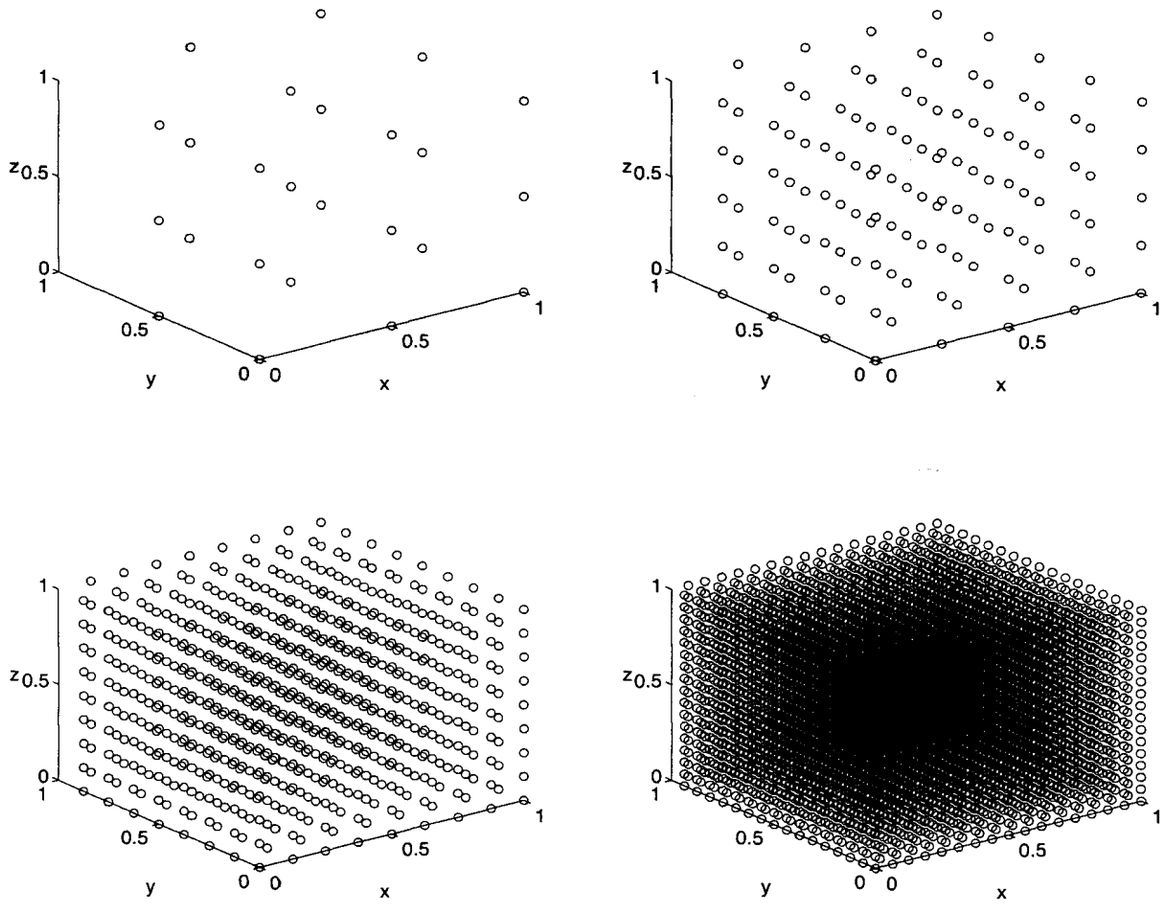


Fig. 32.1 Uniform point sets in the unit cube with 27, 125, 729, and 4913 points (top left to bottom right).

Next we present a two-dimensional multilevel interpolation experiment with scattered data. The data were obtained in the ASCII VRML format from `http://amba.charite.de/~ksch/spsm/beet1s.wrl.gz`. The data set provides a digitized surface model of a bust of the famous German composer Ludwig van Beethoven. The original data set consists of 2663 points in the unit square. In order to provide a nested set of data sites for the multilevel algorithm we use the program `Thin.m` provided in Appendix C. The processed data files are included on the enclosed CD. The resulting point sets are displayed in Figure 32.4. A much more detailed discussion of thinning algorithms for scattered data is presented in [Iske (2004)]. The MATLAB program used to generate the multilevel interpolants in Figure 32.5 is essentially the same as Program 32.1. The main difference is that the data values are also read from the data file instead of being generated by a test function. For rendering purposes the interpolants are evaluated on an 80×80 grid of equally spaced points in the unit square.

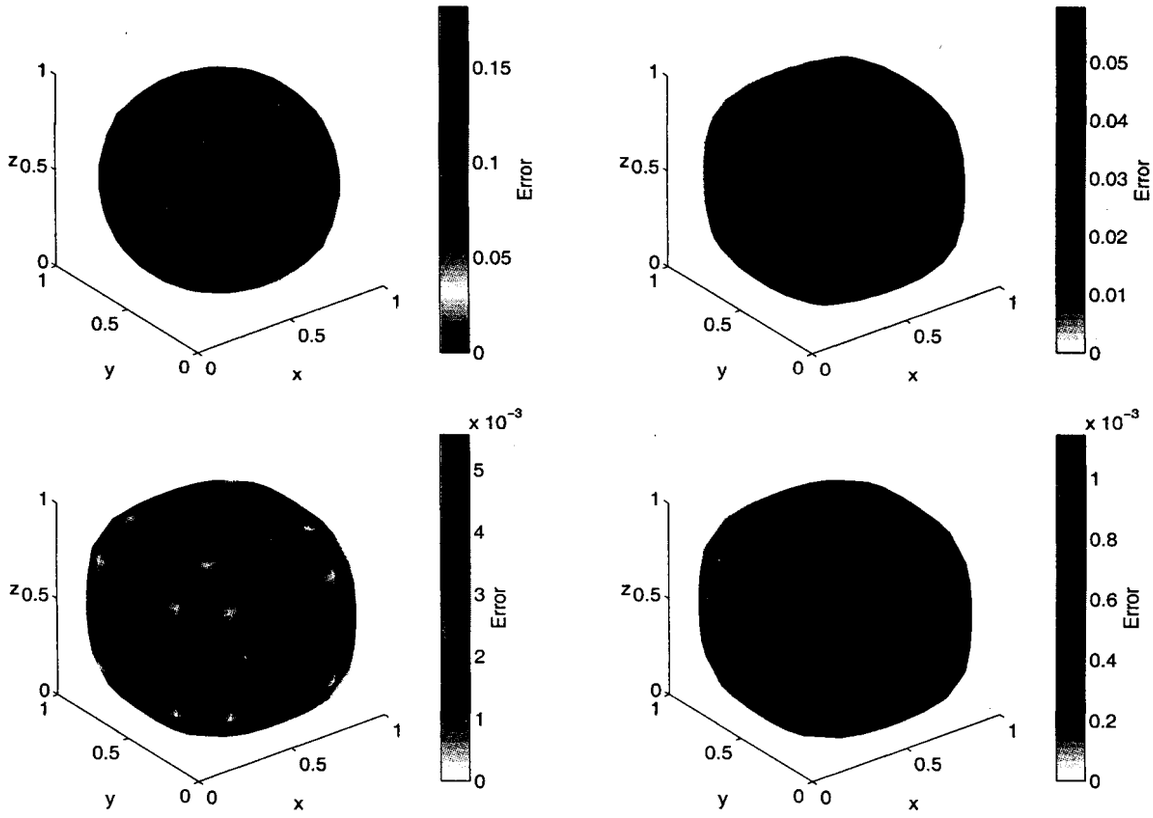


Fig. 32.2 Iso-surface plots for multilevel interpolants at levels 1, 2, 3 and 4 (top left to bottom right) false -colored by absolute error .

32.3 Stationary Multilevel Approximation

The same basic multilevel algorithm can also be used for other approximation methods. In [Fasshauer (2002c)] the idea was applied to moving least squares methods and approximate moving least squares methods. Experiments similar to those of [Fasshauer (2002c)] are now repeated here. In order to be able to provide a comparison between the multilevel interpolation and approximation algorithms we begin with one more example for multilevel interpolation.

We obtain the data for the following numerical examples by sampling a mollified Franke function f at uniformly spaced points in the unit square $[0,1]^2$, *i.e.*,

$$f(x,y) = 15 \exp\left(\frac{-1}{1-(2x-1)^2}\right) \exp\left(\frac{-1}{1-(2y-1)^2}\right) F(x,y),$$

where F denotes Franke's function (2.2).

In Table 32.2 we list the benchmark results for multilevel RBF interpolation with the compactly supported function $\varphi_{3,1}(r) = (1-r)_+^4 (4r+1)$. We again use an initial scale factor of 0.7 for the shape parameter ε. Since the shape parameter ε is equal to the reciprocal of the support scale ρ this means that the initial support scale ρ_1 is chosen so that the (univariate) basic function is fairly wide. Subsequent

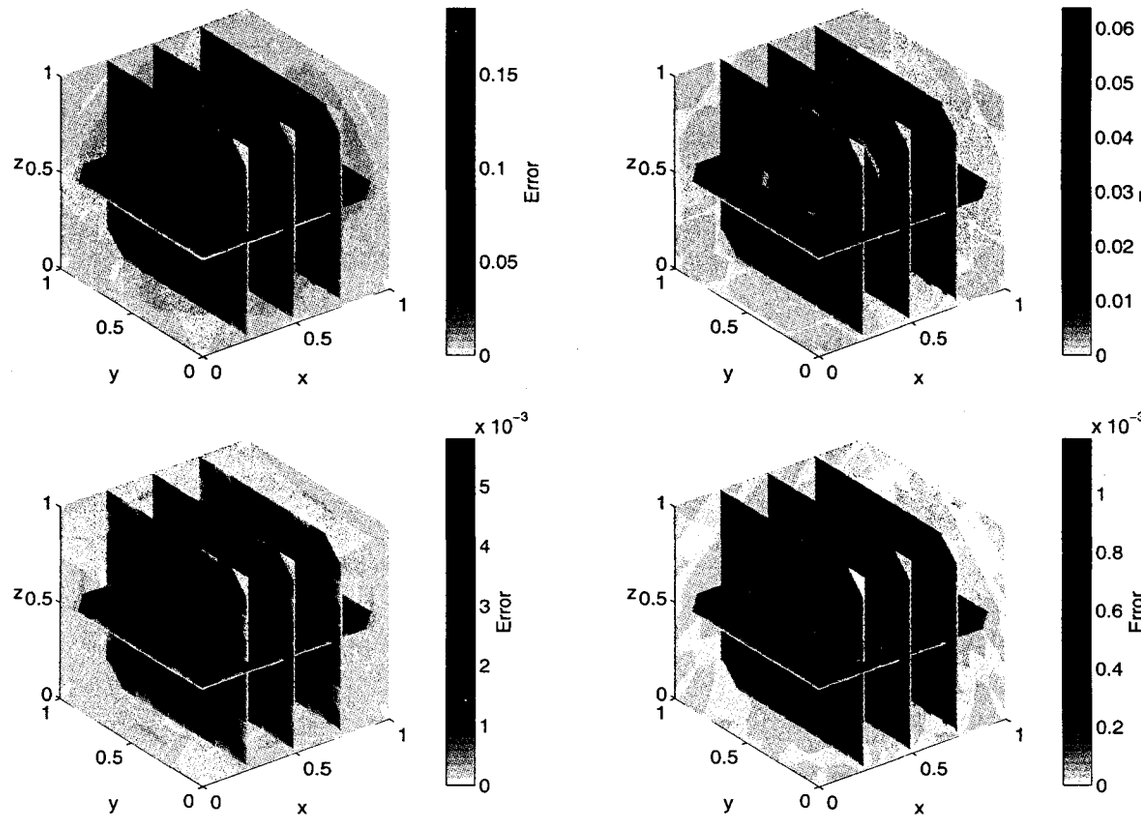


Fig. 32.3 Slice plots of absolute errors for multilevel interpolants at levels 1, 2, 3 and 4 (top left to bottom right).

support scales are successively divided by two (*i.e.*, ε is multiplied by two) — just as the fill distance is halved on successive computational grids $\mathcal{X}_k$.

Table 32.2 2D stationary multilevel interpolation with $\varphi(r) = (1-r)_+^4(4r+1)$ at equally spaced points in $[0,1]^2$.

mesh	RMS-error	rate	% nonzero	time
3×3	2.498505e-001		100	0.13
5×5	7.695304e-002	1.6990	57.76	0.16
9×9	2.092849e-002	1.8785	23.18	0.20
17×17	1.145664e-003	4.1912	7.47	0.42
33×33	1.376035e-004	3.0576	2.13	1.64
65×65	3.303559e-005	2.0584	0.57	7.98
129×129	2.149123e-006	3.9422	0.15	5.98

In Table 32.2 we list the RMS errors computed on a 40×40 uniform evaluation grid along with the percentage of non-zero entries in the RBF interpolation matrix and the computer time required for each iteration. The first four multilevel interpolants are shown in Figure 32.6.

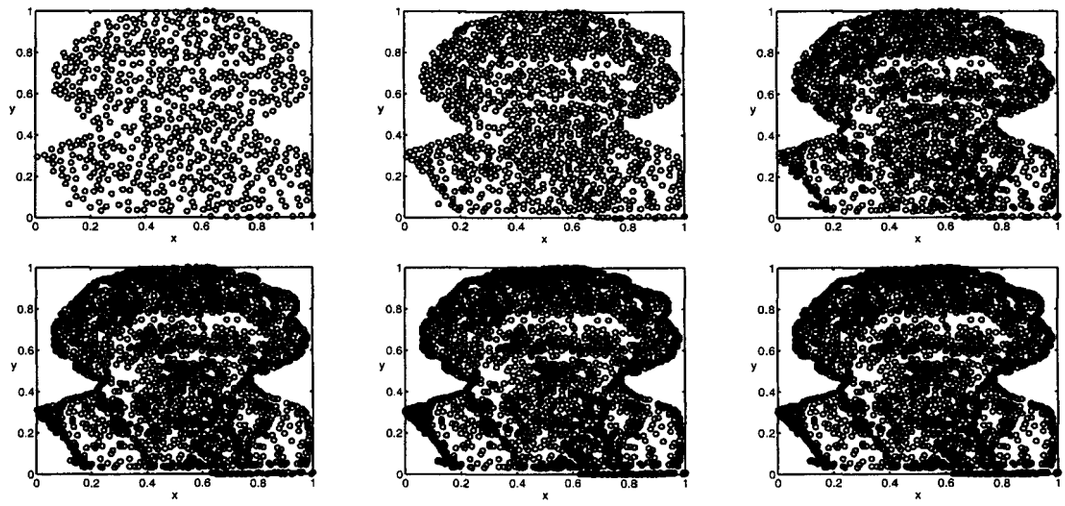


Fig. 32.4 Thinned data sets for Beethoven's head. For top left to bottom right: 163, 663, 1163, 1663, 2163, and 2663 points.

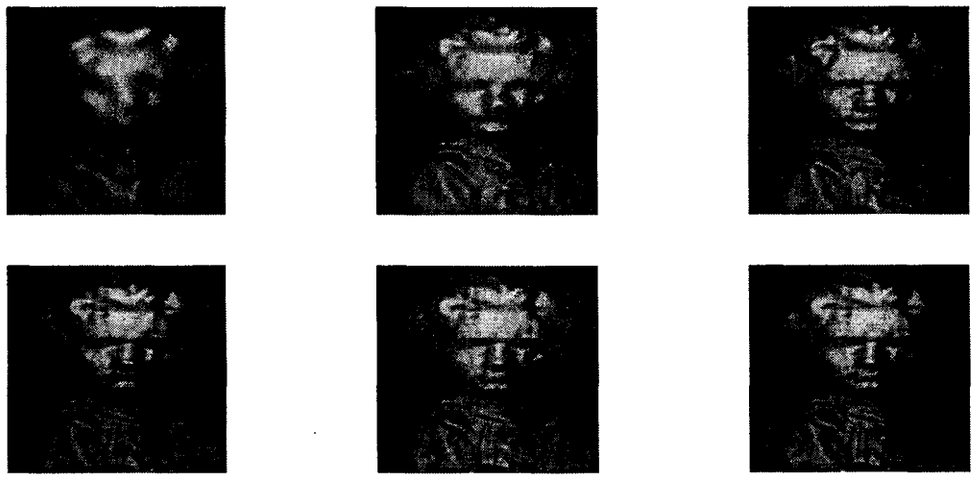


Fig. 32.5 Multilevel interpolants to Beethoven data. For top left to bottom right: 163, 663, 1163, 1663, 2163, and 2663 points.

Next we replace RBF interpolation at each step of the multilevel residual iteration algorithm by standard moving least squares approximation. Table 32.3 illustrates the performance of the multilevel algorithm for Shepard's method and a moving least squares approximation with linear precision, both based on the compactly supported weight function $\varphi_{3,1}(r) = (1-r)_+^4(4r+1)$. The support scaling is the same as in the previous multilevel interpolation example. The MATLAB code for these examples is omitted as it is very similar to that of Program 32.1. We note, however, that our implementation of the linear precision variant based on Program 24.3 is rather inefficient when compared to the other multilevel examples presented here.

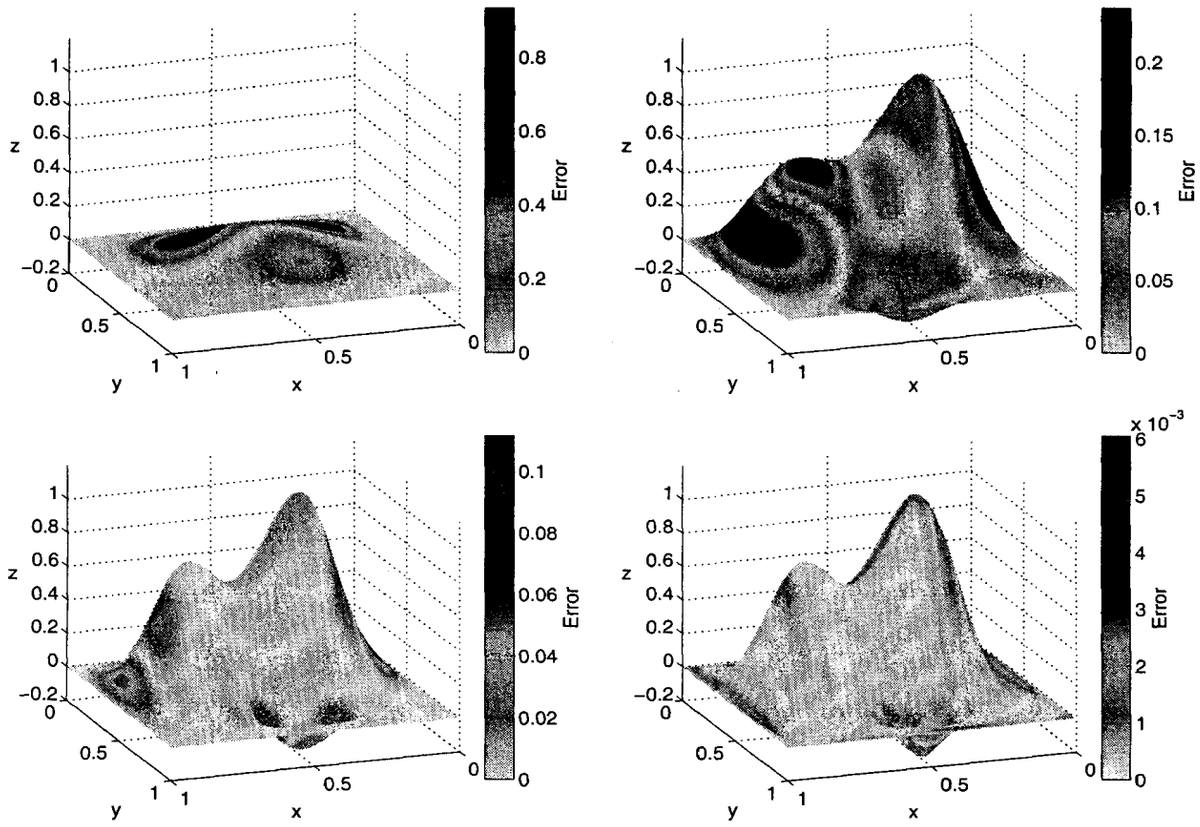


Fig. 32.6 The first four interpolants from Table 32.2.

There seems to be no theoretical investigation of the convergence properties of the multilevel algorithm for moving least squares approximation in the literature.

Table 32.3 2D multilevel MLS approximation with $\varphi(r) = (1-r)_+^4(4r+1)$.

	Shepard			linear precision		
mesh	RMS-error	rate	time	RMS-error	rate	time
3×3	2.776569e-001		0.16	2.812184e-001		0.91
5×5	1.615753e-001	0.7811	0.14	1.481365e-001	0.9248	0.97
9×9	7.519432e-002	1.1035	0.19	7.015497e-002	1.0783	1.36
17×17	1.858696e-002	2.0163	0.39	1.932368e-002	1.8602	3.09
33×33	3.581720e-003	2.3756	1.56	2.639418e-003	2.8721	11.45
65×65	5.458943e-004	2.7140	7.67	3.426412e-004	2.9454	89.64
129×129	1.047351e-004	2.3819	0.36	3.936786e-005	3.1216	1.17

As a third part of this example we use approximate MLS approximation at each level of the residual iteration algorithm. We use the generating functions $\Psi(r) = \frac{7}{\pi}(1-r)_+^4(4r+1)$ (giving rise to an approximate partition of unity) and $\Psi(r) = \frac{252}{229\pi}(1-r)_+^4(4r+1)(14-55r^2)$ (giving rise to an approximate partition of

unity with one vanishing moment). Recall that we constructed these functions in Example 26.2 starting with the initial weight function $\psi_0(y) = (1-\sqrt{y})_+^4 (4\sqrt{y}+1)$. As scale parameter we take $\mathcal{D} = 400/49$. This corresponds to the same scaling as in the other examples since $\varepsilon = 1/(\sqrt{\mathcal{D}}h)$ and the initial fill distance for the 3×3 grid is $h = 1/2$.

Table 32.4 Multilevel approximate MLS approximation with basic function $\psi_0(y) = (1-\sqrt{y})_+^4(4\sqrt{y}+1)$ and $\mathcal{D} = 400/49$.

	basic method			1 vanishing moment		
mesh	RMS-error	rate	time	RMS-error	rate	time
3×3	2.803247e-001		0.13	2.630763e-001		0.13
5×5	1.578062e-001	0.8289	0.16	9.133669e-002	1.5262	0.16
9×9	7.483597e-002	1.0764	0.20	2.783120e-002	1.7145	0.22
17×17	1.784522e-002	2.0682	0.39	3.399671e-003	3.0332	0.45
33×33	2.468958e-003	2.8536	1.52	4.359882e-004	2.9630	1.81
65×65	3.637815e-004	2.7628	7.30	7.856778e-005	2.4723	9.05
129×129	5.636161e-005	2.6903	0.39	2.460906e-005	1.6747	0.38

If we compare the numbers from the three different approaches listed in Tables 32.2–32.4 we see that none of the approximation methods yield more accurate results than the interpolation method. It is surprising, however, that the approximate MLS methods perform better than the regular MLS methods. For noisy data the approximate MLS method would be preferable.

32.4 Multilevel Interpolation with Globally Supported RBFs

So far we have concentrated on the use of compactly supported functions within the multilevel residual iteration algorithm. For globally supported functions we learned in earlier chapters that we can obtain good approximation order estimates in the non-stationary setting. One reason for our focus on compactly supported functions in this chapter is that if we consider the use of globally supported functions in a non-stationary multilevel interpolation framework, then we see that nothing is gained by the multilevel approach (provided we limit ourselves to the solution of linear problems such as the interpolation problems discussed above).

More precisely, if the meshes $\mathcal{X}_k$ at the different levels are nested and the shape parameter ε in the globally supported functions is kept fixed through all levels k, then the function spaces $\mathcal{S}_k = \operatorname{span}\{\varphi(\|\cdot - \boldsymbol{x}_j^{(k)}\|) : \boldsymbol{x}_j^{(k)} \in \mathcal{X}_k\}$ are also nested. Consequently, the richest space $\mathcal{S} = \bigcup_{k=1}^K \mathcal{S}_k$, which is used when all updates have been performed at the finest level, is equal to the space $\mathcal{S}_K$ on the finest mesh $\mathcal{X}_K$. Thus, a direct fit at the finest level K uses the same approximation space, and will therefore yield the same quality of fit, as the multilevel algorithm using all of the meshes $\mathcal{X}_k$, $k = 1, \ldots, K$. However, the multilevel algorithm requires all the

additional (unnecessary) intermediate work on the coarser meshes.

It therefore follows

Theorem 32.1 (Rule 1). *Consider a linear problem of the form $Lu = f$ on $\Omega \subseteq \mathbb{R}^s$, let $\mathcal{X}_1, \ldots, \mathcal{X}_K$ be a nested sequence of point sets in Ω, and let φ be a globally supported RBF with fixed shape parameter ε for all $k = 1, \ldots, K$. Then the approximate solution u_K obtained by the multilevel interpolation algorithm is the same as the solution of the problem $Lu = f$ on the fixed level $\mathcal{X}_K$ using the space $\mathcal{S}_K$, i.e., the use of globally supported RBFs with fixed value of ε within a multilevel residual iteration algorithm for linear problems is pointless.*

Here L could even be a linear differential operator (as used in later chapters). However, for the present discussion of interpolation problems we are only interested in $L = I$.

If one varies the parameter ε with the levels k (*i.e.*, one departs from the non-stationary regime) then the function space $\mathcal{S} = \bigcup_{k=1}^{K} \mathcal{S}_k$ used for the final fit with a multilevel algorithm will be richer than the space $\mathcal{S}_K$ used directly for the finest level $\mathcal{X}_K$ alone. This is clear since the spaces $\mathcal{S}_k$, $k = 1, \ldots, K$, are no longer nested. This implies that, for a "good" sequence of ε-values, one can expect to obtain more accurate fits using the multilevel framework.

This is summarized in

Corollary 32.1 (Rule 2). *The multilevel residual iteration algorithm for linear problems has the potential of being more accurate than a direct fit if the parameter ε is varied with the levels.*

We now illustrate Rules 1 and 2 with a scattered data fitting problem in $\mathbb{R}^2$. We use Franke's function (2.2) on the unit square as our test function. We take the (radial) basic function to be a multiquadric $\varphi(r) = \sqrt{1 + (\varepsilon r)^2}$. The point sets $\mathcal{X}_k$ are given by $(2^k + 1)^2$ equally spaced points in the unit square and are therefore nested with fill-distance $h_{\mathcal{X}_k,\Omega} = 1/2^k$.

In all of our numerical examples we list RMS-errors calculated on a fine evaluation mesh of 40×40 uniformly spaced points in the unit square.

For the first example we fix the multiquadric shape parameter at $\varepsilon = 10/3$ throughout. The errors and rates in Table 32.5 indicate the well-known spectral convergence behavior of multiquadrics. The last row in the table also shows that the parameter ε is too small for this point set and the matrix is so ill-conditioned that the approximation is starting to be contaminated by roundoff errors. According to Rule 1 there is no difference between using the multilevel algorithm and a direct fit on N points. This can also be observed numerically.

In order to illustrate Rule 2 we repeat the above example, but now take $\varepsilon = \sqrt{N}/2$ (essentially a stationary approach). This time we list what happens with the multilevel algorithm and compare this to the results obtained by computing the

Table 32.5 MQ fit to Franke's function with fixed value of $\varepsilon = 10/3$.

mesh	RMS-error	rate
3×3	1.802052e-001	
5×5	2.807009e-002	2.6825
9×9	4.009608e-003	2.8075
17×17	3.885488e-005	6.6892
33×33	2.294910e-008	10.7254
65×65	1.511150e-008	0.6028

approximation directly in one step on the sets $\mathcal{X}_k$, $k = 1, \ldots, 6$ (*c.f.* Table 32.6).

Table 32.6 Multilevel MQ and direct fits to Franke's function with variable value of $\varepsilon = \sqrt{N}/2$.

		multilevel		direct	
mesh	ε	RMS-error	rate	RMS-error	rate
3×3	.667	1.847122e-001		1.847122e-001	
5×5	.400	3.128037e-002	2.5619	3.105411e-002	2.5724
9×9	.222	4.288046e-003	2.8669	4.036275e-003	2.9437
17×17	.118	1.598555e-004	4.7455	1.004206e-004	5.3289
33×33	.061	1.187541e-005	3.7507	1.919679e-005	2.3871
65×65	.031	1.310485e-006	3.1798	4.700632e-006	2.0299

Note that, with the varying parameter ε, up to 289 points the direct approach is more accurate, but then the multilevel approach does a better job. This is due to the fact that the matrices for the denser point sets become increasingly ill-conditioned (even with the adjusted ε-value) and therefore the direct fits with 1089 or 4225 points are likely to be inaccurate. With the multilevel algorithm the fits on the finer grids act only as "corrections" to the coarse grid fits computed earlier. This agrees with the philosophy of the multilevel algorithm, and here is where the richer function space pays off.

Finally, it is also possible to combine the fixed level iteration of the previous chapter with the multilevel iteration of this chapter, *i.e.*, we can replace the interpolation steps in the multilevel scheme by fixed level iteration of an appropriate approximation method. We do not report any such experiments here.

Chapter 33

Adaptive Iteration

The two adaptive algorithms discussed in this chapter were both conceived to yield an approximate solution to the RBF interpolation problem. However, they have some similarity with the least squares knot insertion algorithm of Chapter 21 as well as with the iterative algorithms of the previous two chapters. The contents of this chapter are based mostly on the papers [Faul and Powell (1999); Faul and Powell (2000); Schaback and Wendland (2000a); Schaback and Wendland (2000b)] and the book [Wendland (2005a)].

33.1 A Greedy Adaptive Algorithm

We concentrate on systems for strictly positive definite functions (variations for strictly conditionally positive definite functions also exist). One of the central ingredients (and main differences to the previous iterative algorithms) is the use of the native space inner product discussed in Chapter 13. As always, we assume that our data sites are $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, but now we also consider a second set $\mathcal{Y} \subseteq \mathcal{X}$.

If we let $\mathcal{P}_f^{\mathcal{Y}}$ be the interpolant to f on $\mathcal{Y} \subseteq \mathcal{X}$, then $\langle f - \mathcal{P}_f^{\mathcal{Y}}, \mathcal{P}_f^{\mathcal{Y}} \rangle_{\mathcal{N}_\Phi(\Omega)} = 0$ by Lemma 18.1 (with $u = f$) and we obtain the energy split (see Corollary 18.1)

$$\|f\|^2_{\mathcal{N}_\Phi(\Omega)} = \|f - \mathcal{P}_f^{\mathcal{Y}}\|^2_{\mathcal{N}_\Phi(\Omega)} + \|\mathcal{P}_f^{\mathcal{Y}}\|^2_{\mathcal{N}_\Phi(\Omega)}.$$

One possible point of view is now to consider an iteration on residuals. To this end we pretend to start with our desired interpolant $r_0 = \mathcal{P}_f^{\mathcal{X}}$ on the entire set $\mathcal{X}$, and an appropriate sequence of sets $\mathcal{Y}_k$, $k = 0, 1, \ldots$ (we will discuss some possible heuristics for choosing these sets later). Then, just as in our earlier residual iterations, we iteratively define

$$r_{k+1} = r_k - \mathcal{P}_{r_k}^{\mathcal{Y}_k}, \qquad k = 0, 1, \ldots. \tag{33.1}$$

Now, the energy splitting identity with $f = r_k$ gives us

$$\|r_k\|^2_{\mathcal{N}_\Phi(\Omega)} = \|r_k - \mathcal{P}_{r_k}^{\mathcal{Y}_k}\|^2_{\mathcal{N}_\Phi(\Omega)} + \|\mathcal{P}_{r_k}^{\mathcal{Y}_k}\|^2_{\mathcal{N}_\Phi(\Omega)} \tag{33.2}$$

or, using the iteration formula (33.1),

$$\|r_k\|^2_{\mathcal{N}_\Phi(\Omega)} = \|r_{k+1}\|^2_{\mathcal{N}_\Phi(\Omega)} + \|r_k - r_{k+1}\|^2_{\mathcal{N}_\Phi(\Omega)}. \tag{33.3}$$

Therefore, using (33.1) and (33.3), we have the following telescoping sum for the partial sums of the norm of the residual updates $\mathcal{P}_{r_k}^{\mathcal{Y}_k}$

$$\begin{aligned}\sum_{k=0}^{K} \|\mathcal{P}_{r_k}^{\mathcal{Y}_k}\|_{\mathcal{N}_\Phi(\Omega)}^2 &= \sum_{k=0}^{K} \|r_k - r_{k+1}\|_{\mathcal{N}_\Phi(\Omega)}^2 \\ &= \sum_{k=0}^{K} \left\{ \|r_k\|_{\mathcal{N}_\Phi(\Omega)}^2 - \|r_{k+1}\|_{\mathcal{N}_\Phi(\Omega)}^2 \right\} \\ &= \|r_0\|_{\mathcal{N}_\Phi(\Omega)}^2 - \|r_{K+1}\|_{\mathcal{N}_\Phi(\Omega)}^2 \le \|r_0\|_{\mathcal{N}_\Phi(\Omega)}^2.\end{aligned}$$

This estimate shows that the sequence of partial sums is monotone increasing and bounded, and therefore convergent — even for a poor choice of the sets $\mathcal{Y}_k$. If we can show that the residuals r_k converge to zero, then we would have that the iteratively computed approximation

$$u_{K+1} = \sum_{k=0}^{K} \mathcal{P}_{r_k}^{\mathcal{Y}_k} = \sum_{k=0}^{K} (r_k - r_{k+1}) = r_0 - r_{K+1} \tag{33.4}$$

converges to the original interpolant $r_0 = \mathcal{P}_f^{\mathcal{X}}$.

While this residual iteration algorithm has some structural similarities with the fixed level algorithm of Chapter 31 we now are considering a way to efficiently compute the interpolant $\mathcal{P}_f^{\mathcal{X}}$ on some fine set $\mathcal{X}$ by using an iteration on subsets of the data. Earlier we approximated the interpolant by iterating an *approximation method* on the full data set, whereas now we are approximating the interpolant by iterating an *interpolation method* on nested (increasing) adaptively chosen subsets of the data.

The present method also has some similarities with the multilevel algorithms of Chapter 32. However, now we are interested in computing the interpolant $\mathcal{P}_f^{\mathcal{X}}$ on the set $\mathcal{X}$ based on a single function Φ, whereas earlier, our final interpolant was the result of using the spaces $\bigcup_{k=1}^{K} \mathcal{N}_{\Phi_k}(\Omega)$, where Φ_k was an appropriately scaled version of the basic function Φ. Moreover, the goal in Chapter 32 was to approximate f, not $\mathcal{P}_f$.

In order to prove convergence of the residual iteration, let us assume that we can find sets of points $\mathcal{Y}_k$ such that at step k at least some fixed percentage of the energy of the residual is picked up by its interpolant, *i.e.*,

$$\|\mathcal{P}_{r_k}^{\mathcal{Y}_k}\|_{\mathcal{N}_\Phi(\Omega)}^2 \ge \gamma \|r_k\|_{\mathcal{N}_\Phi(\Omega)}^2 \tag{33.5}$$

with some fixed $\gamma \in (0, 1]$. Then (33.3) and the iteration formula (33.1) imply

$$\|r_{k+1}\|_{\mathcal{N}_\Phi(\Omega)}^2 = \|r_k\|_{\mathcal{N}_\Phi(\Omega)}^2 - \|\mathcal{P}_{r_k}^{\mathcal{Y}_k}\|_{\mathcal{N}_\Phi(\Omega)}^2,$$

and therefore

$$\|r_{k+1}\|_{\mathcal{N}_\Phi(\Omega)}^2 \le \|r_k\|_{\mathcal{N}_\Phi(\Omega)}^2 - \gamma \|r_k\|_{\mathcal{N}_\Phi(\Omega)}^2 = (1 - \gamma) \|r_k\|_{\mathcal{N}_\Phi(\Omega)}^2.$$

Applying this bound recursively yields

Theorem 33.1. *If the choice of sets $\mathcal{Y}_k$ satisfies (33.5), then the residual iteration (33.4) converges linearly in the native space norm, and after K steps of iterative refinement there is an error bound*

$$\|r_0 - u_K\|^2_{\mathcal{N}_\Phi(\Omega)} = \|r_K\|^2_{\mathcal{N}_\Phi(\Omega)} \leq (1-\gamma)^K \|r_0\|^2_{\mathcal{N}_\Phi(\Omega)}.$$

This theorem has various limitations. In particular, the norm involves the function Φ which makes it difficult to find sets $\mathcal{Y}_k$ that satisfy (33.5). Moreover, the native space norm of the initial residual r_0 is not known, either. Therefore, using an equivalent discrete norm on the set $\mathcal{X}$, Schaback and Wendland establish an estimate of the form

$$\|r_0 - u_K\|^2_{\mathcal{X}} \leq \frac{C}{c}\left(1 - \delta\frac{c^2}{C^2}\right)^{K/2} \|r_0\|^2_{\mathcal{X}},$$

where c and C are constants denoting the norm equivalence, *i.e.*,

$$c\|u\|_{\mathcal{X}} \leq \|u\|_{\mathcal{N}_\Phi(\Omega)} \leq C\|u\|_{\mathcal{X}}$$

for any $u \in \mathcal{N}_\Phi(\Omega)$, and where δ is a constant analogous to γ (but based on use of the discrete norm $\|\cdot\|_{\mathcal{X}}$ in (33.5)). In fact, any discrete ℓ_p norm on $\mathcal{X}$ can be used. In the implementation below we will use the maximum norm.

In [Schaback and Wendland (2000b)] a basic version of this algorithm — where the sets $\mathcal{Y}_k$ consist of a single point — is described and tested. The resulting approximation yields the *best K-term approximation* to the interpolant. This idea is related to the concept of *greedy approximation algorithms* (see, *e.g.*, [Temlyakov (1998)]) and *sparse approximation* (see, *e.g.*, [Girosi (1998)]).

If the set $\mathcal{Y}_k$ consists of only a single point $\boldsymbol{y}_k$, then the partial interpolant $\mathcal{P}^{\mathcal{Y}_k}_{r_k}$ is particularly simple, namely

$$\mathcal{P}^{\mathcal{Y}_k}_{r_k} = \beta\Phi(\cdot, \boldsymbol{y}_k)$$

with

$$\beta = \frac{r_k(\boldsymbol{y}_k)}{\Phi(\boldsymbol{y}_k, \boldsymbol{y}_k)}.$$

This follows immediately from the usual RBF expansion (which consists of only one term here) and the interpolation condition $\mathcal{P}^{\mathcal{Y}_k}_{r_k}(\boldsymbol{y}_k) = r_k(\boldsymbol{y}_k)$.

The point $\boldsymbol{y}_k$ is picked to be the point in $\mathcal{X}$ where the residual is largest, *i.e.*, $|r_k(\boldsymbol{y}_k)| = \|r_k\|_\infty$. This choice of "set" $\mathcal{Y}_k$ certainly satisfies the constraint (33.5) since Φ is strictly positive definite and therefore has its maximum at the origin (cf. Property (4) in Theorem 3.1). Moreover, the interpolation problem is (approximately) solved without having to invert any linear systems. The algorithm can be summarized as

Algorithm 33.1. Greedy one-point algorithm

Input data locations $\mathcal{X}$, associated values of f, tolerance tol > 0
Set initial residual $r_0 = \mathcal{P}^{\mathcal{X}}_f$, initialize $u_0 = 0$, $e = \infty$, $k = 0$

Choose starting point $\boldsymbol{y}_k \in \mathcal{X}$
While $e >$ tol do
  Set $\beta = \dfrac{r_k(\boldsymbol{y}_k)}{\Phi(\boldsymbol{y}_k, \boldsymbol{y}_k)}$
  For $1 \le i \le N$ do
    $r_{k+1}(\boldsymbol{x}_i) = r_k(\boldsymbol{x}_i) - \beta\Phi(\boldsymbol{x}_i, \boldsymbol{y}_k)$
    $u_{k+1}(\boldsymbol{x}_i) = u_k(\boldsymbol{x}_i) + \beta\Phi(\boldsymbol{x}_i, \boldsymbol{y}_k)$
  end
  Find $e = \max_{\mathcal{X}} |r_{k+1}|$ and the point $\boldsymbol{y}_{k+1}$ where it occurs
  Increment $k = k + 1$
end

A MATLAB implementation of the greedy one-point algorithm is presented as Program 33.1. The implementation is quite straightforward using our function `DistanceMatrix` in conjunction with the anonymous function `rbf` to compute both $\Phi(\boldsymbol{y}_k, \boldsymbol{y}_k)$ (on lines 18 and 19) and $\Phi(\boldsymbol{x}_i, \boldsymbol{y}_k)$ needed for the updates of the residual r_{k+1} and the approximation u_{k+1} on lines 21–24. The algorithm demands that we compute the residuals r_k on the data sites. On the other hand, the partial approximants u_k to the interpolant can be evaluated anywhere. If we choose to do this also at the data sites, then we are required to use a plotting routine that differs from our usual one (such as `trisurf` built on a triangulation of the data sites obtained with the help of `delaunayn`). We instead choose to follow the same procedure as in all of our other programs, *i.e.*, to evaluate u_k on a 40×40 grid of equally spaced points. This has been implemented on lines 21–25 of the program. Note that the updating procedure has been vectorized in MATLAB allowing us to avoid the for-loop over i in the algorithm.

It is important to realize that we never actually compute the initial residual $r_0 = \mathcal{P}_f^{\mathcal{X}}$. All we require are the *values* of r_0 on the grid $\mathcal{X}$ of data sites. However, since $\mathcal{P}_f^{\mathcal{X}}|_{\mathcal{X}} = f|_{\mathcal{X}}$ the values $r_0(\boldsymbol{x}_i)$ are given by the interpolation data $f(\boldsymbol{x}_i)$ (see line 13 of the code). Moreover, since the sets $\mathcal{Y}_k$ are subsets of $\mathcal{X}$ the value $r_k(\boldsymbol{y}_k)$ required to determine β is actually one of the current residual values (see line 20 of the code).

The final approximation to the interpolant and the approximation error are plotted with the commands given on lines 36–38. The commands for the plots of the points $\boldsymbol{y}_k$ selected by the algorithm and the norm of the residual displayed in Figures 33.1 and 33.3 are included on lines 39 and 40.

Program 33.1. `RBFGreedyOnePoint2D.m`

```
% RBFGreedyOnePoint2D
% Script that performs greedy one point algorithm for adaptive
% 2D RBF interpolation
% Calls on: DistanceMatrix
```

```
rbf = @(e,r) exp(-(e*r).^2);      % Gaussian RBF
ep = 5.5;  % Parameter for basis function
% Define Franke's function as testfunction
f1 = @(x,y) 0.75*exp(-((9*x-2).^2+(9*y-2).^2)/4);
f2 = @(x,y) 0.75*exp(-((9*x+1).^2/49+(9*y+1).^2/10));
f3 = @(x,y) 0.5*exp(-((9*x-7).^2+(9*y-3).^2)/4);
f4 = @(x,y) 0.2*exp(-((9*x-4).^2+(9*y-7).^2));
testfunction = @(x,y) f1(x,y)+f2(x,y)+f3(x,y)-f4(x,y);
% Number and type of data points
N = 16641; gridtype = 'h';
neval = 40; grid = linspace(0,1,neval);
[xe,ye] = meshgrid(grid); epoints = [xe(:) ye(:)];
% Tolerance; stopping criterion
tol = 1e-5; kmax = 1000;
% Load data points
name = sprintf('Data2D_%d%s',N,gridtype); load(name)
% Initialize residual and fit
r_old = testfunction(dsites(:,1),dsites(:,2));
u_old = 0;
k = 1; maxres(k) = 999999;
% Use an (arbitrary) initial point
ykidx = (N+1)/2; yk(k,:) = dsites(ykidx,:);
while (maxres(k) > tol && k < kmax)
   % Evaluate basis function at yk
   DM_data = DistanceMatrix(yk(k,:),yk(k,:));
   IM = rbf(ep,DM_data);
   beta = r_old(ykidx)/IM;
   % Compute evaluation matrices for residual and fit
   DM_res = DistanceMatrix(dsites,yk(k,:));
   RM = rbf(ep,DM_res);
   DM_eval = DistanceMatrix(epoints,yk(k,:));
   EM = rbf(ep,DM_eval);
   % Update residual and fit
   r = r_old - beta*RM;    u = u_old + beta*EM;
   % Find new point to add
   [sr,idx] = sort(abs(r));
   maxres(k+1) = sr(end);
   ykidx = idx(end);    yk(k+1,:) = dsites(ykidx,:);
   r_old = r;    u_old = u;
   k = k + 1;
end
% Compute exact solution
```

```
exact = testfunction(epoints(:,1),epoints(:,2));
maxerr = norm(u-exact,inf); rms_err = norm(u-exact)/neval;
fprintf('RMS error:     %e\n', rms_err)
fprintf('Maximum error: %e\n', maxerr)
fview = [160,20]; % viewing angles for plot
PlotSurf(xe,ye,u,neval,exact,maxerr,fview);
PlotError2D(xe,ye,u,exact,maxerr,neval,fview);
figure; plot(yk(:,1),yk(:,2),'ro')
figure; semilogy(maxres,'b');
```

To illustrate the greedy one-point algorithm we perform two experiments. Both tests use data obtained by sampling Franke's function at 16641 Halton points in $[0,1]^2$. However, the first test is based on Gaussians, while the second one uses inverse multiquadrics. For both tests we use the same shape parameter $\varepsilon = 5.5$. This results in the inverse multiquadrics having a more global influence than the Gaussians. This effect is clearly evident in the first few approximations to the interpolants in Figures 33.2 and 33.4.

Figure 33.4, in particular, shows that the greedy algorithm enforces interpolation of the data only on the most recent set $\mathcal{Y}_k$ (*i.e.*, for the one-point algorithm studied here only at a single point). If one wants to maintain the interpolation achieved in previous iterations, then the sets $\mathcal{Y}_k$ should be nested. This, however, would have a significant effect on the execution time of the algorithm since the matrices at each step would increase in size.

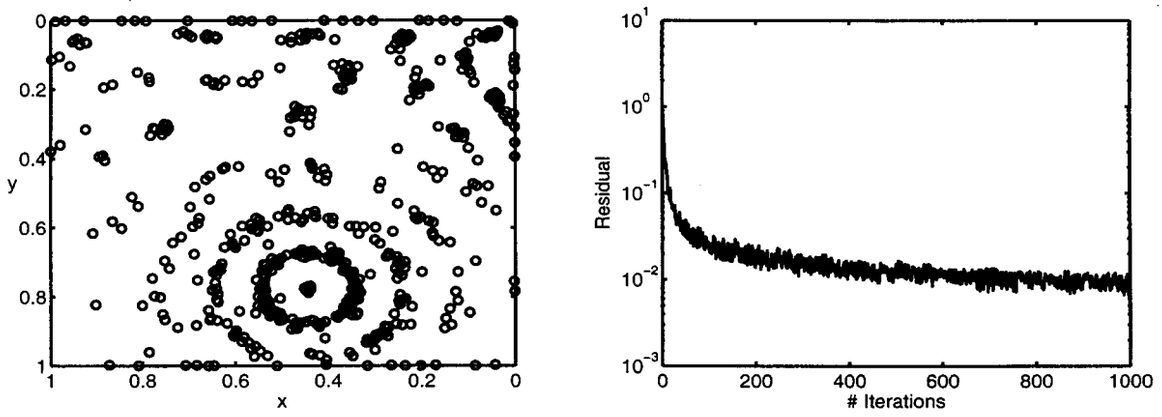


Fig. 33.1 1000 selected points and residual for greedy one point algorithm with Gaussian RBFs and $N = 16641$ data points.

In order to obtain our approximate interpolants we used a tolerance of 10^{-5} along with an additional upper limit of `kmax=1000` on the number of iterations. For both tests the algorithm uses up all 1000 iterations. The final maximum residual for Gaussians is `maxres` $= 0.0075$, while for inverse MQs we have `maxres` $= 0.0035$. In both cases there occurred several multiple point selections. Contrary to interpo-

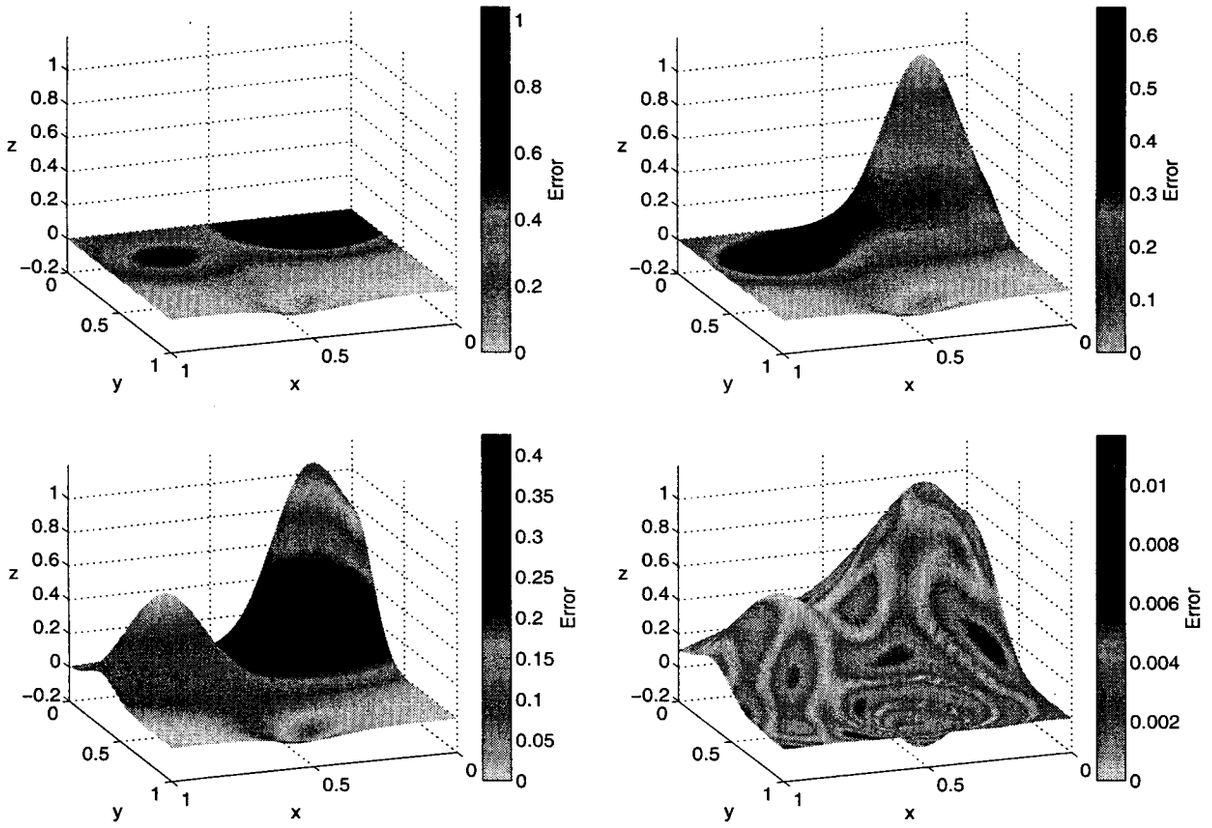


Fig. 33.2 Fits of Franke's function for greedy one point algorithm with Gaussian RBFs and $N = 16641$ data points. Top left to bottom right: 1 point, 2 points, 4 points, final fit with 1000 points.

lation problems based on the solution of a linear system, multiple point selections do not pose a problem here.

One advantage of this very simple algorithm is that no linear systems need to be solved. This allows us to approximate the interpolants for large data sets even for globally supported basis functions, and also with small values of ε (and therefore an associated ill-conditioned interpolation matrix). One should not expect too much in this case, however, as the results in Figure 33.5 show where we used a value of $\varepsilon = 0.1$ for the shape parameter. As with the fixed level iteration of approximate MLS approximants based on flat generating functions, a lot of smoothing occurs so that the convergence to the RBF interpolant is very slow.

Moreover, in the pseudo-code of the algorithm matrix-vector multiplications are not required, either. However, MATLAB allows for a vectorization of the for-loop which does result in two matrix-vector multiplications.

For practical situations, *e.g.*, for smooth radial basis functions and densely distributed points in $\mathcal{X}$ the convergence can be rather slow. The simple greedy algorithm described above is extended in [Schaback and Wendland (2000b)] to a version that adaptively uses basis functions of varying scales.

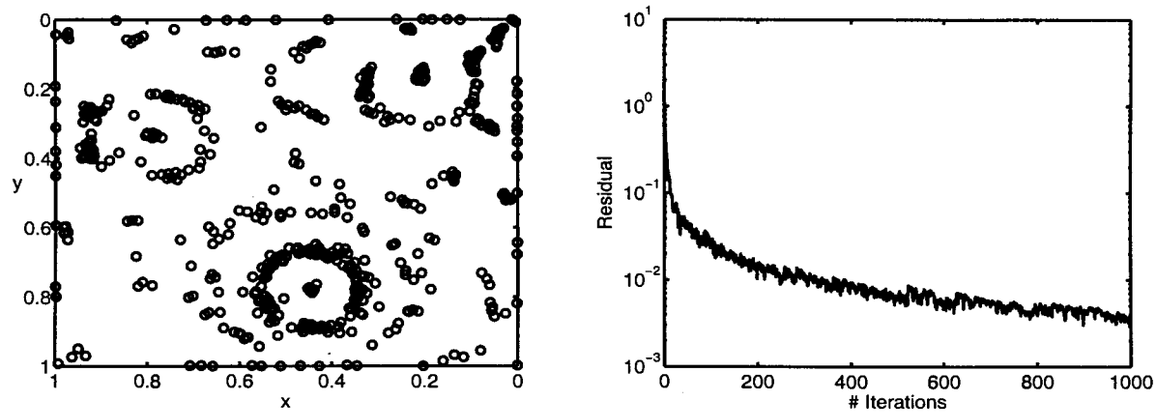


Fig. 33.3 1000 selected points and residual for greedy one point algorithm with IMQ RBFs and $N = 16641$ data points.

33.2 The Faul-Powell Algorithm

Another iterative algorithm was suggested in [Faul and Powell (1999); Faul and Powell (2000)]. From our earlier discussions we know that it is possible to express the radial basis function interpolant in terms of cardinal functions u_j^*, $j = 1, \ldots, N$, *i.e.*,

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{j=1}^{N} f(\boldsymbol{x}_j) u_j^*(\boldsymbol{x}).$$

The basic idea of the Faul-Powell algorithm is to use *approximate cardinal functions* Ψ_j instead. Of course, this will only give an approximate value for the interpolant, and therefore an iteration on the residuals is suggested to improve the accuracy of this approximation.

The basic philosophy of this algorithm is very similar to that of the fixed level iteration of Chapter 31. In particular, the Faul-Powell algorithm can also be interpreted as a Krylov subspace method. However, instead of taking approximate MLS generating functions, the approximate cardinal functions Ψ_j, $j = 1, \ldots, N$, are determined as linear combinations of the basis functions $\Phi(\cdot, \boldsymbol{x}_\ell)$ for the interpolant, *i.e.*,

$$\Psi_j = \sum_{\ell \in \mathcal{L}_j} b_{j\ell} \Phi(\cdot, \boldsymbol{x}_\ell), \tag{33.6}$$

where $\mathcal{L}_j$ is an index set consisting of n ($n \approx 50$) indices that are used to determine the approximate cardinal function. For example, the n nearest neighbors of $\boldsymbol{x}_j$ will usually do. In general, the choice of index sets allows much freedom, and this is the reason why we include the algorithm in this chapter on adaptive iterative methods. Also, as pointed out at the end of this section, there is a certain duality between the Faul-Powell algorithm and the greedy algorithm of the previous section.

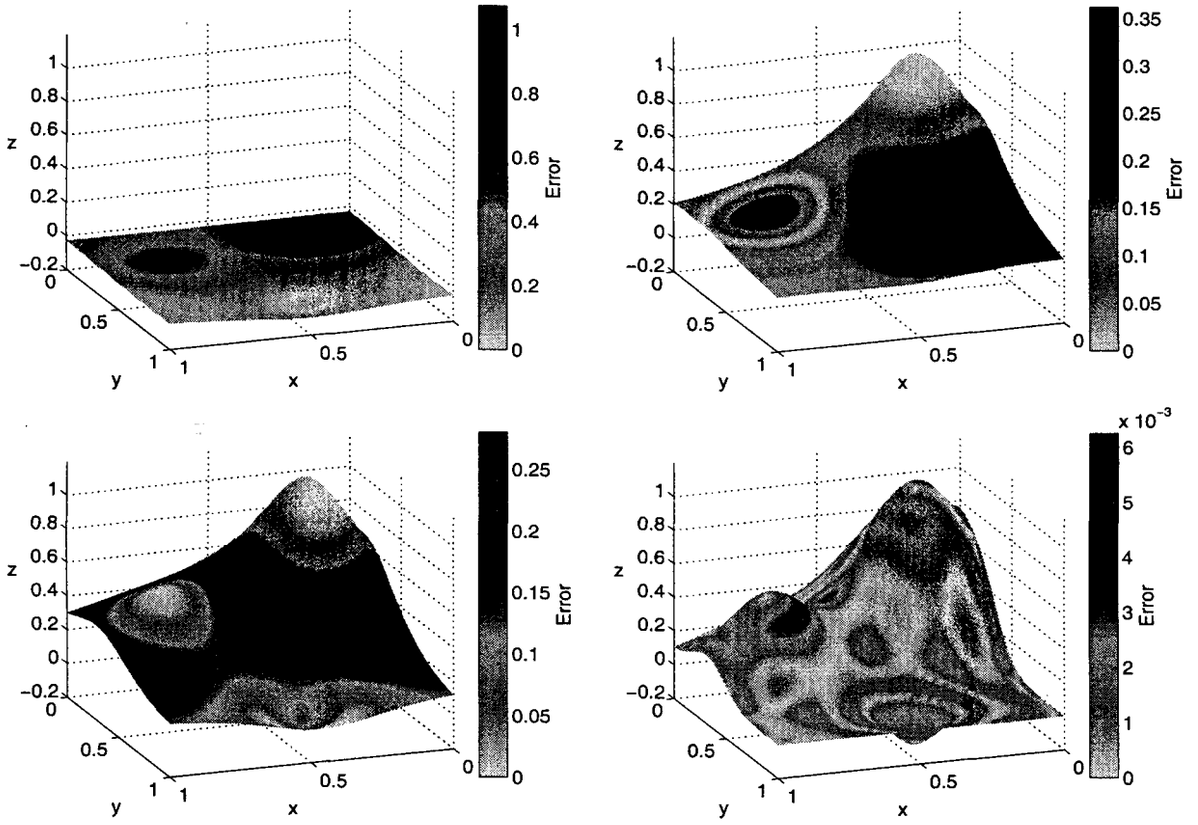


Fig. 33.4 Fits of Franke's function for greedy one point algorithm with IMQ RBFs and $N = 16641$ data points. Top left to bottom right: 1 point, 2 points, 4 points, final fit with 1000 points.

For every $j = 1, \ldots, N$, the coefficients $b_{j\ell}$ are found as solution of the (relatively small) $n \times n$ linear system

$$\Psi_j(\boldsymbol{x}_i) = \delta_{jk}, \qquad i \in \mathcal{L}_j. \tag{33.7}$$

These approximate cardinal functions are computed in a pre-processing step.

As before, in its simplest form the residual iteration can be formulated as

$$u^{(0)}(\boldsymbol{x}) = \sum_{j=1}^{N} f(\boldsymbol{x}_j)\Psi_j(\boldsymbol{x})$$

$$u^{(k+1)}(\boldsymbol{x}) = u^{(k)}(\boldsymbol{x}) + \sum_{j=1}^{N} \left[f(\boldsymbol{x}_j) - u^{(k)}(\boldsymbol{x}_j) \right] \Psi_j(\boldsymbol{x}), \qquad k = 0, 1, \ldots.$$

Instead of adding the contribution of all approximate cardinal functions at the same time, this is done in a three-step process in the Faul-Powell algorithm. To this end, we choose index sets $\mathcal{L}_j$, $j = 1, \ldots, N-n$, such that $\mathcal{L}_j \subseteq \{j, j+1, \ldots, N\}$ while making sure that $j \in \mathcal{L}_j$. Also, if one wants to use this algorithm to approximate the interpolant based on conditionally positive definite functions of order m, then one needs to ensure that the corresponding centers form an $(m-1)$-unisolvent set and append a polynomial to the local expansion (33.6).

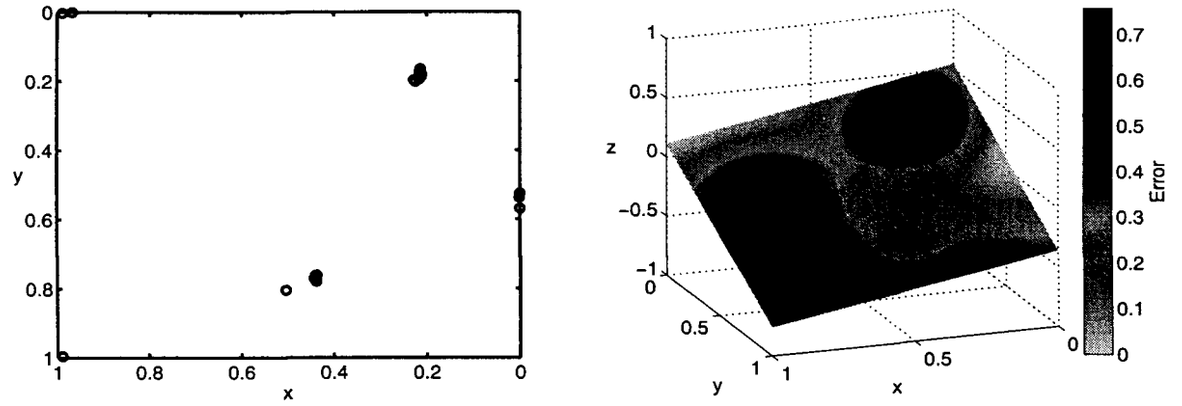


Fig. 33.5 1000 selected points (only 20 of them distinct) and fit of Franke's function for greedy one point algorithm with flat Gaussian RBFs ($\varepsilon = 0.1$) and $N = 16641$ data points.

Now, in the first step we define $u_0^{(k)} = u^{(k)}$, and then iterate

$$u_j^{(k)} = u_{j-1}^{(k)} + \theta_j^{(k)} \Psi_j, \qquad j = 1, \ldots, N - n, \tag{33.8}$$

with

$$\theta_j^{(k)} = \frac{\langle \mathcal{P}_f - u_{j-1}^{(k)}, \Psi_j \rangle_{\mathcal{N}_\Phi(\Omega)}}{\langle \Psi_j, \Psi_j \rangle_{\mathcal{N}_\Phi(\Omega)}}. \tag{33.9}$$

The stepsize $\theta_j^{(k)}$ is chosen so that the native space best approximation to the residual $\mathcal{P}_f - u_{j-1}^{(k)}$ from the space spanned by the approximate cardinal functions Ψ_j is added. Using the representation (33.6) of Ψ_j in terms of the global basis $\{\Phi(\cdot, \boldsymbol{x}_i) : \ i = 1, \ldots, N\}$, the reproducing kernel property of Φ, and the (local) cardinality property (33.7) of Ψ_j we can calculate the denominator of (33.9) as

$$\begin{aligned}
\langle \Psi_j, \Psi_j \rangle_{\mathcal{N}_\Phi(\Omega)} &= \langle \Psi_j, \sum_{\ell \in \mathcal{L}_j} b_{j\ell} \Phi(\cdot, \boldsymbol{x}_\ell) \rangle_{\mathcal{N}_\Phi(\Omega)} \\
&= \sum_{\ell \in \mathcal{L}_j} b_{j\ell} \langle \Psi_j, \Phi(\cdot, \boldsymbol{x}_\ell) \rangle_{\mathcal{N}_\Phi(\Omega)} \\
&= \sum_{\ell \in \mathcal{L}_j} b_{j\ell} \Psi_j(\boldsymbol{x}_\ell) = b_{jj}
\end{aligned}$$

since we have $j \in \mathcal{L}_j$ by construction of the index set $\mathcal{L}_j$. Similarly, we get for the numerator

$$\begin{aligned}
\langle \mathcal{P}_f - u_{j-1}^{(k)}, \Psi_j \rangle_{\mathcal{N}_\Phi(\Omega)} &= \langle \mathcal{P}_f - u_{j-1}^{(k)}, \sum_{\ell \in \mathcal{L}_j} b_{j\ell} \Phi(\cdot, \boldsymbol{x}_\ell) \rangle_{\mathcal{N}_\Phi(\Omega)} \\
&= \sum_{\ell \in \mathcal{L}_j} b_{j\ell} \langle \mathcal{P}_f - u_{j-1}^{(k)}, \Phi(\cdot, \boldsymbol{x}_\ell) \rangle_{\mathcal{N}_\Phi(\Omega)} \\
&= \sum_{\ell \in \mathcal{L}_j} b_{j\ell} \left(\mathcal{P}_f - u_{j-1}^{(k)} \right) (\boldsymbol{x}_\ell)
\end{aligned}$$

$$= \sum_{\ell \in \mathcal{L}_j} b_{j\ell} \left(f(\boldsymbol{x}_\ell) - u_{j-1}^{(k)}(\boldsymbol{x}_\ell) \right).$$

Therefore (33.8) and (33.9) can be written as

$$u_j^{(k)} = u_{j-1}^{(k)} + \frac{\Psi_j}{b_{jj}} \sum_{\ell \in \mathcal{L}_j} b_{j\ell} \left(f(\boldsymbol{x}_\ell) - u_{j-1}^{(k)}(\boldsymbol{x}_\ell) \right), \qquad j = 1, \ldots, N-n.$$

In the second step of the Faul-Powell algorithm the residual is interpolated on the remaining n points (collected via the index set $\mathcal{L}^*$). Thus, we find a function $v^{(k)}$ in $\operatorname{span}\{\Phi(\cdot, \boldsymbol{x}_j) : j \in \mathcal{L}^*\}$ such that

$$v^{(k)}(\boldsymbol{x}_i) = f(\boldsymbol{x}_i) - u_{N-n}^{(k)}(\boldsymbol{x}_i), \qquad i \in \mathcal{L}^*,$$

and the approximation is updated, *i.e.*,

$$u^{(k+1)} = u_{N-n}^{(k)} + v^{(k)}.$$

In the third step the residuals are updated, *i.e.*,

$$r_i^{(k+1)} = f(\boldsymbol{x}_i) - u^{(k+1)}(\boldsymbol{x}_i), \qquad i = 1, \ldots, N. \tag{33.10}$$

The outer iteration (on k) is now repeated unless the largest of these residuals is small enough.

We can summarize this algorithm as

Algorithm 33.2. Faul-Powell algorithm

Input data locations $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, associated values of f, and tolerance $\text{tol} > 0$

Pre-processing step

Choose n

For $1 \le j \le N - n$ do

Determine the index set $\mathcal{L}_j$

Find the coefficients $b_{j\ell}$ of the approximate cardinal function Ψ_j by solving

$$\Psi_j(\boldsymbol{x}_i) = \delta_{jk}, \qquad i \in \mathcal{L}_j$$

end

Set $k = 0$ and $u_0^{(k)} = 0$

Initialize residuals $r_i^{(k)} = f(\boldsymbol{x}_i),\ i = 1, \ldots, N$

Set $e = \max_{i=1,\ldots,N} |r_i^{(k)}|$

While $e > \text{tol}$ do

For $1 \le j \le N - n$ do

Update

$$u_j^{(k)} = u_{j-1}^{(k)} + \frac{\Psi_j}{b_{jj}} \sum_{\ell \in \mathcal{L}_j} b_{j\ell} \left(f(\boldsymbol{x}_\ell) - u_{j-1}^{(k)}(\boldsymbol{x}_\ell) \right)$$

end
Solve the interpolation problem

$$v^{(k)}(\boldsymbol{x}_i) = f(\boldsymbol{x}_i) - u^{(k)}_{N-n}(\boldsymbol{x}_i), \qquad i \in \mathcal{L}^*$$

Update the approximation

$$u_0^{(k+1)} = u^{(k)}_{N-n} + v^{(k)}$$

Compute new residuals $r_i^{(k+1)} = f(\boldsymbol{x}_i) - u_0^{(k+1)}(\boldsymbol{x}_i),\ i = 1, \ldots, N$
Set new value for $e = \max_{i=1,\ldots,N} |r_i^{(k+1)}|$
Increment $k = k + 1$
end

Faul and Powell prove that this algorithm converges to the solution of the original interpolation problem. Similar to some of the other algorithms, one needs to make sure that the residuals are evaluated efficiently by using, *e.g.*, a fast multipole expansion, fast Fourier transform, or compactly supported functions.

In its most basic form the Krylov subspace algorithm of Faul and Powell can also be explained as a dual approach to the greedy residual iteration algorithm of Schaback and Wendland. Instead of defining appropriate sets of points $\mathcal{Y}_k$, in the Faul and Powell algorithm one picks certain subspaces U_k of the native space. In particular, if U_k is the one-dimensional space $U_k = \text{span}\{\Psi_k\}$ (where Ψ_k is a local approximation to the cardinal function) we get the algorithm described above. For more details see [Schaback and Wendland (2000b)].

We leave the implementation of this algorithm to the reader.

Chapter 34

Improving the Condition Number of the Interpolation Matrix

In Chapter 16 we noted that the system matrices arising in scattered data interpolation with radial basis functions tend to become very ill-conditioned as the minimal separation distance $q_{\mathcal{X}}$ between the data sites $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, is reduced. Therefore it is natural to devise strategies to prevent such instabilities by either preconditioning the system, or by finding a better basis for the approximation space we are using. The former approach is standard procedure in numerical linear algebra, and in fact we can use any of the well-established methods (such as preconditioned conjugate gradient iteration) to improve the stability and convergence of the interpolation systems that arise for strictly positive definite functions. In particular, the sparse systems that arise in (multilevel) interpolation with compactly supported radial basis functions can be solved efficiently with the preconditioned conjugate gradient method. However, in our implementation (see the discussion in Section 12.1) we use MATLAB's `sparse` function which takes advantage of state-of-the-art direct methods for sparse linear systems.

The second approach to improving the condition number of the interpolation system, *i.e.*, the idea of using a more stable basis, is well known from univariate polynomial and spline interpolation. The Lagrange basis functions for univariate polynomial interpolation are the ideal basis if we are interested in stably solving the interpolation equations since the resulting interpolation matrix is the identity matrix (which is certainly much better conditioned than, *e.g.*, the Vandermonde matrix that we get if we use a monomial basis). Similarly, B-splines give rise to diagonally dominant, sparse system matrices which are much easier to deal with than the matrices we would get if we were to represent a spline interpolant using the alternative truncated power basis. Both of these examples are studied in great detail in standard numerical analysis texts (see, *e.g.*, [Kincaid and Cheney (2002)]) or in the literature on splines (see, *e.g.*, [Schumaker (1981)]). We will address an analogous approach for radial basis functions in Section 34.4 below.

Before we describe any of the specialized preconditioning procedures for radial basis function interpolation matrices we give two examples presented in the early RBF paper [Jackson (1989a)] to illustrate the effects of and motivation for preconditioning in the context of radial basis functions.

34.1 Preconditioning: Two Simple Examples

Example 34.1. Let $s = 1$ and consider interpolation based on $\varphi(r) = r$ with no polynomial terms added. As data sites we choose $\mathcal{X} = \{1, 2, \ldots, 10\}$. This leads to the system matrix

$$A = \begin{bmatrix} 0 & 1 & 2 & \mathbf{3} & \ldots & 9 \\ 1 & 0 & 1 & 2 & \ldots & 8 \\ 2 & 1 & 0 & 1 & \ldots & 7 \\ \mathbf{3} & 2 & 1 & 0 & \ldots & 6 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 9 & 8 & 7 & 6 & \ldots & 0 \end{bmatrix}$$

with ℓ_2-condition number $\text{cond}(A) \approx 67$. Instead of solving the linear system $A\boldsymbol{c} = \boldsymbol{y}$, where $\boldsymbol{y} = [y_1, \ldots, y_{10}]^T \in \mathbb{R}^{10}$ is a vector of given real numbers (data values), we can find a suitable matrix B to pre-multiply both sides of the equation such that the system is simpler to solve. Ideally, the new system matrix BA should be the identity matrix, *i.e.*, B should be an *approximate inverse* of A. Thus, having found an appropriate matrix B, we must now solve the linear system $BA\boldsymbol{c} = B\boldsymbol{y}$. The matrix B is usually referred to as the *preconditioner* of the linear system. For the matrix A above we can choose a preconditioner B as

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 & \ldots & 0 & 0 \\ \frac{1}{2} & -1 & \frac{1}{2} & 0 & \ldots & 0 & 0 \\ 0 & \frac{1}{2} & -1 & \frac{1}{2} & \ldots & 0 & 0 \\ 0 & 0 & \frac{1}{2} & -1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & -1 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & \ldots & 0 & 1 \end{bmatrix}.$$

This leads to the following preconditioned system matrix

$$BA = \begin{bmatrix} 0 & 1 & 2 & \ldots & 8 & 9 \\ 0 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & 0 \\ 9 & 8 & 7 & \ldots & 1 & 0 \end{bmatrix}$$

in the system $BA\boldsymbol{c} = B\boldsymbol{y}$. Note that BA is almost an identity matrix. One can easily check that now $\text{cond}(BA) \approx 45$.

The motivation for this choice of B is the following. The function $\varphi(r) = r$ or $\Phi(x) = |x|$ is a fundamental solution of the Laplacian Δ ($= \frac{d^2}{dx^2}$ in the one-dimensional case), *i.e.*

$$\Delta\Phi(x) = \frac{d^2}{dx^2}|x| = \frac{1}{2}\delta_0(x),$$

where δ_0 is the Dirac delta function centered at zero. Thus, B is chosen as a discretization of the Laplacian with special choices at the endpoints of the data set.

Example 34.2. For non-uniformly distributed data we can use a different discretization of the Laplacian Δ for each row of B. To see this, let $s = 1$, $\mathcal{X} = \{1, \frac{3}{2}, \frac{5}{2}, 4, \frac{3}{2}\}$, and again consider interpolation with the radial function $\varphi(r) = r$. Then

$$A = \begin{bmatrix} 0 & \frac{1}{2} & \frac{3}{2} & \mathbf{3} & \frac{7}{2} \\ \frac{1}{2} & 0 & 1 & \frac{5}{2} & 3 \\ \frac{3}{2} & 1 & 0 & \frac{3}{2} & 2 \\ 3 & \frac{5}{2} & \frac{3}{2} & 0 & \frac{1}{2} \\ \frac{7}{2} & 3 & 2 & \frac{1}{2} & 0 \end{bmatrix}$$

with $\text{cond}(A) \approx 18.15$, and if we choose

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & -\frac{3}{2} & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & -\frac{5}{6} & \frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{3} & -\frac{4}{3} & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

based on second-order backward differences of the points in $\mathcal{X}$, then the preconditioned system to be solved becomes

$$\begin{bmatrix} 0 & \frac{1}{2} & \frac{3}{2} & 3 & \frac{7}{2} \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \frac{7}{2} & 3 & 2 & \frac{1}{2} & 0 \end{bmatrix} \boldsymbol{c} = B\boldsymbol{y}.$$

Once more, this system is almost trivial to solve and has an improved condition number of $\text{cond}(BA) \approx 8.94$.

34.2 Early Preconditioners

Ill-conditioning of the interpolation matrices was identified as a serious problem very early, and Nira Dyn along with some of her co-workers (see, *e.g.*, [Dyn (1987); Dyn (1989); Dyn and Levin (1983); Dyn *et al.* (1986)]) provided some of the first preconditioning strategies tailored especially to radial basis function interpolants.

For the following discussion we consider the general interpolation problem that includes polynomial reproduction (see Chapter 6). Therefore, we have to solve the following system of linear equations

$$\begin{bmatrix} A & P \\ P^T & O \end{bmatrix} \begin{bmatrix} \boldsymbol{c} \\ \boldsymbol{d} \end{bmatrix} = \begin{bmatrix} \boldsymbol{y} \\ 0 \end{bmatrix}, \tag{34.1}$$

with the individual pieces given by $A_{jk} = \varphi(\|\boldsymbol{x}_j - \boldsymbol{x}_k\|)$, $j,k = 1,\ldots,N$, $P_{j\ell} = p_\ell(\boldsymbol{x}_j)$, $j = 1,\ldots,N$, $\ell = 1,\ldots,M$, $\boldsymbol{c} = [c_1,\ldots,c_N]^T$, $\boldsymbol{d} = [d_1,\ldots,d_M]^T$, $\boldsymbol{y} = [y_1,\ldots,y_N]^T$, O an $M \times M$ zero matrix, and $\mathbf{0}$ a zero vector of length M with $M = \dim \mathrm{H}^s_{m-1}$. Here, as discussed earlier, φ should be strictly conditionally positive definite of order m and radial on $\mathbb{R}^s$ and the set $\mathcal{X} = \{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N\}$ should be $(m-1)$-unisolvent.

The preconditioning scheme proposed by Dyn and her co-workers is a generalization of the simple differencing scheme discussed above. It is motivated by the fact that the polyharmonic splines (*i.e.*, thin plate splines and radial powers)

$$\varphi(r) = \begin{cases} r^{2k-s} \log r, & s \text{ even}, \\ r^{2k-s}, & s \text{ odd}, \end{cases}$$

$2k > s$, are fundamental solutions of the k-th iterated Laplacian in $\mathbb{R}^s$, *i.e.*

$$\Delta^k \varphi(\|\boldsymbol{x}\|) = c\delta_{\mathbf{0}}(\boldsymbol{x}),$$

where $\delta_{\mathbf{0}}$ is the Dirac delta function centered at the origin, and c is an appropriate constant.

For the (inverse) multiquadrics $\varphi(r) = (1+r^2)^{\pm 1/2}$, which are also discussed in the papers mentioned above, application of the Laplacian yields a similar limiting behavior, *i.e.*

$$\lim_{r\to\infty} \Delta^k \varphi(r) = 0,$$

and for $r \to 0$

$$\Delta^k \varphi(r) \gg 1.$$

One now wants to discretize the Laplacian on the (irregular) mesh given by the (scattered) data sites in $\mathcal{X}$. To this end the authors of [Dyn *et al.* (1986)] suggest the following procedure for the case of scattered data interpolation over $\mathbb{R}^2$.

(1) Start with a triangulation of the set $\mathcal{X}$, *e.g.*, the will do. This triangulation can be visualized as follows.

   (a) Begin with the points in $\mathcal{X}$ and construct their or *Voronoi diagram*. The Dirichlet tile of a particular point $\boldsymbol{x}$ is that subset of points in $\mathbb{R}^2$ which are closer to $\boldsymbol{x}$ than to any other point in $\mathcal{X}$. The dashed lines in Figure 34.1 denote the Dirichlet tesselation for the set of 25 Halton points (circles) in $[0,1]^2$.

   (b) Construct the Delaunay triangulation, which is the dual of the Dirichlet tesselation, *i.e.*, connect all strong neighbors in the Dirichlet tesselation, *i.e.*, points whose tiles share a common edge. The solid lines in Figure 34.1 denote the corresponding Delaunay triangulation of the 25 Halton points.

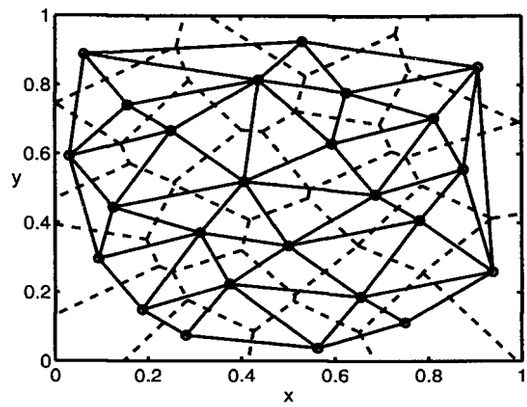


Fig. 34.1 Dirichlet tesselation (dashed lines) and corresponding Delaunay triangulation (solid lines) of 25 Halton points (circles).

(2) Discretize the Laplacian on this triangulation. In order to also take into account the boundary points Dyn, Levin and Rippa instead use a discretization of an iterated Green's formula which has the space Π^2_{m-1} as its null space. The necessary partial derivatives are then approximated on the triangulation using certain sets of vertices of the triangulation. (three points for first order partials, six for second order).

Figure 34.1 was created in MATLAB using the commands

```
load('Data2D_25h')
tes = delaunayn(dsites);
triplot(tes,dsites(:,1),dsites(:,2),'k-')
hold on
[vx, vy] = voronoi(dsites(:,1),dsites(:,2),tes);
plot(dsites(:,1),dsites(:,2),'ko',vx,vy,'k--')
axis([0 1 0 1])
```

As in our other MATLAB examples, the file `Data2D_25h` contains the coordinates of the 25 Halton points in the array `dsites`.

The discretization described above yields the matrix $B = (b_{ji})_{j,i=1}^N$ as the preconditioning matrix in a way analogous to the previous section. We now obtain

$$(BA)_{jk} = \sum_{i=1}^{N} b_{ji}\varphi(\|\boldsymbol{x}_i - \boldsymbol{x}_k\|) \approx \Delta^m \varphi(\|\cdot - \boldsymbol{x}_k\|)(\boldsymbol{x}_j), \quad j,k = 1,\ldots,N. \tag{34.2}$$

This matrix has the property that the entries close to the diagonal are large compared to those away from the diagonal, which decay to zero as the distance between the two points involved goes to infinity. Since the construction of B (in step 2 above) ensures that part of the preconditioned block matrix vanishes, namely $BP = O$,

one must now solve the non-square system

$$\begin{bmatrix} BA \\ P^T \end{bmatrix} \boldsymbol{c} = \begin{bmatrix} B\boldsymbol{y} \\ \boldsymbol{0} \end{bmatrix}.$$

Actually, the top part of the system, the square system $BA\boldsymbol{c} = B\boldsymbol{y}$, is singular, but it is shown in the paper [Dyn *et al.* (1986)] that the additional constraints $P^T\boldsymbol{c} = 0$ guarantee existence of a unique solution. Furthermore, the coefficients $\boldsymbol{d}$ in the original expansion of the interpolant $\mathcal{P}_f$ can be obtained by solving

$$P\boldsymbol{d} = \boldsymbol{y} - A\boldsymbol{c},$$

i.e., by fitting the polynomial part of the expansion to the residual $\boldsymbol{y} - A\boldsymbol{c}$.

The approach just described leads to localized basis functions Ψ that are linear combinations of the original basis functions φ. More precisely,

$$\Psi_j(\boldsymbol{x}) = \sum_{i=1}^{N} b_{ji}\varphi(\|\boldsymbol{x} - \boldsymbol{x}_i\|) \approx \Delta^m\varphi(\|\cdot - \boldsymbol{x}_j\|)(\boldsymbol{x}), \tag{34.3}$$

where the coefficients b_{ji} are determined via the discretization described above.

The localized basis functions Ψ_j, $j = 1, \ldots, N$, (see (34.3)) can be viewed as an alternative (better conditioned) basis for the approximation space spanned by the functions $\Phi_j = \varphi(\|\cdot - \boldsymbol{x}_j\|)$. We will come back to this idea in Section 34.4.

In [Dyn *et al.* (1986)] the authors describe how the preconditioned matrices can be used efficiently in conjunction with various iterative schemes such as Chebyshev iteration or a version of the conjugate gradient method. The authors also mention smoothing of noisy data, or low-pass filtering as other applications for this preconditioning scheme.

The effectiveness of the above preconditioning strategy was illustrated with some numerical examples in [Dyn *et al.* (1986)]. We list some of their results in Table 34.1. Thin plate splines and multiquadrics were tested on two different data sets (grid I and grid II) in $\mathbb{R}^2$. The shape parameter ε for the multiquadrics was chosen to be the reciprocal of the average mesh size. A linear term was added for thin plate splines, and a constant for multiquadrics.

Table 34.1 Condition numbers without and with preconditioning.

φ	N	grid I orig.	grid I precond.	grid II orig.	grid II precond.
TPS	49	1181	4.3	1885	3.4
	121	6764	5.1	12633	3.9
MQ	49	7274	69.2	17059	222.8
	121	10556	126.0	107333	576.0

One can see that the most dramatic improvement is achieved for thin plate splines. This is to be expected since the method described above is tailored to these functions. As noted earlier, for multiquadrics an application of the Laplacian does

not yield the delta function, but for values of r close to zero gives just relatively large values.

Another early preconditioning strategy was suggested in [Powell (1994a)]. Powell uses Householder transformations to convert the matrix of the interpolation system (34.1) to a symmetric positive definite matrix, and then uses the conjugate gradient method. However, Powell reports that this method is not particularly effective for large thin plate spline interpolation problems in $\mathbb{R}^2$.

In [Baxter (1992a); Baxter (2002)] preconditioned conjugate gradient methods for solving the interpolation problem are discussed in the case when Gaussians or multiquadrics are used on a regular grid. The resulting matrices are Toeplitz matrices, and a large body of literature exists for dealing with matrices having this special structure (see, *e.g.*, [Chan and Strang (1989)]).

34.3 Preconditioned GMRES via Approximate Cardinal Functions

More recently, Beatson, Cherrie and Mouat [Beatson *et al.* (1999)] proposed a preconditioner for the iterative solution of radial basis function interpolation systems in conjunction with the GMRES method of [Saad and Schultz (1986)]. The GMRES method is a general purpose iterative solver that can be applied to nonsymmetric (nondefinite) systems. For fast convergence the matrix should be preconditioned such that its eigenvalues are clustered around one and away from the origin. Obviously, if the basis functions for the radial basis function space were cardinal functions, then the matrix would be the identity matrix with all its eigenvalues equal to one. Therefore, the GMRES method would converge in a single iteration. Consequently, the preconditioning strategy employed by the authors of [Beatson *et al.* (1999)] for the GMRES method is to obtain a preconditioning matrix B that is close to the inverse of A.

Since it is too expensive to find the true cardinal basis (this would involve at least as much work as solving the interpolation problem), the idea pursued in [Beatson *et al.* (1999)] (and suggested earlier in [Beatson *et al.* (1996); Beatson and Powell (1993)]) is to find *approximate* cardinal functions similar to the functions Ψ_j in the previous subsection. Now, however, there is also an emphasis on efficiency, *i.e.*, we are interested in *local* approximate cardinal functions, if possible (*c.f.* also the use of approximate cardinal functions in the Faul-Powell algorithm of Section 33.2). Several different strategies for the construction of these approximate cardinal functions were suggested in [Beatson *et al.* (1999)]. We will now explain the basic idea.

Given the centers $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ for the basis functions in the RBF interpolant

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \varphi(\|\boldsymbol{x} - \boldsymbol{x}_j),$$

the j-th approximate cardinal function is given as a linear combination of the basis

functions $\Phi_i = \varphi(\|\cdot - \boldsymbol{x}_i\|)$, where i runs over (some subset of) $\{1, \ldots, N\}$, *i.e.*,

$$\Psi_j = \sum_{i=1}^{N} b_{ji}\varphi(\|\cdot - \boldsymbol{x}_i\|) + p_j. \tag{34.4}$$

Here p_j is a polynomial in Π_{m-1}^s that is used only in the conditionally positive definite case, and the coefficients b_{ji} satisfy the usual conditions

$$\sum_{i=1}^{N} b_{ji}p_j(\boldsymbol{x}_i) = 0 \qquad \text{for all } p_j \in \Pi_{m-1}^s. \tag{34.5}$$

The key feature in designing the approximate cardinal functions is to have only a few $n \ll N$ coefficients in (34.4) to be nonzero. In that case the functions Ψ_j are found by solving small $n \times n$ linear systems, which is much more efficient than dealing with the original $N \times N$ system. For example, in [Beatson *et al.* (1999)] the authors use $n \approx 50$ for problems involving up to 10000 centers. The resulting preconditioned system is of the same form as the earlier preconditioner (34.2), *i.e.*, we now have to solve the preconditioned problem

$$(BA)\boldsymbol{c} = B\boldsymbol{y},$$

where the entries of the matrix BA are just $\Psi_j(\boldsymbol{x}_k)$, $j, k = 1, \ldots, N$.

The simplest strategy for determining the coefficients b_{ji} is to select the n nearest neighbors of $\boldsymbol{x}_j$, and to find b_{ji} by solving the (local) cardinal interpolation problem

$$\Psi_j(\boldsymbol{x}_i) = \delta_{ij}, \qquad i = 1, \ldots, n,$$

subject to the moment constraint (34.5) listed above. Here δ_{ij} is the Kronecker-delta, so that Ψ_j is one at $\boldsymbol{x}_j$ and zero at all of the neighboring centers $\boldsymbol{x}_i$.

This basic strategy is improved by adding so-called *special points* that are distributed (very sparsely) throughout the domain (for example near corners of the domain, or at other significant locations).

A few numerical results for thin plate spline and multiquadric interpolation in $\mathbb{R}^2$ from [Beatson *et al.* (1999)] are listed in Table 34.2. The condition numbers are ℓ_2-condition numbers, and the points were randomly distributed in the unit square. The "local precond." column uses the $n = 50$ nearest neighbors to determine the approximate cardinal functions, whereas the right-most column uses the 41 nearest neighbors plus nine special points placed uniformly in the unit square. The effect of the preconditioning on the performance of the GMRES algorithm is, *e.g.*, a reduction from 103 to 8 iterations for the 289 point data set for thin plate splines, or from 145 iterations to 11 for multiquadrics.

An extension of the ideas of [Beatson *et al.* (1999)] to linear systems arising in the collocation solution of partial differential equations (see Chapter 38) was explored in Mouat's Ph.D. thesis [Mouat (2001)] and also in the recent paper [Ling and Kansa (2005)].

Table 34.2 Condition numbers without and with preconditioning.

φ	N	unprecond.	local precond.	local precond. w/special
TPS	289	4.005e+006	1.464e+003	5.721e+000
	1089	2.753e+008	6.359e+005	1.818e+002
	4225	2.605e+009	2.381e+006	1.040e+006
MQ	289	1.506e+008	3.185e+003	2.639e+002
	1089	2.154e+009	8.125e+005	5.234e+004
	4225	3.734e+010	1.390e+007	4.071e+004

34.4 Change of Basis

As pointed out at the beginning of this chapter, another approach to obtaining a better conditioned interpolation system is to work with a different basis for the approximation space. While this idea is implicitly addressed in the preconditioning strategies discussed above, we will now make it our primary goal to find a better conditioned basis for the RBF approximation space. Univariate piecewise linear splines and natural cubic splines can be interpreted as radial basis functions, and we know that B-splines form stable bases for those spaces. Therefore, it should be possible to generalize this idea for other RBFs.

The process of finding a "better" basis for conditionally positive definite radial basis functions is closely connected to finding the reproducing kernel of the associated native space. Since we did not elaborate on the construction of native spaces for conditionally positive definite functions earlier, we will now present the relevant formulas without going into any further details. In particular, for polyharmonic splines we will be able to find a basis that is in a certain sense *homogeneous*, and therefore the condition number of the related interpolation matrix will depend only on the number N of data points, but *not* on their separation distance (*c.f.* the discussion in Chapter 16). This approach was suggested by Beatson, Light and Billings [Beatson *et al.* (2000)], and has its roots in [Sibson and Stone (1991)].

Let Φ be a strictly conditionally positive definite kernel of order m, and $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \Omega \subset \mathbb{R}^s$ be an $(m-1)$-unisolvent set of centers. Then the reproducing kernel for the native space $\mathcal{N}_\Phi(\Omega)$ is given by

$$K(\boldsymbol{x}, \boldsymbol{y}) = \Phi(\boldsymbol{x}, \boldsymbol{y}) - \sum_{k=1}^{M} p_k(\boldsymbol{x})\Phi(\boldsymbol{x}_k, \boldsymbol{y}) - \sum_{\ell=1}^{M} p_\ell(\boldsymbol{y})\Phi(\boldsymbol{x}, \boldsymbol{x}_\ell) + \sum_{k=1}^{M}\sum_{\ell=1}^{M} p_k(\boldsymbol{x})p_\ell(\boldsymbol{y})\Phi(\boldsymbol{x}_k, \boldsymbol{x}_\ell) + \sum_{\ell=1}^{M} p_\ell(\boldsymbol{x})p_\ell(\boldsymbol{y}), \tag{34.6}$$

where the points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}$ comprise an $(m-1)$-unisolvent subset of $\mathcal{X}$ and the polynomials p_k, $k = 1, \ldots, M$, form a *cardinal* basis for Π^s_{m-1} on this subset whose dimension is $M = \binom{s+m-1}{m-1}$, *i.e.*,

$$p_\ell(\boldsymbol{x}_k) = \delta_{k,\ell}, \qquad k, \ell = 1, \ldots, M.$$

This formulation of the reproducing kernel for the conditionally positive definite case also appears in the statistics literature in the context of *kriging* (see, *e.g.*, [Berlinet and Thomas-Agnan (2004)]). In that context the kernel K is a covariance kernel associated with the generalized covariance Φ. These two kernels give rise to the kriging equations and dual kriging equations, respectively.

An immediate consequence of having found the reproducing kernel K is that we can express the radial basis function interpolant to values of some function f given on $\mathcal{X}$ in the form

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{j=1}^{N} \mathrm{c}_j K(\boldsymbol{x}, \boldsymbol{x}_j), \qquad \boldsymbol{x} \in \mathbb{R}^s.$$

Note that the kernel K used here is a strictly positive definite kernel (since it is a reproducing kernel) with built-in polynomial precision. The coefficients c_j are determined by satisfying the interpolation conditions

$$\mathcal{P}_f(\boldsymbol{x}_i) = f(\boldsymbol{x}_i), \qquad i = 1, \ldots, N.$$

We will see below (in Tables 34.3 and 34.4) that this basis already performs "better" (*i.e.*, is better conditioned) than the standard basis $\{\Phi(\cdot, \boldsymbol{x}_1), \ldots, \Phi(\cdot, \boldsymbol{x}_N)\}$ if we keep the number of centers fixed, and vary only their separation distance.

To obtain the homogeneous basis referred to above we modify K by subtracting the tensor product polynomial, *i.e.*,

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = K(\boldsymbol{x}, \boldsymbol{y}) - \sum_{\ell=1}^{M} p_\ell(\boldsymbol{x}) p_\ell(\boldsymbol{y}).$$

Now, if $\boldsymbol{y}$ denotes any one of the points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$ in the $(m-1)$-unisolvent subset of $\mathcal{X}$ used in the construction of K above, then we have

$$\begin{aligned}
\kappa(\cdot, \boldsymbol{y}) &= \Phi(\cdot, \boldsymbol{y}) - \sum_{k=1}^{M} p_k(\cdot)\Phi(\boldsymbol{x}_k, \boldsymbol{y}) - \sum_{\ell=1}^{M} p_\ell(\boldsymbol{y})\Phi(\cdot, \boldsymbol{x}_\ell) \\
&\quad + \sum_{k=1}^{M}\sum_{\ell=1}^{M} p_k(\cdot) p_\ell(\boldsymbol{y}) \Phi(\boldsymbol{x}_k, \boldsymbol{x}_\ell) \\
&= \Phi(\cdot, \boldsymbol{y}) - \sum_{k=1}^{M} p_k(\cdot)\Phi(\boldsymbol{x}_k, \boldsymbol{y}) - \Phi(\cdot, \boldsymbol{y}) + \sum_{k=1}^{M} p_k(\cdot)\Phi(\boldsymbol{x}_k, \boldsymbol{y}) = 0
\end{aligned}$$

since the polynomials p_k are cardinal on $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$, *i.e.*, only one of the $p_k(y)$ will "survive".

This means that the functions $\kappa(\cdot, \boldsymbol{x}_j)$, $j = 1, \ldots, N$, cannot be used as a basis of our approximation space. Instead we need to remove the points used to define the cardinal polynomials above from the set of centers used for κ. Once we do this it turns out that the matrix C with entries $C_{i,j} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $i, j = M+1, \ldots, N$, is positive definite, and therefore we obtain the following basis

$$\{p_1, \ldots, p_M\} \cup \{\kappa(\cdot, \boldsymbol{x}_{M+1}), \ldots, \kappa(\cdot, \boldsymbol{x}_N)\}$$

for the space $\operatorname{span}\{\Phi(\cdot, \boldsymbol{x}_1), \ldots, \Phi(\cdot, \boldsymbol{x}_N)\}$. Therefore the interpolant can be represented in the form

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{j=1}^{M} d_j p_j(\boldsymbol{x}) + \sum_{k=M+1}^{N} c_k \kappa(\boldsymbol{x}, \boldsymbol{x}_k), \qquad \boldsymbol{x} \in \mathbb{R}^s. \tag{34.7}$$

The coefficients are determined as usual by enforcing the interpolation conditions $\mathcal{P}_f(\boldsymbol{x}_i) = f(\boldsymbol{x}_i)$, $i = 1, \ldots, N$. Since the polynomials p_j are cardinal on $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}$ and κ was shown to be zero if centered at these points, this leads to the following linear system

$$\begin{bmatrix} I & O \\ P^T & C \end{bmatrix} \begin{bmatrix} \boldsymbol{d} \\ \boldsymbol{c} \end{bmatrix} = \begin{bmatrix} \boldsymbol{y}_p \\ \boldsymbol{y}_\kappa \end{bmatrix}, \tag{34.8}$$

with I an $M \times M$ identity matrix, O an $M \times (N - M)$ zero matrix, C as above, $P_{ij} = p_j(\boldsymbol{x}_i)$, $j = 1, \ldots, M$, $i = M + 1, \ldots, N$, $\boldsymbol{c} = [c_{M+1}, \ldots, c_N]^T$, $\boldsymbol{d} = [d_1, \ldots, d_M]^T$, and the right-hand side vectors $\boldsymbol{y}_p = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_M)]^T$ and $\boldsymbol{y}_\kappa = [f(\boldsymbol{x}_{M+1}), \ldots, f(\boldsymbol{x}_N)]^T$. The identity block (cardinality of the polynomial basis functions) implies that the coefficient vector $\boldsymbol{d}$ is given by

$$d_j = f(\boldsymbol{x}_j), \qquad j = 1, \ldots, M,$$

and therefore the system (34.8) can be solved as

$$C\boldsymbol{c} = \boldsymbol{y}_\kappa - P^T \boldsymbol{d}. \tag{34.9}$$

As mentioned above, one can show that the matrix C is symmetric and positive definite.

Most importantly, for polyharmonic splines, the ℓ_2-condition number of the matrix C is invariant under a uniform scaling of the centers, *i.e.*, if $C^h = (\kappa(h\boldsymbol{x}_i, h\boldsymbol{x}_j))$, then

$$\operatorname{cond}(C^h) = \operatorname{cond}(C).$$

This is proved to varying degrees of detail in the papers [Beatson *et al.* (2000); Iske (2003a)] and the book [Wendland (2005a)].

Example 34.3. The simplest example is given by the polyharmonic spline $\varphi(r) = r$. In this case $M = 1$ so that the only polynomial term is given by the constant $p \equiv 1$. For simplicity we use the origin as a special point. Using these conventions we have the following three representations of the various kernels:

$$\begin{aligned} \Phi(\boldsymbol{x}, \boldsymbol{y}) &= \|\boldsymbol{x} - \boldsymbol{y}\|, \\ K(\boldsymbol{x}, \boldsymbol{y}) &= \|\boldsymbol{x} - \boldsymbol{y}\| - \|\boldsymbol{y}\| - \|\boldsymbol{x}\| + 1, \\ \kappa(\boldsymbol{x}, \boldsymbol{y}) &= \|\boldsymbol{x} - \boldsymbol{y}\| - \|\boldsymbol{y}\| - \|\boldsymbol{x}\|. \end{aligned}$$

Note that in this case the condition number of the matrix C associated with the kernel κ is clearly invariant under uniform scaling of the problem. However, the matrix A associated with the basic norm RBF Φ enjoys the same invariance. It is only when we add the polynomial blocks P and P^T to ensure reproduction of constants that the condition number of the resulting block matrix will vary greatly with the problem scaling. A similar dependence of the condition number of the system matrix on the scaling is associated with the kernel K.

34.5 Effect of the "Better" Basis on the Condition Number of the Interpolation Matrix

We reproduce some numerical experiments from [Beatson *et al.* (2000)] based on the use of thin plate splines in $\mathbb{R}^2$. We compute the ℓ_2-condition numbers of the interpolation matrix for the three different approaches mentioned above, *i.e.*, using the standard basis consisting of functions $\Phi(\cdot, \boldsymbol{x}_j)$ and monomials (obtained with the MATLAB program `RBFInterpolation2Dlinear.m` of Chapter 6), using the reproducing kernels $K(\cdot, \boldsymbol{x}_j)$, and using the matrix C based on the kernel κ. The matrix for the kernels $K(\cdot, \boldsymbol{x}_j)$ is computed with the program `tpsK.m` provided in Program 34.1. The three polynomial cardinal functions are based on the three corners $(0,0)$, $(0,1)$, and $(1,0)$ of the unit square, *i.e.*,

$$\begin{aligned} p_1(\boldsymbol{z}) &= 1 - z_1 - z_2, \\ p_2(\boldsymbol{z}) &= z_1, \\ p_3(\boldsymbol{z}) &= z_2, \end{aligned}$$

where $\boldsymbol{z} = (z_1, z_2) \in \mathbb{R}^2$.

The program `tpsK.m` is completely vectorized, *i.e.*, we input arrays of points `x` and `y`, and create the entire matrix with entries $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $i, j = 1, \ldots, N$ (denoted by `rbf` in the program). We assemble the matrix according to the terms in (34.6). On lines 3 and 4 we fill two matrices, `px` and `py`, whose columns contain the values of the polynomials p_1, p_2 and p_3 (defined as separate functions at the end of the program) at all of the points in `x` and `y`, respectively. The first term of (34.6), the matrix $\Phi(\boldsymbol{x}, \boldsymbol{y})$, is assembled on line 5 where we call the subroutine `tps.m` listed as Program C.4 in Appendix C. Next, on lines 6–11 we add the next two sums from (34.6) simultaneously. The double sum is added to the matrix `rbf` on lines 12–17, and finally the tenor product polynomial term is computed and added on lines 18–20.

Program 34.1. `tpsK.m`

```
% rbf = tpsK(x,y)
% Computes matrix for thin plate spline kernel K with
% linear polynomials cardinal on (0,0), (1,0), (0,1)
% Calls on: tps
 1  function rbf = tpsK(x,y)
    % Define points for cardinal polynomials
 2  ppoints = [0 0; 1 0; 0 1];
 3  px = [p1(x) p2(x) p3(x)];
 4  py = [p1(y) p2(y) p3(y)];
 5  r = DistanceMatrix(x,y); rbf = tps(1,r);
 6  for k=1:3
 7     r = DistanceMatrix(ppoints(k,:),y);
```

```
 8     rbf = rbf - px(:,k)*tps(1,r);
 9     r = DistanceMatrix(x,ppoints(k,:));
10     rbf = rbf - tps(1,r)*py(:,k)';
11  end
12  for j=1:3
13     for k=1:3
14        r = DistanceMatrix(ppoints(j,:),ppoints(k,:));
15        rbf = rbf + px(:,j)*py(:,k)'*tps(1,r);
16     end
17  end
18  for k=1:3
19     rbf = rbf + px(:,k)*py(:,k)';
20  end
21  return
    % The cardinal polynomials
22  function w = p1(z)
23  w = 1 - z(:,1) - z(:,2);
24  return
25  function w = p2(z)
26  w = z(:,1);
27  return
28  function w = p3(z)
29  w = z(:,2);
30  return
```

Since Program 34.1 produces the entire interpolation (or evaluation) matrix, we can use Program 2.1 and replace lines 13 and 14 by

```
IM = tpsK(dsites,ctrs);
```

and lines 15 and 16 by

```
EM = tpsK(epoints,ctrs);
```

in order to solve the interpolation problem in this case.

The matrix C is obtained in a similar fashion by using a program `tpsH.m` that is identical to `tpsK.m` except that lines 18–20 are removed. In addition, we need to remove the corner points $(0,0)$, $(1,0)$, and $(0,1)$ from the `ctrs` and `dsites` in the driver program.

In the first experiment (illustrated in Table 34.3) the problem is formulated on the unit square $[0,1]^2$. Here both the number of points and the separation distance vary from one row in the table to the next. The three different columns list the ℓ_2-condition numbers of the interpolation matrix for the three different approaches mentioned above. With this setup all three methods perform comparably.

Table 34.3 Condition numbers for different thin plate spline bases on $[0,1]^2$ with increasing number of points and varying separation distance.

spacing h	standard matrix	reproducing kernel	homogeneous matrix
1/8	3.515800e+003	1.839030e+004	7.583833e+003
1/16	3.893850e+004	2.651373e+005	1.108581e+005
1/32	5.136252e+005	4.000679e+006	1.686431e+006
1/64	7.618277e+006	6.202918e+007	2.626402e+007

In the second experiment (shown in Table 34.4) the number of points is kept fixed at 5×5 equally spaced points. However, the domain is scaled to the square $[0,a]^2$ with scale parameter a (this can easily be done using the same programs as above by introducing a scale parameter at the appropriate places, see also Program 34.2). The effect of this is that only the separation distance $q_{\mathcal{X}}$ changes from one row to the next in the table. Now, clearly, the two new methods show less dependence on the separation distance, with the condition number of the homogeneous matrix C being completely insensitive to the re-scaling as claimed earlier.

Table 34.4 Condition numbers for different thin plate spline bases on $[0,a]^2$ with fixed number of 25 points and varying separation distance.

scale parameter a	standard matrix	reproducing kernel	homogeneous matrix
0.001	2.434883e+008	8.463509e+008	5.493771e+002
0.01	2.436378e+006	8.464002e+006	5.493771e+002
0.1	2.517866e+004	8.513354e+004	5.493771e+002
1.0	3.645782e+002	1.366035e+003	5.493771e+002
10	1.874215e+006	1.260864e+003	5.493771e+002
100	1.151990e+011	1.139634e+005	5.493771e+002
1000	3.548239e+015	1.138572e+007	5.493771e+002

We close this section by pointing out that Iske an co-workers take advantage of the scale invariance of polyharmonic splines (and thin plate splines in particular) in the construction of a numerical multiscale solver for transport problems (see, *e.g.*, [Behrens *et al.* (2002)]).

34.6 Effect of the "Better" Basis on the Accuracy of the Interpolant

In this section we provide an example illustrating the surprising fact that for polyharmonic splines not only the homogeneous kernel κ can be used successfully for poorly scaled problems, but also the standard kernel Φ.

Example 34.4. We use the thin plate spline basic function $\varphi(r) = r^2 \log r$ and a scaled version of Franke's testfunction to generate test data on a 5×5 uniform

grid in the square $[0, a]^2$ as in Table 34.4. However, now the scale parameter a will range from 10^{-9} to 10^9. We will present condition numbers and root-mean-square errors computed on a 40×40 uniform grid for the three different kernels discussed previously. We list only the MATLAB code for the homogeneous case since the two other programs are very similar to previous ones. The function `tpsH.m` called by Program 34.2 is almost the same as Program 34.1 listed above. The required modifications are noted there.

Many parts of Program 34.2 are familiar. However, in order to deal with the kernel κ and the associated matrix C we need to define the special points at which the cardinal polynomials are defined. This is done on line 9, where the special points are taken as three corners of the scaled unit square. Since the kernel is the zero function when centered at these points they need to be removed from the set of centers. This is accomplished on lines 11, 12 and 14. The scaling of the problem happens on lines 8, 9 and 13. The scale also enters in a number of other places such as the definition of the evaluation grid on line 15, computation of the right-hand side on lines 17–19, and the computation of the exact solution on line 26. In contrast to most of our other interpolation programs here we compute the interpolation and evaluation matrices with a single subroutine (*c.f.* the calls to `tpsH` on lines 20 and 22). Note that the scale parameter a is passed to `tpsH`. Equations 34.9 and 34.7 for the solution and evaluation of the interpolant are implemented together on line 25. Finally, the three cardinal polynomials are coded on lines 35–43. Since these polynomials are defined on the unit square they need to be called with re-scaled arguments (cf. lines 17 and 23).

Program 34.2. `RBFInterpolation2DtpsH.m`

```
% RBFInterpolation2DtpsH
% Script that performs 2D TPS interpolation with homogeneous kernel
% Calls on: tpsH
 1  function RBFInterpolation2DtpsH
    % Define Franke's function as testfunction
 2  f1 = @(x,y) 0.75*exp(-((9*x-2).^2+(9*y-2).^2)/4);
 3  f2 = @(x,y) 0.75*exp(-((9*x+1).^2/49+(9*y+1).^2/10));
 4  f3 = @(x,y) 0.5*exp(-((9*x-7).^2+(9*y-3).^2)/4);
 5  f4 = @(x,y) 0.2*exp(-((9*x-4).^2+(9*y-7).^2));
 6  testfunction = @(x,y) f1(x,y)+f2(x,y)+f3(x,y)-f4(x,y);
 7  N = 25; gridtype = 'u';
 8  a = 1e9;
 9  ppoints = a*[0 0; 1 0; 0 1];
    % Load data points
10  name = sprintf('Data2D_%d%s',N,gridtype); load(name)
    % Remove (0,0), (1,0), (0,1) to work with C matrix
11a remove = [find(dsites(:,1)==0 & dsites(:,2)==0);...
```

```
1ib       find(dsites(:,1)==1 & dsites(:,2)==0);...
11c       find(dsites(:,1)==0 & dsites(:,2)==1)]
12  dsites(remove,:) = [];
    % Scale problem to square [0,a]^2
13  dsites = a*dsites;
    % Let centers coincide with data sites
14  ctrs=dsites;
15  neval = 40; grid = linspace(0,a,neval);
16  [xe,ye] = meshgrid(grid); epoints = [xe(:) ye(:)];
    % Create right-hand side for homogeneous problem
17  DP = [p1(dsites/a) p2(dsites/a) p3(dsites/a)]';
18  d = testfunction(ppoints(:,1)/a,ppoints(:,2)/a);
19  rhs = testfunction(dsites(:,1)/a,dsites(:,2)/a) - DP'*d;
    % Compute interpolation matrix for the special case of TPS
    % native space kernel (no need to add polynomials)
20  IM = tpsH(dsites,ctrs,a);
    % Compute condition number of interpolation matrix
21  fprintf('l2-condition          : %e\n', cond(IM))
    % Compute evaluation matrix
22  EM = tpsH(epoints,ctrs,a);
23  EP = [p1(epoints/a) p2(epoints/a) p3(epoints/a)];
24  EM = [EM EP];
    % Compute RBF interpolant
25  Pf = EM * [(IM\rhs); d];
    % Compute exact solution
26  exact = testfunction(epoints(:,1)/a,epoints(:,2)/a);
    % Compute errors on evaluation grid
27  maxerr = norm(Pf-exact,inf);
28  rms_err = norm(Pf-exact)/neval;
29  fprintf('RMS error:     %e\n', rms_err)
30  fprintf('Maximum error: %e\n', maxerr)
31  fview = [160,20];  % viewing angles for plot
32  PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview);
33  PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview);
34  return
    % The cardinal polynomials
35  function w = p1(z)
36  w = 1 - z(:,1) - z(:,2);
37  return
38  function w = p2(z)
39  w = z(:,1);
40  return
```

```
41 function w = p3(z)
42 w = z(:,2);
43 return
```

In Table 34.5 we list the root-mean-square errors resulting from the three different interpolation methods. The scaling of the domain was chosen more extreme than in Table 34.4 so that the sensitivity of the reproducing kernel K becomes clearly visible. Its condition number of the interpolation matrix for $a = 10^{-9}$ was 5.354134e+018, while for $a = 10^{9}$ it was 6.994062e+019. While both of these condition numbers are clearly very high and therefore indicate that we might expect numerical difficulties solving a problem on these length scales, the other two methods (standard TPS basis functions and the homogeneous kernel κ) perform perfectly throughout the entire range of scalings. Moreover, the condition numbers for the standard TPS interpolation matrix are much higher than for the K kernel: 7.299408e+021 for $a = 10^{-9}$, and even 2.749537e+038 for $a = 10^{9}$. Nevertheless, the standard TPS interpolant does *not* suffer from instability due to this ill-conditioning. The same is true for all other tests we have performed with standard polyharmonic spline interpolants (such as, *e.g.*, the norm basic function).

Table 34.5 RMS errors for different thin plate spline interpolants on $[0, a]^2$ with fixed number of 25 points and varying separation distance.

scale parameter o	standard matrix	reproducing kernel	homogeneous matrix
10^{-9}	2.969478e-002	NAN	2.969478e-002
10^{-6}	2.969478e-002	2.970740e-002	2.969478e-002
10^{-3}	2.969478e-002	2.969478e-002	2.969478e-002
1.0	2.969478e-002	2.969478e-002	2.969478e-002
10^{3}	2.969478e-002	2.969478e-002	2.969478e-002
10^{6}	2.969478e-002	2.969218e-002	2.969478e-002
10^{9}	2.969478e-002	1.207446e+003	2.969478e-002

Chapter 35

Other Efficient Numerical Methods

In earlier chapters we have already mentioned various algorithms for meshfree scattered data interpolation that are more efficient than the straightforward solution of the linear system obtained by enforcing the interpolation conditions. In particular, we suggested the use of the non-uniform fast Fourier transform (NFFT) for fast evaluation of globally supported functions, a fixed level iterative algorithm based on approximate MLS approximation, the greedy algorithm of Schaback and Wendland, the Faul-Powell algorithm, and the preconditioned GMRES method of Beatson et al.

We now add three more numerical techniques that can be used to make the computation with globally supported functions on large data sets more efficient and also more stable. In the first two sections of this chapter we discuss the fast multipole method and fast tree codes, and how these methods can be adapted to radial basis functions. In the third section we present a brief introduction to domain decomposition methods, which not only make the solution of large interpolation problems more efficient, but also provide a way to avoid the ill-conditioning issue by breaking the large problem into many well-conditioned smaller ones.

35.1 The Fast Multipole Method

Another technique for dealing with fast summation problems is known as the *fast multipole method.* It was first proposed by Greengard and Rokhlin in the late 1980s (see, *e.g.*, the original paper [Greengard and Rokhlin (1987)], the popular discussion [Greengard (1994)], or the short course tailored to radial basis functions [Beatson and Greengard (1997)]). This method has quickly become rather popular in the computational sciences. The breakthrough accomplishment of this algorithm was the ability to perform fast evaluations of sums of the type

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{k=1}^{N} c_k \Phi(\boldsymbol{x}, \boldsymbol{x}_k), \qquad \boldsymbol{x} \in \mathbb{R}^s.$$

In particular, M such evaluations can be performed in $\mathcal{O}(M \log N)$ (or even $\mathcal{O}(M)$) operations instead of the standard $\mathcal{O}(MN)$ operations for a naive implementation of the summation. The non-uniform fast Fourier transform of Chapter 28 was able to do this also, and in a fairly general way for a very large class of kernels Φ. However, the fast multipole method is a little older and it may be more efficient than the NFFT since special expansions are used that are chosen with the particular kernel Φ in mind.

We will now outline the basic idea of the *fast Gauss transform* [Greengard and Strain (1991)]. This transform can be applied directly to the approximate moving least squares approximands based on Gaussians used in earlier chapters (see the numerical experiments reported in Table 35.1 below). The higher-order Laguerre-Gaussian kernels, however, require a completely new derivation. Using our standard abbreviation $\varepsilon = 1/(\sqrt{\mathcal{D}}h)$, we are now interested in a fast summation technique for M simultaneous evaluations of the Gaussian quasi-interpolant (or *discrete Gauss transform*)

$$\mathcal{G}_f(\boldsymbol{y}_j) = \sum_{k=1}^{N} f(\boldsymbol{x}_k) e^{-\|\varepsilon(\boldsymbol{y}_j - \boldsymbol{x}_k)\|^2}, \qquad j = 1, \ldots, M. \tag{35.1}$$

In [Greengard and Strain (1991)] such an algorithm was developed, and in [Strain (1991)] a modification was suggested to cover also the case of variable scales ε_k as needed with quasi-interpolation at scattered sites or with variable shape parameters.

One of the central ingredients for the fast Gauss transform are the *multivariate Hermite functions* $h_{\boldsymbol{\alpha}}$ defined as

$$h_{\boldsymbol{\alpha}}(\boldsymbol{x}) = (-1)^{|\boldsymbol{\alpha}|} D^{\boldsymbol{\alpha}} e^{-\|\boldsymbol{x}\|^2}, \tag{35.2}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_s) \in \mathbb{N}^s$ is a multi-index. These functions are related to the multivariate Hermite polynomials $H_{\boldsymbol{\alpha}}$ via

$$H_{\boldsymbol{\alpha}}(\boldsymbol{x}) = \prod_{d=1}^{s} H_{\alpha_d}(x_d) = e^{\|\boldsymbol{x}\|^2} h_{\boldsymbol{\alpha}}(\boldsymbol{x}) \tag{35.3}$$

(see, *e.g.*, the univariate formula (6.1.3) in [Andrews *et al.* (1999)]). It is beneficial that the Hermite functions can be evaluated recursively using the (univariate) recurrence relation

$$\begin{aligned} h_{n+1}(x) &= 2x h_n(x) - 2n h_{n-1}(x), \qquad n = 1, 2, \ldots, \\ h_0(x) &= e^{-|x|^2}, \; h_1(x) = 2x e^{-|x|^2}, \end{aligned}$$

which follows immediately from (35.3) and the recursion relation for Hermite polynomials (see, *e.g.*, formula (6.1.10) in [Andrews *et al.* (1999)]).

The algorithm of Greengard and Strain is based on three basic expansions which we list below as Theorems 35.1–35.3 (see [Greengard and Strain (1991); Greengard and Sun (1998)]). The main effect of these expansions is the fact that the variables $\boldsymbol{y}_j$ and $\boldsymbol{x}_k$ will be separated. This is the fundamental "trick" used with all

fast summation algorithms (see our discussion of the NFFT based fast summation method in Chapter 28). This will allow for the pre-computation and storage of certain *moments* below.

The first step in the algorithm is to scale the problem to the unit box $[0,1]^s$ and then subdivide the unit box into smaller boxes B and C which usually coincide. They can, however, also differ. The boxes B contain the *sources* $\boldsymbol{x}_k$ (*i.e.*, our centers), and the boxes C the *targets* $\boldsymbol{y}_j$ (*i.e.*, our evaluation points). For each source box B one then determines its *interaction region* $IR(B)$. The interaction region of B is a set of nearest neighbors of B such that the error of truncating the sum over all boxes is below a certain threshold. Due to the fast decay of the Gaussians it is suggested (see [Greengard and Sun (1998)]) to use the 9^s nearest neighbors for single precision and the 13^s nearest neighbors for double precision.

Theorem 35.1. *Let I_B be the index set denoting the sources $\boldsymbol{x}_k$ that lie in a box B with center $\boldsymbol{x}_B$ and side length $1/\varepsilon$, and let $\boldsymbol{y}_C$ be the center of the target box $C \in IR(B)$ of radius r_c containing the targets $\boldsymbol{y}_j$. Then the Gaussian field due to the sources in B,*

$$\mathcal{G}_f^{(B)}(\boldsymbol{y}_j) = \sum_{k \in I_B} f(\boldsymbol{x}_k) e^{-\|\varepsilon(\boldsymbol{y}_j - \boldsymbol{x}_k)\|^2},$$

has the following Taylor expansion about $\boldsymbol{y}_C$:

$$\mathcal{G}_f^{(B)}(\boldsymbol{y}_j) = \sum_{\boldsymbol{\alpha} \geq 0} a_{\boldsymbol{\alpha}} \left(\varepsilon(\boldsymbol{y}_j - \boldsymbol{y}_C)\right)^{\boldsymbol{\alpha}}, \tag{35.4}$$

where the coefficients $a_{\boldsymbol{\alpha}}$ are given by

$$a_{\boldsymbol{\alpha}} = \frac{(-1)^{|\boldsymbol{\alpha}|}}{\boldsymbol{\alpha}!} \sum_{k \in I_B} f(\boldsymbol{x}_k) h_{\boldsymbol{\alpha}} \left(\varepsilon(\boldsymbol{x}_k - \boldsymbol{y}_C)\right).$$

The error $E_T(p)$ due to truncating the series (35.4) after the p-th order terms satisfies the bound

$$|E_T(p)| = \Big| \sum_{\boldsymbol{\alpha} > p} a_{\boldsymbol{\alpha}} \left(\varepsilon(\boldsymbol{y}_j - \boldsymbol{y}_C)\right)^{\boldsymbol{\alpha}} \Big| \leq (1.09)^s F^{(B)} \frac{1}{\sqrt{(p+1)!^s}} \left[\frac{(\varepsilon r_c)^{p+1}}{1 - \varepsilon r_c}\right]^s,$$

where $F^{(B)} = \sum_{k \in I_B} |f(\boldsymbol{x}_k)|$.

Here we used the multi-index notation $\boldsymbol{\alpha} \geq 0$ to denote the constraints $\alpha_d \geq 0$ for all $d = 1, \ldots, s$. More generally, for some integer p we say $\boldsymbol{\alpha} \geq p$ if $\alpha_d \geq p$ for all $d = 1, \ldots, s$. This implies that we have $\boldsymbol{\alpha} > p$ for some integer p, if $\boldsymbol{\alpha} \geq p$ and $\alpha_d > p$ for some d. We also use $\boldsymbol{\alpha} \geq \boldsymbol{\beta}$ if $\alpha_d \geq \beta_d$ for all $d = 1, \ldots, s$.

The expansion (35.4) will be used in the case when the source box B contains relatively few sources, but the target box C contains many targets.

By reversing the role of the Hermite functions and the shifted monomials one can write a single Gaussian as a multivariate Hermite expansion about a point $\boldsymbol{z}_0 \in \mathbb{R}^s$, *i.e.*,

$$e^{-\|\varepsilon(\boldsymbol{y}_j - \boldsymbol{x}_k)\|^2} = \sum_{\boldsymbol{\alpha} \geq 0} \frac{1}{\boldsymbol{\alpha}!} \left(\varepsilon(\boldsymbol{x}_k - \boldsymbol{z}_0)\right)^{\boldsymbol{\alpha}} h_{\boldsymbol{\alpha}} \left(\varepsilon(\boldsymbol{y}_j - \boldsymbol{z}_0)\right). \tag{35.5}$$

This idea is used in

Theorem 35.2 (Far-field expansion). *Let I_B be the index set denoting the sources $\boldsymbol{x}_k$ that lie in a box B with center $\boldsymbol{x}_B$ and side length $1/\varepsilon$. Then the Gaussian field due to the sources in B,*

$$\mathcal{G}_f^{(B)}(\boldsymbol{y}_j) = \sum_{k \in I_B} f(\boldsymbol{x}_k) e^{-\|\varepsilon(\boldsymbol{y}_j - \boldsymbol{x}_k)\|^2},$$

is equal to an Hermite expansion about $\boldsymbol{x}_B$:

$$\mathcal{G}_f^{(B)}(\boldsymbol{y}_j) = \sum_{\boldsymbol{\alpha} \geq 0} b_{\boldsymbol{\alpha}} h_{\boldsymbol{\alpha}} \left(\varepsilon(\boldsymbol{y}_j - \boldsymbol{x}_B)\right). \tag{35.6}$$

The moments $b_{\boldsymbol{\alpha}}$ are given by

$$b_{\boldsymbol{\alpha}} = \frac{1}{\boldsymbol{\alpha}!} \sum_{k \in I_B} f(\boldsymbol{x}_k) \left(\varepsilon(\boldsymbol{x}_k - \boldsymbol{x}_B)\right)^{\boldsymbol{\alpha}}.$$

The error $E_H(p)$ due to truncating the series (35.6) after p-th order terms satisfies the bound

$$|E_H(p)| = |\sum_{\boldsymbol{\alpha} > p} b_{\boldsymbol{\alpha}} h_{\boldsymbol{\alpha}} \left(\varepsilon(\boldsymbol{y}_j - \boldsymbol{x}_B)\right)| \leq (1.09)^s F^{(B)} \frac{1}{\sqrt{(p+1)!}^s} \left[\frac{(\varepsilon r_c)^{p+1}}{1 - \varepsilon r_c}\right]^s.$$

Theorem 35.2 is used when B contains many sources, but C only few targets. Finally, in the case when both B and C contain relatively many points we use

Theorem 35.3 (Translation operation). *Let the sources $\boldsymbol{x}_k$ lie in a box B with center $\boldsymbol{x}_B$ and side length $1/\varepsilon$ and let $\boldsymbol{y}_j$ be an evaluation point in a box C with center $\boldsymbol{y}_C$. Then the corresponding truncated Hermite expansion (35.6) can be expanded as a Taylor series of the form*

$$\mathcal{G}_f^{(BC)}(\boldsymbol{y}_j) = \sum_{\boldsymbol{\beta} \geq 0} c_{\boldsymbol{\beta}} \left(\varepsilon(\boldsymbol{y}_j - \boldsymbol{y}_C)\right)^{\boldsymbol{\beta}}. \tag{35.7}$$

The coefficients $c_{\boldsymbol{\beta}}$ are given by

$$c_{\boldsymbol{\beta}} = \frac{(-1)^{|\boldsymbol{\beta}|}}{\boldsymbol{\beta}!} \sum_{\boldsymbol{\alpha} \leq p} b_{\boldsymbol{\alpha}} h_{\boldsymbol{\alpha}+\boldsymbol{\beta}} \left(\varepsilon(\boldsymbol{x}_B - \boldsymbol{y}_C)\right),$$

with $b_{\boldsymbol{\alpha}}$ as in Theorem 35.2. The error $E_T(p)$ due to truncating the series (35.7) after p-th order terms satisfies the bound

$$|E_T(p)| = |\sum_{\boldsymbol{\beta} > p} b_{\boldsymbol{\beta}} \left(\varepsilon(\boldsymbol{x} - \boldsymbol{y}_C)\right)^{\boldsymbol{\beta}}| \leq (1.09)^s F^{(B)} \frac{1}{\sqrt{(p+1)!}^s} \left[\frac{(\varepsilon r_c)^{p+1}}{1 - \varepsilon r_c}\right]^s.$$

Theorem 35.3 is based on the multivariate Taylor series expansion of the Hermite functions $h_{\boldsymbol{\alpha}}$, *i.e.*,

$$h_{\boldsymbol{\alpha}} \left(\varepsilon(\boldsymbol{y}_j - \boldsymbol{x}_B)\right) = \sum_{\boldsymbol{\beta} \geq 0} \frac{(-1)^{|\boldsymbol{\beta}|}}{\boldsymbol{\beta}!} \left(\varepsilon(\boldsymbol{y}_j - \boldsymbol{y}_C)\right)^{\boldsymbol{\beta}} h_{\boldsymbol{\alpha}+\boldsymbol{\beta}} \left(\varepsilon(\boldsymbol{x}_B - \boldsymbol{y}_C)\right).$$

Note that the error estimates in the original paper on the fast Gauss transform [Greengard and Strain (1991)] were incorrect. In the mean time a number of other authors have provided alternate error bounds in their papers (see, *e.g.*, [Baxter and Roussos (2002); Florence and van Loan (2000); Greengard and Sun (1998); Wendland (2004)]).

For ID calculations on the order of $p = 20$ terms are required to achieve double precision accuracy. For the 2D case one can get by with a smaller value of p (about 15), but the number of terms is of course much higher (on the order of p^s for s-dimensional problems).

The basic outline of the algorithm is as follows:

Algorithm 35.1. Fast Gauss transform

(1) If necessary, scale the problem so that the coarsest box $B_0 = [0,1]^s$. Subdivide B_0 into smaller boxes with side length $1/\varepsilon$ parallel to the axes. Assign each source $\boldsymbol{x}_k$ to the box B in which it lies and each evaluation point $\boldsymbol{y}_j$ to the box C in which it lies.

(2) Choose p so that the truncation error satisfies the desired accuracy, and for each box B compute and store the coefficients (or moments)

$$b_{\boldsymbol{\alpha}} = \frac{1}{\boldsymbol{\alpha}!} \sum_{k \in I_B} f(\boldsymbol{x}_k) \left(\varepsilon(\boldsymbol{x}_k - \boldsymbol{x}_B)\right)^{\boldsymbol{\alpha}}, \qquad \boldsymbol{\alpha} \leq p,$$

of its Hermite expansion (35.6).

(3) For each evaluation box C, determine its interaction region $IR(C)$.

(4) For each evaluation box C transform all Hermite expansions in source boxes within the interaction region $IR(C)$ into a single Taylor expansion using (35.7), *i.e.*,

$$\mathcal{G}_f(\boldsymbol{y}_j) \approx \sum_{\boldsymbol{\beta} \leq p} c_{\boldsymbol{\beta}} \left(\varepsilon(\boldsymbol{y}_j - \boldsymbol{y}_C)\right)^{\boldsymbol{\beta}},$$

where

$$c_{\boldsymbol{\beta}} = \frac{(-1)^{|\boldsymbol{\beta}|}}{\boldsymbol{\beta}!} \sum_{B \in IR(C)} \sum_{\boldsymbol{\alpha} \leq p} b_{\boldsymbol{\alpha}} h_{\boldsymbol{\alpha}+\boldsymbol{\beta}} \left(\varepsilon(\boldsymbol{x}_B - \boldsymbol{y}_C)\right).$$

For a small number of points direct summation is more efficient than the fast transform. Unfortunately, the value of the "crossover point" grows with the space dimension s. This makes the fast Gauss transform in its basic form virtually unusable for 3D applications.

Note that the algorithm presented here does not use a hierarchical decomposition of space as is typical for so-called *tree codes*, as well as many other more general fast multipole algorithms. In the algorithm above the interaction region is determined simply based on the fast decay of the Gaussian.

Clearly, the majority of the work has to be performed in step 4. The performance of this step can be improved by using *plane wave* expansions to diagonalize the

translation operators (see [Greengard and Sun (1998)]). In order to keep matters as simple as possible, we will not discuss this feature.

A more complete algorithm (designed for radial basis function interpolation with multiquadrics and thin plate splines) has been developed by Beatson and co-workers (see, *e.g.*, [Beatson and Newsam (1992); Cherrie *et al.* (2002)]).

For the numerical experiments in Table 35.1 we used the C-code **FGT** which can also be used as a MEX-file with MATLAB. The code was written by Adam Florence and can be obtained at `http://www.cs.cornell.edu/aflorenc/research/-fgt.html` (see also [Florence and van Loan (2000)]). The numerical results presented in Table 35.1 were obtained by performing quasi-interpolation of the form

$$\mathcal{Q}_f^{(h)}(x) = (\pi\mathcal{D})^{-1/2} \sum_{k=1}^{N} f(x_k)\Phi\left(\frac{x - x_k}{\sqrt{\mathcal{D}}h}\right),$$

with a Gaussian Φ on $N = 2^\ell + 1$, $\ell = 2, 3, 4, \ldots, 18$, equally spaced points in $[0, 1]$ with the mollified test function

$$f(x) = 15e^{\frac{-1}{1-(2x-1)^2}}\left[\frac{3}{4}e^{-(x-2)^2/4} + \frac{3}{4}e^{-(x+1)^2/49} + \frac{1}{2}e^{-(x-7)^2/4} - \frac{1}{5}e^{-(x-4)^2}\right].$$

All errors were computed on $M = 524289$ equally spaced points in $[0, 1]$. In the "rate" column we list the number $\text{rate} = \ln(e_{\ell-1}/e_\ell)/\ln 2$ corresponding to the exponent in the $\mathcal{O}(h^{\text{rate}})$ notation. Other parameters were $\mathcal{D} = 4$, and the default values for the **FGT** code (*i.e.*, $R = 0.5$). All times were measured in seconds.

Table 35.1 1D quasi-interpolation using fast Gauss transform.

	direct			fast			
N	max-error	rate	time	max-error	rate	time	speedup
5	3.018954e-00		1.93	5.495125e-00		1.07	1.80
9	2.037762e-00	0.57	3.40	2.037762e-00	1.43	5.31	0.64
17	9.617170e-01	1.08	6.39	9.617170e-01	1.08	5.33	1.20
33	3.609205e-01	1.41	12.28	3.609205e-01	1.41	5.35	2.30
65	1.190192e-01	1.60	24.72	1.190192e-01	1.60	5.39	4.59
129	3.354132e-02	1.83	53.38	3.354132e-02	1.83	5.46	10.14
257	8.702868e-03	1.95	113.35	8.702868e-03	1.95	5.61	20.20
513	2.196948e-03	1.99	226.15	2.196948e-03	1.99	5.94	38.07
1025			450*	5.505832e-04	2.00	6.67	67.47
2049			900*	1.377302e-04	2.00	7.87	114.36
4097			1800*	3.443783e-05	2.00	10.56	170.45
8193			3600*	8.609789e-06	2.00	15.78	228.14
16385			7200*	2.152468e-06	2.00	26.27	274.08
32769			14400*	5.381182e-07	2.00	47.39	303.86
65537			28800*	1.345296e-07	2.00	89.91	320.32
131073			57600*	3.363241e-08	2.00	174.74	329.63
262145			115200*	8.408103e-09	2.00	343.59	335.28

An asterisk * on the entries in the lower part of the "direct" column indicates estimated times. The fast Gauss transform yields a speedup of roughly a factor of

300. Another way to interpret these results is that for roughly the same amount of work we can obtain an answer which is about 100000 times more accurate. The predicted $\mathcal{O}(h^2)$ convergence of the Gaussian quasi-interpolant (*c.f.* Chapter 26) is perfectly illustrated by the entries in the "rate" columns.

35.2 Fast Tree Codes

An alternative to fast multipole methods are so-called *fast tree codes*. These kind of algorithms originated in computational chemistry. For the interested reader we recommend recent mathematical papers by Krasny and co-workers (*e.g.*, [Duan and Krasny (2001); Lindsay and Krasny (2001)]). An advantage of fast tree code methods is that they make use of standard Taylor expansions instead of the specialized expansions that are used in the context of the fast multipole expansions of the previous section (such as, e.g, in terms of Hermite functions, spherical harmonics, spherical Hankel functions, plane waves, or hypergeometric functions [Cherrie *et al.* (2002)]). This simplifies their implementation. However, their convergence properties are probably not as good as those of fast multipole expansions.

We now present a very general discussion of fast summation via Taylor expansions. The presentation of this material is motivated by the work of Krasny and co-workers (see, *e.g.*, [Duan and Krasny (2001); Lindsay and Krasny (2001)]) as well as the algorithm for the fast Gauss transform reviewed in the previous section. Since we are interested in many simultaneous evaluations of our quasi-interpolants (or other radial basis function expansion), we split the set of M evaluation points $\boldsymbol{y}_j$ into groups (contained in boxes C with centers $\boldsymbol{y}_C$). We also split the N data locations $\boldsymbol{x}_k$ into boxes B with centers $\boldsymbol{x}_B$, and use the index set I_B to denote the points in B.

In order to set the stage for a fast summation of the quasi-interpolant

$$\begin{aligned} \mathcal{Q}_f(\boldsymbol{y}_j) &= \sum_{k=1}^{N} f(\boldsymbol{x}_k)\Phi(\boldsymbol{y}_j - \boldsymbol{x}_k) \\ &= \sum_{B} \sum_{k \in I_B} f(\boldsymbol{x}_k)\Phi(\boldsymbol{y}_j - \boldsymbol{x}_k) \end{aligned} \tag{35.8}$$

with generating function Φ we require the multivariate Taylor expansion of Φ about a point $\boldsymbol{z}_0 \in \mathbb{R}^s$, *i.e.*,

$$\Phi(\boldsymbol{z}) = \sum_{\boldsymbol{\alpha} \geq 0} D^{\boldsymbol{\alpha}}\Phi(\boldsymbol{z})|_{\boldsymbol{z}=\boldsymbol{z}_0} \frac{(\boldsymbol{z} - \boldsymbol{z}_0)^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}, \tag{35.9}$$

where α is a multi-index. Now — as for the fast Gauss transform — we consider three basic expansions.

Theorem 35.4 (Taylor Series Expansion about Centers of Target Boxes). *Let I_B be the index set denoting the sources $\boldsymbol{x}_k$ that lie in a box B with center $\boldsymbol{x}_B$,*

and let $\boldsymbol{y}_C$ be the center of the target box C containing an evaluation point $\boldsymbol{y}_j$. Then the quasi-interpolant due to sources in B

$$\mathcal{Q}_f^{(B)}(\boldsymbol{y}_j) = \sum_{k \in I_B} f(\boldsymbol{x}_k)\Phi(\boldsymbol{y}_j - \boldsymbol{x}_k)$$

can be written as a Taylor expansion about $\boldsymbol{y}_C$:

$$\mathcal{Q}_f^{(B)}(\boldsymbol{y}_j) = \sum_{\boldsymbol{\alpha} \geq 0} a_{\boldsymbol{\alpha}} (\boldsymbol{y}_j - \boldsymbol{y}_C)^{\boldsymbol{\alpha}},$$

where

$$a_{\boldsymbol{\alpha}} = \frac{(-1)^{|\boldsymbol{\alpha}|}}{\boldsymbol{\alpha}!} \sum_{k \in I_B} f(\boldsymbol{x}_k) T_{\boldsymbol{\alpha}}(\boldsymbol{y}_C, \boldsymbol{x}_k)$$

with $T_{\boldsymbol{\alpha}}(\boldsymbol{y}_C, \boldsymbol{x}_k) = (-1)^{|\boldsymbol{\alpha}|} D^{\boldsymbol{\alpha}} \Phi(\boldsymbol{z})|_{\boldsymbol{z} = \boldsymbol{y}_C - \boldsymbol{x}_k}$.

Proof. We combine the contribution for the source box B of (35.8) with (35.9), and let $\boldsymbol{z} = \boldsymbol{y}_j - \boldsymbol{x}_k$ and $\boldsymbol{z}_0 = \boldsymbol{y}_C - \boldsymbol{x}_k$. Then (35.8) becomes

$$\mathcal{Q}_f^{(B)}(\boldsymbol{y}_j) = \sum_{k \in I_B} f(\boldsymbol{x}_k) \sum_{\boldsymbol{\alpha} \geq 0} D^{\boldsymbol{\alpha}} \Phi(\boldsymbol{z})|_{\boldsymbol{z} = \boldsymbol{y}_C - \boldsymbol{x}_k} \frac{(\boldsymbol{y}_j - \boldsymbol{y}_C)^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}.$$

Using the abbreviation $T_{\boldsymbol{\alpha}}(\boldsymbol{y}_C, \boldsymbol{x}_k) = (-1)^{|\boldsymbol{\alpha}|} D^{\boldsymbol{\alpha}} \Phi(\boldsymbol{z})|_{\boldsymbol{z} = \boldsymbol{y}_C - \boldsymbol{x}_k}$ we can rewrite this as

$$\mathcal{Q}_f^{(B)}(\boldsymbol{y}_j) = \sum_{\boldsymbol{\alpha} \geq 0} a_{\boldsymbol{\alpha}} (\boldsymbol{y}_j - \boldsymbol{y}_C)^{\boldsymbol{\alpha}},$$

where

$$a_{\boldsymbol{\alpha}} = \frac{(-1)^{|\boldsymbol{\alpha}|}}{\boldsymbol{\alpha}!} \sum_{k \in I_B} f(\boldsymbol{x}_k) T_{\boldsymbol{\alpha}}(\boldsymbol{y}_C, \boldsymbol{x}_k).$$

□

Example 35.1. If we take $\Phi(\boldsymbol{x}) = e^{-\|\boldsymbol{x}\|^2}$ then

$$T_{\boldsymbol{\alpha}}(\boldsymbol{y}_C, \boldsymbol{x}_k) = h_{\boldsymbol{\alpha}}(\boldsymbol{y}_C - \boldsymbol{x}_k) = h_{\boldsymbol{\alpha}}(\boldsymbol{x}_k - \boldsymbol{y}_C),$$

and Theorem 35.4 is equivalent to Theorem 35.1 given above.

We can see that the Taylor expansion has allowed us to separate the evaluation points $\boldsymbol{y}_j$ from the data points $\boldsymbol{x}_k$.

Theorem 35.5 (Taylor Series Expansion about Centers of Source Boxes). *Let I_B be the index set denoting the sources $\boldsymbol{x}_k$ that lie in a box B with center $\boldsymbol{x}_B$. Then the quasi-interpolant due to sources in B*

$$\mathcal{Q}_f^{(B)}(\boldsymbol{y}_j) = \sum_{k \in I_B} f(\boldsymbol{x}_k)\Phi(\boldsymbol{y}_j - \boldsymbol{x}_k)$$

can be written as a reversed Taylor expansion about $\boldsymbol{x}_B$:

$$\mathcal{Q}_f^{(B)}(\boldsymbol{y}_j) = \sum_{\boldsymbol{\alpha} \geq 0} b_{\boldsymbol{\alpha}} T_{\boldsymbol{\alpha}}(\boldsymbol{y}_j, \boldsymbol{x}_B),$$

with the moments *$b_{\boldsymbol{\alpha}}$ given by*

$$b_{\boldsymbol{\alpha}} = \frac{1}{\boldsymbol{\alpha}!} \sum_{k \in I_B} f(\boldsymbol{x}_k)(\boldsymbol{x}_k - \boldsymbol{x}_B)^{\boldsymbol{\alpha}},$$

and $T_{\boldsymbol{\alpha}}(\boldsymbol{y}_j, \boldsymbol{x}_B) = (-1)^{|\boldsymbol{\alpha}|} D^{\boldsymbol{\alpha}} \Phi(\boldsymbol{z})|_{\boldsymbol{z} = \boldsymbol{y}_j - \boldsymbol{x}_B}$.

Proof. We combine the contribution for the source box B of (35.8) with (35.9), and let $\boldsymbol{z} = \boldsymbol{y}_j - \boldsymbol{x}_k$ and $\boldsymbol{z}_0 = \boldsymbol{y}_j - \boldsymbol{x}_B$. Then (35.8) becomes

$$\mathcal{Q}_f^{(B)}(\boldsymbol{y}_j) = \sum_{k \in I_B} f(\boldsymbol{x}_k) \sum_{\boldsymbol{\alpha} \geq 0} D^{\boldsymbol{\alpha}} \Phi(\boldsymbol{z})|_{\boldsymbol{z}=\boldsymbol{y}_j - \boldsymbol{x}_B} (-1)^{|\boldsymbol{\alpha}|} \frac{(\boldsymbol{x}_k - \boldsymbol{x}_B)^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}.$$

Using the abbreviation $T_{\boldsymbol{\alpha}}(\boldsymbol{y}_j, \boldsymbol{x}_B) = (-1)^{|\boldsymbol{\alpha}|} D^{\boldsymbol{\alpha}} \Phi(\boldsymbol{z})|_{\boldsymbol{z}=\boldsymbol{y}_j - \boldsymbol{x}_B}$ we can reverse the role of the Taylor coefficients and the polynomials to write this as

$$\mathcal{Q}_f^{(B)}(\boldsymbol{y}_j) = \sum_{\boldsymbol{\alpha} \geq 0} b_{\boldsymbol{\alpha}} T_{\boldsymbol{\alpha}}(\boldsymbol{y}_j, \boldsymbol{x}_B),$$

with

$$b_{\boldsymbol{\alpha}} = \frac{1}{\boldsymbol{\alpha}!} \sum_{k \in I_B} f(\boldsymbol{x}_k)(\boldsymbol{x}_k - \boldsymbol{x}_B)^{\boldsymbol{\alpha}}.$$

□

Example 35.2. Using $\Phi(\boldsymbol{x}) = e^{-\|\boldsymbol{x}\|^2}$ this is equivalent to Theorem 35.2.

The moments $b_{\boldsymbol{\alpha}}$ can be pre-computed and stored during the setup phase of the algorithm.

Theorem 35.6 (Conversion). *Let I_B be the index set denoting the sources $\boldsymbol{x}_k$ that lie in a box B with center $\boldsymbol{x}_B$, and let $\boldsymbol{y}_C$ be the center of the target box C containing $\boldsymbol{y}_j$. Then a fast summation formula for the quasi-interpolant*

$$\mathcal{Q}_f(\boldsymbol{y}_j) = \sum_{k=1}^{N} f(\boldsymbol{x}_k) \Phi(\boldsymbol{y}_j - \boldsymbol{x}_k)$$

can be given as an expansion about $\boldsymbol{y}_C$:

$$\mathcal{Q}_f(\boldsymbol{y}_j) \approx \sum_{\boldsymbol{\beta} \leq p} c_{\boldsymbol{\beta}} (\boldsymbol{y}_j - \boldsymbol{y}_C)^{\boldsymbol{\beta}},$$

where

$$c_{\boldsymbol{\beta}} = \frac{(-1)^{|\boldsymbol{\beta}|}}{\boldsymbol{\beta}!} \sum_{\boldsymbol{\alpha}+\boldsymbol{\beta} \leq p} \sum_{B} T_{\boldsymbol{\alpha}+\boldsymbol{\beta}}(\boldsymbol{y}_C, \boldsymbol{x}_B) b_{\boldsymbol{\alpha}},$$

$T_{\boldsymbol{\alpha}+\boldsymbol{\beta}}(\boldsymbol{y}_C, \boldsymbol{x}_B) = (-1)^{|\boldsymbol{\alpha}+\boldsymbol{\beta}|} D^{\boldsymbol{\alpha}+\boldsymbol{\beta}} \Phi(\boldsymbol{z})|_{\boldsymbol{z}=\boldsymbol{y}_C - \boldsymbol{x}_B}$, and the moments $b_{\boldsymbol{\alpha}}$ are as in Theorem 35.5.

Proof. We combine (35.8) with (35.9), and now replace $\boldsymbol{z}$ by $\boldsymbol{y}_j - \boldsymbol{x}_k$ and $\boldsymbol{z}_0$ by $\boldsymbol{y}_C - \boldsymbol{x}_B$. Then (35.8) becomes

$$\mathcal{Q}_f(\boldsymbol{y}_j) = \sum_{B} \sum_{k \in I_B} f(\boldsymbol{x}_k) \sum_{\boldsymbol{\alpha} \geq 0} D^{\boldsymbol{\alpha}} \Phi(\boldsymbol{z})|_{\boldsymbol{z}=\boldsymbol{y}_C - \boldsymbol{x}_B} \frac{(\boldsymbol{y}_j - \boldsymbol{x}_k - (\boldsymbol{y}_C - \boldsymbol{x}_B))^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}.$$

Using the abbreviation $T_{\boldsymbol{\alpha}}(\boldsymbol{y}_C, \boldsymbol{x}_B) = (-1)^{|\boldsymbol{\alpha}|} D^{\boldsymbol{\alpha}} \Phi(\boldsymbol{z})|_{\boldsymbol{z}=\boldsymbol{y}_C-\boldsymbol{x}_B}$ along with the multivariate binomial theorem we can rewrite this as

$$\begin{aligned}
\mathcal{Q}_f(\boldsymbol{y}_j) &= \sum_B \sum_{k \in I_B} f(\boldsymbol{x}_k) \sum_{\boldsymbol{\alpha} \geq 0} (-1)^{|\boldsymbol{\alpha}|} \frac{T_{\boldsymbol{\alpha}}(\boldsymbol{y}_C, \boldsymbol{x}_B)}{\boldsymbol{\alpha}!} \times \\
&\quad \sum_{\boldsymbol{\beta} \leq \boldsymbol{\alpha}} \binom{\boldsymbol{\alpha}}{\boldsymbol{\beta}} (-1)^{|\boldsymbol{\beta}|} (\boldsymbol{y}_j - \boldsymbol{y}_C)^{\boldsymbol{\alpha}-\boldsymbol{\beta}} (\boldsymbol{x}_k - \boldsymbol{x}_B)^{\boldsymbol{\beta}} \\
&= \sum_{\boldsymbol{\alpha} \geq 0} \sum_B (-1)^{|\boldsymbol{\alpha}|} T_{\boldsymbol{\alpha}}(\boldsymbol{y}_C, \boldsymbol{x}_B) \sum_{\boldsymbol{\beta} \leq \boldsymbol{\alpha}} (-1)^{|\boldsymbol{\beta}|} \frac{(\boldsymbol{y}_j - \boldsymbol{y}_C)^{\boldsymbol{\alpha}-\boldsymbol{\beta}}}{(\boldsymbol{\alpha}-\boldsymbol{\beta})!} \times \\
&\quad \sum_{k \in I_B} f(\boldsymbol{x}_k) \frac{(\boldsymbol{x}_k - \boldsymbol{x}_B)^{\boldsymbol{\beta}}}{\boldsymbol{\beta}!}.
\end{aligned}$$

In fact, we can introduce the moments of Theorem 35.5 and write

$$\mathcal{Q}_f(\boldsymbol{y}_j) = \sum_{\boldsymbol{\alpha} \geq 0} \sum_B (-1)^{|\boldsymbol{\alpha}|} T_{\boldsymbol{\alpha}}(\boldsymbol{y}_C, \boldsymbol{x}_B) \sum_{\boldsymbol{\beta} \leq \boldsymbol{\alpha}} (-1)^{|\boldsymbol{\beta}|} \frac{(\boldsymbol{y}_j - \boldsymbol{y}_C)^{\boldsymbol{\alpha}-\boldsymbol{\beta}}}{(\boldsymbol{\alpha}-\boldsymbol{\beta})!} b_{\boldsymbol{\beta}},$$

where

$$b_{\boldsymbol{\beta}} = \frac{1}{\boldsymbol{\beta}!} \sum_{k \in I_B} f(\boldsymbol{x}_k)(\boldsymbol{x}_k - \boldsymbol{x}_B)^{\boldsymbol{\beta}}.$$

A fast algorithm is now obtained by truncating the infinite series after the p-th order terms, *i.e.*,

$$\mathcal{Q}_f(\boldsymbol{y}_j) \approx \sum_{\boldsymbol{\alpha} \leq p} \sum_B (-1)^{|\boldsymbol{\alpha}|} T_{\boldsymbol{\alpha}}(\boldsymbol{y}_C, \boldsymbol{x}_B) \sum_{\boldsymbol{\beta} \leq \boldsymbol{\alpha}} (-1)^{|\boldsymbol{\beta}|} \frac{(\boldsymbol{y}_j - \boldsymbol{y}_C)^{\boldsymbol{\alpha}-\boldsymbol{\beta}}}{(\boldsymbol{\alpha}-\boldsymbol{\beta})!} b_{\boldsymbol{\beta}}.$$

Using the fact that

$$\sum_{\boldsymbol{\alpha} \leq p} a_{\boldsymbol{\alpha}} \sum_{\boldsymbol{\beta} \leq \boldsymbol{\alpha}} b_{\boldsymbol{\alpha}-\boldsymbol{\beta}} = \sum_{\boldsymbol{\alpha} \leq p} b_{\boldsymbol{\alpha}} \sum_{\boldsymbol{\alpha} \leq \boldsymbol{\beta} \leq p} a_{\boldsymbol{\beta}} = \sum_{\boldsymbol{\alpha} \leq p} b_{\boldsymbol{\alpha}} \sum_{\boldsymbol{\alpha}+\boldsymbol{\beta} \leq p} a_{\boldsymbol{\alpha}+\boldsymbol{\beta}},$$

which can be verified by a simple rearrangement of the summations and an index transformation, we obtain (interchanging the role of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$) the following fast summation formula:

$$\mathcal{Q}_f(\boldsymbol{y}_j) \approx \sum_{\boldsymbol{\beta} \leq p} \sum_{\boldsymbol{\alpha}+\boldsymbol{\beta} \leq p} (-1)^{|\boldsymbol{\alpha}|} \frac{1}{\boldsymbol{\beta}!} \sum_B (-1)^{|\boldsymbol{\alpha}+\boldsymbol{\beta}|} T_{\boldsymbol{\alpha}+\boldsymbol{\beta}}(\boldsymbol{y}_C, \boldsymbol{x}_B) b_{\boldsymbol{\alpha}} (\boldsymbol{y}_j - \boldsymbol{y}_C)^{\boldsymbol{\beta}}.$$

This is equivalent to the statement of the theorem. □

Example 35.3. Using $\Phi(\boldsymbol{x}) = e^{-\|\boldsymbol{x}\|^2}$ Theorem 35.6 is almost equivalent to Theorem 35.2. However, our alternate formula is more efficient since only Hermite functions up to order p are required (as opposed to order $2p$ in the Greengard/Strain version). This gain is achieved by using the binomial theorem instead of a second Taylor expansion. The Hermite series expansion used in the traditional fast Gauss transform is equivalent to a Taylor expansion.

Note that the Taylor coefficients $T_\alpha(\boldsymbol{y}_C, \boldsymbol{x}_B)$ depend only on the box centers $\boldsymbol{y}_C$ and $\boldsymbol{x}_B$.

In order to make the algorithm efficient one will use a decision rule (as in Strain's code for the fast Gauss transform) to decide when to use which of the three expansions. Error estimation is very similar to Greengard/Strain. The only difference is that one needs bounds on the Taylor coefficients instead of the Hermite functions.

In order to adapt this fast transform to Laguerre-Gaussian generating functions (or any other generating function) one needs to compute the required Taylor coefficients. This is a task that goes beyond the scope of this book.

35.3 Domain Decomposition

Finally, another method commonly used to deal with large computational problems is the *domain decomposition* method. Domain decomposition is frequently implemented on parallel computers in order to speed up the computation. A standard reference (based mostly on finite difference and finite element methods) is the book by Smith, Bjørstad and Gropp [Smith *et al.* (1996)]. For radial basis functions there is a recent paper by Beatson, Light and Billings [Beatson *et al.* (2000)].

The main aim of the paper [Beatson *et al.* (2000)] is to solve the radial basis function interpolation problem discussed many times in previous chapters. In particular, a so-called *multiplicative Schwarz* algorithm (which is analogous to Gauss-Seidel iteration) is presented, and linear convergence of the algorithm is proved. A section with numerical experiments reports results for an *additive Schwarz* method (which is analogous to Jacobi iteration).

In particular, the authors implemented polyharmonic radial basis functions and used the scale invariant basis discussed in Section 34.4.

The classical additive Schwarz algorithm is usually discussed in the context of partial differential equations, and it is known that one should add a coarse level correction in order to ensure convergence and to filter out some of the low-frequency oscillations (see, *e.g.*, [Smith *et al.* (1996)]).

In [Beatson *et al.* (2000)] a two-level additive algorithm for interpolation problems was presented. One begins by subdividing the set of interpolation points $\mathcal{X}$ into M smaller sets $\mathcal{X}_i$, $i = 1, \ldots, M$, whose pairwise intersection is non-empty. The points that belong to one set $\mathcal{X}_i$ only are called *inner points* of $\mathcal{X}_i$. Those points in the intersection of more than one set need to be assigned in some way as inner points to only one of the subsets $\mathcal{X}_i$ so that the collection of all inner points yields the entire set $\mathcal{X}$. This corresponds to the concept of *overlapping domains*. One also needs to choose a coarse grid $\mathcal{Y}$ that contains points from all of the inner point sets.

In the setup phase of the algorithm the radial basis function interpolation matrices for the smaller problems on each of the subsets $\mathcal{X}_i$, $i = 1, \ldots, M$, are computed and factored. At this point one can use the homogeneous basis of Section 34.4 to ensure numerical stability. Now the algorithm proceeds as follows:

Algorithm 35.2.

Input: Data f, point sets $\mathcal{X}_i$ and factored interpolation matrices A_i, $i = 1, \ldots, M$, tolerance tol
Initialize $r = f$, $u = 0$
While $\|r\| >$ tol do

For $i = 1$ to M (*i.e.*, for each subset $\mathcal{X}_i$) do

Determine the coefficient vector $\boldsymbol{c}^{(i)}$ of the interpolant to the residual $r|_{\mathcal{X}_i}$ on $\mathcal{X}_i$.

end
Make $\boldsymbol{c}$ orthogonal to Π^s_{m-1}.
Assemble an intermediate approximation $u_1 = \sum_{j=1}^{N} c_j \Phi(\cdot, \boldsymbol{x}_j)$.
Compute the residual on the coarse grid, *i.e.*,

$$r_1 = r - u_1|_{\mathcal{Y}}.$$

Interpolate to r_1 on the coarse grid $\mathcal{Y}$ using an RBF expansion u_2.
Update $u \leftarrow u + u_1 + u_2$.
Re-evaluate the global residual $r = f - u$ on the whole set $\mathcal{X}$

end

In [Beatson *et al.* (2000)] it is proved that a multiplicative version of this algorithm converges at least linearly. However, the additive version can be more easily implemented on a parallel computer.

If strictly positive definite kernels such as Gaussians are used, then it is not necessary to make the coefficients $\boldsymbol{c}$ orthogonal to polynomials.

As in many algorithms before, the evaluation of the residuals needs to be made "fast" using either a fast multipole method or a version of the fast Fourier transform.

In the case of very large data sets it may be necessary to use more than two levels so that one ends up with a *multigrid* algorithm.

The authors of [Beatson *et al.* (2000)] report having solved interpolation problems with several millions of points using the domain decomposition algorithm above.

A number of other papers discussing domain decomposition methods for radial basis functions have appeared in the literature (see, *e.g.*, [Dubal (1994); Hon and Wu (2000); Ingber *et al.* (2004); Li and Hon (2004); Ling and Kansa (2004); Wong *et al.* (1999)]). However, most of these papers contain little theory, focusing mostly on numerical experiments.

Chapter 36

Generalized Hermite Interpolation

In 1975 Rolland Hardy mentioned the possibility of using multiquadric basis functions for Hermite interpolation, *i.e.*, interpolation to data that also contains derivative information (see [Hardy (1975)] or the survey paper [Hardy (1990)]). This problem, however, was not further investigated in the RBF literature until the paper [Wu (1992)] appeared. Since then, the interest in this topic has increased significantly. In particular, since there is a close connection between the generalized Hermite interpolation approach and symmetric collocation for elliptic partial differential equations (see Chapter 38). Wu deals with Hermite-Birkhoff interpolation in $\mathbb{R}^s$ and his method is limited in the sense that one can have only one interpolation condition per data point (*i.e.*, some linear combination of function value and derivatives). In [Sun (1994a)] this restriction is eliminated. Sun deals with the Euclidean setting and gives results analogous to the (Lagrange) interpolation results of [Micchelli (1986)]. In [Narcowich and Ward (1994a)] an even more general theory of Hermite interpolation for conditionally positive definite (matrix-valued) kernels in $\mathbb{R}^s$ is developed. Hermite interpolation with conditionally positive definite functions is also discussed in [Iske (1995)]. A number of authors have also considered the Hermite interpolation setting on spheres (see, *e.g.*, [Fasshauer (1999b); Freeden (1982); Freeden (1987); Ron and Sun (1996)]) or even general Riemannian manifolds [Dyn *et al.* (1999); Narcowich (1995)].

36.1 The Generalized Hermite Interpolation Problem

We now consider data $\{\boldsymbol{x}_i, \lambda_i f\}$, $i = 1, \ldots, N$, $\boldsymbol{x}_i \in \mathbb{R}^s$, where $\Lambda = \{\lambda_1, \ldots, \lambda_N\}$ is a linearly independent set of continuous linear functionals and f is some (smooth) data function. For example, λ_i could denote point evaluation at the point $\boldsymbol{x}_i$ and thus yield a Lagrange interpolation condition, or it could denote evaluation of some derivative at the point $\boldsymbol{x}_i$. However, we allow the set Λ to contain more general functionals such as, *e.g.*, local integrals. This kind of problem was recently studied in [Beatson and Langton (2006)]. Furthermore, we stress that there is no assumption that requires the derivatives to be in consecutive order as is usually the case for

polynomial or spline-type Hermite interpolation problems.

We try to find an interpolant of the form

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \psi_j(\|\boldsymbol{x}\|), \qquad \boldsymbol{x} \in \mathbb{R}^s, \tag{36.1}$$

with appropriate (radial) basis functions ψ_j so that $\mathcal{P}_f$ satisfies the *generalized interpolation conditions*

$$\lambda_i \mathcal{P}_f = \lambda_i f, \qquad i = 1, \ldots, N.$$

To keep the discussion that follows as transparent as possible we now introduce the notation $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_N$ for the *centers* of the radial basis functions. They will usually be selected to coincide with the data sites $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$. However, the following is clearer if we formally distinguish between centers $\boldsymbol{\xi}_j$ and data sites $\boldsymbol{x}_i$.

As we will show in the next section, it is natural to let $\psi_j(\|\boldsymbol{x}\|) = \lambda_j^{\boldsymbol{\xi}} \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}\|)$ with the same functionals λ_j that generated the data and φ one of the usual radial basic functions. However, the notation $\lambda^{\boldsymbol{\xi}}$ indicates that the functional λ now acts on φ viewed as a function of its second argument $\boldsymbol{\xi}$. We will not add any superscript if λ acts on a single variable function or on the kernel φ as a function of its first variable. Therefore, we assume the generalized Hermite interpolant to be of the form

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \lambda_j^{\boldsymbol{\xi}} \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}\|), \qquad \boldsymbol{x} \in \mathbb{R}^s, \tag{36.2}$$

and require it to satisfy

$$\lambda_i \mathcal{P}_f = \lambda_i f, \qquad i = 1, \ldots, N.$$

The linear system $A\boldsymbol{c} = \boldsymbol{f}_\lambda$ which arises in this case has matrix entries

$$A_{ij} = \lambda_i \lambda_j^{\boldsymbol{\xi}} \varphi, \qquad i, j = 1, \ldots, N, \tag{36.3}$$

and right-hand side $\boldsymbol{f}_\lambda = [\lambda_1 \boldsymbol{f}, \ldots, \lambda_N \boldsymbol{f}]^T$.

In the references mentioned at the beginning of this chapter it is shown that A is non-singular for the same classes of φ that were admissible for scattered data interpolation in our earlier chapters.

Note that when we are assembling the interpolation matrix A the functionals λ act on φ both as a function of the first variable as well as the second variable. This implies that we need to use C^{2k} functions in order to interpolate C^k data. This is the price we need to pay to ensure invertibility of A.

It is interesting to note that the effect of the derivative acting on the second variable (*i.e.*, the center) of φ (which leads to a sign change for derivatives of odd orders) was not taken into account in the early paper [Hardy (1975)], and thus his interpolation matrix is not symmetric.

It should be pointed out that the formulation in (36.2) is very general and goes considerably beyond the standard notion of Hermite interpolation (which refers to

interpolation of successive derivative values only). Here any kind of linear functionals are allowed as long as the set Λ is linearly independent. For example, in Chapter 38 we will see how this formulation can be applied to the solution of partial differential equations.

One could also envision use of a simpler RBF expansion of the form

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}\|), \qquad \boldsymbol{x} \in \mathbb{R}^s.$$

However, in this case the interpolation matrix will not be symmetric and much more difficult to analyze theoretically. In fact, the approach just suggested is frequently used for the solution of elliptic partial differential equations (see the description of Kansa's method in Chapter 38), and it is known that for certain configurations of the collocation points and certain differential operators the system matrix does indeed become singular.

The question of when the functionals in Λ are linearly independent is not addressed in most papers on the subject. However, the book [Wendland (2005a)] contains the following reassuring theorem that covers both Hermite interpolation and collocation solutions of PDEs.

Theorem 36.1. *Suppose that $\Phi \in L_1(\mathbb{R}^s) \cap C^{2k}(\mathbb{R}^s)$ is a strictly positive definite kernel. If the functionals $\lambda_j = \delta_{\boldsymbol{x}_j} \circ D^{\boldsymbol{\alpha}^{(j)}}$, $j = 1, \ldots, N$, with multi-indices $|\boldsymbol{\alpha}^{(j)}| \le k$ are pairwise distinct, meaning that $\boldsymbol{\alpha}^{(j)} \ne \boldsymbol{\alpha}^{(\ell)}$ if $\boldsymbol{x}_j = \boldsymbol{x}_\ell$ for different $j \ne \ell$, then they are also linearly independent over the native space $\mathcal{N}_\Phi(\mathbb{R}^s)$.*

In the theorem above the functional $\delta_{\boldsymbol{x}_j}$ denotes point evaluation at the point $\boldsymbol{x}_j$, and the kernel Φ is related to φ as usual, *i.e.*, $\Phi(\boldsymbol{x}, \boldsymbol{\xi}) = \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}\|)$. Like most results on strictly positive definite functions, this theorem can also be generalized to the strictly conditionally positive definite case.

36.2 Motivation for the Symmetric Formulation

In this section we illustrate why the formulation used in (36.2) is natural for the Hermite interpolation problem. That is, aside from the fact that the symmetric interpolation matrix (36.3) is guaranteed to be invertible for all commonly used RBFs, we will show that by choosing the basis functions as in (36.2) the matrix associated with (Hermite) interpolation to function value and first derivative value at a point corresponds to a limit of the matrix for Lagrange interpolation to clusters of points. We will also illustrate this fact numerically in the next section.

In [Franke *et al.* (1995)] the authors investigated adaptive least squares approximation with multiquadrics in $\mathbb{R}^2$ by means of inserting knots (similar to our algorithm of Chapter 21). The authors describe numerical experiments which suggest that (Lagrange) multiquadric basis functions associated with clusters of centers in

adaptive least squares approximation should be replaced by appropriate directional derivatives of one of the basis functions.

We now present a theoretical justification for this observation based on an analysis of a one-dimensional example. A more general analysis involving higher derivatives and higher-dimensional spaces would be of the same flavor using the multivariate Taylor theorem. We discuss interpolation to function values and first derivatives at given points on the real line using radial basis functions.

To show how one general sub-block in the Hermite matrix relates to an associated block of a Lagrange matrix, it will suffice to analyze the sub-block of the Lagrange interpolation matrix corresponding to two pairs of nearby points. Let these points be x_i, $x_i + \Delta x$, and ξ_j, $\xi_j + \Delta\xi$ for some indices i and j and some small distances Δx and $\Delta\xi$. Furthermore, let the radial function be of the form $\varphi = \varphi(|x - \xi|)$, $x, \xi \in \mathbb{R}$. We also assume φ is differentiable at the origin. In the proof of the following lemma we make use of the following identities, which are straightforward applications of the univariate Taylor theorem

$$\varphi(|(x + \Delta x) - \xi|) = \varphi(|x - \xi|) + \Delta x \frac{\partial}{\partial x}\varphi(|x - \xi|) + \mathcal{O}((\Delta x)^2), \tag{36.4}$$

$$\varphi(|x - (\xi + \Delta\xi)|) = \varphi(|x - \xi|) - \Delta\xi \frac{\partial}{\partial \xi}\varphi(|x - \xi|) + \mathcal{O}((\Delta\xi)^2). \tag{36.5}$$

To keep the notation as simple as possible we write $\frac{\partial}{\partial x}\varphi(|x_i - \xi_j|)$ to denote $\frac{\partial}{\partial x}\varphi(|x - \xi_j|)|_{x=x_i}$, $\frac{\partial}{\partial \xi}\varphi(|x_i - \xi_j|)$ to denote $\frac{\partial}{\partial \xi}\varphi(|x_i - \xi|)|_{\xi=\xi_j}$, and $\frac{\partial^2}{\partial x \partial \xi}\varphi(|x_i - \xi_j|)$ to denote $\frac{\partial^2}{\partial x \partial \xi}\varphi(|x - \xi|)|_{x=x_i, \xi=\xi_j}$.

Lemma 36.1. *For the 1D situation described above we have*

$$\frac{\det M_L}{\Delta x \Delta \xi} = \det M_H + \mathcal{O}(\Delta x) + \mathcal{O}(\Delta \xi),$$

where M_L is the part of the Lagrange matrix corresponding to the basis functions centered at ξ_j and $\xi_j + \Delta\xi$ interpolating to values at x_i and $x_i + \Delta x$, i.e.,

$$M_L = \begin{bmatrix} \varphi(|x_i - \xi_j|) & \varphi(|x_i - (\xi_j + \Delta\xi)|) \\ \varphi(|(x_i + \Delta x) - \xi_j|) & \varphi(|(x_i + \Delta x) - (\xi_j + \Delta\xi)|) \end{bmatrix},$$

and M_H is the associated Hermite block

$$M_H = \begin{bmatrix} \varphi(|x_i - \xi_j|) & -\frac{\partial}{\partial \xi}\varphi(|x_i - \xi_j|) \\ \frac{\partial}{\partial x}\varphi(|x_i - \xi_j|) & -\frac{\partial^2}{\partial x \partial \xi}\varphi(|x_i - \xi_j|) \end{bmatrix}.$$

Proof. If we use (36.4) to modify the second row of M_L, and then subtract the first row from the second one, we obtain

$$\det M_L = \Delta x \begin{vmatrix} \varphi(|x_i - \xi_j|) & \varphi(|x_i - (\xi_j + \Delta\xi)|) \\ \frac{\partial}{\partial x}\varphi(|x_i - \xi_j|) + \mathcal{O}(\Delta x) & \frac{\partial}{\partial x}\varphi(|x_i - (\xi_j + \Delta\xi)|) + \mathcal{O}(\Delta x) \end{vmatrix}.$$

This technique is commonly used when analyzing sign properties of Hermite matrices (see, *e.g.*, [Schumaker (1981)]). Now we repeat this process with (36.5) and the

second column of M_L to get

$$\det M_L = \Delta x \Delta \xi \times \begin{vmatrix} \varphi(|x_i - \xi_j|) & -\frac{\partial}{\partial \xi}\varphi(|x_i - \xi_j|) + \mathcal{O}(\Delta\xi) \\ \frac{\partial}{\partial x}\varphi(|x_i - \xi_j|) + \mathcal{O}(\Delta x) & -\frac{\partial^2}{\partial x \partial \xi}\varphi(|x_i - \xi_j|) + \mathcal{O}(\Delta x) + \mathcal{O}(\Delta\xi) \end{vmatrix},$$

and thus the statement follows. □

We now illustrate the Hermite interpolation approach with a simple 2D example using first-order partial derivative functionals.

Example 36.1. Let data $\{\boldsymbol{x}_i, f(\boldsymbol{x}_i)\}_{i=1}^n$ and $\{\boldsymbol{x}_i, \frac{\partial f}{\partial x}(\boldsymbol{x}_i)\}_{i=n+1}^N$ with $\boldsymbol{x} = (x, y) \in \mathbb{R}^2$ be given. Thus

$$\lambda_i = \begin{cases} \delta_{\boldsymbol{x}_i}, & i = 1, \ldots, n, \\ \delta_{\boldsymbol{x}_i} \circ \frac{\partial}{\partial x}, & i = n+1, \ldots, N. \end{cases}$$

Then

$$\begin{aligned} \mathcal{P}_f(\boldsymbol{x}) &= \sum_{j=1}^N c_j \lambda_j^{\boldsymbol{\xi}} \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}\|) \\ &= \sum_{j=1}^n c_j \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}_j\|) + \sum_{j=n+1}^N c_j \frac{\partial \varphi}{\partial \xi}(\|\boldsymbol{x} - \boldsymbol{\xi}_j\|) \\ &= \sum_{j=1}^n c_j \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}_j\|) - \sum_{j=n+1}^N c_j \frac{\partial \varphi}{\partial x}(\|\boldsymbol{x} - \boldsymbol{\xi}_j\|). \end{aligned}$$

After enforcing the interpolation conditions the system matrix is given by

$$A = \begin{bmatrix} \tilde{A} & \tilde{A}_\xi \\ \tilde{A}_x & \tilde{A}_{x\xi} \end{bmatrix}$$

with

$$\begin{aligned} \tilde{A}_{ij} &= \varphi(\|\boldsymbol{x}_i - \boldsymbol{\xi}_j\|), && i, j = 1, \ldots, n, \\ (\tilde{A}_\xi)_{ij} &= \frac{\partial \varphi}{\partial \xi}(\|\boldsymbol{x}_i - \boldsymbol{\xi}_j\|) = -\frac{\partial \varphi}{\partial x}(\|\boldsymbol{x}_i - \boldsymbol{\xi}_j\|), && i = 1, \ldots, n,\ j = n+1, \ldots, N, \\ (\tilde{A}_x)_{ij} &= \frac{\partial \varphi}{\partial x}(\|\boldsymbol{x}_i - \boldsymbol{\xi}_j\|), && i = n+1, \ldots, N,\ j = 1, \ldots, n, \\ (\tilde{A}_{x\xi})_{ij} &= \frac{\partial^2 \varphi}{\partial x^2}(\|\boldsymbol{x}_i - \boldsymbol{\xi}_j\|), && i, j = n+1, \ldots, N. \end{aligned}$$

Note that the two blocks $\tilde{A}_\xi$ and $\tilde{A}_x$ are identical provided the data sites and centers coincide since in this case the sign change due to differentiation with respect to the second variable in $\tilde{A}_\xi$ is cancelled by the interchange of the roles of $\boldsymbol{x}_i$ and $\boldsymbol{\xi}_j$ when compared to $\tilde{A}_x$. Here one needs to realize that the partial derivative of φ with respect to the coordinate x will always contain a linear factor in x, *i.e.*, (for

the 2D example considered here) $\varphi(\|\boldsymbol{x}\|) = \varphi(r) = \varphi(\sqrt{x^2+y^2})$, so that by the chain rule

$$\begin{aligned}\frac{\partial}{\partial x}\varphi(\|\boldsymbol{x}\|) &= \frac{d}{dr}\varphi(r)\frac{\partial}{\partial x}r(x,y)\\ &= \frac{d}{dr}\varphi(r)\frac{x}{\sqrt{x^2+y^2}}\\ &= \frac{d}{dr}\varphi(r)\frac{x}{r}\end{aligned} \tag{36.6}$$

since $r = \|\boldsymbol{x}\| = \sqrt{x^2+y^2}$. This argument generalizes for any odd-order derivative.

Note that the matrix A is also symmetric for even-order derivatives. For example, one can easily verify that

$$\frac{\partial^2}{\partial x^2}\varphi(\|\boldsymbol{x}\|) = \frac{1}{r^2}\left(x^2\frac{d^2}{dr^2}\varphi(r) + \frac{y^2}{r}\frac{d}{dr}\varphi(r)\right),$$

so that now the interchange of $\boldsymbol{x}_i$ and $\boldsymbol{\xi}_j$ does not cause a sign change. On the other hand, two derivatives of φ with respect to the second variable $\boldsymbol{\xi}$ do not lead to a sign change, either.

A catalog of RBFs and their derivatives is provided in Appendix D.

Chapter 37

RBF Hermite Interpolation in MATLAB

We now illustrate the symmetric approach to Hermite interpolation with a set of numerical experiments for first-order Hermite interpolation (*i.e.*, to positional and gradient data) in 2D using the MATLAB program `RBFHermite_2D.m` listed below as Program 37.1. Since derivatives of both the RBFs and the test function need to be included in the program we use the function

$$f(x,y) = \frac{\tanh(9(y-x))+1}{\tanh(9)+1}$$

which has fairly simple partial derivatives (see lines 9–10 of the program) to generate the data. The RBF used in this set of experiments is the multiquadric with shape parameter $\varepsilon = 6$.

We compare four different problems:

(1) Lagrange interpolation, *i.e.*, interpolation to function values only, at a set of N equally spaced points in the unit square.
(2) Lagrange interpolation to function values at $3N$ clustered points with separation distance $q = 0.1h$, where h is the fill distance of the set of equally spaced points (see the left plot in Figure 37.1).
(3) The same as above, but with $q = 0.01h$ (see the right plot in Figure 37.1).
(4) Hermite interpolation to function value, and values of both first-order partial derivatives at the N equally spaced points used in the first experiment.

The standard Lagrange interpolants were computed via Program 2.1 with the required modification of the RBF and test function definitions, *i.e.*, line 1 is replaced by

```
1  rbf = @(e,r) sqrt(1+(e*r).^2); ep = 6;
```

and lines 2–6 are replaced by the single line

```
2  testfunction = @(x,y) (tanh(9*(y-x))+1)/(tanh(9)+1);
```

The experiments with Lagrange interpolation at clustered data sites were accomplished by the same program by adding the following code between lines 8 and 9 in Program 2.1:

```
q = 0.1/(sqrt(N)-1);
grid = linspace(0,1,sqrt(N));
shifted = linspace(q,1+q,sqrt(N)); shifted(end) = 1-q;
[xc1,yc1] = meshgrid(shifted,grid);
[xc2,yc2] = meshgrid(grid,shifted);
dsites = [dsites; xc1(:) yc1(:); xc2(:) yc2(:)];
```

The resulting data point sets for `q = 0.1/(sqrt(N)-1)`, *i.e.*, $q = h/10$, and for `q = 0.01/(sqrt(N)-1)` (or $q = h/100$) are shown in Figure 37.1.

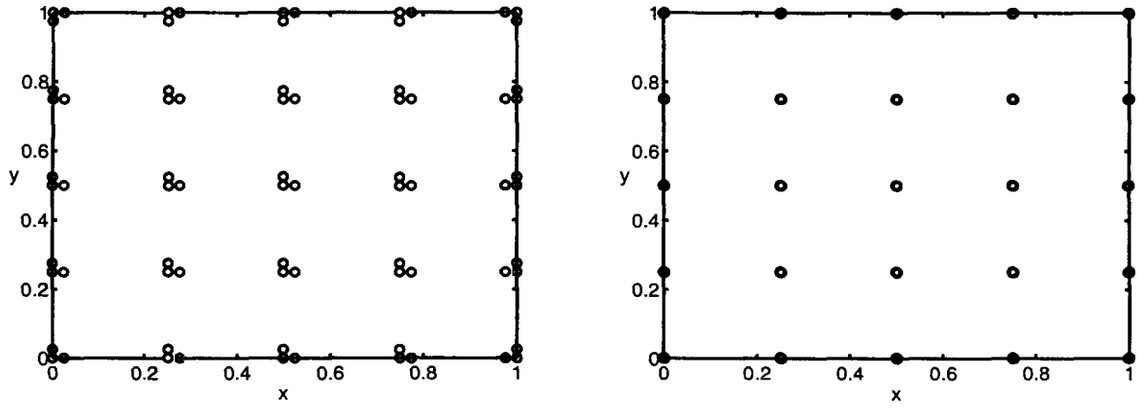


Fig. 37.1 Clustered point sets with $N = 25$ basic data points. Cluster size $h/10$ (left) and cluster size $h/100$ (right).

The program `RBFHermite_2D.m` maintains the same basic structure as earlier interpolation programs. Now, however, we need to define derivatives of the RBF of up to twice the order of the data. This is done for the MQ basic function on lines 1–6. Note that the second-order partials could be expressed either as stated in Program 37.1 or as

```
4  dxxrbf = @(e,r,dy) e^2*(1+(e*dy).^2)./(1+(e*r).^2).^(3/2);
6  dyyrbf = @(e,r,dx) e^2*(1+(e*dx).^2)./(1+(e*r).^2).^(3/2);
```

Here `dx` and `dy` denote non-radial difference terms of the x or y-components, respectively (see, *e.g.*, (36.6)). We choose the former representation since for many other basic functions these second-order partials are more naturally expressed in terms of the differences of the variable of differentiation (*i.e.*, `dxxrbf` is expressed in terms of x-differences, *etc.*).

Since the derivatives of the basic function now also contain the difference terms mentioned above, we need another subroutine that computes matrices of differences of point coordinates. This subroutine is called `DifferenceMatrix.m` (see Program 37.2), and it is built analogous to Program 1.1 (`DistanceMatrix.m`). Thus, on lines 17–22 of Program 37.1 we compute not only distance matrices of data sites and centers (or evaluation points and centers), but also the corresponding difference

matrices. These three matrices are then required when we evaluate the RBF and its partials to obtain the building blocks of the interpolation and evaluation matrices (see lines 25–35). Note the minus signs used with the blocks in columns 2 and 3 of the block matrices IM and EM on lines 31 and 35. They reflect differentiation of the basic function with respect to its second variable (*c.f.* (36.2) and (36.3)).

The data are generated by sampling the test function and its derivatives (see lines 8–10, and line 23). Evaluation of the interpolant, error computation and rendering are exactly the same as in earlier programs.

Program 37.1. RBFHermite_2D.m

```
% RBFHermite_2D
% Script that performs first-order 2D RBF Hermite interpolation
% Calls on: DistanceMatrix, DifferenceMatrix
    % Define RBF and its derivatives
 1  rbf = @(e,r) sqrt(1+(e*r).^2);   % MQ RBF
 2  dxrbf = @(e,r,dx) dx*e^2./sqrt(1+(e*r).^2);
 3  dyrbf = @(e,r,dy) dy*e^2./sqrt(1+(e*r).^2);
 4a dxxrbf = @(e,r,dx) e^2*(1+(e*r).^2-(e*dx).^2)./...
 4b                          (1+(e*r).^2).^(3/2);
 5  dxyrbf = @(e,r,dx,dy) -e^4*dx.*dy./(1+(e*r).^2).^(3/2);
 6a dyyrbf = @(e,r,dy) e^2*(1+(e*r).^2-(e*dy).^2)./...
 6b                          (1+(e*r).^2).^(3/2);
 7  ep = 6;
    % Define test function and its derivatives
 8  tf = @(x,y) (tanh(9*(y-x))+1)/(tanh(9)+1);
 9  tfDx = @(x,y) 9*(tanh(9*(y-x)).^2-1)/(tanh(9)+1);
10  tfDy = @(x,y) 9*(1-tanh(9*(y-x)).^2)/(tanh(9)+1);
11  N = 289; gridtype = 'u';
12  neval = 40;
    % Load data points
13  name = sprintf('Data2D_%d%s',N,gridtype); load(name)
14  ctrs = dsites;
    % Create neval-by-neval equally spaced evaluation locations
   % in the unit square
15  grid = linspace(0,1,neval); [xe,ye] = meshgrid(grid);
16  epoints = [xe(:) ye(:)];
    % Compute the distance and difference matrices for
    % evaluation matrix
17  DM_eval = DistanceMatrix(epoints,ctrs);
18  dx_eval = DifferenceMatrix(epoints(:,1),ctrs(:,1));
19  dy_eval = Differencematrix(epoints(:,2),ctrs(:,2));
    % Compute the distance and difference matrices for
```

```
    % interpolation matrix
20  DM_data = DistanceMatrix(dsites,ctrs);
21  dx_data = DifferenceMatrix(dsites(:,1),ctrs(:,1));
22  dy_data = DifferenceMatrix(dsites(:,2),ctrs(:,2));
23a rhs = [tf(dsites(:,1),dsites(:,2)); ...
23b       tfDx(dsites(:,1),dsites(:,2)); ...
23c       tfDy(dsites(:,1),dsites(:,2))];
24  exact = tf(epoints(:,1),epoints(:,2));
    % Compute blocks for interpolation matrix
25  IM = rbf(ep,DM_data);
26  DxIM = dxrbf(ep,DM_data,dx_data);
27  DylM = dyrbf(ep,DM_data,dy_data);
28  DxxIM = dxxrbf(ep,DM_data,dx_data);
29  DxylM = dxyrbf(ep,DM_data,dx_data,dy_data);
30  DyyIM = dyyrbf(ep,DM_data,dy_data);
    % Assemble symmetric interpolation matrix
31a IM = [IM -DxIM -DylM;
31b      DxIM -DxxIM -DxylM;
31c      DylM -DxylM -DyyIM];
    % Compute blocks for evaluation matrix
32  EM = rbf(ep,DM_eval);
33  DxEM = dxrbf(ep,DM_eval,dx_eval);
34  DyEM = dyrbf(ep,DM_eval,dy_eval);
    % Assemble evaluation matrix
35  EM = [EM -DxEM -DyEM];
    % RBF Hermite interpolant
36  Pf = EM * (IM\rhs);
    % Compute errors on evaluation grid
37  maxerr = norm(Pf-exact,inf);
38  rms_err = norm(Pf-exact)/neval;
39  fprintf('RMS error:     %e\n', rms_err)
40  fprintf('Maximum error: %e\n', maxerr)
41  fview = [-30,30];  % viewing angles for plot
42  PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview);
43  PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview);
```

Program 37.2. DifferenceMatrix.m

```
% DM = DifferenceMatrix(datacoord,centercoord)
% Forms the difference matrix of two sets of points in R
% (some fixed coordinate of point in R^s), i.e.,
% DM(j,k) = datacoord_j - centercoord_k .
 1  function DM = DifferenceMatrix(datacoord,centercoord)
```

```
    % The ndgrid command produces two MxN matrices:
    %   dr, consisting of N identical columns
    %       (each containing the M data sites)
    %   cc, consisting of M identical rows
    %       (each containing the N centers)
2   [dr,cc] = ndgrid(datacoord(:),centercoord(:));
3   DM = dr-cc;
```

In Tables 37.1 and 37.2 as well as Figure 37.2 we display RMS-errors, ℓ_2-condition numbers of the interpolation matrices, and plots of the interpolants for the experiments described above.

Several observations can be made. First, the limiting relation between clustered Lagrange interpolants and Hermite interpolants as discussed in the previous section is obvious. Moreover, it is also obvious that interpolation to function and derivative data at a given point is more accurate than interpolation to function values alone.

Table 37.1 2D interpolation with clustered data vs. Hermite interpolation (part 1).

	Lagrange		clustered, $q = 0.1h$	
mesh	RMS-error	cond(A)	RMS-error	cond(A)
3×3	1.620492e-001	6.078349e+001	8.471301e-002	9.052247e+003
5×5	6.148258e-002	9.464176e+002	2.733258e-002	3.073957e+005
9×9	8.521994e-003	6.523036e+004	2.678543e-003	8.811980e+007
17×17	2.246810e-004	9.017750e+007	3.138761e-005	3.555214e+012
33×33	2.017643e-006	4.799960e+013	2.925784e-007	6.474324e+020

Table 37.2 2D interpolation with clustered data vs. Hermite interpolation (part 2).

	clustered, $q = 0.01h$		Hermite	
mesh	RMS-error	cond(A)	RMS-error	cond(A)
3×3	9.084939e-002	8.580483e+005	9.128193e-002	1.326346e+002
5×5	2.792157e-002	2.829762e+007	2.794943e-002	2.292450e+003
9×9	2.687753e-003	8.325283e+009	2.688346e-003	2.185224e+005
17×17	3.147808e-005	3.426489e+014	3.148843e-005	2.486624e+009
33×33	8.941613e-006	8.943758e+020	5.731027e-009	6.261336e+018

The advantage of the Hermite interpolation approach over the clustered Lagrange approach is clearly evident for the experiments with $N = 33 \times 33 = 1089$ basic data points (or $N = 3267$ clustered data points). In this case the ℓ_2-condition number of A for the clustered interpolants is on the order of 10^{20}, while it is "only" 6.261336e+018 for the Hermite matrix. This difference, however, has a significant

impact on the numerical stability, and the resulting RMS-errors. The Hermite interpolant is more than three orders of magnitude more accurate than the Lagrange interpolant to clusters with $q = h/100$ (see the last row of Table 37.2).

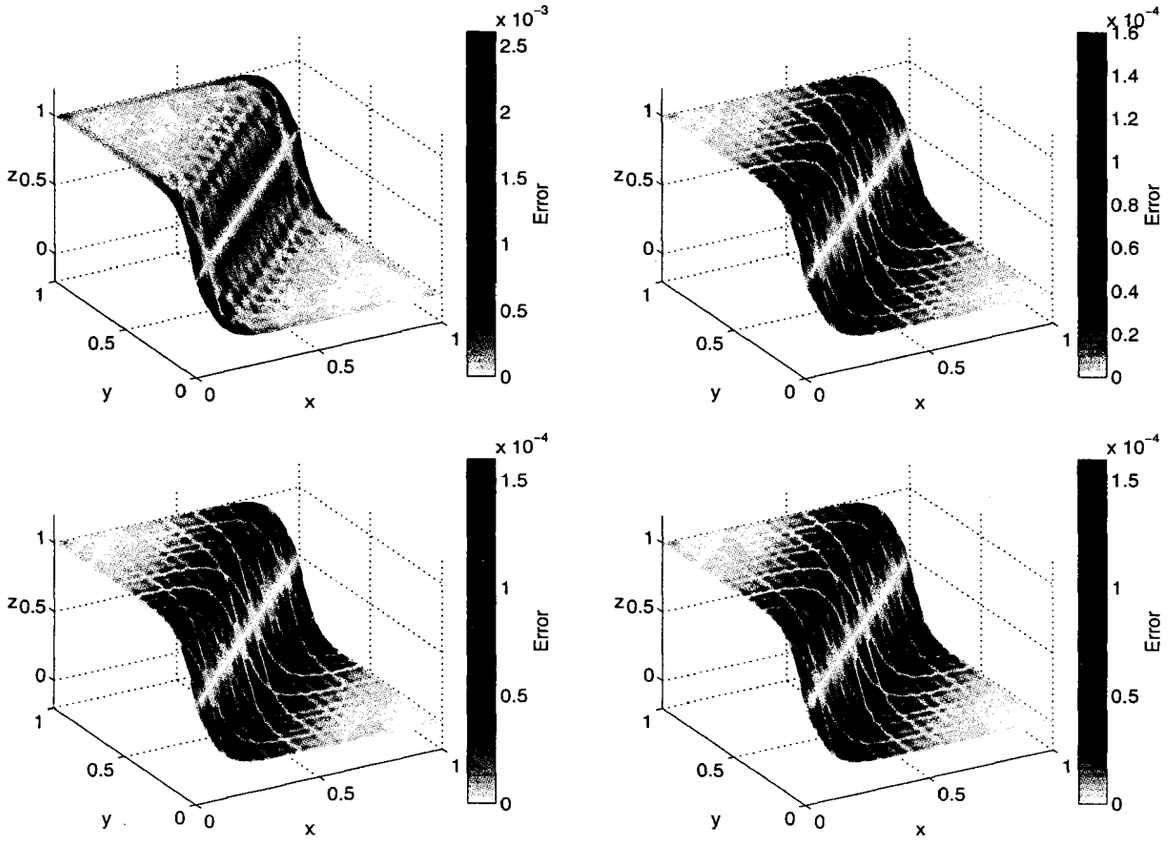


Fig. 37.2 Fits for clustered interpolants with $N = 289$ basic data points. Top left to bottom right: Lagrange interpolant, interpolant with cluster size $h/10$, interpolant with cluster size $h/100$, Hermite interpolant.

Chapter 38

Solving Elliptic Partial Differential Equations via RBF Collocation

In this chapter we discuss how the techniques used in previous chapters for Lagrange and Hermite interpolation can be applied to the numerical solution of elliptic partial differential equations. The resulting numerical method will be a *collocation approach* based on radial basis functions. In the PDE literature this is also often referred to as a *strong form solution.*

To make the discussion transparent we will initially focus on the case of a time independent linear elliptic partial differential equation in $\mathbb{R}^2$.

38.1 Kansa's Approach

A now very popular *non-symmetric method* for the solution of elliptic PDEs with radial basis functions was suggested by Ed Kansa in [Kansa (1990b)]. In order to be able to clearly point out the differences between Kansa's method and a symmetric approach proposed in [Fasshauer (1997)] we recall some of the basics of scattered data interpolation with radial basis functions in $\mathbb{R}^s$.

In the scattered data interpolation context we are given data $\{\boldsymbol{x}_i, f_i\}$, $i = 1, \ldots, N$, $\boldsymbol{x}_i \in \mathbb{R}^s$, where we can think of the values f_i being sampled from a function $f : \mathbb{R}^s \to \mathbb{R}$. The goal is to find an interpolant of the form

$$\mathcal{P}_f(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \varphi(\|\boldsymbol{x} - \boldsymbol{x}_j\|), \qquad \boldsymbol{x} \in \mathbb{R}^s, \tag{38.1}$$

such that

$$\mathcal{P}_f(\boldsymbol{x}_i) = f_i, \qquad i = 1, \ldots, N.$$

The solution of this problem leads to a linear system $A\boldsymbol{c} = \boldsymbol{f}$ with the entries of A given by

$$A_{ij} = \varphi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|), \qquad i, j = 1, \ldots, N. \tag{38.2}$$

As discussed earlier, the matrix A is non-singular for a large class of radial functions including (inverse) multiquadrics, Gaussians, and the strictly positive definite compactly supported functions of Wendland, Wu, Gneiting or Buhmann. In the case

of strictly conditionally positive definite functions such as polyharmonic splines the problem needs to be augmented by polynomials.

We now switch to the collocation solution of partial differential equations. Assume we are given a domain $\Omega \subset \mathbb{R}^s$, and a linear elliptic partial differential equation of the form

$$\mathcal{L}u(\boldsymbol{x}) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \text{ in } \Omega, \tag{38.3}$$

with (for simplicity of description) Dirichlet boundary conditions

$$u(\boldsymbol{x}) = g(\boldsymbol{x}), \qquad \boldsymbol{x} \text{ on } \partial\Omega. \tag{38.4}$$

For Kansa's collocation method we then choose to represent the approximate solution $\hat{u}$ by a radial basis function expansion analogous to that used for scattered data interpolation, *i.e.*,

$$\hat{u}(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}_j\|). \tag{38.5}$$

As in the previous chapter on Hermite interpolation we now formally distinguish in our notation between *centers* $\Xi = \{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_N\}$ and *collocation points* $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \Omega$. While formally different, these points will often physically coincide. A scenario with $\Xi \neq \mathcal{X}$ will be explored in Chapters 39 and 40. For the following discussion we assume the simplest possible setting, *i.e.*, $\Xi = \mathcal{X}$ and no polynomial terms are added to the expansion (38.5).

The collocation matrix that arises when matching the differential equation (38.3) and the boundary conditions (38.4) at the collocation points $\mathcal{X}$ will be of the form

$$A = \begin{bmatrix} \tilde{A}_{\mathcal{L}} \\ \tilde{A} \end{bmatrix}, \tag{38.6}$$

where the two blocks are generated as follows:

$$\begin{aligned} (\tilde{A}_{\mathcal{L}})_{ij} &= \mathcal{L}\varphi(\|\boldsymbol{x} - \boldsymbol{\xi}_j\|)|_{\boldsymbol{x}=\boldsymbol{x}_i}, \qquad \boldsymbol{x}_i \in \mathcal{I},\ \boldsymbol{\xi}_j \in \Xi, \\ \tilde{A}_{ij} &= \varphi(\|\boldsymbol{x}_i - \boldsymbol{\xi}_j\|), \qquad \boldsymbol{x}_i \in \mathcal{B},\ \boldsymbol{\xi}_j \in \Xi. \end{aligned}$$

Here the set $\mathcal{X}$ of collocation points is split into a set $\mathcal{I}$ of interior points, and a set $\mathcal{B}$ of boundary points. The problem is well-posed if the linear system $A\boldsymbol{c} = \boldsymbol{y}$, with $\boldsymbol{y}$ a vector consisting of entries $f(\boldsymbol{x}_i)$, $\boldsymbol{x}_i \in \mathcal{I}$, followed by $g(\boldsymbol{x}_i)$, $\boldsymbol{x}_i \in \mathcal{B}$, has a unique solution.

We note that a change in the boundary conditions (38.4) is as simple as making changes to a few rows of the matrix A in (38.6) as well as on the right-hand side $\boldsymbol{y}$.

We also point out that while this is a rather general description of a numerical method with no particular RBF in mind, Kansa specifically proposed to use multiquadrics in (38.5), and consequently this non-symmetric collocation approach often appears in the literature as the *multiquadric method*. In the paper [Kansa (1990b)] the author describes three sets of experiments using the multiquadric method and he

comments on the superior performance of multiquadrics in terms of computational complexity and accuracy when compared to finite difference methods.

Moreover, Kansa suggests the use of varying shape parameters ε_j, $j = 1, \ldots, N$. While the theoretical analysis of the resulting method is near intractable, Kansa shows that this technique improves the accuracy and stability of the method when compared to using only one constant value of ε (see [Kansa (1990b)]). Except for one paper by Bozzini, Lenarduzzi and Schaback [Bozzini *et al.* (2002)] (which addresses only the interpolation setting) the theoretical aspects of varying shape parameters have not been discussed in the literature.

A problem with Kansa's method is that — for a constant shape parameter ε — the matrix A *may be singular* for certain configurations of the centers $\boldsymbol{\xi}_j$. Originally, Kansa assumed that the non-singularity results established by Micchelli for interpolation matrices (see the discussion in the earlier chapters of this book) would carry over to the PDE case. However, as the numerical experiments of [Hon and Schaback (2001)] show, this is not so. This fact is not really surprising since the matrix for the collocation problem is composed of rows that are built from *different* functions, which — depending on the differential operator $\mathcal{L}$ — might not even be radial. The results for the non-singularity of interpolation matrices, however, are based on the fact that A is generated by a *single* function φ.

Nevertheless, an indication of the success of Kansa's method are the early papers [Dubai (1992); Dubai (1994); Golberg *et al.* (1996); Kansa (1992); Moridis and Kansa (1994)] and many more since. Since the numerical experiments of Hon and Schaback show that Kansa's method cannot be well-posed for arbitrary center locations, it is now an open question to find sufficient conditions on the center locations that guarantee invertibility of the Kansa matrix. One possible approach — built on the basic ideas of the greedy algorithm of Chapter 33 — is to adaptively select "good" centers from a large set of possible candidates. Following this strategy it is possible to ensure invertibility of the collocation matrix throughout the iterative algorithm. This approach is described in the recent paper [Ling *et al.* (2006)].

Before we discuss an alternate approach (based on the symmetric Hermite interpolation method) which does ensure well-posedness of the resulting collocation matrix we would like to point out that in [Moridis and Kansa (1994)] the authors suggest how Kansa's method can be applied to other types of partial differential equation problems such as non-linear elliptic PDEs, systems of elliptic PDEs, and time-dependent parabolic or hyperbolic PDEs. We will also see in the next chapter that Kansa's method is well-suited for elliptic problems with variable coefficients. We will come back to the use of Kansa's method for time-dependent problems in Chapter 42.

38.2 An Hermite-based Approach

The following symmetric collocation method is based on the generalized Hermite interpolation method detailed in Chapter 36. Assume we are given the same linear elliptic PDE (38.3) with Dirichlet boundary conditions (38.4) as in the previous section on Kansa's method. In order to be able to apply the results from generalized Hermite interpolation that will ensure the non-singularity of the collocation matrix we propose the following expansion for the unknown function u:

$$\hat{u}(\boldsymbol{x}) = \sum_{j=1}^{N_{\mathcal{I}}} c_j \mathcal{L}^{\boldsymbol{\xi}} \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}\|)|_{\boldsymbol{\xi}=\boldsymbol{\xi}_j} + \sum_{j=N_{\mathcal{I}}+1}^{N} c_j \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}_j\|). \tag{38.7}$$

Here $N_{\mathcal{I}}$ denotes the number of nodes in the interior of Ω, and $\mathcal{L}^{\boldsymbol{\xi}}$ is the differential operator used in the differential equation (38.3), but acting on φ viewed as a function of the second argument, *i.e.*, $\mathcal{L}\varphi$ is equal to $\mathcal{L}^{\boldsymbol{\xi}}\varphi$ up to a possible difference in sign. Thus, the linear functionals λ in (36.2) are given by $\lambda_j = \delta_{\boldsymbol{\xi}_j} \circ \mathcal{L}$, $j = 1, \ldots, N_{\mathcal{I}}$, and $\lambda_j = \delta_{\boldsymbol{\xi}_j}$, $j = N_{\mathcal{I}} + 1, \ldots, N$.

After enforcing the collocation conditions

$$\begin{aligned} \mathcal{L}\hat{u}(\boldsymbol{x}_i) &= f(\boldsymbol{x}_i), & \boldsymbol{x}_i \in \mathcal{I}, \\ \hat{u}(\boldsymbol{x}_i) &= g(\boldsymbol{x}_i), & \boldsymbol{x}_i \in \mathcal{B}, \end{aligned}$$

we end up with a collocation matrix A that is of the form

$$A = \begin{bmatrix} \hat{A}_{\mathcal{L}\mathcal{L}^{\boldsymbol{\xi}}} & \hat{A}_{\mathcal{L}} \\ \hat{A}_{\mathcal{L}^{\boldsymbol{\xi}}} & \hat{A} \end{bmatrix}. \tag{38.8}$$

Here the four blocks are generated as follows:

$$\begin{aligned} (\hat{A}_{\mathcal{L}\mathcal{L}^{\boldsymbol{\xi}}})_{ij} &= \mathcal{L}\mathcal{L}^{\boldsymbol{\xi}} \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}\|)|_{\boldsymbol{x}=\boldsymbol{x}_i, \boldsymbol{\xi}=\boldsymbol{\xi}_j}, & \boldsymbol{x}_i, \boldsymbol{\xi}_j \in \mathcal{I}, \\ (\hat{A}_{\mathcal{L}})_{ij} &= \mathcal{L}\varphi(\|\boldsymbol{x} - \boldsymbol{\xi}_j\|)|_{\boldsymbol{x}=\boldsymbol{x}_i}, & \boldsymbol{x}_i \in \mathcal{I}, \ \boldsymbol{\xi}_j \in \mathcal{B}, \\ (\hat{A}_{\mathcal{L}^{\boldsymbol{\xi}}})_{ij} &= \mathcal{L}^{\boldsymbol{\xi}} \varphi(\|\boldsymbol{x}_i - \boldsymbol{\xi}\|)|_{\boldsymbol{\xi}=\boldsymbol{\xi}_j}, & \boldsymbol{x}_i, \in \mathcal{B}, \ \boldsymbol{\xi}_j \in \mathcal{I}, \\ \hat{A}_{ij} &= \varphi(\|\boldsymbol{x}_i - \boldsymbol{\xi}_j\|), & \boldsymbol{x}_i, \boldsymbol{\xi}_j \in \mathcal{B}. \end{aligned}$$

Note that we have identified the two sets $\mathcal{X} = \mathcal{I} \cup \mathcal{B}$ of collocation points and Ξ of centers.

The matrix A of (38.8) is of the same type as the generalized Hermite interpolation matrices (36.3), and therefore non-singular as long as φ is chosen appropriately. Thus, viewed using the new expansion (38.7) for $\hat{u}$, the collocation approach is certainly well-posed. Another point in favor of the Hermite-based approach is that the matrix (38.8) is symmetric as opposed to the completely unstructured matrix (38.6) of the same size used in the non-symmetric approach. This property is of value when trying to devise an efficient implementation of the collocation method. Also note that although A now consists of four blocks, it still is of the same size, namely $N \times N$, as the collocation matrix (38.6) obtained for Kansa's approach.

However, the symmetric collocation matrix is more complicated to assemble, it requires smoother basis functions than the non-symmetric Kansa method, and it does not lend itself very nicely to the solution of non-linear problems.

One attempt to obtain an efficient implementation of the Hermite-based collocation method is a variation of the greedy algorithm described in Section 33.1. We refer the reader to the original paper [Hon *et al.* (2003)] for details.

38.3 Error Bounds for Symmetric Collocation

A convergence analysis for the symmetric collocation method was provided in [Franke and Schaback (1998a); Franke and Schaback (1998b)]. The error estimates established in those papers require the solution of the PDE to be very smooth. Therefore, one should be able to use meshfree radial basis function collocation techniques especially well for (high-dimensional) PDE problems with smooth solutions on possibly irregular domains. Due to the known counterexamples from [Hon and Schaback (2001)] for the non-symmetric method, a convergence analysis is still lacking for that method. However, for an adaptive version of the non-symmetric method Schaback recently analyzed the convergence in [Schaback (2006a)].

In [Wendland (2005a)] one can find the following convergence result for the symmetric collocation method:

Theorem 38.1. *Let $\Omega \subseteq \mathbb{R}^s$ be a polygonal and open region. Let $\mathcal{L} \neq 0$ be a second-order linear elliptic differential operator with coefficients in $C^{2(k-2)}(\bar{\Omega})$ that either vanish on $\bar{\Omega}$ or have no zero there. Suppose that $\Phi \in C^{2k}(\mathbb{R}^s)$ is a strictly positive definite function. Suppose further that the boundary value problem*

$$\begin{aligned} \mathcal{L}u &= f \quad \text{in } \Omega, \\ u &= g \quad \text{on } \partial\Omega \end{aligned}$$

has a unique solution $u \in \mathcal{N}_\Phi(\Omega)$ for given $f \in C(\Omega)$ and $g \in C(\partial\Omega)$. Let $\hat{u}$ be the approximate collocation solution of the form (38.7) based on $\Phi = \varphi(\|\cdot\|)$. Then

$$\|u - \hat{u}\|_{L_\infty(\Omega)} \leq Ch^{k-2}\|u\|_{\mathcal{N}_\Phi(\Omega)}$$

for all sufficiently small h, where h is the larger of the fill distances in the interior and on the boundary of Ω, respectively.

The proof uses the same techniques as in Chapter 14 and takes advantage of a "splitting theorem" that permits splitting the error into a boundary error and an error in the interior. As a consequence of the proof Wendland suggests that the collocation points and centers be chosen so that the fill distance on the boundary is smaller than in the interior since the approximation orders differ by a factor ℓ (for differential operators of order ℓ). More precisely, he suggests distributing the points so that

$$h_{\mathcal{I},\Omega}^{k-\ell} \approx h_{\mathcal{B},\partial\Omega}^{k}.$$

Some numerical evidence for convergence rates of the symmetric collocation method is given by the examples in the next chapter, and in the papers [Jumarhon *et al.* (2000); Power and Barraco (2002)].

38.4 Other Issues

Since the methods described above were both originally used with globally supported basis functions, the same concerns about stability and numerical efficiency apply as for interpolation problems. The two recent papers [Ling and Kansa (2004); Ling and Kansa (2005)] address these issues. In particular, the authors develop a preconditioner in the spirit of the one described in Section 34.3, and describe their experience with a domain decomposition algorithm.

Recently, Miranda [Miranda (2004)] has shown that Kansa's method will be well-posed if it is combined with so-called *R-functions*. This idea was also used by Höllig and his co-workers in their development of web-splines (see, *e.g.*, [Höllig (2003)]).

Other recent papers investigating various aspects of radial basis function collocation are, *e.g.*, [Cheng *et al.* (2003); Fedoseyev *et al.* (2002); Kansa and Hon (2000); Larsson and Fornberg (2003); Leitão (2001); Mai-Duy and Tran-Cong (2001a); Young *et al.* (2004)].

For example, in the paper [Fedoseyev *et al.* (2002)] the authors suggest that the collocation points on the boundary should also be used to satisfy the PDE. The motivation for this modification is the well-known fact that both for interpolation and collocation with radial basis functions the error is largest near the boundary. In order to prevent the collocation matrix from becoming trivially singular (by using duplicate columns, *i.e.*, basis functions) it is suggested in [Fedoseyev *et al.* (2002)] that the corresponding centers lie outside the domain Ω (thus creating additional basis functions). In various numerical experiments this strategy is shown to improve the accuracy of Kansa's non-symmetric method. We implement this approach in the next chapter. However, it should be noted that there is once more no theoretical foundation for this modification of either the non-symmetric or the symmetric method.

Larsson and Fornberg compare Kansa's basic collocation method, the modification just described, and the Hermite-based symmetric approach mentioned earlier (see [Larsson and Fornberg (2003)]). Using multiquadric basis functions in a standard implementation they conclude that the symmetric method is the most accurate, followed by the non-symmetric method with boundary collocation. The reason for this is the better conditioning of the system for the symmetric method. Larsson and Fornberg also discuss an implementation of the three methods using the complex Contour-Padé integration method mentioned in Section 16.1. With this technique stability problems are overcome, and it turns out that both the symmetric and the non-symmetric method perform with comparable accuracy. Boundary

collocation of the PDE yields an improvement only if these conditions are used as additional equations, *i.e.*, by increasing the problem size. It should also be noted that often the most accurate results were achieved with values of the multiquadric shape parameter ε that would lead to severe ill-conditioning using a standard implementation, and therefore these results could be achieved only using the complex integration method. Moreover, in [Larsson and Fornberg (2003)] radial basis function collocation is deemed to be far superior in accuracy to standard second-order finite differences or even a standard Fourier-Chebyshev pseudospectral method.

Leitão applies the symmetric collocation method to a fourth-order Kirchhoff plate bending problem (see [Leitão (2001)]) and emphasizes the simplicity of the implementation of the radial basis function collocation method. Mai-Duy and Tran-Cong suggest a collocation method for which the basis functions are taken to be anti-derivatives of the usual radial basis functions (see [Mai-Duy and Tran-Cong (2001a)]). And, finally, in [Young *et al.* (2004)] the authors discuss the solution of 2D and 3D Stokes' systems by a self-consistent iterative approach based on Kansa's non-symmetric method.

Chapter 39

Non-Symmetric RBF Collocation in MATLAB

In this and the next two chapters we present a number of MATLAB implementations for standard Laplace/Possion problems, problems with variable coefficients, and problems with mixed or piecewise defined boundary conditions. The non-symmetric Kansa method is discussed in this chapter. We provide a fairly detailed presentation since the MATLAB code changes rather significantly from one problem to another.

Most of the following test examples are similar to those studied in [Li *et al.* (2003)]. We restrict ourselves to two-dimensional elliptic problems whose analytic solution is readily available and therefore can easily be verified. We will refer to a point $\boldsymbol{x}$ in $\mathbb{R}^2$ as (x, y).

39.1 Kansa's Non-Symmetric Collocation Method

Example 39.1. Consider the following Poisson problem with Dirichlet boundary conditions:

$$\begin{aligned} \nabla^2 u(x,y) &= -\frac{5}{4}\pi^2 \sin(\pi x)\cos\left(\frac{\pi y}{2}\right), \quad (x,y) \in \Omega = [0,1]^2, \\ u(x,y) &= \sin(\pi x), \quad (x,y) \in \Gamma_1, \\ u(x,y) &= 0, \quad (x,y) \in \Gamma_2, \end{aligned} \tag{39.1}$$

where $\Gamma_1 = \{(x,y) : 0 \le x \le 1,\ y = 0\}$ and $\Gamma_2 = \partial\Omega \setminus \Gamma_1$. As can easily be verified, the exact solution is given by

$$u(x,y) = \sin(\pi x)\cos\left(\frac{\pi y}{2}\right).$$

A MATLAB program for the non-symmetric collocation solution of this problem using inverse multiquadric RBFs is provided as Program 39.1. While this program still is of the same general structure as earlier interpolation programs we now require not only a definition of the basic function, but also of its Laplacian (see line 2). On lines 3 and 4 we define the exact solution and its Laplacian for this test problem. Note that when we define the right-hand side of the problem, instead of breaking the boundary condition down into two pieces as given in the problem definition above

we simply evaluate the known solution on the boundary (see line 26 of the code). Of course, this is not possible in general since the solution will not be known. In that case one would have to replace line 26 by the slightly more complicated expression

```
rhs = [Lu(intdata(:,1),intdata(:,2));...
       sin(pi*bdydata(1:sn-1,1)); zeros(3*(sn-1),1)];
```

In order to stay as close as possible to the code used in earlier programs we load the (interior) collocation points from data files. For example, on line 7 we read $N = 289$ uniformly spaced points in $[0,1]^2$ from the file `Data2D_289u` into the variable `dsites`. As always, the centers for the basis functions associated with interior points are taken to be the same as the collocation (*i.e.*, data) sites.

However, as explained in the previous chapter, we now also require collocation points and centers to fit the boundary conditions. There are several approaches we could take to accomplish this:

- We could use those collocation points read from file that lie on the boundary as boundary collocation points (and centers). This means identifying those points in the array `dsites`. This approach would be the closest in spirit to the theory discussed in the previous chapter. In MATLAB one could easily code this with the commands

  ```
  indx = find(dsites(:,1)==0 | dsites(:,1)==1 | ...
              dsites(:,2)==0 | dsites(:,2)==1);
  bdydata = dsites(indx,:);
  intdata = dsites(setdiff([1:N],indx),:);
  bdyctrs = bdydata;
  ```

  However, we do not follow this approach here.
- We can create additional collocation points for the boundary conditions. These points can lie anywhere on the boundary. We take them to be equally spaced (see lines 9–11). Note that we arrange the boundary points in a counter-clockwise manner starting from the origin. Now we have several choices for the boundary centers:
  - We can let the boundary centers coincide with the boundary collocation points. However, this approach will lead to a singular collocation matrix for uniform interior points (since that set already contains points on the boundary, and therefore duplicate columns are created). Note, however, that this approach works fine if we take the interior collocation points to be Halton points (since those points do not lie on the boundary of the unit square). This approach can be realized by replacing lines 12–14 by

    ```
    bdyctrs = bdydata;
    ```
  - We can create additional boundary centers *outside* the domain (see lines 12–15). We follow this approach in most of our experiments since it seems

to provide a slightly more accurate solution. Placing boundary centers away from the boundary has been recommended recently by a number of authors. Note that this approach takes us into the realm of RBF methods for which the centers differ from the data sites (or collocation points), and we stated earlier that not much is known theoretically about this setting (*i.e.*, invertibility of system matrices or error bounds). It is an open problem how to find the best location for the boundary centers. We take them a small distance perpendicularly from the boundary collocation points (see Figure 39.1).

Program 39.1. KansaLaplace_2D.m

```
% KansaLaplace_2D
% Script that performs Kansa collocation for 2D Laplace equation
% Calls on: DistanceMatrix
    % IMQ RBF and its Laplacian
 1  rbf = @(e,r) 1./sqrt(1+(e*r).^2); ep = 3;
 2  Lrbf = @(e,r) e^2*((e*r).^2-2)./(1+(e*r).^2).^(5/2);
    % Exact solution and its Laplacian for test problem
 3  u = @(x,y) sin(pi*x).*cos(pi*y/2);
 4  Lu = @(x,y) -1.25*pi^2*sin(pi*x).*cos(pi*y/2);
    % Number and type of collocation points
 5  N = 289; gridtype = 'u';
 6  neval = 40;
    % Load (interior) collocation points
 7  name = sprintf('Data2D_%d%s',N,gridtype); load(name);
 8  intdata = dsites;
    % Additional (equally spaced) boundary collocation points
 9  sn = sqrt(N); bdylin = linspace(0,1,sn)';
10  bdy0 = zeros(sn-1,1); bdy1 = ones(sn-1,1);
11a bdydata = [bdylin(1:end-1) bdy0; bdy1 bdylin(1:end-1);...
11b      flipud(bdylin(2:end)) bdy1; bdy0 flipud(bdylin(2:end))];
    % Create additional boundary centers OUTSIDE the domain
12  h = 1/(sn-1); bdylin = (h:h:1-h)';
13  bdy0 = -h*ones(sn-2,1); bdy1 = (1+h)*ones(sn-2,1);
14a bdyctrs = [-h -h; bdylin bdy0; 1+h -h; bdy1 bdylin;...
14b     1+h 1+h; flipud(bdylin) bdy1; -h 1+h; bdy0 flipud(bdylin)];
15  ctrs = [intdata; bdyctrs];
    % Create neval-by-neval equally spaced evaluation locations
    % in the unit square
16  grid = linspace(0,1,neval); [xe,ye] = meshgrid(grid);
17  epoints = [xe(:) ye(:)];
    % Compute evaluation matrix
```

```
18  DM_eval = DistanceMatrix(epoints,ctrs);
19  EM = rbf(ep,DM_eval);
20  exact = u(epoints(:,1),epoints(:,2));
    % Compute blocks for collocation matrix
21  DM_intdata = DistanceMatrix(intdata,ctrs);
22  LCM = Lrbf(ep,DM_intdata);
23  DM_bdydata = DistanceMatrix(bdydata,ctrs);
24  BCM = rbf(ep,DM_bdydata);
25  CM = [LCM; BCM];
    % Create right-hand side
26a rhs = [Lu(intdata(:,1),intdata(:,2)); ...
26b       u(bdydata(:,1),bdydata(:,2))];
    % Compute RBF solution
27  Pf = EM * (CM\rhs);
    % Compute maximum error on evaluation grid
28  maxerr = norm(Pf-exact,inf);
29  rms_err = norm(Pf-exact)/neval;
30  fprintf('RMS error:     %e\n', rms_err)
31  fprintf('Maximum error: %e\n', maxerr)
    % Plot collocation points and centers
32  hold on; plot(intdata(:,1),intdata(:,2),'bo');
33  plot(bdydata(:,1),bdydata(:,2),'rx');
34  plot(bdyctrs(:,1),bdyctrs(:,2),'gx'); hold off
35  fview = [-30,30];  % viewing angles for plot
36  PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview);
37  PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview);
```

In Tables 39.1 and 39.2 we list RMS-errors and condition numbers for the non-symmetric collocation solution of the PDE problem (39.1). In Table 39.1 and the right part of Table 39.2 we present results for collocation with inverse multiquadric RBFs using a shape parameter of $\varepsilon = 3$, $N = 289$ interior, and an additional 64 boundary collocation points. In Table 39.1 the interior points are irregularly spaced Halton points, while in Table 39.2 we use uniformly spaced interior points. The boundary centers are placed outside the domain for the results in Table 39.2 (see the explanation above and the left part of Figure 39.1). In Table 39.1 we compare the effect of placing the boundary centers directly on the boundary (coincident with the boundary collocation points) as opposed to placement outside the domain as in Figure 39.1.

The left part of Table 39.2 compares the use of Gaussians (with the same shape parameter $\varepsilon = 3$) to inverse multiquadrics. For Gaussians we replace lines 1 and 2 of Program 39.1 by

```
1  rbf = @(e,r) exp(-(e*r).^2); ep = 3;
```

Table 39.1 Non-symmetric collocation solution of Example 39.1 with IMQs, $\varepsilon = 3$ and interior Halton points.

N (interior points)	centers on boundary		centers outside	
	RMS-error	cond(A)	RMS-error	cond(A)
9	5.642192e-002	5.276474e+002	6.029293e-002	4.399608e+002
25	1.039322e-002	3.418858e+003	4.187975e-003	2.259698e+003
81	2.386062e-003	1.726995e+006	4.895870e-004	3.650369e+005
289	4.904715e-005	1.706884e+010	2.668524e-005	5.328110e+009
1089	3.676576e-008	1.446865e+018	1.946954e-008	5.015917e+017

```
2  Lrbf = @(e,r) 4*e^2*exp(-(e*r).^2).*((e*r).^2-1);
```

Table 39.2 Non-symmetric collocation solution of Example 39.1 with Gaussians and IMQs, $\varepsilon = 3$ and uniform interior points and boundary centers outside the domain.

N (interior points)	Gaussian		IMQ	
	RMS-error	cond(A)	RMS-error	cond(A)
3×3	1.981675e-001	1.258837e+003	1.526456e-001	2.794516e+002
5×5	7.199931e-003	4.136193e+003	6.096534e-003	2.409431e+003
9×9	1.947108e-004	2.529708e+010	8.071271e-004	8.771630e+005
17×17	4.174290e-008	5.335000e+019	3.219110e-005	5.981238e+010
33×33	1.408750e-005	7.106505e+020	1.552047e-007	1.706638e+020

Several observations can be made by looking at Tables 39.1 and 39.2. The use of Halton points instead of uniform points seems to be beneficial since both the errors and the condition numbers are smaller (*c.f.* the right part of Table 39.1 vs. the right part of Table 39.2). Placement of the boundary centers outside the domain seems to be advantageous since again both the errors and the condition numbers decrease (*c.f.* Table 39.1). Also, the last row of Table 39.2 seems to indicate that Gaussians are more prone to ill-conditioning than inverse multiquadrics.

Of course, these are rather superficial observations based on only a few numerical experiments. For many of these claims there is no theoretical foundation, and many more experiments would be needed to make a more conclusive statement (for example, no attempt was made here to find the best approximations, *i.e.*, optimize the value of the shape parameter). Also, one could experiment with different values of the shape parameter on the boundary and in the interior (as suggested, *e.g.*, in [Kansa and Carlson (1992)]).

The collocation points and centers used here (and in most of the following examples) are displayed in the left plot of Figure 39.1, while the right plot contains a solution for $N = 289$ interior Halton points corresponding to row 4 in the right part of Table 39.1.

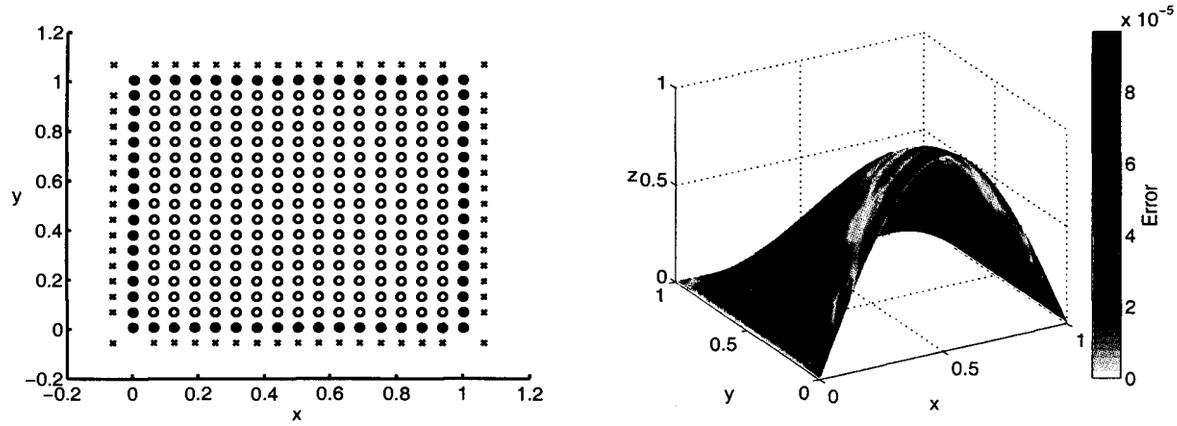


Fig. 39.1 Collocation points (interior: blue circles, boundary: red crosses) and centers (interior: blue circles, boundary: green crosses) (left) and non-symmetric RBF collocation solution (right) for Example 39.1 using IMQs with $\varepsilon = 3$ and $N = 289$ interior points.

Example 39.2. Consider the following elliptic equation with variable coefficients and homogeneous Dirichlet boundary conditions:

$$\frac{\partial}{\partial x}\left(a(x,y)\frac{\partial}{\partial x}u(x,y)\right) + \frac{\partial}{\partial y}\left(b(x,y)\frac{\partial}{\partial y}u(x,y)\right) = f(x,y), \quad (x,y) \in \Omega = [0,1]^2,$$
$$u(x,y) = 0, \quad (x,y) \in \Gamma = \partial\Omega,$$

where

$$f(x,y) = -16x(1-x)(3-2y)e^{x-y} + 32y(1-y)(3x^2+y^2-x-2),$$

and the coefficients are given by

$$a(x,y) = 2 - x^2 - y^2, \qquad b(x,y) = e^{x-y}.$$

As can easily be verified, the exact solution for this problem is given by

$$u(x,y) = 16x(1-x)y(1-y).$$

The corresponding MATLAB program is listed as Program 39.2. The definition section of this program (lines 1–9) is much longer than before since we need to work with first and second-order partial derivatives of the basic function. Also, the coefficients a and b and their partials are required.

While most of the remainder of the program is identical to the previous one, the assembly of the collocation matrix (lines 26–32) is much more involved since we need to apply the differential operator to the basis functions (see line 30 for the computation of the block LCM which corresponds to the block $\tilde{A}_{\mathcal{L}}$ in our earlier discussion (38.6)).

Program 39.2. KansaEllipticVC_2D.m

```
% KansaEllipticVC_2D
% Script that performs Kansa collocation for 2D elliptic PDE
```

```
% with variable coefficients
% Calls on: DistanceMatrix, DifferenceMatrix
    % IMQ RBF and its derivatives
 1  rbf = @(e,r) 1./sqrt(1+(e*r).^2); ep = 3;
 2  dxrbf = @(e,r,dx) -dx*e^2./(1+(e*r).^2).^(3/2);
 3  dyrbf = @(e,r,dy) -dy*e^2./(1+(e*r).^2).^(3/2);
 4a dxxrbf = @(e,r,dx) e^2*(3*(e*dx).^2-1-(e*r).^2)./...
 4b                        (1+(e*r).^2).^(5/2);
 5a dyyrbf = @(e,r,dy) e^2*(3*(e*dy).^2-1-(e*r).^2)./...
 5b                        (1+(e*r).^2).^(5/2);
    % Test problem input (right-hand side, coefficients)
 6  u = @(x,y) 16*x.*(1-x).*y.*(1-y);
 7a Lu = @(x,y) -16*x.*exp(x-y).*(1-x).*(3-2*y)+...
 7b            32*y.*(1-y).*(3*x.^2+y.^2-x-2);
 8  a = @(x,y) 2-x.^2-y.^2; ax = @(x,y) -2*x;
 9  b = @(x,y) exp(x-y); by = @(x,y)-exp(x-y);
10  N = 289; gridtype = 'h';
11  neval = 40;
    % Load (interior) collocation points
12  name = sprintf('Data2D_%d%s',N,gridtype); load(name);
13  intdata = dsites;
    % Additional boundary collocation points
14  sn = sqrt(N); bdylin = linspace(0,1,sn)';
15  bdy0 = zeros(sn-1,1); bdy1 = ones(sn-1,1);
16a bdydata = [bdylin(1:end-1) bdy0; bdy1 bdylin(1:end-1);...
16b      flipud(bdylin(2:end)) bdy1; bdy0 flipud(bdylin(2:end))];
    % Create additional boundary centers OUTSIDE the domain
17  h = 1/(sn-1); bdylin = (h:h:1-h)';
18  bdy0 = -h*ones(sn-2,1); bdy1 = (1+h)*ones(sn-2,1);
19a bdyctrs = [-h -h; bdylin bdy0; 1+h -h; bdy1 bdylin;...
19b     1+h 1+h; flipud(bdylin) bdy1; -h 1+h; bdy0 flipud(bdylin)];
20  ctrs = [intdata; bdyctrs];
    % Create neval-by-neval equally spaced evaluation locations
    % in the unit square
21  grid = linspace(0,1,neval); [xe,ye] = meshgrid(grid);
22  epoints = [xe(:) ye(:)];
    % Compute evaluation matrix
23  DM_eval = DistanceMatrix(epoints,ctrs);
24  EM = rbf(ep,DM_eval);
25  exact = u(epoints(:,1),epoints(:,2));
    % Compute blocks for collocation matrix
26  DM_intdata = DistanceMatrix(intdata,ctrs);
```

```
27  DM_bdydata = DistanceMatrix(bdydata,ctrs);
28  dx_intdata = Differencematrix(intdata(:,1),ctrs(:,1));
29  dy_intdata = Differencematrix(intdata(:,2),ctrs(:,2));
30a LCM = diag(ax(intdata(:,1))) * ...
30b       dxrbf(ep,DM_intdata,dx_intdata) + ...
30c       diag(a(intdata(:,1),intdata(:,2))) * ...
30d       dxxrbf(ep,DM_intdata,dx_intdata) + ...
30e       diag(by(intdata(:,1),intdata(:,2))) * ...
30f       dyrbf(ep,DM_intdata,dy_intdata) + ...
30g       diag(b(intdata(:,1),intdata(:,2))) * ...
30h       dyyrbf(ep,DM_intdata,dy_intdata);
31  BCM = rbf(ep,DM_bdydata);
32  CM = [LCM; BCM];
    % Create right-hand side
33  rhs = [Lu(intdata(:,1),intdata(:,2)); zeros(4*(sn-1),1)];
    % RBF solution
34  Pf = EM * (CM\rhs);
    % Compute maximum error on evaluation grid
35  maxerr = norm(Pf-exact,inf);
36  rms_err = norm(Pf-exact)/neval;
37  fprintf('RMS error:     %e\n', rms_err)
38  fprintf('Maximum error: %e\n', maxerr)
    % Plot collocation points and centers
39  hold on; plot(intdata(:,1),intdata(:,2),'bo');
40  plot(bdydata(:,1),bdydata(:,2),'rx');
41  plot(bdyctrs(:,1),bdyctrs(:,2),'gx'); hold off
42  fview = [-30,30];  % viewing angles for plot
43  PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview);
44  PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview);
```

In Table 39.3 we compare the solution obtained with Gaussians and inverse multiquadrics based on interior Halton points. The boundary centers are taken to lie outside the domain as in Figure 39.1. Again, the solution with inverse multiquadrics is slightly better conditioned. For Gaussians we need to replace lines 1–5 of Program 39.2 by

```
1  rbf = @(e,r) exp(-(e*r).^2); ep = 3;
2  dxrbf = @(e,r,dx) -2*dx*e^2.*exp(-(e*r).^2);
3  dyrbf = @(e,r,dy) -2*dy*e^2.*exp(-(e*r).^2);
4  dxxrbf = @(e,r,dx) 2*e^2*(2*(e*dx).^2-1).*exp(-(e*r).^2);
5  dyyrbf = @(e,r,dy) 2*e^2*(2*(e*dy).^2-1).*exp(-(e*r).^2);
```

The top part of Figure 39.2 contains plots of the approximate solution and maximum error for the inverse multiquadric solution based on $N = 289$ interior

Table 39.3 Solution of Example 39.2 with Gaussians and IMQs, $\varepsilon = 3$ and interior Halton points.

N (interior points)	Gaussian		IMQ	
	RMS-error	cond(A)	RMS-error	cond(A)
9	6.852103e-002	8.874341e+003	1.123770e-001	6.954910e+002
25	1.091888e-002	4.898291e+003	1.123575e-002	3.302471e+003
81	1.854386e-004	1.286993e+009	1.370992e-003	4.992219e+005
289	8.445637e-007	7.031011e+019	8.105109e-005	7.527456e+009
1089	2.559824e-005	4.553162e+020	7.041415e-008	7.785955e+017

and 64 boundary points.

Example 39.3. Consider the Poisson problem with mixed boundary conditions

$$\begin{aligned} \nabla^2 u(x,y) &= -5.4x, \quad (x,y) \in \Omega = [0,1]^2, \\ \frac{\partial}{\partial \boldsymbol{n}} u(x,y) &= 0, \quad (x,y) \in \Gamma_1 \cup \Gamma_3, \\ u(x,y) &= 0.1, \quad (x,y) \in \Gamma_2, \\ u(x,y) &= 1, \quad (x,y) \in \Gamma_4, \end{aligned}$$

where

$$\begin{aligned} \Gamma_1 &= \{(x,y) : \ 0 \le x \le 1, \ y = 0\}, \\ \Gamma_2 &= \{(x,y) : \ x = 1, \ 0 \le y \le 1\}, \\ \Gamma_3 &= \{(x,y) : \ 0 \le x \le 1, \ y = 1\}, \\ \Gamma_4 &= \{(x,y) : \ x = 0, \ 0 \le y \le 1\}. \end{aligned}$$

For this problem the exact solution is given by

$$u(x,y) = 1 - 0.9x^3.$$

Note that the normal derivative on the edges Γ_1 and Γ_3 is given by $\frac{\partial}{\partial y}$ and $-\frac{\partial}{\partial y}$, respectively. Therefore, for the MATLAB program we require the y-partial of the basic function in addition to its definition and its Laplacian (see lines 1–3 of Program 39.3). Again, the main difference in the code is in the assembly of the collocation matrix on lines 22–30. Note that this time we need to deal carefully with the boundary conditions and right-hand side (see lines 26–29 and 30). It is important that the orientation of the boundary points is consistent.

Program 39.3. `KansaLaplaceMixedBC_2D.m`

```
% KansaLaplaceMixedBC_2D
% Script that performs Kansa collocation for 2D Laplace equation
% with mixed BCs
% Calls on: DistanceMatrix, DifferenceMatrix
```

```
    % IMQ RBF and its Laplacian
 1  rbf = @(e,r) 1./sqrt(1+(e*r).^2); ep = 3;
 2  dyrbf = @(e,r,dy) -dy*e^2./(1+(e*r).^2).^(3/2);
 3  Lrbf = @(e,r) e^2*((e*r).^2-2)./(1+(e*r).^2).^(5/2);
    % Exact solution and its Laplacian for test problem
 4  u = @(x,y) 1-0.9*x.^3+0*y;
 5  Lu = @(x,y) -5.4*x+0*y;
    % Number and type of collocation points
 6  N = 289; gridtype = 'h';
 7  neval = 40;
    % Load (interior) collocation points
 8  name = sprintf('Data2D_%d%s',N,gridtype); load(name);
 9  intdata = dsites;
    % Additional boundary collocation points
10  sn = sqrt(N); bdylin = linspace(0,1,sn)';
11  bdy0 = zeros(sn-1,1); bdy1 = ones(sn-1,1);
12a bdydata = [bdylin(1:end-1) bdy0; bdy1 bdylin(1:end-1); ...
12b      flipud(bdylin(2:end)) bdy1; bdy0 flipud(bdylin(2:end))];
   % Create additional boundary centers OUTSIDE the domain
13  h = 1/(sn-1); bdylin = (h:h:1-h)';
14  bdy0 = -h*ones(sn-2,1); bdy1 = (1+h)*ones(sn-2,1);
15a bdyctrs = [-h -h; bdylin bdy0; 1+h -h; bdy1 bdylin; ...
15b     1+h 1+h; flipud(bdylin) bdy1; -h 1+h; bdy0 flipud(bdylin)];
16  ctrs = [intdata; bdyctrs];
    % Create neval-by-neval equally spaced evaluation locations
   % in the unit square
17  grid = linspace(0,1,neval); [xe,ye] = meshgrid(grid);
18  epoints = [xe(:) ye(:)];
    % Compute evaluation matrix
19  DM_eval = DistanceMatrix(epoints,ctrs);
20  EM = rbf(ep,DM_eval);
21  exact = u(epoints(:,1),epoints(:,2));
    % Compute blocks for collocation matrix
22  DM_intdata = DistanceMatrix(1ntdata,ctrs);
23  DM_bdydata = DistanceMatrix(bdydata,ctrs);
24  dy_bdydata = Differencematrix(bdydata(:,2),ctrs(:,2));
25  LCM = Lrbf(ep,DM_intdata);
26  BCM1 = -dyrbf(ep,DM_bdydata(1:sn-1,:),dy_bdydata(1:sn-1,:));
27  BCM2 = rbf(ep,DM_bdydata(sn:2*sn-2,:));
28a BCM3 = dyrbf(ep,DM_bdydata(2*sn-1:3*sn-3,:),...
28b              dy_bdydata(2*sn-1:3*sn-3,:));
29  BCM4 = rbf(ep,DM_bdydata(3*sn-2:end,:));
```

```
30  CM = [LCM; BCM1; BCM2; BCM3; BCM4];
    % Create right-hand side
31a rhs = [Lu(intdata(:,1),intdata(:,2)); zeros(sn-1,1); ...
31b        0.1*ones(sn-1,1); zeros(sn-1,1); ones(sn-1,1)];
    % RBF solution
32  Pf = EM * (CM\rhs);
    % Compute maximum error on evaluation grid
33  maxerr = norm(Pf-exact,inf);
34  rms_err = norm(Pf-exact)/neval;
35  fprintf('RMS error:     %e\n', rms_err)
36  fprintf('Maximum error: %e\n', maxerr)
    % Plot collocation points and centers
37  hold on; plot(intdata(:,1),intdata(:,2),'bo');
38  plot(bdydata(:,1),bdydata(:,2),'rx');
39  plot(bdyctrs(:,1),bdyctrs(:,2),'gx'); hold off
40  fview = [-30,30];  % viewing angles for plot
41  PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview);
42  PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview);
```

In Table 39.4 we again compare the use of Gaussians and inverse multiquadrics on a set of $N = 9, 25, 81, 289$ and 1089 interior Halton points (with additional boundary centers outside the domain). As in the previous experiments the Gaussian solution is slightly inferior in terms of stability for the same value of the shape parameter.

Table 39.4 Non-symmetric collocation solution of Example 39.3 with Gaussians and IMQs, $\varepsilon = 3$ and interior Halton points.

N	Gaussian		IMQ	
(interior points)	RMS-error	cond(A)	RMS-error	cond(A)
9	3.423330e-001	5.430073e+003	7.937403e-002	2.782348e+002
25	1.065826e-002	1.605086e+003	5.605445e-003	1.680888e+003
81	5.382387e-004	3.684159e+008	1.487160e-003	2.611650e+005
289	6.181855e-006	1.452124e+019	1.822077e-004	3.775455e+009
1089	2.060470e-006	1.628262e+021	1.822221e-007	3.155751e+017

In the bottom part of Figure 39.2 we show the inverse multiquadric solution for $N = 289$ interior points along with its maximum error. Note that (even though the problem has a symmetric solution) the approximate solution is not quite symmetric (as demonstrated by the error plot, *c.f.* also the top part of Figure 39.2).

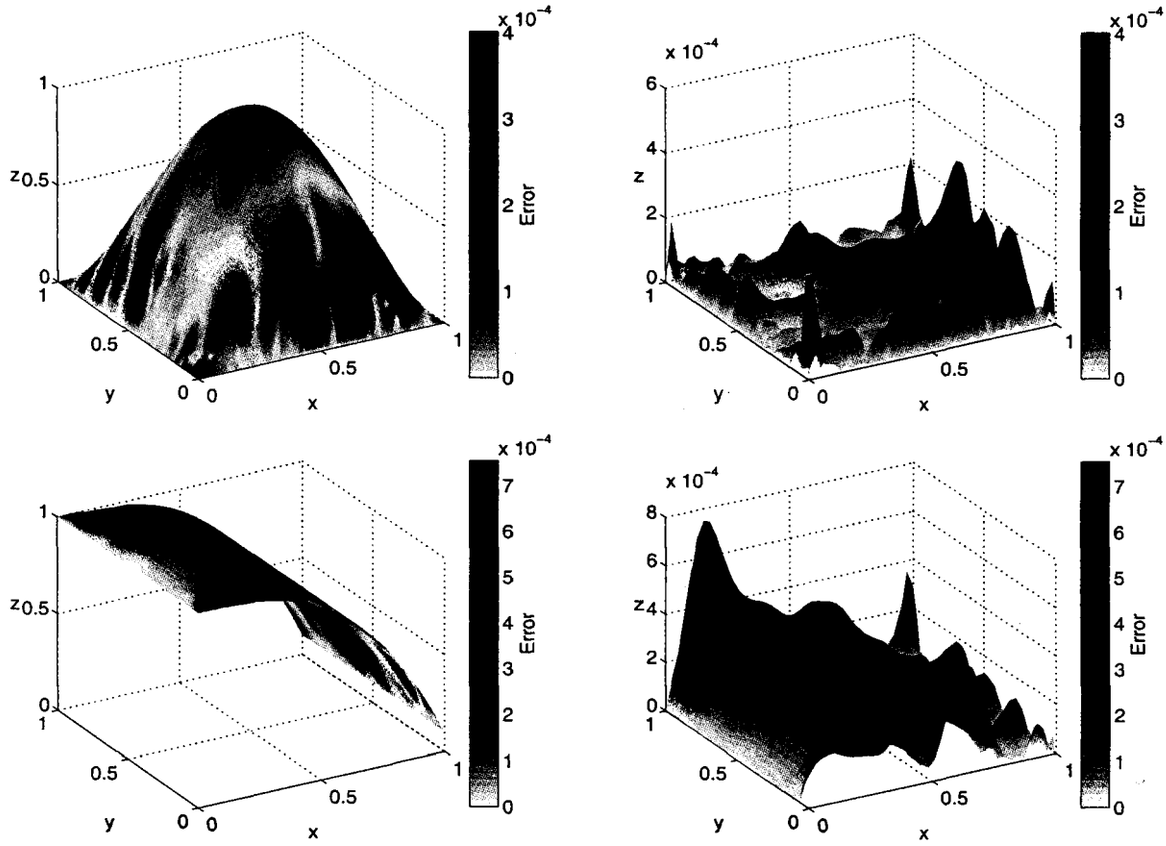


Fig. 39.2 Top: Non-symmetric collocation solution (left) and error plot (right) for Example 39.2 using IMQs with $\varepsilon = 3$ and $N = 289$ interior Halton points. Bottom: Approximate solution (left) and error plot (right) for Example 39.3 using IMQs with $\varepsilon = 3$ and $N = 289$ interior Halton points.

In [Li *et al.* (2003)] the authors report that the non-symmetric collocation solution for this problem with multiquadric RBFs is several orders of magnitude more accurate than a solution with piecewise linear finite elements using the same number of nodes.

Chapter 40

Symmetric RBF Collocation in MATLAB

In this chapter we discuss the implementation of the Hermite-based symmetric collocation method. Again, our discussion is fairly detailed with complete MATLAB code. As in the previous chapter we restrict ourselves to two-dimensional elliptic problems whose analytic solution is readily available and therefore can easily be verified. We will refer to a point $\boldsymbol{x}$ in $\mathbb{R}^2$ as (x, y).

40.1 Symmetric Collocation Method

For problems involving the Laplacian we now require also the differential operator

$$\begin{aligned}\nabla_{\boldsymbol{\xi}}^2\nabla^2 &= \left(\frac{\partial^2}{\partial\xi^2}+\frac{\partial^2}{\partial\eta^2}\right)\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)\\ &= \left(\frac{\partial^2}{\partial\xi^2}\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial\eta^2}\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial\xi^2}\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial\eta^2}\frac{\partial^2}{\partial y^2}\right)\\ &= \left(\frac{\partial^4}{\partial x^4}+2\frac{\partial^4}{\partial x^2y^2}+\frac{\partial^4}{\partial y^4}\right),\end{aligned}$$

where the simplification in the last line is justified since we are working with even-order derivatives. For example, using the chain rule with $r = \|\boldsymbol{x}-\boldsymbol{\xi}\|$ we get for various radial basis functions in $\mathbb{R}^2$:

$$\nabla_{\boldsymbol{\xi}}^2\nabla^2 e^{-(\varepsilon r)^2} = 16\varepsilon^4\left(2-4(\varepsilon r)^2+(\varepsilon r)^4\right)e^{-(\varepsilon r)^2},\ \text{Gaussian}, \tag{40.1}$$

$$\nabla_{\boldsymbol{\xi}}^2\nabla^2\frac{1}{\sqrt{1+(\varepsilon r)^2}} = \frac{3\varepsilon^4\left(3(\varepsilon r)^4-24(\varepsilon r)^2+8\right)}{\left(1+(\varepsilon r)^2\right)^{9/2}},\quad \text{IMQ}, \tag{40.2}$$

$$\nabla_{\boldsymbol{\xi}}^2\nabla^2\sqrt{1+(\varepsilon r)^2} = \frac{\varepsilon^4\left((\varepsilon r)^4+8(\varepsilon r)^2-8\right)}{\left(1+(\varepsilon r)^2\right)^{7/2}},\quad \text{MQ}. \tag{40.3}$$

More examples of RBFs and their derivatives are collected in Appendix D.

Example 40.1. We use the same PDE and boundary condition as in Example 39.1. A MATLAB program for symmetric Hermite-based collocation is given as Program 40.1. Note that this program is quite a bit more complicated than the

corresponding one for the non-symmetric collocation method (*c.f.* Program 39.1). The evaluation matrix EM now consists of two blocks (similar to the collocation matrix for the non-symmetric case, see lines 19–23), whereas the collocation matrix is assembled from four blocks (*c.f.* lines 25–33). Note that we now also require one of the iterated Laplacians of the basic function as listed in (40.1)–(40.3).

Program 40.1. HermiteLaplace_2D.m

```
% HermiteLaplace_2D
% Script that performs Hermite collocation for 2D Laplace equation
% Calls on: DistanceMatrix
    % IMQ RBF and its Laplacian and double Laplacian
 1  rbf = @(e,r) 1./sqrt(1+(e*r).^2); ep = 3;
 2  Lrbf = @(e,r) e^2*((e*r).^2-2)./(1+(e*r).^2).^(5/2);
 3a L2rbf = @(e,r) 3*e^4*(3*(e*r).^4-24*(e*r).^2+8)./...
 3b                        (1+(e*r).^2).^(9/2);
    % Exact solution and its Laplacian for test problem
 4  u = @(x,y) sin(pi*x).*cos(pi*y/2);
 5  Lu = @(x,y) -1.25*pi^2*sin(pi*x).*cos(pi*y/2);
    % Number and type of collocation points
 6  N = 289; gridtype = 'u';
 7  neval = 40;
    % Load (interior) collocation points
 8  name = sprintf('Data2D_%d%s',N,gridtype); load(name);
 9  intdata = dsites;
    % Additional (equally spaced) boundary collocation points
10  sn = sqrt(N); bdylin = linspace(0,1,sn)';
11  bdy0 = zeros(sn-1,1); bdy1 = ones(sn-1,1);
12a bdydata = [bdylin(1:end-1) bdy0; bdy1 bdylin(1:end-1); ...
12b      flipud(bdylin(2:end)) bdy1; bdy0 flipud(bdylin(2:end))];
    % Create additional boundary centers OUTSIDE the domain
13  h = 1/(sn-1); bdylin = (h:h:1-h)';
14  bdy0 = -h*ones(sn-2,1); bdy1 = (1+h)*ones(sn-2,1);
15a bdyctrs = [-h -h; bdylin bdy0; 1+h -h; bdy1 bdylin; ...
15b      1+h 1+h; flipud(bdylin) bdy1; -h 1+h; bdy0 flipud(bdylin)];
16  intctrs = intdata;
    % Create neval-by-neval equally spaced evaluation locations
    % in the unit square
17  grid = linspace(0,1,neval); [xe,ye] = meshgrid(grid);
18  epoints = [xe(:) ye(:)];
    % Compute evaluation matrix
19  DM_inteval = DistanceMatrix(epoints,intctrs);
20  LEM = Lrbf(ep,DM_inteval);
21  DM_bdyeval = DistanceMatrix(epoints,bdyctrs);
```

```
22  BEM = rbf(ep,DM_bdyeval);
23  EM = [LEM BEM];
24  exact = u(epoints(:,1),epoints(:,2));
    % Compute blocks for collocation matrix
25  DM_IIdata = DistanceMatrix(intdata,intctrs);
26  LLCM = L2rbf(ep,DM_IIdata);
27  DM_IBdata = DistanceMatrix(intdata,bdyctrs);
28  LBCM = Lrbf(ep,DM_IBdata);
29  DM_BIdata = DistanceMatrix(bdydata,intctrs);
30  BLCM = Lrbf(ep,DM_BIdata);
31  DM_BBdata = DistanceMatrix(bdydata,bdyctrs);
32  BBCM = rbf(ep,DM_BBdata);
33  CM = [LLCM LBCM; BLCM BBCM];
    % Create right-hand side
34a rhs = [Lu(intdata(:,1),intdata(:,2)); ...
34b        sin(pi*bdydata(1:sn-1,1)); zeros(3*(sn-1),1)];
    % Compute RBF solution
35  Pf = EM * (CM\rhs);
   % Compute maximum error on evaluation grid
36  maxerr = norm(Pf-exact,inf);
37  rms_err = norm(Pf-exact)/neval;
38  fprintf('RMS error:     %e\n', rms_err)
39  fprintf('Maximum error: %e\n', maxerr)
    % Plot collocation points and centers
40  hold on; plot(intdata(:,1),intdata(:,2),'bo');
41  plot(bdydata(:,1),bdydata(:,2),'rx');
42  plot(bdyctrs(:,1),bdyctrs(:,2),'gx'); hold off
43  fview = [-30,30];  % viewing angles for plot
44  PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview);
45  PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview);
```

As above we deal with the boundary by allowing the use of different collocation points and centers along the boundary. This causes the collocation matrix to be non-symmetric, and therefore the theoretical foundation of Chapter 38 no longer applies, *i.e.*, it is not clear that in this case the matrix is invertible. In order to work with a "safe" symmetric (and guaranteed invertible) matrix one should replace lines 13–15 with

```
bdyctrs = bdydata;
```

Note that, contrary to the non-symmetric Kansa approach, we can do this for both uniform and non-uniform interior points. In this case it is also possible to simplify the assembly of the collocation matrix. We can remove lines 29–30 and replace line 33 by

```
CM = [LLCM LBCM; LBCM' BBCM];
```

The same set of experiments as for the non-symmetric Kansa method (see Tables 39.1 and 39.2) are displayed in Tables 40.1 and 40.2 for the symmetric Hermite-based method.

Table 40.1 Symmetric collocation solution of Example 40.1 with IMQs, $\varepsilon = 3$ and Halton points.

N	centers on boundary		centers outside	
(interior points)	RMS-error	cond(A)	RMS-error	cond(A)
9	1.869505e-001	9.055720e+003	2.438041e-001	3.549895e+004
25	7.698471e-002	8.506782e+004	9.429580e-002	1.162027e+005
81	4.839682e-003	1.338599e+007	5.070833e-003	1.017388e+007
289	4.480250e-005	9.991615e+010	3.448546e-005	7.180249e+010
1089	2.481407e-008	2.820823e+018	1.907000e-008	2.262777e+018

We note that, as for the non-symmetric collocation method, inverse multiquadrics with interior Halton points and exterior boundary centers seems to perform overall slightly better than the other choices (*i.e.*, Gaussians, interior uniform points, or boundary centers on the boundary).

Table 40.2 Symmetric collocation solution of Example 40.1 with Gaussians and IMQs, $\varepsilon = 3$ and uniform points with boundary centers outside the domain.

N	Gaussian		IMQ	
(interior points)	RMS-error	cond(A)	RMS-error	cond(A)
3×3	4.088188e-001	1.196486e+005	2.806897e-001	3.105155e+004
5×5	7.704584e-003	1.359899e+005	1.583948e-001	1.216534e+005
9×9	2.272289e-004	2.453107e+010	8.650782e-004	2.016503e+007
17×17	5.271776e-008	4.338406e+021	3.962654e-005	6.051588e+011
33×33	5.805757e-007	1.438258e+022	1.870210e-007	2.324115e+020

It is remarkable, however, how small the difference in performance between the symmetric and non-symmetric approach is. This can be concluded by comparing the tables in Example 39.1 with those in Example 40.1. Also, Figure 40.1 shows error plots for the two methods using the same set of parameters, *i.e.*, inverse multiquadrics with $\varepsilon = 3$, $N = 289$ interior Halton points and 64 boundary points with the boundary centers placed outside the domain as in Figure 39.1.

The example above shows very high convergence rates as predicted by the estimate in Theorem 38.1 when using infinitely smooth inverse multiquadrics on a problem that has a smooth solution.

Example 40.2. A MATLAB implementation of the variable coefficient problem of Example 39.2, while theoretically possible, is very cumbersome using the symmetric

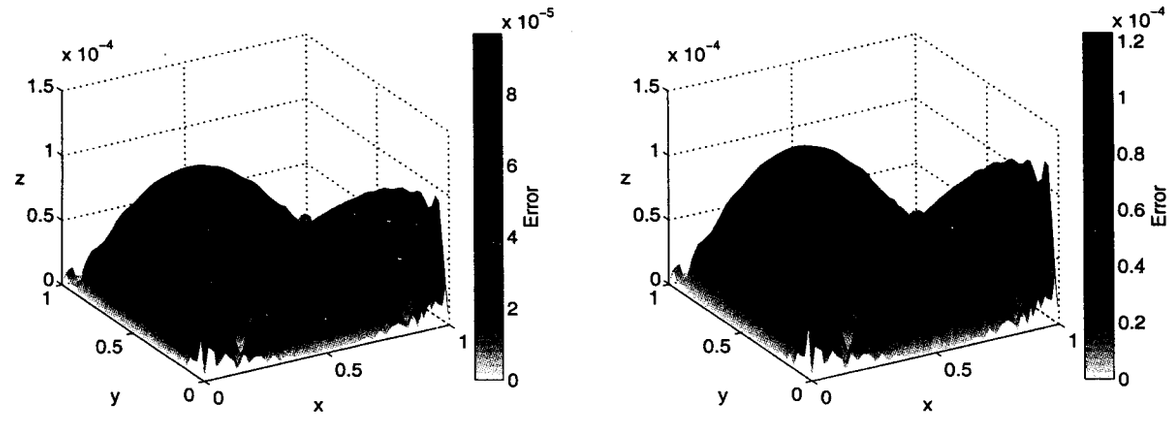


Fig. 40.1 Error plots for the collocation solution of Example 39.1 (Example 40.1) using IMQs with $\varepsilon = 3$ and $N = 289$ interior Halton points; boundary centers outside domain. Kansa's method (left) and symmetric method (right).

collocation method. For example, since the differential operator $\mathcal{L}$ is given by

$$\mathcal{L} = \frac{\partial}{\partial x}\left(a(x,y)\frac{\partial}{\partial x}\right) + \frac{\partial}{\partial y}\left(b(x,y)\frac{\partial}{\partial y}\right)$$

the basic expansion for the RBF solution is

$$\sum_{j=1}^{N_B} c_j \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}_j\|) + \sum_{j=N_B+1}^{N} c_j \mathcal{L}^{\boldsymbol{\xi}} \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}\|)|_{\boldsymbol{\xi}=\boldsymbol{\xi}_j},$$

with

$$\mathcal{L}^{\boldsymbol{\xi}} = \frac{\partial}{\partial \xi}\left(a(\xi,\eta)\frac{\partial}{\partial \xi}\right) + \frac{\partial}{\partial \eta}\left(b(\xi,\eta)\frac{\partial}{\partial \eta}\right).$$

This, however implies that the block $\hat{A}_{\mathcal{L}\mathcal{L}^{\boldsymbol{\xi}}}$ of the symmetric collocation matrix has entries computed with the differential operator

$$\begin{aligned}
\mathcal{L}\mathcal{L}^{\boldsymbol{\xi}} &= \left[\frac{\partial}{\partial x}\left(a(x,y)\frac{\partial}{\partial x}\right) + \frac{\partial}{\partial y}\left(b(x,y)\frac{\partial}{\partial y}\right)\right]\left[\frac{\partial}{\partial \xi}\left(a(\xi,\eta)\frac{\partial}{\partial \xi}\right) + \frac{\partial}{\partial \eta}\left(b(\xi,\eta)\frac{\partial}{\partial \eta}\right)\right] \\
&= a(x,y)a(\xi,\eta)\frac{\partial^4}{\partial x^4} + (a(x,y)b(\xi,\eta) + b(x,y)a(\xi,\eta))\frac{\partial^4}{\partial x^2 \partial y^2} + \\
&\quad b(x,y)b(\xi,\eta)\frac{\partial^4}{\partial y^4} + \left(\frac{\partial a(x,y)}{\partial x}a(\xi,\eta) - a(x,y)\frac{\partial a(\xi,\eta)}{\partial \xi}\right)\frac{\partial^3}{\partial x^3} + \\
&\quad \left(\frac{\partial b(x,y)}{\partial y}a(\xi,\eta) - a(x,y)\frac{\partial b(\xi,\eta)}{\partial \eta}\right)\frac{\partial^3}{\partial x^2 \partial y} + \\
&\quad \left(\frac{\partial a(x,y)}{\partial x}b(\xi,\eta) - b(x,y)\frac{\partial a(\xi,\eta)}{\partial \xi}\right)\frac{\partial^3}{\partial x \partial y^2} + \\
&\quad \left(\frac{\partial b(x,y)}{\partial y}b(\xi,\eta) - b(x,y)\frac{\partial b(\xi,\eta)}{\partial \eta}\right)\frac{\partial^3}{\partial y^3} + \frac{\partial a(x,y)}{\partial x}\frac{\partial a(\xi,\eta)}{\partial \xi}\frac{\partial^2}{\partial x^2} - \\
&\quad \left(\frac{\partial a(x,y)}{\partial x}\frac{\partial b(\xi,\eta)}{\partial \eta} + \frac{\partial b(x,y)}{\partial y}\frac{\partial a(\xi,\eta)}{\partial \xi}\right)\frac{\partial^2}{\partial x \partial y} + \frac{\partial b(x,y)}{\partial y}\frac{\partial b(\xi,\eta)}{\partial \eta}\frac{\partial^2}{\partial y^2}.
\end{aligned}$$

Here we expressed derivatives with respect to the second variable $\boldsymbol{\xi} = (\xi, \eta)$ of the basic function in terms of those with respect to the first variable $\boldsymbol{x} = (x, y)$ remembering that every differentiation introduces a sign change (*c.f.* the discussion at the end of Chapter 36).

Example 40.3. Instead of repeating the calculations for Example 39.3, we present a different problem with piecewise defined boundary conditions.

$$\begin{aligned} \nabla^2 u(x,y) &= 0, \quad (x,y) \in \Omega = (-1,1)^2, \\ u(x,y) &= 0, \quad (x,y) \in \Gamma_1 \cup \Gamma_3 \cup \Gamma_5, \\ u(x,y) &= \frac{1}{5}\sin(3\pi y), \quad (x,y) \in \Gamma_2, \\ u(x,y) &= \sin^4(\pi x), \quad (x,y) \in \Gamma_4, \end{aligned}$$

where

$$\begin{aligned} \Gamma_1 &= \{(x,y) : \ -1 \le x \le 1, \ y = -1\}, \\ \Gamma_2 &= \{(x,y) : \ x = 1, \ -1 \le y \le 1\}, \\ \Gamma_3 &= \{(x,y) : \ 0 \le x \le 1, \ y = 1\}, \\ \Gamma_4 &= \{(x,y) : \ -1 \le x \le 0, \ y = 1\}, \\ \Gamma_5 &= \{(x,y) : \ x = -1, \ 0 \le y \le 1\}. \end{aligned}$$

For this problem we do not have an exact solution available. However, this problem is taken from [Trefethen (2000)] and we use the pseudospectral solution from there for comparison. We will revisit this problem later when we discuss RBF-PS methods in Chapter 42.

Program 40.2. `HermiteLaplaceMixedBCTref_2D.m`

```
% HermiteLaplaceMixedBCTref_2D
% Script that performs Hermite collocation for 2D Laplace equation
% Note: Prog 36 in Trefethen (2000), exact solution not provided
% Calls on: DistanceMatrix
    % IMQ RBF and its Laplacian
 1  rbf = @(e,r) 1./sqrt(1+(e*r).^2); ep = 3;
 2  Lrbf = @(e,r) e^2*((e*r).^2-2)./(1+(e*r).^2).^(5/2);
 3a L2rbf = @(e,r) 3*e^4*(3*(e*r).^4-24*(e*r).^2+8)./...
 3b                      (1+(e*r).^2).^(9/2);
    %  Laplacian for test problem
 4  Lu = @(x,y) zeros(size(x));
    % Number and type of collocation points
 5  N = 289; gridtype = 'u';
 6  neval = 41;
    % Load (interior) collocation points
 7  name = sprintf('Data2D_%d%s',N,gridtype); load(name);
```

```
 8  intdata = 2*dsites-1;
    % Additional boundary collocation points
 9  sn = sqrt(N); bdylin = linspace(-1,1,sn)';
10  bdy1 = ones(sn-1,1);
11a bdydata = [bdylin(1:end-1) -bdy1; bdy1 bdylin(1:end-1); ...
11b      flipud(bdylin(2:end)) bdy1; -bdy1 flipud(bdylin(2:end))];
    % Create additional boundary centers OUTSIDE the domain
12  h = 2/(sn-1); bdylin = (-1+h:h:1-h)';
13  bdy0 = repmat(-1-h,sn-2,1); bdy1 = repmat(1+h,sn-2,1);
14a bdyctrs = [-1-h -1-h; bdylin bdy0; 1+h -1-h; bdy1 bdylin; ...
14b     1+h 1+h; flipud(bdylin) bdy1; -1-h 1+h; bdy0 flipud(bdylin)];
15  intctrs = intdata;
    % Create neval-by-neval equally spaced evaluation locations
    % in the unit square
16  grid = linspace(-1,1,neval); [xe,ye] = meshgrid(grid);
17  epoints = [xe(:) ye(:)];
    % Compute evaluation matrix
18  DM_inteval = DistanceMatrix(epoints,intctrs);
19  LEM = Lrbf(ep,DM_inteval);
20  DM_bdyeval = DistanceMatrix(epoints,bdyctrs);
21  BEM = rbf(ep,DM_bdyeval);
22  EM = [LEM BEM];
    % Compute blocks for collocation matrix
23  DM_IIdata = DistanceMatrix(intdata,intctrs);
24  LLCM = L2rbf(ep,DM_IIdata);
25  DM_IBdata = DistanceMatrix(intdata,bdyctrs);
26  LBCM = Lrbf(ep,DM_IBdata);
27  DM_BIdata = DistanceMatrix(bdydata,intctrs);
28  BLCM = Lrbf(ep,DM_BIdata);
29  DM_BBdata = DistanceMatrix(bdydata,bdyctrs);
30  BBCM = rbf(ep,DM_BBdata);
31  CM = [LLCM LBCM; BLCM BBCM];
    % Create right-hand side
32a rhs = [Lu(intdata(:,1),intdata(:,2)); zeros(sn-1,1); ...
32b       0.2*sin(3*pi*bdydata(sn:2*sn-2,2)); zeros((sn-1)/2,1);...
32c       sin(pi*bdydata((5*sn-3)/2:3*sn-3,1)).^4; zeros(sn-1,1)];
    % Compute RBF solution
33  Pf = EM * (CM\rhs);
34  surf(xe,ye,reshape(Pf,neval,neval));
35  view(-20,45),  axis([-1 1 -1 1 -.2 1]);
36  text(0,.8,.5,sprintf('u(0,0) = %12.10f',Pf(841)))
```

The definition of the boundary conditions in the MATLAB code for Program 40.2 is similar to that for Program 39.3. However, now we are working on the square $[-1, 1]^2$ instead of $[0, 1]^2$, and therefore slight adjustments are required. For example, the collocation points we load from file are now transformed on line 8. Also, the boundary centers have to be offset from a different boundary (see lines 12–14).

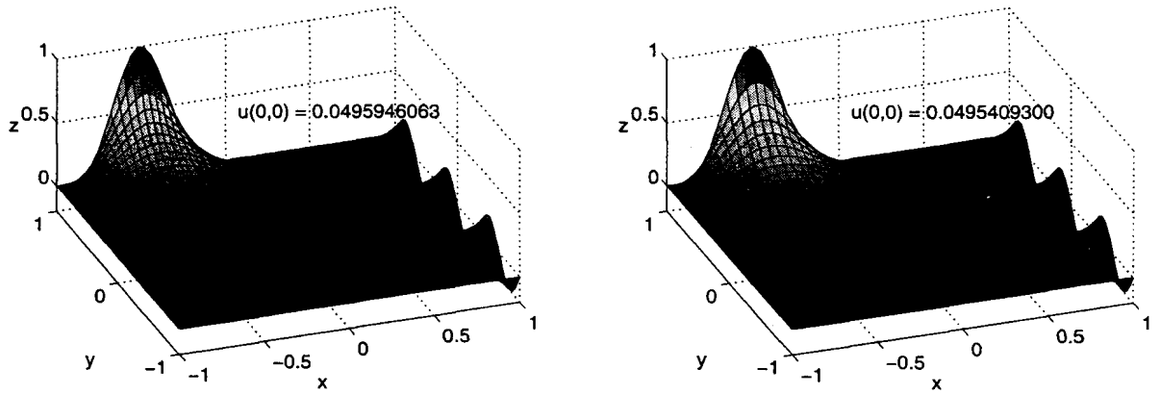


Fig. 40.2 Plots for solution of Example 40.3 using pseudospectral method with 361 points from [Trefethen (2000)] (left), and with symmetric collocation using IMQs with $\varepsilon = 3$ and $N = 289$ uniform interior points; 64 additional boundary centers outside domain.

We note that the quality of the two solutions displayed in Figure 40.2 is quite similar. The total number of points used for the PS solution is 361, while 353 points (289 interior plus 64 boundary) are used for the RBF solution.

40.2 Summarizing Remarks on the Symmetric and Non-Symmetric Collocation Methods

All in all, the non-symmetric (Kansa) method seems to perform just a little bit better than the symmetric (Hermite) method (compare Tables 39.1 and 39.2 with Tables 40.1 and 40.2). For the same value of the shape parameter ε the errors as well as the condition numbers are slightly smaller. This does not agree with the findings in [Larsson and Fornberg (2003)] where the authors concluded that the symmetric method is more accurate (see also our discussion at the end of Chapter 38).

An advantage of the Hermite approach over Kansa's method is that the collocation matrices resulting from the Hermite approach are symmetric if all of the centers coincide with the collocation points. Therefore the amount of computation can be reduced considerably by using a solver for symmetric systems. Since Kansa's method requires fewer derivatives of the basic function it has the added advantages of being simpler to implement and applicable to problems with less smooth solutions. Moreover, as we saw in Examples 39.2 and 40.2, the non-symmetric method is much simpler for problems with non-constant coefficients. Further-

more, it is not clear how to deal with non-linear problems using the symmetric method. For a treatment of non-linear PDEs based on the non-symmetric collocation method within an operator Newton framework see [Bernal and Kindelan (2006); Fasshauer (2001a)].

Another contraposition of the two methods will be presented in the context of pseudospectral methods in Chapter 42.

Both of the methods described in this section have been implemented for many different applications. Comparisons of the two methods were reported in, *e.g.*, [Fasshauer (1997); Larsson and Fornberg (2003); Power and Barraco (2002)].

Chapter 41

Collocation with CSRBFs in MATLAB

In this third chapter describing the MATLAB implementation of RBF collocation methods we look at how compactly supported functions can be used in both a direct approach and within a multilevel framework. As in the previous two chapters we present only two-dimensional elliptic problems and will refer to a point $\boldsymbol{x}$ in $\mathbb{R}^2$ as (x, y).

41.1 Collocation with Compactly Supported RBFs

While Kansa initially proposed the non-symmetric collocation method for multiquadrics, the general method applies to any kind of RBF including those with compact support. The same goes for the symmetric method. We now present MATLAB code for the symmetric collocation method based on Wendland's C^6 function $\varphi_{3,3}(r) = (1-r)_+^8(32r^3 + 25r^2 + 8r + 1)$. Its Laplacian and biharmonic derivatives are given by

$$\nabla^2\varphi_{3,3}(r) = 44(1-r)_+^6(88r^3 + 3r^2 - 6r - 1),$$
$$\nabla^4\varphi_{3,3}(r) = 1056(1-r)_+^4(297r^3 - 212r^2 + 16r + 4).$$

Note, however, that in order for us to be able to take advantage of the subroutine `DistanceMatrixCSRBF.m` we provided earlier in Program 12.1 we need to represent the basic function and its derivatives in the shifted form

$$\widetilde{\varphi}_{3,3}(r) = r^8(66 - 154r + 121r^2 - 32r^3),$$
$$\nabla^2\widetilde{\varphi}_{3,3}(r) = 44r^6(84 - 264r + 267r^2 - 88r^3),$$
$$\nabla^4\widetilde{\varphi}_{3,3}(r) = 1056r^4(105 - 483r + 679r^2 - 297r^3),$$

as implemented on lines 1–3 of Program 41.1. While we technically do not need to include a scale factor ε in the MATLAB code for the basic function (since the support size is already used to determine the matrix entries in Program 12.1), the derivatives of the basic function still require the scale factor which appears as a consequence of the chain rule (see lines 2 and 3).

Another point to reconsider in the compact support setting is the placement of the boundary centers. While we saw for globally supported basic functions that

it was actually beneficial to place some centers outside the domain, this no longer makes much sense if we decide to use compactly supported functions. Clearly, any basic function whose support radius is smaller than the distance of its center from the boundary of the domain will not contribute to the solution of the problem. Therefore, we now use interior Halton points augmented by equally spaced boundary points for both the collocation points and the centers. This change is reflected on lines 14 and 15 of Program 41.1. Otherwise, Program 41.1 is essentially identical to Program 40.1. However, for the convenience of the reader we decided to print the entire program for the compactly supported case, also.

Program 41.1. `HermiteLaplace_2D_CSRBF.m`

```
% HermiteLaplace_2D_CSRBF
% Script that performs Hermite collocation for 2D Laplace equation
% with sparse matrices
% Calls on: DistanceMatrixCSRBF
    % Wendland C6 RBF, its Laplacian and double Laplacian
 1  rbf = @(e,r) r.^8.*(66*spones(r)-154*r+121*r.^2-32*r.^3);
 2  Lrbf = @(e,r) 44*e^2*r.^6.*(84*spones(r)-264*r+267*r.^2-88*r.^3);
 3a L2rbf = @(e,r) 1056*e^4*r.^4.*...
 3b                (105*spones(r)-483*r+679*r.^2-297*r.^3);
 4  ep = 0.25;
    % Exact solution and its Laplacian for test problem
 5  u = @(x,y) sin(pi*x).*cos(pi*y/2);
 6  Lu = @(x,y) -1.25*pi^2*sin(pi*x).*cos(pi*y/2);
    % Number and type of collocation points
 7  N = 289; gridtype = 'h';
 8  neval = 40;
    % Load (interior) collocation points
 9  name = sprintf('Data2D_%d%s',N,gridtype); load(name);
10  intdata = dsites;
    % Additional (equally spaced) boundary collocation points
11  sn = sqrt(N); bdylin = linspace(0,1,sn)';
12  bdy0 = zeros(sn-1,1); bdy1 = ones(sn-1,1);
13a bdydata = [bdylin(1:end-1) bdy0; bdy1 bdylin(1:end-1); ...
13b      flipud(bdylin(2:end)) bdy1; bdy0 flipud(bdylin(2:end))];
    % Let centers coincide with ALL data sites
14  bdyctrs = bdydata;
15  ctrs = [intdata; bdyctrs];
    % Create neval-by-neval equally spaced evaluation locations
    % in the unit square
16  grid = linspace(0,1,neval); [xe,ye] = meshgrid(grid);
17  epoints = [xe(:) ye(:)];
```

```
    % Compute evaluation matrix
18  DM_inteval = DistanceMatrixCSRBF(epoints,intdata,ep);
19  DM_bdyeval = DistanceMatrixCSRBF(epoints,bdyctrs,ep);
20  LEM = Lrbf(ep,DM_inteval);
21  BEM = rbf(ep,DM_bdyeval);
22  EM = [LEM BEM];
23  exact = u(epoints(:,1),epoints(:,2));
    % Compute blocks for collocation matrix
24  DM_IIdata = DistanceMatrixCSRBF(intdata,intdata,ep);
25  DM_IBdata = DistanceMatrixCSRBF(intdata,bdyctrs,ep);
26  DM_BIdata = DistanceMatrixCSRBF(bdydata,intdata,ep);
27  DM_BBdata = DistanceMatrixCSRBF(bdydata,bdyctrs,ep);
28  LLCM = L2rbf(ep,DM_IIdata);
29  LBCM = Lrbf(ep,DM_IBdata);
30  BLCM = Lrbf(ep,DM_BIdata);
31  BBCM = rbf(ep,DM_BBdata);
32  CM = [LLCM LBCM; BLCM BBCM];
    % Create right-hand side
33a rhs = [Lu(intdata(:,1),intdata(:,2)); ...
33b       sin(pi*bdydata(1:sn-1,1)); zeros(3*(sn-1),1)];
    % Compute RBF solution
34  Pf = EM * (CM\rhs);
    % Compute maximum error on evaluation grid
35  maxerr = norm(Pf-exact,inf);
36  rms_err = norm(Pf-exact)/neval;
37  fprintf('RMS error:     %e\n', rms_err)
38  fprintf('Maximum error: %e\n', maxerr)
    % Plot collocation points and centers
39  hold on; plot(intdata(:,1),intdata(:,2),'bo');
40  plot(bdydata(:,1),bdydata(:,2),'rx');
41  plot(bdyctrs(:,1),bdyctrs(:,2),'gx'); hold off
42  fview = [-30,30];  % viewing angles for plot
43  PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview);
44  PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview);
```

If we want to replace the symmetric collocation method in Program 41.1 by the non-symmetric one, then lines 18–32 need to be replaced by

```
% Compute evaluation matrix
DM_eval = DistanceMatrixCSRBF(epoints,ctrs,ep);
EM = rbf(ep,DM_eval);
exact = u(epoints(:,1),epoints(:,2));
% Compute blocks for collocation matrix
```

```
DM_intdata = DistanceMatrixCSRBF(intdata,ctrs,ep);
DM_bdydata = DistanceMatrixCSRBF(bdydata,ctrs,ep);
LCM = Lrbf(ep,DM_intdata);
BCM = rbf(ep,DM_bdydata);
CM = [LCM; BCM];
```

Example 41.1. We use the test problem of Examples 39.1 and 40.1. However, this time we compare a stationary approximation scheme to a non-stationary one for both the non-symmetric and the symmetric collocation method. In the stationary setting we take an initial parameter value of $\varepsilon = 0.25$ (*i.e.*, very wide basis functions that cover the entire domain), and then double its value for every successive experiment. This keeps the "bandwidth" of the collocation matrix fixed and results in an efficient approximation method. However, as for scattered data interpolation (*c.f.* Table 12.1) we should not expect convergence for $h \to 0$ in this stationary setting. This can be seen clearly in the left part of Tables 41.1 and 41.2. In fact, we can observe that for the collocation approach things are even worse than for interpolation. Not only is there no convergence for $h \to 0$, the errors actually increase. This is especially pronounced for the symmetric method.

Table 41.1 Non-symmetric collocation solution of Example 40.1 with CSRBFs, stationary ε (initial value $\varepsilon = 0.25$) and non-stationary $\varepsilon = 0.25$. N interior Halton points, $4(\sqrt{N}-1)$ equally spaced boundary centers coinciding with boundary collocation points.

N	stationary		non-stationary	
(interior points)	RMS-error	cond(A)	RMS-error	cond(A)
3×3	6.077025e-003	1.495127e+005	6.077025e-003	1.495127e+005
5×5	2.352498e-003	6.274058e+005	3.810928e-004	2.720833e+007
9×9	1.947271e-003	6.192333e+006	4.430301e-005	2.851716e+010
17×17	1.326745e-003	6.113189e+007	2.200286e-006	2.293235e+013
33×33	5.703309e-003	1.824487e+008	9.986944e-008	8.537419e+015

Table 41.2 Symmetric collocation solution of Example 40.1 with CSRBFs, stationary ε (initial value $\varepsilon = 0.25$) and non-stationary $\varepsilon = 0.25$. N interior Halton points, $4(\sqrt{N}-1)$ equally spaced boundary centers coinciding with boundary collocation points.

N	stationary		non-stationary	
(interior points)	RMS-error	cond(A)	RMS-error	cond(A)
3×3	5.866837e-003	4.448249e+004	5.866837e-003	4.448249e+004
5×5	3.190108e-003	2.454493e+006	4.757992e-004	3.582872e+007
9×9	8.381144e-003	8.693440e+007	3.825086e-005	2.360746e+010
17×17	1.115179e-001	2.162078e+009	2.099321e-006	1.392838e+013
33×33	3.696962e-001	3.164425e+010	8.680882e-008	6.127122e+015

In the non-stationary setting we observe convergence whose rate is remarkably similar for both approaches (*c.f.* the right part of Tables 41.1 and 41.2). However, the collocation matrices are now completely dense, and therefore this approach — as in the case of interpolation — defies the use of compactly supported functions.

It is also interesting to note that we can see that the accuracy obtained with the C^6 Wendland functions (in "global mode") is similar to that of the globally supported C^∞ Gaussians and inverse multiquadrics used in Example 39.1 — an indication that the solution to the PDE does not lie in the native space of the Gaussians or inverse multiquadrics.

On the other hand, if the basic functions are chosen too local (to keep the method efficient), then the boundary information can not penetrate to the inside of the problem. This is essentially what happens in the stationary setting. Figure 41.1 shows fits with $\varepsilon = 2$ (small support) and with $\varepsilon = 0.25$ (large support) for $N = 289$ interior Halton points plus 64 equally spaced boundary points corresponding to the fourth row in Tables 41.1 and 41.2. As indicated above, the centers are chosen to coincide with the collocation points. The collocation matrix in the $\varepsilon = 2$ case is sparse and has only 11% nonzero entries when using the symmetric method and 43% for the non-symmetric method. The rather significant difference in the sparsity patterns of the symmetric and non-symmetric methods is due to the fact that the entries for the symmetric matrix are given by higher-order derivatives of the basic function than those in the non-symmetric case. While the derivatives of φ theoretically retain the same support as φ, numerically the size of the support appears to shrink with increasing differentiation. Thus the resulting approximation in the symmetric stationary case is much poorer because the basis functions and their derivatives are too local and the boundary information in prevented from traveling across the domain. Similar observations were reported in [Jumarhon *et al.* (2000); Fasshauer (1999d)].

In [Fasshauer (1999d)] use of a diagonal (Jacobi) preconditioner was proposed to speed up the convergence of the conjugate gradient method used there to solve the linear system. However, the *accuracy* of the method does not benefit from this measure and therefore we do not pursue the idea any further.

The experiments above, as well as those reported in [Jumarhon *et al.* (2000)] using Wendland's C^4 compactly supported RBF $\varphi_{3,2}$, indicate that the error bounds of Theorem 38.1 may be too pessimistic. For elliptic problems and C^6 basis functions the theorem predicts an error on the order of $\mathcal{O}(h)$ while the numerical experiments in Table 41.2 suggest an order of about $\mathcal{O}(h^3)$ for the non-stationary setting.

Even more than in the interpolation setting, for the numerical solution of PDEs with compactly supported RBFs we need to use a multilevel approach to have the potential to combine efficiency with accuracy. A coarse solution with wide functions will have to provide an initial fit that captures the main features of the solution, and then more refined residual updates can improve this initial solution locally.

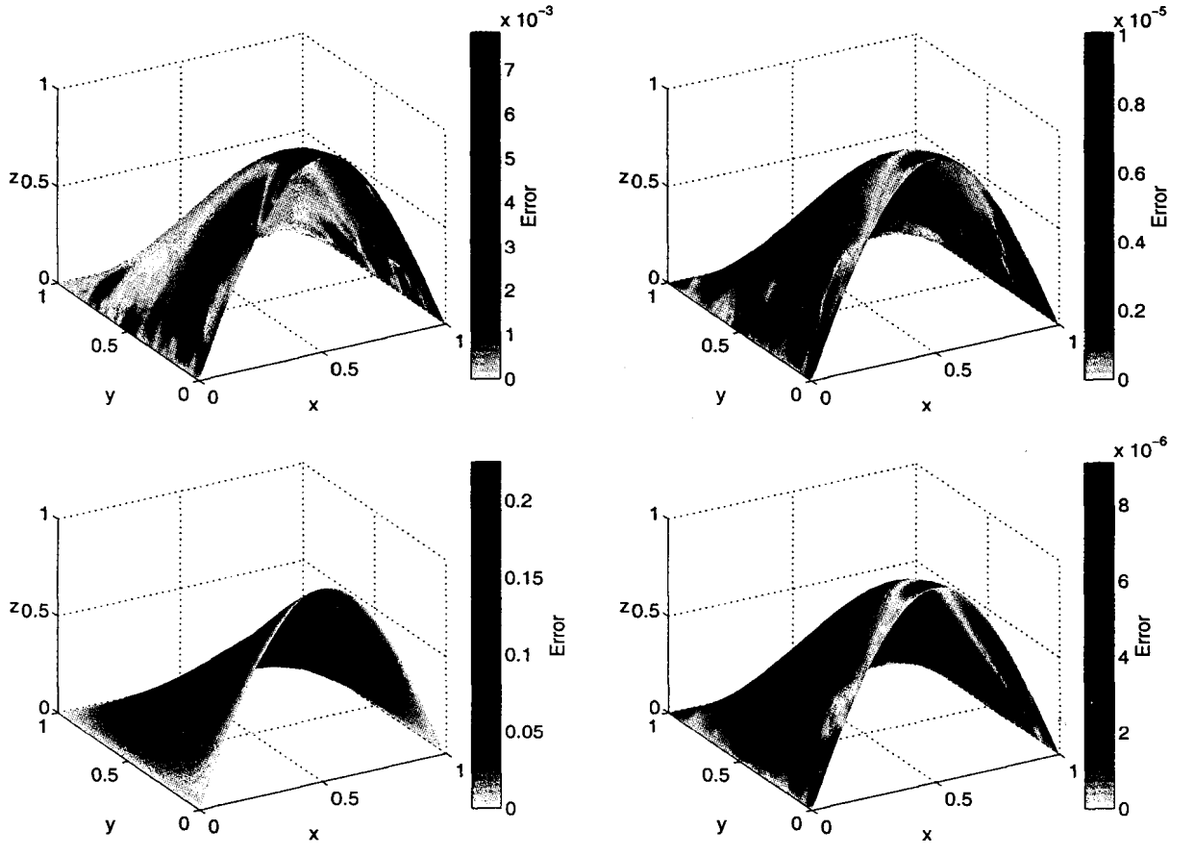


Fig. 41.1 Plots for solution of Example 40.1 with non-symmetric (top) and symmetric (bottom) collocation using CSRBFs with $\varepsilon = 2$ (left) and $\varepsilon = 0.25$ (right) and $N = 289$ interior Halton points; boundary centers coincide with equally spaced boundary collocation points.

41.2 Multilevel RBF Collocation

We end the discussion of the collocation approach by looking at a multilevel implementation of RBF collocation with compactly supported functions.

The most significant difference between the use of compactly supported RBFs for scattered data interpolation and for the numerical solution of PDEs by collocation appears when we turn to the multilevel approach. Recall that the use of the multilevel method is motivated by our desire to obtain a convergent scheme while at the same time keeping the bandwidth fixed, and thus the computational complexity at $\mathcal{O}(N)$.

Here is an adaptation of the stationary multilevel algorithm of Chapter 32 to the case of a collocation solution of the linear problem $\mathcal{L}u = f$:

Algorithm 41.1. Stationary Multilevel Collocation

(1) Let $u_0 = 0$
(2) For $k = 1, 2, \ldots, K$ do
  (a) Find $u_k \in \mathcal{S}_{\mathcal{X}_k}$ such that $\mathcal{L}u_k = f - \mathcal{L}u_{k-1}$ on grid $\mathcal{X}_k$

(b) Update $u_k \leftarrow u_{k-1} + u_k$

Here $\mathcal{S}_{\mathcal{X}_k}$ is the space of functions used for expansion (38.5) for the non-symmetric method or (38.7) for the symmetric method on grid $\mathcal{X}_k$. Note that the operator $\mathcal{L}$ encodes both the differential equation and the boundary condition.

Whereas we noted in Chapter 32 that there is strong numerical (and limited theoretical) evidence that the stationary multilevel interpolation algorithm converges at least linearly, the following example shows that *we cannot in general expect the stationary multilevel collocation algorithm to converge at all.*

Example 41.2. Once more we take the same test problem as in Examples 39.1, 40.1, and 41.1. As for the numerical experiments in the previous example, we let the boundary centers coincide with the boundary collocation points (see line 25 of Program 41.2 below).

An important difference between the multilevel interpolation code of Chapter 32 and the code presented here for the collocation solution of PDEs lies in the computation of the residuals. Note that in the interpolation setting the residual is of the form $f - \mathcal{P}_f$, while for PDEs we have $f - \mathcal{L}u_{k-1}$. Thus, the *evaluation* matrix in the interpolation setting is formed directly from the basis functions, while in the collocation setting (in both the symmetric and non-symmetric case) the evaluation matrix for the residuals is formed using the *derivatives* of the basis functions. These differences can be seen by comparing lines 23–33 of Program 32.1 with lines 39–56 of Program 41.2 below.

Program 41.2. ML_HermiteLaplaceCSRBF2D.m

```
% ML_HermiteLaplaceCSRBF2D
% Script that performs symmetric multilevel RBF collocation
% using sparse matrices
% Calls on: DistanceMatrixCSRBF
    % Wendland C6 RBF, its Laplacian and double Laplacian
 1  rbf = @(e,r) r.^8.*(66*spones(r)-154*r+121*r.^2-32*r.^3);
 2  Lrbf = @(e,r) 44*e^2*r.^6.*(84*spones(r)-264*r+267*r.^2-88*r.^3);
3a L2rbf = @(e,r) 1056*e^4*r.^4.*...
3b                 (105*spones(r)-483*r+679*r.^2-297*r.^3);
    % Exact solution and its Laplacian for test problem
 4  u = @(x,y) sin(pi*x).*cos(pi*y/2);
 5  Lu = @(x,y) -1.25*pi^2*sin(pi*x).*cos(pi*y/2);
 6  K = 6; neval = 40; gridtype = 'h';
 7  ep = 0.5*2.^[0:K-1];
    % Create neval-by-neval equally spaced evaluation locations
    % in the unit square
 8  grid = linspace(0,1,neval); [xe,ye] = meshgrid(grid);
 9  epoints = [xe(:) ye(:)];
```

```
    % Compute exact solution
10  exact = u(epoints(:,1),epoints(:,2));
11  Rf_old = zeros(17,1);
12  for k=1:K
13     N1 = (2^k+1)^2;    N2 = (2^(k+1)+1)^2;
14     name1 = sprintf('Data2D_%d%s',N1,gridtype);
15     name2 = sprintf('Data2D_%d%s',N2,gridtype);
16     load(name2)
       % Additional boundary points for residual evaluation
17     sn = sqrt(N2); bdylin = linspace(0,1,sn)';
18     bdy0 = zeros(sn-1,1); bdy1 = ones(sn-1,1);
19a    bdyres = [bdylin(1:end-1) bdy0; bdy1 bdylin(1:end-1); ...
19b         flipud(bdylin(2:end)) bdy1; bdy0 flipud(bdylin(2:end))];
20     intres = dsites;
21     load(name1); intdata = dsites;
       % Additional boundary points
22     sn = sqrt(N1); bdylin = linspace(0,1,sn)';
23     bdy0 = zeros(sn-1,1); bdy1 = ones(sn-1,1);
24a    bdydata = [bdylin(1:end-1) bdy0; bdy1 bdylin(1:end-1); ...
24b         flipud(bdylin(2:end)) bdy1; bdy0 flipud(bdylin(2:end))];
25     bdyctrs{k} = bdydata;
26     intctrs{k} = intdata;
       % Compute new right-hand side (= residual)
27a    Tf = [Lu(intdata(:,1),intdata(:,2)); ...
27b      sin(pi*bdydata(1:sn-1,1)); zeros(3*(sn-1),1)];
28     rhs = Tf - Rf_old;
       % Compute blocks for collocation matrix
29     DM_IIdata = DistanceMatrixCSRBF(1ntdata,intctrs{k},ep(k));
30     DM_IBdata = DistanceMatrixCSRBF(intdata,bdyctrs{k},ep(k));
31     DM_BIdata = DistanceMatrixCSRBF(bdydata,intctrs{k},ep(k));
32     DM_BBdata = DistanceMatrixCSRBF(bdydata,bdyctrs{k},ep(k));
33     LLCM = L2rbf(ep(k),DM_IIdata);
34     LBCM = Lrbf(ep(k),DM_IBdata);
35     BLCM = Lrbf(ep(k),DM_BIdata);
36     BBCM = rbf(ep(k),DM_BBdata);
37     CM = [LLCM LBCM; BLCM BBCM];
       % Compute coefficients for RBF solution of detail level
38     coef{k} = CM\rhs;
39     if (k < K)
          % based on the distances between the next finer
          % points (respoints) and centers
40        for j=1:k
```

```
41          DM_IIres = DistanceMatrixCSRBF(intres,intctrs{j},ep(j));
42          DM_IBres = DistanceMatrixCSRBF(intres,bdyctrs{j},ep(j));
43          DM_BIres = DistanceMatrixCSRBF(bdyres,intctrs{j},ep(j));
44          DM_BBres = DistanceMatrixCSRBF(bdyres,bdyctrs{j},ep(j));
45          LLRM = L2rbf(ep(j),DM_IIres);
46          LBRM = Lrbf(ep(j),DM_IBres);
47          BLRM = Lrbf(ep(j),DM_BIres);
48          BBRM = rbf(ep(j),DM_BBres);
49          RM{j} = [LLRM LBRM; BLRM BBRM];
50       end
         % Evaluate RBF approximation (sum of all previous fits,
         % but evaluated on current grid)
51       Rf = zeros(N2+4*sqrt(N2)-4,1);
52       for j=1:k
53          Rf = Rf + RM{j}*coef{j};
54       end
55       Rf_old = Rf;
56    end
      % Compute evaluation matrix
57    DM_inteval = DistanceMatrixCSRBF(epoints,intctrs{k},ep(k));
58    DM_bdyeval = DistanceMatrixCSRBF(epoints,bdyctrs{k},ep(k));
59    LEM = Lrbf(ep(k),DM_inteval);
60    BEM = rbf(ep(k),DM_bdyeval);
61    EM = [LEM BEM];
      % Evaluate RBF approximation
62    Pf = EM*coef{k};
63    if (k > 1)
64       Pf = Pf_old + Pf;
65    end
66    Pf_old = Pf;
      % Compute maximum error on evaluation grid
67    maxerr = norm(Pf-exact,inf);
68    rms_err = norm(Pf-exact)/neval;
69    fprintf('RMS error:     %e\n', rms_err)
70    fprintf('Maximum error: %e\n', maxerr)
71    if (k > 1)
72       max_rate = log(maxerr_old/maxerr)/log(2);
73       rms_rate = log(rms_err_old/rms_err)/log(2);
74       fprintf('RMS rate:      %f\n', rms_rate)
75       fprintf('Maxerror rate: %f\n', max_rate)
76    end
77    maxerr_old = maxerr;  rms_err_old = rms_err;
```

```
    % Plot collocation solution
78  fview = [-30,30];
79  PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview);
80  PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview);
81  end
```

For non-symmetric collocation we can delete line 3 of Program 41.2 and need to replace lines 29–37 by

```
DM_intdata = DistanceMatrixCSRBF(intdata,ctrs{k},ep(k));
DM_bdydata = DistanceMatrixCSRBF(bdydata,ctrs{k},ep(k));
LCM = Lrbf(ep(k),DM_intdata);
BCM = rbf(ep(k),DM_bdydata);
CM = [LCM; BCM];
```

lines 41–49 by

```
DM_intres = DistanceMatrixCSRBF(intres,ctrs{j},ep(j));
DM_bdyres = DistanceMatrixCSRBF(bdyres,ctrs{j},ep(j));
LRM = Lrbf(ep(j),DM_intres);
BRM = rbf(ep(j),DM_bdyres);
RM{j} = [LRM; BRM];
```

and lines 57–61 by

```
DM_eval = DistanceMatrixCSRBF(epoints,ctrs{k},ep(k));
EM = rbf(ep(k),DM_eval);
```

Table 41.3 Stationary symmetric and non-symmetric multilevel collocation solutions of Example 39.1 using CSRBFs with initial scale parameter $\varepsilon = 0.5$. Interior Halton points, additional centers on boundary.

N	symmetric		non-symmetric		
(interior points)	RMS-error	rate	RMS-error	rate	% nonzero
3×3	2.491453e-002		1.512255e-002		100
5×5	1.011182e-002	1.3009	3.785466e-003	1.9982	89.53
9×9	9.016652e-003	0.1654	7.873870e-004	2.2653	40.03
17×17	8.995221e-003	0.0034	2.470405e-004	1.6723	13.55
33×33	8.994046e-003	0.0002	1.070175e-004	1.2069	4.00
65×65	8.993892e-003	0.0000	8.334939e-005	0.3606	1.12
129×129	8.993969e-003	−0.0000	7.863637e-005	0.0840	0.29

We note that the non-convergence behavior can be observed for both the symmetric and the non-symmetric approach. However, with the non-symmetric approach the convergence ceases at a significantly later stage. The explanation for the different convergence behavior of the two methods is the same as that presented at

the end of the previous example. The higher derivatives required for the symmetric method are numerically of a more localized nature — even though here the sparsity patterns of the symmetric and non-symmetric matrices are identical (c.*f.* the right-most column in Tables 41.3 and 41.4).

Table 41.4 Stationary symmetric and non-symmetric multilevel collocation solutions of Example 39.1 using CSRBFs with initial scale parameter $\varepsilon = 0.25$. Interior Halton points, additional centers on boundary.

N	symmetric		non-symmetric		
(interior points)	RMS-error	rate	RMS-error	rate	% nonzero
3×3	5.866837e-003		6.077025e-003		100
5×5	7.287305e-004	3.0091	5.751941e-004	3.4012	100
9×9	1.193468e-004	2.6102	1.168520e-004	2.2994	91.90
17×17	3.188905e-005	1.9040	1.482129e-005	2.9789	42.69
33×33	2.530241e-005	0.3338	2.550329e-006	2.5389	14.34
65×65	2.487657e-005	0.0245	6.067264e-007	2.0716	4.17
129×129	2.485184e-005	0.0014	7.348170e-008	3.0456	1.13

Also, the accuracy that can be obtained with the multilevel algorithm — while better than using the stationary approach without residual iteration in Example 41.1 — is considerably poorer than what we were able to obtain with globally supported functions (*c.f.* Examples 39.1, 40.1, or the non-stationary part of Example 41.1).

The same saturation phenomenon was observed by Wendland in the context of a multilevel Galerkin algorithm for compactly supported RBFs (see [Wendland (1999b)] as well as our discussion in Chapter 44).

It has been suggested that the convergence behavior of the multilevel collocation algorithm may be linked to the phenomenon of approximate approximation. However, so far no connection has been established.

As was shown in [Fasshauer (1999d)] a possible remedy for the non-convergence problem is . One might also expect that a slightly different scaling of the support sizes of the basis functions (such that the bandwidth of the matrix is allowed to increase slowly from one iteration to the next, *i.e.*, moving toward the non-stationary setting) will lead to better results. In [Fasshauer (1999d)] it was shown that this is in fact true. However, smoothing further improved the convergence. A discussion of the idea of post-conditioning via smoothing is beyond the scope of this text. We refer the reader to the paper [Fasshauer and Jerome (1999)].

Chapter 42

Using Radial Basis Functions in Pseudospectral Mode

Pseudospectral (PS) methods are known as highly accurate solvers for partial differential equations. The basic idea (see, *e.g.*, [Fornberg (1998); Trefethen (2000)]) is to use a set of very smooth and global basis functions B_j, $j = 1, \ldots, N$, such as polynomials to represent the approximate solution of the PDE via

$$\hat{u}(x) = \sum_{j=1}^{N} c_j B_j(x), \qquad x \in \mathbb{R}. \tag{42.1}$$

Since most of our discussion will focus on a representation of the spatial part of the solution we will at first ignore the time variable that may be a part of the the formulas for $\hat{u}$. We will later employ standard time-stepping procedures to deal with the temporal part of the solution. Moreover, since standard pseudospectral methods are designed for the univariate case we initially limit ourselves to single-variable functions. Later we will generalize the discussion to multivariate (spatial) problems by using radial basis functions.

An important feature of pseudospectral methods is the fact that one usually is content with obtaining an approximation to the solution on a discrete set of grid points x_i, $i = 1, \ldots, N$. One of several ways to implement the pseudospectral method is via so-called *differentiation matrices*, *i.e.*, one finds a matrix D such that at the collocation points x_i we have

$$\boldsymbol{u}' = D\boldsymbol{u}, \tag{42.2}$$

where $\boldsymbol{u} = [\hat{u}(x_1), \ldots, \hat{u}(x_N)]^T$ is the vector of values of the approximate solution $\hat{u}$ at the collocation points. Frequently, orthogonal polynomials such as Chebyshev polynomials are used as basis functions, and the collocation points are corresponding Chebyshev points. In this case the entries of the differentiation matrix are explicitly known (see, *e.g.*, [Trefethen (2000)]).

Example 42.1. In order to get an idea of how the differentiation matrix is used to solve a partial differential equation we consider the following simple one-dimensional

transport equation (*c.f.* the numerical experiments in Section 43.1.1 below):

$$\begin{aligned} u_t(x,t) + cu_x(x,t) &= 0, \quad x > -1,\ t > 0, \\ u(-1,t) &= 0, \\ u(x,0) &= f(x). \end{aligned} \tag{42.3}$$

In order to solve this problem we discretize the spatial domain with the collocation points x_i, $i = 1, \ldots, N$, so that for any fixed time t_n we have the vector $\boldsymbol{u}^{(n)} = [\hat{u}(x_1, t_n), \ldots, \hat{u}(x_N, t_n)]^T$ of values of the approximate solution. In order to march in time we use a standard forward difference approximation of the time derivative, *i.e.*,

$$u_t(x, t_n) \approx \frac{u(x, t_{n+1}) - u(x, t_n)}{\Delta t}, \tag{42.4}$$

where $\Delta t = t_{n+1} - t_n$. In our vectorized notation at the collocation points application of the differentiation matrix to express the spatial derivative along with (42.4) for the time derivative leads to

$$\boldsymbol{u}^{(n+1)} = \boldsymbol{u}^{(n)} - c\Delta t D\boldsymbol{u}^{(n)}$$

for the solution of (42.3). Thus, there is no need — as with the RBF collocation methods studied earlier — to compute the expansion coefficients c_j in the representation (42.1) of the approximate solution. Also, no linear systems are solved during the time-marching phase of the code. The determination of the differentiation matrix will, however, involve solution of a linear system in the RBF framework.

We are interested in using infinitely smooth radial basis functions in the pseudospectral expansion (42.1), *i.e.*, $B_j(x) = \varphi(\|x - x_j\|)$, where φ is one of our strictly positive definite basic functions such as a Gaussian or an inverse multiquadric. We will also experiment with the use of functions having only limited smoothness such as the globally supported Matérn functions or Wendland functions with a large support. With some additional notational effort all that follows can also be formulated for conditionally positive definite functions such as multiquadrics.

42.1 Differentiation Matrices

In order to understand how to find a differentiation matrix consider the expansion (42.1) and let B_j, $j = 1, \ldots, N$, be an arbitrary linearly independent set of smooth functions that will serve as the basis for our approximation space.

If we evaluate (42.1) at the collocation points x_i, $i = 1, \ldots, N$, then we get

$$\hat{u}(x_i) = \sum_{j=1}^{N} c_j B_j(x_i), \qquad i = 1, \ldots, N,$$

or in matrix-vector notation

$$\boldsymbol{u} = A\mathbf{c}, \tag{42.5}$$

where $\boldsymbol{c} = [c_1, \ldots, c_N]^T$ is the coefficient vector, the *evaluation matrix* A has entries $A_{ij} = B_j(x_i)$, and $\boldsymbol{u}$ is as before.

By linearity we can also use the expansion (42.1) to compute the derivative of $\hat{u}$ by differentiating the basis functions

$$\frac{d}{dx}\hat{u}(x) = \sum_{j=1}^{N} c_j \frac{d}{dx} B_j(x).$$

If we again evaluate at the collocation points x_i, then we get in matrix-vector notation

$$\boldsymbol{u}' = A_x \boldsymbol{c}, \tag{42.6}$$

where $\boldsymbol{u}$ and $\boldsymbol{c}$ are as in (42.5) above, and the *derivative matrix* A_x has entries $\frac{d}{dx}B_j(x_i)$, or, in the case of radial basis functions, $\frac{d}{dx}\,\varphi(\|x - x_j\|)|_{x=x_i}$.

In order to obtain the differentiation matrix D we need to ensure invertibility of the evaluation matrix A. This depends both on the basis functions chosen as well as the location of the collocation points x_i. For univariate polynomials it is well-known that the evaluation matrix is invertible for any set of distinct collocation points. In particular, if the polynomials are written in cardinal (or Lagrange) form, then the evaluation matrix is the identity matrix. If we use strictly positive definite radial basis functions, then the matrix A is invertible for any set of distinct collocation points (also non-uniformly spaced points and in $\mathbb{R}^s$, $s > 1$) according to our discussion in Chapter 3. Cardinal radial basis functions, on the other hand, are rather difficult to obtain. For the special case of uniform one-dimensional grids explicit formulas can be found in [Platte and Driscoll (2005)]. A general discussion of the cardinal representation of RBFs is given in Chapter 14. In the following we will not insist on having a cardinal representation.

Now that we have discussed the invertibility of A, we can use (42.5) to formally solve for the coefficient vector $\boldsymbol{c} = A^{-1}\boldsymbol{u}$, and with this (42.6) yields

$$\boldsymbol{u}' = A_x A^{-1} \boldsymbol{u},$$

so that the *differentiation matrix* D corresponding to (42.2) is given by

$$D = A_x A^{-1}.$$

For more complex linear differential operators $\mathcal{L}$ with constant coefficients we proceed in an analogous fashion to obtain a discretized differential operator (differentiation matrix)

$$L = A_{\mathcal{L}} A^{-1}, \tag{42.7}$$

where the matrix $A_{\mathcal{L}}$ has entries $(A_{\mathcal{L}})_{ij} = \mathcal{L}B_j(x_i)$. In the case of radial basis functions these entries are of the form $(A_{\mathcal{L}})_{ij} = \mathcal{L}\varphi(\|x - x_j\|)|_{x=x_i}$.

In the context of pseudospectral methods the differentiation matrices D or L can now be used to solve all kinds of PDEs (time-dependent as well as time-independent). Sometimes only multiplication by L is required, *e.g.*, for many time-stepping algorithms such as the example given at the beginning of the chapter. For

other problems one needs to be able to invert L. In the standard PS case it is known that the Chebyshev differentiation matrix has an N-fold zero eigenvalue (see [Canuto *et al.* (1988)], p. 70), and thus is not invertible by itself. However, once boundary conditions are taken into consideration the situation changes (see, *e.g.*, [Trefethen (2000)], p. 67).

Example 42.2. To obtain a little more insight into the special properties of radial basis functions let us pretend to solve the *ill-posed* linear elliptic PDE of the form $\mathcal{L}u = f$ by ignoring boundary conditions. An approximate solution at the collocation points x_i might be obtained by solving the discrete linear system

$$L\boldsymbol{u} = \boldsymbol{f},$$

where $\boldsymbol{f}$ contains the values of f at the collocation points and L is as above. In other words, the solution at the collocation points is given (see (42.7)) by

$$\boldsymbol{u} = L^{-1}\boldsymbol{f} = A(A_{\mathcal{L}})^{-1}\boldsymbol{f},$$

and we see that invertibility of L (and therefore $A_{\mathcal{L}}$) is required to proceed.

As mentioned above, the differentiation matrix for pseudospectral methods based on Chebyshev polynomials is singular. This is only natural since the problem of reconstructing an unknown function from the values of its derivatives alone is ill-posed.

However, if we use radial basis functions the results on generalized Hermite interpolation cited in Chapter 36 ensure that the matrix $A_{\mathcal{L}}$ is invertible provided a strictly positive definite basic function is used and the differential operator is elliptic. Therefore, the basic differentiation matrix L for RBF-based pseudospectral methods *is* invertible.

The observation just made suggests that RBF methods are sometimes "too good to be true". They may deliver a "solution" even for ill-posed problems. This is a consequence of the optimality principles of Chapter 18, *i.e.*, as the minimizer of the native space norm RBF methods possess a built-in *regularization capability.* This interesting feature of RBFs has recently been used to solve ill-posed problems (see, e.g., [Cheng and Cabral (2005)]).

42.2 PDEs with Boundary Conditions via Pseudospectral Methods

First we discuss how the linear elliptic PDE problem

$$\mathcal{L}u = f \quad \text{in } \Omega$$

with Dirichlet boundary condition

$$u = g \quad \text{on } \Gamma = \partial\Omega$$

can be solved using pseudospectral methods. Sometimes one can find basis functions that already satisfy the boundary conditions (especially for periodic problems).

However, if the basis functions do not satisfy the boundary conditions we can follow a very simple procedure (see, e.g., Program 36 of [Trefethen (2000)]). Just take the differentiation matrix L based on all collocation points x_i, and then replace those rows of L corresponding to collocation at boundary points with unit vectors that have a one in the position corresponding to the diagonal of L. Thus, the condition $u = g$ will be explicitly enforced at this point as soon as we set the right-hand side to the corresponding value of g.

By reordering the rows and columns of the resulting matrix we obtain a block matrix of the form

$$L_\Gamma = \begin{bmatrix} M & P \\ 0 & I \end{bmatrix}, \tag{42.8}$$

where the non-zero square blocks M and I are of size $(N - N_\mathcal{B}) \times (N - N_\mathcal{B})$ and $N_\mathcal{B} \times N_\mathcal{B}$, respectively. Here $N_\mathcal{B}$ denotes the number of grid points on the boundary Γ.

On the grid of collocation points the solution of the PDE with boundary conditions is then obtained by solving the block linear system

$$L_\Gamma \boldsymbol{u} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{bmatrix}, \tag{42.9}$$

where the vectors $\boldsymbol{f}$ and $\boldsymbol{g}$ collect the values of f and g at the respective collocation points, and the vector $\boldsymbol{u}$ of grid values of the approximate solution has been reordered along with the columns of the matrix so that it can be decomposed into $\boldsymbol{u} = [\boldsymbol{u}_\mathcal{I}, \boldsymbol{u}_\mathcal{B}]^T$. Here $\boldsymbol{u}_\mathcal{I}$ collects the values in the interior of the domain Ω and $\boldsymbol{u}_\mathcal{B}$ collects the values on the boundary.

Solving (42.9) for $\boldsymbol{u}_\mathcal{B} = \boldsymbol{g}$ and substituting this back into the top part we obtain

$$\boldsymbol{u}_\mathcal{I} = M^{-1}(\boldsymbol{f} - P\boldsymbol{g}),$$

or, in the case of homogeneous boundary conditions,

$$\boldsymbol{u}_\mathcal{I} = M^{-1}\boldsymbol{f}.$$

We now see that all that really matters is whether the matrix M is invertible. In the case of Chebyshev polynomial basis functions and the second-derivative operator $\frac{d^2}{dx^2}$ coupled with different types of boundary conditions this question has been answered affirmatively by Gottlieb and Lustman (see [Gottlieb and Lustman (1983)], or, *e.g.*, Section 11.4 of [Canuto *et al.* (1988)]). Program 15 of [Trefethen (2000)] also provides a discussion and an illustration of one such problem. We will look at the matrix M in the RBF context in the next section.

42.3 A Non-Symmetric RBF-based Pseudospectral Method

Once boundary conditions are added to the PDE $\mathcal{L}u = f$, then either of the two collocation approaches discussed in Chapters 38–41 are commonly used in the RBF

community. Recall that in Kansa's non-symmetric method [Kansa (1990b)] one starts with the expansion

$$\hat{u}(\boldsymbol{x}) = \sum_{j=1}^{N} c_j \Phi_j(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega \subseteq \mathbb{R}^s, \tag{42.10}$$

just as before (*c.f.* (42.1)). However, the coefficient vector $\boldsymbol{c}$ is now actually computed by inserting (42.10) into the PDE and boundary conditions by forcing these equations to be satisfied at the collocation points $\boldsymbol{x}_i$. The RBF *collocation solution* is therefore obtained by solving the linear system

$$\begin{bmatrix} \tilde{A}_{\mathcal{L}} \\ \tilde{A} \end{bmatrix} \boldsymbol{c} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{bmatrix}, \tag{42.11}$$

where $\boldsymbol{f}$ and $\boldsymbol{g}$ are as above, and the (rectangular) matrices $\tilde{A}_{\mathcal{L}}$ and $\tilde{A}$ are of the form

$$\begin{aligned} (\tilde{A}_{\mathcal{L}})_{ij} &= \mathcal{L}\Phi_j(\boldsymbol{x}_i) = \mathcal{L}\varphi(\|\boldsymbol{x} - \boldsymbol{x}_j\|)|_{\boldsymbol{x}=\boldsymbol{x}_i}, \quad i = 1, \ldots, N - N_{\mathcal{B}},\ j = 1, \ldots, N, \\ \tilde{A}_{ij} &= \Phi_j(\boldsymbol{x}_i) = \varphi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|), \quad i = N - N_{\mathcal{B}} + 1, \ldots, N,\ j = 1, \ldots, N. \end{aligned}$$

Assuming that the system matrix in (42.11) is invertible one then obtains the approximate solution (at any point $\boldsymbol{x}$) by using the coefficients $\boldsymbol{c}$ in (42.10). However, as was mentioned earlier, counterexamples in [Hon and Schaback (2001)] show that certain collocation grids do not allow invertibility of the system matrix in (42.11).

If we are interested in the RBF collocation solution at the collocation points only, then (using $\boldsymbol{c}$ from (42.11) and once again assuming invertibility of the system matrix) we get

$$\boldsymbol{u} = A\boldsymbol{c} = A \begin{bmatrix} \tilde{A}_{\mathcal{L}} \\ \tilde{A} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{bmatrix}$$

with evaluation matrix A such that $A_{ij} = \Phi_j(\boldsymbol{x}_i)$ as above. This suggests that (according to our discussion in Section 42.1) the discretized differential operator L based on the grid points $\boldsymbol{x}_i$, $i = 1, \ldots, N$, and basis functions Φ_j, $j = 1, \ldots, N$, is given by

$$L_\Gamma = \begin{bmatrix} \tilde{A}_{\mathcal{L}} \\ \tilde{A} \end{bmatrix} A^{-1}.$$

Indeed, we have

$$\begin{bmatrix} \tilde{A}_{\mathcal{L}} \\ \tilde{A} \end{bmatrix} A^{-1} = \begin{bmatrix} M & P \\ 0 & I \end{bmatrix}$$

with the same blocks M, P, 0 and I as above. To see this we introduce the following notation:

$$A = \begin{bmatrix} \boldsymbol{a}_1^T \\ \vdots \\ \boldsymbol{a}_N^T \end{bmatrix} \quad \text{and} \quad A^{-1} = \begin{bmatrix} \boldsymbol{b}_1 & \ldots & \boldsymbol{b}_N \end{bmatrix}$$

with column vectors $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ such that $\boldsymbol{a}_i^T \boldsymbol{b}_j = \delta_{ij}$. For Kansa's matrix from (42.11) this notation implies

$$\begin{bmatrix} \tilde{A}_{\mathcal{L}} \\ \tilde{A} \end{bmatrix} = \begin{bmatrix} \boldsymbol{a}_{\mathcal{L},1}^T \\ \vdots \\ \boldsymbol{a}_{\mathcal{L},N-N_{\mathcal{B}}}^T \\ \boldsymbol{a}_{N-N_{\mathcal{B}}+1}^T \\ \vdots \\ \boldsymbol{a}_N^T \end{bmatrix},$$

where we have used an analogous notation to denote the rows of the block $\tilde{A}_{\mathcal{L}}$. Now the discretized differential operator based on the non-symmetric collocation approach is given by

$$\begin{aligned} \begin{bmatrix} \tilde{A}_{\mathcal{L}} \\ \tilde{A} \end{bmatrix} A^{-1} &= \begin{bmatrix} \boldsymbol{a}_{\mathcal{L},1}^T \\ \vdots \\ \boldsymbol{a}_{\mathcal{L},N-N_{\mathcal{B}}}^T \\ \boldsymbol{a}_{N-N_{\mathcal{B}}+1}^T \\ \vdots \\ \boldsymbol{a}_N^T \end{bmatrix} \begin{bmatrix} \boldsymbol{b}_1 \dots \boldsymbol{b}_N \end{bmatrix} \\ &= \begin{bmatrix} \tilde{A}_{\mathcal{L}} A_{\mathcal{I}}^{-1} & \tilde{A}_{\mathcal{L}} A_{\mathcal{B}}^{-1} \\ \underbrace{\tilde{A} A_{\mathcal{I}}^{-1}}_{=0} & \underbrace{\tilde{A} A_{\mathcal{B}}^{-1}}_{=I} \end{bmatrix}. \end{aligned}$$

Here we partitioned A^{-1} into the blocks $A_{\mathcal{I}}^{-1}$ with $N - N_{\mathcal{B}}$ columns corresponding to interior points, and $A_{\mathcal{B}}^{-1}$ with $N_{\mathcal{B}}$ columns corresponding to the remaining boundary points. Also, we made use of the fact that $\boldsymbol{a}_i^T \boldsymbol{b}_j = \delta_{ij}$.

This is the same as (see (42.8))

$$\begin{bmatrix} M & P \\ 0 & I \end{bmatrix} = L_{\Gamma},$$

where M and P were obtained from the discrete differential operator (42.7)

$$L = A_{\mathcal{L}} A^{-1} = \begin{bmatrix} A_{\mathcal{L}} A_{\mathcal{I}}^{-1} & A_{\mathcal{L}} A_{\mathcal{B}}^{-1} \end{bmatrix}$$

by replacing certain rows with unit vectors as we explained is common practice for handling the Dirichlet boundary conditions in the PS approach.

Thus, we have just seen that — provided we use the same basis functions Φ_j and the same grid of collocation points $\boldsymbol{x}_i$ — the non-symmetric RBF collocation approach for the solution of an elliptic PDE with Dirichlet boundary conditions followed by evaluation at the grid points is identical to a pseudospectral approach. However, *neither of the two methods is well-defined in general* since they both rely on the invertibility of Kansa's collocation matrix.

On the other hand, we showed above that we can always *form* the discretized differential operator

$$L_\Gamma = \begin{bmatrix} \tilde{A}_{\mathcal{L}} \\ \tilde{A} \end{bmatrix} A^{-1} = \begin{bmatrix} M & P \\ 0 & I \end{bmatrix}$$

— even if Kansa's matrix is not invertible. This implies that we can safely use the non-symmetric RBF pseudospectral approach whenever inversion of the discretized differential operator is not required (*e.g.*, in the context of explicit time-stepping for parabolic PDEs).

Another interesting feature that we will illustrate in the next chapter is the fact shown recently by a number of authors (see, *e.g.*, [de Boor (2006); Driscoll and Fornberg (2002); Schaback (2005); Schaback (2006b)]) that in the limiting case of "flat" basis functions the one-dimensional RBF interpolant yields a polynomial interpolant. Since we also mentioned earlier that the discretized differential operator L_Γ is invertible if a univariate polynomial basis is used we can conclude that Kansa's collocation matrix *is* invertible in the limiting case $\varepsilon \to 0$.

42.4 A Symmetric RBF-based Pseudospectral Method

The second RBF collocation method is the symmetric approach whose system matrix is invertible for all grid configurations and any strictly positive definite basic function as explained in Chapter 38.

Recall that for the symmetric collocation method one uses a different basis than for the non-symmetric approach (42.10), *i.e.*, a different function space to represent the approximate solution. For the same elliptic PDE and Dirichlet boundary conditions as above one now starts with

$$\hat{u}(\boldsymbol{x}) = \sum_{j=1}^{N-N_{\mathcal{B}}} c_j \mathcal{L}_j^{\boldsymbol{\xi}} \Phi(\boldsymbol{x}) + \sum_{j=N-N_{\mathcal{B}}+1}^{N} c_j \Phi_j(\boldsymbol{x}). \tag{42.12}$$

Since the Φ_j are assumed to be radial functions, *i.e.*, $\Phi_j(\boldsymbol{x}) = \varphi(\|\boldsymbol{x} - \boldsymbol{x}_j\|)$ the functionals $\mathcal{L}_j^{\boldsymbol{\xi}}$ can be interpreted as an application of $\mathcal{L}$ to φ viewed as a function of the second variable followed by evaluation at $\boldsymbol{x}_j$ (see the discussion in Chapters 36 and 38). One obtains the coefficients $\boldsymbol{c} = [\boldsymbol{c}_{\mathcal{I}}, \boldsymbol{c}_{\mathcal{B}}]^T$ by solving the linear system

$$\begin{bmatrix} \hat{A}_{\mathcal{L}\mathcal{L}^{\boldsymbol{\xi}}} & \hat{A}_{\mathcal{L}} \\ \hat{A}_{\mathcal{L}^{\boldsymbol{\xi}}} & \hat{A} \end{bmatrix} \begin{bmatrix} \boldsymbol{c}_{\mathcal{I}} \\ \boldsymbol{c}_{\mathcal{B}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{bmatrix}. \tag{42.13}$$

Here the blocks $\hat{A}_{\mathcal{L}\mathcal{L}^{\boldsymbol{\xi}}}$ and $\hat{A}$, respectively, are square matrices corresponding to the interaction of interior collocation points with each other and boundary collocation points with each other. As discussed in Chapter 38 (for centers coinciding with collocation points) their entries are given by

$$(\hat{A}_{\mathcal{L}\mathcal{L}^{\boldsymbol{\xi}}})_{ij} = \left[\mathcal{L}\left[\mathcal{L}^{\boldsymbol{\xi}} \varphi(\|\boldsymbol{x} - \boldsymbol{\xi}\|)\right]_{\boldsymbol{\xi}=\boldsymbol{x}_j}\right]_{\boldsymbol{x}=\boldsymbol{x}_i}, \qquad i,j = 1,\ldots,N - N_{\mathcal{B}},$$

$$\hat{A}_{ij} = \varphi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|), \qquad i,j = N - N_{\mathcal{B}} + 1, \ldots, N.$$

The other two blocks are rectangular, and correspond to interaction of interior points with boundary points and vice versa. They are defined as

$$(\hat{A}_{\mathcal{L}})_{ij} = [\mathcal{L}\varphi(\|\boldsymbol{x} - \boldsymbol{x}_j\|)]_{\boldsymbol{x}=\boldsymbol{x}_i}, \; i = 1, \ldots, N - N_{\mathcal{B}}, \; j = N - N_{\mathcal{B}} + 1, \ldots, N,$$
$$(\hat{A}_{\mathcal{L}\xi})_{ij} = \left[\mathcal{L}^{\boldsymbol{\xi}}\varphi(\|\boldsymbol{x}_i - \boldsymbol{x}\|)\right]_{\boldsymbol{x}=\boldsymbol{x}_j}, \; i = N - N_{\mathcal{B}} + 1, \ldots, N, \; j = 1, \ldots, N - N_{\mathcal{B}}.$$

As already mentioned, it is well known that the system matrix in (42.13) is invertible for strictly positive definite radial functions. This implies that we can obtain the approximate solution at any point $\boldsymbol{x}$ by using the computed coefficients $\boldsymbol{c}$ in (42.12). Thus this RBF collocation method is rather similar to Kansa's non-symmetric method with the notable difference that the collocation approach is well-defined.

A nice connection between the symmetric and non-symmetric collocation methods appears if we consider the corresponding symmetric pseudospectral approaches.

To this end we use the expansion (42.12) on which the symmetric RBF collocation method is based as starting point for a pseudospectral method, *i.e.*,

$$\hat{u}(\boldsymbol{x}) = \sum_{j=1}^{N-N_{\mathcal{B}}} c_j \mathcal{L}_j^{\boldsymbol{\xi}}\Phi(\boldsymbol{x}) + \sum_{j=N-N_{\mathcal{B}}+1}^{N} c_j \Phi_j(\boldsymbol{x}). \tag{42.14}$$

In vectorized notation this corresponds to

$$\hat{u}(\boldsymbol{x}) = \left[\, \boldsymbol{a}_{\mathcal{L}\xi}^T(\boldsymbol{x}) \; \tilde{\boldsymbol{a}}^T(\boldsymbol{x}) \,\right] \begin{bmatrix} \boldsymbol{c}_{\mathcal{I}} \\ \boldsymbol{c}_{\mathcal{B}} \end{bmatrix}$$

with appropriate row vectors $\boldsymbol{a}_{\mathcal{L}\xi}^T(\boldsymbol{x})$ and $\tilde{\boldsymbol{a}}^T(\boldsymbol{x})$. Evaluated on the grid of collocation points this becomes

$$\boldsymbol{u} = \left[\, A_{\mathcal{L}\xi} \; \tilde{A}^T \,\right] \begin{bmatrix} \boldsymbol{c}_{\mathcal{I}} \\ \boldsymbol{c}_{\mathcal{B}} \end{bmatrix}.$$

Here the blocks $A_{\mathcal{L}\xi}$ and $\tilde{A}^T$ of the evaluation matrix are rectangular matrices with entries

$$(A_{\mathcal{L}\xi})_{ij} = \left[\mathcal{L}^{\boldsymbol{\xi}}\varphi(\|\boldsymbol{x}_i - \boldsymbol{x}\|)\right]_{\boldsymbol{x}=\boldsymbol{x}_j}, \qquad i = 1, \ldots, N, \; j = 1, \ldots, N - N_{\mathcal{B}},$$
$$(\tilde{A}^T)_{ij} = \varphi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|), \qquad i = 1, \ldots, N, \; j = N - N_{\mathcal{B}} + 1, \ldots, N,$$

corresponding to evaluation of the basis functions used in (42.12) at the collocation points $\boldsymbol{x}_i$. Note that the second matrix with entries $\varphi(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|)$ is in fact the transpose of the corresponding block of the system matrix in (42.11) for Kansa's method (and thus use of the tilde-notation is justified).

Moreover, the radial symmetry of the basis functions implies that the first block of the evaluation matrix for the symmetric collocation method, $A_{\mathcal{L}\xi}$, is again the transpose of the corresponding block in Kansa's collocation method, $\tilde{A}_{\mathcal{L}}$.

To see this we consider differential operators of even orders and odd orders separately. If $\mathcal{L}$ is a linear differential operator of odd order, then $\mathcal{L}^{\boldsymbol{\xi}}$ will introduce

a sign change since it is acting on φ as a function of the second variable. In addition, odd order derivatives (evaluated at $\boldsymbol{x} = \boldsymbol{x}_j$) include a factor of the form $\boldsymbol{x}_i - \boldsymbol{x}_j$. Now, transposition of this factor will again lead to a sign change. The combination of these two effects ensures that $A_{\mathcal{L}\xi} = \tilde{A}_{\mathcal{L}}^T$. For even orders the effects of the operators $\mathcal{L}$ and $\mathcal{L}^{\xi}$ are indistinguishable and the linear factor is not present.

Therefore, using symmetric RBF collocation we obtain the approximate solution of the boundary value problem on the collocation grid as

$$\boldsymbol{u} = \begin{bmatrix} \tilde{A}_{\mathcal{L}} \\ \tilde{A} \end{bmatrix}^T \begin{bmatrix} \hat{A}_{\mathcal{L}\mathcal{L}^{\xi}} & \hat{A}_{\mathcal{L}} \\ \hat{A}_{\mathcal{L}^{\xi}} & \hat{A} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{bmatrix}.$$

We emphasize that this is *not* the solution of a pseudospectral method built on the same function space (same basis functions and same collocation points) as the symmetric RBF collocation method.

For a pseudospectral method we would require the discretized differential operator. Formally (assuming invertibility of Kansa's matrix) we would have

$$\hat{L}_{\Gamma} = \begin{bmatrix} \hat{A}_{\mathcal{L}\mathcal{L}^{\xi}} & \hat{A}_{\mathcal{L}} \\ \hat{A}_{\mathcal{L}^{\xi}} & \hat{A} \end{bmatrix} \begin{bmatrix} \tilde{A}_{\mathcal{L}} \\ \tilde{A} \end{bmatrix}^{-T}, \tag{42.15}$$

where we already incorporated the boundary conditions in a way analogous to our earlier discussion.

The problem with the differentiation matrix (42.15) for the symmetric pseudospectral approach is that we cannot be assured that the method itself, *i.e.*, the discretized differential operator, is well-defined. In fact, due to the Hon-Schaback counterexample [Hon and Schaback (2001)] we know that there exist grid configurations for which the "basis" used for the symmetric PS expansion is not linearly independent.

Therefore, the symmetric RBF collocation approach is well-suited for problems that require inversion of the differential operator (such as elliptic PDEs). Subsequent evaluation on a grid makes the symmetric collocation *look like* a pseudospectral method — but it may not be one (since we may not be able to formulate the pseudospectral *Ansatz*).

42.5 A Unified Discussion

In both the symmetric and non-symmetric collocation approaches we can think of the approximate solution as a linear combination of appropriate basis functions. In vectorized notation this can be written as

$$\hat{u}(\boldsymbol{x}) = \boldsymbol{p}^T(\boldsymbol{x})\mathrm{c}, \tag{42.16}$$

where the vector $\boldsymbol{p}(\boldsymbol{x})$ contains the values of the basis functions at $\boldsymbol{x}$. If we consider the non-symmetric method these basis functions are just Φ_j, $j = 1, \ldots, N$, while

for the symmetric method they are comprised of both functions of the type Φ_j and $\mathcal{L}_j^{\xi}\Phi$ (*c.f.* (42.14)).

We now let $\mathcal{D}$ denote the linear operator that combines both the differential operator $\mathcal{L}$ and the boundary operator (for Dirichlet boundary conditions the latter is just the identity). Then we have

$$\mathcal{D}\hat{u}(\boldsymbol{x}) = \mathcal{D}\boldsymbol{p}^T(\boldsymbol{x})\boldsymbol{c} = \boldsymbol{q}^T(\boldsymbol{x})\boldsymbol{c} \tag{42.17}$$

for an appropriately defined vector $\boldsymbol{q}(\boldsymbol{x})$. With this notation the boundary value problem for our approximate solution is given by

$$\mathcal{D}\hat{u}(\boldsymbol{x}) = f(\boldsymbol{x}),$$

where f is a piecewise defined function that collects the forcing functions in both the interior and on the boundary.

Now we evaluate the two representations (42.16) and (42.17) on the grid of collocation points $\boldsymbol{x}_i$, $i = 1, \ldots, N$, and obtain

$$\boldsymbol{u} = P\boldsymbol{c} \quad \text{and} \quad \boldsymbol{u}_{\mathcal{D}} = Q\boldsymbol{c}$$

with matrices P and Q whose rows correspond to evaluation of the vectors $\boldsymbol{p}^T(\boldsymbol{x})$ and $\boldsymbol{q}^T(\boldsymbol{x})$, respectively, at the collocation points $\boldsymbol{x}_i$, $i = 1, \ldots, N$. The discretized boundary value problem is then

$$\boldsymbol{u}_{\mathcal{D}} = \boldsymbol{f} \quad \Longleftrightarrow \quad Q\boldsymbol{c} = \boldsymbol{f}, \tag{42.18}$$

where $\boldsymbol{f}$ is the vector of values of f on the grid.

For the non-symmetric collocation approach the evaluation matrix P is the standard RBF interpolation matrix, and the derivative matrix Q is Kansa's matrix, whereas for symmetric collocation P is given by the transpose of Kansa's matrix, and Q is the symmetric collocation matrix.

It is our goal to find the vector $\boldsymbol{u}$, *i.e.*, the values of the approximate solution on the grid of collocation points. There are two ways by which we can potentially obtain this answer:

(1) We solve $Q\boldsymbol{c} = \boldsymbol{f}$ for $\boldsymbol{c}$, *i.e.*,

$$\boldsymbol{c} = Q^{-1}\boldsymbol{f}.$$

Then we use the discretized version of (42.16) to get the desired vector $\boldsymbol{u}$ as

$$\boldsymbol{u} = PQ^{-1}\boldsymbol{f}.$$

(2) Alternatively, we first formally transform the coefficients, *i.e.*, we rewrite $\boldsymbol{u} = P\boldsymbol{c}$ as

$$\boldsymbol{c} = P^{-1}\boldsymbol{u}.$$

Then the discretized boundary value problem (42.18) becomes

$$QP^{-1}\boldsymbol{u} = \boldsymbol{f},$$

and we can obtain the solution vector $\boldsymbol{u}$ by solving this system.

The first approach corresponds to RBF collocation, the second to the pseudospectral approach. Both of these approaches are equivalent as long as all of the matrices involved are invertible. Unfortunately, as mentioned earlier, there are configurations of collocation points for which Kansa's matrix is not invertible. This means that for the non-symmetric case (when Q is Kansa's matrix) Approach 1 cannot be assured to work in general, and Approach 2 can only be used if the discretized differential operator is applied directly (but not inverted). For the symmetric approach (when P is Kansa's matrix), on the other hand, Approach 1 is guaranteed to work in general, but Approach 2 may not be well-defined.

42.6 Summary

Our discussion above revealed that for the non-symmetric (Kansa) *Ansatz* (42.10) we can always formulate the discrete differential operator

$$L_\Gamma = \begin{bmatrix} \tilde{A}_{\mathcal{L}} \\ \tilde{A} \end{bmatrix} A^{-1}.$$

However, we cannot ensure in general the invertibility of L_Γ. This implies that the non-symmetric RBF pseudospectral approach is justified for time-dependent PDEs (with explicit time-stepping methods).

For the symmetric *Ansatz* (42.12), on the other hand, we can in general ensure the solution of $\mathcal{L}u = f$ by RBF collocation. However, it is not possible in general to even formulate the discrete differential operator

$$\hat{L}_\Gamma = \begin{bmatrix} \hat{A}_{\mathcal{L}\mathcal{L}^\epsilon} & \hat{A}_{\mathcal{L}} \\ \hat{A}_{\mathcal{L}^\epsilon} & \hat{A} \end{bmatrix} \begin{bmatrix} A_{\mathcal{L}^\epsilon} & \tilde{A}^T \end{bmatrix}^{-1}$$

needed for the pseudospectral approach. This suggests that we should use the symmetric approach for time-independent PDEs and possibly for time-dependent PDEs with implicit time-stepping.

The difficulties with both approaches can be attributed to the possible singularity of Kansa's matrix which appears as discretized differential operator for the non-symmetric approach, and (via its transpose) as the evaluation matrix in the symmetric approach.

Since the non-symmetric approach is quite a bit easier to implement than the symmetric approach, and since the grid configurations for which the Kansa matrix is singular seem to be very rare (see [Hon and Schaback (2001)]) many researchers (including ourselves) often prefer to use the non-symmetric approach — even under questionable circumstances (such as with implicit time-stepping procedures, or for elliptic problems). The connection to polynomials in the limiting case $\varepsilon = 0$ justifies this point of view at least for 1-D problems.

An interesting question for future research is the study of RBF-pseudospectral methods with moving or adaptive grids. This will be computationally much more

involved than the case discussed here (and illustrated in the next chapter), but the use of RBFs should imply that there is no major restriction imposed by the use of moving (scattered) collocation grids. In particular, with RBF-PS methods one will no longer be restricted to a tensor-product structure as with traditional polynomial pseudospectral methods, *i.e.*, with RBF expansions we should be able to take advantage of scattered multivariate grids as well as spatial domains with non-rectangular geometries.

Chapter 43

RBF-PS Methods in MATLAB

Overall, the coupling of RBF collocation and pseudospectral methods discussed in the previous chapter has provided a number of new insights. For example, it should now be clear that we can apply many standard pseudospectral procedures to RBF solvers. In particular, we now have "standard" procedures for solving time-dependent PDEs with RBFs.

In this chapter we illustrate how the RBF pseudospectral approach can be applied in a way very similar to standard polynomial pseudospectral methods. Among our numerical illustrations are several examples taken from the book [Trefethen (2000)] (see Programs 17, 35 and 36 there). We will also use the 1D transport equation of Example 42.1 to compare the RBF and polynomial pseudospectral methods.

43.1 Computing the RBF-Differentiation Matrix in MATLAB

We begin by explaining how to compute the discretized differential operators (differentiation matrices) that came up in our discussion in the previous chapter.

In order to compute, for example, a first-order differentiation matrix we need to remember that — by the chain rule — the derivative of an RBF will be of the general form

$$\frac{\partial}{\partial x}\varphi(\|\boldsymbol{x}\|) = \frac{x}{r}\frac{d}{dr}\varphi(r).$$

Thus, we require not only the distances, r, but also differences in x, where x is the first component of $\boldsymbol{x}$. Therefore, the main statements in our first MATLAB subroutine (listed as Program 43.1) are the computation of these distance and difference matrices on lines 5 and 6. According to the discussion in the previous chapter, the differentiation matrix is then given by $D = A_x A^{-1}$. This is implemented on lines 9–11. Note the use of the matrix right division operator `/` or `mrdivide` in MATLAB on line 11 used to solve the system $DA = A_x$ for D.

Program 43.1 is actually a little more complicated than it needs to be since we included an optimization of the RBF shape parameter via leave-one-out cross validation as described in Chapter 17 (see lines 4,7 and 8). Here we use a mod-

ification of the basic routine CostEpsilon which we call CostEpsilonDRBF (see Program 43.2 below) so that we optimize the choice of ε for the matrix problem $D = A_x A^{-1} \iff A^T D^T = (A_x)^T$.

Program 43.1. DRBF.m

```
% [D,x] = DRBF(N,rbf,dxrbf)
% Computes the differentiation matrix D for 1-D derivative
% using Chebyshev points and LOOCV for optimal shape parameter
% Input: N, create N+1 collocation points
%        rbf, dxrbf function handles for rbf and its derivative
% Calls on: DistanceMatrix, DifferenceMatrix
% Requires: CostEpsilonDRBF
 1  function [D,x] = DRBF(N,rbf,dxrbf)
 2  if N==0, D=0; x=1; return, end
 3  x = cos(pi*(0:N)/N)';    % Chebyshev points
 4  mine = .1; maxe = 10;    % Shape parameter interval
 5  r = DistanceMatrix(x,x);
 6  dx = DifferenceMatrix(x,x);
 7  ep = fminbnd(@(ep) CostEpsilonDRBF(ep,r,dx,rbf,dxrbf),mine,maxe);
 8  fprintf('Using epsilon = %f\n', ep)
 9  A = rbf(ep,r);
10  Ax = dxrbf(ep,r,dx);
11  D = Ax/A;
```

Note that CostEpsilonDRBF.m is very similar to CostEpsilon.m (*c.f.* Program 17.3). Now, however, we compute a right-hand side *matrix* corresponding to the transpose of A_x. Therefore, the denominator — which remains the same for all right-hand sides (see formula (17.1)) — needs to be cloned on line 6 via the repmat command. The cost of ε is now the Frobenius norm of the matrix EF. Other measures for the error may also be appropriate. For the standard interpolation setting Rippa compared use of the ℓ_1 and ℓ_2 norms (see [Rippa (1999)]) and concluded that the ℓ_1 norm yields more accurate "optima". For the RBF-PS problems to be presented here we have observed very good results with the ℓ_2 (or Frobenius) norm, and therefore that is what is used on line 7 of Program 43.2.

Program 43.2. CostEpsilonDRBF.m

```
% ceps = CostEpsilonDRBF(ep,r,dx,rbf,dxrbf)
% Provides the "cost of epsilon" function for LOOCV optimization
% of shape parameter
% Input: ep, values of shape parameter
%        r, dx, Distance and Difference matrices
%        rbf, dxrbf, definition of rbf and its derivative
 1  function ceps = CostEpsilonDRBF(ep,r,dx,rbf,dxrbf)
```

```
2  N = size(r,2);
3  A = rbf(ep,r);   % = A^T since A is symmetric
4  rhs = dxrbf(ep,r,dx)';   % A_x^T
5  invA = pinv(A);
6  EF = (invA*rhs)./repmat(diag(invA),1,N);
7  ceps = norm(EF(:));
```

43.1.1 *Solution of a 1-D Transport Equation*

We illustrate the use of the subroutine `DBRF.m` by solving a one-dimensional transport equation. Consider

$$\begin{aligned} u_t(x,t) + cu_x(x,t) = 0, \quad x > -1,\ t > 0, \\ u(-1,t) = 0, \\ u(x,0) = f(x), \end{aligned}$$

with the well-known solution

$$u(x,t) = f(x - ct).$$

In Program 43.3 we implement a solution of this problem with the help of the differentiation matrix from Program 43.1 above. Note that we could use almost the identical code to solve this problem with a Chebyshev pseudospectral method as discussed in [Trefethen (2000)]. In fact, in Figure 43.1 we display side-by-side the solutions obtained with Gaussian RBFs and with Chebyshev polynomials. Both solutions were computed on a grid of 21 Chebyshev points. The cross-validation algorithm returned a value of $\varepsilon = 1.874049$ for the Gaussian. The maximum error for the Gaussian solution at time $t = 1$ was 0.0416, while for the PS solution we get 0.0418. The only difference in the PS-code is the replacement of line 4 in Program 43.3 by

```
4  [D,x] = cheb(N)
```

where `cheb.m` is the subroutine provided on page 54 of [Trefethen (2000)] for spectral differentiation.

Program 43.3. `TransportDRBF.m`

```
% TransportDRBF
% Script that solves constant coefficient wave equation
% u_t + c*u_x = 0, using RBF-PS approach
% Calls on: DRBF
1  rbf = @(e,r) exp(-(e*r).^2);      % Gaussian RBF
2  dxrbf = @(e,r,dx) -2*dx*e^2.*exp(-(e*r).^2);
3  N = 20;
4  [D,x] = DRBF(N,rbf,dxrbf);
```

```
x = flipud(x); dt = 0.001; t = 0; c = -1;
v = 64*(-x).^3.*(1+x).^3;
v(find(x>0)) = zeros(length(find(x>0)),1);
% Time-stepping by explicit Euler formula:
tmax = 1; tplot = .02; plotgap = round(tplot/dt);
dt = tplot/plotgap; nplots = round(tmax/tplot);
data = [v'; zeros(nplots,N+1)]; tdata = t;
I = eye(size(D));
for i = 1:nplots
   for n = 1:plotgap
      t = t+dt;
      vv = v(end-1);
      v = v - dt*c*(D*v);      % explicit Euler
      v(1) = 0; v(end) = vv;
   end
   data(i+1,:) = v'; tdata = [tdata; t];
end
surf(x,tdata,data), view(10,70), colormap('default');
axis([-1 1 0 tmax 0 1]), ylabel t, zlabel u, grid off
% exact solution and error
xx = linspace(-1,1,101);
vone = 64*(1-xx).^3.*xx.^3;
vone(find(xx<0)) = zeros(length(find(xx<0)),1);
w = interp1(x,v,xx);
maxErr = norm(w-vone,inf)
```

The graph in Figure 43.1 shows the time profile of the solutions for the time interval $[0, 1]$ with initial profile $f(x) = 64(1-x)^3x^3$ and unit wave speed.

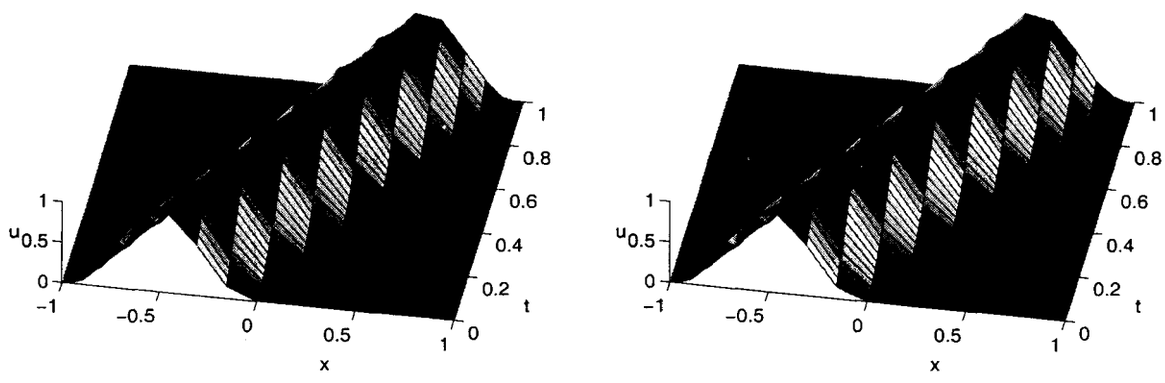


Fig. 43.1 Solution to transport equation based on Gaussian RBFs with $\varepsilon = 1.874049$ (left) and Chebyshev PS method (right). Explicit Euler time-stepping with ($\Delta t = 0.001$), and 21 Chebyshev points.

43.2 Use of the Contour-Padé Algorithm with the PS Approach

We now give a brief explanation of how the Contour-Padé algorithm of [Fornberg and Wright (2004)] can be used to compute RBF differentiation matrices. In its original form the Contour-Padé algorithm allows us to stably evaluate radial basis function interpolants based on infinitely smooth RBFs for extreme choices of the shape parameter ε (in particular $\varepsilon \to 0$). More specifically, the Contour-Padé algorithm uses FFTs and Padé approximations to evaluate the function

$$\hat{\boldsymbol{u}}(\boldsymbol{x}, \varepsilon) = \boldsymbol{b}^T(\boldsymbol{x}, \varepsilon)(A(\varepsilon))^{-1}\boldsymbol{f} \tag{43.1}$$

with $\boldsymbol{b}(\boldsymbol{x}, \varepsilon)_j = \varphi_\varepsilon(\|\boldsymbol{x} - \boldsymbol{x}_j\|)$ at some evaluation point $\boldsymbol{x}$ and $A(\varepsilon)_{i,j} = \varphi_\varepsilon(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|)$ (*c.f.* the discussion in the previous chapter and in Chapter 17). The parameter ε is used to denote the dependence of $\boldsymbol{b}$ and A on the choice of that parameter as a scaling factor in the basic function $\varphi_\varepsilon = \varphi(\varepsilon \cdot)$.

If we evaluate $\hat{u}$ at all of the collocation points $\boldsymbol{x}_i$, $i = 1, \ldots, N$, for some fixed value of ε, then $\boldsymbol{b}^T(\boldsymbol{x}, \varepsilon)$ turns into the matrix $A(\varepsilon)$. In the case of interpolation this exercise is, of course, pointless. However, if the Contour-Padé algorithm is adapted to replace the vector $\boldsymbol{b}^T(\boldsymbol{x}, \varepsilon)$ (corresponding to evaluation at a single point $\boldsymbol{x}$) with the matrix $A_{\mathcal{L}}$ based on the differential operator used earlier (corresponding to evaluation at all collocation points), then

$$A_{\mathcal{L}}(\varepsilon)(A(\varepsilon))^{-1}\boldsymbol{u}$$

computes the values of the (spatial) derivative of $\boldsymbol{u}$ on the collocation points $\boldsymbol{x}_i$. Boundary conditions can then be incorporated later as in the standard pseudospectral approach (see, *e.g.*, [Trefethen (2000)] or our discussion in Section 42.2).

This means that we are able to supply yet another subroutine to compute the differentiation matrix on line 4 of Program 43.3 via the Contour-Padé algorithm.

43.2.1 *Solution of the 1D Transport Equation Revisited*

We use the same example as in Subsection 43.1.1. In this subsection we compare a solution based on the Contour-Padé algorithm for Gaussian RBFs in the limiting case $\varepsilon \to 0$ to the two methods described earlier (based on `DRBF` and `cheb`). All of these approaches use an implicit Euler method with time step $\Delta t = 0.001$ for the time discretization. We point out that for an implicit time-stepping method both the Contour-Padé approach and the `DRBF` approach used earlier, of course, require an inversion of the differentiation matrix. Recall that our theoretical discussion suggested that this is justified as long as we confine ourselves to the limiting case $\varepsilon \to 0$ and one space dimension. We will see that the non-limiting case (using `DRBF`) seems to work just as well.

In Figures 43.2 and 43.3 we plot the maximum errors at time $t = 1$ for a time step $\Delta t = 0.001$) and spatial discretizations consisting of $N + 1 = 7, \ldots, 19$ collocation points. Errors for the Contour-Padé Gaussian RBF solution are on the

left of Figure 43.2 and for the Chebyshev PS solution on the right. The errors for the Gaussian RBF solution with N-dependent "optimal" shape parameter are shown in the left part of Figure 43.3, while the corresponding "optimal" ε-values are displayed in the right plot. They range almost linearly increasing from 0.122661 at $N = 6$ to 1.566594 at $N = 18$.

We can see that the errors for all three methods are virtually identical. Unfortunately, in this experiment we are limited to this small range of N since for $N \geq 19$ the Contour-Padé solution becomes unreliable. However, the agreement of all three solutions for these small values of N is remarkable. In fact, this seems to indicate that the errors in the solution are mostly due to the time-stepping method used.

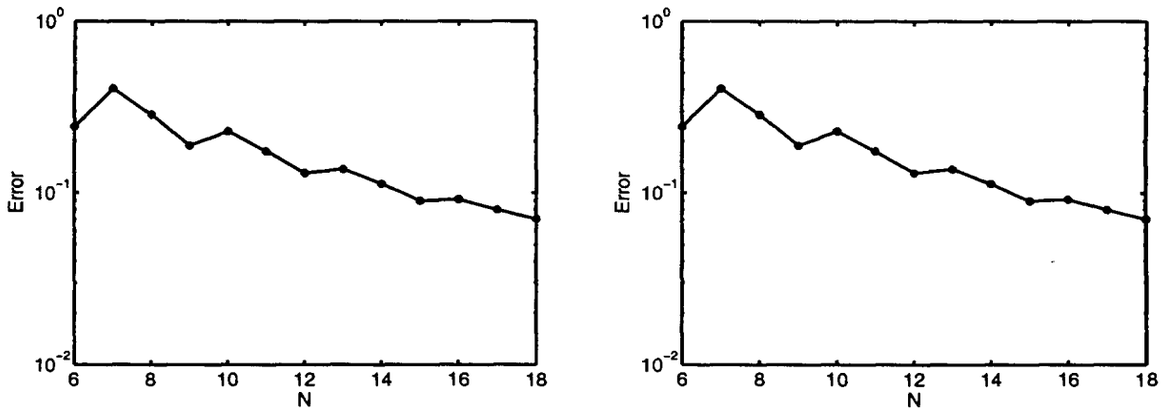


Fig. 43.2 Errors at $t = 1$ for transport equation. Gaussian RBF with $\varepsilon = 0$ (left) and Chebyshev PS-solution (right); variable spatial discretization N. Implicit Euler method with $\Delta t = 0.001$.

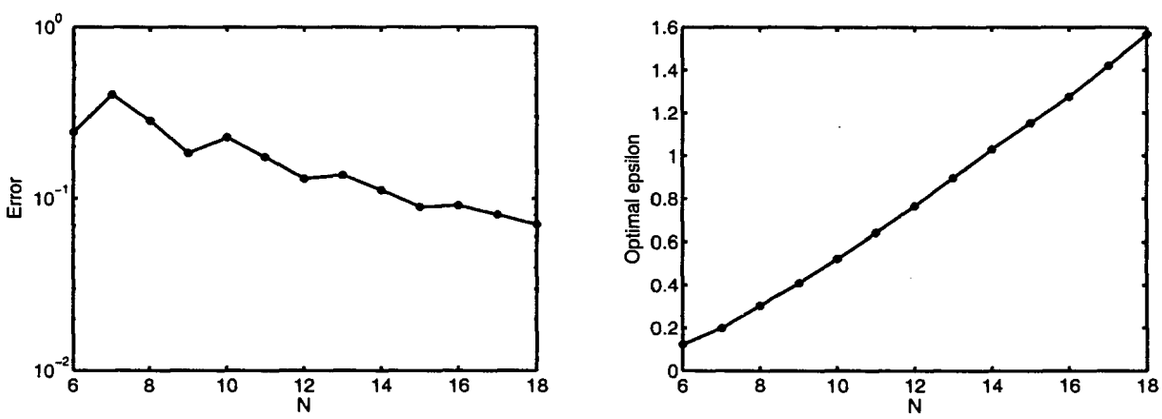


Fig. 43.3 Errors at $t = 1$ for transport equation using Gaussian RBF with "optimal" ε (left) and corresponding ε-values (right); variable spatial discretization N. Implicit Euler method with $\Delta t = 0.001$.

The spectra of the differentiation matrices for both the Gaussian Contour-Padé and the Chebyshev PS approaches are plotted in Figures 43.4 and 43.5, respectively.

The subplots correspond to the use of $N + 1 = 5, 9, 13, 17$ Chebyshev collocation points for the spatial discretization. The plots for the Gaussian and Chebyshev methods show some similarities, but also some differences. The general distribution of the eigenvalues for the two methods is quite similar. However, the spectra for the Contour-Padé algorithm with Gaussian RBFs seem to be more or less a slightly stretched reflection about the imaginary axis of the spectra of the Chebyshev pseudospectral method. The differences increase as N increases. This, however, is not surprising since the Contour-Padé algorithm is known to be unreliable for larger values of N.

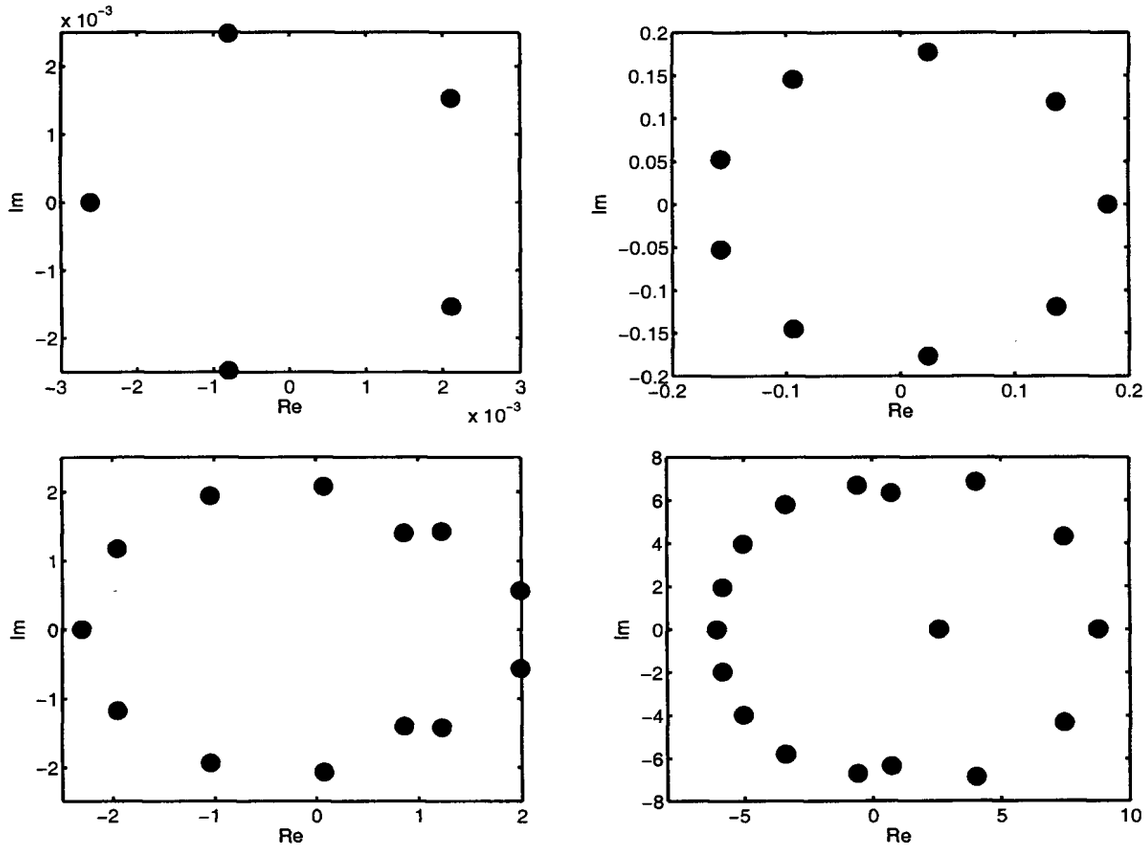


Fig. 43.4 Spectra of differentiation matrices for Gaussian RBF with $\varepsilon = 0$ on Chebyshev collocation points obtained with the Contour-Padé algorithm and $N = 5, 9, 13, 17$.

43.3 Computation of Higher-Order Derivatives

A rather nice feature of polynomial differentiation matrices is the fact that higher-order derivatives can be computed by repeatedly applying the first-order differentiation matrix, *i.e.*, $D^{(k)} = D^k$, where D is the standard first-order differentiation matrix and $D^{(k)}$ is the matrix corresponding to the k-th (univariate) derivative. Unfortunately, this nice feature does not carry over to the general RBF case (just

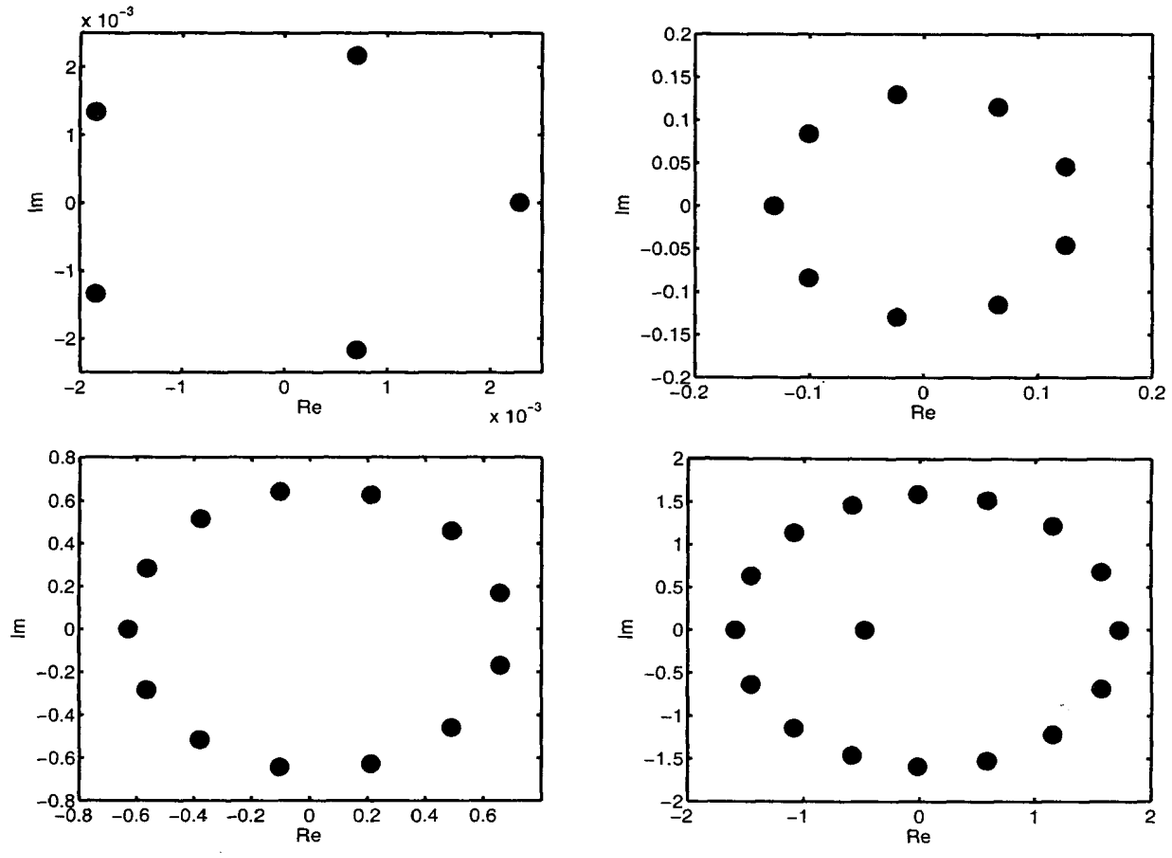


Fig. 43.5 Spectra of differentiation matrices for Chebyshev pseudospectral method on Chebyshev collocation points with $N = 5, 9, 13, 17$.

as is does not hold for periodic Fourier spectral differentiation matrices, either). We therefore need to provide separate MATLAB code for higher-order differentiation matrices. As Program 43.4 shows, this is not fundamentally more complicated than the first-order case. The only differences between Programs 43.1 and 43.4 are given by the computation of the $A_{D^{(k)}}$ matrix on line 10 for the first-order case in Program 43.1 and lines 9 for the second-order case in Program 43.4, and by the use of the subroutine `CostEpsilonD2RBF` instead of `CostEpsilonDRBF`. These differences are minute, and essentially all that is needed is the appropriate formula for the derivative of the RBF passed to `D2RBF` via the parameter `d2rbf`. We do not list the function `CostEpsilonD2RBF`. It differs from `CostEpsilonDRBF` only in the definition of the right-hand side matrix which now becomes

```
4  rhs = d2rbf(ep,r)';
```

Also, the number and type of parameters that are passed to the functions are different since the first-order derivative requires differences of collocation points and the second-order derivative does not.

Program 43.4. D2RBF.m

```
% [D2,x] = D2RBF(N,rbf,d2rbf)
% Computes the second-order differentiation matrix D2 for 1-D
% derivative using Chebyshev points and LOOCV for optimal epsilon
% Input: N, number of points -1
%        rbf, d2rbf, function handles for rbf and its derivative
% Calls on: DistanceMatrix, DifferenceMatrix
% Requires: CostEpsilonD2RBF
 1  function [D2,x] = D2RBF(N,rbf,d2rbf)
 2  if N==0, D2=0; x=1; return, end
 3  x = cos(pi*(0:N)/N)';   % Chebyshev points
 4  mine = .1; maxe = 10;   % Shape parameter interval
 5  r = DistanceMatrix(x,x);
 6  ep = fminbnd(@(ep) CostEpsilonD2RBF(ep,r,rbf,d2rbf),mine,maxe);
 7  fprintf('Using epsilon = %f\n', ep)
 8  A = rbf(ep,r);
 9  AD2 = d2rbf(ep,r);
10  D2 = AD2/A;
```

43.3.1 *Solution of the Allen-Cahn Equation*

To illustrate the use of the subroutine D2RBF.m we present a modification of Program 35 in [Trefethen (2000)] which is concerned with the solution of the nonlinear reaction-diffusion (or Allen-Cahn) equation. The specific problem we will solve is of the form

$$u_t = \mu u_{xx} + u - u^3, \qquad x \in (-1,1),\ t \geq 0,$$

with parameter μ, initial condition

$$u(x,0) = 0.53x + 0.47 \sin\left(-\frac{3}{2}\pi x\right), \qquad x \in [-1,1],$$

and non-homogeneous (time-dependent) boundary conditions $u(-1,t) = -1$ and $u(1,t) = \sin^2(t/5)$. The solution to this equation has three steady states ($u = -1, 0, 1$) with the two nonzero solutions being stable. The transition between these states is governed by the parameter μ. In our calculations below we use $\mu = 0.01$, and the unstable state should vanish around $t = 30$.

The modified MATLAB code is presented in Program 43.5. Note how easily the nonlinearity is dealt with by incorporating it into the time-stepping method on line 13.

Program 43.5. Modification of Program 35 of [Trefethen (2000)]

```
% p35
% Script that solves Allen-Cahn equation with boundary condition
```

```
% imposed explicitly ("method (II)") (from Trefethen (2000))
% We replace the Chebyshev method by an RBF-PS method
% Calls on: D2RBF
    % Matern cubic as RBF basic function
 1  rbf = @(e,r) exp(-e*r).*(15+15*e*r+6*(e*r).^2+(e*r).^3);
 2  d2rbf = @(e,r) e^2*((e*r).^3-3*e*r-3).*exp(-e*r);
 3  N = 20;
 4  [D2,x] = D2RBF(N,rbf,d2rbf);
    % Here is the rest of Trefethen's code.
 5  mu = 0.01; dt = min([.01,50*N^(-4)/mu]);
 6  t = 0; v = .53*x + .47*sin(-1.5*pi*x);
    % Solve PDE by Euler formula and plot results:
 7  tmax = 100; tplot = 2; nplots = round(tmax/tplot);
 8  plotgap = round(tplot/dt); dt = tplot/plotgap;
 9  xx = -1:.025:1; vv = polyval(polyfit(x,v,N),xx);
10  plotdata = [vv; zeros(nplots,length(xx))]; tdata = t;
11  for i = 1:nplots
12     for n = 1:plotgap
13        t = t+dt; v = v + dt*(mu*D2*v + v - v.^3); % Euler
14        v(1) = 1 + sin(t/5)^2; v(end) = -1; % BC
15     end
16     vv = polyval(polyfit(x,v,N),xx);
17     plotdata(i+1,:) = vv; tdata = [tdata; t];
18  end
19  surf(xx,tdata,plotdata), grid on
20  axis([-1 1 0 tmax -1 2]), view(-40,55)
21  colormap('default'); xlabel x, ylabel t, zlabel u
```

The original program in [Trefethen (2000)] is obtained by deleting lines 1–2 and replacing line 4 by a call to `cheb.m` followed by the statement `D2 = D^2` which yields the second-order differentiation matrix in the Chebyshev case.

Note that in our RBF-PS implementation the majority of the matrix computations are required only once outside the time-stepping procedure when computing the derivative matrix as the solution of a linear system. Inside the time-stepping loop (lines 12–15) we require only matrix-vector multiplication. We point out that this approach is much more efficient than computation of RBF expansion coefficients at every time step (as suggested, *e.g.*, in [Hon and Mao (1999)]). In fact, this is the main difference between the RBF-PS approach and the collocation approach of Chapters 38–40 (see also our comparison of the collocation approaches and the RBF-PS approach in the previous chapter).

In Figure 43.6 we show the solution obtained via the Chebyshev pseudospectral method and via an RBF pseudospectral approach based on the "cubic" Matérn function $\varphi(r) = (15 + 15\varepsilon r + 6(\varepsilon r)^2 + (\varepsilon r)^3)e^{-\varepsilon r}$ with "optimal" shape parameter

$\varepsilon = 0.350952$. Note that these computations are rather sensitive to the value of ε and the norm used to measure the "cost" of ε in `CostEpsilonD2RBF.m`. In fact, use of the ℓ_1 or ℓ_∞ norms instead of the ℓ_2 norm both lead to inacceptable results for this test problem. The reasons for this high sensitivity of the solution to the value of ε are the extreme ill-conditioning of the matrix along with the changes of the solution over time. An adaptive method would most likely perform much better in this case.

The computations for this example are based on 21 Chebyshev points, and the differentiation matrix for the RBF is obtained directly with the subroutine `D2RBF.m` (*i.e.*, without the Contour-Padé algorithm). We use this approach since for 21 points the Contour-Padé algorithm no longer can be relied upon. Moreover, it is apparent from the figures that reasonable solutions can also be obtained via this direct (and much simpler) RBF approach. True spectral accuracy, however, will no longer be given if $\varepsilon > 0$. We can see from the figure that the solution based on Chebyshev polynomials appears to be slightly more accurate since the transition occurs at a slightly later and correct time (*i.e.*, at $t \approx 30$) and is also a little "sharper".

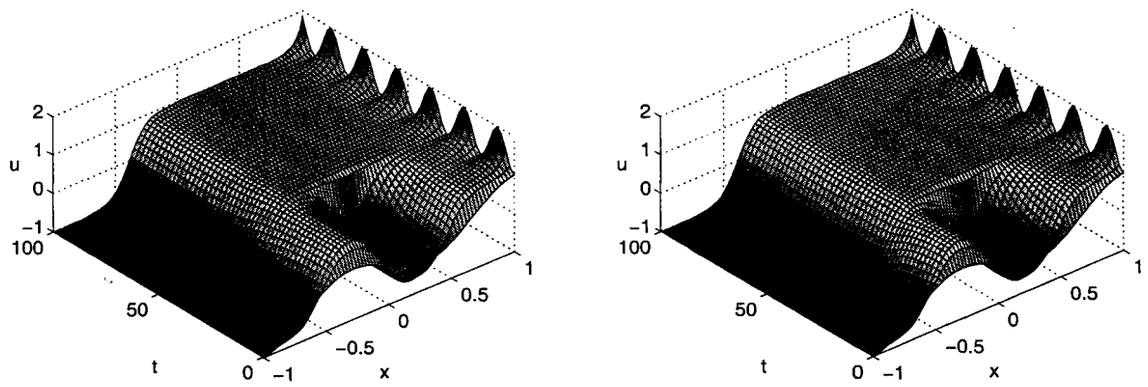


Fig. 43.6 Solution of the Allen-Cahn equation using the Chebyshev pseudospectral method (left) and an RBF-PS method with cubic Matérn functions (right) with $N = 20$.

43.4 Solution of a 2D Helmholtz Equation

We consider the 2D Helmholtz equation (see Program 17 in [Trefethen (2000)])

$$u_{xx} + u_{yy} + k^2 u = f(x, y), \quad x, y \in (-1, 1)^2,$$

with boundary condition $u = 0$ and

$$f(x, y) = \exp\left(-10\left[(y-1)^2 + (x - \frac{1}{2})^2\right]\right).$$

To solve this type of (elliptic) problem we again need to assume invertibility of the differentiation matrix. Even though this may not be warranted theoretically

(see our discussion in the previous chapter), we compare a non-symmetric RBF pseudospectral method with a Chebyshev pseudospectral method.

We attempt to solve the problem with radial basis functions in two different ways. First, we apply the same technique as in [Trefethen (2000)] using the `kron` function to express the disretized Laplacian on a tensor-product grid of $(N+1) \times (N+1)$ points as

$$L = I \otimes D2 + D2 \otimes I, \tag{43.2}$$

where $D2$ is the (univariate) second-order differentiation matrix, I is an identity matrix of size $(N+1) \times (N+1)$, and $\otimes$ denotes the *Kronecker tensor-product*. For polynomial PS methods the second-order differentiation matrix can be computed as the square of the one for the first-order derivative, *i.e.*, $D2 = D^2$, and this is what is used in [Trefethen (2000)].

As we pointed out earlier, for RBFs we cannot follow this approach directly since $D^2 \neq D^{(2)}$. Thus, we generate the matrix $D2$ directly with the help of the subroutine `D2RBF`. However, as long as the collocation points form a tensor-product grid and the RBF is separable (such as a Gaussian or a polynomial), we can still employ the Kronecker tensor-product construction (43.2). This is implemented in lines 4 and 9 of Program 43.6

Program 43.6. Modification of Program 17 of [Trefethen (2000)]

```
% p17
% Script that solves Helmholtz equation
% u_xx + u_yy + (k^2)u = f       on [-1,1]x[-1,1]
% We replace the Chebyshev method by an RBF-PS method
% and explicitly enforce the boundary conditions
% Calls on: D2RBF
    % Gaussian RBF basic function
 1  rbf = @(e,r) exp(-(e*r).^2);
 2  d2rbf = @(e,r) 2*e^2*(2*(e*r).^2-1).*exp(-(e*r).^2);
 3  N = 24;
 4  [D2,x] = D2RBF(N,rbf,d2rbf); y = x;
 5  [xx,yy] = meshgrid(x,y);
 6  xx = xx(:); yy = yy(:);
 7  I = eye(N+1);
 8  k = 9;
 9  L = kron(I,D2) + kron(D2,I) + k^2*eye((N+1)^2);
    % Impose boundary conditions by replacing appropriate rows of L
10  b = find(abs(xx)==1 | abs(yy)==1);              % boundary pts
11  L(b,:) = zeros(4*N,(N+1)^2); L(b,b) = eye(4*N);
12  f = exp(-10*((yy-1).^2+(xx-.5).^2));
13  f(b) = zeros(4*N,1);
    % Solve for u, reshape to 2D grid, and plot:
```

```
14  u = L\f;
15  uu = reshape(u,N+1,N+1);
16  [xx,yy] = meshgrid(x,y);
17  [xxx,yyy] = meshgrid(-1:.0333:1,-1:.0333:1);
18  uuu = interp2(xx,yy,uu,xxx,yyy,'cubic');
19  figure, clf, surf(xxx,yyy,uuu),
20  xlabel x, ylabel y, zlabel u
21  text(.2,1,.022,sprintf('u(0,0) = %13.11f',uu(N/2+1,N/2+1)))
```

The solution of the Helmholtz equation for $k = 9$ with Gaussians using an "optimal" shape parameter $\varepsilon = 2.549845$ and $N = 24$ (*i.e.*, 625 total points) is displayed next to the Chebyshev pseudospectral solution of [Trefethen (2000)] in Figure 43.7. Again, the similarity of the two solutions is remarkable.

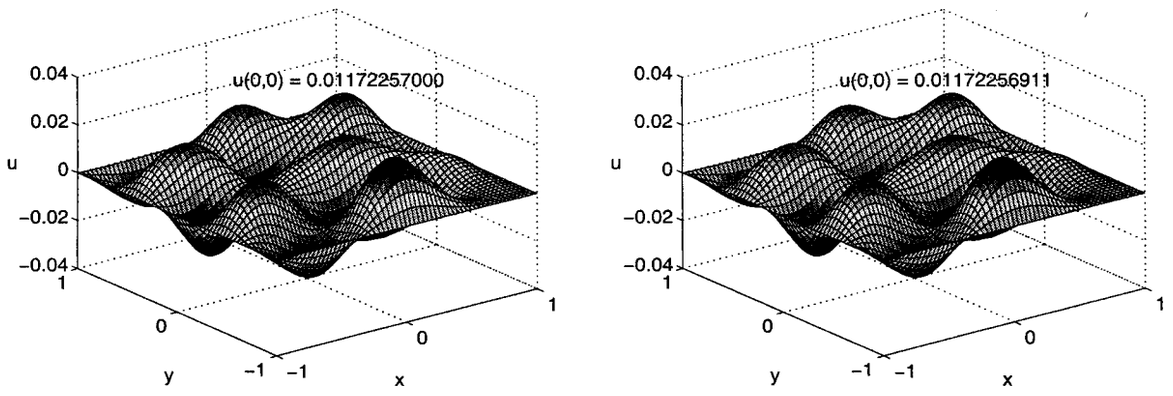


Fig. 43.7 Solution of the 2D Helmholtz equation with $N = 24$ using the Chebyshev pseudospectral method (left) and Gaussians with $\varepsilon = 2.549845$ (right).

As an alternative approach — that allows also the use of non-tensor product collocation grids — we modify Program 43.6 and use a direct implementation of the Laplacian of the RBFs. The only advantage of doing this on a tensor-product grid is that now all radial basis functions can be used. This variation of the code takes considerably longer to execute since the differentiation matrix is now computed with matrices of size 625×625 instead of the 25×25 matrices used for the univariate differentiation matrix $D2$ earlier. Moreover, the results are likely to be less accurate since the larger matrices are more prone to ill-conditioning. However, the advantage of this approach is that it frees us of the limitation of polynomial PS methods to tensor-product collocation grids.

The modified code is listed in Program 43.7 where we have used the C^6 Wendland function $\varphi_{3,3}(r) = (1-\varepsilon r)_+^8(32(\varepsilon r)^3 + 25(\varepsilon r)^2 + 8\varepsilon r + 1)$ with an "optimal" scale parameter $\varepsilon = 0.129440$. Note that we used the compactly supported Wendland functions in "global mode" (with small ε, *i.e.*, large support size) and this explains the definition of the basic function as in lines 1 and 2 of Program 43.7 in preparation

for the use with the dense code `DistanceMatrix.m` in the subroutine `LRBF.m` (which is listed below as Program 43.8). The output of Program 43.7 is displayed in Figure 43.8.

Program 43.7. Modification II of Program 17 of [Trefethen (2000)]

```
% p17_2D
% Script that solves Helmholtz equation
% u_xx + u_yy + (k^2)u = f    on [-1,1]x[-1,1]
% We replace the Chebyshev method by an RBF-PS method,
% explicitly enforce the boundary conditions, and
% use a 2-D implementation of the Laplacian
% Calls on: LRBF
    % Wendland C6 RBF basic function
 1  rbf = @(e,r) max(1-e*r,0).^8.*(32*(e*r).^3+25*(e*r).^2+8*e*r+1);
 2a Lrbf = @(e,r) 44*e^2*max(1-e*r,0).^6.*...
 2b                (88*(e*r).^3+3*(e*r).^2-6*e*r-1);
 3  [L,x,y] = LRBF(N,rbf,Lrbf);
 4  [xx,yy] = meshgrid(x,y);
 5  xx = xx(:); yy = yy(:);
 6  k = 9;
 7  L = L + k^2*eye((N+1)^2);
    % Impose boundary conditions by replacing appropriate rows of L
 8  b = find(abs(xx)==1 | abs(yy)==1);              % boundary pts
 9  L(b,:) = zeros(4*N,(N+1)^2); L(b,b) = eye(4*N);
10  f = exp(-10*((yy-1).^2+(xx-.5).^2));
11  f(b) = zeros(4*N,1);
    % Solve for u, reshape to 2D grid, and plot:
12  u = L\f;
13  uu = reshape(u,N+1,N+1);
14  [xx,yy] = meshgrid(x,y);
15  [xxx,yyy] = meshgrid(-1:.0333:1,-1:.0333:1);
16  uuu = interp2(xx,yy,uu,xxx,yyy,'cubic');
17  figure, clf, surf(xxx,yyy,uuu),
18  xlabel x, ylabel y, zlabel u
19  text(.2,1,.022,sprintf('u(0,0) = %13.11f',uu(N/2+1,N/2+1)))
```

Program 43.8. `LRBF.m`

```
% [L,x,y] = LRBF(N,rbf,Lrbf)
% Computes the Laplacian differentiation matrix L for 2-D
% derivatives using Chebyshev points and LOOCV for optimal epsilon
% Input: N number of points -1
%        rbf, Lrbf, function handles for rbf and its derivative
```

```
% Calls on: DistanceMatrix
% Requires: CostEpsilonLRBF
 1  function [L,x,y] = LRBF(N,rbf,Lrbf)
 2  if N==0, L=0; x=1; return, end
 3  x = cos(pi*(0:N)/N)';   % Chebyshev points
 4  y = x; [xx,yy] = meshgrid(x,y);
    % Stretch 2D grids to 1D vectors and put in one array
 5  points = [xx(:) yy(:)];
 6  mine = .1; maxe = 10;   % Shape parameter interval
 7  r = DistanceMatrix(points,points);
 8  ep = fminbnd(@(ep) CostEpsilonLRBF(ep,r,rbf,Lrbf),mine,maxe);
 9  fprintf('Using epsilon = %f\n', ep)
10  A = rbf(ep,r);
11  AL = Lrbf(ep,r);
12  L = AL/A;
```

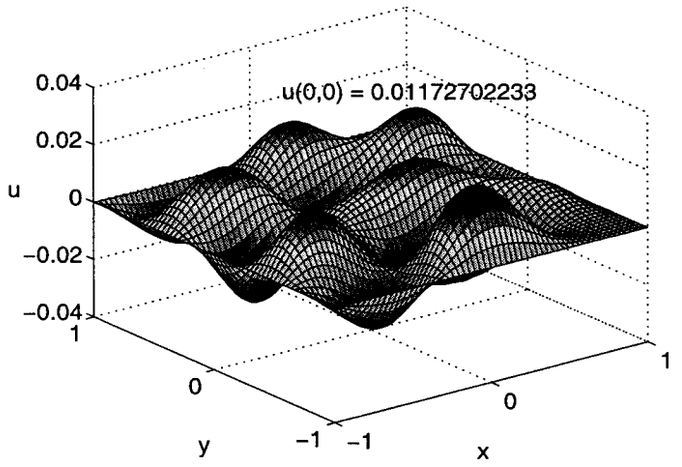


Fig. 43.8 Solution of the 2D Helmholtz equation using a direct implementation of the Laplacian based on C^6 Wendland functions with $\varepsilon = 0.129440$ on 625 tensor-product Chebyshev collocation points.

43.5 Solution of a 2D Laplace Equation with Piecewise Boundary Conditions

Our final example is another elliptic equation. This time we use the Gaussian RBF with an "optimal" shape parameter $\varepsilon = 2.549845$. Again, the spatial discretization consists of a tensor product of 25×25 Chebyshev points, and the differentiation matrix for the RBF-PS approach is computed using the `D2RBF` and `kron` construction as in the previous example.

We consider the 2D Laplace equation

$$u_{xx} + u_{yy} = 0, \quad x, y \in (-1, 1)^2,$$

with boundary conditions

$$u(x,y) = \begin{cases} \sin^4(\pi x), & y = 1 \text{ and } -1 < x < 0, \\ \frac{1}{5}\sin(3\pi y), & x = 1, \\ 0, & \text{otherwise.} \end{cases}$$

This is the same problem as used in Program 36 of [Trefethen (2000)], and we do not list it here due to the similarity with previous examples and the original code in [Trefethen (2000)].

Figure 43.9 shows the solution obtained via the Chebyshev and RBF pseudospectral methods, respectively. The qualitative behavior of the two solutions is very similar.

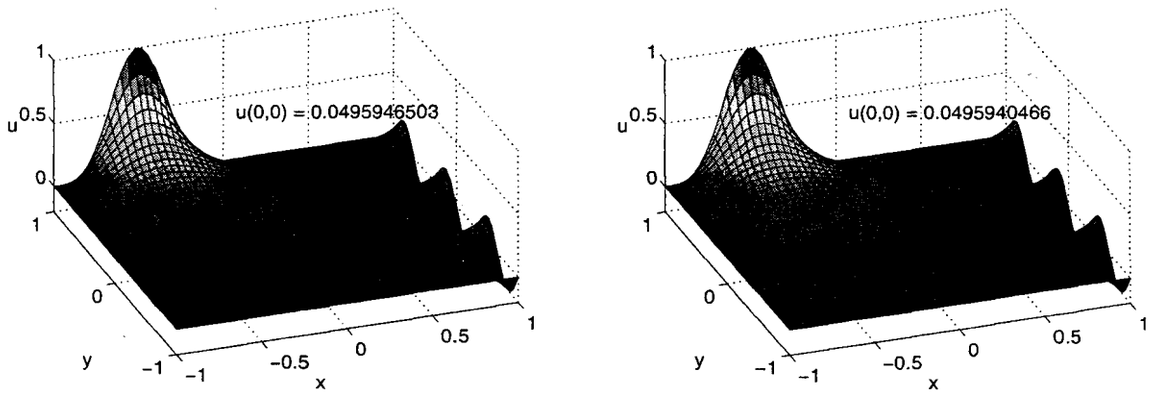


Fig. 43.9 Solution of the 2D Laplace equation using a Chebyshev PS approach (left) and Gaussian RBFs (right) with $\varepsilon = 2.549845$ on 625 tensor-product Chebyshev collocation points.

43.6 Summary

While there is no advantage in going to arbitrarily spaced irregular collocation points for any of the problems presented here, there is nothing that prevents us from doing so for the RBF pseudospectral approach. In particular, as we saw in Section 43.4, we are not limited to using tensor product grids for higher-dimensional spatial discretizations. This is a potential advantage of the RBF pseudospectral approach over the standard polynomial methods.

More applications of the RBF-PS method can be found in the recent papers [Ferreira and Fasshauer (2006); Ferreira and Fasshauer (2007)].

Future challenges include dealing with larger problems in an efficient and stable way. Thus, such issues as preconditioning and FFT-type algorithms need to

be studied in the context of RBF pseudospectral methods. Some first studies of the eigenvalue stability of RBF pseudospectral methods have been reported very recently in [Platte and Driscoll (2006)].

Chapter 44

RBF Galerkin Methods

44.1 An Elliptic PDE with Neumann Boundary Conditions

A variational approach to the solution of PDEs with RBFs in Euclidean spaces has so far only been considered in [Wendland (1999a); Wendland (1999b)] and the very recent paper [Hu *et al.* (2005)]. On the sphere — where we do not have to worry about boundary conditions — we also have [Le Gia (2004)]. In [Wendland (1999b)] the author studies the Helmholtz equation with natural boundary conditions, *i.e.*,

$$\begin{aligned} -\Delta u + u &= f && \text{in } \Omega, \\ \frac{\partial}{\partial \boldsymbol{n}} u &= 0 && \text{on } \partial\Omega, \end{aligned}$$

where $\boldsymbol{n}$ denotes the unit outer normal vector.

The classical Galerkin formulation then leads to the problem of finding a function $u \in H^1(\Omega)$ such that

$$a(u, v) = (f, v)_{L_2(\Omega)} \qquad \text{for all } v \in H^1(\Omega),$$

where $(f, v)_{L_2(\Omega)}$ is the usual L_2-hnner product, and for the Helmholtz equation the bilinear form a is given by

$$a(u, v) = \int_\Omega (\nabla u \cdot \nabla v + uv) d\boldsymbol{x}.$$

In order to obtain a numerical scheme the infinite-dimensional space $H^1(\Omega)$ is replaced by some finite-dimensional subspace $\mathcal{S}_\mathcal{X} \subseteq H^1(\Omega)$, where $\mathcal{X}$ denotes the computational grid to be used for the solution. In the context of RBFs $\mathcal{S}_\mathcal{X}$ is taken as

$$\mathcal{S}_\mathcal{X} = \text{span}\{\varphi(\| \cdot - \boldsymbol{x}_j \|),\ \boldsymbol{x}_j \in \mathcal{X}\}.$$

This results in a square system of linear equations for the coefficients of $\hat{u} \in \mathcal{S}_\mathcal{X}$ determined by

$$a(\hat{u}, v) = (f, v)_{L_2(\Omega)} \qquad \text{for all } v \in \mathcal{S}_\mathcal{X}. \tag{44.1}$$

More specifically, if $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, then

$$\hat{u} = \sum_{j=1}^{N} c_j \varphi(\| \cdot - \boldsymbol{x}_j \|),$$

and the system (44.1) is given by

$$\int_\Omega [\nabla \hat{u}(\boldsymbol{x}) \cdot \nabla \varphi(\|\boldsymbol{x} - \boldsymbol{x}_i\|) + \hat{u}\varphi(\|\boldsymbol{x} - \boldsymbol{x}_i\|)]\, d\boldsymbol{x} = \int_\Omega f(\boldsymbol{x})\varphi(\|\boldsymbol{x} - \boldsymbol{x}_i\|)d\boldsymbol{x}, \quad i = 1, \ldots, N.$$

Using linearity and the definition of $\hat{u}$ given above this turns into

$$\sum_{j=1}^{N} c_j \left\{ \int_\Omega [\nabla \varphi(\|\boldsymbol{x} - \boldsymbol{x}_j\|) \cdot \nabla \varphi(\|\boldsymbol{x} - \boldsymbol{x}_i\|) + \varphi(\|\boldsymbol{x} - \boldsymbol{x}_j\|)\varphi(\|\boldsymbol{x} - \boldsymbol{x}_i\|)]\, d\boldsymbol{x} \right\}$$

$$= \int_\Omega f(\boldsymbol{x})\varphi(\|\boldsymbol{x} - \boldsymbol{x}_i\|)d\boldsymbol{x}, \qquad i = 1, \ldots, N.$$

Clearly, this can be written in matrix-vector form as

$$A\boldsymbol{c} = \boldsymbol{f}$$

with the entries of the *stiffness matrix* A given by

$$A_{ij} = \int_\Omega [\nabla \varphi(\|\boldsymbol{x} - \boldsymbol{x}_j\|) \cdot \nabla \varphi(\|\boldsymbol{x} - \boldsymbol{x}_i\|) + \varphi(\|\boldsymbol{x} - \boldsymbol{x}_j\|)\varphi(\|\boldsymbol{x} - \boldsymbol{x}_i\|)]\, d\boldsymbol{x},$$

and the right-hand side entries

$$\boldsymbol{f}_i = \int_\Omega f(\boldsymbol{x})\varphi(\|\boldsymbol{x} - \boldsymbol{x}_i\|)d\boldsymbol{x}.$$

The evaluation of these integrals is what is most time-consuming in the RBF Galerkin approach (see the numerical experiments of the next chapter). Wendland reports that the numerical evaluation of these weak-form integrals presents a major problem for the radial basis function Galerkin approach.

In addition, RBF Galerkin methods will face difficulties with Dirichlet (or sometimes also called *essential*) boundary conditions. Both of these difficulties are also well-known in many other flavors of meshfree weak-form methods. An especially promising solution to the issue of Dirichlet boundary conditions seems to be the use of R-functions as proposed by Höllig and Reif in the context of web-splines (see, *e.g.*, [Höllig (2003)] or our earlier discussion in the context of collocation methods in Chapter 38). Another popular approach uses Lagrange multipliers in a constrained optimization setting.

For more on the Galerkin method see, *e.g.*, [Braess (1997); Brenner and Scott (1994)] (in the context of finite elements), or [Babuška *et al.* (2003)] (in the context of MLS-based meshfree methods).

44.2 A Convergence Estimate

It was shown in [Wendland (1999a)] that for those RBFs (globally as well as locally supported) whose Fourier transform decays like $(1 + \|\cdot\|_2)^{-2\beta}$ the following convergence estimate for the RBF Galerkin method holds:

$$\|u - \hat{u}\|_{H^1(\Omega)} \leq Ch^{\sigma-1}\|u\|_{H^\sigma(\Omega)}, \tag{44.2}$$

where $h = h_{\mathcal{X},\Omega}$ is the fill distance of $\mathcal{X}$, the solution satisfies the regularity requirements $u \in H^\sigma(\Omega)$, and where the convergence rate is determined by $\beta \geq \sigma > s/2+1$.

From our discussion in Chapter 13 we know that the Fourier transform of Wendland's compactly supported functions decays as $(1 + \|\cdot\|_2)^{-s-2\kappa-1}$. So for these functions the above estimate implies that functions which are in $C^{2\kappa}$ and strictly positive definite on $\mathbb{R}^s$ satisfying $\kappa \geq \sigma - \frac{s+1}{2}$ will have $\mathcal{O}(h^{\kappa+(s-1)/2})$ convergence order, *i.e.*, the C^0 function $\varphi_{3,0} = (1-r)_+^2$ yields $\mathcal{O}(h)$ and the C^2 function $\varphi_{3,1} = (1-r)_+^4(4r+1)$ delivers $\mathcal{O}(h^2)$ convergence in $\mathbb{R}^3$.

As with the convergence estimate for symmetric collocation there is a link between the regularity requirements on the solution and the space dimension s. Also, we point out that so far the theory is only established for PDEs with natural boundary conditions.

The convergence estimate (44.2) holds for the non-stationary setting, *i.e.*, if we are using compactly supported basis functions, for fixed support radii. By the same arguments used in Chapters 12, 16 and 41, one will want to switch to the stationary setting and employ a multilevel algorithm in which the solution at each step is updated by a fit to the most recent residual. This should ensure both convergence and numerical efficiency.

44.3 A Multilevel RBF Galerkin Algorithm

Here is the variant of the stationary multilevel collocation algorithm listed in Chapter 41 adapted for the weak formulation of the PDE discussed at the beginning of this chapter (see [Wendland (1999b)]):

Algorithm 44.1. Multilevel Galerkin

(1) $u_0 = 0$
(2) For k from 1 to K do
   (a) Find $u_k \in \mathcal{S}_{\mathcal{X}_k}$ such that $a(u_k, v) = (f, v) - a(u_{k-1}, v)$ for all $v \in \mathcal{S}_{\mathcal{X}_k}$
   (b) Update $u_k \leftarrow u_{k-1} + u_k$

This algorithm *does not converge in general* (see Tab. 1 in [Wendland (1999b)]).

Since the weak formulation can be interpreted as a Hilbert space projection method, Wendland was able to show that a modified version of the multilevel Galerkin algorithm, namely

Algorithm 44.2. Nested Multilevel Galerkin

(1) Fix K and $M \in \mathbb{N}$, and set $v_0 = 0$.
(2) For j from 0 while *resiudal* $>$ *tolerance* to M do
   (a) Set $u_0 = v_j$.
   (b) Apply the k-loop of the previous algorithm and denote the result with $\tilde{u}(v_j)$.

(c) Set $v_{j+1} = \tilde{u}(v_j)$.

does converge. In fact, using this algorithm Wendland proves, and also observes numerically, convergence which is at least linear (see Theorem 3 and Tab. 2 in [Wendland (1999b)]).

The important difference between the two multilevel Galerkin algorithms is the added outer iteration in the nested version which is a well-known idea from linear algebra introduced in [Kaczmarz (1937)]. A proof of the linear convergence for general Hilbert space projection methods coupled with Kaczmarz iteration can be found in [Smith *et al.* (1977)]. This alternate projection idea is also the fundamental ingredient in the convergence proof of the domain decomposition method of [Beatson *et al.* (2000)] described in the Chapter 35. We mention here that in the multigrid literature Kaczmarz' method is frequently used as a smoother (see *e.g.* [McCormick (1992)]).

In the recent paper [Schaback (2003)] the author presents a framework for the radial basis function solution of problems both in the strong (collocation) and weak (Galerkin) form.

Many other meshfree methods for the solution of partial differential equations in the weak form appear in the (mostly engineering) literature. These methods come under such names as smoothed particle hydrodynamics (SPH) (*e.g.*, [Monaghan (1988)]), reproducing kernel particle method (RKPM) (see, *e.g.*, [Li and Liu (1996); Liu *et al.* (1997)]), point interpolation method (PIM) (see, [Liu (2002)]), element free Galerkin method (EFG) (see, *e.g.*, [Belytschko *et al.* (1996)]), meshless local Petrov-Galerkin method (MLPG) [Atluri and Zhu (1998)], *h-p*-cloud method [Duarte and Oden (1996b)], partition of unity finite element method (PUFEM) [Babuška and Melenk (1997); Melenk and Babuška (1996)], or generalized finite element method (GFEM) [Babuška *et al.* (2003)]. Most of these methods are based on the moving least squares approximation method discussed in Chapter 22. The two recent books [Atluri and Shen(2002a)] and [Liu (2002)] summarize many of these methods. However, these books focus mostly on a survey of the various methods and related computational and implementation issues with little emphasis on the mathematical foundation of these methods. The recent survey paper [Babuška *et al.* (2003)] fills a large part of this void.

Chapter 45

RBF Galerkin Methods in MATLAB

We consider the following Helmholtz test problem (*c.f.* [Wendland (1999b)]):

$$\begin{aligned} -\Delta u(x,y) + u(x,y) &= \cos(\pi x)\cos(\pi y) \qquad \text{in } \Omega = [-1,1]^2, \\ \frac{\partial}{\partial \boldsymbol{n}} u(x,y) &= 0 \qquad \text{on } \partial\Omega, \end{aligned}$$

where $\boldsymbol{x} = (x,y) \in \mathbb{R}^2$ and $\boldsymbol{n}$ denotes the unit outer normal vector. It is easy to verify that the exact solution for this problem is given by

$$u(x,y) = \frac{\cos(\pi x)\cos(\pi y)}{2\pi^2 + 1}.$$

In Program 45.1 we provide a simple MATLAB implementation for the Galerkin solution of this problem. Note that our program does not attempt to provide a multilevel solution as described in the previous chapter, nor do we pretend to be especially efficient (and therefore the program is very slow). As pointed out in the previous chapter, the most time consuming part is the calculation of the integrals needed for the stiffness matrix A with entries

$$\begin{aligned} A_{ij} = &\int_{[-1,1]^2} \nabla\varphi(\|\boldsymbol{x} - \boldsymbol{x}_i\|) \cdot \nabla\varphi(\|\boldsymbol{x} - \boldsymbol{x}_j\|) d\boldsymbol{x} \\ &+ \int_{[-1,1]^2} \varphi(\|\boldsymbol{x} - \boldsymbol{x}_i\|)\varphi(\|\boldsymbol{x} - \boldsymbol{x}_j\|) d\boldsymbol{x}, \end{aligned}$$

and the right-hand side vector with entries

$$\int_{[-1,1]^2} f(\boldsymbol{x})\varphi(\|\boldsymbol{x} - \boldsymbol{x}_i\|) d\boldsymbol{x}.$$

We compute these integrals using the `dblquad` numerical integration routine on lines 15–20 of Program 45.1. Note that we exploit the symmetry of the stiffness matrix in the `for-loop`, and then complete the matrix on line 21. The functions needed for the integration are provided on lines 1–3 and 5–6. In [Wendland (1999b)] the author details a strategy for converting the double integrals to univariate integrals since all the functions involved are radially symmetric. We do not pursue that possibility here.

For this example we use the C^2 Wendland function $\varphi_{3,1}(r) = (1-r)_+^4(4r+1)$ with a support scaled by $\varepsilon = 0.7$. On line 4 we provide the standard representation

of the basic function as it is needed for the evaluation and plotting part of the program (lines 23–34, which are of the same form as our earlier programs).

Program 45.1. RBFGalerkin2D.m

```
% RBFGalerkin2D
% Script that performs Galerkin solution of 2D Helmholtz equation
% - u_xx - u_yy + u = f
% Calls on: DistanceMatrix, PlotSurf, PlotError2D
   % Definition of the RBF and its gradient, Wendland C2
 1a rbf = @(e,x,y,xi,yi) max(1-e*sqrt((x-xi).^2+(y-yi).^2),0).^4.*...
 1b                     (4*e*sqrt((x-xi).^2+(y-yi).^2)+1);
 2a dxrbf = @(e,x,y,xi,yi) -20*(x-xi)*e^2.*...
 2b                       max(1-e*sqrt((x-xi).^2+(y-yi).^2),0).^3;
 3a dyrbf = @(e,x,y,xi,yi) -20*(y-yi)*e^2.*...
 3b                       max(1-e*sqrt((x-xi).^2+(y-yi).^2),0).^3;
 4  evalrbf = @(e,r) max(1-e*r,0).^4.*(4*e*r+1);
   % Products for integration
 5  rp = @(e,x,y,xi,yi,xj,yj) rbf(e,x,y,xi,yi).*rbf(e,x,y,xj,yj);
 6a gp = @(e,x,y,xi,yi,xj,yj) dxrbf(e,x,y,xi,yi).*...
 6b       dxrbf(e,x,y,xj,yj)+dyrbf(e,x,y,xi,yi).*dyrbf(e,x,y,xj,yj);
   % Parameter for basis function
 7  ep = .7;
   % Right-hand side function for Helmholtz equation
 8  f = @(x,y) cos(pi*x).*cos(pi*y);
   % Exact solution
 9  u = @(x,y) cos(pi*x).*cos(pi*y)/(2*pi^2+1);
   % Number and type of centers:
10  N = 25; gridtype = 'u';
   % Resolution of evaluation grid for errors and plotting
11  neval = 40;
   % Load data points
12  name = sprintf('Data2D_%d%s', N,gridtype); load(name)
   % Shift centers to the square [-1,1]^2
13  ctrs = 2*dsites-1;
   % Build stiffness matrix and right-hand side
14  A = zeros(N,N); rhs = zeros(N,1);
15  for i=1:N
16     for j=1:i
17a       A(i,j) = dblquad(@(x,y) gp(ep,x,y,ctrs(i,1),ctrs(i,2),...
17b                              ctrs(j,1),ctrs(j,2)),-1,1,-1,1) + ...
17c            dblquad(@(x,y) rp(ep,x,y,ctrs(i,1),ctrs(i,2),...
17d                                 ctrs(j,1),ctrs(j,2)),-1,1,-1,1);
```

```
18      end
19a     rhs(i) = dblquad(@(x,y) f(x,y).*...
19b                      rbf(ep,x,y,ctrs(i,1),ctrs(i,2)),-1,1,-1,1);
20  end
    % Make matrix symmetric
21  A = A + A' - diag(diag(A));
    % Solve linear system, i.e., compute expansion coefficients
22  c = A\rhs;
    % Evaluation
23  grid = linspace(-1,1,neval); [xe, ye] = meshgrid(grid);
24  epoints = [xe(:) ye(:)];
25  exact = u(epoints(:,1),epoints(:,2));
26  DM_eval = DistanceMatrix(epoints,ctrs);
27  EM = evalrbf(ep,DM_eval);
28  Pf = EM * c;
    % Compute maximum error on evaluation grid
29  maxerr = norm(Pf-exact,inf); rms_err = norm(Pf-exact)/neval;
30  fprintf('RMS error:     %e\n', rms_err)
31  fprintf('Maximum error: %e\n', maxerr)
    % Plot approximate solution
32  fview = [-30,30]; % viewing angle for plot
33  PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview);
34  PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview);
```

Errors, condition numbers of the stiffness matrix, and observed convergence rates are listed in Table 45.1. A plot of the approximate solution and error distribution using 81 equally spaced centers to generate the trial and test spaces is provided in Figure 45.1.

Table 45.1 Errors and condition numbers for Galerkin solution of Helmholtz equation using the C^2 Wendland function with $\varepsilon = 0.7$.

N	ℓ_∞-error	rate	RMS-error	rate	cond(A)
9	4.774434e-003		1.013915e-003		8.159139e+000
25	3.223359e-003	0.5668	9.561258e-004	0.0847	1.408312e+002
81	9.346870e-005	5.1079	2.494297e-005	5.2605	3.232525e+004
289	9.701313e-005	-0.0537	2.239018e-005	0.1558	6.897924e+007

We can see that the convergence is rather erratic, and that the condition number increases rapidly. The W_2^1-convergence rate predicted in [Wendland (1999a)] for the basic function used here is $\mathcal{O}(h)$. On average, the results listed in Table 45.1 indicate roughly an $\mathcal{O}(h^2)$ RMS-convergence rate.

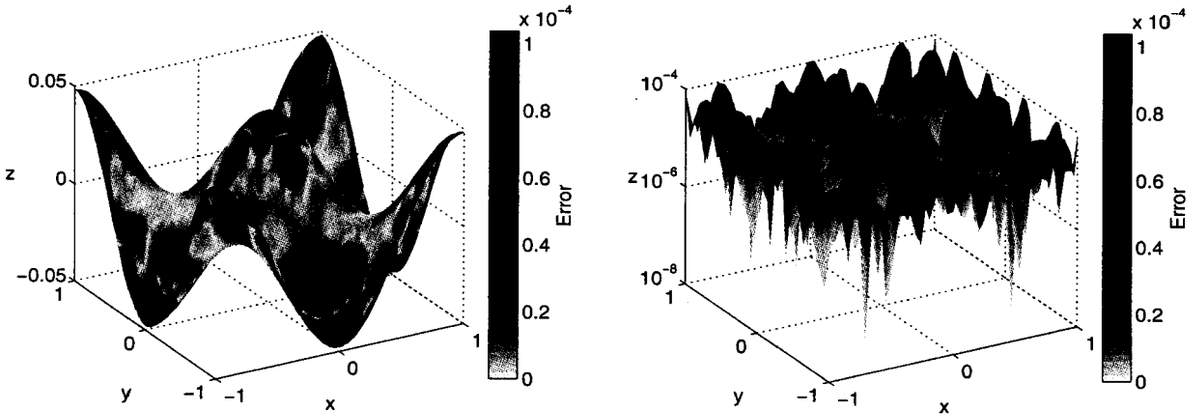


Fig. 45.1 Approximate solution (left) and maximum error (right) for Galerkin solution of Helmholtz equation with C^2 Wendland functions using 81 equally spaced points in $[-1, 1]^2$.

Appendix A

Useful Facts from Discrete Mathematics

A.1 Halton Points

Halton points (see [Halton (1960); Wong *et al.* (1997)]) are created from *van der Corput sequences.* They form so-called low discrepancy sequences and are used frequently in quasi-Monte Carlo methods for multi-dimensional integration applications.

The starting point in the construction of a van der Corput sequence is the fact that every nonnegative integer n can be written uniquely using a prime base p, *i.e.*,

$$n = \sum_{i=0}^{k} a_i p^i,$$

where each coefficient a_i is an integer such that $0 \leq a_i < p$. For example, if $n = 10$ and $p = 3$, then

$$10 = 1 \cdot 3^0 + 0 \cdot 3^1 + 1 \cdot 3^2,$$

so that $k = 2$ and $a_0 = a_2 = 1$ and $a_1 = 0$.

Next we define a function h_p that maps the nonnegative integers to the interval $[0, 1)$ via

$$h_p(n) = \sum_{i=0}^{k} \frac{a_i}{p^{i+1}}.$$

For example

$$h_3(10) = \frac{1}{3} + \frac{1}{3^3} = \frac{10}{27}.$$

The resulting sequence $h_{p,N} = \{h_p(n) : \ n = 0, 1, 2, \ldots, N\}$ is known as a van der Corput sequence. For example

$$h_{3,10} = \{0, 1/3, 2/3, 1/9, 4/9, 7/9, 2/9, 5/9, 8/9, 1/27, 10/27\}.$$

In order to generate a *Halton point set* in s-dimensional space (more precisely in the s-dimensional unit cube $[0, 1)^s$) we take s (usually distinct) primes $p_1, \ldots, p_s$

and use the resulting van der Corput sequences $h_{p_1,N}, \ldots, h_{p_s,N}$ as the coordinates of the s-dimensional Halton points, *i.e.*, the set

$$H_{s,N} = \{(h_{p_1}(n), \ldots, h_{p_s}(n)) : \ n = 0, 1, \ldots, N\}$$

is the set of $N+1$ Halton points in $[0,1)^s$. Halton point sets for $s = 2$ are displayed in Figure 1.1 and the bottom part of Figure 14.5.

An nice property of Halton points is the fact that they are *nested* point sets, *i.e.*, $H_{s,M} \subset H_{s,N}$ for $M < N$. In fact, the point sets can even be constructed sequentially, *i.e.*, one does not need to start over if one wants to add more points to an existing set of Halton points. This distinguishes the Halton points from the related Hammersley points.

It is known that in low space dimensions, the multi-dimensional Halton sequence quickly "fills up" the unit cube in a well-distributed pattern. However, for higher dimensions (such as $s = 40$), using a relatively small value of N results in poorly distributed Halton points. Only when N is large enough relative to s do the points become well-distributed. Since none of our examples exceed $s = 6$ this is not a concern for us.

In the MATLAB programs throughout this book we use the function `haltonseq` written by Daniel Dougherty. This function can be downloaded from the MATLAB Central File Exchange, see [MCFE]. In this implementation of Halton sequences the origin is not part of the point set, *i.e.*, the Halton points are generated starting with $n = 1$ instead of $n = 0$ as described above.

A.2 *kd*-Trees

In order to deal with large sets of data efficiently we frequently use *compactly supported* basic functions (see, *e.g.*, Chapter 12). For their successful implementation certain geometric information is required. Most importantly, we need to know which data sites lie in the support of a given basis function. Such a query is known as a *range search*. We also may be interested in finding all centers whose support contains a given (evaluation) point $\boldsymbol{x}$. Such a query is known as a *containment query*. Furthermore, we might also be interested in finding the (n) *nearest neighbors* of a given point (for instance if we need to find the separation distance $q_{\mathcal{X}}$ of a set of points $\mathcal{X}$). One way to accomplish these tasks is via *kd-trees*. A *kd*-tree (short for k-dimensional tree) is a space-partitioning data structure for organizing points in k-dimensional space. Thus, if we were to be true to the notation used throughout this book, we should technically be referring to these trees as *sd*-trees. We will, however, stick with the usual terminology and refer to them as *kd*-trees.

The purpose of *kd*-trees is to hierarchically decompose a set of N data points in $\mathbb{R}^s$ into a relatively small number of subsets such that each subset contains roughly the same number of data sites. Each node in the tree is defined by a splitting plane that is perpendicular to one of the coordinate axes and passes through one of the

data points. Therefore the splitting planes partition the set of points at the median into "left" and "right" (or "top" and "bottom") subsets, each with roughly half the points of the parent node. These children are again partitioned into equal halves, using planes through a different dimension (usually one keeps on cycling through the dimensions when determining the next splitting plane). This partitioning process stops after $\log N$ levels. In the end every node of the *kd*-tree, from the root to the leaves, stores a point. The computational complexity for building a *kd*-tree from N points in $\mathbb{R}^s$ is $\mathcal{O}(sN \log N)$. Once the tree is built, a range query can be performed in $\mathcal{O}(\log N)$ time. This compares favorably with the $\mathcal{O}(N)$ time it would take to search the "raw" data set.

In our MATLAB examples we use the functions `kdtree` and `kdrangequery` from the *kd*-tree library (given as a set of MATLAB MEX-files written by Guy Shechter that can be downloaded from the MATLAB Central File Exchange, see [MCFE]).

Figure A.1 shows a standard median-based partitioning of nine Halton points in $[0,1]^2$ on the left along with the associated *kd*-tree on the right.

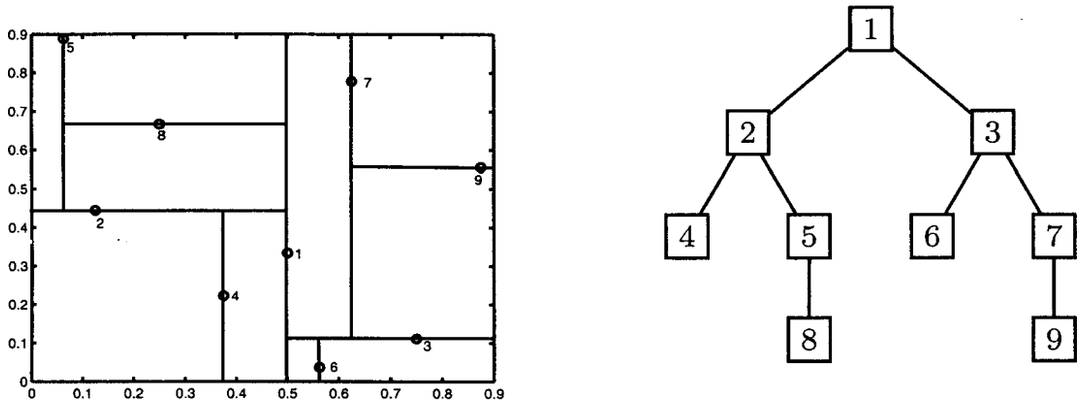


Fig. A.1 *kd* partitioning (left) and tree (right) for 9 Halton points.

Appendix B

Useful Facts from Analysis

B.1 Some Important Concepts from Measure Theory

Bochner's theorem (*c.f.* Theorem 3.3) and a number of other results are formulated in terms of *Borel measures.*

Since we refer to the book [Wendland (2005a)] for many of the theoretical results presented in this book we follow the exposition in [Wendland (2005a)]. We start with an arbitrary set X, and denote the set of all subsets of X by $\mathcal{P}(X)$. The empty set is denoted by $\emptyset$.

Definition B.1. A subset $\mathcal{A}$ of $\mathcal{P}(X)$ is called a *σ-algebra* on X if

(1) $X \in \mathcal{A}$,
(2) $A \in \mathcal{A}$ implies that its complement (in X) is also contained in $\mathcal{A}$,
(3) $A_i \in \mathcal{A}$, $i \in \mathbb{N}$, implies that the union of these sets is contained in $\mathcal{A}$.

Definition B.2. Given an arbitrary set X and a σ-algebra $\mathcal{A}$ of subsets of X, a *measure* on $\mathcal{A}$ is a function $\mu : \mathcal{A} \to [0, \infty]$ such that

(1) $\mu(\emptyset) = 0$,
(2) for any sequence $\{A_i\}$ of disjoint sets in $\mathcal{A}$ we have

$$\mu(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} \mu(A_i).$$

Definition B.3. If X is a topological space, and $\mathcal{O}$ is the collection of open sets in X, then the σ-algebra generated by $\mathcal{O}$ is called the *Borel σ-algebra* and denoted by $\mathcal{B}(X)$. If in addition X is a Hausdorff space, then a measure μ defined on $\mathcal{B}(X)$ that satisfies $\mu(K) < \infty$ for all compact sets $K \subseteq X$ is called a *Borel measure.*

The *carrier* of a Borel measure is given by the set $X \backslash \{O : O \in \mathcal{O} \text{ and } \mu(O) = 0\}$.

B.2 A Brief Summary of Integral Transforms

We summarize formulas for various integral transforms used throughout the text. The Fourier transform conventions we adhere to are laid out in

Definition B.4. The *Fourier transform* of $f \in L_1(\mathbb{R}^s)$ is given by

$$\hat{f}(\boldsymbol{\omega}) = \frac{1}{\sqrt{(2\pi)^s}} \int_{\mathbb{R}^s} f(\boldsymbol{x}) e^{-i\boldsymbol{\omega}\cdot\boldsymbol{x}} d\boldsymbol{x}, \qquad \boldsymbol{\omega} \in \mathbb{R}^s, \tag{B.1}$$

and its *inverse Fourier transform* is given by

$$\check{f}(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^s}} \int_{\mathbb{R}^s} f(\boldsymbol{\omega}) e^{i\boldsymbol{x}\cdot\boldsymbol{\omega}} d\boldsymbol{\omega}, \qquad \boldsymbol{x} \in \mathbb{R}^s.$$

This definition of the Fourier transform can be found in [Rudin (1973)]. Another, just as common, definition uses

$$\hat{f}(\boldsymbol{\omega}) = \int_{\mathbb{R}^s} f(\boldsymbol{x}) e^{-2\pi i\boldsymbol{\omega}\cdot\boldsymbol{x}} d\boldsymbol{x}, \tag{B.2}$$

and can be found in [Stein and Weiss (1971)]. The form (B.1) we use can also be found in the books [Wendland (2005a); Schölkopf and Smola (2002)], whereas (B.2) is used in the books [Buhmann (2003); Cheney and Light (1999)].

Similarly, we can define the Fourier transform of a finite (signed) measure μ on $\mathbb{R}^s$ by

$$\hat{\mu}(\boldsymbol{\omega}) = \frac{1}{\sqrt{(2\pi)^s}} \int_{\mathbb{R}^s} e^{-i\boldsymbol{\omega}\cdot\boldsymbol{x}} d\mu(\boldsymbol{x}), \qquad \boldsymbol{\omega} \in \mathbb{R}^s.$$

Since we are mostly interested in positive definite radial functions, we note that the Fourier transform of a radial function is again radial. Indeed,

Theorem B.1. *Let $\Phi \in L_1(\mathbb{R}^s)$ be continuous and radial, i.e., $\Phi(\boldsymbol{x}) = \varphi(\|\boldsymbol{x}\|)$. Then its Fourier transform $\hat{\Phi}$ is also radial, i.e., $\hat{\Phi}(\boldsymbol{\omega}) = \mathcal{F}_s\varphi(\|\boldsymbol{\omega}\|)$ with*

$$\mathcal{F}_s\varphi(r) = \frac{1}{\sqrt{r^{s-2}}} \int_0^\infty \varphi(t) t^{\frac{s}{2}} J_{(s-2)/2}(rt) dt,$$

where $J_{(s-2)/2}$ is the classical Bessel function of the first kind of order $(s-2)/2$.

The proof of this theorem can be found in [Wendland (2005a)]. The integral transform appearing in Theorem B.1 is also referred to as a *Fourier-Bessel transform* or *Hankel transform.*

The *Hankel inversion theorem* [Sneddon (1972)] ensures that the Fourier transform for radial functions is its own inverse, *i.e.*, for radial functions φ we have

$$\mathcal{F}_s\left[\mathcal{F}_s\varphi\right] = \varphi.$$

A third integral transform that plays an important role is the *Laplace transform.* We have

Definition B.5. Let f be a piecewise continuous function that satisfies $|f(t)| \leq Me^{at}$ for some constants a and M. The *Laplace transform* of f is given by

$$\mathcal{L}f(s) = \int_0^\infty f(t) e^{-st} dt, \qquad s > a.$$

Similarly, the Laplace transform of a Borel measure μ on $[0,\infty)$ is given by

$$\mathcal{L}\mu(s) = \int_0^\infty e^{-st} d\mu(t).$$

The Laplace transform is continuous at the origin if and only if μ is finite.

B.3 The Schwartz Space and the Generalized Fourier Transform

Generalized Fourier transforms are required in the treatment of conditionally positive definite functions. For the definition of the generalized Fourier transform given below we have to define the *Schwartz space* of rapidly decreasing test functions

$$\mathcal{S} = \{\gamma \in C^\infty(\mathbb{R}^s) : \lim_{\|\boldsymbol{x}\|\to\infty} \boldsymbol{x}^{\boldsymbol{\alpha}}(D^{\boldsymbol{\beta}}\gamma)(\boldsymbol{x}) = 0,\ \boldsymbol{\alpha},\boldsymbol{\beta} \in \mathbb{N}_0^s\},$$

where we use the multi-index notation

$$D^{\boldsymbol{\beta}} = \frac{\partial^{|\boldsymbol{\beta}|}}{\partial x_1^{\beta_1} \cdots \partial x_s^{\beta_s}}, \qquad |\boldsymbol{\beta}| = \sum_{i=1}^s \beta_i.$$

The space $\mathcal{S}$ consists of all those functions $\gamma \in C^\infty(\mathbb{R}^s)$ which, together with all their derivatives, decay faster than any power of $1/\|\boldsymbol{x}\|$. The space $\mathcal{S}$ contains the space $C_0^\infty(\mathbb{R}^s)$, the space of all infinitely differentiable functions on $\mathbb{R}^s$ with compact support. We also note that $C_0^\infty(\mathbb{R}^s)$ is a true subspace of $\mathcal{S}$ since, *e.g.*, the function $\gamma(\boldsymbol{x}) = e^{-\|\boldsymbol{x}\|^2}$ belongs to $\mathcal{S}$ but not to $C_0^\infty(\mathbb{R}^s)$. A remarkable fact about the Schwartz space is that $\gamma \in \mathcal{S}$ has a classical Fourier transform $\hat{\gamma}$ which is also in $\mathcal{S}$.

Of particular importance are the following subspaces $\mathcal{S}_m$ of $\mathcal{S}$

$$\mathcal{S}_m = \{\gamma \in \mathcal{S} :\ \gamma(\boldsymbol{x}) = \mathcal{O}(\|\boldsymbol{x}\|^m) \text{ for } \|\boldsymbol{x}\| \to 0,\ m \in \mathbb{N}_0\}.$$

Furthermore, the set $\mathcal{V}$ of slowly increasing functions is given by

$$\mathcal{V} = \{f \in C(\mathbb{R}^s) :\ |f(\boldsymbol{x})| \le |p(\boldsymbol{x})| \text{ for some polynomial } p \in \Pi^s\}.$$

The generalized Fourier transform is now given by

Definition B.6. Let $f \in \mathcal{V}$ be complex-valued. A continuous function $\hat{f} : \mathbb{R}^s \setminus \{0\} \to \mathbb{C}$ is called the *generalized Fourier transform* of f if there exists an integer $m \in \mathbb{N}_0$ such that

$$\int_{\mathbb{R}^s} f(\boldsymbol{x})\hat{\gamma}(\boldsymbol{x})d\boldsymbol{x} = \int_{\mathbb{R}^s} \hat{f}(\boldsymbol{x})\gamma(\boldsymbol{x})d\boldsymbol{x}$$

is satisfied for all $\gamma \in \mathcal{S}_{2m}$. The smallest such integer m is called the *order* of $\hat{f}$.

Various definitions of the generalized Fourier transform exist in the literature. A classical reference is the book [Gel'fand and Vilenkin (1964)].

Since one can show that the generalized Fourier transform of an s-variate polynomial of degree at most $2m$ is zero, it follows that the inverse generalized Fourier

transform is only unique up to addition of such a polynomial. The order of the generalized Fourier transform is nothing but the order of the singularity at the origin of the generalized Fourier transform. For functions in $L_1(\mathbb{R}^s)$ the generalized Fourier transform coincides with the classical Fourier transform, and for functions in $L_2(\mathbb{R}^s)$ it coincides with the distributional Fourier transform.

Appendix C

Additional Computer Programs

In this appendix we list several MATLAB and one Maple program that are used in various places throughout the book.

C.1 MATLAB Programs

As a test function for multi-dimensional problems we sometimes use

$$f_s(\boldsymbol{x}) = 4^s \prod_{d=1}^{s} x_d(1 - x_d), \qquad \boldsymbol{x} = (x_1, \ldots, x_s) \in [0,1]^s.$$

Program C.1. `testfunction.m`

```
% tf = testfunction(s,points)
% Evaluates testfunction
% prod_{d=1}^s x_d*(1-x_d)  (normalized so that its max is 1)
% at s-dimensional points
function tf = testfunction(s,points)
tf = 4^s*prod(points.*(1-points),2);
```

Another test function used in some of the numerical experiments is the sine function defined for any $\boldsymbol{x} = (x_1, \ldots, x_s) \in \mathbb{R}^s$ as

$$\text{sine}\,(\boldsymbol{x}) = \prod_{d=1}^{s} \frac{\sin(\pi x_d)}{\pi x_d}.$$

The sine function is not a standard MATLAB function. It can, however, be found in the Signal Processing Toolbox. For the sake of completeness we provide MATLAB code for the sine function of a single variable, $x \in \mathbb{R}$.

Program C.2. `sinc.m`

```
% f = sinc(x)
% Defines sinc function
function f = sinc(x)
```

```
f = ones(size(x));
nz = find(x~=0);
f(nz) = sin(pi*x(nz))./(pi*x(nz));
```

Note that while `sinc.m` takes a vector input `x` it produces a vector of values of the univariate sinc function at the components of `x` — not the value of the multivariate sinc function at the vector argument `x`.

A multi-dimensional grid of equally spaced points is used several times throughout the book. MATLAB provides the command `ndgrid` that can accomplish this. However, in order to be able to use this command flexibly for all space dimensions s we require a little extra work. This is implemented `MakeSDGrid.m`.

Program C.3. `MakeSDGrid.m`

```
% gridpoints = MakeSDGrid(s,neval)
% Produces matrix of equally spaced points in s-dimensional unit cube
% (one point per row)
% Input
%   s:     space dimension
%   neval: number of points in each coordinate direction
% Output
%   gridpoints: neval^s-by-s matrix (one point per row,
%               d-th column contains d-th coordinate of point)
function gridpoints = MakeSDGrid(s,neval)
if (s==1)
    gridpoints = linspace(0,1,neval)';
    return;
end
% Mimic this statement for general s:
% [x1, x2] = ndgrid(linspace(0,1,neval));
outputarg = 'x1';
for d = 2:s
    outputarg = strcat(outputarg,',x',int2str(d));
end
makegrid = strcat('[',outputarg,'] = ndgrid(linspace(0,1,neval));');
eval(makegrid);
% Mimic this statement for general s:
% gridpoints = [x1(:) x2(:)];
gridpoints = zeros(neval^s,s);
for d = 1:s
    matrices = strcat('gridpoints(:,d) = x',int2str(d),'(:);');
    eval(matrices);
end
```

Due to its removable singularity at the origin the thin-plate spline basic function requires a separate function definition.

Program C.4. `tps.m`

```
% rbf = tps(e,r)
% Defines thin plate spline RBF
function rbf = tps(e,r)
rbf = zeros(size(r));
nz = find(r~=0);   % to deal with singularity at origin
rbf(nz) = (e*r(nz)).^2.*log(e*r(nz));
```

Standard plotting routines for 2D function and error graphs are used by most programs.

Program C.5. `PlotSurf.m`

```
% PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview)
% Generates plot of surface Pf false colored by the
% maximum error abs(Pf-exact)
% fview defines the view.
function PlotSurf(xe,ye,Pf,neval,exact,maxerr,fview)
    % Plot surface
figure
Pfplot = surf(xe,ye,reshape(Pf,neval,neval),...
              reshape(abs(Pf-exact),neval,neval));
set(Pfplot,'FaceColor','interp','EdgeColor','none')
[cmin cmax] = caxis;
caxis([cmin-.25*maxerr cmax]);
view(fview);
colormap hsv
vcb = colorbar('vert');
ylim(vcb,[0 maxerr])
set(get(vcb,'YLabel'),'String','Error')
```

Program C.6. `PlotError2D.m`

```
% PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview)
% Generates plot of abs error for surface Pf, i.e., abs(Pf-exact)
% fview defines the view.
function PlotError2D(xe,ye,Pf,exact,maxerr,neval,fview)
    % Plot maximum error
figure
errorplot = surf(xe,ye,reshape(abs(Pf-exact),neval,neval));
set(errorplot,'FaceColor','interp','EdgeColor','none')
```

```
[cmin cmax] = caxis;
caxis([cmin-.25*maxerr cmax])
view(fview);
colormap hsv
vcb = colorbar('vert');
ylim(vcb,[0 maxerr])
set(get(vcb,'YLabel'),'String','Error')
```

For 3D plots we use the following routines.

Program C.7. `Plotlsosurf.m`

```
% Plotlsosurf(xe,ye,ze,Pf,neval,exact,maxerr,isomin,
%              isostep,isomax)
% Generates plot of isosurfaces of Pf false colored by
% the error abs(Pf-exact)
% isomin,isostep,isomax define the range and number of
% isosurfaces.
function Plotlsosurf(xe,ye,ze,Pf,neval,exact,maxerr,...
         isomin,isostep,isomax)
% Plot isosurfaces
figure
hold on
for isovalue=isomin:isostep:isomax
    pfit = patch(isosurface(xe,ye,ze,reshape(Pf,neval,...
          neval,neval),isovalue,reshape(abs(Pf-exact),...
          neval,neval,neval)));
    isonormals(xe,ye,ze,reshape(Pf,neval,neval,neval),pfit)
    set(pfit,'FaceColor','interp','EdgeColor','none');
    daspect([1 1 1])
    view(3); axis([0 1 0 1 0 1])
end
[cmin cmax] = caxis;
caxis([cmin-.25*cmax cmax])
colormap hsv
vcb = colorbar('vert');
ylim(vcb,[0 cmax])
set(get(vcb,'YLabel'),'String','Error')
hold off
```

Program C.8. `PlotSlices.m`

```
% PlotSlices(xe,ye,ze,Pf,neval,xslice,yslice,zslice)
% Generates slice plot of volume Pf
% xslice,yslice,zslice define the range and number of slices.
```

```
function PlotSlices(xe,ye,ze,Pf,neval,xslice,yslice,zslice)
% Plot slices
figure
pfit = slice(xe,ye,ze,reshape(Pf,neval,neval,neval),...
             xslice,yslice,zslice);
set(pfit,'FaceColor','interp','EdgeColor','none')
daspect([1 1 1])
view(3); axis([0 1 0 1 0 1])
vcb = colorbar('vert');
set(get(vcb,'YLabel'),'String','Function value')
```

Program C.9. PlotErrorSlices.m

```
% PlotErrorSlices(xe,ye,ze,Pf,exact,ne,xslice,yslice,zslice)
% Generates slice plot of volume error abs(Pf-exact)
% xslice,yslice,zslice define the range and number of slices.
function PlotErrorSlices(xe,ye,ze,Pf,exact,ne,...
                         xslice,yslice,zslice)
% Plot slices for error
figure
errorplot = slice(xe,ye,ze,reshape(abs(Pf-exact),ne,ne,ne),...
                  xslice,yslice,zslice);
set(errorplot,'FaceColor','interp','EdgeColor','none')
daspect([1 1 1])
view(3); axis([0 1 0 1 0 1])
[cmin cmax] = caxis;
caxis([cmin-.25*cmax cmax])
colormap hsv
vcb = colorbar('vert');
ylim(vcb,[0 cmax])
set(get(vcb,'YLabel'),'String','Error')
```

The following algorithm is a very primitive (and very inefficient) implementation of an adaptive thinning algorithm for scattered data. It removes 500 points at a time and writes the intermediate result to a file.

Program C.10. Thin.m

```
load('Data2D_Beethoven')
% This loads variables dsites and rhs
x = dsites(:,1);
y = dsites(:,2);
figure
tes = delaunayn(dsites);
triplot(tes,x,y,'g')
```

```
for 1=1:5
   for j=1:500
      n = size(dsites,1);
      d = zeros(1,n);
      for i=1:n
         temp = dsites;
         temp(i,:) = [];
         [k,d(i)] = dsearchn(temp,dsites(i,:));
         if (k >= i)
            k=k+1;
         end
      end
      r = min(d);
      idx = find(d==r);
      dsites(idx(1),:) = [];
      x(idx(1)) = [];
      y(idx(1)) = [];
      rhs(idx(1)) = [];
   end
   figure
   tes = delaunayn(dsites);
   triplot(tes,x,y,'r')
   name = sprintf('Data2D_Beethoven%d', 1);
   save(name, 'dsites', 'rhs')
end
```

C.2 Maple Programs

The MLS basis functions and dual basis functions displayed in Chapter 24 were computed with the following Maple code.

Program C.11. MLSDualBases.mws

```
restart; with(plots): with(linalg):
N:=10: m:=3: DD:=4: h:=1/N: ep:=1/(sqrt(DD)*h):
phi := (x,y) -> exp(-ep^2*(x-y)^2);
for k from 1 to m do
   pp||k := plot(x^(k-1), x=0..1):
od:
display([seq(pp||k,k=1..m)],insequence=true,thickness=2);
X := vector([seq(h*k, k=0..N)]);
# or use 11 Halton points
# X := vector([0.5000,0.2500,0.7500,0.1250,0.6250,
```

```
#      0.3750,0.8750,0.0625,0.5625,0.3125,0.8125]);
G := matrix(m,m):
for i from 1 to m do
   for j from 1 to m do
      G[i,j] := evalf(add((X[k])^(i-1)*(X[k])^(j-1)*
                    phi(x,X[k]), k=1..N+1));
   od:
od:
P := vector([evalf(seq(y^(k-1), k=1..m))]);
Lambda := linsolve(G,P):
for k from 1 to m do
   l||k := unapply(Lambda[k],(x,y));
od:
for k from 1 to m do
   lp||k := plot(l||k(x,x), x=0..1):
od:
display([seq(lp||k, k=1..m)],insequence=true,thickness=2);
K := (x,y) -> phi(x,y)*add(l||k(x,x)*y^(k-1), k=1..m):
approxK := (x,y) -> 1/sqrt(DD*Pi)*(3/2-ep^2*(x-y)^2)
                    *phi(x,y);
for i from 1 to N+1 do
   aKp||i := plot([K(x,X[i]),approxK(x,X[i])], x=0..1,
                 color=[green,red]):
od:
display(seq(aKp||i,i=1..N+1),insequence=true,thickness=2);
```

Appendix D

Catalog of RBFs with Derivatives

D.1 Generic Derivatives

We provide formulas for all first and second-order derivatives of radial functions of two variables, *i.e.*, $\varphi(r) = \varphi(\|\boldsymbol{x}\|) = \varphi(\sqrt{x^2+y^2})$, where $\boldsymbol{x} = (x, y) \in \mathbb{R}^2$. The chain rule implies

$$\begin{aligned}\frac{\partial}{\partial x}\varphi(\|\boldsymbol{x}\|) &= \frac{d}{dr}\varphi(r)\frac{\partial}{\partial x}r(x,y)\\ &= \frac{d}{dr}\varphi(r)\frac{x}{\sqrt{x^2+y^2}}\\ &= \frac{x}{r}\frac{d}{dr}\varphi(r)\end{aligned}$$

since $r = \|\boldsymbol{x}\| = \sqrt{x^2+y^2}$. Similarly, $\frac{\partial}{\partial y}\varphi(\|\boldsymbol{x}\|) = \frac{y}{r}\frac{d}{dr}\varphi(r)$. The generic second-order derivatives are given by

$$\begin{aligned}\frac{\partial^2}{\partial x^2}\varphi(\|\boldsymbol{x}\|) &= \frac{d^2}{dr^2}\varphi(r)\left(\frac{\partial}{\partial x}r(x,y)\right)^2 + \frac{d}{dr}\varphi(r)\frac{\partial^2}{\partial x^2}r(x,y)\\ &= \frac{x^2}{r^2}\frac{d^2}{dr^2}\varphi(r) + \frac{y^2}{r^3}\frac{d}{dr}\varphi(r),\end{aligned}$$

as well as

$$\begin{aligned}\frac{\partial^2}{\partial y^2}\varphi(\|\boldsymbol{x}\|) &= \frac{y^2}{r^2}\frac{d^2}{dr^2}\varphi(r) + \frac{x^2}{r^3}\frac{d}{dr}\varphi(r),\\ \frac{\partial^2}{\partial x\partial y}\varphi(\|\boldsymbol{x}\|) &= \frac{xy}{r^2}\frac{d^2}{dr^2}\varphi(r) - \frac{xy}{r^3}\frac{d}{dr}\varphi(r),\end{aligned}$$

and the Laplacian

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\varphi(\|\boldsymbol{x}\|) = \frac{d^2}{dr^2}\varphi(r) + \frac{1}{r}\frac{d}{dr}\varphi(r).$$

Derivatives of higher order or in higher space dimensions can be computed similarly by applying the chain rule. For example, the generic fourth-order biharmonic (or double Laplacian) turns out to be

$$\left(\frac{\partial^4}{\partial x^4} + 2\frac{\partial^4}{\partial x^2 y^2} + \frac{\partial^4}{\partial y^4}\right)\varphi(\|\boldsymbol{x}\|) = \frac{d^4}{dr^4}\varphi(r) + \frac{2}{r}\frac{d^3}{dr^3}\varphi(r) - \frac{1}{r^2}\frac{d^2}{dr^2}\varphi(r) + \frac{1}{r^3}\frac{d}{dr}\varphi(r).$$

D.2 Formulas for Specific Basic Functions

The generic derivatives of the basic function with respect to r in the previous section need to be replaced by the following formulas.

D.2.1 *Globally Supported, Strictly Positive Definite Functions*

Example D.1. Gaussian RBF:

$$\begin{aligned}\varphi(r) &= e^{-(\varepsilon r)^2},\\ \frac{d}{dr}\varphi(r) &= -2\varepsilon^2 r e^{-(\varepsilon r)^2},\\ \frac{d^2}{dr^2}\varphi(r) &= 2\varepsilon^2 e^{-(\varepsilon r)^2}\left(2(\varepsilon r)^2-1\right).\end{aligned}$$

This function is C^∞ at the origin.

Example D.2. Inverse multiquadric (IMQ) RBF:

$$\begin{aligned}\varphi(r) &= \frac{1}{\sqrt{1+(\varepsilon r)^2}},\\ \frac{d}{dr}\varphi(r) &= -\frac{\varepsilon^2 r}{(1+(\varepsilon r)^2)^{3/2}},\\ \frac{d^2}{dr^2}\varphi(r) &= \varepsilon^2\frac{2(\varepsilon r)^2-1}{(1+(\varepsilon r)^2)^{5/2}}.\end{aligned}$$

This function is C^∞ at the origin.

Example D.3. Generalized IMQ RBF:

$$\begin{aligned}\varphi(r) &= \frac{1}{(1+(\varepsilon r)^2)^2},\\ \frac{d}{dr}\varphi(r) &= -\frac{4\varepsilon^2 r}{(1+(\varepsilon r)^2)^3},\\ \frac{d^2}{dr^2}\varphi(r) &= 4\varepsilon^2\frac{5(\varepsilon r)^2-1}{(1+(\varepsilon r)^2)^4}.\end{aligned}$$

This function is C^∞ at the origin.

Example D.4. Inverse quadratic (IQ) RBF:

$$\begin{aligned}\varphi(r) &= \frac{1}{(1+(\varepsilon r)^2)},\\ \frac{d}{dr}\varphi(r) &= -\frac{2\varepsilon^2 r}{(1+(\varepsilon r)^2)^2},\\ \frac{d^2}{dr^2}\varphi(r) &= 2\varepsilon^2\frac{3(\varepsilon r)^2-1}{(1+(\varepsilon r)^2)^3}.\end{aligned}$$

This function is C^∞ at the origin.

Example D.5. Basic Matérn RBF:

$$\varphi(r) = e^{-\varepsilon r}.$$

This function is not differentiable at the origin.

Example D.6. Linear Matérn RBF:

$$\begin{aligned}\varphi(r) &= e^{-\varepsilon r}(1+\varepsilon r),\\ \frac{d}{dr}\varphi(r) &= -\varepsilon^2 r e^{-\varepsilon r},\\ \frac{d^2}{dr^2}\varphi(r) &= \varepsilon^2 e^{-\varepsilon r}(\varepsilon r - 1).\end{aligned}$$

This function is C^2 at the origin, but not smoother.

Example D.7. Quadratic Matérn RBF:

$$\begin{aligned}\varphi(r) &= e^{-\varepsilon r}(3+3\varepsilon r + (\varepsilon r)^2),\\ \frac{d}{dr}\varphi(r) &= -\varepsilon^2 r e^{-\varepsilon r}(1+\varepsilon r),\\ \frac{d^2}{dr^2}\varphi(r) &= \varepsilon^2 e^{-\varepsilon r}\left((\varepsilon r)^2 - \varepsilon r - 1\right).\end{aligned}$$

This function is C^4 at the origin.

Example D.8. Cubic Matérn RBF:

$$\begin{aligned}\varphi(r) &= e^{-\varepsilon r}(15+15\varepsilon r + 6(\varepsilon r)^2 + (\varepsilon r)^3),\\ \frac{d}{dr}\varphi(r) &= -\varepsilon^2 r e^{-\varepsilon r}\left((\varepsilon r)^2 + 3\varepsilon r + 3\right),\\ \frac{d^2}{dr^2}\varphi(r) &= \varepsilon^2 e^{-\varepsilon r}\left((\varepsilon r)^3 - 3\varepsilon r - 3\right).\end{aligned}$$

This function is C^6 at the origin.

D.2.2 *Globally Supported, Strictly Conditionally Positive Definite Functions of Order 1*

Example D.9. Linear or norm RBF:

$$\varphi(r) = r.$$

This function is not differentiable at the origin.

Example D.10. Multiquadric (MQ) RBF:

$$\begin{aligned}\varphi(r) &= \sqrt{1+(\varepsilon r)^2},\\ \frac{d}{dr}\varphi(r) &= \frac{\varepsilon^2 r}{\sqrt{1+(\varepsilon r)^2}},\\ \frac{d^2}{dr^2}\varphi(r) &= \frac{\varepsilon^2}{(1+(\varepsilon r)^2)^{3/2}}.\end{aligned}$$

This function is C^∞ at the origin.

D.2.3 *Globally Supported, Strictly Conditionally Positive Definite Functions of Order 2*

Example D.11. Generalized MQ RBF:

$$\begin{aligned}\varphi(r) &= (1+(\varepsilon r)^2)^{3/2},\\ \frac{d}{dr}\varphi(r) &= 3\varepsilon^2 r\sqrt{1+(\varepsilon r)^2},\\ \frac{d^2}{dr^2}\varphi(r) &= 3\varepsilon^2\frac{2(\varepsilon r)^2+1}{\sqrt{1+(\varepsilon r)^2}}.\end{aligned}$$

This function is C^∞ at the origin.

Example D.12. Cubic RBF:

$$\begin{aligned}\varphi(r) &= r^3,\\ \frac{d}{dr}\varphi(r) &= 3r^2,\\ \frac{d^2}{dr^2}\varphi(r) &= 6r.\end{aligned}$$

Example D.13. Thin plate spline (TPS) RBF:

$$\begin{aligned}\varphi(r) &= r^2\log(r),\\ \frac{d}{dr}\varphi(r) &= r\left(2\log(r)+1\right),\\ \frac{d^2}{dr^2}\varphi(r) &= 2\log(r)+3.\end{aligned}$$

While the singularities of the function and first derivative at the origin are removable, the singularity of the second derivative at the origin is not.

D.2.4 *Globally Supported, Strictly Conditionally Positive Definite Functions of Order 3*

Example D.14. Generalized MQ RBF:

$$\begin{aligned}\varphi(r) &= (1+(\varepsilon r)^2)^{5/2},\\ \frac{d}{dr}\varphi(r) &= 5\varepsilon^2 r\left(1+(\varepsilon r)^2\right)^{3/2},\\ \frac{d^2}{dr^2}\varphi(r) &= 5\varepsilon^2\sqrt{1+(\varepsilon r)^2}\left(4(\varepsilon r)^2+1\right).\end{aligned}$$

This function is C^∞ at the origin.

Example D.15. Quintic RBF:

$$\begin{aligned}\varphi(r) &= r^5,\\ \frac{d}{dr}\varphi(r) &= 5r^4,\\ \frac{d^2}{dr^2}\varphi(r) &= 20r^3.\end{aligned}$$

Example D.16. Second-order TPS RBF:

$$\begin{aligned}\varphi(r) &= r^4 \log(r),\\ \frac{d}{dr}\varphi(r) &= r^3\left(4\log(r)+1\right),\\ \frac{d^2}{dr^2}\varphi(r) &= r^2\left(12\log(r)+7\right).\end{aligned}$$

D.2.5 *Globally Supported, Strictly Conditionally Positive Definite Functions of Order 4*

Example D.17. Septic RBF:

$$\begin{aligned}\varphi(r) &= r^7,\\ \frac{d}{dr}\varphi(r) &= 7r^6,\\ \frac{d^2}{dr^2}\varphi(r) &= 42r^5.\end{aligned}$$

D.2.6 *Globally Supported, Strictly Positive Definite and Oscillatory Functions*

Example D.18. Linear Laguerre-Gaussian RBF for $\mathbb{R}^2$:

$$\begin{aligned}\varphi(r) &= e^{-(\varepsilon r)^2}(2-(\varepsilon r)^2),\\ \frac{d}{dr}\varphi(r) &= 2\varepsilon^2 r e^{-(\varepsilon r)^2}\left((\varepsilon r)^2-3\right),\\ \frac{d^2}{dr^2}\varphi(r) &= -2\varepsilon^2 e^{-(\varepsilon r)^2}\left(2(\varepsilon r)^4-9(\varepsilon r)^2+3\right).\end{aligned}$$

This function is C^∞ at the origin.

Example D.19. Quadratic Laguerre-Gaussian RBF for $\mathbb{R}^2$:

$$\begin{aligned}\varphi(r) &= e^{-(\varepsilon r)^2}(3-3(\varepsilon r)^2+\frac{1}{2}(\varepsilon r)^4),\\ \frac{d}{dr}\varphi(r) &= -\varepsilon^2 r e^{-(\varepsilon r)^2}\left((\varepsilon r)^4-8(\varepsilon r)^2+12\right),\\ \frac{d^2}{dr^2}\varphi(r) &= \varepsilon^2 e^{-(\varepsilon r)^2}\left(2(\varepsilon r)^6-21(\varepsilon r)^4+48(\varepsilon r)^2-12\right).\end{aligned}$$

This function is C^∞ at the origin.

Example D.20. Linear generalized IMQ RBF:

$$\begin{aligned}\varphi(r) &= \frac{2-(\varepsilon r)^2}{(1+(\varepsilon r)^2)^4},\\ \frac{d}{dr}\varphi(r) &= 6\varepsilon^2 r\frac{(\varepsilon r)^2-3}{(1+(\varepsilon r)^2)^5},\\ \frac{d^2}{dr^2}\varphi(r) &= -6\varepsilon^2\frac{7(\varepsilon r)^4-30(\varepsilon r)^2+3}{(1+(\varepsilon r)^2)^6}.\end{aligned}$$

This function is C^∞ at the origin.

Example D.21. Quadratic generalized IMQ RBF:

$$\begin{aligned}\varphi(r) &= \frac{3-6(\varepsilon r)^2+(\varepsilon r)^4}{(1+(\varepsilon r)^2)^6},\\ \frac{d}{dr}\varphi(r) &= -8\varepsilon^2 r\frac{(\varepsilon r)^4-8(\varepsilon r)^2+6}{(1+(\varepsilon r)^2)^7},\\ \frac{d^2}{dr^2}\varphi(r) &= 24\varepsilon^2\frac{3(\varepsilon r)^6-31(\varepsilon r)^4+34(\varepsilon r)^2-2}{(1+(\varepsilon r)^2)^8}.\end{aligned}$$

D.2.7 *Compactly Supported, Strictly Positive Definite Functions*

Example D.22. Wendland's $\varphi_{3,0}$ (strictly positive definite in $\mathbb{R}^3$):

$$\varphi(r) = (1-\varepsilon r)_+^2.$$

This function is not differentiable at the origin.

Example D.23. Wendland's $\varphi_{3,1}$ (strictly positive definite in $\mathbb{R}^3$):

$$\begin{aligned}\varphi(r) &= (1-\varepsilon r)_+^4(4\varepsilon r+1),\\ \frac{d}{dr}\varphi(r) &= -20\varepsilon^2 r(1-\varepsilon r)_+^3,\\ \frac{d^2}{dr^2}\varphi(r) &= 20\varepsilon^2(4\varepsilon r-1)(1-\varepsilon r)_+^2.\end{aligned}$$

This function is C^2 at the origin.

Example D.24. Wendland's $\varphi_{3,2}$ (strictly positive definite in $\mathbb{R}^3$):

$$\begin{aligned}\varphi(r) &= (1-\varepsilon r)_+^6(35(\varepsilon r)^2+18\varepsilon r+3),\\ \frac{d}{dr}\varphi(r) &= -56\varepsilon^2 r(5\varepsilon r+1)(1-\varepsilon r)_+^5,\\ \frac{d^2}{dr^2}\varphi(r) &= 56\varepsilon^2\left(35(\varepsilon r)^2-4\varepsilon r-1\right)(1-\varepsilon r)_+^4.\end{aligned}$$

This function is C^4 at the origin.

Example D.25. Wendland's $\varphi_{3,3}$ (strictly positive definite in $\mathbb{R}^3$)

$$\begin{aligned}\varphi(r) &= (1-\varepsilon r)_+^8(32(\varepsilon r)^3+25(\varepsilon r)^2+8\varepsilon r+1),\\ \frac{d}{dr}\varphi(r) &= -22\varepsilon^2 r\left(16(\varepsilon r)^2+7\varepsilon r+1\right)(1-\varepsilon r)_+^7,\\ \frac{d^2}{dr^2}\varphi(r) &= 22\varepsilon^2\left(160(\varepsilon r)^3+15(\varepsilon r)^2-6\varepsilon r-1\right)(1-\varepsilon r)_+^6.\end{aligned}$$

This function is C^6 at the origin.

Example D.26. Wu's $\psi_{3,3}$ (strictly positive definite in $\mathbb{R}^7$):

$$\varphi(r) = (1-\varepsilon r)_+^4(5(\varepsilon r)^3 + 20(\varepsilon r)^2 + 29\varepsilon r + 16).$$

This function is not differentiable at the origin.

Example D.27. Wu's $\psi_{2,3}$ (strictly positive definite in $\mathbb{R}^5$):

$$\begin{aligned}
\varphi(r) &= (1-\varepsilon r)_+^5(5(\varepsilon r)^4 + 25(\varepsilon r)^3 + 48(\varepsilon r)^2 + 40\varepsilon r + 8),\\
\frac{d}{dr}\varphi(r) &= -9\varepsilon^2 r\left(5(\varepsilon r)^3 + 20(\varepsilon r)^2 + 29\varepsilon r + 16\right)(1-\varepsilon r)_+^4,\\
\frac{d^2}{dr^2}\varphi(r) &= 18\varepsilon^2\left(20(\varepsilon r)^4 + 60(\varepsilon r)^3 + 57(\varepsilon r)^2 + 11\varepsilon r - 8\right)(1-\varepsilon r)_+^3.
\end{aligned}$$

This function is C^2 at the origin.

Example D.28. Wu's $\psi_{1,3}$ (strictly positive definite in $\mathbb{R}^3$):

$$\begin{aligned}
\varphi(r) &= (1-\varepsilon r)_+^5(5(\varepsilon r)^5 + 30(\varepsilon r)^4 + 72(\varepsilon r)^3 + 82(\varepsilon r)^2 + 36\varepsilon r + 6),\\
\frac{d}{dr}\varphi(r) &= -11\varepsilon^2 r(\varepsilon r + 2)\left(5(\varepsilon r)^3 + 15(\varepsilon r)^2 + 18\varepsilon r + 4\right)(1-\varepsilon r)_+^5,\\
\frac{d^2}{dr^2}\varphi(r) &= 22\varepsilon^2\left(25(\varepsilon r)^5 + 100(\varepsilon r)^4 + 142(\varepsilon r)^3 + 68(\varepsilon r)^2 - 16\varepsilon r - 4\right)(1-\varepsilon r)_+^4.
\end{aligned}$$

This function is C^4 at the origin.

Example D.29. Wu's $\psi_{0,3}$ (strictly positive definite in $\mathbb{R}$):

$$\begin{aligned}
\varphi(r) &= (1-\varepsilon r)_+^7(5(\varepsilon r)^6 + 35(\varepsilon r)^5 + 101(\varepsilon r)^4 + 147(\varepsilon r)^3 + 101(\varepsilon r)^2 + 35\varepsilon r + 5),\\
\frac{d}{dr}\varphi(r) &= -13\varepsilon^2 r\left(5(\varepsilon r)^5 + 30(\varepsilon r)^4 + 72(\varepsilon r)^3 + 82(\varepsilon r)^2 + 36\varepsilon r + 6\right)(1-\varepsilon r)_+^6,\\
\frac{d^2}{dr^2}\varphi(r) &= 78\varepsilon^2\left(10(\varepsilon r)^6 + 50(\varepsilon r)^5 + 95(\varepsilon r)^4 + 75(\varepsilon r)^3 + 7(\varepsilon r)^2 - 5\varepsilon r - 1\right)(1-\varepsilon r)_+^5.
\end{aligned}$$

This function is C^6 at the origin, but only strictly positive definite in $\mathbb{R}$.

Example D.30. Euclid's hat φ_1:

$$\varphi(r) = (1-\varepsilon r/2)_+.$$

None of the Euclid's hat functions are differentiable at the origin.

Bibliography

Abramowitz, M. and Stegun, I. A. (1972). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Dover (New York).

Acosta, F. M. (1995). Radial basis functions and related models, *Signal Proc.* **45**, pp. 37–58.

Adams, R. (1975). *Sobolev Spaces*, Academic Press (New York).

Alfeld, P. (1989). Scattered data interpolation in three or more variables, in *Mathematical Methods in Computer Aided Geometric Design*, T. Lyche and L. Schumaker (eds.), Academic Press (New York), pp. 1–33.

Allasia, G. and Giolito, P. (1997). Fast evaluation of cardinal radial basis interpolants, in *Surface Fitting and Multiresolution Methods*, A. Le Méhauté, C. Rabut, and L. L. Schumaker (eds.), Vanderbilt University Press (Nashville, TN), pp. 1–8.

Allison, J. (1993). Multiquadratic radial basis functions for representing multidimensional high energy physics data, *Comp. Phys. Comm.* 77, pp. 377–395.

Alves, C. J. S. and Silvestre, A. L. (2004). Density results using Stokeslets and a method of fundamental solutions for the Stokes equations, *Engineering Analysis with Boundary Elements* **28**, pp. 1245–1252.

Andrews, G. E., Askey, R. and Roy, R. (1999). *Special Functions*, Cambridge University Press (Cambridge).

Arad, N., Dyn, N., Reisfeld, D. and Yeshurun, Y. (1994). Image warping by radial basis functions: applications to facial expressions, *CVGIP: Graphical models and image processing* **56**, pp. 161–172.

Armentano, M. G. (2001). Error estimates in Sobolev spaces for moving least square approximations, *SIAM J. Numer. Anal.* **39** 1, pp. 38–51.

Arnott, R. (1993). An adaptive radial basis function diversity combiner for multipath channels, *IEE Electronics Letters* **29**, pp. 1092–1094.

Aronszajn, N. (1950). Theory of reproducing kernels, *Trans. Amer. Math. Soc.* **68**, pp. 337–404.

Askey, R. (1973). Radial characteristic functions, TSR #1262, University of Wisconsin-Madison.

Askey, R. (1975). *Orthogonal Polynomials and Special Functions*, Reg. Conf. Ser. in Appl. Math. (SIAM).

Atluri, S. N. (2004). *The Meshless Methods (MLPG) For Domain & BIE Discretizations*, Tech Science Press (Encino, CA).

Atluri, S. N. and Shen, S. (2002a). *The Meshless Local Petrov-Galerkin (MLPG) Method*, Tech Science Press (Encino, CA).

Atluri, S. N. and Shen, S. (2002b). The meshless local Petrov-Galerkin (MLPG) method: a

simple & less costly alternative to the finite element and boundary element methods, *Comput. Model. Eng. Sci.* **3**, pp. 11–51.

Atluri, S. N. and Zhu, T. (1998). A new meshless local Petrov-Galerkin (MLPG) approach in computational mechanics, *Comput. Mech.* 22, pp. 117–127.

Babuška, I., Banerjee U. and Osborn, J. E. (2003). Survey of meshless and generalized finite element methods: A unified approach, *Acta Numerica* 12, pp. 1–125.

Babuška, I. and Melenk, M. (1997). The partition of unity method, *Int. J. Numer. Meths. Eng.* **40**, pp. 727–758.

Backus, G. and Gilbert, F. (1968). The resolving power of gross earth data, *Geophys. J. R. Astr. Soc.* **16**, pp 169–205.

Ball, K. (1992). Eigenvalues of Euclidean distance matrices, *J. Approx. Theory* **68**, pp. 74–82.

Ball, K., Sivakumar, N. and Ward, J. D. (1992). On the sensitivity of radial basis interpolation to minimal data separation distance, *Constr. Approx.* **8**, pp. 401–426.

Barnes, R. J. (1994). A modified conjugate gradient algorithm for scattered data interpolation using radial basis functions, *J. Appl. Sci. Comput.* 1, pp. 227–238.

Barnhill, R. E. and Ou, H. S. (1990). Surfaces defined on surfaces, *Comput. Aided Geom. Design* **7**, pp. 323–336.

Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function, *IEEE Trans. Inform. Theory* **39**, pp. 930–945.

Bates, D., Lindstrom, M., Wahba, G. and Yandell, B. (1987). GCVPACK — routines for generalized cross validation, *Commun. Statist. B — Simulation and Computation* **16**, pp. 263–297.

Bates, D. and Wahba, G. (1982). Computational methods for generalized cross validation with large data sets, in *Treatment of Integral Equations by Numerical Methods*, C. T. H. Baker, and G. F. Miller (eds.), Academic Press (New York), pp. 283–296.

Baule, R. (2000). Moving Least Squares Approximation mit parameterabhängigen Gewichtsfunktionen, Diplomarbeit, Universität Göttingen.

Baxter, B. J. C. (1991). Conditionally positive functions and p-norm distance matrices, *Constr. Approx.* **7**, pp. 427–440.

Baxter, B. J. C. (1992a). Norm estimates for inverses of distance matrices, in *Mathematical Methods in Computer Aided Geometric Design II*, T. Lyche and L. Schumaker (eds.), Academic Press (New York), pp. 9–18.

Baxter, B. J. C. (1992b). The asymptotic cardinal function of the multiquadric $\phi(r) = (r^2 + c^2)^{1/2}$ as $c \to \infty$, *Comput. Math. Appl.* 24, pp. 1–6.

Baxter, B. J. C. (1992c). The interpolation theory of radial basis functions, Ph.D. Dissertation, University of Cambridge.

Baxter, B. J. C. (2002). Preconditioned conjugate gradients, radial basis functions, and Toeplitz matrices, *Comput. Math. Appl.* **43**, pp. 305–318.

Baxter, B. J. C. (2006). Scaling radial basis functions via Euclidean distance matrices, *Comput. Math. Appl.* **51** 8, pp. 1163–1170.

Baxter, B. J. C. and Hubbert, S. (2001). Radial basis functions for the sphere, in *Recent progress in multivariate approximation*, Internat. Ser. Numer. Math., 137, Birkhäuser (Basel), pp. 33–47.

Baxter, B. J. C. and Roussos, G. (2002). A new error estimate of the fast Gauss transform, *SIAM J. Sci. Comput.* 24 1, pp. 257–259.

Baxter, B. J. C. and Sivakumar, N. (1996). On shifted cardinal interpolation by Gaussians and multiquadrics, *J. Approx. Theory* **87**, pp. 36–59.

Baxter, B. J. C., Sivakumar, N. and Ward, J. D. (1994). Regarding the p-norms of radial basis interpolation matrices, *Constr. Approx.* 10, pp. 451–468.

Beatson, R. K., Cherrie, J. B. and Mouat, C. T. (1999). Fast fitting of radial basis functions: methods based on preconditioned GMRES iteration, *Adv. Comput. Math.* 11, pp. 253–270.

Beatson, R. K. and Bui, H.-Q. (2003). Mollification formulas and implicit smoothing, Research Report UCDMS 2003/19, University of Canterbury.

Beatson, R. K., Bui, H.-Q. and Levesley, J. (2005). Embeddings of Beppo-Levi spaces in Hölder-Zygmund spaces, and a new method for radial basis function interpolation error estimates, *J. Approx. Theory* **137**, pp. 166–178.

Beatson, R. K. and Dyn, N. (1996). Multiquadric B-splines, *J. Approx. Theory* **87**, pp. 1–24.

Beatson, R. K., Goodsell, G. and Powell, M. J. D. (1996). On multigrid techniques for thin plate spline interpolation in two dimensions, in *The Mathematics of Numerical Analysis*, Lectures in Appl. Math., 32, Amer. Math. Soc. (Providence, RI), pp. 77–97.

Beatson, R. K. and Greengard, L. (1997). A short course on fast multipole methods, in *Wavelets, Multilevel Methods and Elliptic PDEs* (Leicester, 1996), M. Ainsworth, J. Levesley, M. Marietta and W. A. Light (eds.), Numer. Math. Sci. Comput., Oxford Univ. Press (New York), pp. 1–37.

Beatson, R. K. and Langton, M. K. (2006). Integral interpolation, in *Algorithms for Approximation V*, A. Iske and J. Levesley (eds.), Springer-Verlag, Heidelberg, pp. 199–218.

Beatson, R. K. and Levesley, J. (2002). Good point/bad point iterations for solving the thin-plate spline interpolation equations, in *Approximation Theory X*, C. K. Chui, L. L. Schumaker, and J. Stöckier (eds.), Vanderbilt Univ. Press (Nashville, TN), pp. 17–25.

Beatson, R. K. and Light, W. A. (1992). Quasi-interpolation in the absence of polynomial reproduction, in *Numerical Methods in Approximation Theory*, ISNM 105, D. Braess, L. L. Schumaker (ed.), Birkhäuser (Basel), pp. 21–39.

Beatson, R. K. and Light, W. A. (1993). Quasi-interpolation with thin plate splines on a square, *Constr. Approx.* **9**, pp. 407–433.

Beatson, R. K. and Light, W. A. (1997). Fast evaluation of radial basis functions: methods for two-dimensional polyharmonic splines, *IMA J. Numer. Anal.* **17**, pp. 343–372.

Beatson, R. K., Light, W. A. and Billings, S. (2000). Fast solution of the radial basis function interpolation equations: domain decomposition methods, *SIAM J. Sci. Comput.* **22**, pp. 1717–1740.

Beatson, R. K. and Newsam, G. N. (1992). Fast evaluation of radial basis functions: I, *Comput. Math. Appl.* **24**, pp. 7–19.

Beatson, R. K. and Newsam, G. N. (1998). Fast evaluation of radial basis functions: Moment based methods, *SIAM J. Sci. Comput.* **19**, pp. 1428–1449.

Beatson, R. K. and Powell, M. J. D. (1992a). Univariate multiquadric approximation: quasi-interpolation to scattered data, *Constr. Approx.* **8**, pp. 275–288.

Beatson, R. K. and Powell, M. J. D. (1992b). Univariate interpolation on a regular finite grid by a multiquadric plus a linear polynomial, *J. Inst. Math. Applies.* **12**, pp. 107–133.

Beatson, R. K. and Powell, M. J. D. (1993). An iterative method for thin plate spline interpolation that employs approximations to Lagrange functions, in *Numerical Analysis 1993 (Dundee, 1993)*, G. A. Watson and D. F. Griffiths (ed.), Pitman Res. Notes Math. Ser., 303, Longman Sci. Tech. (Harlow), pp. 17–39.

Behrens, J., Iske, A. and Käser, M. (2002). Adaptive meshfree method of backward characteristics for nonlinear transport equations, in *Lecture Notes in Computer Science and Engineering* Vol.26: Meshfree Methods for Partial Differential Equations, M. Griebel and M. A. Schweitzer (eds.), Springer Verlag, pp. 21–36.

Bejancu, A. (1999)., Local accuracy for radial basis function interpolation on finite uniform grids, *J. Approx. Theory* **99**, pp. 242–257.

Belytschko, T., Krongauz, Y., Organ, O., Fleming, M. and Krysl, P. (1996). Meshless methods: an overview and recent developments, *Comp. Meth. Appl. Mech. Eng.* **139**, pp. 3–47.

Ben Moussa, B., Lanson, N. and Vila, J. P. (1999). Convergence of meshless methods for conservation laws. Applications to Euler equations, in *Hyperbolic problems: Theory, numerics, applications. Proceedings of the 7th international conference, Zürich, Switzerland, February 1998. Vol. I*, Fey, Michael *et al.* (eds.), Birkhäuser, Basel, pp. 31–40.

Berg, C., Christensen, J. P. R. and Ressel, P. (1984). *Harmonic Analysis on Semigroups*, Springer (Berlin).

Berlinet, A. and Thomas-Agnan, C. (2004). *Reproducing Kernel Hilbert Spaces in Probability and Statistics*, Kluwer (Dordrecht).

Bernal, F. and Kindelan, M. (2006). Meshless simulation of Hele-Shaw flow, submitted.

Beyer, A. (1994). Optimale Centerverteilung bei Interpolation mit radialen Basisfunktionen, Diplomarbeit, Universität Göttingen.

Binev, P. and Jetter, K. (1992). Estimating the condition number for multivariate interpolation problems, in *Numerical Methods in Approximation Theory*, ISNM 105, D. Braess, L. L. Schumaker (ed.), Birkhäuser (Basel), pp. 41–52.

Bingham, N. H. (1973). Positive definite functions on spheres, *Proc. Camb. Phil. Soc.* **73**, pp. 145–156.

Bishop, C. (1991). Improving the generalization properties of radial basis function neural networks, *Neural Computation* **3**, pp. 579–588.

Blumenthal, L. M. (1938). *Distance Geometries*, Univ. of Missouri Studies, **13**, 142pp.

Bochner, S. (1932). *Vorlesungen über Fouriersche Integrale*, Akademische Verlagsgesellschaft (Leipzig).

Bochner, S. (1933). Monotone Funktionen, Stieltjes Integrale und harmonische Analyse, *Math. Ann.* **108**, pp. 378–410.

Bochner, S. (1941). Hilbert distances and positive definite functions, *Ann. of Math.* 42, pp. 647–656.

de Boor, C. (1993). Approximation order without quasi-interpolants, in *Approximation Theory VII*, E. W. Cheney, C. Chui, and L. Schumaker (eds.), Academic Press (New York), pp. 1–18.

de Boor, C. On interpolation by radial polynomials, *Adv. in Comput. Math.* 24, pp. 143–153.

de Boor, C., DeVore, R. A. and Ron, A. (1994a). Approximation from shift-invariant subspaces of $L_2(\mathbb{R}^d)$, *Trans. Amer. Math. Soc.* **341**, pp. 787–806.

de Boor, C., DeVore, R. A. and Ron, A. (1994b). The structure of finitely generated shift-invariant spaces in $L_2(\mathbb{R}^d)$, *J. Funct. Anal.* **119**, pp. 37–78.

de Boor, C. and Ron, A. (1990). On multivariate polynomial interpolation, *Constr. Approx.* **6**, pp. 287–302.

de Boor, C. and Ron, A. (1992a). The least solution for the polynomial interpolation problem, *Math. Z.* **210**, pp. 347–378.

de Boor, C. and Ron, A. (1992b). Fourier analysis of the approximation power of principal shift-invariant spaces, *Constr. Approx.* **8**, pp. 427–462.

Bors, A.G. and Pitas, I. (1996). Median radial basis function neural network, *IEEE Trans. Neural Networks* 7, pp. 1351–1364.

Bos, L. P. and Maier, U. (2002). On the asymptotics of Fekete-type points for univariate radial basis interpolation, *J. Approx. Theory* **119**, pp. 252–270.

Bos, L. P. and Šalkauskas, K. (1987). On the matrix $[|x_i - x_j|^3]$ and the cubic spline continuity equations, *J. Approx. Theory* **51**, pp. 81–88.

Bos, L. P. and Šalkauskas, K. (1989). Moving least-squares are Backus-Gilbert optimal, *J. Approx. Theory* **59**, pp. 267–275.

Bouhamidi, A. and Le Méhauté, A. (2004). Radial basis functions under tension, *J. Approx. Theory* **127**, pp. 135–154.

Bozzini, M., Lenarduzzi, L. and Schaback, R. (2002). Adaptive interpolation by scaled multiquadrics, *Adv. in Comp. Math.* **16**, pp. 375–387.

Bozzini, M., Lenarduzzi, L. and Schaback, R. (2006). Kernel B-splines and interpolation, *Numer. Algorithms* 41 1, pp. 1–16.

Braess, D. (1997). *Finite Elements: Theory, Fast Solvers, and Applications in Solid Mechanics*, Cambridge University Press (Cambridge).

Brenner, S. C. and Scott, L. R. (1994). *The Mathematical Theory of Finite Element Methods*, Springer Verlag (New York).

Broomhead, D. S. and Lowe, D. (1988). Multivariate functional interpolation and adaptive networks, *Complex Systems* **2**, pp. 321–355.

Brown, A. L. (1992). Uniform approximation by radial basis functions, Appendix B to Radial basis functions in 1990, in *Advances in Numerical Analysis II: Wavelets, Subdivision, and Radial Basis Functions*, W. Light (ed.), Oxford University Press (Oxford), pp. 203–206.

Brown, D., Ling, L., Kansa, E. and Levesley, J. (2005). On approximate cardinal preconditioning methods for solving PDEs with radial basis functions, *Engineering Analysis with Boundary Elements* **29**, pp. 343–353.

Brownlee, R. and Light, W. (2004). Approximation orders for interpolation by surface splines to rough functions, *IMA J. Numer. Anal.* **24**, pp. 179–192.

Buckley, M. J. (1994). Fast computation of a discretized thin-plate smoothing spline for image data, *Biometrika* **81**, pp. 247–258.

Buhmann, M. D. (1988). Convergence of univariate quasi-interpolation using multiquadrics, *IMA J. Numer. Anal.* 8, pp. 365–383.

Buhmann, M. D. (1989a). Multivariate interpolation using radial basis functions, Ph.D. Dissertation, University of Cambridge.

Buhmann, M. D. (1989b). Cardinal interpolation with radial basis functions: an integral approach, in *Multivariate Approximation Theory IV*, ISNM 90, C. Chui, W. Schempp, and K. Zeller (eds.), Birkhäuser Verlag (Basel), pp. 41–64.

Buhmann, M. D. (1990a). Multivariate interpolation in odd-dimensional Euclidean spaces using multiquadrics, *Constr. Approx.* 6, pp. 21–34.

Buhmann, M. D. (1990b). Multivariate cardinal interpolation with radial basis functions, *Constr. Approx.* 6, pp. 225–255.

Buhmann, M. D. (1993a). New developments in the theory of radial basis function interpolation, in *Multivariate Approximation: From CAGD to Wavelets*, Kurt Jetter and Florencio Utreras (eds.), World Scientific Publishing (Singapore), pp. 35–75.

Buhmann, M. D. (1993b). On quasi-interpolation with radial basis functions, *J. Approx. Theory* **72**, pp. 103–130.

Buhmann, M. D. (1993c). Discrete least squares approximation and prewavelets from radial function spaces, *Proc. Camb. Phil. Soc.* 114, pp. 533–558.

Buhmann, M. D. (1995a). Multiquadric prewavelets on nonequally spaced knots in one dimension, *Math. Comp.* **64**, pp. 1611–1625.

Buhmann, M. D. (1995b). Pre-wavelets on scattered knots and from radial function spaces: a review, in *The Mathematics of Surfaces VI*, R. R. Martin (ed.), Clarendon Press (Oxford), pp. 309–324.

Buhmann, M. D. (1998). Radial functions on compact support, *Proc. Edin. Math. Soc. II* **41**, pp. 33-46.

Buhmann, M. D. (2000). Radial basis functions, *Acta Numerica 2000* 9, pp. 1–38.

Buhmann, M. D. (2003). *Radial Basis Functions: Theory and Implementations*, Cambridge University Press (Cambridge).

Buhmann, M. D. (2006). Half-plane approximation with radial functions I, *Comput. Math. Appl.* **51** 8, pp. 1171–1184.

Buhmann, M. D. and Chui, C. K. (1993). A note on the local stability of translates of radial basis functions, *J. Approx. Theory* **74**, pp. 36–40.

Buhmann, M. D., Derrien, F. and Le Méhauté, A. (1995). Spectral properties and knot removal for interpolation by pure radial sums, in *Mathematical Methods for Curves and Surfaces*, M. Dæhlen, T. Lyche, and L. Schumaker (eds.), Vanderbilt University Press (Nashville), pp. 55–62.

Buhmann, M. D. and Dyn, N. (1991). Error estimates for multiquadric interpolation, in *Curves and Surfaces*, P.-J. Laurent, A. Le Méhauté, and L. L. Schumaker (eds.), Academic Press (New York), pp. 51–58.

Buhmann, M. D. and Dyn, N. (1993). Spectral convergence of multiquadric interpolation, *Proc. Edinburgh Math. Soc.* **36**, pp. 319–333.

Buhmann, M. D., Dyn, N. and Levin, D. (1993). On quasi-interpolation with radial basis functions on non-regular grids, *Constr. Approx.* **11**, pp. 239–254.

Buhmann, M. D. and Le Méhauté, A. (1995). Knot removal with radial basis function interpolantion, *C. R. Acad. Sci. Paris Sér. I Math.* **320**, pp. 501–506.

Buhmann, M. D. and Micchelli, C. A. (1989). Completely monotonic functions for cardinal interpolation, in *Approximation Theory VI*, C. Chui, L. Schumaker, and J. Ward (eds.), Academic Press (New York), pp. 1–4.

Buhmann, M. D. and Micchelli, C. A. (1991). Multiply monotone functions for cardinal interpolation, *Adv. in Appl. Math.* **12**, pp. 358–386.

Buhmann, M. D. and Micchelli, C. A. (1992a). Multiquadric interpolation improved, *Comput. Math. Appl.* **24**, pp. 21–25.

Buhmann, M. D. and Micchelli, C. A. (1992b). On radial basis approximation on periodic grids, *Proc. Camb. Phil. Soc.* **112**, pp. 317–334.

Buhmann, M. D. and Pinkus, A. (1997). On a recovery problem, *Annals of Numerical Mathematics* **4**, pp. 129–142,

Buhmann, M. D. and Powell, M. J. D. (1990). Radial basis function interpolation on an infinite regular grid, in *Algorithms for Approximation II*, M. G. Cox and J. C. Mason (eds.), Chapman & Hall (London), pp. 146–169.

Buhmann, M. D. and Ron, A. (1994). Radial basis functions: L^p-approximation orders with scattered centres, in *Wavelets, Images, and Surface Fitting*, P.-J. Laurent, A. Le Méhauté, and L. L. Schumaker (eds.), A. K. Peters (Wellesley, MA), pp. 93–112.

Caiti, A., Magenes, G., Panisini, T. and Simpson, R. (1994). Smooth approximation by radial basis functions: three case studies, *J. Appl. Sc. Comp.* **1**, pp. 88–113.

Cambanis, S., Keener, R. and Simons, G. (1983). On α-symmetric multivariate distributions, *J. Multivariate Anal.* **13**, pp.213–233.

Canuto, C., Hussaini, M. Y., Quarteroni, A. and Zang, T. A. (1988). *Spectral Methods in Fluid Dynamics*, Springer Verlag, Berlin.

Carlson, R. E. and Foley, T. A. (1991). The parameter R^2 in multiquadric interpolation, *Comput. Math. Appl.* **21**, pp. 29–42.

Carlson, R. E. and Foley, T. A. (1992). Interpolation of track data with radial basis methods, *Comput. Math. Appl.* **24**, pp. 27–34.

Carlson, R. E. and Natarajan, B. K. (1994). Sparse approximate multiquadric interpolation, *Comput. Math. Appl.* **27**, pp. 99–108.

Carr, J. C., Beatson, R. K., Cherie, J. B., Mitchell, T. J., Fright, W. R., McCallum, B. C. and Evans, T. R. (2001). Reconstruction and representation of 3D objects with radial basis functions, in *SIGGRAPH 2001*, pp. 67–76.

Carr, J. C., Fright, W. R. and Beatson, R. K. (1997). Surface Interpolation with Radial Basis Functions for Medical Imaging, *IEEE Transactions on Medical Imaging* 16, pp. 96–107.

Casciola, G., Lazzaro, D., Montefusco, L. B. and Morigi, S. (2006). Shape preserving surface reconstruction using locally anisotropic RBF interpolants, *Comput. Math. Appl.* **51** 8, pp. 1185–1198.

Cha, I. and Kassam, S. A. (1992). Blind equalisation using radial basis function networks, *Proc. Canadian Conference Elec. Eng. and Comp. Sci.* 1, TM.3.13.14.

Cha, I. and Kassam, S. A. (1995a). Channel equalization using adaptive complex radial basis function networks, *IEEE J. on Selected Areas in Communications* **13**, pp. 121–131.

Cha, I. and Kassam, S. A. (1995b). Interference canellation using radial basis function networks, *EURASIP Signal Processing* 14**7**, pp. 247–268.

Chakravarthy, S. V. and Ghosh, J. (1996). Scale-based clustering using the radial basis function network, *IEEE Trans. Neural Networks* 7, pp. 1250–1261.

Chan, A. K., Chui, C. K. and Guan, L. T. (1990). Radial basis function approach to interpolation of large reflecting surfaces, in *Curves and Surfaces in Computer Vision and Graphics*, L. A. Ferrari and R. J. P. de Figueiredo (eds.), SPIE (Vol. 1251), pp. 62–72.

Chan, R. and Strang, G. (1989). Toeplitz equations by conjugate gradients with circulant preconditioners, *SIAM J. Sci. Statist. Comput.* 10, pp. 104–119.

Chan, T. F. and Foulser, D. E. (1988). Effectively well-conditioned linear systems, *SIAM J. Sci. Statist. Comput.* 9, pp. 963–969.

Chang, K. F. (1996). Strictly positive definite functions, *J. Approx. Theory* **87**, 148–158. Addendum: *J. Approx. Theory* **88** 1997, p. 384.

Chen, C. S. and Brebbia, C. A. (1998). The dual reciprocity method for Helmholtz-type operators, in *Boundary Elements XX*, A. Kassab, C. A. Brebbia and M. Chopra (eds.), Computational Mechanics Publications, pp. 495–504.

Chen, C. S., Brebbia, C. A. and Power, H. (1999). Boundary element methods using compactly supported radial basis functions, *Comm. Numer. Meths. Eng.* **15**, pp. 137–150.

Chen, C. S., Ganesh, M., Golberg, M. A. and Cheng, A. H.-D. (2002). Multilevel compact radial functions based computational schemes for some elliptic problems, *Comput. Math. Appl.* **43**, pp. 359–378.

Chen, C. S., Golberg, M. A. and Schaback, R. (2003). Recent developments in the dual reciprocity method using compactly supported radial basis functions, in *Transformation of Domain Effects to the Boundary*, Y. F. Rashed and C. A. Brebbia (eds.), WIT Press, Southampton, Boston, pp. 138–225.

Chen, J. S., Wu, C. T., Yoon, S. and You, Y. (2001). A stabilized conforming nodal integration for Galerkin meshfree methods, *Int. J. Numer. Meth. Engng.* **50** 2, pp. 435–466.

Chen, J.-T., Lee, Y.-T., Chen, I-L. and Chen, K.-H. (2004). Mathematical analysis and treatment for the true and spurious eigenequations of circular plates by the meshless method using radial basis function, *J. Chinese Inst. Engineers* **27**, pp. 547–561.

Chen, S. (1994). Radial basis functions for signal prediction and system modelling, *J. Appl. Sci. Comput.* 1, pp. 172–207.

Chen, S., Billings, S. A., Cowan, C. F. N. and Grant, P. M. (1990). Non-linear systems identification using radial basis functions, *Int. J. Systems Science* **21**, pp. 2513–2539.

Chen, S., Billings, S. A. and Grant, P. M. (1992). Recursive hybrid algorithm for non-linear system identification using radial basis function networks, *Int. J. Control* **55**, pp. 1051–1070.

Chen, S., Cowan, C. F. N. and Grant, P. M. (1991a). Orthogonal least squares learning algorithm for radial basis function networks, *IEEE Trans. Neural Networks* **2**, pp. 302–309.

Chen, S., Gibson, G. J., Cowan, C. F. N. and Grant, P. M. (1991b). Reconstruction of binary signals using an adaptive radial-basis-function equaliser, *EURASIP Signal Processing* **22**, pp. 77–93.

Chen, S., McLaughlin, S., Grant, P. M. and Mulgrew, B. (1993). Reduced-complexity multistage blind clustering equaliser, *IEEE Int. Communications Conference (ICC) Proceedings*, pp. 1149–1153.

Chen, S., McLaughlin, S. and Mulgrew, B. (1994a). Complex valued radial basis function networks: network architecture and learning algorithms (part I), *EURASIP Signal Processing J.* **25**, pp. 19–31.

Chen, S., McLaughlin, S. and Mulgrew, B. (1994b). Complex valued radial basis function networks: application to digital communications channel equalisation (part II), *EURASIP Signal Processing J.* **36**, pp. 175–188.

Chen, S. and Mulgrew, B. (1992). Overcoming co-channel interference using an adaptive radial basis function equaliser, *EURASIP Signal Processing J.* **28**, pp. 91–107.

Chen, T. and Chen, H. (1995a). Approximation capability to functions of several variables, nonlinear functionals and operators by radial basis function neural networks, *IEEE Trans. Neural Networks* **6**, pp. 904–910.

Chen, T. and Chen, H. (1995b). Denseness of radial-basis functions in $L^2(\mathbb{R}^n)$ and its applications in neural networks, in *Approximation Theory VIII, Vol. 1: Approximation and Interpolation*, C. Chui, and L. Schumaker (eds.), World Scientific Publishing (Singapore), pp. 137–144.

Cheney, E. W. (1992). Approximation by functions of nonclassical form, in *Approximation Theory, Spline Functions and Applications*, S. P. Singh (ed.), Kluwer (Dordrecht), pp. 1–18.

Cheney, E. W. (1995a). Approximation and interpolation on spheres, in *Approximation Theory, Wavelets and Applications*, S. P. Singh (ed.), Kluwer (Dordrecht), pp. 47–53.

Cheney, E. W. (1995b). Approximation using positive definite functions, in *Approximation Theory VIII, Vol. 1: Approximation and Interpolation*, C. Chui, and L. Schumaker (eds.), World Scientific Publishing (Singapore), pp. 145–168.

Cheney, E. W. and Light, W. A. (1999). *A Course in Approximation Theory*, Brooks/Cole (Pacific Grove, CA).

Cheney, E. W., Light, W. A. and Xu, Y. (1992). On kernels and approximation orders, in *Proc. Sixth Southeastern Approximation Theory Conference*, George Anastassiou (ed.), Lecture Notes in Pure and Applied Mathematics, Vol. 138, pp. 227–242.

Cheney, E. W. and Xu, Y. (1993). A set of research problems in approximation theory, in *Topics in Polynomials of One and Several Variables and Their Applications*, T. M. Rassias, H. M. Srivastava, and A. Yanushauskas (eds.), World Scientific Publishers (London), pp. 109–123.

Cheng, A. H. D., Golberg, M. A., Kansa, E. J. and Zammito, G. (2003). Exponential convergence and *H-c* multiquadric collocation method for partial differential equations, *Numer. Methods Partial Differential Equations* **19** 5, pp. 571–594.

Cheng, A. H.-D. and Cabral, J. J. S. P. (2005). Direct solution of certain ill-posed boundary

value problems by collocation method, in *Boundary Elements XXVII*, A. Kassab, C. A. Brebbia, E. Divo, and D. Poljak (eds.), WIT Press (Southampton), pp. 35–44.

Cheng, C. C. and Zheng, Y. F. (1994). Thin plate spline surface approximation using Coons' patches, *Comput. Aided Geom. Design* **11**, pp. 269–287.

Cherrie, J. B., Beatson, R. K. and Newsam, G. N. (2002). Fast evaluation of radial basis functions: methods for generalized multiquadrics in $\mathbb{R}^n$, *SIAM J. Sci. Comput.* **23**, pp. 1549–1571.

Chien, C.-C. and Wu, T.-Y. (2004). An accurate solution to the meshless local Petrov-Galerkin formulation in elastodynamics, *J. Chinese inst. Engineers* **27**, pp. 463–471.

Chng, E. S., Chen, S. and Mulgrew, B. (1996). Gradient radial basis function networks for nonlinear and nonstationary time series prediction, *IEEE Trans. Neural Networks* **7**, pp. 190–194.

Chng, E. S., Chen, S., Mulgrew, B. and Gibson, G. J. (1995). Realising the Bayesian decision boundary for channel equalisation using radial basis function network and linear equaliser, in *Mathematics for Neural Networks and Applications*, Oxford.

Chui, C. K. (1988). *Multivariate Splines*, CBMS-NSF Reg. Conf. Ser. in Appl. Math. 54 (SIAM).

Chui, C. K., Stöckier, J. and Ward, J. D. (1996). Analytic wavelets generated by radial functions, *Adv. Comput. Math.* **5**, pp. 95–123.

Chui, C. K., Ward, J. D. and Jetter, K. (1992). Cardinal interpolation with differences of tempered functions, *Comput. Math. Appl.* **24**, pp. 35–48.

Chung, K. C. and Yao, T. H. (1977). On lattices admitting unique Lagrange interpolations, *SIAM J. Numer. Anal.* **14**, pp. 735–743.

Cid-Sueiro, J., Artes-Rodriguez, A. and Figueiras-Vidal, A. R. (1994). Recurrent radial basis function networks for optimal symbol-by-symbol equalisation, *EURASIP Signal Processing* 40, pp. 53–63.

Cid-Sueiro, J. and Figueiras-Vidal, A. R. (1993). Recurrent radial basis function networks for optimal equalisation, *Proc. IEEE Workshop on Neural Networks for Signal Processing*, pp. 562–571.

Cleveland, W. S. and Loader, C. L. (1996). Smoothing by local regression: Principles and methods, in *Statistical Theory and Computational Aspects of Smoothing*, W. Haerdle and M. G. Schimek (eds.), Springer (New York), pp. 10–49.

Cuppens, R. (1975). *Decomposition of Multivariate Probability*, Academic Press (New York).

Curtis, P. C., Jr. (1959) n-parameter families and best approximation, *Pacific J. Math.* 9, pp. 1013–1027.

Davidov, O., Sestini, A. and Morandi, R. (2005). Local RBF approximation for scattered data fitting with bivariate splines, in *Trends and Applications in Constructive Approximation*, M. G. de Bruin, D. H. Mache, and J. Szabados (eds.), Birkhäuser (Basel), pp. 91–102.

De Forest, E. L. (1873). On some methods of interpolation applicable to the graduation of irregular series, Annual Report of the Board of Regents of the Smithsonian Institution for 1871, pp. 275–339.

De Forest, E. L. (1874). Additions to a memoir on methods of interpolation applicable to the graduation of irregular series, Annual Report of the Board of Regents of the Smithsonian Institution for 1873, pp. 319–353.

Delvos, F. J. (1985). Convergence of interpolation by translation, *Colloq. Math. Soc. J. Bolya* 49, pp. 273–287.

Delvos, F. J. (1987). Periodic interpolation on uniform meshes, *J. Approx. Theory* **51**, pp. 71–80.

De Marchi, S., Schaback, R. and Wendland, H. (2005). Near-optimal data-independent point locations for radial basis function interpolation, *Adv. in Comput. Math.* **23** 3, pp. 317–330.

Dilts, G. (1996). Equivalence of the SPH method and a space-time Galerkin moving particle method, Los Alamos National Laboratory.

Dix, J. G. and Ogden, R. D. (1994). An interpolation scheme with radial basis in Sobolev spaces $H^s(\mathbb{R}^n)$, *Rocky Mountain J. Math.* **24**, pp. 1319–1337.

Djokovič, D. and van Damme, R. (1996). Radial splines and moving grids, Technical report, University of Twente.

Djokovič, D. and van Damme, R. (1997). Moving node methods for PDE's using radial basis functions and B-splines, Technical report, University of Twente.

Dolbow, J. and Belytschko, T. (1997). An introduction to programming the meshless element free Galerkin method, Technical report, Northwestern University.

Donoghue, W. F. (1974). *Monotone Matrix Functions and Analytic Continuation*, Springer (Berlin).

Driscoll, T. A. and Fornberg, B. (2002). Interpolation in the limit of increasingly flat radial basis functions, *Comput. Math. Appl.* **43**, pp. 413–422.

Driscoll, T. A. and Heryudono, A. (2006). Adaptive residual subsampling methods for radial basis function interpolation and collocation problems, *Comput. Math. Appl.*, submitted.

Duan, Z.-H. and Krasny, R. (2001). An adaptive treecode for computing nonbonded potential energy in classical molecular systems, *J. Comput. Chemistry* **22**, pp. 184–195.

Duarte, C. A. (1995). A review of some meshless methods to solve partial differential equations, TICAM Report 95-06, University of Texas.

Duarte, C. A. and Oden, J. T. (1995a). *Hp* clouds — a meshless method to solve boundary-value problems, TICAM Report 95-05, University of Texas.

Duarte, C. A. and Oden, J. T. (1996a). An *h-p* adaptive method using clouds, *Comput. Meth. Appl. Mech. Engg.* **139** 1, pp. 237–262.

Duarte, C. A. and Oden, J. T. (1996b). *H-p* clouds — an *h-p* meshless method, *Num. Meth. for Part. Diff. Eq.* **12**, pp. 673–705.

Dubai, M. R. (1992). Construction of three-dimensional black-hole initial data via multiquadrics, *Phys. Rev. D* **45**, pp. 1178–1187.

Dubai, M. R. (1994). Domain decomposition and local refinement for multiquadric approximations. I: Second-order equations in one-dimension, *J. Appl. Sc. Comp.* **1**, pp. 146–171.

Dubai, M. R., Oliveira, S. R. and Matzner, R. A. (1992). Solution of elliptic equations in numerical relativity using multiquadrics, in *Approaches to Numerical Relativity*, R. d'Inverno (ed.), Cambridge University Press (Cambridge), pp. 265–280.

Duchon, J. (1976). Interpolation des fonctions de deux variables suivant le principe de la flexion des plaques minces, *Rev. Francaise Automat. Informat. Rech. Opér., Anal. Numer.* **10**, pp. 5–12.

Duchon, J. (1977). Splines minimizing rotation-invariant semi-norms in Sobolev spaces, in *Constructive Theory of Functions of Several Variables, Oberwolfach 1976*, W. Schempp and K. Zeller (eds.), Springer Lecture Notes in Math. 571, Springer-Verlag (Berlin), pp. 85–100.

Duchon, J. (1978). Sur l'erreur d'interpolation des fonctions de plusieurs variables par les D^m-splines, *Rev. Francaise Automat. Informat. Rech. Opér., Anal. Numer.* **12**, pp. 325–334.

Duchon, J. (1980). Fontions splines homogènes à plusiers variables, Université de Grenoble.

Dyn, N. (1987). Interpolation of scattered data by radial functions, in *Topics in Multivari-*

ate Approximation, C. K. Chui, L. L. Schumaker, and F. Utreras (eds.), Academic Press (New York), pp. 47–61.

Dyn, N. (1989). Interpolation and approximation by radial and related functions, in *Approximation Theory VI*, C. Chui, L. Schumaker, and J. Ward (eds.), Academic Press (New York), pp. 211–234.

Dyn, N., Goodman, T. and Micchelli, C. A. (1986). Positive powers of certain conditionally negative definite matrices, *Indaga. Math. Proc. Ser.A.* **89**, pp. 163–178.

Dyn, N., Jackson, I. R. H., Levin, D. and Ron, A. (1992). On multivariate approximation by the integer translates of a basis function, *Israel J. Math.* **78**, pp. 95–130.

Dyn, N. and Levin, D. (1981). Bell shaped basis functions for surface fitting, in *Approximation Theory and Applications*, Z. Ziegler (ed.), Academic Press (New York), pp. 113–129.

Dyn, N. and Levin, D. (1982). Construction of surface spline interpolants of scattered data over finite domains, *Rev. Francaise Automat. Informat. Rech. Opér., Anal. Numer.* **16**, pp. 201–209.

Dyn, N. and Levin, D. (1983). Iterative solution of systems originating from integral equations and surface interpolation, *SIAM J. Numer. Anal.* **20**, pp. 377–390.

Dyn, N., Levin, D. and Rippa, S. (1983). Surface interpolation and smoothing by "thin plate" splines, in *Approximation Theory IV*, C. Chui, L. Schumaker, and J. Ward (eds.), Academic Press (New York), pp. 445–449.

Dyn, N., Levin, D. and Rippa, S. (1986). Numerical procedures for surface fitting of scattered data by radial functions, *SIAM J. Sci. Statist. Comput.* **7**, pp. 639–659.

Dyn, N., Light, W. A. and Cheney, E. W. (1989). Interpolation by piecewise-linear radial basis functions, I, *J. Approx. Theory* **59**, pp. 202–223.

Dyn, N. and Micchelli, C. A. (1990). Interpolation by sums of radial functions, *Numer. Math.* **58**, pp. 1–9.

Dyn, N., Narcowich, F. J. and Ward, J. D. (1997). A framework for interpolation and approximation on Riemannian manifolds, in *Approximation theory and optimization (Cambridge, 1996)*, Cambridge Univ. Press (Cambridge), pp. 133–144.

Dyn, N., Narcowich, F. J. and Ward, J. D. (1999). Variational principles and Sobolev-type estimates for generalized interpolation on a Riemannian manifold, *Constr. Approx.* **15** 2, pp. 175–208.

Dyn, N. and Ron, A. (1995a). Radial basis function approximation: from gridded centers to scattered centers, *Proc. London Math. Soc.* **71**, pp. 76–108.

Dyn, N. and Ron, A. (1995b). Multiresolution analysis by infinitely differentiable compactly supported functions, *Appl. Comput. Harmon. Anal.* **2**, pp. 15–20.

Evgeniou, T., Pontil, M. and Poggio, T. (2000). Regularization networks and support vector machines, *Adv. Comput. Math.* **13** 1, pp. 1–50.

Fan, J. and Gijbels, I. (1996). *Local Polynomial Modelling and its Applications*, Chapman & Hall (New York).

Farwig, R. (1986). Multivariate interpolation of arbitrarily spaced data by moving least squares methods, *J. Comput. Appl. Math.* **16**, pp. 79–93.

Farwig, R. (1987). Multivariate interpolation of scattered data by moving least squares methods, in *Algorithms for Approximation*, Oxford Univ. Press (New York), pp. 193–211, 1987.

Farwig, R. (1991). Rate of convergence of moving least squares interpolation methods: the univariate case, in *Progress in Approximation Theory*, Academic Press (Boston, MA), pp. 313–327.

Fasshauer, G. E. (1994a). Scattered data interpolation with radial basis functions on the sphere, Technical Report, Vanderbilt University.

Fasshauer, G. E. (1994b). On the density of certain classes of radial basis functions on spheres, Technical Report, Vanderbilt University.

Fasshauer, G. E. (1995a). Adaptive least squares fitting with radial basis functions on the sphere, in *Mathematical Methods for Curves and Surfaces*, M. Dæhlen, T. Lyche, and L. Schumaker (eds.), Vanderbilt University Press (Nashville), pp. 141–150.

Fasshauer, G. E. (1995b). Radial Basis Functions on Spheres, Ph.D. Dissertation, Vanderbilt University.

Fasshauer, G. E. (1997). Solving partial differential equations by collocation with radial basis functions, in *Surface Fitting and Multiresolution Methods*, A. Le Méhauté, C. Rabut, and L. L. Schumaker (eds.), Vanderbilt University Press (Nashville, TN), pp. 131–138.

Fasshauer, G. E. (1999a). On smoothing for multilevel approximation with radial basis functions, in *Approximation Theory IX, Vol.II: Computational Aspects*, Charles K. Chui, and L. L. Schumaker (eds.), Vanderbilt University Press, pp. 55–62.

Fasshauer, G. E. (1999b). Hermite interpolation with radial basis functions on spheres, *Adv. in Comp. Math.* **10**, pp. 81–96.

Fasshauer, G. E. (1999c). On the numerical solution of differential equations with radial basis functions, in *Boundary Element Technology XIII*, C. S. Chen, C. A. Brebbia, and D. W. Pepper (eds.), WIT Press, pp. 291–300.

Fasshauer, G. E. (1999d). Solving differential equations with radial basis functions: multilevel methods and smoothing, *Adv. in Comp. Math.* **11**, pp. 139–159.

Fasshauer, G. E. (2001a). Nonsymmetric multilevel RBF collocation within an operator Newton framework for nonlinear PDEs, in *Trends in Approximation Theory*, K. Kopotun, T. Lyche, and M. Neamtu (eds.), Vanderbilt University Press, pp. 103–112.

Fasshauer, G. E. (2001b). High-order moving least-squares approximation via fast radial Laguerre transforms, Technical Report, Illinois Institute of Technology.

Fasshauer, G. E. (2002a). Newton iteration with multiquadrics for the solution of nonlinear PDEs, *Comput. Math. Applic.* **43**, pp. 423–438.

Fasshauer, G. E. (2002b). Matrix-free multilevel moving least-squares methods, in *Approximation Theory X: Wavelets, Splines, and Applications*, C. K. Chui, L. L. Schumaker, and J. Stöckier (eds.), Vanderbilt University Press (Nashville), pp. 271–281.

Fasshauer, G. E. (2002c). Approximate moving least-squares approximation with compactly supported weights, in *Lecture Notes in Computer Science and Engineering Vol.26: Meshfree Methods for Partial Differential Equations*, M. Griebel and M. A. Schweitzer (eds.), Springer Verlag (Berlin), pp. 105–116.

Fasshauer, G. E. (2002d). Approximate moving least-squares approximation for time-dependent PDEs, in *WCCM V, Fifth World Congress on Computational Mechanics* (http://wccm.tuwien.ac.at), H. A. Mang, F. G. Rammerstorfer, and J. Eberhardsteiner (eds.), Vienna University of Technology (Vienna).

Fasshauer, G. E. (2003). Approximate moving least-squares approximation: A fast and accurate multivariate approximation method, in *Curve and Surface Fitting: Saint-Malo 2002*, A. Cohen, J.-L. Merrien, and L. L. Schumaker (eds.), Nashboro Press (Nashville), pp. 139–148.

Fasshauer, G. E. (2004). Toward approximate moving least squares approximation with irregularly spaced centers, *Computer Methods in Applied Mechanics & Engineering* **193**, pp. 1231–1243.

Fasshauer, G. E. (2005a). Dual bases and discrete reproducing kernels: a unified framework for RBF and MLS approximation, *Journal of Engineering Analysis with Boundary Elements* **29**, pp. 313–325.

Fasshauer, G. E. (2005b). RBF collocation methods as pseudospectral methods, in *Boundary Elements XXVII*, A. Kassab, C. A. Brebbia, E. Divo, and D. Poljak (eds.), WIT Press (Southampton), pp. 47–56.

Fasshauer, G. E. (2006). Meshfree Methods, in *Handbook of Theoretical and Computational Nanotechnology, Vol. 2*, M. Rieth and W. Schommers (eds.), American Scientific Publishers, pp. 33–97.

Fasshauer, G. E., Gartland, E. C. and Jerome, J. W. (2000a). Algorithms defined by Nash iteration: some implementations via multilevel collocation and smoothing, *J. Comp. Appl. Math.* **119**, pp. 161–183.

Fasshauer, G. E., Gartland, E. C. and Jerome, J. W. (2000b). Newton iteration for partial differential equations and the approximation of the identity, *Numerical Algorithms* **25**, pp. 181–195.

Fasshauer, G. E. and Jerome, J. W. (1999). Multistep approximation algorithms: Improved convergence rates through postconditioning with smoothing kernels, *Adv. in Comput. Math.* **10**, pp. 1–27.

Fasshauer, G. E., Khaliq, A. Q. M. and Voss, D. A. (2004). Using meshfree approximation for multi-asset American option problems, *J. Chinese Institute Engineers* **27**, pp. 563–571.

Fasshauer, G. E. and Schumaker, L. L. (1998). Scattered data fitting on the sphere, in *Mathematical Methods for Curves and Surfaces II*, M. Dæhlen, T. Lyche, and L. L. Schumaker (eds.), Vanderbilt University Press, pp. 117–166.

Fasshauer, G. E. and Zhang, J. G. (2004). Recent results for moving least squares approximation, in *Geometric Modeling and Computing: Seattle 2003*, M. L. Lucian and M. Neamtu (eds.), Nashboro Press (Brentwood, TN), pp. 163–176.

Fasshauer, G. E. and Zhang, J. G. (2006). Iterated approximate moving least squares approximation, submitted.

Faul, A. C. and Powell, M. J. D. (1999). Proof of convergence of an iterative technique for thin plate spline interpolation in two dimensions, *Adv. Comput. Math.* 11, pp. 183–192.

Faul, A. C. and Powell, M. J. D. (2000). Krylov subspace methods for radial basis function interpolation, in *Numerical Analysis 1999 (Dundee)*, Chapman & Hall/CRC (Boca Raton, FL), pp. 115–141.

Fedoseyev, A. I., Friedman, M. J. and Kansa, E. J. (2000). Continuation for nonlinear elliptic partial differential equations discretized by the multiquadric method, *Int. J. Bifurcation and Chaos* **10** 2, pp. 481–492.

Fedoseyev, A. I., Friedman, M. J. and Kansa, E. J. (2002). Improved multiquadric method for elliptic partial differential equations via PDE collocation on the boundary, *Comput. Math. Applic.* **43**, pp. 439–455.

Feller, W. (1966). *An Introduction to Probability Theory and Its Application, Vol. 2*, Wiley & Sons, New York.

Fenn, M. and Steidl, G. (2006). Robust local approximation of scattered data, in *Geometric Properties from Incomplete Data*, R. Klette, R. Kozera, L. Noakes, and J. Weickert (eds.), Kluwer (Dordrecht), pp. 317–334.

Ferreira, A. J. M. and Fasshauer, G. E. (2006) Computation of natural frequencies of shear deformable beams and plates by an RBF-pseudospectral method, *Comput. Meth. Appl. Mech. Engng.* **196**, 134–146.

Ferreira, A. J. M. and Fasshauer, G. E. (2007) Analysis of natural frequencies of composite plates by an RBF-pseudospectral method, *Composite Structures*, to appear.

Floater, M. S. and Iske, A. (1996a). Thinning and approximation of large sets of scattered data, in *Advanced Topics in Multivariate Approximation*, F. Fontanella, K. Jetter, and P.-J. Laurent (eds.), World Scientific Publishing (Singapore), pp. 87–96.

Floater, M. S. and Iske, A. (1996b). Multistep scattered data interpolation using compactly supported radial basis functions, *J. Comput. Applied Math.* **73**, pp. 65–78.

Florence, A. G. and van Loan, C. F. (2000). A Kronecker product formulation of the fast Gauss transform, preprint.

Flyer, N. (2006). Exact polynomial reprodction for oscillatory radial basis functions on infinite lattices, *Comput. Math. Appl.* **51** 8, pp. 1199–1208.

Foley, T. A. (1987). Interpolation and approximation of 3-D and 4-D scattered data, *Comput. Math. Appl.* **13**, pp. 711–740.

Foley, T. A. (1990). Interpolation of scattered data on a spherical domain, in *Algorithms for Approximation II*, M. G. Cox and J. C. Mason (eds.), Chapman & Hall (London), pp. 303–310.

Foley, T. A. (1992). The map and blend scattered data interpolant on the sphere, *Comput. Math. Appl.* **24**, pp. 49–60.

Foley, T. A. (1994). Near optimal parameter selection for multiquadric interpolation, *J. Appl. Sc. Comp.* **1**, pp. 54–69.

Foley, T. A., Dayanand, S. and Zeckzer, D. (1995). Localized radial basis methods using rational triangle patches, in *Geometric Modelling — Dagstuhl 1993*, H. Hagen, G. Farin and H. Noltemeier (eds.), Springer Verlag (Berlin), pp. 163–176.

Foley, T. A., Lane, D. A., Nielson, G. M., Franke, R. and Hagen, H. (1990a). Interpolation of scattered data on closed surfaces, *Comput. Aided Geom. Design* **7**, pp. 303–312.

Foley, T. A., Lane, D. A., Nielson, G. M. and Ramaraj, R. (1990b). Visualizing functions over a sphere, *Comp. Graphics and Applies.* **10**, pp. 32–40.

Fornberg, B. (1998). *A Practical Guide to Pseudospectral Methods*, Cambridge Univ. Press.

Fornberg, B., Driscoll, T. A., Wright, G. and Charles, R. (2002). Observations on the behavior of radial basis function approximations near boundaries, *Comput. Math. Appl.* **43**, pp. 473–490.

Fornberg, B. and Flyer, N. (2005). Accuracy of radial basis function interpolation and derivative approximations on 1-D infinite grids, *Adv. Comput. Math.* **23** 1-2, pp. 5–20.

Fornberg, B., Larsson, E. and Wright, G. (2004). A new class of oscillatory radial basis functions, *Comput. Math. Appl.* **51** 8, pp. 1209–1222.

Fornberg, B. and Wright, G. (2004). Stable computation of multiquadric interpolants for all values of the shape parameter, *Comput. Math. Appl.* **47**, pp. 497–523.

Fornberg, B. and Zuev, J. (2006). The Runge phenomenon and spatially variable shape parameters in RBF interpolation, *Comput. Math. Appl.*, submitted.

Franke, C. and Schaback, R. (1998a). Solving partial differential equations by collocation using radial basis functions, *Appl. Math. Comp.* **93**, pp. 73–82.

Franke, C. and Schaback, R. (1998b). Convergence orders of meshless collocation methods using radial basis functions, *Adv. in Comput. Math.* **8**, pp. 381–399.

Franke, R. (1977). Locally determined smooth interpolation at irregularly spaced points in several variables, *J. Inst. Math. Applic.* **19**, pp. 471–482.

Franke, R. (1982a). Scattered data interpolation: tests of some methods, *Math. Comp.* **48**, pp. 181–200.

Franke, R. (1982b). Smooth interpolation of scattered data by local thin plate splines, *Comput. Math. Appl.* **8**, pp. 273–281.

Franke, R. (1985). Thin plate splines with tension, *Comput. Aided Geom. Design* **2**, pp. 87–95.

Franke, R. (1987). Recent advances in the approximation of surfaces from scattered data, in *Topics in Multivariate Approximation*, C. K. Chui, L. L. Schumaker, and F. Utreras (eds.), Academic Press (New York), pp. 79–98.

Franke, R. (1990). Approximation of scattered data for meteorological applications, in *Multivariate Approximation and Interpolation*, ISNM 94, W. Haussman and K. Jetter (eds.), Birkhäuser (Basel), pp. 107–120.

Franke, R. (1995). Paper presented at SIAM Geometric Modeling Conference in Nashville Nov. 1995.

Franke, R., Hagen, H. and Nielson, G. M. (1994). Least squares surface approximation to scattered data using multiquadric functions, *Adv. in Comput. Math.* **2**, pp. 81–99.

Franke, R., Hagen, H. and Nielson, G. M. (1995). Repeated knots in least squares multiquadric functions, in *Geometric Modelling — Dagstuhl 1993*, H. Hagen, G. Farin and H. Noltemeier (eds.), Springer Verlag (Berlin), pp. 177–187.

Franke, R. and Nielson, G. M. (1991). Scattered data interpolation and applications: A tutorial and survey, in *Geometric Modeling, Methods and Applications*, H. Hagen and D. Roller (eds.), Springer Verlag (Berlin), pp. 131–160.

Franke, R. and Šalkauskas, K. (1995). Localization of multivariate interpolation and smoothing methods, Paper presented at SIAM Geometric Modeling Conference, Nashville, 1995.

Freeden, W. (1982). Spline methods in geodetic approximation problems, *Math. Meth. Appl. Sci.* **4**, pp. 382–396.

Freeden, W. (1984). Spherical spline interpolation — basic theory and computational aspects, *J. Comput. Appl. Math.* **11**, pp. 367–375.

Freeden, W. (1987). A spline interpolation method for solving boundary value problems of potential theory from discretely given data, *Num. Meth. Part. Diff. Eq.* **3**, pp. 375–398.

Freeden, W., Gervens, T. and Schreiner, M. (1998). *Constructive Approximation on the Sphere*, Oxford University Press (Oxford).

Freeden, W., Schreiner, M. and Franke, R. (1997). A survey on spherical spline approximation, *Surv. Math. Ind.* **7**, pp. 29–85.

Freeman, J. A. and Saad, D. (1995). Learning and generalization in radial basis function networks, *Neural Computation* **7**, pp. 1000-1020.

Frigo, M. and Johnson, S. G. FFTW: The fastest Fourier transform in the West (C library), `http://www.fftw.org/`.

Galperin, E. A. and Zheng, Q. (1993). Solution and control of PDE via global optimization methods, *Comput. Math. Appl.* **25**, pp. 103–118.

Gáspár, C. (2004). A meshless polyharmonic-type boundary interpolation method for solving boundary integral equations, *Engineering Analysis with Boundary Elements* **28**, pp. 1207–1216.

Gasper, G. (1975). Positivity and special functions, in *Theory and Application of Special Functions*, R. Askey (ed.), Academic Press (New York), pp. 375–433.

Gel'fand, I. M. and Schilow, G. E. (1960). *Verallgemeinerte Funktionen, Vol. 1*, Deutscher Verlag der Wissenschaften (Leipzig).

Gel'fand, I. M. and Vilenkin, N. Ya. (1964). *Generalized Functions Vol. 4*, Academic Press (New York).

Ghorbany, M. and Soheili, A. R. (2004). Moving element free Petrov-Galerkin viscous method, *J. Chinese Inst. Engineers* **27**, pp. 473–479.

Girosi, F. (1992). Some extensions of radial basis functions and their applications in artificial intelligence, *Comput. Math. Appl.* **24**, pp. 61–80.

Girosi, F. (1998). An equivalence between sparse approximation and support vector machines, *Neural Computation* **10**, pp. 1455–1480.

Girosi, F. and Anzellotti, G. (1993). Rates of convergence for radial basis functions and neural networks, in *Artificial Networks for Speech and Vision*, R. J. Mammone (ed.), Chapman & Hall, London, 1993, pp. 97–114.

Girosi, F., Jones, M. and Poggio, T. (1993). Regularization theory and neural network architectures, MIT AI memo 1430.

Gneiting, T. (1999). Correlation functions for atmospheric data analysis, *Quart. J. Meteorol. Soc.* **125**, pp. 2449–2464.

Gneiting, T. (2002). Compactly supported correlation functions, *J. Multivariate Analysis* **83**, pp. 493–508.

Golberg, M. A. (1995). A note on the sparse representation of discrete integral operators, *Appl. Math. Comp.* **70**, pp. 97–118.

Golberg, M. A. (1996). Recent developments in the numerical evaluation of particular solutions in the boudary element method, *Appl. Math. Comp.* **75**, pp. 91–101.

Golberg, M. A. and Chen, C. S. (1994). The theory of radial basis functions applied to the BEM for inhomogeneous partial differential equations, *Boundary Elements Comm.* **5**, pp. 57–61.

Golberg, M. A. and Chen, C. S. (1996) A bibliography on radial basis function approximation, UNLV.

Golberg, M. A. and Chen, C. S. (1997). *Discrete Projection Methods for Integral Equations*, Computational Mechanics Publications (Southampton).

Golberg, M. A., Chen, C. S. and Bowman, H. (1999). Some recent results and proposals for the use of radial basis functions in the BEM, *Eng. Anal. with Bound. Elem.* **23**, pp. 285–296.

Golberg, M. A., Chen, C. S., Bowman, H. and Power, H. (1998). Some comments on the use of radial basis functions in the dual reciprocity method, *Comput. Mech.* **21**, pp. 141–148.

Golberg, M. A., Chen, C. S. and Karur, S. R. (1996). Improved multiquadric approximation for partial differential equations, *Eng. Anal. with Bound. Elem.* **18**, pp. 9–17.

von Golitschek, M. and Schumaker, L. L. (1990). Data fitting by penalized least squares, in *Algorithms for Approximation II*, M. G. Cox and J. C. Mason (eds.), Chapman & Hall (London), pp. 210–227.

Golomb, M. and Weinberger, H. F. (1959). Optimal approximation and error bounds, in *On Numerical Approximation*, R. E. Langer(ed.), University of Wisconsin Press, pp. 117–190.

Golub, G. H. and Van Loan, C. F. (1989). *Matrix Computations*, Johns Hopkins University Press (Baltimore).

Goodsell, G. (1997). A multigrid-type method for thin plate spline interpolation on a circle, *IMA J. Numer. Anal.* **17**, pp. 321–327.

Gottlieb, D. and Lustman, L. (1983). The spectrum of the Chebyshev collocation operator for the heat equation, *SIAM J. Numer. Anal.* **20** 5, pp. 909–921.

Gram, J. P. (1883). Über Entwicklung reeler Functionen in Reihen mittelst der Methode der kleinsten Quadrate, *J. Math.* 94, pp. 41–73.

Greengard, L. (1994). Fast algorithms for classical physics, *Science* **265**, pp. 909–914.

Greengard, L. and Rokhlin, V. (1987). A fast algorithm for particle simulations, *J. Comput. Phys.* **73** 2, pp. 325–348.

Greengard, L. and Strain, J. (1991). The fast Gauss transform, *SIAM J. Sci. Statist. Comput.* **12**, pp. 79–94.

Greengard, L. F. and Sun, X. (1998). A new version of the fast Gauss transform, *Doc. Math. J. DMV*, Extra Volume ICM, pp. 575–584.

Gu, C. and Wahba, G. (1993). Semiparametric analysis of variance with tensor product thin plate splines, *J. Roy. Statist. Soc. Ser. B* **55**, pp. 353–368.

Guo, K., Hu, S. and Sun, X. (1993a). Conditionally positive definite functions and Laplace-Stieltjes integrals, *J. Approx. Theory* **74**, pp. 249–265.

Guo, K., Hu, S. and Sun, X. (1993b). Cardinal interpolation using linear combinations of translates of conditionally positive definite functions, *Numer. Func. Anal. Optim.* 14, pp. 371–381.

Guo, K. and Sun, X. (1991). Scattered data interpolation by linear combinations of translates of conditionally positive definite functions, *Numer. Func. Anal. Optim.* **12**, pp. 137–152.

Gutzmer, T. (1996). Interpolation by positive definite functions on locally compact groups with application to SO(3), *Results Math.* **29**, pp. 69–77.

Hagan, R. E. (1993). Multiquadrics in engineering modeling, in *Proceedings ASCE National Conference on Irrigation and Drainage Engineering*, Park City, Utah.

Hagan, R. E. and Kansa, E. J. (1994). Studies of the R parameter in the multiquadric function applied to ground water pumping, *J. Appl. Sci. Comput.* 1, pp. 266–281.

Hales, S. J. and Levesley, J. (2000). On compactly supported, positive definite, radial basis functions, Technical Report 2000/30, University of Leicester, UK.

Hales, S. J. and Levesley, J. (2002). Error estimates for multilevel approximation using polyharmonic splines, *Numer. Algorithms* **30**, pp. 1–10.

Halgamuge, S. K., Poechmueller, W. and Glesner, M. (1995). An alternative approach for generation of membership functions and fuzzy rules based on radial and cubic basis function networks, *Internat. J. Approx. Reason.* **12**, pp. 279–298.

Halton, E. J. and Light, W. A. (1993). On local and controlled approximation order, *J. Approx. Theory* **72**, pp. 268–277.

Halton, J. H. (1960). On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals, *Numer. Math.* **2**, pp. 84–90.

Handscomb, D. (1993). Local recovery of a solenoidal vector field by an extension of the thin-plate spline technique, *Numer. Algorithms* 5 1–4, pp. 121–129.

Hangelbroek, T. (2006). Error estimates for thin plate spline approximation in the disc, *Constr. Approx.*, to appear.

Harder, R. L. and Desmarais, R. N. (1972). Interpolation using surface splines, *J. Aircraft* 9, pp. 189–191.

Hardy, R. L. (1971). Multiquadric equations of topography and other irregular surfaces, *J. Geophys. Res.* **76**, pp. 1905–1915.

Hardy, R. L. (1972). Geodetic applications of multiquadric analysis, *AVN Allg. Vermess. Nachr.* 79, pp. 389–406.

Hardy, R. L. (1975). Research results in the application of multiquadric equations to surveying and mapping problems, *Survg. Mapp.* **35**, pp. 321–332.

Hardy, R. L. (1977). Least squares prediction, *Photogramm. Eng. and Remote Sensing* **43**, pp. 475–492.

Hardy, R. L. (1990). Theory and applications of the multiquadric-biharmonic method, *Comput. Math. Appl.* 19, pp. 163–208.

Hardy, R. L. (1992a). Comments on the discovery and evolution of the multiquadric method, *Comput. Math. Appl.* **24**, pp. xi–xii.

Hardy, R. L. (1992b). A contribution of the multiquadric method: interpolation of potential inside the earth, *Comput. Math. Appl.* **24**, pp. 81–97.

Hardy, R. L. and Göpfert, W. M. (1975). Least squares prediction of gravity anomalies, geodial undulations, and deflections of the vertical with multiquadric harmonic functions, *Geophys. Res. Letters* **2**, pp. 423–426.

Hardy, R. L. and Nelson, S. A. (1986). A multiquadric-biharmonic representation and approximation of disturbing potential, *Geophys. Res. Letters* **13**, pp. 18–21.

Hardy, R. L. and Siranyone, S. (1989). The multiquadric-biharmonic method of a three

dimensional mapping inside ore deposits, in *Proceedings of Multinational Conference on Mine Planning and Design*, University of Kentucky.

Hartman, E. J., Keeler, J. D. and Kowalski, J. M. (1990). Layered neural networks with Gaussian hidden units as universal approximations, *Neural Computations* **2**, pp. 210–215.

Hartmann, S. (1998). Multilevel-Fehlerabschätzung bei der Interpolation mit radialen Basisfunktionen, Ph.D. Dissertation, Universität Göttingen.

Heiss, M. and Kampl, S. (1996). Multiplication-free radial basis function network, *IEEE Trans. Neural Networks* **7**, pp. 1461–1464.

Höllig, K. (2003). *Finite Element Methods With B-splines*, SIAM Frontiers in Applied Mathematics no. 26 (Philadelphia).

Hon, Y. C. (1999). Multiquadric collocation method with adaptive technique for problems with boundary layer, *Appl. Sci. Comput.* **6** 3, pp. 173–184.

Hon, Y. C. and Mao, X. Z. (1997). A multiquadric interpolation method for solving initial value problems, *J. Sci. Comput.* **12** 1, pp. 51–55.

Hon, Y. C. and Mao, X. Z. (1998). An efficient numerical scheme for Burgers' equation, *Appl. Math. Comput.* **95**, pp. 37–50.

Hon, Y. C. and Mao, X. Z. (1999). A radial basis function method for solving options pricing model, *Financial Engineering* **8**, pp. 31–49.

Hon, Y. C. and Schaback, R. (2001). On nonsymmetric collocation by radial basis functions, *Appl. Math. Comput.* **119**, pp. 177–186.

Hon, Y. C., Schaback, R. and Zhou, X. (2003). An adaptive greedy algorithm for solving large RBF collocation problems, *Numer. Algorithms* **32**, pp. 13–25.

Hon, Y. C. and Wu, Z. (2000). Additive Schwarz domain decomposition with a radial basis approximation, *Int. J. Appl. Math.* 4, pp. 81–98.

Horn, R. A. and Johnson, C. R. (1985). *Matrix Analysis*, Cambridge University Press (Cambridge).

Horn, R. A. and Johnson, C. R. (1991). *Topics in Matrix Analysis*, Cambridge University Press (Cambridge).

Hu, H.-Y., Li, Z.-C. and Cheng, A. H.-D. (2005). Radial basis collocation methods for elliptic boundary value problems, *Comput. Math. Appl.* **50**, pp. 289–320.

Hubbert, S. and Morton, T. M. (2004a). A Duchon framework for the sphere, *J. Approx. Theory* **129**, pp. 28–57.

Hubbert, S. and Morton, T. M. (2004b). L_p-error estimates for radial basis function interpolation on the sphere, *J. Approx. Theory* **129**, pp. 58–77.

Hunt, K. J., Haas, R. and Murray Smith, R. (1996). Extending the functional equivalence of radial basis function networks and fuzzy inference systems, *IEEE Trans. Neural Networks* **7**, pp. 776–781.

Hutchinson, M. F. (1991). The application of thin plate smoothing splines to continent-wide data assimilation, in *BMRC Research Report No.27, Data Assimilation Systems*, J. D. Jasper (ed.), Bureau of Meteorology (Melbourne), pp. 104–113.

Hutchinson, M. F. (1993). On thin plate splines and kriging, in *Computing and Science in Statistics 25*, M. E. Tarter and M. D. Lock (eds.), Interface Foundation of North America, University of California, Berkeley, pp. 55–62.

Hutchinson, M. F. (1995). Interpolating mean rainfall using thin plate smoothing splines, *Int. J. GIS* **9**, pp. 305–403.

Ingber, M. S., Chen, C. S. and Tanski, J. A. (2004). A mesh free approach using radial basis functions and parallel domain decomposition for solving three dimensional diffusion equations, *Int. J. Num. Meths. Engng.* **60**, pp. 2183–2201.

Iske, A. (1994). Charakterisierung bedingt positiv definiter Funktionen für multivariate

Interpolationsmethoden mit radial Basisfunktionen, Ph.D. Dissertation, Universität Göttingen.

Iske, A. (1995). Reconstruction of functions from generalized Hermite-Birkhoff data, in *Approximation Theory VIII, Vol. 1: Approximation and Interpolation*, C. Chui, and L. Schumaker (eds.), World Scientific Publishing (Singapore), pp. 257–264.

Iske, A. (1999a). Reconstruction of smooth signals from irregular samples by using radial basis function approximation, in *Proceeding of the 1999 International Workshop on Sampling Theory and Applications*, Y. Lyubaskii (ed.), The Norwegian University of Science and Technology (Trondheim), pp. 82–87.

Iske, A. (1999b). Perfect centre placement for radial basis function methods, Technical Report TUM M9809, Technische Universität Miinchen.

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

Iske, A. (2002). Multiresolution Methods in Scattered Data Modelling, Habilitation Thesis, Technische Universität Miinchen.

Iske, A. (2003a). On the approximation order and numerical stability of local Lagrange interpolation by polyharmonic splines, in *Modern Developments in Multivariate Approximation*, W. Haussmann, K. Jetter, M. Reimer, and J. Stöckler (eds.), ISNM 145, Birkhäuser Verlag (Basel), pp. 153–165.

Iske, A. (2003b). Radial basis functions: basics, advanced topics and meshfree methods for transport problems, *Rend. Sem. Mat. Univ. Pol. Torino* 61, pp. 247–285.

Iske, A. (2004). *Multiresolution Methods in Scattered Data Modelling*, Lecture Notes in Computational Science and Engineering 37, Springer Verlag (Berlin).

Iske, A. and Sonar, T. (1996). On the structure of function spaces in optimal recovery of point functionals for ENO-schemes by radial basis functions, *Numer. Math.* **74**, pp. 177–201.

Ivanov, T., Maz'ya, V. and Schmidt, G. (1999). Boundary layer approximate approximations and cubature of potentials in domains, *Adv. in Comput. Math.* **10**, pp. 311–342.

Jackson, I. R. H. (1987). Approximation to continuous functions using radial basis functions, in *The Mathematics of Surfaces II*, R. R. Martin (ed.), Clarendon Press (Oxford), pp. 115–135.

Jackson, I. R. H. (1988a). Radial basis function methods from multivariate approximation, Ph.D. Dissertation, University of Cambridge.

Jackson, I. R. H. (1988b). Convergence properties of radial basis functions, *Constr. Approx.* 4, pp. 243–264.

Jackson, I. R. H. (1989a). An order of convergence for some radial basis functions, *IMA J. Numer. Anal.* **9**, pp. 567–587.

Jackson, I. R. H. (1989b). Radial basis functions — a survey and new results, in *Mathematics of Surfaces III*, D. C. Handscomb (ed.), Clarendon Press (Oxford), pp. 115–133.

Jekeli, C. (1994). Hardy's multiquadric-biharmonic method for gravity field predictions, *Comput. Math. Appl.* **28**, pp. 43–46.

Jetter, K. (1993a). Multivariate approximation from the cardinal interpolation point of view, in *Approximation Theory VII*, E. W. Cheney, C. Chui, and L. Schumaker (eds.), Academic Press (New York), pp. 131–161.

Jetter, K. (1993b). Riesz bounds in scattered data interpolation and L_2-approximation, in *Multivariate Approximation: From CAGD to Wavelets*, Kurt Jetter and Florencio Utreras (eds.), World Scientific Publishing (Singapore), pp. 167–177.

Jetter, K. (1994). Conditionally lower Riesz bounds in scattered data interpolation, in *Wavelets, Images, and Surface Fitting*, P.-J. Laurent, A. Le Méhauté, and L. L. Schumaker (eds.), A. K. Peters (Wellesley, MA), pp. 295–302.

Jetter, K. and Stöckier, J. (1991). Algorithms for cardinal interpolation using box splines and radial basis functions, *Numer. Math.* **60**, pp. 97–114.

Jetter, K. and Stöckier, J. (1997). Topics in scattered data interpolation and non-uniform sampling, in *Surface Fitting and Multiresolution*, A. Le Méhauté, C. Rabut, and L. L. Schumaker (eds.), Vanderbilt University Press, pp. 191–208.

Jia, R. Q. and Lei, J. J. (1991). Approximation by multiinteger translates of functions having global support, *J. Approx. Theory* **72**, pp. 2–23.

Jia, R. Q. and Lei, J. J. (1993). A new version of the Strang-Fix conditions, *J. Approx. Theory* **74**, pp. 221-225.

Johnson, M. J. (1998). A bound on the approximation order of surface splines, *Constr. Approx.* **14** 3, pp. 429–438.

Johnson, M. J. (2000a). Overcoming the boundary effects in surface spline interpolation, *IMA J. Numer. Anal.* **20** 3, pp. 405–422.

Johnson, M. J. (2000b). Approximation in $L_p(R^d)$ from spaces spanned by the perturbed integer translates of a radial function, *J. Approx. Theory* **107** 2, pp. 163–203.

Johnson, M. J. (2000c). An improved order of approximation for thin-plate spline interpolation in the unit disc, *Numer. Math.* **84** 3, pp. 451–474.

Johnson, M. J. (2001a). On the error in surface spline interpolation of a compactly supported function, *Kuwait J. Sci. Engrg.* **28** 1, pp. 37–54.

Johnson, M. J. (2001b). The $L_2$2-approximation order of surface spline interpolation, *Math. Comp.* **70** 234, pp. 719–737.

Johnson, M. J. (2004a). An error analysis for radial basis function interpolation, *Numer. Math.* **98** 4, pp. 675–694.

Johnson, M. J. (2004b). The L_p-approximation order of surface spline interpolation for $1 \leq p \leq 2$, *Constr. Approx.* **20** 2, pp. 303–324.

Johnson, M. J. (2006). A note on the limited stability of surface spline interpolation, *J. Approx. Theory* **141** 2, pp. 182–188.

Jumarhon, B., Amini, S. and Chen, K. (2000). The Hermite collocation method using radial basis functions, *Engineering Analysis with Boundary Elements* **24** 7–8, pp. 607–611.

Kaczmarz, S. (1937). Approximate solution of systems of linear equations (originally published as: Angenäherte Auflösung von Systemen linearer Gleichungen, Bulletin International de l'Academie Polonaise des Sciences, Lett. A, 1937, pp. 355–357), *Int. J. Control* **57**, pp. 1269–1271.

Kansa, E. J. (1986). Application of Hardy's multiquadric interpolation to hydrodynamics, *Proc. 1986 Simul. Conf.* **4**, pp. 111–117.

Kansa, E. J. (1990a). Multiquadrics — A scattered data approximation scheme with applications to computational fluid-dynamics — I: Surface approximations and partial derivative estimates, *Comput. Math. Appl.* **19**, pp. 127–145.

Kansa, E. J. (1990b). Multiquadrics — A scattered data approximation scheme with applications to computational fluid-dynamics — II: Solutions to parabolic, hyperbolic and elliptic partial differential equations, *Comput. Math. Appl.* **19**, pp. 147–161.

Kansa, E. J. (1992). A strictly conservative spatial approximation scheme for the governing engineering and physics equations over irregular regions and inhomogeneous scattered nodes, *Comput. Math. Appl.* **24**, pp. 169–190.

Kansa, E. J. and Carlson, R. E. (1992). Improved accuracy of multiquadric interpolation using variable shape parameters, *Comput. Math. Appl.* **24**, pp. 99–120.

Kansa, E. J. and Carlson, R. E. (1995). Radial basis functions: a class of grid free scattered data approximations, *Computational Fluid Dynamics J.* **3**, pp. 479–496.

Kansa, E. J., Fasshauer, G. E., Power, H. and Ling, L. (2004). A volumetric integral radial basis function method for time-dependent partial differential equations: I. Formulation, *Engineering Analysis with Boundary Elements* **28**, pp. 1191–1206.

Kansa, E. J. and Hon, Y. C. (2000). Circumventing the ill-conditioning problem with multiquadric radial basis functions: Applications to elliptic partial differential equations, *Comput. Math. Applic.* **39**, pp. 123–137.

Karlin, S. (1968). *Total Positivity*, Stanford University Press (Stanford).

Karur, S. R. and Ramachandran, P. A. (1994). Radial basis function approximation in the dual reciprocity method, *Math. Comput. Modell.* **20**, pp. 59–70.

Karur, S. R. and Ramachandran, P. A. (1995). Augmented thin plate spline approximation in DRM, *Boundary Elements Comm.* **6**, pp. 55–58.

Kent, J. T. and Mardia, K. V. (1994). The link between kriging and thin-plate splines, *Probability, statistics and optimisation*, *Wiley Ser. Probab. Math. Statist.*, Wiley (Chichester), pp. 325–339.

Kimeldorf, G. and Wahba, G. (1971). Some results on Tchebycheffian spline functions, *J. Math. Anal. Applic.* **33**, pp. 82–95.

Kincaid, D. and Cheney, W. (2002). *Numerical Analysis: Mathematics of Scientific Computing* (3rd ed.), Brooks/Cole (Pacific Grove, CA).

Komargodsky, Z. and Levin, D. (2006). Hermite type moving-least-squares approximation, *Comput. Math. Appl.* **51** 8, pp. 1223–1232.

Kounchev, O. (2001). *Multivariate Polysplines: Applications to Numerical and Wavelet Analysis*, Academic Press (New York).

Krzyzak, A., Linder, T. and Lugosi, G. (1996). Nonparametric estimation and classification using radial basis function nets and empirical risk minimization, *IEEE Trans. Neural Networks* 7, pp. 475–487.

Kunis, S. and Potts, D. (2002). NFFT, Softwarepackage (C-library), Universität Lübeck, `http://www.math.uni-luebeck.de/potts/nfft/`.

Kunis, S., Potts, D. and Steidl, G. (2002). Fast Fourier transforms at nonequispaced knots: A user's guide to a C-library, Universität Lübeck, `http://www.math.uni-luebeck.de/potts/nfft/`.

Kurdila, A. J., Narcowich, F. J. and Ward, J. D. (1995). Persistency of excitation in identification using radial basis function approximants, *SIAM J. Control Optim.* **33**, pp. 625–642.

Kybic, J., Blu, T. and Unser, M. (2002a). Generalized sampling: A variational approach — Part I: Theory, *IEEE Trans. Signal Proc.* **50**, pp. 1965–1976.

Kybic, J., Blu, T. and Unser, M. (2002b). Generalized sampling: A variational approach — Part II: Applications, *IEEE Trans. Signal Proc.* **50**, pp. 1977–1985.

Lancaster, P. and Šalkauskas, K. (1981). Surfaces generated by moving least squares methods, *Math. Comp.* **37**, pp. 141–158.

Lancaster, P. and Šalkauskas, K. (1986). *Curve and Surface Fitting*, Academic Press (New York).

Lanzara, F., Maz'ya, V. and Schmidt, G. (2006). Approximate Approximations from scattered data, *J. Approx. Theory*, in press.

La Rocca, A., Hernandez Rosales, A. and Power, H. (2005). Radial basis function Hermite collocation approach for the solution of time dependent convection-diffusion problems, *Engineering Analysis with Boundary Elements* 29, pp. 359–370.

Larsson, E. and Fornberg, B. (2003). A numerical study of some radial basis function based solution methods for elliptic PDEs, *Comput. Math. Appl.* **46**, pp. 891–902.

Larsson, E. and Fornberg, B. (2005). Theoretical and computational aspects of multivariate interpolation with increasingly flat radial basis functions, *Comput. Math. Appl.* 49, pp. 103–130.

Lee, D. and Shiau, J. H. (1994). Thin plate splines with discontinuities and fast algorithms for their computation, *SIAM J. Scient. Computing* **15**, pp. 1311-1330.

Lee, S. and Pan, J. C. J. (1996). Unconstrained handwritten numeral recognition based on radial basis competitive and cooperative networks with spatiotemporal feature representation, *IEEE Trans. Neural Networks* **7**, pp. 455–474.

Lee, Y. (1991). Handwritten digit recognition using K nearest-neighbor, radial-basis function, and backpropagation neural networks, *Neural Computation* **3**, pp. 440–449.

Le Gia, Q. T. (2004). Galerkin approximation for elliptic PDEs on spheres, *J. Approx. Theory* **130**, pp. 125–149.

Leitão, V. M. A. (2001). A meshless method for Kirchhoff plate bending problems, *Int. J. Numer. Meth. Engng.* **52**, pp. 1107–1130.

Leitão, V. M. A. (2004). RBF-based meshless methods for 2D elastostatic problems, *Engineering Analysis with Boundary Elements* **28**, pp. 1271–1281.

Leitão, V. M. A. (2006). Application of radial basis functions to linear and non-linear structural analysis problems, *Comput. Math. Appl.* **51** 8, pp. 1311–1334.

Le Méhauté, A. (1995). Knot removal for scattered data, in *Approximation Theory, Wavelets and Applications*, S. P. Singh (ed.), Kluwer (Dordrecht), pp. 197–213.

Le Méhauté, A. (1999). Inf-Convolution and radial basis functions, in *New Developments in Approximation Theory*, M. W. Miiller, M. D. Buhmann, D. H. Mache and M. Felten (eds.), Birkhäuser (Basel), pp. 95–108.

Leonard, J. A., Kramer, M. A. and Ungar, J. H. (1992). Using radial basis functions to approximate a function and its error bounds, *IEEE Trans. on Neural Networks* **3**, pp. 624–627.

Levesley, J. (1994). Convolution kernels for approximation by radial basis functions, in *Wavelets, Images, and Surface Fitting*, P.-J. Laurent, A. Le Méhauté, and L. L. Schumaker (eds.), A. K. Peters (Wellesley, MA), pp. 343–350.

Levesley, J. (1995a). Pointwise estimates for multivariate interpolation using conditionally positive definite functions, in *Approximation Theory, Wavelets and Applications*, S. P. Singh (ed.), Kluwer (Dordrecht), pp. 381–401.

Levesley, J. (1995b). Convolution kernels based on thin plate splines, *Numer. Algorithms* **10**, pp. 401–419.

Levesley, J. and Kushpel, A. K. (1995). Interpolation on compact Abelian groups using generalised sk-splies, in *Approximation Theory VIII, Vol. 1: Approximation and Interpolation*, C. Chui, and L. Schumaker (eds.), World Scientific Publishing (Singapore), pp. 317–324.

Levesley, J. and Kushpel, A. K. (1999). Generalised sk-spline interpolation on compact Abelian groups, *J. Approx. Theory* **97** 2, 311–333.

Levesley, J., Light, W., Ragozin, D. and Sun, X. (1999). A simple approach to the variational theory for interpolation on spheres, in *New Developments in Approximation Theory*, M. W. Miiller, M. D. Buhmann, D. H. Mache and M. Felten (eds.), Birkhäuser (Basel), pp. 117-144.

Levesley, J. and Ragozin, D. (2002). Positive definite kernel interpolation on manifolds: convergence rates, in *Approximation Theory X: Abstract and Classical Analysis*, C. K. Chui, L. L. Schumaker, and J. Stöckier (eds.), Vanderbilt University Press, Nashville, 2002, pp. 277–285.

Levesley, J. and Roach, M. (1995). Quasi-interpolation on compact domains, in *Approximation Theory, Wavelets and Applications*, S. P. Singh (ed.), Kluwer (Dordrecht), pp. 557–566.

Levesley, J. and Sun, X. (1995). Scattered Hermite interpolation by ridge functions, *Numer. Func. Anal. Optim.* **16**, pp. 989–1001.

Levesley, J. and Sun, X. (2005). Approximation in rough native spaces by shifts of smooth kernels on spheres, *J. Approx. Theory* **133**, pp. 269–283.

Levesley, J. and Sun, X. (2006). Corrigendum to and two open questions arising from the article "Approximation in rough native spaces by shifts of smooth kernels on spheres" [J. Approx. Theory 133 (2005) 269–283], *J. Approx. Theory* **138** 1, pp. 124–127.

Levesley, J., Xu, Y., Light, W. and Cheney, W. (1996). Convolution operators for radial basis approximation, *SIAM J. Math. Anal.* **27**, pp. 286–304.

Levin, D. (1998). The approximation power of moving least-squares, *Math. Comp.* **67**, pp. 1517–1531.

Li, J. (2005). Mixed methods for fourth-order elliptic and parabolic problems using radial basis functions, *Adv. in Comput. Math.* **23**, pp. 21–30.

Li, J., Cheng, A. H.-D. and Chen, C-S. (2003). On the efficiency and exponential convergence of multiquadric collocation method compared to finite element method, *Engineering Analysis with Boundary Elements* **27** 3, pp. 251–257.

Li, J. and Hon, Y. C. (2004). Domain decomposition for radial basis meshless methods, *Numerical Methods for Partial Differential Equations* **20** 3, pp. 450–462.

Li, S. and Liu, W. K. (1996). Moving least-square reproducing kernel method Part II: Fourier analysis, *Comp. Meth. Appl. Mech. Eng.* **139**, pp. 159–193.

Li, S. and Liu, W. K. (2002). Meshfree and particle methods and their applications, *Applied Mechanics Review* **55**, pp. 1–34.

Li, X. (1998). On simultaneous approximations by radial basis function neural networks, *Appl. Math. Comput.* **95**, pp. 75–89.

Li, X. and Chen, C. S. (2004). A mesh free method using hyperinterpolation and fast Fourier transform for solving differential equations, *Engineering Analysis with Boundary Elements* **28**, pp. 1253–1260.

Li, X., Ho, C. H. and Chen, C. S. (2002). Computational test of approximation of functions and their derivatives by radial basis functions, *Neural Parallel Sci. Comput.* **10**, pp. 25–46.

Li, X. and Micchelli, C. A. (2000). Approximation by radial bases and neural networks, *Numer. Algorithms* **25**, pp. 241–262.

Lindsay, K. and Krasny, R. (2001). A particle method and adaptive treecode for vortex sheet motion in three-dimensional flow, *J. Comput. Physics* **172**, pp. 879–907.

Light, W. A. (1991). Recent developments in the Strang-Fix theory for approximation orders, in *Curves and Surfaces*, P.-J. Laurent, A. Le Méhauté, and L. L. Schumaker (eds.), Academic Press (New York), pp. 285–292.

Light, W. A. (1992). Some aspects of radial basis function approximation, in *Approximation Theory, Spline Functions and Applications*, S. P. Singh (ed.), Kluwer (Dordrecht), pp. 163–190.

Light, W. A. (1994). Using radial functions on compact domains, in *Wavelets, Images, and Surface Fitting*, P.-J. Laurent, A. Le Méhauté, and L. L. Schumaker (eds.), A. K. Peters (Wellesley, MA), pp. 351–370.

Light, W. A. (1996). Variational error bounds for radial basis functions, in *Numerical Analysis 1995*, D. F. Griffiths and G. A. Watson (eds.), Longman (Harlow), pp. 94–106.

Light, W. A. and Cheney, E. W. (1991). Interpolation by piecewise-linear radial basis functions, II, *J. Approx. Theory* **64**, pp. 38–54.

Light, W. A. and Cheney, E. W. (1992a). Interpolation by periodic radial basis functions, *J. Math. Anal. Appl.* **168**, pp. 111–130.

Light, W. A. and Cheney, E. W. (1992b). Quasi-interpolation with translates of a function having noncompact support, *Constr. Approx.* **8**, pp. 35–48.

Light, W. A. and Wayne, H. (1995). Error estimates for approximation by radial basis

functions, in *Approximation Theory, Wavelets and Applications*, S. P. Singh (ed.), Kluwer (Dordrecht), pp. 215–246.

Light, W. A. and Wayne, H. (1998). On power functions and error estimates for radial basis function interpolation, *J. Approx. Theory* **92**, pp. 245–266.

Light, W. A. and Wayne, H. (1999). Spaces of distributions and interpolation by translates of a basis function, *Numer. Math.* **81**, pp. 415–450.

Ling, L. and Hon, Y. C. (2005). Improved numerical solver for Kansa's method based on affine space decomposition, *Engineering Analysis with Boundary Elements* **29**, pp. 1077–1085.

Ling, L. and Kansa, E. J. (2004). Preconditioning for radial basis functions with domain decomposition methods, *Math. and Comput. Modelling* 40, pp. 1413–1427.

Ling, L. and Kansa, E. J. (2005). A least-squares preconditioner for radial basis functions collocation methods, *Adv. in Comput. Math.* **23**, pp. 31–54.

Ling, L., Opfer, R. and Schaback, R. (2006). Results on meshless collocation techniques, *Engineering Analysis with Boundary Elements* **30**, pp. 247–253.

Liu, G. R. (2002). *Mesh Free Methods: Moving beyond the Finite Element Method*, CRC Press (Boca Raton, FL).

Liu, W. K., Chen, Y., Aziz Uras, R. and Chang, C. T. (1996). Generalized multiple scale reproducing kernel particle methods, *Comp. Meth. Appl. Mech. Eng.* **139**, pp. 91–157.

Liu, W. K., Li, S. and Belytschko, T. (1997). Moving least square reproducing kernel methods (I) Methodology and convergence, *Comp. Meth. Appl. Mech. Eng.* **143**, pp. 113–154.

Locher, F. (1981). Interpolation on uniform meshes by the translates of one function and related attenuation factors, *Math. Comp.* **37**, pp. 403–416.

Lorentz, R., Narcowich, F. J. and Ward, J. D. (2003). Collocation discretizations of the transport equation with radial basis functions, *Applied Mathematics and Computation* **145** 1, pp. 97–116.

Lowe, D. (1993). Novel 'topographic' nonlinear feature extraction using radial basis functions for concentration coding in the 'artificial nose', in *3rd IEE International Conference on Artificial Neural Networks*, IEE (London).

Lowe, D. and Matthews, R. (1995). Shakespeare vs. Fletcher: a stylistic analysis by radial basis functions, *Computers and Humanities* **29**, pp. 449–461.

Lowe, D. and McLachlan, A. (1995). Modelling of nonstationary processes using radial basis function networks, in *Artificial Neural Networks, 1995*, 300–305.

Lowe, D. and Tipping, M. E. (1995). A novel neural network technique for exploratory data analysis, in *Proceedings of ICANN '95*, Vol. 1, EC2 & Cie (Paris), pp. 339–344.

Lowe, D. and Tipping, M. E. (1996). Feed-forward neural networks and topographic mappings for exploratory data analysis, *Neural Computing and Applications* 4, pp. 83-95.

Lowitzsch, S. (2002). Approximation and Interpolation Employing Divergence-Free Radial Basis Functions with Applications, Ph.D. Dissertation, Texas A&M University.

Lowitzsch, S. (2005). Matrix-valued radial basis functions: stability estimates and applications, *Adv. in Comput. Math.* **23** 3, pp. 299–315.

Lukacs, E. (1970). *Characteristic Functions*, Griffin (London).

Lyche, T. (1992). Knot Removal for Spline Curves and Surfaces, in *Approximation Theory VII*, E. W. Cheney, C. Chui, and L. Schumaker (eds.), Academic Press (New York), pp. 207–226.

Madych, W. R. (1989). Cardinal interpolation with polyharmonic splines, in *Multivariate Approximation Theory IV*, ISNM 90, C. Chui, W. Schempp, and K. Zeller (eds.), Birkhäuser Verlag (Basel), pp. 241–248.

Madych, W. R. (1990). Polyharmonic splines, multiscale analysis, and entire functions, in *Multivariate Approximation and Interpolation*, ISNM 94, W. Haussman and K. Jetter (eds.), Birkhäuser (Basel), pp. 205–216.

Madych, W. R. (1991) Error estimates for interpolation by generalized splines, in *Curves and Surfaces*, P.-J. Laurent, A. Le Méhauté, and L. L. Schumaker (eds.), Academic Press (New York), pp. 297–306.

Madych, W. R. (1992). Miscellaneous error bounds for multiquadric and related interpolators, *Comput. Math. Appl.* **24**, pp. 121–138.

Madych, W. R. and Nelson, S. A. (1983). Multivariate interpolation: a variational theory, manuscript.

Madych, W. R. and Nelson, S. A. (1988). Multivariate interpolation and conditionally positive definite functions, *Approx. Theory Appl.* 4, pp. 77–89.

Madych, W. R. and Nelson, S. A. (1989). Error bounds for multiquadric interpolation, in *Approximation Theory VI*, C. Chui, L. Schumaker, and J. Ward (eds.), Academic Press (New York), pp. 413–416.

Madych, W. R. and Nelson, S. A. (1990a). Multivariate interpolation and conditionally positive definite functions, II, *Math. Comp.* **54**, pp. 211–230.

Madych, W. R. and Nelson, S. A. (1990b). Polyharmonic cardinal splines, *J. Approx. Theory* **60**, pp. 141–156.

Madych, W. R. and Nelson, S. A. (1990c). Polyharmonic cardinal splines: A minimization property, *J. Approx. Theory* **63**, pp. 303–320.

Madych, W. R. and Nelson, S. A. (1992). Bounds on multivariate polynomials and exponential error estimates for multiquadric interpolation, *J. Approx. Theory* **70**, pp. 94–114.

Mai-Duy, N. and Tran-Cong, T. (2001a). Numerical solution of differential equations using multiquadric radial basis function networks, *Neural Networks* 14, pp. 185–199.

Mai-Duy, N. and Tran-Cong, T. (2001b). Numerical solution of Navier-Stokes equations using radial basis function networks, *Int. J. Numer. Meth. Fluids* **37**, pp. 65–86.

Maiorov, V. (2005). On lower bounds in radial basis approximation, *Adv. in Comp. Math.* **22**, pp. 103–113.

Mairhuber, J. C. (1956). On Haar's theorem concerning Chebyshev approximation problems having unique solutions, *Proc. Am. Math. Soc.* **7**, pp. 609–615.

Makroglu, A. (1992). Radial basis functions in the numerical solution of Fredholm and integro-differential equations, in *Advances in Computer Methods for Partial Differential Equations – VII*, R. Vichnevetsky, D. Knight, and G. Richter (eds.), IMACS (New Brunswick), pp. 478–484.

Makroglu, A. (1994a). Radial basis functions in the numerical solution of nonlinear Volterra integral equations, *J. Appl. Sci. Comput.* 1, pp. 33–53.

Makroglu, A. (1994b). Multiquadric collocation methods in the numerical solution of Volterra integral and integro-differential equations, in *Mathematics of Computation 1943–1993*, Proc. Sympos. Appl. Math., 48, Amer. Math. Soc. (Providence, RI), pp. 337–341.

Makroglu, A. and Kansa, E. J. (1993). Multiquadric collocation methods in the numerical solution of Volterra integral equations with weakly singular kernels, Lawrence Livermore National Laboratory.

Marcozzi, M. D., Choi, S. and Chen, C. S. (1999). RBF and optimal stopping problems: an application to the pricing of vanilla options on one risky asset, in *Boundary Element Technology XIII*, C. S. Chen, C. A. Brebbia, and D. W. Pepper (eds.), WIT Press, pp. 345–354.

Marcus, M. and Mine, H. (1965). *Introduction to Linear Algebra*, Dover Publications (New York), republished 1988.

Matérn, B. (1986). *Spatial variation* (Second ed.), Lecture Notes in Statistics, 36, Springer-Verlag (Berlin).

Matheron, G. (1965). *Les variables régionalisées et leur estimation*, Masson (Paris).

Matheron, G. (1973). The intrinsic random functions and their applications, *Adv. Appl. Prob.* **5**, pp. 439–468.

Mathias, M. (1923). Über positive Fourier-Integrale, *Math. Zeit.* **16**, pp. 103–125.

MATLAB Central File Exchange, available online at `http://www.mathworks.com/-matlabcentral/fileexchange/`.

Maz'ya, V. (1991). A new approximation method and its applications to the calculation of volume potentials. Boundary point method, in *DFG-Kolloquium des DFG-Forschungsschwerpunktes "Randelementmethoden"*.

Maz'ya, V. (1994). Approximate approximations, in *The Mathematics of Finite Elements and Applications*, J. R. Whiteman (ed.), Wiley (Chichester), pp. 77–104.

Maz'ya, V. and Schmidt, G. (1996). On approximate approximations using Gaussian kernels, *IMA J. Numer. Anal.* **16**, pp. 13–29.

Maz'ya, V. and Schmidt, G. (2001). On quasi-interpolation with non-uniformly distributed centers on domains and manifolds, *J. Approx. Theory* **110**, pp. 125–145.

McCormick, S. F. (1992). *Multilevel Projection Methods for Partial Differential Equations*, CBMS-NSF Regional Conference Series in Applied Mathematics 62, SIAM (Philadelphia).

McLain, D. H. (1974). Drawing contours from arbitrary data points, *Comput. J.* **17**, pp. 318–324.

McMahon, J. (1986). Knot selection for least squares approximation using thin plate splines, M.S. Thesis, Naval Postgraduate School.

McMahon, J. and Franke, R. (1992). Knot selection for least squares thin plate splines, *SIAM J. Sci. Statist. Comput.* **13**, pp. 484–498.

Meinguet, J. (1979a). Multivariate interpolation at arbitrary points made simple, *Z. Angew. Math. Phys.* **30**, pp. 292–304.

Meinguet, J. (1979b). An intrinsic approach to multivariate spline interpolation at arbitrary points, in *Polynomial and Spline Approximations*, N. B. Sahney (ed.), Reidel (Dordrecht), pp. 163–190.

Meinguet, J. (1979c). Basic mathematical aspects of surface spline interpolation, in *Numerische Integration*, G. Hämmerlin (ed.), Birkhäuser (Basel), pp. 211–220.

Meinguet, J. (1984). Surface spline interpolation: basic theory and computational aspects, in *Approximation Theory and Spline Functions*, S. P. Singh, J. H. W. Burry, and B. Watson (eds.), Reidel (Dordrecht), pp. 127–142.

Melenk, J. M. and Babuška, I. (1996). The partition of unity finite element method: basic theory and applications, *Comput. Methods Appl. Mech. Engrg.* **139**, pp. 289–314.

Menegatto, V. A. (1992). Interpolation on Spherical Spaces, Ph.D. Dissertation, University of Texas Austin.

Menegatto, V. A. (1994a). Interpolation on spherical domains, *Analysis* **14**, pp. 415–424.

Menegatto, V. A. (1994b). Strictly positive definite kernels on the Hilbert sphere, *Appl. Anal.* **55**, pp. 91–101.

Menegatto, V. A. (1995a). Interpolation on the complex Hilbert sphere, *Approx. Theory Appl.* **11**, pp. 1–9.

Menegatto, V. A. (1995b). Strictly positive definite kernels on the circle, *Rocky Mountain J. Math.* **25**, pp. 1149–1163.

Menegatto, V. A. (1997). Interpolation on the complex Hilbert sphere using positive definite and conditionally negative definite functions, *Acta Mathematica Hungarica* **75** 3, pp. 215–225.

Menegatto, V. A., Oliveira, C. P. and Peron, A. P. (2006) Strictly positive definite kernels on subsets of the complex plane, *Comput. Math. Appl.* **51** 8, pp. 1233–1250.

Menegatto, V. A. and Peron, A. P. (2004). Conditionally positive definite kernels on Euclidean domains, *J. Math. Anal. Appl.* **294** 1, pp. 345–359.

Menger, K. (1928). Die Metrik des Hilbertschen Raumes, *Anzeiger der Akad. der Wissenschaften in Wien*, Nat. Kl. **65**, pp. 159–160.

Meyer, C. D. (2000). *Matrix Analysis and Applied Linear Algebra*, SIAM (Philadelphia).

Mhaskar, H. N. (1990). Weighted polynomials, radial basis functions and potentials on locally compact spaces, *Numer. Func. Anal. Optim.* 11, pp. 987–1017.

Mhaskar, H. N. and Micchelli, C. A. (1992). Approximation by superposition of sigmoidal and radial basis functions, *Adv. in Appl. Math.* **13**, pp. 350–373.

Micchelli, C. A. (1986). Interpolation of scattered data: distance matrices and conditionally positive definite functions, *Constr. Approx.* **2**, pp. 11–22.

Micchelli, C. A. and Rivlin, T. J. (1977). A survey of optimal recovery, in *Optimal Estimation in Approximation Theory (Proc. Internat. Sympos., Freudenstadt, 1976)*, C. A. Micchelli and T. J. Rivlin (eds.), Plenum Press (New York), pp. 1–54.

Micchelli, C. A. and Rivlin, T. J. (1980). Optimal recovery of best approximations, *Resultate Math.* 3 1, pp. 25–32.

Micchelli, C. A. and Rivlin, T. J. (1985). Lectures on optimal recovery, in *Numerical analysis, Lancaster 1984*, P. R. Turner (ed.), Lecture Notes in Math., 1129, Springer (Berlin), pp. 21–93.

Micchelli, C. A., Rivlin, T. J. and Winograd, S. (1976). The optimal recovery of smooth functions, *Numer. Math.* **26** 2, pp. 191–200.

Miranda, J. (2004). Incorporating R-functions into the Theory of Positive Definite Functions to Solve Elliptic Partial Differential Equations, Ph.D. Dissertation, Illinois Institute of Technology.

Monaghan, J. J. (1988). An introduction to SPH, *Comp. Phys. Comm.* **48**, pp. 89–96.

Montès, P. (1991). Local kriging interpolation: Application to scattered data on the sphere, in *Curves and Surfaces*, P.-J. Laurent, A. Le Méhauté, and L. L. Schumaker (eds.), Academic Press (New York), pp. 325–329.

Montès, P. (1994). Smoothing noisy data by kriging with nugget effects, in *Wavelets, Images, and Surface Fitting*, P.-J. Laurent, A. Le Méhauté, and L. L. Schumaker (eds.), A. K. Peters (Wellesley, MA), pp. 371–378.

Moody, J. and Darken, C. J. (1989). Fast learning in networks of locally-tunes processing units, *Neural Computations* 1, pp. 281–294.

Moridis, G. J. and Kansa, E. J. (1992). The method of Laplace transform multiquadrics for the solution of the groundwater flow equation, in *Advances in Computer Methods for Partial Differential Equations – VII*, R. Vichnevetsky, D. Knight, and G. Richter (eds.), IMACS (New Brunswick), pp. 539–545.

Moridis, G. J. and Kansa, E. J. (1994). The Laplace transform multiquadric method: A highly accurate scheme for the numerical solution of linear partial differential equations, *J. Appl. Sc. Comp.* 1, pp. 375–407.

Morse, B., Yoo, T. S., Rheingans, P., Chen, D. T. and Subramanian, K. R. (2001). Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions, in *2001 International Conference on Shape Modeling and Applications (SMI 2001)*, IEEE Computer Society Press (Los Alamitos, CA), pp. 89–98.

Morton, T. M. (2004). Two approaches to solving pseudodifferential equations on spheres using zonal kernels, *Engineering Analysis with Boundary Elements* **28**, pp. 1227–1232.

Mouat, C. T. (2001). Fast algorithms and preconditioning techniques for fitting radial basis functions, Ph.D. Dissertation, University of Canterbury, NZ.

Miiller, C. (1966). *Spherical Harmonics*, Springer Lecture Notes in Mathematics (Vol. 17).

Mulgrew, B. (1996). Applying radial basis functions, *IEEE Signal Proc. Magazine* **13**, pp. 50–65.

Musavi, M. T., Ahmed, W., Chan, K. H., Faris, K. B. and Hummels, D. M. (1992). On the training of radial basis function classifiers, *Neural Networks* **5**, pp. 595–603.

Muttiah, R., Hutchinson, M. and Dyke, P. (1995). Climate surfaces of Texas using thin plate splines, in *Approximation Theory VIII, Vol. 1: Approximation and Interpolation*, C. Chui, and L. Schumaker (eds.), World Scientific Publishing (Singapore), pp. 427–434.

Myers, D. E. (1988). Interpolation with positive definite functions, *Sciences de la Terre* **28**, pp. 252–265.

Myers, D. E. (1992). Kriging, co-Kriging, radial basis functions and the role of positive definiteness, *Comput. Math. Appl.* 24, pp. 139–148.

Myers, D. E. (1993). Selection of a radial basis function for interpolation, in *Advances in Computer Methods for Partial Differential Equations – VII*, R. Vichnevetsky, D. Knight, and G. Richter (eds.), IMACS (New Brunswick), pp. 553–558.

Nadaraya, E. A. (1964). On estimating regression, *Theor. Probab. Appl.* **9**, pp. 141–142.

Narcowich, F. J. (1995). Generalized Hermite interpolation and positive definite kernels on a Riemannian manifold, *J. Math. Anal. Appl.* **190**, pp. 165–193.

Narcowich, F. J., Schaback, R. and Ward, J. D. (1999). Multilevel interpolation and approximation, *Appl. Comput. Harmon. Anal.* **7**, pp. 243–261.

Narcowich, F. J., Sivakumar, N. and Ward, J. D. (1994). On condition numbers associated with radial-function interpolation, *J. Math. Anal. Appl.* **186**, pp. 457–485.

Narcowich, F. J., Smith, P. and Ward, J. D. (1995). Density of translates of radial functions on compact sets, in *Approximation Theory VIII, Vol. 1: Approximation and Interpolation*, C. Chui, and L. Schumaker (eds.), World Scientific Publishing (Singapore), pp. 435–442.

Narcowich, F. J. and Ward, J. D. (1991a). Norms of inverses and condition numbers for matrices associated with scattered data, *J. Approx. Theory* 64, pp. 69–94.

Narcowich, F. J. and Ward, J. D. (1991b). Norms of inverses for matrices associated with scattered data, in *Curves and Surfaces*, P.-J. Laurent, A. Le Méhauté, and L. L. Schumaker (eds.), Academic Press (New York), pp. 341–348.

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

Narcowich, F. J. and Ward, J. D. (1994a). Generalized Hermite interpolation via matrix-valued conditionally positive definite functions, *Math. Comp.* **63**, pp. 661–687.

Narcowich, F. J. and Ward, J. D. (1994b). Wavelets associated with periodic basis functions, CAT #340, Texas A&M University.

Narcowich, F. J. and Ward, J. D. (1995). Nonstationary spherical wavelets for scattered data, in *Approximation Theory VIII, Vol. 2: Wavelets and Multilevel Approximation*, C. Chui, and L. Schumaker (eds.), World Scientific Publishing (Singapore), pp. 301–308.

Narcowich, F. J. and Ward, J. D. (1996). Nonstationary wavelets on the m-sphere for scattered data, *Appl. Comput. Harmon. Anal.* **3** 4, pp. 324–336.

Narcowich, F. J. and Ward, J. D. (2002). Scattered-data interpolation on spheres: error estimates and locally supported basis functions, *SIAM J. Math. Anal.* **33** 6, pp. 1393–1410.

Narcowich, F. J. and Ward, J. D. (2004). Scattered-data interpolation on $\mathbb{R}^n$: error estimates for radial basis and band-limited functions, *SIAM J. Math. Anal.* **36** 1, pp. 284–300.

Narcowich, F. J., Ward, J. D. and Wendland, H. (2003). Refined error estimates for radial basis function interpolation, *Constr. Approx.* **19** 4, pp. 541–564.

Narcowich, F. J., Ward, J. D. and Wendland, H. (2005). Sobolev bounds on functions with scattered zeros, with applications to radial basis function surface fitting, *Math. Comp.* **74**, pp. 743–763.

Narcowich, F. J., Ward, J. D. and Wendland, H. (2006). Sobolev error estimates and a Bernstein inequality for scattered data interpolation via radial basis functions, *Constr. Approx.* **24** 2, pp. 175–186.

Neyman, J. and Pearson, E. S. (1936). Contributions to the Theory of Testing Statistical Hypotheses, *Statistical Research Memoirs* **1**, pp. 1–37.

Nielson, G. M. (1987). Coordinate free scattered data interpolation, in *Topics in Multivariate Approximation*, C. K. Chui, L. L. Schumaker, and F. Utreras (eds.), Academic Press (New York), pp. 175–184.

Nielson, G. M. and Foley, T. A. (1989). A survey of applications of an affine invariant norm, in *Mathematical Methods in Computer Aided Geometric Design*, T. Lyche and L. Schumaker (eds.), Academic Press (New York), pp. 445–467.

Nieslony, A., Potts, D. and Steidl, G. (2004). Rapid evaluation of radial functions by fast Fourier transforms at nonequispaced knots, *Numer. Math.* **98**, pp. 329–351.

Nuss, W. A. and Titley, D. W. (1994). Use of multiquadric interpolation for meteorological objective analysis, *Monthly Weather Review* **12**, pp. 1011–1031.

Ohtake, Y., Belyaev, A. and Seidel, H. P. (2003a). A multi-scale approach to 3D scattered data interpolation with compactly supported basis functions, in *Shape Modeling International, 2003*, pp. 153–161.

Ohtake, Y., Belyaev, A., Alexa, M., Turk, G. and Seidel, H. P. (2003b). Multi-level partition of unity implicits, *ACM Trans. on Graphics (Proc. SIGGRAPH 2003)* **22** 3 pp. 463–470.

Opfer, R. (2004). Multiscale Kernels, Ph.D. Dissertation, Universität Göttingen.

Opfer, R. (2006). Multiscale Kernels, *Adv. in Comput. Math.* **25** 4, pp. 357–380.

Orr, M. J. L. (1995). Regularization in the selection of radial basis function centres, *Neural Computation* **7**, pp. 606–623.

Orr, M. J. L. (1996). Introduction to radial basis function networks, Centre for Cognitive Sciences, University of Edinburgh, available online at `http://www.anc.ed.ac.uk/rbf/intro/intro.html`.

Orr, M., Hallam, J., Takezawa, K., Murray, A., Ninomiya, S., Oide, M. and Leonard, T. (2000). Combining regression trees and radial basis function networks, *Int. J. Neural Systems* **10**, pp. 453–465.

Pahner, U. and Hameyer, K. (2000). Adaptive coupling of differential evolution and multiquadrics approximation for the tuning of the optimization process, *IEEE Trans. Magn.* **36** 1, pp. 1047–1051.

Park, J. and Sandberg, I. W. (1991). Universal approximation using radial bais function networks, *Neural Computation* **3**, pp. 246–257.

Park, J. and Sandberg, I. W. (1993). Approximation and radial basis function networks, *Neural Computation* **5**, pp. 305–316.

Parzen, E. (1962). On estimation of a probability density function and mode, *Ann. Math. Statist.* **33**, pp. 1065–1076.

Pegram, C. and Pegram, D. S. (1993). Integration of rainfall via multiquadric surfaces over polygons, *J. Hydraulic Eng.* **114**, pp. 151–163.

Pinkus, A. (2004). Strictly positive definite functions on a real inner product space, *Adv. in Comput. Math.* **20**, pp. 263–271.

Platte, R. B. and Driscoll, T. A. (2004). Computing eigenmodes of elliptic operators using radial basis functions, *Comput. Math. Appl.* **48**, pp. 561–576.

Platte, R. B. and Driscoll, T. A. (2005). Polynomials and potential theory for Gaussian radial basis function interpolation, *SIAM J. Numer. Anal.*, **43**, pp. 750–766.

Platte, R. B. and Driscoll, T. A. (2006). Eigenvalue stability of radial basis function discretizations for time-dependent problems, *Comput. Math. Appl.* **51** 8, pp. 1251–1268.

Poggio, T. and Girosi, F. (1990). Networks for approximation and learning, *Proc. IEEE* **78**, pp. 1481–1497.

Potter, E. H. (1981). Multivariate polyharmonic spline interpolation, Ph.D. Dissertation, Iowa State University.

Pottmann, H. and Eck, M. (1990). Modified multiquadric methods for scattered data interpolation over a sphere, *Comput. Aided Geom. Design* 7, pp. 313–321.

Pottmann, M. and Jörgl, H. P. (1995). Radial basis function networks for internal model control, *Appl. Math. Comput.* **70**, pp. 283–298.

Potts, D. and Steidl, G. (2003). Fast summation at nonequispaced knots by NFFTs, *SIAM J. Sci. Comput.* **24**, pp. 2013–2037.

Powell, M. J. D. (1987). Radial basis functions for multivariable interpolation: a review, in *Algorithms for the Approximation of Functions and Data*, J. C. Mason and M. G. Cox (eds.), Oxford Univ. Press (Oxford), pp. 143–167.

Powell, M. J. D. (1988). Radial basis function approximations to polynomials, in *Numerical Analysis 1987*, D. F. Griffiths and G. A. Watson (eds.), Longman Scientific and Technical (Essex), pp. 223–241.

Powell, M. J. D. (1990). Univariate multiquadric approximation: reproduction of linear polynomials, in *Multivariate Approximation and Interpolation*, ISNM 94, W. Haussman and K. Jetter (eds.), Birkhäuser (Basel), pp. 227–240.

Powell, M. J. D. (1991). Univariate multivariate interpolation: some recent results, in *Curves and Surfaces*, P.-J. Laurent, A. Le Méhauté, and L. L. Schumaker (eds.), Academic Press (New York), pp. 371–382.

Powell, M. J. D. (1992a). The theory of radial basis functions in 1990, in *Advances in Numerical Analysis II: Wavelets, Subdivision, and Radial Basis Functions*, W. Light (ed.), Oxford University Press (Oxford), pp. 105–210.

Powell, M. J. D. (1992b). Tabulation of thin-plate splines on a very fine two-dimensional grid, in *Numerical Methods in Approximation Theory*, ISNM 105, D. Braess, L. L. Schumaker (ed.), Birkhäuser (Basel), pp. 221–244.

Powell, M. J. D. (1993). Truncated Laurent expansions for the fast evaluation of thin plate splines, in *Algorithms for Approximation III*, J. C. Mason and M. G. Cox (eds.), Chapman and Hall (London), pp. 99–120.

Powell, M. J. D. (1994a). Some algorithms for thin plate spline interpolation to functions of two variables, in *Advances in Computational Mathematics (New Delhi, 1993)*, H. P. Dikshit and C. A. Micchelli (eds.), World Sci. Publishing, pp. 303–319.

Powell, M. J. D. (1994b). The uniform convergence of thin plate spline interpolation in two dimensions, *Numer. Math.* **68**, pp. 107–128.

Powell, M. J. D. (1997). A new iterative algorithm for thin plate spline interpolation in two dimensions, *Annals of Numerical Mathematics* **4**, pp. 519–528.

Powell, M. J. D. (1999). Recent research at Cambridge on radial basis functions, in *New Developments in Approximation Theory*, M. W. Müller, M. D. Buhmann, D. H. Mache and M. Felten (eds.), Birkhäuser (Basel), pp. 215–232.

Power, H. and Barraco, V. (2002). A comparison analysis between unsymmetric and symmetric radial basis function collocation methods for the numerical solution of partial differential equations, *Comput. Math. Appl.* **43**, pp. 551–583.

Qian, L.-F. and Ching, H.-K. (2004). Static and dynamic analysis of 2-D functionally graded elasticity by using meshless local Petrov-Galerkin method, *J. Chinese Inst. Engineers* **27**, pp. 491–503.

Quak, E., Sivakumar, N. and Ward, J. D. (1993). Least squares approximation by radial functions, *SIAM J. Numer. Anal.* **24**, 1043-1066.

Rabut, C. (1989). Fast quasi-interpolation of surfaces with generalized B-splines on regular nets, in *Mathematics of Surfaces III*, D. C. Handscomb (ed.), Clarendon Press (Oxford), pp. 429–449.

Rabut, C. (1990). B-splines polyharmoniques cardinales: Interpolation, quasi-interpolation, filtrage, Ph.D. Dissertation, University of Toulouse.

Rabut, C. (1992a). An introduction to Schoenberg's approximation, *Comput. Math. Appl.* **24**, pp. 149–175.

Rabut, C. (1992b). Elementary m-harmonic cardinal B-splines, *Numer. Algorithms* **2**, pp. 39–67.

Rabut, C. (1992c). High level m-harmonic cardinal B-splines, *Numer. Algorithms* **2**, pp. 68–84.

Ragozin, D. L. and Levesley, J. (1996). Zonal kernels, approximations and positive definiteness on spheres and compact homogeneous spaces, in *Curves and Surfaces with Applications in CAGD*, A. Le Méhauté, C. Rabut, and L. L. Schumaker (eds.), Vanderbilt University Press (Nashville, TN), pp. 371–378.

Reinsch, C. H. (1967). Smoothing by spline functions, *Numer. Math.* **10**, pp. 177–183.

Riesz, F. and Sz.-Nagy, B. (1955). *Functional Analysis*, Dover Publications (New York), republished 1990.

Rippa, S. (1984). Interpolation and smoothing of scattered data by radial basis functions, M.S. Thesis, Tel Aviv University.

Rippa, S. (1999). An algorithm for selecting a good value for the parameter c in radial basis function interpolation, *Adv. in Comput. Math.* **11**, pp. 193–210.

Ron, A. (1992). The L_2-approximation orders of principal shift-invariant spaces generated by a radial basis function, in *Numerical Methods in Approximation Theory*, ISNM 105, D. Braess, L. L. Schumaker (ed.), Birkhäuser (Basel), pp. 245–268.

Ron, A. and Sun, X. (1996). Strictly positive definite functions on spheres, *Math. Comp.* **65** 216, pp. 1513–1530.

Rosenblatt, M. (1956). Remarks on some nonparametric estimates of a density function, *Ann. Math. Statist.* **27**, pp. 832–837.

Rosenblum, M. and Davis, L. S. (1996). An improved radial basis function network for visual autonomous road following, *IEEE Trans. Neural Networks* **7**, pp. 1111–1120.

Rosenblum, M., Yacoob, Y. and Davis, L. S. (1996). Human expression recognition from motion using a radial basis function network architecture, *IEEE Trans. Neural Networks* **7**, pp. 1121–1138.

Rudin, W. (1973). *Functional Analysis*, McGraw-Hill (New York).

Saad, Y. and Schultz, M. H. (1986). GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear equations, *SIAM J. Sci. Statist. Comput.* **7**, pp. 856–869.

Saha, A., Wu, C. L., and Tang, D. S. (1993). Approximation, dimension reduction and nonconvex optimization using linear superpositions of Gaussians, *IEEE Trans. on Computers* **42**, pp. 1222–1233.

Šalkauskas, K. (1992). Moving least squares interpolation with thin-plate splines and radial basis functions, *Comput. Math. Appl.* **24**, pp. 177–185.

Šalkauskas, K. and Bos, L. (1992). Weighted splines as optimal interpolants, *Rocky Mountain J. Math.* **22**, pp. 205–217.

Sanner, R. M. and Slotine, J. J. E. (1992). Gaussian networks for direct adaptive control, *IEEE Trans. Neural Networks* **3**, pp. 837–863.

Šarler, B. and Vertnik, R. (2006). Meshfree explicit local radial basis function collocation method for diffusion problems, *Comput. Math. Appl.* **51** 8, pp. 1163–1170.

Sarra, S. A. (2005). Adaptive radial basis function methods for time dependent partial differential equations, *Appl. Numer. Math.* **54**, pp. 79–94.

Sarra, S. A. (2006). Integrated multiquadric radial basis function approximation methods, *Comput. Math. Appl.* **51** 8, pp. 1283–1296.

Saundersen, H. C. (1992). Multiquadric interpolation of fluid speeds in a natural river channel, *Comput. Math. Appl.* **24**, pp. 187–193.

Schaback, R. (1993). Comparison of radial basis function interpolants, in *Multivariate Approximation: From CAGD to Wavelets*, Kurt Jetter and Florencio Utreras (eds.), World Scientific Publishing (Singapore), pp. 293–305.

Schaback, R. (1994a). Approximation of polynomials by radial basis functions, in *Wavelets, Images, and Surface Fitting*, P.-J. Laurent, A. Le Méhauté, and L. L. Schumaker (eds.), A. K. Peters (Wellesley, MA), pp. 459–466.

Schaback, R. (1994b). Lower bounds for norms of inverses of interpolation matrices for radial basis functions, *J. Approx. Theory* **79**, pp. 287–306.

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

Schaback, R. (1995b). Error estimates and condition numbers for radial basis function interpolation, *Adv. in Comput. Math.* **3**, pp. 251–264.

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

Schaback, R. (1996). Approximation by radial basis functions with finitely many centers, *Constr. Approx.* **12**, pp. 331–340.

Schaback, R. (1997a). Optimal recovery in translation-invariant spaces of functions, *Annals of Numerical Mathematics* 4, pp. 547–556.

Schaback, R. (1997b). On the efficiency of interpolation by radial basis functions, in *Surface Fitting and Multiresolution Methods*, A. Le Méhauté, C. Rabut, and L. L. Schumaker (eds.), Vanderbilt University Press (Nashville, TN), pp. 309–318.

Schaback, R. (1999a). Native Hilbert spaces for radial basis functions I, in *New Developments in Approximation Theory*, M. W. Müller, M. D. Buhmann, D. H. Mache and M. Felten (eds.), Birkhäuser (Basel), pp. 255–282.

Schaback, R. (1999b). Improved error bounds for scattered data interpolation by radial basis functions, *Math. Comp.* **68** 225, pp. 201–216.

Schaback, R. (2000a). A unified theory of radial basis functions. Native Hilbert spaces for radial basis functions II, *J. Comput. Appl. Math.* **121**, pp. 165–177.

Schaback, R. (2000b). Remarks on meshless local construction of surfaces, in *The Mathematics of Surfaces, IX*, R. Cipolla and R. Martin (eds.), Springer, pp. 34–58.

Schaback, R. (2002) Stability of radial basis function interpolants, in *Approximation Theory X*, C. K. Chui, L. L. Schumaker, and J. Stöckler (eds.), Vanderbilt Univ. Press (Nashville, TN), pp. 433–440.

Schaback, R. (2003). On the versatility of meshless kernel methods, in *Advances in Com-*

putational and Experimental Engineering and Sciences, S. N. Atluri, D. E. Beskos, and D. Polyzos (eds.), CD ROM, ICCES proceedings paper #428.

Schaback, R. (2005). Multivariate interpolation by polynomials and radial basis functions, *Constr. Approx.* **21**, pp. 293–317.

Schaback, R. (2006a) Convergence of unsymmetric kernel-based meshless collocation methods, *SIAM J. Numer. Anal.*, to appear.

Schaback, R. (2006b) Limit problems for interpolation by analytic radial basis functions, *J. Comp. Appl. Math.*, to appear.

Schaback, R. and Wendland, H. (1999). Using compactly supported radial basis functions to solve partial differential equations, in *Boundary Element Technology XIII*, C. S. Chen, C. A. Brebbia, and D. W. Pepper (eds.), WIT Press, pp. 311–324.

Schaback, R. and Wendland, H. (2000a). Numerical techniques based on radial basis functions, in *Curve and Surface Fitting: Saint-Malo 1999*, A. Cohen, C. Rabut, and L. L. Schumaker (eds.), Vanderbilt University Press (Nashville, TN), 359-374.

Schaback, R. and Wendland, H. (2000b). Adaptive greedy techniques for approximate solution of large RBF systems, *Numer. Algorithms* **24**, pp. 239–254.

Schaback, R. and Wendland, H. (2001). Characterization and construction of radial basis functions, in *Multivariate Approximation and Applications*, N. Dyn, D. Leviatan, D. Levin, and A. Pinkus (eds.), Cambridge Univ. Press (Cambridge), pp. 1–24.

Schaback, R. and Wendland, H. (2006). Kernel techniques: From machine learning to meshless methods, *Acta Numerica*, **15**, pp. 543–639.

Schaback, R. and Werner, J. (2006). Linearly constrained reconstruction of functions by kernels with applications to machine learning, *Adv. in Comput. Math.* **25**, pp. 237–258.

Schaback, R. and Wu, Z. (1996). Operators on radial functions, *J. Comput. Appl. Math.* **73**, pp. 257–270.

Schagen, I. P. (1984). Sequential exploration of unknown multidimensional functions as an aid to optimization, *IMA J. Numer. Anal.* **4**, pp. 337–347.

Schimming, R. and Belger, M. (1991). Polyharmonic radial functions on a Riemannian manifold, *Math. Nachr.* **153**, pp. 207–216.

Schiro, R. and Williams, G. (1984). An adaptive application of multiquadric interpolants for numerically modeling large numbers of irregularly spaced hydrographic data, *Surv. Mapp.* **44**, pp. 365–381.

Schölkopf, B. and Smola, A. J. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press (Cambridge, MA).

Schoenberg, I. J. (1937). On certain metric spaces arising from Euclidean spaces by a change of metric and their imbedding in Hilbert space, *Ann. of Math.* **38**, pp. 787–793.

Schoenberg, I. J. (1938a). Metric spaces and completely monotone functions, *Ann. of Math.* **39**, pp. 811–841.

Schoenberg, I. J. (1938b). Metric spaces and positive definite functions, *Trans. Amer. Math. Soc.* **44**, pp. 522–536.

Schoenberg, I. J. (1942). Positive definite functions on spheres, *Duke Math. J.* **9**, pp. 96–108.

Schoenberg, I. J. (1964). Spline functions and the problem of graduation, *Proc. Nat. Acad. Sci.* **52**, pp. 947–950.

Schreiner, M. (1997). On a new condition for strictly positive definite functions on spheres, *Proc. Amer. Math. Soc.* **125** 2, pp. 531–539.

Schumaker, L. L. (1981). *Spline Functions: Basic Theory*, John Wiley & Sons (New York), reprinted by Krieger Publishing 1993.

Schweitzer, M. A. (2003). *A Parallel Multilevel Partition of Unity Method for Elliptic Partial Differential Equations*, Lecture Notes in Computational Science and Engineering, Vol. 29, Springer Verlag (Berlin).

Sethuraman, R. and Reddy, C. S. (2004). Pseudo elastic analysis of material non-linear problems using element free Galerkin method, *J. Chinese Inst. Engineers* **27**, pp. 505–516.

Shannon, C. (1949). Communication in the presence of noise, *Proc. IRE* **37**, pp. 10–21.

Sharan, M., Kansa, E. J. and Gupta, S. (1997). Application of the multiquadric method for numerical solution of elliptic partial differential equations, *Appl. Math. Comp.*, **84** 2-3, pp. 275–302.

Shaw, E. W. and Lynn, P. P. (1972). Area rainfall evaluation using two surface fitting techniques, *Bull. Int. Ass. Hydrol. Sci.* **17**, pp. 419–433.

Shepard, D. (1968). A two dimensional interpolation function for irregularly spaced data, *Proc. 23rd Nat. Conf. ACM*, pp. 517–524.

Shepherd, T. J. and Broomhead, D. S. (1990). Nonlinear signal processing using radial basis functions, *SPE Advanced Signal Processing Algorithms, Architectures, and Implementations* **1348**, pp. 51–61.

Sherstinsky, A. and Picard, R. W. (1996). On the efficiency of the orthogonal least squares training method for radial basis function networks, *IEEE Trans. Neural Networks* **7**, pp. 195–200.

Shu, C., Ding, H. and Yeo, K. S. (2003). Local radial basis function-based differential quadrature method and its application to solve two-dimensional incompressible Navier-Stokes equations, *Comput. Methods Appl. Mech. Engrg.* **192**, pp. 941–954.

Shu, C., Ding, H. and Yeo, K. S. (2004). Solution of partial differential equations by global radial basis function-based differential quadrature method, *Engineering Analysis with Boundary Elements* **28**, pp. 1217–1226.

Shu, C., Ding, H. and Zhao, N. (2006). Numerical comparison of least square-based finite-difference (LSFD) and radial basis function-based finite-difference (RBFFD) methods, *Comput. Math. Appl.* **51** 8, pp. 1297–1310.

Sibson, R. and Stone, G. (1991). Computation of thin-plate splines, *SIAM J. Sci. Statist. Comput.* **12**, pp. 1304–1313.

Sivakumar, N. and Ward, J. D. (1993). On the least squares fit by radial functions to multidimensional scattered data, *Numer. Math.* **65**, pp. 219–243.

Smith, B. F., Bjørstad, P. and Gropp, W. D. (1996). *Domain Decomposition. Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press (Cambridge).

Smith, K. T., Solmon, D. C. and Wagner, S. L. (1977). Practical and mathematical aspects of the problem of reconstructing objects from radiographs, *Bull. Amer. Math. Soc.* **83**, pp. 1227–1270.

Sneddon, I. H. (1972). *The Use of Integral Transforms*, McGraw-Hill (New York).

Sonar, T. (1995). Optimal recovery using thin plate splines in finite volume methods for the numerical solution of hyperbolic conservation laws, *IMA J. Numer. Anal.* **16**, pp. 549–581.

Stead, S. (1984). Estimation of gradients from scattered data, *Rocky Mountain J. Math.* **14**, pp. 265–279.

Steele, N. C., Reeves, C. R., Nicholas, M. and King, P. J. (1995). Radial basis function artificial neural networks for the inference process in fuzzy logic based control, *Computing* **54**, pp. 99–117.

Stein, E. M. and Weiss, G. (1971). *Introduction to Fourier Analysis in Euclidean Spaces*, Princeton University Press (Princeton).

Stein, M. L. (1999). *Interpolation of spatial data. Some theory for Kriging*, Springer Series in Statistics, Springer-Verlag (New York).

Stewart, J. (1976). Positive definite functions and generalizations, an historical survey, *Rocky Mountain J. Math.* 6, pp. 409–434.

Strain, J. (1991). The fast Gauss transform with variable scales, *SIAM J. Sci. Statist. Comput.* **12**, pp. 1131–1139.

Sun, X. (1989). On the solvability of radial function interpolation, in *Approximation Theory VI*, C. Chui, L. Schumaker, and J. Ward (eds.), Academic Press (New York), pp. 643–646.

Sun, X. (1990). Multivariate interpolation using ridge or related functions, Ph.D. Dissertation, University of Texas.

Sun, X. (1992a). Norm estimates for inverses of Euclidean distance matrices, *J. Approx. Theory* **70**, pp. 339–347.

Sun, X. (1992b). Cardinal and scattered-cardinal interpolation by functions having non-compact support, *Comput. Math. Appl.* **24**, pp. 195–200.

Sun, X. (1993a). Solvability of multivariate interpolation by radial or related functions, *J. Approx. Theory* **72**, pp. 252–267.

Sun, X. (1993b). Conditionally positive definite functions and their application to multivariate interpolation, *J. Approx. Theory* **74**, pp. 159–180.

Sun, X. (1994a). Scattered Hermite interpolation using radial basis functions, *Linear Algebra Appl.* **207**, pp. 135–146.

Sun, X. (1994b). The fundamentality of translates of a continuous function on spheres, *Numerical Algorithms* **8**, pp. 131–134.

Sun, X. (1994c). Cardinal Hermite interpolation using positive definite functions, *Numerical Algorithms* **7**, pp. 253–268.

Sun, X. (1995). Conditional positive definiteness and complete monotonicity, in *Approximation Theory VIII, Vol. 1: Approximation and Interpolation*, C. Chui, and L. Schumaker (eds.), World Scientific Publishing (Singapore), pp. 537–540.

Sun, X. and Cheney, E. W. (1997). Fundamental sets of continuous functions on spheres, *Constr. Approx.* **13** 2, pp. 245–250.

Suykens, J. A. K., De Brabanter, J., Lukas, L. and Vandewalle, J. (2002). Weighted least squares support vector machines: robustness and sparse approximation, *Neurocomputing* **48** 1-4, pp. 85–105.

Szegő, G. (1959). *Orthogonal Polynomials*, Amer. Math. Soc. Coll. Publ. Vol. XXIII (Providence).

Tan, S., Hao, J. and Vandewalle, J. (1995). Multivariable nonlinear system identification by radial basis function neural networks, *Arch. Control Sci.* **4**, pp. 55–67.

Tarwater, A. E. (1985). A parameter study of Hardy's multiquadric method for scattered data interpolation, Lawrence Livermore National Laboratory, TR UCRL-563670.

Temlyakov, V. N. (1998). The best m-term approximation and greedy algorithms, *Adv. in Comp. Math.* **8**, pp. 249–265.

Tiago, C. M. and Leitão, V. M. A. (2006). Application of radial basis functions to linear and nonlinear structural analysis problems, *Comput. Math. Appl.* **51** 8, pp. 1311–1334.

Trefethen, L. N. (2000). *Spectral Methods in* MATLAB, SIAM (Philadelphia, PA).

Trefethen, L. N. and Bau, D. (1997). *Numerical Linear Algebra*, SIAM (Philadelphia, PA).

Treinish, L. A. (1995). Visualization of Scattered Meteorological Data, *IEEE Computer Graphics & Applications* **15**, pp. 20–26.

Tsukanov, I. and Shapiro, V. (2005). Meshfree modeling and analysis of physical fields in heterogeneous media, *Adv. in Comput. Math.* **23** 1-2, pp. 95–124.

Turk, G. and O'Brien, J. F. (2002). Modelling with implicit surfaces that interpolate, *ACM Transactions on Graphics* **21**, pp. 855–873.

Unser, M. (2000). Sampling — 50 years after Shannon, *Proc. IEEE* 88, pp. 569–587.

Utreras, F. I. (1985). Positive thin plate splines, *Approx. Theory Appl.* **1**, pp. 77–108.

Utreras, F. I. (1993). Multiresolution and pre-wavelets using radial basis functions, in *Multivariate Approximation: From CAGD to Wavelets*, Kurt Jetter and Florencio Utreras (eds.), World Scientific Publishing (Singapore), pp. 321–333.

Utreras, F. I. and Vargas, M. L. (1991). Monotone interpolations of scattered data in $\mathbb{R}^2$, *Constr. Approx.* **7**, pp. 49–68.

Villalobos, M. A. and Wahba, G. (1983). Multivariate thin plate spline estimates for the posterior probabilities in the classification problem, *Comm. Statist. A—Theory Methods* **12**, pp. 1449–1479.

Vrankar, L., Turk, G. and Runovc, F. (2004). modelling of radionuclide migration through the geosphere with radial basis function method and geostatistics, *J. Chinese Inst. Engineers* **27**, pp. 455–462.

Wahba, G. (1979). Convergence rate of "thin plate" smoothing splines when the data are noisy (preliminary report), *Springer Lecture Notes in Math.* **757**, pp. 233–245.

Wahba, G. (1981). Spline Interpolation and smoothing on the sphere, *SIAM J. Sci. Statist. Comput.* **2**, pp. 5–16.

Wahba, G. (1982). Erratum: Spline interpolation and smoothing on the sphere, *SIAM J. Sci. Statist. Comput.* 3, pp. 385–386.

Wahba, G. (1986). Multivariate thin plate spline smoothing with positivity and other linear inequality constraints, in *Statistical image processing and graphics (Luray, Va., 1983)*, Statist.: Textbooks Monographs, 72, Dekker (New York), pp. 275–289.

Wahba, G. (1990a). Multivariate model building with additive interaction and tensor product thin plate splines, in *Curves and Surfaces*, P.-J. Laurent, A. Le Méhauté, and L. L. Schumaker (eds.), Academic Press (New York), pp. 491–504.

Wahba, G. (1990b). *Spline Models for Observational Data*, CBMS-NSF Regional Conference Series in Applied Mathematics 59, SIAM (Philadelphia).

Wahba, G. and Luo, Z. (1997). Smoothing spline ANOVA fits for very large, nearly regular data sets, with application to historical global climate data, in *The Heritage of P. L. Chebyshev: a Festschrift in honor of the 70th birthday of T. J. Rivlin, Ann. Numer. Math.* 4 1–4, pp. 579–597.

Wahba, G. and Wendelberger, J. (1980). Some new mathematical methods for variational objective analysis using splines and cross validation, *Monthly Weather Review* **108**, pp. 1122–1143.

Wang, S. and Wang, M. Y. (2006). Radial basis functions and level set method for structural topology optimization, *Int. J. Numer. Meth. Engng.* **65** 12, pp. 2060–2090.

Ward, J. D. (2004). Least squares approximation using radial basis functions: an update, in *Advances in Constructive Approximation: Vanderbilt 2003*, M. Neamtu and E. B. Saff (eds.), Nashboro Press (Brentwood, TN), pp. 499–508.

Watson, G. S. (1964). Smooth regression analysis, *Sankhya, Ser. A* **26**, pp. 359–372.

Webb, A. R. (1995). Multidimensional scaling by iterative majorisation using radial basis functions, *Pattern Recognition* **28**, pp. 753–759.

Weinrich, M. (1994). Charakterisierung von Funktionenräumen bei der Interpolationn mit radialen Basisfunktionen, Ph.D. Dissertation, Universität Göttingen.

Wells, J. H. and Williams, R. L. (1975). *Embeddings and Extensions in Analysis*, Springer (Berlin).

Wendland, H. (1994). Ein Beitrag zur Interpolation mit radialen Basisfunktionen, Diplomarbeit, Universität Göttingen.

Wendland, H. (1995). Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree, *Adv. in Comput. Math.* **4**, pp. 389–396.

Wendland, H. (1997). Sobolev-type error estimates for interpolation by radial basis functions, in *Surface Fitting and Multiresolution Methods*, A. Le Méhauté, C. Rabut, and L. L. Schumaker (eds.), Vanderbilt University Press (Nashville, TN), pp. 337–344.

Wendland, H. (1998). Error estimates for interpolation by compactly supported radial basis functions of minimal degree, *J. Approx. Theory* **93**, pp. 258–272.

Wendland, H. (1999a). Meshless Galerkin approximation using radial basis functions, *Math. Comp.* **68**, pp. 1521–1531.

Wendland, H. (1999b). Numerical solution of variational problems by radial basis functions, in *Approximation Theory IX, Vol.II: Computational Aspects*, Charles K. Chui, and L. L. Schumaker (eds.), Vanderbilt University Press, pp. 361–368.

Wendland, H. (2001a). Local polynomial reproduction and moving least squares approximation, *IMA J. Numer. Anal.* **21** 1, pp. 285–300.

Wendland, H. (2001b). Gaussian interpolation revisited, in *Trends in Approximation Theory*, K. Kopotun, T. Lyche, and M. Neamtu (eds.), Vanderbilt University Press, pp. 417–426.

Wendland, H. (2001c). Moving least squares approximation on the sphere, in *Mathematical Methods for Curves and Surfaces: Oslo 2000*, T. Lyche and L. L. Schumaker (eds.), Vanderbilt University Press, pp. 517–526.

Wendland, H. (2002a). Fast evaluation of radial basis functions: Methods based on partition of unity, in *Approximation Theory X: Wavelets, Splines, and Applications*, C. K. Chui, L. L. Schumaker, and J. Stöckier (eds.), Vanderbilt University Press (Nashville), 473-483.

Wendland, H. (2002b). Surface reconstruction from unorganized points, `http://www.num.math.uni-goettingen.de/wendland/Forschung/recon-html/reconhtml.html`.

Wendland, H. (2004). Solving large generalized interpolation problems efficiently, in *Advances in Constructive Approximation: Vanderbilt 2003*, M. Neamtu and E. B. Saff (eds.), Nashboro Press, Brentwood, TN, pp. 509–518.

Wendland, H. (2005a). *Scattered Data Approximation*, Cambridge University Press (Cambridge).

Wendland, H. (2005b). On the convergence of a general class of finite volume methods, *SIAM J. Numer. Anal.* **43**, pp. 987–1002.

Wendland, H. (2005c). Private communications.

Wendland, H. and Rieger, C. (2005). Approximate interpolation with applications to selecting smoothing parameters, *Numer. Math.* 101 4, pp. 729–748.

Wertz, J., Kansa, E. J. and Ling, L. (2006). The role of the multiquadric shape parameters in solving elliptic partial differential equations, *Comput. Math. Appl.* **51** 8, pp. 1335–1348.

Wetterschereck, D. and Dietterich, T. (1992). Improving the performance of radial basis function networks by learning center locations, in *Advances in Neural Information Processing Systems 4. Proceedings of the 1991 Conference*, J. E. Moody, S. J. Hanson, and R. P. Lippmann (eds.), Morgan Kaufmann (San Mateo, CA), pp. 1133–1140.

Whittaker, J. M. (1915). On the functions which are represented by expansions of the interpolation theory, *Proc. Roy. Soc. Edinburgh* **35**, pp. 181–194.

Whittaker, E. T. (1923). On a new method of graduation, *Proc. Edinburgh Math. Soc.* **41**, pp. 63–75.

Whitehead, B. A. (1996). Genetic evolution of radial basis function coverage using orthogonal niches, *IEEE Trans. Neural Networks* 7, pp. 1525–1528.

Whitehead, B. A. and Choate, T. D. (1996). Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction, *IEEE Trans. Neural Networks* **7**, pp. 869–880.

Widder, D. V. (1941). *The Laplace Transform*, Princeton University Press (Princeton).

Williamson, R. E. (1956). Multiply monotone functions and their Laplace transform, *Duke Math. J.* **23**, pp. 189–207.

Wong, S. M., Hon, Y. C. and Li, T. S. (1999). Radial basis functions with compactly support and multizone decomposition: applications to environmental modelling, in *Boundary Element Technology XIII*, C. S. Chen, C. A. Brebbia, and D. W. Pepper (eds.), WIT Press, pp. 355–364.

Wong, T.-T., Luk, W.-S. and Heng, P.-A. (1997). Sampling with Hammersley and Halton points, *J. Graphics Tools* **2**, pp. 9–24.

Woolhouse, W. S. B. (1870). Explanation of a new method of adjusting mortality tables, with some observations upon Mr. Makeham's modification of Gompertz's theory, *J. Inst. Act.* **15**, pp. 389–410.

Wright, G. B. (2003). Radial Basis Function Interpolation: Numerical and Analytical Developments, Ph.D. Dissertation, University of Colorado at Boulder.

Wright, G. and Fornberg, B. (2006). Scattered node compact finite difference-type formulas generated from radial basis functions, *J. Comput. Physics* **212** 1, pp. 99–123.

Wu, Z. (1986). Die Kriging-Methode zur Lösung mehrdimensionaler Interpolationsprobleme, Ph.D. Dissertation, Universität Göttingen.

Wu, Z. (1990). A class of convexity-preserving bases for radial basis interpolation (in Chinese), *Math. Appl.* **3**, pp. 33–37.

Wu, Z. (1991). A convergence analysis for a class of radial basis interpolations (in Chinese), *Gaoxiao Yingyong Shuxue Xuebao* **6**, pp. 331–336.

Wu, Z. (1992). Hermite-Birkhoff interpolation of scattered data by radial basis functions, *Approx. Theory Appl.* **8**, pp. 1–10.

Wu, Z. (1993a). On the convergence of the interpolation with radial basis function, *Chinese J. Contemp. Math.* **14**, pp. 269–277.

Wu, Z. (1993b). Convergence of interpolation by radial basis functions (in Chinese), *Chinese Ann. Math. Ser. A* **14**, pp. 480–486.

Wu, Z. (1995a). Characterization of positive definite radial functions, in *Mathematical Methods for Curves and Surfaces*, M. Dæhlen, T. Lyche, and L. Schumaker (eds.), Vanderbilt University Press (Nashville), pp. 573–578.

Wu, Z. (1995b). Compactly supported positive definite radial functions, *Adv. in Comput. Math.* 4, pp. 283–292.

Wu, Z. (2005). Dynamical knot and shape parameter setting for simulating shock wave by using multi-quadric quasi-interpolation, *Engineering Analysis with Boundary Elements* **29**, pp. 354–358.

Wu, Z. M. and Liu, J. P. (2005). Generalized Strang-Fix condition for scattered data quasi-interpolation, *Adv. in Comput. Math.* **23**, pp. 201–214.

Wu, Z. and Schaback, R. (1993). Local error estimates for radial basis function interpolation of scattered data, *IMA J. Numer. Anal.* **13**, pp. 13–27.

Wu, Z. and Schaback, R. (1994). Shape preserving properties and convergence of univariate multiquadric quasi-interpolation, *Acta Math. Appl. Sinica (English Ser.)* **10**, pp. 441–446.

Xu, L., Kryzak, A. and Yuille, A. (1994). On radial basis function nets and kernel regression statistical consistency, convergence rates, and receptive field sizes, *Neural Networks* **7**, pp. 609–628.

Xu, Y. and Cheney, E. W. (1992a). Interpolation by periodic radial functions, *Comput. Math. Appl.* **24**, pp. 201–215.

Xu, Y. and Cheney, E. W. (1992b). Strictly positive definite functions on spheres, *Proc. Amer. Math. Soc.* **116**, pp. 977–981.

Xu, Y., Light, W. A. and Cheney, E. W. (1993). Constructive methods of approximation by ridge functions and radial functions, *Numer. Algorithms* **4**, pp. 205–223.

Yamada, T. and Wrobel, L. C. (1993). Properties of Gaussian radial basis functions in the dual reciprocity boundary element method, *Z. Angew. Math. Phys.* **44**, pp. 1054–1067.

Yang, Y. and Barron, A. (1999). Information-theoretic determination of minimax rates of convergence, *Ann. Statist.* **27**, pp. 1564–1599.

Yee, P. V. and Haykin, S. (2001). *Regularized Radial Basis Function Networks: Theory and Applications*, Wiley-Interscience.

Yoon, J. (2001). Interpolation by radial basis functions on Sobolev space, *J. Approx. Theory* **112**, pp. 1–15.

Yoon, J. (2003). L_p-error estimates for "shifted" surface spline interpolation on Sobolev space, *Math. Comp.* **72**, pp. 1349–1367.

Yoon, Y.-C., Lee, S.-H. and Belytschko, T. (2006). Enriched meshfree collocation method with diffuse derivatives for elastic fracture, *Comput. Math. Appl.* **51** 8, pp. 1349–1366.

Young, D. L., Jane, S. C., Lin, C. Y., Chiu, C. L. and Chen, K. C. (2004). Solutions of 2D and 3D Stokes laws using multiquadrics method, *Engineering Analysis with Boundary Elements 28*, pp. 1233–1243.

Zastavnyi, V. P. (1993). Positive definite functions depending on the norm, *Russian J. Math. Phys.* **1**, pp. 511–522.

Zeckzer, D. (1992). Dreiecks-basierte lokale Scattered Data Interpolation unter Verwendung radialer Basismethoden, Diplomarbeit, Universität Kaiserslautern.

Index