ITERATIVE TECHNIQUES FOR RADIAL BASIS FUNCTION INTERPOLATION

by

Anita Christine Faul

of

Churchill College

A dissertation presented in fulfilment of the requirements for the degree of Doctor of Philosophy of the University of Cambridge

ITERATIVE TECHNIQUES FOR RADIAL BASIS FUNCTION INTERPOLATION

A.C. Faul

Summary

The problem of interpolating functions comes up naturally in many areas of applied mathematics and natural sciences. Radial basis function methods provide an interpolant to values of a real function of d variables and are highly useful in many applications, especially if the function values are given at scattered data points.

The need for iterative procedures arises when the number of interpolation conditions n is large, since hardly any sparsity occurs in the linear system of interpolation equations. Solving this system with direct methods would require $\mathcal{O}(n^3)$ operations.

This dissertation considers several iterative techniques. They were developed from an algorithm described by Beatson, Goodsell and Powell (1995), which is examined first. By gaining more and more theoretical insight into the original algorithm, new algorithms are developed and connections to known methods are made. We establish the important role a certain semi-inner product plays in the convergence analysis of the original algorithm, and the first proof of convergence is given. This leads to a new technique using line searches described later. Then it is shown that the original algorithm is equivalent to solving a certain symmetric and positive definite system of equations by Gauss–Seidel iterations. Thus iterative techniques like Jacobi iterations and conjugate gradient methods follow. This symmetric and positive definite system of equations can be derived from the original system of equations by preconditioning it with a certain matrix. The preconditioned

conjugate gradient algorithm was first suggested for this problem by Dyn *et al.* (1983, 1986), motivated by the variational theory of thin plate splines. It is helpful to view the original algorithm as a linear operator working on a certain linear space equipped with the aforementioned semi-inner product.

The original algorithm had the drawback that the residuals had to be updated at several stages during each iteration. Another algorithm defers the updates till the end of each iteration, which usually improves efficiency greatly, but divergence occurs in some cases. Therefore a line search method is developed that ensures convergence.

The last technique described is a Krylov subspace method which proved to be very successful. It can be applied to any algorithm that fulfils certain criteria. If the underlying algorithm is convergent, the Krylov subspace technique speeds the convergence up. In cases of divergence, the Krylov subspace method enforces convergence. It is shown that the Krylov subspace method applied to the algorithm where updating the residuals is deferred till the end of each iteration is analogous to the conjugate gradient technique applied to the aforementioned symmetric and positive definite system of equations.

All algorithms are related to each other and theoretical insight into some properties of one algorithm leads to an improved algorithm. These considerations provide a highly useful theory, linking different techniques for iterative radial basis function interpolation.

Declaration

In this dissertation, all of the work is my own with the exception of Section 2 of Chapter 3 and Chapter 6 which contains results of my collaboration with Professor M.J.D. Powell. This collaboration was approved by the Board of Graduate Studies. No part of this thesis has been submitted for a degree

elsewhere. However some parts have appeared, or will appear, in journals. In particular, we refer the reader to Faul and Powell (1998,1999). Furthermore, several chapters formed a Smith's Prize essay in an earlier incarnation.

Preface

It is a pleasure to acknowledge the support I have received during my doctoral research.

First, I would like to thank my supervisor, Mike Powell, for his support, patience and understanding. His enthusiasm, insight, precision and wide mathematical knowledge have always inspired me. His guidance will be a lasting benefit. Also Rick Beatson and George Goodsell made highly important contributions to the original algorithm. Indeed, Rick had the idea originally of an iterative procedure for revising the coefficients of a radial basis function interpolant, and George wrote the computer program that guided the development of the technique into its present form. I am also very thankful for the input by Will Light.

Acknowledgement is also due to the different sources of financial support towards my studies. During my PhD I was financed through the EPSRC, the Cusanuswerk and the Cambridge European Trust. Visits to various conferences were made possible by the financial support from Churchill College, Cambridge, the Department of Applied Mathematics and Theoretical Physics and the fund of the John Humphrey Plummer Chair of Applied Numerical Analysis.

I also want to thank my parents Helmut and Marieluise Faul and my fiancee James Briginshaw for their continuous support.

Contents

Chapter 1

Introduction

1.1 Approximation theory

Interpolation of data, which might have been gathered by measurements of a physical quantity, or by sampling a function at scattered or periodic points of several variables, is a problem that occurs in many areas of science and engineering. Weather stations, for example, measure the atmospheric pressure above the earth surface and these data are then used to construct a contour map of the atmospheric pressure. Especially in cases where the data are sampled by measurements, they are scattered, which means that the positions of the data points do not obey any regular pattern. Generally, a discrete set $X = \{\underline{x}_1, \ldots, \underline{x}_n\}$ of points in d-dimensional space $\mathbb{R}^d$ and real valued data f_i, $i = 1, \ldots, n$, are given, and the task is to construct a continuous or sufficiently differentiable function $s^* : \mathbb{R}^d \to \mathbb{R}$ that satisfies the interpolation equations

$$s^*(\underline{x}_i) = f_i, \qquad i = 1, \ldots, n. \tag{1.1.1}$$

If s^* depends linearly on n parameters, these equations define a $n \times n$ system of linear equations.

Apart from approximating an unknown function, where only function values at the data points in X are known, there are other applications of interpolation methods. For example the underlying function might be too complex for many evaluations on a computer, and it might be necessary to replace it with a simpler function, retaining certain properties of the original function, in order to keep computing time low. One also might find that one has a set of data that is unacceptably large for computer storage. We then choose X to be a subset of that data and s^* is used to estimate the remaining data, which is a form of data compression.

Another issue occurs when we know that the data f_i, $i = 1, \ldots, n$, are subject to errors, which are often called noise, as we cannot "hear" the real information properly. In this case interpolation is not necessarily desirable. Then the interpolation equations (1.1.1) can be relaxed and replaced by the condition

$$\max\{|s^*(\underline{x}_i) - f_i| : i = 1, \ldots, n\} < \epsilon, \tag{1.1.2}$$

where ϵ is the known magnitude of the errors.

In other cases, we might want to choose a smooth s^* which depends on fewer than n parameters. We then have an under-determined system of equations and one option to choose the function s^* is by minimizing the sum of the squares of the residuals $s^*(\underline{x}_i) - f_i$, $i = 1, \ldots, n$, which is

$$\sum_{i=1}^{n} (s^*(\underline{x}_i) - f_i)^2. \tag{1.1.3}$$

This method is known as least-squares fitting or data-smoothing. The sum of squares is a functional acting on the difference $s^* - f$. In other applications

it might be more suitable to choose a different functional which is minimized to determine s^*.

Another possibility is known as quasi-interpolation. There we let s^* be the sum of rapidly decaying functions, each of which is centred at a data point $\underline{x}_i$ and is premultiplied by the factor f_i in the sum. Here it is assumed that $s^*(\underline{x})$ depends little on f_i if $\underline{x}$ is far away from $\underline{x}_i$. In general, s^* does not satisfy the interpolation conditions (1.1.1) in this case.

These methods are important in both practice and theory, but this dissertation is restricted to interpolation. There are many different techniques of interpolation and we give some examples here.

When interpolating with polynomials, we choose a linear space, say $\mathcal{P}$, of polynomials in d variables which is spanned by $p_1, \ldots, p_n$, where n is the number of interpolation conditions. Then an interpolant $s^* : \mathbb{R}^d \to \mathbb{R}$ of the form

$$s^*(\underline{x}) = \sum_{i=1}^{n} c_i \, p_i(\underline{x}), \qquad \underline{x} \in \mathbb{R}^d, \tag{1.1.4}$$

exists if and only if the $n \times n$ matrix P with entries $P_{ji} = p_i(\underline{x}_j)$ is invertible. In more than one dimension, i.e. if $d > 1$, this property depends on the positions of the data points $\underline{x}_1, \ldots, \underline{x}_n$, which presents a difficulty. One possibility to overcome this difficulty is to choose a particular geometry of the data points. In two dimensions for example, one might require that the data points form a finite square grid or more generally a "tartan grid". Here we are given real numbers $x_1 < \cdots < x_k$ and $y_1 < \cdots < y_l$. Let $n = k * l$, $X = \{\underline{x}_{i*j} = (x_i, y_j) : i = 1, \ldots, k, j = 1, \ldots, l\}$ and let the data be given by f_{i*j}, $i = 1, \ldots, k$, $j = 1, \ldots, l$. Further we let $L_{x,i}$, $i = 1, \ldots, k$, and $L_{y,j}$, $j = 1, \ldots, l$, be the usual univariate Lagrange interpolating polynomials

defined by

$$L_{x,i}(x_a) = \delta_{ia}, \quad i, a \in [1, k]$$
$$L_{y,j}(y_b) = \delta_{jb}, \quad j, b \in [1, l], \tag{1.1.5}$$

where δ_{ia} is the Kronecker symbol. Then our interpolant $s^* : \mathbb{R}^2 \to \mathbb{R}$ is given by

$$s^*(\underline{x}) = \sum_{i=1}^{k} \sum_{j=1}^{l} f_{i*j}\, L_{x,i}(x)\, L_{y,j}(y), \qquad \underline{x} = (x, y) \in \mathbb{R}^2. \tag{1.1.6}$$

Thus the interpolant is formed using a tensor product. This method can be extended to any number of dimensions d. It is not restricted to polynomials. We can choose suitable univariate functions $P_{x,i}$, $i = 1, \ldots, k$, and $Q_{y,j}$, $j = 1, \ldots, l$, and let our interpolant take the form

$$s^*(\underline{x}) = \sum_{i=1}^{k} \sum_{j=1}^{l} c_{i*j}\, P_{x,i}(x)\, Q_{y,j}(y), \qquad \underline{x} = (x, y) \in \mathbb{R}^2. \tag{1.1.7}$$

The coefficients $c_1, \ldots, c_n$ are obtained by solving the interpolation equations (1.1.1). The functions $P_{x,i}$, $i = 1, \ldots, k$, and $Q_{y,j}$, $j = 1, \ldots, l$, can be chosen, for example, to be univariate B-splines, if extra points are added outside the two intervals $[x_1, x_k]$ and $[y_1, y_k]$.

These methods are important if the data are given on a tartan grid, but what if the data points are in general positions. Some useful techniques in this case are finite element methods. When $d = 2$ for example, a triangulation of the data points is chosen, i.e. triangles are constructed such that every data point lies on the vertex of some triangle, but not in the interior or on the edge of any triangle. Then a polynomial is constructed on each of the triangles using function values and partial derivative values at the vertices of the triangulation and possibly additional points. Some of these values have to be estimated. Usually we require that the interpolant is smooth in some way.

Thus it is not trivial to construct polynomials such that they satisfy some global differentiability property. As an example we refer to the Powell–Sabin element. The Delaunay triangulation might be used in two dimensions for example. Fast methods for constructing triangulations in higher dimensions are not known and this limits this technique to two or three dimensions.

This dissertation considers a class of interpolants, known as radial basis function interpolants. They are introduced in the next chapter and some of their properties that will be useful later are established there.

1.2 Contents of the thesis

Radial basis function interpolation has the disadvantage that the matrix is dense. Computing s^* by direct means would require $\mathcal{O}(n^3)$ operations. Thus the work may become prohibitive for large n. Hence the need for fast iterative methods arises.

We begin this thesis by describing interpolation by radial basis function and show some of their properties which are important to our analysis.

We then first consider the algorithm developed by Beatson, Goodsell and Powell, which will be called Algorithm A here. The first section of the third chapter describes the ideas which lead to this algorithm and the algorithm itself. Numerical experiments were very promising, but no proof of convergence existed. The beautiful proof of convergence, which is the subject of the next section, still excites the author. This work was done in collaboration with Mike Powell. The semi-norm $\|\cdot\|_\theta$ which will be introduced in Section 2.2 is the key to the understanding of the convergence properties of Algorithm A. In every stage of each iteration the semi-norm of the difference of the required interpolant s^* and the current approximation s is minimized in

a way to give a better approximation. It still fascinates the author that such a beautiful theory appears for an algorithm whose development was based on numerical experiments.

Then we show that Algorithm A is equivalent to solving a certain symmetric and positive definite system of equations by Gauss–Seidel iterations. This system can be obtained from the original system by preconditioning it from the left and from the right with certain matrices. We briefly mention other choices of preconditioners such as suggested by Powell (1996) for thin plate spline interpolation. Having established the system of equations which is solved by Gauss–Seidel iterations, it is straightforward to find the iteration matrix by which the vector of residuals $f_i - s(\underline{x}_i)$, $i = 1, \ldots, n$, at the beginning of an iteration is multiplied to give the vector of residuals after the iteration, where s denotes the current approximation to s^*. The spectral radius of this iteration matrix gives a measurement for the speed of convergence.

We conclude the third chapter by considering Algorithm A as a linear operator acting on a certain finite dimensional linear space equipped with a semi-inner product. Some properties of the linear operator are established. These give more insight into the algorithm and allow us to apply the Krylov subspace method described in Chapter 6 to Algorithm A.

The next three chapters consider improvements to Algorithm A and new techniques which arose from the insights gained from the analysis of Algorithm A. One drawback of Algorithm A is that the residuals $f_i - s(\underline{x}_i)$, $i = 1, \ldots, n$, have to be updated at several stages during each iteration. This can be very time consuming. An obvious way to speed the algorithm up is to delay the updates till the end of each iteration. This leads to a new algorithm, called Algorithm B, which is the subject of Chapter 4. We show

that this method is equivalent to solving the aforementioned symmetric and positive definite system by Jacobi iterations. Algorithm B can also be viewed as a linear operator acting on the aforementioned linear subspace and it has some properties in common with the linear operator associated with Algorithm A. It has the additional property that it is self-adjoint. Unfortunately, however, Algorithm B has the disadvantage that it diverges in certain cases.

Chapter 5 considers two more techniques. The idea of minimizing the semi-norm leads to a new technique using line searches. By adding a line search to Algorithm B, we can ensure convergence. The conjugate gradient technique is an established method to solve symmetric and positive definite systems. We apply it to the aforementioned preconditioned system and establish certain properties of the preconditioners. The conjugate gradient technique is considered in detail here, because we show later that the Krylov subspace technique applied to Algorithm B is equivalent to the preconditioned conjugate gradient method.

The next chapter is the culmination of this work. The Krylov subspace technique developed from the idea to use orthogonal search directions. It can improve the convergence properties of any given algorithm provided it fulfils certain criteria. We show how it is applied to Algorithm A and B in particular. The Krylov subspace technique ensures that after the ℓ-th iteration, $\ell = 1, 2, \ldots$, the current approximation is a best approximation to s^* from a certain ℓ-dimensional subspace of the aforementioned linear space.

Chapter 7 finally considers some numerical experiments and compares the different techniques presented in this thesis both in two and three dimensions.

To show the development of ideas is important to the author. Analysing the concepts behind the original method gives rise to new, improved techniques. Mathematical insight gives us a beautiful and useful theory to work

with in practice.

Chapter 2

Radial basis functions

2.1 The radial basis function method

Radial basis function methods provide interpolants to function values given at irregularly positioned points for any value d, i.e. they can be applied in any dimension. Often these interpolants are excellent approximations to the underlying function. This makes these techniques very attractive. Franke (1982) compares some thirty interpolation methods, including radial basis functions. Compared to other tested methods, radial basis function techniques obtain excellent accuracy when interpolating scattered data. The range of fields in which radial basis function methods are used is very large, including geophysics, signal processing, meteorology, orthopaedics, pattern recognition and computational fluid dynamics, for instance (Hardy, 1990). The amount of data which needs to be processed gets also larger and thus the demand for faster methods grows.

We are given a fixed univariate, continuous function $\phi : \mathbb{R}_{\geq 0} \to \mathbb{R}$, which is called a "radial basis function". A radial basis function interpolant s^* is a

linear combination of translates of the function $\underline{x} \to \phi(\| \underline{x} \|_2)$, $\underline{x} \in \mathbb{R}^d$, where here and throughout $\| \cdot \|_2$ denotes the Euclidean norm. Written explicitly, s^* has the form

$$s^*(\underline{x}) = \sum_{j=1}^{n} \lambda_j^* \phi(\| \underline{x} - \underline{x}_j \|_2), \qquad \underline{x} \in \mathbb{R}^d, \tag{2.1.1}$$

The function $\underline{x} \to \phi(\| \underline{x} \|_2)$, $\underline{x} \in \mathbb{R}^d$, is "radially symmetric", since it clearly depends only on the length of the vector $\underline{x}$ measured in the Euclidean norm. If we define Φ to be the $n \times n$ matrix that has the elements $\Phi_{ij} = \phi(\| \underline{x}_i - \underline{x}_j \|_2)$, $\underline{f}$ to be the vector $(f_1, \ldots, f_n)^T$ in $\mathbb{R}^n$ whose elements are the right hand sides of the interpolation equations (1.1.1), and $\underline{\lambda}^*$ to be the vector $(\lambda_1^*, \ldots, \lambda_n^*)^T$ in $\mathbb{R}^n$, then the interpolation equations (1.1.1) provide the linear system

$$\Phi \underline{\lambda}^* = \underline{f}. \tag{2.1.2}$$

For several important choices of ϕ, the matrix Φ is invertible under rather mild conditions on the positions of the interpolation points $\underline{x}_1, \ldots, \underline{x}_n$.

The matrix Φ associated with the Gaussian radial basis function $\phi(r) = \exp(-cr^2)$, where c is a positive constant, is positive definite, if all the data points are distinct. If the distances $\| \underline{x}_j - \underline{x}_k \|_2$, $j \neq k$, are very large, the positive definiteness follows from the diagonal dominance. Otherwise, we can express the (j,k)-entry of Φ as

$$e^{-c\|\underline{x}_j - \underline{x}_k\|_2^2} = \left(\frac{c}{\pi}\right)^{d/2} \int_{\underline{t} \in \mathbb{R}^d} e^{-c\,\|\,\underline{t}\,\|_2^2}\, e^{2ci\,\underline{t}^T \underline{x}_j}\, e^{-2ci\,\underline{t}^T \underline{x}_k} d\underline{t}. \tag{2.1.3}$$

Let v_j, $j = 1, \ldots, n$, be the components of a general vector $\underline{v} \in \mathbb{R}^n$. Then the above identity provides the formulae

$$\begin{aligned} \underline{v}^T \Phi\, \underline{v} &= \left(\frac{c}{\pi}\right)^{d/2} \sum_{j,k=1}^{n} v_j\, v_k \int_{\underline{t} \in \mathbb{R}^d} e^{-c\,\|\,\underline{t}\,\|_2^2}\, e^{2ci\,\underline{t}^T \underline{x}_j}\, e^{-2ci\,\underline{t}^T \underline{x}_k} d\underline{t} \\ &= \left(\frac{c}{\pi}\right)^{d/2} \int_{\underline{t} \in \mathbb{R}^d} e^{-c\,\|\,\underline{t}\,\|_2^2} \left| \sum_{j=1}^{n} v_j e^{2ci\,\underline{t}^T \underline{x}_j} \right|^2 d\underline{t}, \end{aligned} \tag{2.1.4}$$

which yields $\underline{v}^T \Phi \underline{v} \geq 0$, because the final integrand is nonnegative. Since the integrand is a continuous function of $\underline{t} \in \mathbb{R}^d$, $\underline{v}^T \Phi \underline{v}$ vanishes only if the integrand is identically zero, which implies

$$\sum_{j=1}^{n} v_j \, e^{\,2ci\,\underline{t}^T \underline{x}_j} = 0, \qquad \text{for all } \underline{t} \in \mathbb{R}^d. \tag{2.1.5}$$

We can choose $\alpha \geq 1$ such that the distances $\| \alpha \underline{x}_j - \alpha \underline{x}_k \|_2$, $j \neq k$, are large enough so that the matrix Φ_α with entries $(\Phi_\alpha)_{ij} = e^{-c\|\alpha \underline{x}_j - \alpha \underline{x}_k\|_2^2}$ is diagonally dominant. Now equation (2.1.5) implies

$$\underline{v}^T \Phi_\alpha \underline{v} = \left(\frac{c}{\pi}\right)^{d/2} \int_{\underline{t} \in \mathbb{R}^d} e^{-c\,\|\,\underline{t}\,\|_2^2} \left| \sum_{j=1}^{n} v_j \, e^{\,2ci\,\alpha \underline{t}^T \underline{x}_j} \right|^2 d\underline{t} = 0, \tag{2.1.6}$$

from which $\underline{v} = 0$ follows, since Φ_α is positive definite. Hence expression (2.1.4) vanishes only if $\underline{v} = 0$, so Φ is positive definite.

For practical purposes, there are reasons to avoid this radial basis function and it is therefore not considered in this thesis. For example, it is very sensitive to the choice of the constant c as Franke found in 1982. On the other hand its smoothness, rapid decay and probabilistic interpretation induces many to use it in spite of its drawbacks.

Next we consider the choice $\phi(r) = (r^2 + c^2)^{1/2}$, $r \geq 0$, where c is a positive constant. It is known as the "multiquadric radial function" and it is highly useful in practice. By setting $c = 0$, we obtain the "linear radial function" $\phi(r) = r$, $r \geq 0$. The matrix Φ associated with these two radial basis functions is nonsingular for all choices of d and n provided that the data points are all different and that $n \geq 2$ if $c = 0$. We can express the entries of Φ by the formula

$$\Phi_{jk} = \frac{1}{2\sqrt{\pi}} \int_0^\infty \alpha^{-3/2} \left(1 - e^{-\alpha c^2} e^{-\alpha \,\|\, \underline{x}_j - \underline{x}_k \,\|_2^2}\right) d\alpha. \tag{2.1.7}$$

Let $\underline{v}$ be any nonzero vector in $\mathbb{R}^n$ whose components sum to zero. Thus $\underline{v}$ lies in an $(n-1)$-dimensional subspace of $\mathbb{R}^n$, and we can write

$$\begin{aligned} \underline{v}^T \Phi \, \underline{v} &= \frac{1}{2\sqrt{\pi}} \int_0^\infty \sum_{j,k=1}^{n} v_j v_k \, \alpha^{-3/2} \, (1 - e^{-\alpha c^2} e^{-\alpha \| \underline{x}_j - \underline{x}_k \|_2^2}) \, d\alpha \\ &= \frac{-1}{2\sqrt{\pi}} \int_0^\infty \alpha^{-3/2} e^{-\alpha c^2} \left[\sum_{j,k=1}^{n} v_j v_k e^{-\alpha \| \underline{x}_j - \underline{x}_k \|_2^2} \right] d\alpha. \end{aligned} \tag{2.1.8}$$

From the analysis of the Gaussian radial basis function we know that the term inside the square brackets is positive for $0 < \alpha < \infty$. Thus the strict inequality $\underline{v}^T \Phi \underline{v} < 0$ follows. It is now possible to deduce that Φ has $n-1$ negative eigenvalues. Furthermore, the trace of Φ is nonnegative and hence Φ has one positive eigenvalue too. Therefore the matrix is nonsingular.

The multiquadric radial basis function has beautiful polynomial reproduction properties discovered by Buhmann (1990). The main result is that the degree of polynomials reproduced by interpolation on an infinite regular grid is $d+1$, so it is actually an increasing function of the dimension d.

In practice, the choice of the parameter c gives some difficulties. Franke (1982) tested the multiquadric basis function on distributions of data points in two dimensions such that the smallest and the largest nearest neighbour distances did not differ too much. He obtained excellent results when c was close to the average distance between nearest neighbours, so he recommends this choice for c. Thus the constant c is made suitable for the interpolation problem. We do not consider the multiquadric radial function in this thesis, but the linear radial basis function is tested.

Another interesting choice for the radial basis function is the "inverse multiquadric radial function" $\phi(r) = (r^2 + c^2)^{-1/2}$, $r \geq 0$, where c is a positive constant. Again the entries of Φ can be expressed using an integral,

$$\Phi_{jk} = \pi^{-1/2} \int_0^\infty \alpha^{-1/2} e^{-\alpha c^2} e^{-\alpha \| \underline{x}_j - \underline{x}_k \|_2^2} d\alpha. \tag{2.1.9}$$

Then for any nonzero vector $\underline{v} \in \mathbb{R}^n$ we can write

$$\underline{v}^T \Phi \, \underline{v} = \pi^{-1/2} \int_0^\infty \alpha^{-1/2} e^{-\alpha c^2} \left[\sum_{j,k=1}^{n} v_j v_k e^{-\alpha \| \underline{x}_j - \underline{x}_k \|_2^2} \right] d\alpha. \qquad (2.1.10)$$

The analysis of the Gaussian radial function shows that the term in square brackets is positive for $0 < \alpha < \infty$, if all the data points are distinct. Hence we can deduce that Φ is positive definite for this choice of radial function.

The inverse multiquadric function can provide excellent approximations (Franke, 1982). As for the multiquadric radial function, the choice of c may be difficult. This function was not included in the experiments.

There are radial basis functions for which the matrix Φ is not always invertible. One example is the "thin plate spline basis function" $\phi(r) = r^2 \log r$ which was introduced by Duchon (1975,1976). If one data point lies at the centre of the unit sphere and the others are distinct points on the unit sphere, then one row and column of Φ consist entirely of zeros and thus Φ is singular. Fortunately, it is possible to remove this difficulty by augmenting (2.1.1) by adding a polynomial of degree at least one. The interpolant s^* then has the form

$$s^*(\underline{x}) = \sum_{j=1}^{n} \lambda_j^* \phi(\| \underline{x} - \underline{x}_j \|_2) + p^*(\underline{x}), \qquad \underline{x} \in \mathbb{R}^d, \qquad (2.1.11)$$

where p^* is at least a linear polynomial. This approach is not limited to the thin plate spline radial function. Every radial basis function approximation can be augmented by a polynomial term of degree at most m, i.e. it lies in the space $\Pi_m(\mathbb{R}^d)$, for some fixed nonnegative integer m. As we will see later this augmentation is very important for our purposes. We can view the translates of the function ϕ and the polynomials in $\Pi_m(\mathbb{R}^d)$ as the bricks with which we build our interpolant s^*. Suitable values of m will be given.

From (2.1.11) one sees that s^* depends linearly on the n real coefficients λ_j^*, $j = 1, \ldots, n$. The other part of s^*, the polynomial p^*, depends linearly on M parameters, where $M = \binom{d+m}{d}$ is the dimension of $\Pi_m(\mathbb{R}^d)$. Thus s^* depends on $n + M$ parameters, but the interpolation equations (1.1.1) give only n restrictions. It is usual to take up the remaining degrees of freedom by imposing on the coefficients λ_j^* the orthogonality conditions

$$\sum_{j=1}^{n} \lambda_j^* q(\underline{x}_j) = 0, \qquad \forall\, q \in \Pi_m(\mathbb{R}^d). \tag{2.1.12}$$

There are also some other approaches, for example minimizing the sum of the squares of the coefficients λ_j^*, $j = 1, \ldots, n$. Conditions (2.1.12), however, will be used throughout this dissertation.

If a basis of $\Pi_m(\mathbb{R}^d)$ is chosen, then the above conditions (2.1.12) and the interpolation equations (1.1.1) can be written as an $(n+M)\times(n+M)$ linear system of equations. Specifically, let $p_1, \ldots, p_M$ be a basis of $\Pi_m(\mathbb{R}^d)$ and let P be the $n \times M$ matrix whose i-th row is $(p_1(\underline{x}_i) \cdots p_M(\underline{x}_i))$. If $\underline{\lambda}^*$ is the vector $(\lambda_1^*, \ldots, \lambda_n^*)^T$ in $\mathbb{R}^n$, then the additional constraints (2.1.12) can be expressed in the form

$$P^T \underline{\lambda}^* = 0. \tag{2.1.13}$$

Thus $\underline{\lambda}^*$ lies in the null space of P^T. If $\underline{c}^*$ is the vector in $\mathbb{R}^M$ whose components are the coefficients of the required polynomial

$$p^*(\underline{x}) = c_1^* p_1(\underline{x}) + \cdots + c_M^* p_M(\underline{x}), \qquad \underline{x} \in \mathbb{R}^d, \tag{2.1.14}$$

then the interpolation equations (1.1.1) are in matrix form

$$\Phi \underline{\lambda}^* + P \underline{c}^* = \underline{f}. \tag{2.1.15}$$

Combining (2.1.13) and (2.1.15), we obtain

$$\left(\begin{array}{c|c} \Phi & P \\ \hline P^T & 0 \end{array}\right) \left(\begin{array}{c} \underline{\lambda}^* \\ \hline \underline{c}^* \end{array}\right) = \left(\begin{array}{c} \underline{f} \\ \hline 0 \end{array}\right). \tag{2.1.16}$$

It is important to ask whether the matrix in (2.1.16) is nonsingular. As we will see, this question was answered by Micchelli (1986). To explain this point further we need the concept of "conditional definiteness".

Definition 2.1.1 (conditional definiteness) A function $\theta : \mathbb{R}^d \to \mathbb{R}$ with $\theta(-\underline{x}) = \theta(\underline{x})$, $\underline{x} \in \mathbb{R}^d$, is conditionally positive (or negative) definite of order m on $\mathbb{R}^d$, if, for all sets $X = \{\underline{x}_1, \ldots, \underline{x}_n\} \subset \mathbb{R}^d$ with n distinct points and all nonzero vectors $\underline{\lambda}^* = (\lambda_1^*, \ldots, \lambda_n^*)^T \in \mathbb{R}^n$ subject to the conditions (2.1.12), or equivalently (2.1.13), the quadratic form

$$\sum_{j=1}^{n} \sum_{k=1}^{n} \lambda_j^* \lambda_k^* \theta(\underline{x}_j - \underline{x}_k) \tag{2.1.17}$$

is positive (or negative). Therefore, if (2.1.12) holds and (2.1.17) is zero, then $\underline{\lambda}^* = 0$.

For radial basis functions with $\theta(\underline{x}) = \phi(\| \underline{x} \|_2)$, $\underline{x} \in \mathbb{R}^d$, (2.1.17) simplifies to $\underline{\lambda}^{*T} \Phi \underline{\lambda}^*$. The term "conditionally" is used, because we do not require the quadratic form (2.1.17) to be positive (or negative) for all nonzero vectors in $\mathbb{R}^n$. Attention is restricted to vectors of coefficients that lie in the null space of P^T.

We choose ϕ such that the function $\theta(\underline{x}) = \phi(\| \underline{x} \|_2)$, $\underline{x} \in \mathbb{R}^d$, is conditional positive (or negative) definite of order m. This property is very important since it enables us to define a certain semi-inner product which is the main ingredient in the analysis of the following chapters. Thus we ensure that $\underline{\lambda}^{*T} \Phi \underline{\lambda}^*$ is positive (or negative) for all nonzero vectors $\underline{\lambda}^*$ satisfying (2.1.13). We then refer to Φ as being conditionally positive (or negative) definite. We further ensure that the positions of the interpolation points $\underline{x}_1, \ldots, \underline{x}_n$ cause P to have its maximum rank of M. That is equivalent to the requirement that the data points do not lie in the zero set of a nonzero

polynomial of degree at most m, i.e. we require that X is polynomially unisolvent – which is a rather mild condition and which implies that m is small enough to ensure $M \leq n$. It is now easy to deduce that the matrix in (2.1.16) is nonsingular.

We need to show that if $\underline{f} = 0$, then it follows that $\underline{\lambda}^*$ and $\underline{c}^*$ are zero. Multiplying (2.1.15) by $\underline{\lambda}^{*T}$ from the left and using $\underline{\lambda}^{*T} P = 0$, we get $\underline{\lambda}^{*T} \Phi \underline{\lambda}^* = 0$ which implies that $\underline{\lambda}^*$ equals zero, since $\underline{\lambda}^{*T} \Phi \underline{\lambda}^* > 0$ (or < 0) for all nonzero vectors $\underline{\lambda}^*$ satisfying (2.1.13). Inserting $\underline{\lambda}^* = \underline{f} = 0$ in (2.1.15) gives $\underline{c}^* = 0$, since P has full rank.

Micchelli (1986) establishes a link between conditionally definite functions which are derived from radially symmetric functions, and strictly completely monotonic derivatives.

Definition 2.1.2. (complete monotonicity) An infinitely differentiable function f defined on $\mathbb{R}_{\geq 0}$ is strictly completely monotonic if and only if each derivative in the infinite sequence of derivatives $f^{(0)}, f^{(1)}, f^{(2)}, \ldots$ has no zeros and if these derivatives alternate in sign.

Micchelli proves the following significant result:

Theorem 2.1.1. (Micchelli's Theorem) Let the function ψ be defined by $\psi(r) = \phi(r^{1/2})$, $r > 0$. Let further m_0 be the least integer such that the $(m_0 + 1)$-th derivative of ψ is strictly completely monotonic, then the inequality

$$(-1)^{m_0} \underline{\lambda}^{*T} \Phi \underline{\lambda}^* < 0 \tag{2.1.18}$$

holds for every nonzero vector $\underline{\lambda}^*$ that satisfies $P^T \underline{\lambda}^* = 0$. Hence Φ is conditionally positive definite if m_0 is odd and conditionally negative definite if m_0 is even ($m_0 \neq 0$). For $m_0 = 0$, Φ is negative definite. Considering the function $\theta(\underline{x}) = \phi(\| \underline{x} \|_2)$, $\underline{x} \in \mathbb{R}^d$, it follows that it is conditionally definite.

We choose m to be greater or equal to m_0. For example, in the case of the thin plate spline function $\phi(r) = r^2 \log r$, $\psi(r)$ is the function $\frac{1}{2} r \log r$ with derivatives $\psi^{(1)}(r) = \frac{1}{2} \log r + \frac{1}{2}$ and $\psi^{(k)}(r) = \frac{1}{2}(-1)^k (k-2)!\; r^{1-k}$, $k \geq 2$. Although the first derivative $\psi^{(1)}$ changes sign, the second derivative $\psi^{(2)}$ is strictly completely monotonic, so we require m to be greater or equal one. Therefore we include at least a linear polynomial p^* in expression (2.1.11), which ensures that the linear system (2.1.16) is nonsingular. In this case the matrix Φ is conditionally positive definite. The second derivative of ψ is also completely monotonic in the case of the cubic radial basis function, $\phi(r) = r^3$. For the multiquadric and linear radial basis functions the first derivative of ψ is strictly completely monotonic and thus a constant polynomial is added to achieve conditional negative definiteness of Φ, while if ψ is defined for the Gaussian or inverse multiquadric function, then ψ itself is strictly completely monotonic. This implies that in latter two cases the matrix Φ is positive definite - as we have seen already - and no polynomial term is added.

Conditional definiteness is very important for our purposes. Therefore we always add a suitable polynomial to achieve the conditional definiteness of Φ, which is used in the next section to define the important semi-inner product on which our analysis is based.

Considering the more general case of interpolation using shifts of a conditionally positive (or negative) definite basis function $\theta : \mathbb{R}^d \to \mathbb{R}$ of order m, we let Θ be the $n \times n$ matrix with entries $\Theta_{ij} = \theta(\underline{x}_i - \underline{x}_j)$. Since $\theta(-\underline{x}) = \theta(\underline{x})$, $\underline{x} \in \mathbb{R}^d$, holds, the matrix Θ is symmetric. In this general case we have the system of equations

$$\left(\begin{array}{c|c} \Theta & P \\ \hline P^T & 0 \end{array}\right) \left(\begin{array}{c} \underline{\lambda}^* \\ \hline \underline{c}^* \end{array}\right) = \left(\begin{array}{c} \underline{f} \\ \hline 0 \end{array}\right), \tag{2.1.19}$$

which is nonsingular if P has rank M, where M is the dimension of $\Pi_m(\mathbb{R}^d)$,

provided that $M \leq n$, since θ is conditionally definite. We also refer to Θ as being conditionally positive (or negative) definite.

We let

$$\left(\begin{array}{c|c} \Lambda & B \\ \hline B^T & C \end{array}\right) \tag{2.1.20}$$

be the inverse of the matrix given in (2.1.19) and we obtain the formula

$$\underline{\lambda}^* = \Lambda \underline{f}. \tag{2.1.21}$$

Setting $\underline{f}$ to the k-th coordinate vector $\underline{e}_k$, we obtain the coefficient vector $\underline{\lambda}_k = (\lambda_{k1}, \ldots, \lambda_{kn})^T$ of the Lagrange function

$$\chi_k(\underline{x}) = \sum_{j=1}^{n} \lambda_{kj}\, \theta(\underline{x} - \underline{x}_j) + \sum_{j=1}^{M} c_{kj} p_j(\underline{x}), \qquad \underline{x} \in \mathbb{R}^d, \tag{2.1.22}$$

that satisfies the Lagrange conditions $\chi_k(\underline{x}_i) = \delta_{ik}$, $i = 1, \ldots, n$, where δ_{ik} is the Kronecker delta. Since $\underline{\lambda}_k$ is the k-th column of Λ and since Λ is symmetric, $\lambda_{kj} = \lambda_{jk}$ follows. This property of coefficients of Lagrange functions over the same set of data points will become useful later.

The following properties of Λ are important to several algorithms. For example they are put to good use by Dyn and Levin (1981, 1983).

The symmetry of Θ and the definition of Λ imply that Λ is also symmetric. Multiplying the two matrices given in (2.1.19) and (2.1.20) gives

$$\left(\begin{array}{c|c} \Theta & P \\ \hline P^T & 0 \end{array}\right) \left(\begin{array}{c|c} \Lambda & B \\ \hline B^T & C \end{array}\right) = \left(\begin{array}{c|c} \Theta\Lambda + PB^T & \Theta B + PC \\ \hline P^T\Lambda & P^T B \end{array}\right) = \left(\begin{array}{c|c} I_n & 0 \\ \hline 0 & I_M \end{array}\right), \tag{2.1.23}$$

where I_k denotes the $k \times k$ identity matrix, $k = n, M$. Inserting equation (2.1.21) on the right of $\underline{\lambda}^{*T}\Theta\underline{\lambda}^*$ and using the remark $\Theta\Lambda = I_n - PB^T$, we find

$$\underline{\lambda}^{*T}\Theta\underline{\lambda}^* = \underline{\lambda}^{*T}\underline{f} - \underline{\lambda}^{*T} P B^T \underline{f}. \tag{2.1.24}$$

The second term vanishes, since $\underline{\lambda}^*$ lies in the null space of P^T. Therefore (2.1.21) implies $\underline{\lambda}^{*T}\Theta\underline{\lambda}^* = \underline{f}^T\Lambda\,\underline{f}$. It follows that $\underline{f}^T\Lambda\,\underline{f}$ inherits the constant sign of $\underline{\lambda}^{*T}\Theta\underline{\lambda}^*$. Thus all the eigenvalues of the symmetric matrix Λ are nonnegative (or nonpositive). Further, $\underline{f}^T\Lambda\,\underline{f}$ is zero if and only if the interpolation conditions (1.1.1) allow s^* to be a polynomial of degree at most m. This is the case if $\underline{f}$ lies in the M-dimensional subspace of $\mathbb{R}^n$ spanned by the columns of P, and then the vector of coefficients $\underline{\lambda}^*$ is zero. Therefore Λ has M eigenvalues of zero, the remaining $n-M$ eigenvalues being positive (or negative). System (2.1.19) causes $\underline{\lambda}^* = \Lambda\underline{f}$ to satisfy $P^T\underline{\lambda}^* = 0$ for every $\underline{f} \in \mathbb{R}^n$, which is equivalent to the condition $P^T\Lambda = 0$. The symmetry of Λ yields $\Lambda P = 0$.

The methods of Dyn and Levin (1981, 1983) generate an estimate, $\hat{W}$ say, of Λ, which has the properties of Λ given in the previous paragraph. To be specific, $\hat{W}$ is also an $n \times n$ symmetric matrix with M zero eigenvalues and $n-M$ positive (or negative) ones that satisfies $P^T\hat{W} = 0$. Then $\hat{W}$ is used as a preconditioner in a conjugate gradient method which calculates approximations to the required vector of coefficients $\underline{\lambda}^*$. The basic idea for generating $\hat{W}$ arises from the fact that radial basis functions of the form

$$\phi(r) = \left\{ \begin{array}{ll} r^{2(m+1)-d} & \text{if } d \text{ is odd and } d < 2(m+1) \\ r^{2(m+1)-d}\log r & \text{if } d \text{ is even and } d < 2(m+1) \end{array} \right\}, \qquad (2.1.25)$$

which are known as "polyharmonic splines", are related to the Dirac $\delta(\cdot)$ distribution. Indeed, a multiple of this distribution occurs if the m-th iterated Laplacian operator ∇^m is applied to $\phi(\|\,\underline{x}\,\|)$, $\underline{x} \in \mathbb{R}^d$. Therefore, $\hat{W}$ is constructed from discrete approximations of the iterated Laplacian.

We will also consider preconditioned conjugate gradient methods, but we use a different choice of preconditioner, which arises from the algorithm developed by Beatson, Goodsell and Powell (1995).

Summarising, we have seen that interpolation by radial basis functions leads to a linear system of equations with a conditionally definite submatrix, the system being nonsingular as long as the set of data points X is not in the zero set of a nonzero polynomial of degree at most m. Further useful properties of these interpolants will be established in the next section.

2.2 Semi-inner products

We have mentioned already that we intend to use the conditional definiteness of Φ, or more generally Θ, to define a certain semi-inner product. To introduce it we present a beautiful result by Duchon (1977). Duchon's treatment is somewhat abstract, using sophisticated techniques from distribution theory. We follow an alternative approach introduced by Powell (1992).

One of the most popular radial basis function in two dimensions is the thin plate spline basis function $\phi(r) = r^2 \log r$ for $r > 0$ and $\phi(0) = 0$, with an added linear polynomial, which gives the form

$$s^*(\underline{x}) = \sum_{j=1}^{n} \lambda_j^* \parallel \underline{x} - \underline{x}_j \parallel_2^2 \log \parallel \underline{x} - \underline{x}_j \parallel_2 + \text{ linear polynomial}, \qquad \underline{x} \in \mathbb{R}^2. \tag{2.2.1}$$

The additional constraints (2.1.12) can be written as

$$\sum_{j=1}^{n} \lambda_j^* = 0 \qquad \text{and} \qquad \sum_{j=1}^{n} \lambda_j^* \, \underline{x}_j = 0. \tag{2.2.2}$$

The points $\underline{x}_i$, $i = 1, \ldots, n$, must not lie in the zero set of a nonzero linear polynomial. In other words, they must not be collinear.

One reason for the success of thin plate spline interpolation is that s^* is the solution of an optimal recovery problem (Duchon, 1977). Specifically, s^* provides the least possible value of the semi-inner product (s^*, s^*) subject to

the interpolation conditions (1.1.1), where for functions f and g with square integrable second derivatives the semi-inner product (f, g) has the value

$$(f,g) = \frac{1}{8\pi}\int\int_{\mathbb{R}^2} \frac{\partial^2 f(x,y)}{\partial x^2}\frac{\partial^2 g(x,y)}{\partial x^2} + 2\,\frac{\partial^2 f(x,y)}{\partial x\,\partial y}\frac{\partial^2 g(x,y)}{\partial x\,\partial y} + \frac{\partial^2 f(x,y)}{\partial y^2}\frac{\partial^2 g(x,y)}{\partial y^2}\,dx\,dy, \tag{2.2.3}$$

which vanishes if either f or g are linear polynomials, x and y being the components of $\underline{x} \in \mathbb{R}^2$. The thin plate spline s^* has square integrable second derivatives, since for large $\| \underline{x} \|_2$ the moduli of the second derivatives of s^* are bounded above by a multiple of $\| \underline{x} \|_2^{-2}$, which can be shown be expanding the second derivatives of the function $\frac{1}{2} \| \underline{x} - \underline{x}_j \|_2^2 \log \| \underline{x} - \underline{x}_j \|_2^2$ in inverse powers of $\| \underline{x} \|_2$, using the identity

$$\| \underline{x} - \underline{x}_j \|_2^2 = \| \underline{x} \|_2^2 \left[1 - \frac{2\underline{x}^T \underline{x}_j}{\| \underline{x} \|_2^2} + \frac{\| \underline{x}_j \|_2^2}{\| \underline{x} \|_2^2} \right] \tag{2.2.4}$$

and the constraints (2.2.2). Indeed, we obtain expansions of the second derivatives of s^* in inverse powers of $\| \underline{x} \|_2$, where constant terms, terms increasing in modulus or terms decreasing in modulus slower than $\| \underline{x} \|_2^{-2}$ cancel due to the additional constraints (2.2.2).

Minimizing (s^*, s^*) for the semi-inner product (2.2.3), subject to the interpolation conditions (1.1.1), means that the interpolant minimizes the bending energy of the surface, which can be regarded as the "smoothest" surface through the data.

We find a useful form of (f, g), by applying integration by parts to the integral (2.2.3). There are no contributions from the boundary conditions at infinity. Therefore integrating the second derivatives of f twice and differentiating the second derivatives of g twice provides the formula

$$(f,g) = \frac{1}{8\pi}\int\int_{\mathbb{R}^2} f(\underline{x})\,\nabla^4 g(\underline{x})\,dx\,dy, \tag{2.2.5}$$

where $\nabla^4 = (\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2})^2$.

To prove the optimal recovery property, we consider $\nabla^4\theta(\underline{x})$ where $\theta(\underline{x}) = \phi(\| \underline{x} \|_2) = \| \underline{x} \|_2^2 \log \| \underline{x} \|_2$, $\underline{x} \in \mathbb{R}^2$.

Lemma 2.2.1. *The function $\theta(\underline{x}) = \| \underline{x} \|_2^2 \log \| \underline{x} \|_2$, $\underline{x} \in \mathbb{R}^2$, has the derivatives*

$$\nabla^2\theta(\underline{x}) = 4 + 4\log \| \underline{x} \|_2 \qquad \text{and} \qquad \nabla^4\theta(\underline{x}) = 8\pi\delta(\underline{x}), \qquad \underline{x} \in \mathbb{R}^2, \quad (2.2.6)$$

where $\delta(\underline{x})$, $\underline{x} \in \mathbb{R}^2$, is the delta function.

Proof: When $r > 0$, the radial symmetry of θ, setting $r = \| \underline{x} \|_2$, supplies the derivatives

$$\begin{aligned} \nabla^2\theta(\underline{x}) &= \left(r^{-1}\frac{d}{dr} + \frac{d^2}{dr^2} \right) (r^2 \log r) \\ &= r^{-1}(2r\log r + r) + (2\log r + 3) \\ &= 4 + 4\log r \end{aligned} \quad (2.2.7)$$

and

$$\nabla^4\theta(\underline{x}) = \left(r^{-1}\frac{d}{dr} + \frac{d^2}{dr^2} \right) (4 + 4\log r) = 4r^{-2} - 4r^{-2} = 0. \quad (2.2.8)$$

Therefore it remains to establish the behaviour of $\nabla^4\theta(\underline{x})$ at $\underline{x} = 0$.

Let $f : \mathbb{R}^2 \to \mathbb{R}$ be a four times continuously differentiable function that has the properties $f(\underline{x}) \equiv 1$ for $\| \underline{x} \|_2 \leq 2$ and $f(\underline{x}) \equiv 0$ for $\| \underline{x} \|_2 \geq 3$. Such functions are known to exist and are called bump functions. We consider the semi-inner product (f, θ), and can restrict the range of integration to the square $[-3, 3] \times [-3, 3]$. The zero boundary conditions of f make integration by parts straightforward and, since $\nabla^4\theta(\underline{x}) = 0$, $\underline{x} \neq 0$, and $f(0) = 1$, we deduce

$$(f, \theta) = \frac{1}{8\pi} \int \int_{\mathbb{R}^2} \nabla^4\theta(\underline{x})\, d\underline{x}. \quad (2.2.9)$$

The second derivatives of f vanish for $\| \underline{x} \|_2 < 2$, since $f(\underline{x}) \equiv 1$ in this region. Therefore, letting $\hat{\theta}$ be a smooth function that is free of singularities and that coincides with θ for $\| \underline{x} \|_2 \geq 1$, it satisfies $(f, \theta) = (f, \hat{\theta})$, which gives the equation

$$\int \int_{\mathbb{R}^2} \nabla^4 \theta(\underline{x}) \, d\underline{x} = 8\pi(f, \theta) = 8\pi(f, \hat{\theta}) = \int \int_{\mathbb{R}^2} \nabla^4 \hat{\theta}(\underline{x}) \, d\underline{x}. \tag{2.2.10}$$

Now let $\mathcal{D}$ be the disc $\{\underline{x} : \| \underline{x} \|_2 \leq 2\}$. Since $\nabla^4 \hat{\theta}(\underline{x}) = \nabla^4 \theta(\underline{x}) = 0$, $\| \underline{x} \|_2 \geq 1$, we can restrict the range of integration to $\mathcal{D}$. We chose $\hat{\theta}$ such that $\nabla^4 \hat{\theta}(\underline{x})$ is continuous. Thus the divergence theorem provides the identity

$$\begin{aligned} \int \int_{\mathcal{D}} \nabla^4 \hat{\theta}(\underline{x}) \, d\underline{x} &= \int \int_{\mathcal{D}} \nabla^2 (\nabla^2 \hat{\theta}(\underline{x})) \, d\underline{x} \\ &= \oint \underline{n}^T \underline{\nabla} (\nabla^2 \hat{\theta}(\underline{x})) ds, \end{aligned} \tag{2.2.11}$$

where $\oint$ denotes the integral around the perimeter of $\mathcal{D}$ and where $\underline{n}$ is the outward pointing normal $\underline{x} / \| \underline{x} \|_2$. When $\| \underline{x} \|_2 \geq 1$, which includes the perimeter of $\mathcal{D}$, we have the identity $\nabla^2 \hat{\theta}(\underline{x}) = \nabla^2 \theta(\underline{x}) = 4 + 4 \log \| \underline{x} \|_2$, which implies $\underline{\nabla} (\nabla^2 \hat{\theta}(\underline{x})) = 4 \, \underline{x} / \| \underline{x} \|_2^2$. Thus the integrand $\underline{n}^T \underline{\nabla} (\nabla^2 \hat{\theta}(\underline{x}))$ takes the value $(\underline{x} / \| \underline{x} \|_2)^T (4 \, \underline{x} / \| \underline{x} \|_2^2) = 2$, since the radius of $\mathcal{D}$ is 2. The length of the perimeter of $\mathcal{D}$ is 4π. Thus

$$\int \int_{\mathbb{R}^2} \nabla^4 \theta(\underline{x}) d\underline{x} = 8\pi, \tag{2.2.12}$$

which gives the required result $\nabla^4 \theta(\underline{x}) = 8\pi \delta(\underline{x})$. $\square$

We express the optimal recovery property as a theorem.

Theorem 2.2.2. (Optimal recovery) *Let $s^*(\underline{x})$, $\underline{x} \in \mathbb{R}^2$, be the thin plate spline that interpolates the data f_i at the points $\underline{x}_i$, $i = 1, \ldots, n$, and let $t(\underline{x})$, $\underline{x} \in \mathbb{R}^2$, be any function different from s^* with square integrable*

second derivatives that also satisfies the interpolation equations $t(\underline{x}_i) = f_i$, $i = 1, \ldots, n$. *Then the strict inequality*

$$(s^*, s^*) < (t, t) \tag{2.2.13}$$

holds. Thus s^* *is the unique function with square integrable second derivatives that minimizes the semi-inner product subject to the interpolation conditions (1.1.1).*

Proof: We let g be the function $t - s^*$ which vanishes at the interpolation points $\underline{x}_i$, $i = 1, \ldots, n$. A lower bound on (t, t) is provided by the inequality

$$(t,t) = (g+s^*, g+s^*) = (g,g)+2(g,s^*)+(s^*,s^*) \geq 2(g,s^*)+(s^*,s^*). \tag{2.2.14}$$

Therefore it is sufficient to deduce that (g, s^*) is zero and that, if $(t, t) = (s^*, s^*)$, it follows from $(g, g) = 0$ that $g \equiv 0$ and thus $t = s^*$.

Treating the semi-inner product (g, s^*) by integration by parts, there are no contributions from the boundary conditions at infinity, due to the decay of the second and the third derivatives of $s^*(\underline{x})$ as $\| \underline{x} \|_2 \to \infty$. We have already mentioned that the moduli of the second derivatives of s^* can be bounded above by a multiple of $\| \underline{x} \|_2^{-2}$ for large $\| \underline{x} \|_2$, while the third derivatives of s^* are of magnitude $\| \underline{x} \|_2^{-3}$ for large $\| \underline{x} \|_2$. Thus, following Powell (1992), we deduce the formula

$$\begin{aligned}
(g, s^*) &= \frac{1}{8\pi} \int \int_{\mathbb{R}^2} g(\underline{x}) \, \nabla^4 s^*(\underline{x}) \, dx \, dy \\
&= \frac{1}{8\pi} \int \int_{\mathbb{R}^2} g(\underline{x}) \sum_{i=1}^{n} \lambda_i^* \nabla^4 \phi \left(\| \underline{x} - \underline{x}_i \|_2 \right) dx \, dy \\
&= \int \int_{\mathbb{R}^2} g(\underline{x}) \sum_{i=1}^{n} \lambda_i^* \delta(\underline{x} - \underline{x}_i) \, dx \, dy \\
&= \sum_{i=1}^{n} \lambda_i^* g(\underline{x}_i) = 0,
\end{aligned} \tag{2.2.15}$$

since g vanishes at the interpolation points.

The semi-inner product

$$(g,g) = \frac{1}{8\pi} \int\int_{\mathbb{R}^2} \left[\frac{\partial^2 g(x,y)}{\partial x^2}\right]^2 + 2\left[\frac{\partial^2 g(x,y)}{\partial x\,\partial y}\right]^2 + \left[\frac{\partial^2 g(x,y)}{\partial y^2}\right]^2 dx\,dy \tag{2.2.16}$$

vanishes only if g is a linear polynomial. On the other hand, g is zero at $\underline{x}_i$, $i = 1,\ldots,n$, which are not collinear. Thus, if $(g,g) = 0$, it follows that g is identically zero and hence s^* and t coincide. □

We are now regarding $\underline{x}_i$, $i = 1,\ldots,n$ as fixed. With $\phi(r) = r^2 \log r$, $r > 0$, and $\phi(0) = 0$, let S be the n-dimensional linear space that contains functions of the form

$$s(\underline{x}) = \sum_{j=1}^{n} \lambda_j\, \phi\left(\| \underline{x} - \underline{x}_j \|_2\right) + p(\underline{x}), \qquad \underline{x} \in \mathbb{R}^2, \tag{2.2.17}$$

whose coefficients satisfy

$$\sum_{j=1}^{n} \lambda_j = 0 \qquad \text{and} \qquad \sum_{j=1}^{n} \lambda_j\, \underline{x}_j = 0, \tag{2.2.18}$$

where $p(\underline{x})$, $\underline{x} \in \mathbb{R}^2$, is a linear polynomial. The results of Theorem 2.2.2 show that, because s^* has the form (2.2.1), it is the unique element in S that interpolates the data f_i, $i = 1,\ldots,n$. The semi-inner product defined by (2.2.3) exists on this space S of thin plate splines, since thin plate splines have square integrable second derivatives. Further, if $t(\underline{x}) = \sum_{j=1}^{n} \nu_j$ $\phi(\| \underline{x} - \underline{x}_j \|_2) + q(\underline{x})$ is another function in S, then, by integration by parts analogous to (2.2.15), the semi-inner product has the value

$$\begin{aligned}
(s,t) &= \sum_{i=1}^{n} \nu_i\, s(\underline{x}_i) = \sum_{j=1}^{n} \lambda_j\, t(\underline{x}_j) \\
&= \sum_{j=1}^{n} \lambda_j \left[\sum_{i=1}^{n} \nu_i\, \phi(\| \underline{x}_j - \underline{x}_i \|_2) + q(\underline{x}_j)\right] \\
&= \sum_{j=1}^{n}\sum_{i=1}^{n} \lambda_j\, \nu_i\, \phi(\| \underline{x}_j - \underline{x}_i \|_2) = \underline{\lambda}^T \Phi \underline{\nu}\,,
\end{aligned} \tag{2.2.19}$$

where $\underline{\lambda} = (\lambda_1, \ldots, \lambda_n)^T$ and $\underline{\nu} = (\nu_1, \ldots, \nu_n)^T$.

Unfortunately, other explicit forms of semi-inner products that are minimized by the other radial basis function interpolants are not known generally. We refer the reader to Schaback (1999). However, let $\theta : \mathbb{R}^d \to \mathbb{R}$ be any conditionally definite function of order m and let $X = \{\underline{x}_1, \ldots, \underline{x}_n\} \subset \mathbb{R}^d$ be a fixed set that does not lie in the zero set of a nonzero polynomial of degree at most m. Then, guided by the above paragraph, we define an n-dimensional linear space S_θ which contains functions of the form

$$s(\underline{x}) = \sum_{j=1}^{n} \lambda_j \, \theta(\underline{x} - \underline{x}_j) + p(\underline{x}), \qquad \underline{x} \in \mathbb{R}^d, \tag{2.2.20}$$

where $p \in \Pi_m(\mathbb{R}^d)$, and where the coefficients λ_j, $j = 1, \ldots, n$, are required to satisfy the constraints

$$\sum_{j=1}^{n} \lambda_j \, q(\underline{x}_j) = 0, \qquad \forall \, q \in \Pi_m(\mathbb{R}^d). \tag{2.2.21}$$

On S_θ we define the semi-inner product that is introduced by Schaback (1993). Specifically, letting $t(\underline{x}) = \sum_{i=1}^{n} \nu_i \, \theta(\underline{x} - \underline{x}_i) + q(\underline{x})$, $\underline{x} \in \mathbb{R}^d$, be another function in S_θ, the semi-inner product between s and t is the bilinear form

$$\begin{aligned} (s,t)_\theta &= \sigma_\theta \sum_{j=1}^{n} \lambda_j \, t(\underline{x}_j) = \sigma_\theta \sum_{i=1}^{n} \nu_i \, s(\underline{x}_i) \\ &= \sigma_\theta \sum_{j=1}^{n} \sum_{i=1}^{n} \lambda_j \, \nu_i \, \theta(\underline{x}_j - \underline{x}_i) = \sigma_\theta \, \underline{\lambda}^T \Theta \, \underline{\nu}, \end{aligned} \tag{2.2.22}$$

where σ_θ is chosen to be $+1$ if θ is conditionally positive definite and $\sigma_\theta = -1$ if θ is conditionally negative definite. The semi-inner product vanishes if s or t lie in $\Pi_m(\mathbb{R}^d)$, because then $\underline{\lambda}$ or $\underline{\nu}$ is zero. A simple rule for the semi-inner product is that it is the sum of the coefficients of one function times the function values at the data points of the other function, with a change of sign in the conditionally negative case.

Forming $(s,s)_\theta$ gives $\sigma_\theta \sum_{i,j=1}^n \lambda_i \lambda_j \theta(\underline{x}_i - \underline{x}_j)$, which is nonnegative by the choice of σ_θ. Thus S_θ has the semi-norm

$$\| s \|_\theta = (s,s)_\theta^{1/2}, \qquad s \in S_\theta. \tag{2.2.23}$$

This semi-norm has a fundamental property, taken from Schaback (1993), which can be compared to the optimal recovery property of thin plate splines. We define a larger space $\hat{S}_\theta$ by letting $\hat{s}$ be in $\hat{S}_\theta$ if it can be written in the form

$$\hat{s}(\underline{x}) = \sum_{j=1}^{\hat{n}} \hat{\lambda}_j \, \theta(\underline{x} - \hat{\underline{x}}_j) + p(\underline{x}), \qquad \underline{x} \in \mathbb{R}^d, \tag{2.2.24}$$

where $p \in \Pi_m(\mathbb{R}^d)$ and where the parameters are subject to the usual constraints

$$\sum_{j=1}^{\hat{n}} \hat{\lambda}_j \, q(\hat{\underline{x}}_j) = 0, \qquad \forall \, q \in \Pi_m(\mathbb{R}^d). \tag{2.2.25}$$

Here θ is as before, but $\hat{n}$ can be any finite number and much larger than n and the set $\{\hat{\underline{x}}_i : i = 1, \ldots, \hat{n}\}$ can consist of any points in $\mathbb{R}^d$. This freedom in the set implies that the dimension of $\hat{S}_\theta$ is infinite, but each $\hat{n}$ is finite. Further, the original space S_θ is an n-dimensional subspace of $\hat{S}_\theta$.

We let $\hat{s}$ and $\check{t}$ be the functions (2.2.24) and $\check{t}(\underline{x}) = \sum_{j=1}^{\check{n}} \check{\nu}_j \theta(\underline{x} - \check{\underline{x}}_j) + q(\underline{x})$, $\underline{x} \in \mathbb{R}^d$, respectively, where $\check{n}$ can be different from $\hat{n}$ and the points $\check{\underline{x}}_i$, $i = 1, \ldots, \check{n}$, can be any points in $\mathbb{R}^d$. We let the set $\{\tilde{\underline{x}}_i : i = 1, \ldots, \tilde{n}\}$ be the union of the two sets $\{\hat{\underline{x}}_i : i = 1, \ldots, \hat{n}\}$ and $\{\check{\underline{x}}_i : i = 1, \ldots, \check{n}\}$. We define $\tilde{\lambda}_i$ to equal $\hat{\lambda}_j$ if $\tilde{\underline{x}}_i = \hat{\underline{x}}_j$ for some $j \in [1, \hat{n}]$ and to equal zero otherwise. The coefficient $\tilde{\nu}_i$ equals $\check{\nu}_j$ if $\tilde{\underline{x}}_i = \check{\underline{x}}_j$ for some $j \in [1, \check{n}]$ and is zero otherwise. We then can write $\hat{s}(\underline{x}) = \sum_{j=1}^{\tilde{n}} \tilde{\lambda}_j \, \theta(\underline{x} - \tilde{\underline{x}}_j) + p(\underline{x})$ and $\check{t}(\underline{x}) = \sum_{j=1}^{\tilde{n}} \tilde{\nu}_j \theta(\underline{x} - \tilde{\underline{x}}_j) + q(\underline{x})$, $\underline{x} \in \mathbb{R}^d$. Now the analogue of expression (2.2.22) is

$$(\hat{s}, \check{t}\,)_\theta \;\; = \;\; \sigma_\theta \sum_{j=1}^{\tilde{n}} \tilde{\lambda}_j \, \check{t}(\tilde{\underline{x}}_j) = \sigma_\theta \sum_{i=1}^{\tilde{n}} \tilde{\nu}_i \, \hat{s}(\tilde{\underline{x}}_i)$$

$$= \sigma_\theta \sum_{j=1}^{\tilde{n}} \sum_{i=1}^{\tilde{n}} \tilde{\lambda}_j \, \tilde{\nu}_i \, \theta(\tilde{\underline{x}}_j - \tilde{\underline{x}}_i). \tag{2.2.26}$$

The fundamental property that corresponds to the optimal recovery property for thin plate splines is that, if $\hat{s}$ is any function in $\hat{S}_\theta$ that interpolates the data f_i, $i = 1, \ldots, n$, then $(\hat{s}, \hat{s})_\theta$ is never less then $(s^*, s^*)_\theta$. It is proved as follows.

An element $\hat{s} \in \hat{S}_\theta$ interpolates the data f_i, $i = 1, \ldots, n$ if and only if it can be written in the form $\hat{s} = s^* + \check{t}$, where $s^* \in S_\theta$ is the required interpolant and where $\check{t} \in \hat{S}_\theta$ satisfies $\check{t}(\underline{x}_i) = 0$, $i = 1, \ldots, n$. By introducing zero coefficients if necessary, we let $\check{t}$ have centres at the points $\{\tilde{\underline{x}}_i : i = 1, \ldots, \tilde{n}\} = \{\underline{x}_i : i = 1, \ldots, n\} \cup \{\hat{\underline{x}}_i : i = 1, \ldots, \hat{n}\}$. We consider the semi-inner product

$$(s^*, \check{t}\,)_\theta = \sigma_\theta \sum_{i=1}^{\tilde{n}} \tilde{\lambda}_i^* \, \check{t}(\tilde{\underline{x}}_i), \tag{2.2.27}$$

where $\tilde{\lambda}_i^*$ equals λ_j^* if $\tilde{\underline{x}}_i = \underline{x}_j$ for some $j \in [1, n]$, but otherwise $\tilde{\lambda}_i^*$ is defined to be zero. Thus $\tilde{\lambda}_i^*$ is nonzero only if $\tilde{\underline{x}}_i$ is one of the given interpolation points $\underline{x}_j$, $j = 1, \ldots, n$, and in this case $\check{t}(\tilde{\underline{x}}_i)$ is zero. Thus expression (2.2.27) vanishes, which gives the required bound

$$(\hat{s}, \hat{s})_\theta = (s^* + \check{t}, s^* + \check{t})_\theta = (s^*, s^*)_\theta + (\check{t}, \check{t})_\theta \geq (s^*, s^*)_\theta. \tag{2.2.28}$$

Equality occurs when $(\check{t}, \check{t})_\theta = \sigma_\theta \sum_{i,j=1}^{\tilde{n}} \tilde{\nu}_i \, \tilde{\nu}_j \, \theta(\tilde{\underline{x}}_j - \tilde{\underline{x}}_i)$ is zero. In this case $\tilde{\nu}_i = 0$, $i = 1, \ldots, \tilde{n}$, since θ is conditionally definite, and hence $\check{t}$ is a polynomial of degree at most m. It follows from $\check{t}(\underline{x}_i) = 0$, $i = 1, \ldots, n$, and from the polynomial unisolvency of the set $X = \{\underline{x}_i : i = 1, \ldots, n\}$ that $\check{t}$ is identically zero. Therefore s^* is the unique interpolant in $\hat{S}_\theta$ that minimizes $(s^*, s^*)_\theta$.

We are going to consider iterative methods that construct a sequence of approximations in the space S_θ. The sequence should converge to the desired

interpolant $s^* \in S_\theta$. The next chapter describes one such algorithm. It makes good use of the native semi-inner product defined above.

Chapter 3

Algorithm A

Algorithm A was developed by Beatson, Goodsell and Powell in 1995 after experimenting with several ideas, which included multigrid techniques. The version presented here is based on the implementation that is described by Powell (1997), which was highly successful in numerical experiments. Particular choices of the ordering of the interpolation points, the sets $\mathcal{L}_k$, $k = 1, \ldots, n-q$, which we introduce in the following section, and a stopping condition are specified there. Here a more general description is given. The technique was first developed for thin plate splines in two dimensions, but we present it for a general conditionally definite function θ from $\mathbb{R}^d$ to $\mathbb{R}$.

3.1 Description

Suppose that

$$s(\underline{x}) = \sum_{j=1}^{n} \lambda_j \, \theta(\underline{x} - \underline{x}_j) + p(\underline{x}), \qquad \underline{x} \in \mathbb{R}^d, \tag{3.1.1}$$

is an approximation to the required interpolant s^*. If the residuals $f_i - s(\underline{x}_i)$, $i = 1, \ldots, n$, are small enough, the calculation terminates. Otherwise, an

iteration of Algorithm A revises the coefficients λ_j, $j = 1, \ldots, n$, and the polynomial part p by adding suitable corrections to s.

Each iteration is divided into three steps. Step 1 is composed of $n - q$ stages, where q is a prescribed integer less than n and greater than the dimension of $\Pi_m(\mathbb{R}^d)$ which has the value $M = \binom{d+m}{d}$. A typical value for q is 30. For $k = 1, \ldots, n - q$, the principal task of the k-th stage is to provide a new value for λ_k. Once we have adjusted λ_k, we do not want it to be altered by any subsequent calculations. Thus the corrections that are added to s in the k-th stage are functions with centres only at the interpolation points $\underline{x}_k, \ldots, \underline{x}_n$, since otherwise some of the already revised coefficients λ_j, $j = 1, \ldots, k - 1$, would be changed.

The second step of each iteration adjusts λ_j, $j = n - q + 1, \ldots, n$, and p by adding a correction which has centres only at the last q data points $\underline{x}_{n-q+1}, \ldots, \underline{x}_n$. This does not alter any of the coefficients revised in Step 1. The method is analogous to forward substitution, since, once the value of λ_k, $k = 1, \ldots, n$, has been revised, we accept it for the remainder of the iteration. The corrections of Step 1 and Step 2 are separated, because they are calculated in fundamentally different ways.

Step 3 checks whether the residuals $f_j - s(\underline{x}_j)$, $j = 1, \ldots, n$, are sufficiently close to zero, where s denotes the current approximation. If this is the case, termination occurs. Otherwise another iteration is performed.

We now consider the corrections in detail. In the k-th stage of Step 1, we want to calculate a new value for λ_k. Suppose that the first $k-1$ coefficients are correct, i.e. that $\lambda_i = \lambda_i^*$, $i = 1, \ldots, k - 1$. The function $s^* - s$ then has centres only at the last $n - k + 1$ data points $\underline{x}_k, \ldots, \underline{x}_n$. Therefore, if we add to s the interpolant σ of the residuals $f_i - s(\underline{x}_i)$, $i = k \ldots, n$, we achieve $s = s^*$. Thus the required correction to λ_k is the coefficient of $\theta(\underline{x} - \underline{x}_k)$ in

σ. For each integer $j = k, \ldots, n$, we let

$$\hat{\chi}_{kj}(\underline{x}) = \sum_{i=k}^{n} \hat{\lambda}_{kji}\, \theta(\underline{x} - \underline{x}_i) + \sum_{i=1}^{M} c_{kji}\, p_i(\underline{x}), \qquad \underline{x} \in \mathbb{R}^d, \tag{3.1.2}$$

be the function in S_θ defined by the Lagrange conditions $\hat{\chi}_{kj}(\underline{x}_i) = \delta_{ij}$, $i = k, \ldots, n$. Then σ can be written as

$$\begin{aligned} \sigma(\underline{x}) &= \sum_{j=k}^{n} \left[f_j - s(\underline{x}_j) \right] \hat{\chi}_{kj}(\underline{x}) \\ &= \sum_{j=k}^{n} \sum_{i=k}^{n} \hat{\lambda}_{kji} \left[f_j - s(\underline{x}_j) \right] \theta(\underline{x} - \underline{x}_i) + \text{polynomial}, \ \underline{x} \in \mathbb{R}^d. \end{aligned} \tag{3.1.3}$$

Thus the coefficient of $\theta(\underline{x} - \underline{x}_k)$ in σ is the sum $\sum_{j=k}^{n} \hat{\lambda}_{kjk}[f_j - s(\underline{x}_j)]$. Now, by a remark following equation (2.1.21) about the symmetry of Λ, we know that $\hat{\lambda}_{kjk}$ equals $\hat{\lambda}_{kkj}$. Hence only the coefficients $\hat{\lambda}_{kkj}$, $j = k, \ldots, n$, of $\hat{\chi}_{kk}$ are needed to calculate the coefficient of $\theta(\underline{x} - \underline{x}_k)$ in σ.

Unfortunately, it is too costly to calculate all the coefficients $\hat{\lambda}_{kkj}$, $j = k, \ldots, n$, when n is large. Therefore we choose, for each integer k in $[1, n-q]$, a set $\mathcal{L}_k \subset \{k, \ldots, n\}$ of size $|\mathcal{L}_k| = q$ which contains k. The function $\hat{\chi}_{kk}$ is then approximated by

$$\hat{\chi}_k(\underline{x}) = \sum_{i \in \mathcal{L}_k} \hat{\lambda}_{ki}\, \theta(\underline{x} - \underline{x}_i) + \sum_{i=1}^{M} \hat{c}_{ki}\, p_i(\underline{x}), \qquad \underline{x} \in \mathbb{R}^d, \tag{3.1.4}$$

which is the solution of the interpolation equations

$$\hat{\chi}_k(\underline{x}_i) = \delta_{ik}, \qquad i \in \mathcal{L}_k. \tag{3.1.5}$$

As usual, the coefficients of $\hat{\chi}_k$ are required to satisfy the constraints

$$\sum_{i \in \mathcal{L}_k} \hat{\lambda}_{ki}\, q(\underline{x}_i) = 0, \qquad \forall\, q \in \Pi_m(\mathbb{R}^d), \tag{3.1.6}$$

so that $\hat{\chi}_k$ is in S_θ for $k = 1, \ldots, n-q$. All the coefficients $\{\hat{\lambda}_{kj} : j \in \mathcal{L}_k\}$, $k = 1, \ldots, n-q$, are calculated explicitly before the iterations of the algorithm are begun. The k-th stage replaces λ_k by $\lambda_k + \mu_k$, where μ_k is the number

$$\mu_k = \sum_{j \in \mathcal{L}_k} \hat{\lambda}_{kj} \left[f_j - s\left(\underline{x}_j\right) \right]. \tag{3.1.7}$$

If the residuals $f_j - s(\underline{x}_j)$, $j \in \mathcal{L}_k$, are available, μ_k can be found in only $\mathcal{O}(q)$ operations.

To revise λ_k, any function in S_θ with centres only at the interpolation points $\underline{x}_k, \ldots, \underline{x}_n$ and coefficient μ_k in front of $\theta(\underline{x} - \underline{x}_k)$ can be added to s. Now $\hat{\chi}_k$ is available and we ensure that $\hat{\lambda}_{kk}$ is nonzero for $k = 1, \ldots, n-q$ by requiring the set $\mathcal{L}_k$ to have the property that the points $\{\underline{x}_j : j \in \mathcal{L}_k, j \neq k\}$ do not lie in the zero set of a nonzero polynomial of degree at most m. Therefore, $\hat{\chi}_k$ is not an element of $\Pi_m(\mathbb{R}^d)$, so the semi-inner product $(\hat{\chi}_k, \hat{\chi}_k)_\theta$ is strictly positive. Further, since the semi-inner product is σ_θ times the sum of the products of the coefficients of one function with the function values of the other function, equations (2.2.22), (3.1.4) and (3.1.5) provide the value

$$(\hat{\chi}_k, \hat{\chi}_k)_\theta = \sigma_\theta \, \hat{\lambda}_{kk} > 0. \tag{3.1.8}$$

In the case of thin plate splines in two dimensions, for example, the points $\{\underline{x}_j : j \in \mathcal{L}_k, j \neq k\}$ are not allowed to be collinear. This can be achieved by ordering the points $\underline{x}_i$, $i = 1, \ldots, n$, initially so that the last three are not collinear. Then for $k = 1, \ldots, n-q$, we include not only k but also the indices $n-2, n-1, n$ in the set $\mathcal{L}_k$. Algorithm A uses the function

$$\frac{\mu_k}{\hat{\lambda}_{kk}} \hat{\chi}_k(\underline{x}), \qquad \underline{x} \in \mathbb{R}^d, \tag{3.1.9}$$

which has μ_k as coefficient in front of $\theta(\underline{x} - \underline{x}_k)$, as the correction to $s(\underline{x})$, $\underline{x} \in \mathbb{R}^d$. We will see later that this choice gives excellent convergence properties.

Examples of graphs of the functions $\hat{\chi}_k$, $k = 1, 100, 600, 800$ for $n = 900$ for $\theta(\underline{x}) = \phi(\| \underline{x} \|_2) = \| \underline{x} \|_2$ in two dimensions are displayed in Figure 3.1. The sets $\mathcal{L}_k$ are chosen to contain the indices of the q data points in $\{ \underline{x}_j : j \geq k\}$ which are closest to $\underline{x}_k$, including $\underline{x}_k$ itself. For $k = 1$, $\mathcal{L}_1$ contains the indices of the q data points nearest to $\underline{x}_1$ and $\hat{\chi}_1$ is very similar to the Lagrange function χ_1 defined by $\chi_1(\underline{x}_i) = \delta_{1i}$, $i = 1, \ldots, n$. For larger values of k the choice of indices is restricted by the condition $\mathcal{L}_k \subset \{k, \ldots, n\}$, which causes the function $\hat{\chi}_k$ to become broader with increasing values of k. We will see later that, for good performance of Algorithm A, we do not require $|\hat{\chi}_k(\underline{x}_i)|$ to be small for $i \in \{1, \ldots, n\} \backslash \mathcal{L}_k$, but it is important for the semi-inner products $(\hat{\chi}_k, \hat{\chi}_j)_\theta$ to be small for $j \neq k$. If, instead of restricting the size of $\mathcal{L}_k$, we let $\mathcal{L}_k = \{k, \ldots, n\}$, then $\hat{\chi}_k = \hat{\chi}_{kk}$, $k = 1, \ldots, n - q$. In this case, $\lambda_i = \lambda_i^*$, $i = 1, \ldots, n$, holds at the end of the first iteration. Further, the semi-inner product $(\hat{\chi}_{kk}, \hat{\chi}_{jj})_\theta$ is zero for $n - q \geq j > k \geq 1$, since $\hat{\chi}_{jj}$ has centres at $\underline{x}_j, \ldots, \underline{x}_n$ and $\hat{\chi}_{kk}$ vanishes there. By a similar argument, $\hat{\chi}_{kk}$, $k = 1, \ldots, n - q$, is orthogonal to all functions with centres only at $\underline{x}_{n-q+1}, \ldots, \underline{x}_n$, with respect to the semi-inner product (2.2.22). It will be proved in Theorem 3.2.3 in Section 3.2 that convergence occurs in one iteration, if these orthogonality conditions hold.

In Step 2, the interpolant σ of the residuals $f_j - s(\underline{x}_j)$, $j = n-q+1, \ldots, n$, is calculated, where s denotes the current approximation, and σ is added as the correction to s. We know from a condition on $\mathcal{L}_{n-q}$ that the points $\{\underline{x}_i : i = n - q + 1, \ldots, n\}$ are polynomially unisolvent and thus σ is defined uniquely. At the end of each iteration, s satisfies $s(\underline{x}_i) = f_i = s^*(\underline{x}_i)$, $i = n - q + 1, \ldots, n$, and it remains in S_θ.

A more detailed description of the above outline of Algorithm A is given below. We let ℓ denote the iteration number. The ℓ-th iteration generates a

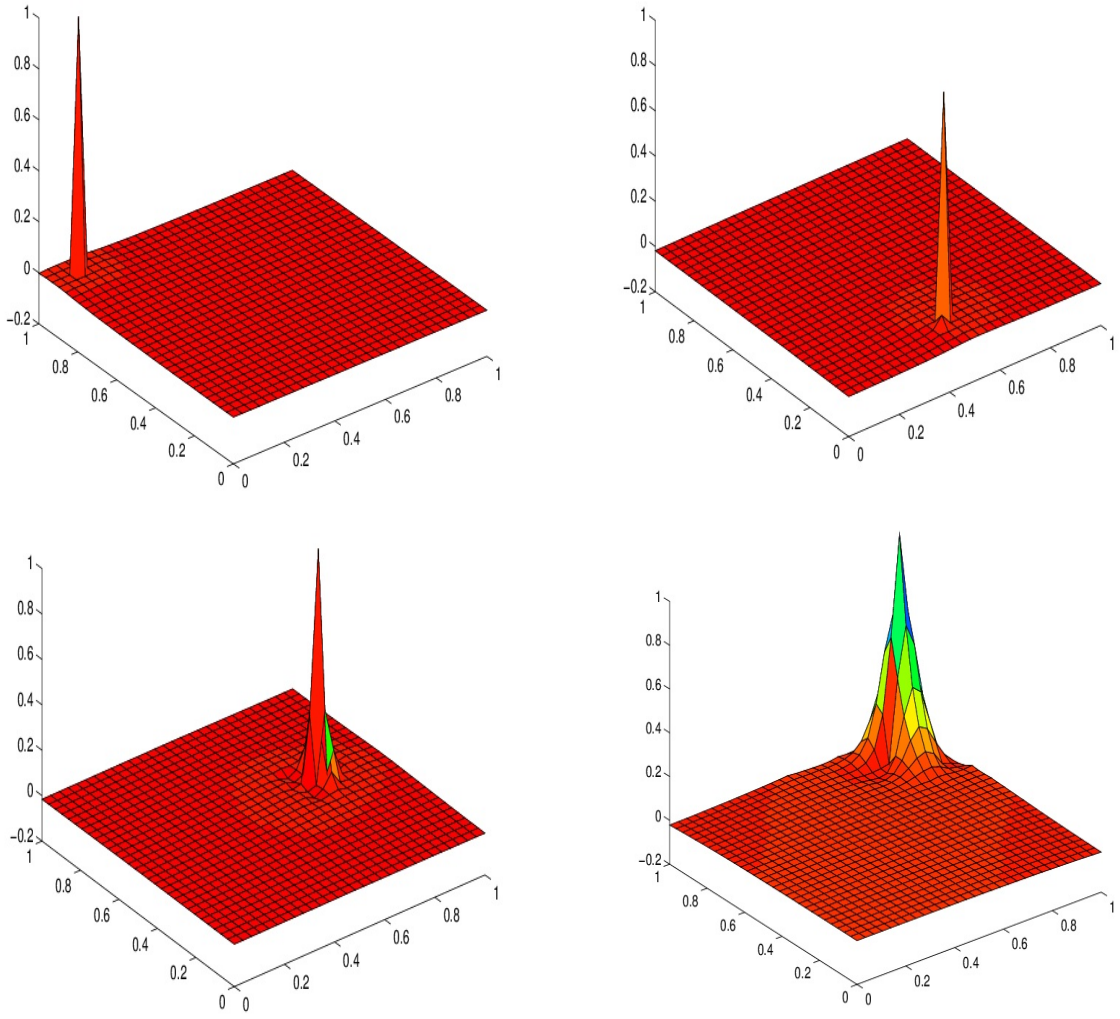


Figure 3.1: Examples of the functions $\hat{\chi}_k$.

new approximation $s^{(\ell+1)}$ from $s^{(\ell)}$, where $s^{(1)} \equiv 0$ initially. The steps are as follows.

Step 0 Set $\ell = 1$ and $s^{(\ell)} = s_0^{(\ell)} \equiv 0$.

Step 1 For $k = 1, \ldots, n-q$, replace $s_{k-1}^{(\ell)}$ by $s_k^{(\ell)}$ according to the rule

$$s_k^{(\ell)} = s_{k-1}^{(\ell)} + \frac{1}{\hat{\lambda}_{kk}} \sum_{j \in \mathcal{L}_k} \hat{\lambda}_{kj} \left[f_j - s_{k-1}^{(\ell)}(\underline{x}_j) \right] \hat{\chi}_k. \tag{3.1.10}$$

Step 2 Generate $s^{(\ell+1)}$ by adding to $s_{n-q}^{(\ell)}$ the solution $\sigma^{(\ell)} \in S_\theta$ of the interpolation equations

$$\sigma^{(\ell)}(\underline{x}_j) = f_j - s_{n-q}^{(\ell)}(\underline{x}_j), \qquad j = n-q+1, \ldots, n. \tag{3.1.11}$$

Step 3 Terminate if the residuals $f_j - s^{(\ell+1)}(\underline{x}_j)$, $j = 1, \ldots, n$, are sufficiently close to zero. Otherwise, increase ℓ by one, set $s_0^{(\ell)} = s^{(\ell)}$ and return to Step 1.

Note that the correction term of Step 1 is taken from equations (3.1.7) and (3.1.9) with $s = s_{k-1}^{(\ell)}$, $k = 1, \ldots, n-q$.

Experiments with thin plate splines in two dimensions gave very good results (Faul and Powell, 1998, Powell, 1997). Some numerical tests with multiquadric and linear functions in two dimensions and with thin plate splines and linear functions in three dimensions have been tried recently. The results are reported in Chapter 7 and are highly successful.

The description of Algorithm A is complete. The next section gives a proof of convergence.

3.2 Convergence analysis

The analysis of convergence for Algorithm A was thought to be difficult. Attempts were made to establish properties of the $n \times n$ matrix, R_A say, such that the vector of residuals $f_i - s^{(\ell+1)}(\underline{x}_i)$, $i = 1, \ldots, n$, at the end of the ℓ-th iteration, $\ell = 1, 2, \ldots$, is R_A times the vector of residuals $f_i - s^{(\ell)}(\underline{x}_i)$, $i = 1, \ldots, n$, at the beginning of the iteration. The spectral radius of R_A is considered briefly by Beatson, Goodsell and Powell (1995), because convergence occurs if it is less than one. The iteration matrix R_A will be derived in Section 3.4.

Here we present a different approach, however, which led to success following Faul and Powell (1998). The proof of convergence depends mainly on the following lemma.

Lemma 3.2.1 *Let $s^{(\ell)}$ and $s_k^{(\ell)}$, $\ell = 1, 2, \ldots$, $k = 0, \ldots, n-q$, be calculated by the algorithm of Section 3.1. Then, after each stage in Step 1 of every iteration, the semi-inner product $(s^* - s_k^{(\ell)}, s^* - s_k^{(\ell)})_\theta$ has a value less than or equal to $(s^* - s_{k-1}^{(\ell)}, s^* - s_{k-1}^{(\ell)})_\theta$ for $k = 1, \ldots, n-q$ and also the inequality $(s^* - s^{(\ell+1)}, s^* - s^{(\ell+1)})_\theta \leq (s^* - s_{n-q}^{(\ell)}, s^* - s_{n-q}^{(\ell)})_\theta$ holds.*

Proof: For $k = 1, \ldots, n-q$, the k-th stage of Step 1 calculates $s_k^{(\ell)}$ according to rule (3.1.10), which gives $(s^* - s_k^{(\ell)}, s^* - s_k^{(\ell)})_\theta = (s^* - s_{k-1}^{(\ell)} - \rho_k \hat{\chi}_k, s^* - s_{k-1}^{(\ell)} - \rho_k \hat{\chi}_k)_\theta$, where

$$\rho_k = \frac{1}{\hat{\lambda}_{kk}} \sum_{j \in \mathcal{L}_k} \hat{\lambda}_{kj} \left[f_j - s_{k-1}^{(\ell)}(\underline{x}_j) \right]. \tag{3.2.1}$$

Now it is elementary that the quadratic $(s^* - s_{k-1}^{(\ell)} - \rho\, \hat{\chi}_k, s^* - s_{k-1}^{(\ell)} - \rho\, \hat{\chi}_k)_\theta$, $\rho \in \mathbb{R}$, is least when ρ has the value

$$\rho = \frac{(s^* - s_{k-1}^{(\ell)}, \hat{\chi}_k)_\theta}{(\hat{\chi}_k, \hat{\chi}_k)_\theta}, \tag{3.2.2}$$

and fortunately this number is just ρ_k. Indeed, the identity $(\hat{\chi}_k, \hat{\chi}_k)_\theta = \sigma_\theta \hat{\lambda}_{kk}$ is noted in equation (3.1.8). Further, definition (2.2.22) and equation (1.1.1) imply the formula

$$\begin{aligned}(s^* - s^{(\ell)}_{k-1}, \hat{\chi}_k)_\theta &= \sigma_\theta \sum_{i \in \mathcal{L}_k} \hat{\lambda}_{ki} \left[s^*(\underline{x}_i) - s^{(\ell)}_{k-1}(\underline{x}_i) \right] \\ &= \sigma_\theta \sum_{j \in \mathcal{L}_k} \hat{\lambda}_{kj} \left[f_j - s^{(\ell)}_{k-1}(\underline{x}_j) \right]. \end{aligned} \tag{3.2.3}$$

Thus these stages of Step 1 provide the monotonicity of $(s^* - s^{(\ell)}_k, s^* - s^{(\ell)}_k)_\theta$, $k = 0, \ldots, n-q$.

It remains to consider Step 2 of an iteration, where the solution $\sigma^{(\ell)}$ of the equations (3.1.11) is added to $s^{(\ell)}_{n-q}$ to form $s^{(\ell+1)}$. Now $\sigma^{(\ell)}$ lies in the subspace T_θ of S_θ which consists of functions of the form

$$\tau(\underline{x}) = \sum_{j=n-q+1}^{n} \tau_j \, \theta(\underline{x} - \underline{x}_j) + \text{polynomial}, \qquad \underline{x} \in \mathbb{R}^d. \tag{3.2.4}$$

Using the definition of the semi-inner product (2.2.22) as before, and the interpolation equations (1.1.1) and (3.1.11), we find that $\sigma^{(\ell)}$ has the property

$$(s^* - s^{(\ell)}_{n-q} - \sigma^{(\ell)}, \tau)_\theta = \sigma_\theta \sum_{j=n-q+1}^{n} \tau_j \left[s^*(\underline{x}_j) - s^{(\ell)}_{n-q}(\underline{x}_j) - \sigma^{(\ell)}(\underline{x}_j) \right] = 0, \tag{3.2.5}$$

for any element τ in T_θ. Therefore $\sigma^{(\ell)}$ is the element of T_θ that minimizes $(s^* - s^{(\ell)}_{n-q} - \tau, s^* - s^{(\ell)}_{n-q} - \tau)_\theta$, $\tau \in T_\theta$, and hence $(s^* - s^{(\ell+1)}, s^* - s^{(\ell+1)})_\theta \leq (s^* - s^{(\ell)}_{n-q}, s^* - s^{(\ell)}_{n-q})_\theta$ holds. The proof is complete. □

The proof of Lemma 3.2.1 shows that the k-th stage, $k = 1, \ldots, n-q$ of Step 1 projects the difference $s^* - s^{(\ell)}_{k-1}$ onto the subspace of S_θ consisting of all functions orthogonal to $\hat{\chi}_k$. Step 2 projects the difference $s^* - s^{(\ell)}_{n-q}$ onto the subspace orthogonal to T_θ. Some other projection methods are examined by Beatson, Light and Billings (1999). We now turn to the main theorem of this section.

Theorem 3.2.2 *Let Algorithm A specified in Section 3.1 generate the sequence of functions $s^{(\ell)}$, $\ell = 1, 2, \ldots$. Then $s^{(\ell)}$ converges to s^* in the linear space S_θ as ℓ tends to infinity.*

Proof: Since S_θ is a finite dimensional space, all norms are equivalent. For our purposes we choose

$$\| s \|_{\max} = \max\{|s(\underline{x}_i)| : i = 1, \ldots, n\}, \qquad s \in S_\theta, \tag{3.2.6}$$

as a norm on S_θ. This is well defined, since interpolation by conditionally definite functions is unique. Hence, if $\| s^* - s^{(\ell)} \|_{\max}$ tends to zero as $\ell \to \infty$, then $s^{(\ell)}$ converges to s^* in S_θ. Thus it is sufficient to prove that $s^{(\ell)}(\underline{x}_i)$ tends to $s^*(\underline{x}_i)$ for $i = 1, \ldots, n$.

Since $(s^* - s_k^{(\ell)}, s^* - s_k^{(\ell)})_\theta$ is monotonically decreasing by Lemma 3.2.1, and since it is bounded below by zero, it converges to a limit. Hence the difference between $(s^* - s_k^{(\ell)}, s^* - s_k^{(\ell)})$ and $(s^* - s_{k-1}^{(\ell)}, s^* - s_{k-1}^{(\ell)})$ tends to zero. Because the method gives ρ_k the value (3.2.2), it provides the identity

$$\begin{aligned} (s^* - s_k^{(\ell)}, s^* - s_k^{(\ell)})_\theta &= (s^* - s_{k-1}^{(\ell)} - \rho_k \hat{\chi}_k, s^* - s_{k-1}^{(\ell)} - \rho_k \hat{\chi}_k)_\theta \\ &= (s^* - s_{k-1}^{(\ell)}, s^* - s_{k-1}^{(\ell)})_\theta - \frac{(s^* - s_{k-1}^{(\ell)}, \hat{\chi}_k)_\theta^2}{(\hat{\chi}_k, \hat{\chi}_k)_\theta}. \end{aligned} \tag{3.2.7}$$

It follows that $(s^* - s_{k-1}^{(\ell)}, \hat{\chi}_k)_\theta$ tends to zero as $\ell \to \infty$ for $k = 1, \ldots, n - q$.

At the beginning of the ℓ-th iteration, $s^{(\ell)} = s_0^{(\ell)}$ and thus, due to the ℓ-th iteration, $(s^* - s^{(\ell)}, \hat{\chi}_1)_\theta$ converges to zero as ℓ tends to infinity. At the beginning of the second stage of Step 1 of the ℓ-th iteration, the current approximation is the function $s^{(\ell)} + (\hat{\chi}_1, \hat{\chi}_1)_\theta^{-1}(s^* - s^{(\ell)}, \hat{\chi}_1)_\theta \, \hat{\chi}_1$, so we find the property

$$\lim_{\ell \to \infty} (s^* - s^{(\ell)} - \frac{(s^* - s^{(\ell)}, \hat{\chi}_1)_\theta \, \hat{\chi}_1}{(\hat{\chi}_1, \hat{\chi}_1)_\theta}, \hat{\chi}_2)_\theta = 0. \tag{3.2.8}$$

It follows from $(s^* - s^{(\ell)}, \hat{\chi}_1)_\theta \to 0$ that $(s^* - s^{(\ell)}, \hat{\chi}_2)_\theta$ also tends to zero. Proceeding in the same manner, one can deduce that $(s^* - s^{(\ell)}, \hat{\chi}_k)_\theta$ converges to zero as ℓ tends to infinity for every integer k in $[1, n-q]$.

Now the definition of the semi-inner product (2.2.22) gives the condition

$$(s^* - s^{(\ell)}, \hat{\chi}_k)_\theta = \sigma_\theta \sum_{j \in \mathcal{L}_k} \hat{\lambda}_{kj} \left[s^*(\underline{x}_j) - s^{(\ell)}(\underline{x}_j) \right], \tag{3.2.9}$$

so we can write the conclusion of the last paragraph in the form

$$\lim_{\ell \to \infty} \left\{ \sum_{j=k}^{n} \hat{\lambda}_{kj} \left[s^*(\underline{x}_j) - s^{(\ell)}(\underline{x}_j) \right] \right\} = 0, \quad k = 1, \ldots, n-q, \tag{3.2.10}$$

where $\hat{\lambda}_{kj}$ is defined to be zero for $j \notin \mathcal{L}_k$. We consider the equations (3.2.10) in reverse order, remembering $s^{(\ell)}(\underline{x}_j) = s^*(\underline{x}_j)$, $j = n-q+1, \ldots, n$, and $\hat{\lambda}_{kk} \neq 0$. It follows by induction on k that $s^*(\underline{x}_k) - s^{(\ell)}(\underline{x}_k)$ tends to zero as $\ell \to \infty$ for $k = n-q, n-q-1, \ldots, 1$. Thus the required convergence of $s^{(\ell)}$ to s^* in the linear space S_θ is obtained. □

The following theorem considers a special case, where convergence is achieved within one iteration.

Theorem 3.2.3 *If all $\hat{\chi}_k$, $k = 1, \ldots, n-q$, are orthogonal to each other and to all functions in T_θ with respect to the semi-inner product (2.2.22), then $s^{(2)} = s^*$ occurs at the end of the first iteration of Algorithm A.*

Proof: Since the functions $\hat{\chi}_k$, $k = 1, \ldots, n-q$, are orthogonal to each other and to all functions in T_θ and since T_θ has dimension q, the functions $\hat{\chi}_k$, $k = 1, \ldots, n-q$, together with a basis of T_θ, form a basis of S_θ. The required interpolant s^* can therefore be written uniquely in the form $s^* = \sum_{j=1}^{n-q} \alpha_j^* \hat{\chi}_j + \tau^*$, where $\tau^* \in T_\theta$. It will be shown by induction on k that the identity

$$s^* - s_k^{(1)} = \sum_{j=k+1}^{n-q} \alpha_j^* \hat{\chi}_j + \tau^* \tag{3.2.11}$$

holds for $k = 1, \ldots, n-q$. Firstly, equation (3.2.11) is true for $k = 0$, since $s_0^{(1)} \equiv 0$. Suppose equation (3.2.11) holds for some integer $k \in [1, n-q-1]$. The $(k+1)$-th stage adds a multiple of $\hat{\chi}_{k+1}$ to $s^* - s_k^{(1)}$ and projects $s^* - s_k^{(1)}$ onto the subspace orthogonal to $\hat{\chi}_{k+1}$. It follows that the coefficients α_j^*, $j = k+2, \ldots, n-q$, are left unchanged and that the new coefficient multiplying $\hat{\chi}_{k+1}$ in $s^* - s_{k+1}^{(1)}$ is zero. Thus (3.2.11) holds, if k is increased by one.

Equation (3.2.11) for $k = n-q$ gives the identity $s^* - s_{n-q}^{(1)} = \tau^*$. Now Step 2 of the first iteration of Algorithm A adds the solution $\sigma^{(1)} \in T_\theta$ of the interpolation equations $\sigma^{(1)}(\underline{x}_i) = s^*(\underline{x}_i) - s_{n-q}^{(1)}(\underline{x}_i)$, $i = n-q+1, \ldots, n$, as correction to $s_{n-q}^{(1)}$. It follows by the uniqueness of interpolation with conditionally definite functions that $\sigma^{(1)} = \tau^*$. This yields $s^{(2)} = s_{n-q}^{(1)} + \sigma^{(1)} = s^*$ and the theorem is proved. $\square$

The important role of the semi-inner product in the convergence analysis inspired the line search technique described in Section 5.1. The next section, however, presents Algorithm A in a different light.

3.3 A different view of Algorithm A

Having established the convergence of Algorithm A via minimization of the semi-inner product, we will see in this section that Algorithm A is equivalent to an iterative technique to solve a certain system of linear equations with a symmetric and positive definite matrix. We take a closer look at this system of equations. We show how it can be derived from the original system (2.1.19) and how this approach is related to other known methods. It is then easy to derive the iteration matrix R_A that is mentioned at the beginning of Section 3.2. The analysis of this section leads to an alternative proof of convergence and to the development of the technique that will be described in Section

5.2.

First we choose further functions $\hat{\chi}_{n-q+1}, \ldots, \hat{\chi}_{n-M}$ as follows such that $\hat{\chi}_1, \ldots, \hat{\chi}_{n-M}$ are linearly independent in S_θ. By a condition on $\mathcal{L}_{n-q}$, we know that the last q points $\underline{x}_{n-q+1}, \ldots, \underline{x}_n$ are not in the zero set of a nonzero polynomial of degree at most m. Reordering the points if necessary, we may assume that the last M data points $\underline{x}_{n-M+1}, \ldots, \underline{x}_n$ are polynomially unisolvent. For $k = n-q+1, \ldots, n-M$ we define $\mathcal{L}_k$ to be the set $\{k, \ldots, n\}$ and

$$\hat{\chi}_k(\underline{x}) = \sum_{j \in \mathcal{L}_k} \hat{\lambda}_{kj}\, \theta(\underline{x} - \underline{x}_j) + \sum_{j=1}^{M} \hat{c}_{kj}\, p_j(\underline{x}), \qquad \underline{x} \in \mathbb{R}^d, \tag{3.3.1}$$

to be the unique function in T_θ which satisfies the interpolation equations $\hat{\chi}_k(\underline{x}_i) = \delta_{ik}$ for $i \in \mathcal{L}_k$. This definition is analogous to (3.1.4) and (3.1.5). Thus the function $\hat{\chi}_k$, $k = n - q + 1, \ldots, n - M$, vanishes at the last $n - k$ data points. Further, the semi-inner product $(\hat{\chi}_k, \hat{\chi}_i)_\theta$ is zero for $n - M \geq k > i > n - q$, because the least index in the first sum in (3.3.1) is $j = k$ and all of the values $\hat{\chi}_i(\underline{x}_j)$, $j = k, \ldots, n$, are zero, and because the semi-inner product is the sum of coefficients of one function times function values of the other. The polynomial unisolvency of $\underline{x}_{n-M+1}, \ldots, \underline{x}_n$ ensures that $\hat{\chi}_k$, $k = n-q+1, \ldots, n-M$, is not a polynomial, and hence $(\hat{\chi}_k, \hat{\chi}_k)_\theta = \sigma_\theta \hat{\lambda}_{kk} > 0$.

For $k = 1, \ldots, n - M$, we define $\tilde{\chi}_k$ to be the function

$$\tilde{\chi}_k(\underline{x}) = \frac{1}{\sqrt{\sigma_\theta \hat{\lambda}_{kk}}} \hat{\chi}_k(\underline{x}), \qquad \underline{x} \in \mathbb{R}^d, \tag{3.3.2}$$

which is well-defined since $\sigma_\theta \hat{\lambda}_{kk} > 0$. This normalisation causes $(\tilde{\chi}_k, \tilde{\chi}_k)_\theta$ to take the value one. We require the following lemma.

Lemma 3.3.1 *Every function $s(\underline{x}) = \sum_{j=1}^{n} \lambda_j\, \theta(\underline{x} - \underline{x}_j) + p(\underline{x}) \in S_\theta$ can be expressed uniquely in the form*

$$s(\underline{x}) = \sum_{k=1}^{n-M} \alpha_k\, \tilde{\chi}_k(\underline{x}) + r(\underline{x}), \qquad \underline{x} \in \mathbb{R}^d, \tag{3.3.3}$$

where r is a polynomial of degree at most m. Formulae for α_k and the polynomial $r \in \Pi_m(\mathbb{R}^d)$ are given in the proof. (Thus the functions $\tilde{\chi}_k$, $k = 1, \ldots, n-M$, and a basis of $\Pi_m(\mathbb{R}^d)$ form a basis of S_θ.) If $\hat{\chi}_k$, $k = 1, \ldots, n-q$, are orthogonal to each other and to every function in T_θ with respect to the semi-inner product (2.2.22), then $\alpha_k = (s, \tilde{\chi}_k)_\theta$, $k = 1, \ldots, n-M$.

Proof: For $j = 1, \ldots, n$, let $\mathcal{M}_j$ be the set of all indices k such that j is an element of $\mathcal{L}_k$. Since $\mathcal{L}_k \subset \{k, \ldots, n\}$ and since it contains k, it follows that $\mathcal{M}_j \subset \{1, \ldots, j\}$, $j = 1, \ldots, n$, and that $j \in \mathcal{M}_j$, $j = 1, \ldots, n-M$. Now the definitions (3.1.4), (3.3.1) and (3.3.2) imply

$$\sum_{k=1}^{n-M} \alpha_k \, \tilde{\chi}_k(\underline{x}) = \sum_{k=1}^{n-M} \alpha_k \, \frac{1}{\sqrt{\sigma_\theta \hat{\lambda}_{kk}}} \sum_{j \in \mathcal{L}_k} \hat{\lambda}_{kj} \, \theta(\underline{x} - \underline{x}_j) + \text{polynomial}, \qquad \underline{x} \in \mathbb{R}^d. \tag{3.3.4}$$

Rearranging the summation, we see that the coefficient of $\theta(\underline{x} - \underline{x}_j)$, $j = 1, \ldots, n-M$, is

$$\sum_{k \in \mathcal{M}_j} \alpha_k \frac{1}{\sqrt{\sigma_\theta \hat{\lambda}_{kk}}} \hat{\lambda}_{kj}, \tag{3.3.5}$$

which has to equal λ_j. Thus, using $j \in \mathcal{M}_j$, $j = 1, \ldots, n-M$, and $\hat{\lambda}_{jj} \neq 0$, we deduce

$$\alpha_j = \frac{\sigma_\theta}{\sqrt{\sigma_\theta \hat{\lambda}_{jj}}} \left[\lambda_j - \sum_{\substack{k \in \mathcal{M}_j \\ k \neq j}} \alpha_k \frac{1}{\sqrt{\sigma_\theta \hat{\lambda}_{kk}}} \hat{\lambda}_{kj} \right], \tag{3.3.6}$$

which defines the coefficients α_j uniquely for $j = 1, \ldots, n-M$ in a recursive way, since $\mathcal{M}_j \subset \{1, \ldots, j\}$.

Now the coefficients of the basis functions $\theta(\underline{x} - \underline{x}_j)$, $j = 1, \ldots, n-M$, of the function $t = s - \sum_{k=1}^{n-M} \alpha_k \, \tilde{\chi}_k$ vanish. Thus t has centres only at the last M data points $\underline{x}_{n-M+1}, \ldots, \underline{x}_n$. Let r be the unique polynomial in $\Pi_m(\mathbb{R}^d)$ that interpolates t at these points. By the uniqueness of interpolation

with conditionally definite functions over $\mathbb{R}^d$, $t = r$ has to hold. Therefore $s = \sum_{k=1}^{n-M} \alpha_k \tilde{\chi}_k + r$ is valid.

If $\hat{\chi}_k$, $k = 1, \ldots, n-q$, are orthogonal to each other and to every function in T_θ, then all the functions $\hat{\chi}_k$, $k = 1, \ldots, n-M$ are mutually orthogonal. It follows that $(\tilde{\chi}_j, \tilde{\chi}_k)_\theta = \delta_{jk}$ and

$$(s, \tilde{\chi}_k)_\theta = \sum_{j=1}^{n-M} \alpha_j (\tilde{\chi}_j, \tilde{\chi}_k)_\theta = \alpha_k, \qquad k = 1, \ldots, n-M. \tag{3.3.7}$$

Therefore the last assertion is true, which completes the proof. □

Restricting our attention to the subspace T_θ of S_θ, we see that $\tilde{\chi}_k$, $k = n-q+1, \ldots, n-M$, together with a basis of $\Pi_m(\mathbb{R}^d)$, form a basis of T_θ. Any element $t \in T_\theta$ can be written as

$$t = \sum_{k=n-q+1}^{n-M} (t, \tilde{\chi}_k)_\theta \, \tilde{\chi}_k + r, \tag{3.3.8}$$

where r is the unique polynomial in $\Pi_m(\mathbb{R}^d)$ interpolating t at the last M data points, since $\tilde{\chi}_{n-q+1}, \ldots, \tilde{\chi}_{n-M}$ are orthonormal and vanish at the last M data points.

The following theorem establishes that Algorithm A is equivalent to an iterative technique for solving the system of linear equations that is given in the theorem.

Theorem 3.3.2 (Gauss–Seidel iteration) *Let $s^* = \sum_{k=1}^{n-M} \alpha_k^* \, \tilde{\chi}_k + r^*$ be the desired interpolant, and let $s^{(\ell)} = \sum_{k=1}^{n-M} \alpha_k^{(\ell)} \, \tilde{\chi}_k + r^{(\ell)}$ be the estimate generated by Algorithm A at the beginning of the ℓ-th iteration, starting with $s^{(1)} \equiv 0$. Then every iteration adjusts the vector of coefficients $\underline{\alpha}^{(\ell)} = (\alpha_1^{(\ell)}, \ldots, \alpha_{n-M}^{(\ell)})^T$ in a way that is analogous to performing one Gauss–Seidel iteration to solve*

the linear system

$$\begin{pmatrix} (\tilde{\chi}_1, \tilde{\chi}_1)_\theta & \cdots & (\tilde{\chi}_1, \tilde{\chi}_{n-M})_\theta \\ \vdots & \ddots & \vdots \\ (\tilde{\chi}_1, \tilde{\chi}_{n-M})_\theta & \cdots & (\tilde{\chi}_{n-M}, \tilde{\chi}_{n-M})_\theta \end{pmatrix} \underline{\alpha}^* = \begin{pmatrix} (s^*, \tilde{\chi}_1)_\theta \\ \vdots \\ (s^*, \tilde{\chi}_{n-M})_\theta \end{pmatrix}, \qquad (3.3.9)$$

starting with $\underline{\alpha}^{(1)} = 0$*, where* $\underline{\alpha}^* = (\alpha_1^*, \ldots, \alpha_{n-M}^*)^T$*. This means that* $\alpha_k^{(\ell+1)}$ *is given by*

$$\alpha_k^{(\ell+1)} = \left[(s^*, \tilde{\chi}_k)_\theta - \sum_{j=1}^{k-1} \alpha_j^{(\ell+1)} (\tilde{\chi}_j, \tilde{\chi}_k)_\theta - \sum_{j=k+1}^{n-M} \alpha_j^{(\ell)} (\tilde{\chi}_j, \tilde{\chi}_k)_\theta \right], \qquad (3.3.10)$$

using $(\tilde{\chi}_k, \tilde{\chi}_k)_\theta = 1$*,* $k = 1, \ldots, n - M$*.*

Proof: The k-th stage, $k = 1, \ldots, n - q$, of Step 1 of Algorithm A changes the current approximation by a multiple of $\tilde{\chi}_k$, which gives

$$s_k^{(\ell)} = \sum_{j=1}^{k} \alpha_j^{(\ell+1)} \tilde{\chi}_j + \sum_{j=k+1}^{n-M} \alpha_j^{(\ell)} \tilde{\chi}_j + r^{(\ell)}. \qquad (3.3.11)$$

Specifically, the k-th stage, $k = 1, \ldots, n - q$, is equivalent to updating the coefficient $\alpha_k^{(\ell)}$, which multiplies $\tilde{\chi}_k$, to give $\alpha_k^{(\ell+1)}$ by adding to $s_{k-1}^{(\ell)}$ the term

$$(\sigma_\theta \hat{\lambda}_{kk})^{-1} (s^* - s_{k-1}^{(\ell)}, \hat{\chi}_k)_\theta \, \hat{\chi}_k = (s^* - s_{k-1}^{(\ell)}, \tilde{\chi}_k)_\theta \, \tilde{\chi}_k, \qquad (3.3.12)$$

where the left hand side is derived from equations (3.1.10) and (3.2.3). The coefficients $\alpha_j^{(\ell)}$, $j > k$, and $\alpha_j^{(\ell+1)}$, $j < k$, and the polynomial part $r^{(\ell)}$ are left unchanged. Further, using (3.3.11) with k replaced by $k - 1$ and $(\tilde{\chi}_k, \tilde{\chi}_k)_\theta = 1$, we deduce, for $k = 1, \ldots, n - q$,

$$\begin{aligned} \alpha_k^{(\ell+1)} &= \alpha_k^{(\ell)} + (s^* - s_{k-1}^{(\ell)}, \tilde{\chi}_k)_\theta \\ &= (s^*, \tilde{\chi}_k)_\theta - \sum_{j=1}^{k-1} \alpha_j^{(\ell+1)} (\tilde{\chi}_j, \tilde{\chi}_k)_\theta - \sum_{j=k+1}^{n-M} \alpha_j^{(\ell)} (\tilde{\chi}_j, \tilde{\chi}_k)_\theta, \end{aligned} \qquad (3.3.13)$$

which establishes equation (3.3.10) for $k \le n - q$.

Step 2 of Algorithm A adds to $s_{n-q}^{(\ell)}$ the element $\sigma^{(\ell)} \in T_\theta$ that is defined by the interpolation equations (3.1.11). Now, by equation (3.3.8) and since $\sigma^{(\ell)} \in T_\theta$, we have the formula

$$\sigma^{(\ell)} = \sum_{k=n-q+1}^{n-M} (\sigma^{(\ell)}, \tilde{\chi}_k)_\theta \, \tilde{\chi}_k + q^{(\ell)} = \sum_{k=n-q+1}^{n-M} (s^* - s_{n-q}^{(\ell)}, \tilde{\chi}_k)_\theta \, \tilde{\chi}_k + q^{(\ell)}, \quad (3.3.14)$$

where $q^{(\ell)}$ is the unique polynomial in $\Pi_m(\mathbb{R}^d)$ that interpolates $\sigma^{(\ell)}(\underline{x}_i) = s^*(\underline{x}_i) - s_{n-q}^{(\ell)}(\underline{x}_i)$, $i = n - M + 1, \ldots, n$. The second identity depends on the fact that $\sigma^{(\ell)}(\underline{x}_i) = s^*(\underline{x}_i) - s_{n-q}^{(\ell)}(\underline{x}_i)$, $i = n - q + 1, \ldots, n$. Therefore $\alpha_k^{(\ell+1)}$, $k = n - q + 1, \ldots, n - M$, can be calculated by adding $(s^* - s_{n-q}^{(\ell)}, \tilde{\chi}_k)_\theta$ to $\alpha_k^{(\ell)}$. Thus, using $s_{n-q}^{(\ell)} = \sum_{j=1}^{n-q} \alpha_j^{(\ell+1)} \, \tilde{\chi}_j + \sum_{j=n-q+1}^{n-M} \alpha_j^{(\ell)} \, \tilde{\chi}_j + r^{(\ell)}$, we find, for $k = n - q + 1, \ldots, n - M$,

$$\begin{aligned} \alpha_k^{(\ell+1)} &= \alpha_k^{(\ell)} + (s^* - s_{n-q}^{(\ell)}, \tilde{\chi}_k)_\theta \\ &= (s^*, \tilde{\chi}_k)_\theta - \sum_{j=1}^{n-q} \alpha_j^{(\ell+1)} (\tilde{\chi}_j, \tilde{\chi}_k)_\theta - \sum_{\substack{j=n-q+1 \\ j \ne k}}^{n-M} \alpha_j^{(\ell)} (\tilde{\chi}_j, \tilde{\chi}_k)_\theta \\ &= (s^*, \tilde{\chi}_k)_\theta - \sum_{j=1}^{k-1} \alpha_j^{(\ell+1)} (\tilde{\chi}_j, \tilde{\chi}_k)_\theta - \sum_{j=k+1}^{n-M} \alpha_j^{(\ell)} (\tilde{\chi}_j, \tilde{\chi}_k)_\theta, \quad (3.3.15) \end{aligned}$$

where the last identity depends on $(\tilde{\chi}_j, \tilde{\chi}_k)_\theta = 0$ for integers $j \ne k$ in $[n - q + 1, n - M]$.

The last lines of equations (3.3.13) and (3.3.15) describe a Gauss–Seidel iteration for solving the system (3.3.9), since $(\tilde{\chi}_k, \tilde{\chi}_k)_\theta = 1$, $k = 1, \ldots, n - M$. The assertion of the theorem is proved. □

Next we derive the system of equations (3.3.9) from the original system (2.1.19). Let V be the $n \times (n - M)$ matrix whose (i, j)-th entry is $(\sigma_\theta \hat{\lambda}_{jj})^{-1/2} \hat{\lambda}_{ji}$ if $i \in \mathcal{L}_j$ and zero otherwise. Thus the j-th column of V contains the vector of coefficients of $\tilde{\chi}_j$. The matrix V is sparse, since at most

q elements in each column are non-zero. It is also lower triangular, since $\mathcal{L}_j \subset \{j, \ldots, n\}$. Lemma 3.3.1 provides the formula

$$\underline{\lambda} = V\underline{\alpha}, \tag{3.3.16}$$

where $\underline{\alpha} = (\alpha_1, \ldots, \alpha_{n-M})^T$ and $\underline{\lambda} = (\lambda_1, \ldots, \lambda_n)^T$. Remembering that P is the $n \times M$ matrix, whose i-th row is $(p_1(\underline{x}_i) \cdots p_M(\underline{x}_i))$ for a basis $p_1, \ldots, p_M$ of $\Pi_m(\mathbb{R}^d)$, the matrix V also has the property $V^T P = 0$. Indeed, for any polynomial $p \in \Pi_m(\mathbb{R}^d)$, the sum

$$\sum_{i=1}^{n} V_{ij}\, p(\underline{x}_i) = (\sigma_\theta \hat{\lambda}_{jj})^{-1/2} \sum_{i \in \mathcal{L}_j} \hat{\lambda}_{ji}\, p(\underline{x}_i) = 0 \tag{3.3.17}$$

due to the conditions (3.1.6) on the coefficients of the approximate Lagrange functions. Thus, multiplying the interpolation equation

$$\Theta\underline{\lambda}^* + P\underline{c}^* = \underline{f}, \tag{3.3.18}$$

which is the first part of formula (2.1.19), by $\sigma_\theta\, V^T$ from the left and inserting $\underline{\lambda}^* = V\underline{\alpha}^*$, we obtain

$$\sigma_\theta\, V^T \Theta V \underline{\alpha}^* = \sigma_\theta\, V^T \underline{f}. \tag{3.3.19}$$

This is exactly the system of equations (3.3.9) shown in Theorem 3.3.2, because

$$\begin{aligned} (\tilde{\chi}_k, \tilde{\chi}_l) &= \sigma_\theta \sum_{i \in \mathcal{L}_k} \sum_{j \in \mathcal{L}_l} \frac{\hat{\lambda}_{ki}}{\sqrt{\sigma_\theta \hat{\lambda}_{kk}}} \frac{\hat{\lambda}_{lj}}{\sqrt{\sigma_\theta \hat{\lambda}_{ll}}}\, \theta(\underline{x}_i - \underline{x}_j) \\ &= \sigma_\theta \sum_{i=1}^{n} \sum_{j=1}^{n} V_{ik}\, \theta(\underline{x}_i - \underline{x}_j)\, V_{jl}, \qquad 1 \le k, l \le n - M, \end{aligned} \tag{3.3.20}$$

and because $(s^*, \tilde{\chi}_k)_\theta = \sigma_\theta \sum_{j \in \mathcal{L}_k} (\sigma_\theta\, \hat{\lambda}_{kk})^{-1/2} \hat{\lambda}_{kj}\, s^*(\underline{x}_j) = \sigma_\theta \sum_{j=1}^{n} V_{jk}\, f_j$.

The matrix Θ in (3.3.18) is preconditioned from the left with $\sigma_\theta V^T$ and from the right with V such that the original system of equations (3.3.18) is

replaced by the modified system (3.3.19) in order to obtain better convergence properties when (3.3.19) is solved by an iterative method. In general, one tries to choose preconditioners such that the modified matrix is close to the identity matrix. Indeed, if the functions $\hat{\chi}_k$, $k = 1, \ldots, n-q$, were orthogonal to each other and to all functions in T_θ with respect to the semi-inner product (2.2.22), then $\sigma_\theta V^T \Theta V$ would be the $(n - M) \times (n - M)$ identity matrix. Thus convergence would occur within one iteration, as proven in Theorem 3.2.3 in the previous section.

Since at most q elements in each column of V are non-zero and since V has only $n-M$ columns, the product $V^T \Theta V$ can be formed in of order $(2n-M)(n-M)q$ operations. The following lemma states some other important properties of $\sigma_\theta V^T \Theta V$.

Lemma 3.3.3 *The matrix $\sigma_\theta V^T \Theta V$ is symmetric and positive definite.*

Proof: Since Θ is symmetric, the matrix $V^T \Theta V$ is obviously symmetric. If $\underline{\alpha}$ is any non-zero vector in $\mathbb{R}^{n-M}$, we let $\underline{\lambda} \in \mathbb{R}^n$ be the vector given by (3.3.16), and with this choice $\sigma_\theta \, \underline{\alpha}^T V^T \Theta V \underline{\alpha} = \sigma_\theta \, \underline{\lambda}^T \Theta \underline{\lambda}$. Now $\underline{\lambda}$ is non-zero, since the columns of V are linearly independent, because V is lower triangular and its diagonal elements $V_{ii} = \sigma_\theta \sqrt{\sigma_\theta \hat{\lambda}_{ii}}$ are nonzero. The property $V^T P = 0$ provides $P^T \underline{\lambda} = 0$. Thus $\sigma_\theta \, \underline{\lambda}^T \Theta \underline{\lambda}$ is positive due to the conditional definiteness of Θ. It follows that $\sigma_\theta \, \underline{\alpha}^T V^T \Theta V \underline{\alpha}$ is also positive, which completes the proof of the positive definiteness of $\sigma_\theta V^T \Theta V$. $\square$

This lemma provides an alternative proof of convergence of Algorithm A. Indeed, the approximations generated by Gauss–Seidel iterations applied to a symmetric and positive definite system of equations converge to the unique solution (Iserles, 1996), which proves that the coefficients $\alpha_k^{(\ell)}$ converge to α_k^* as $\ell \to \infty$. Further, by the construction of $s^{(\ell)}$, we know that $s^* - s^{(\ell)}$

vanishes at the last q data points for $\ell \geq 2$. Hence for $i = n-q+1, \ldots, n$ and $\ell \geq 2$,

$$s^*(\underline{x}_i) - s^{(\ell)}(\underline{x}_i) = \sum_{k=1}^{n-M} (\alpha_k^* - \alpha_k^{(\ell)}) \tilde{\chi}_k(\underline{x}_i) + r^*(\underline{x}_i) - r^{(\ell)}(\underline{x}_i) = 0. \quad (3.3.21)$$

We can choose the basis $p_1, \ldots, p_M$ of $\Pi_m(\mathbb{R}^d)$ such that $p_i(\underline{x}_{n-M+j}) = \delta_{ij}$, $i, j \in [1, M]$, and then (3.3.21) implies

$$r^* - r^{(\ell)} = \sum_{i=1}^{M} \sum_{k=1}^{n-M} (\alpha_k^{(\ell)} - \alpha_k^*) \, \tilde{\chi}_k(\underline{x}_{n-M+i}) \, p_i. \quad (3.3.22)$$

It follows from the convergence of $\alpha_k^{(\ell)}$ to α_k^*, $k = 1, \ldots, n-M$, that $r^{(\ell)}$ converges to r^* as ℓ tends to infinity, which completes the alternative proof.

Some other algorithms employ different $n \times (n-M)$ matrices V such that $\sigma_\theta \, V^T \Theta V$ is positive definite. Powell (1996), for example, considers thin plate spline interpolation in two dimensions, where $M = 3$ and $\sigma_\theta = 1$. He generates V by deleting the first three columns of an $n \times n$ orthogonal matrix, Ω say, where Ω has the property that the $n \times 3$ matrix $\Omega^T P$ is upper triangular. Thus the columns of V are linearly independent and span the null space of P^T, which implies $\underline{\lambda}^* = V \underline{\alpha}^*$ for some vector $\underline{\alpha}^* \in \mathbb{R}^{n-M}$. The condition that $\Omega^T P$ is upper triangular allows Ω to be a product of at most 3 Householder rotations or $3n-6$ Givens rotations, which has the advantage that $\Omega^T \Phi \, \Omega$ can be formed in $\mathcal{O}(n^2)$ operations, and $V^T \Phi V$ is just the bottom right $(n-3) \times (n-3)$ submatrix of $\Omega^T \Phi \, \Omega$. Powell also recommends the solution of the resulting positive definite system by calculating the Cholesky factorisation of $V^T \Phi V$ when n is small as proposed by Sibson and Stone (1991). When n is large, however, it is more efficient to use conjugate gradients, which receives attention in Section 5.2. Chapter 4 considers a method that solves (3.3.9) by Jacobi iterations, in order to avoid the cost of updating the current approximation on every stage.

3.4 The iteration matrix of Algorithm A

In this section we determine the $n \times n$ iteration matrix R_A by which the vector of residuals $f_i - s^{(\ell-1)}(\underline{x}_i)$, $i = 1, \ldots, n$, at the beginning of the $(\ell-1)$-th iteration, $\ell = 2, 3, \ldots$, is multiplied to give the vector of residuals $f_i - s^{(\ell)}(\underline{x}_i)$, $i = 1, \ldots, n$, at the end of the iteration. The previous analysis makes this task straightforward.

Inserting equation (3.3.22) into the identity

$$s^* - s^{(\ell)} = \sum_{k=1}^{n-M} (\alpha_k^* - \alpha_k^{(\ell)})\tilde{\chi}_k + r^* - r^{(\ell)} \tag{3.4.1}$$

gives

$$s^* - s^{(\ell)} = \sum_{k=1}^{n-M} (\alpha_k^* - \alpha_k^{(\ell)})\, \tilde{\chi}_k - \sum_{i=1}^{M} \sum_{k=1}^{n-M} (\alpha_k^* - \alpha_k^{(\ell)})\, \tilde{\chi}_k(\underline{x}_{n-M+i})\, p_i, \tag{3.4.2}$$

where we retain the basis $p_1, \ldots, p_M$ of $\Pi_m(\mathbb{R}^d)$ such that $p_i(\underline{x}_{n-M+j}) = \delta_{ij}$. We let $\mathcal{X}$ be the $n \times (n-M)$ matrix with entries $\mathcal{X}_{ij} = \tilde{\chi}_j(\underline{x}_i)$, we let $\hat{\mathcal{X}}$ be the $M \times (n-M)$ matrix whose rows are the last M rows of $\mathcal{X}$, and we let $\underline{s}^{(\ell)} = (s^{(\ell)}(\underline{x}_1), \ldots, s^{(\ell)}(\underline{x}_n))^T$. Then, remembering $s^*(\underline{x}_i) = f_i$, it follows from equation (3.4.2) that the vector of residuals $\underline{f} - \underline{s}^{(\ell)}$ can be expressed as

$$\underline{f} - \underline{s}^{(\ell)} = \left(\mathcal{X} - P\hat{\mathcal{X}}\right)(\underline{\alpha}^* - \underline{\alpha}^{(\ell)}), \tag{3.4.3}$$

where P is still the $n \times M$ matrix whose i-th row is $(p_1(\underline{x}_i) \cdots p_M(\underline{x}_i))$.

The vector $\underline{\alpha}^{(\ell)}$ is calculated using Gauss–Seidel iterations to solve the system (3.3.19). Hence, letting $\sigma_\theta V^T \Theta V = L + U$, where L is lower triangular and U is strictly upper triangular, we obtain the formula

$$\underline{\alpha}^* - \underline{\alpha}^{(\ell)} = -L^{-1}U(\,\underline{\alpha}^* - \underline{\alpha}^{(\ell-1)}). \tag{3.4.4}$$

Thus the vector of residuals can be written as

$$\underline{f} - \underline{s}^{(\ell)} = \left(\mathcal{X} - P\hat{\mathcal{X}}\right)\left(-L^{-1}U\right)(\underline{\alpha}^* - \underline{\alpha}^{(\ell-1)}). \tag{3.4.5}$$

Equation (3.3.6) in the proof of Lemma 3.3.1 shows that, given any function $s(\underline{x}) = \sum_{k=1}^{n-M} \alpha_k \tilde{\chi}_k(\underline{x}) + r(\underline{x}) = \sum_{j=1}^{n} \lambda_j \theta(\underline{x} - \underline{x}_j) + p(\underline{x})$ in S_θ, then the coefficients α_k, $k = 1, \ldots, n-M$, are uniquely determined by the coefficients λ_j, $j = 1, \ldots, n-M$. These on the other hand are uniquely determined by the function values $s(\underline{x}_i)$, $i = 1, \ldots, n$. Therefore there exists an $(n-M) \times n$ matrix, $\mathcal{Y}$ say, such that

$$\underline{\alpha} = \mathcal{Y}\underline{s}, \tag{3.4.6}$$

where $\underline{\alpha} = (\alpha_1, \ldots, \alpha_{n-M})^T$ and $\underline{s} = (s(\underline{x}_1), \ldots, s(\underline{x}_n))^T$.

Equations (3.4.5) and (3.4.6) allow us, for $\ell \geq 2$, to derive a formula which relates the vector of residuals $\underline{f} - \underline{s}^{(\ell)}$ at the end of one iteration to the vector of residuals $\underline{f} - \underline{s}^{(\ell-1)}$ at the beginning of the iteration. The relation is the expression

$$\underline{f} - \underline{s}^{(\ell)} = \left(\mathcal{X} - P\hat{\mathcal{X}}\right)\left(-L^{-1}U\right)\mathcal{Y}\,(\underline{f} - \underline{s}^{(\ell-1)}). \tag{3.4.7}$$

Therefore the matrix $(\mathcal{X} - P\hat{\mathcal{X}})(-L^{-1}U)\mathcal{Y}$ is the iteration matrix R_A. Further, we deduce

$$\begin{aligned} \underline{f} - \underline{s}^{(\ell)} &= \left(\mathcal{X} - P\hat{\mathcal{X}}\right)(\underline{\alpha}^* - \underline{\alpha}^{(\ell)}) \\ &= \left(\mathcal{X} - P\hat{\mathcal{X}}\right)\left(-L^{-1}U\right)^{\ell-1}(\underline{\alpha}^* - \underline{\alpha}^{(1)}) \\ &= \left(\mathcal{X} - P\hat{\mathcal{X}}\right)\left(-L^{-1}U\right)^{\ell-1}\mathcal{Y}\,(\underline{f} - \underline{s}^{(1)}). \end{aligned} \tag{3.4.8}$$

It follows that the speed of convergence depends on the spectral radius of the $(n-M) \times (n-M)$ matrix $L^{-1}U$. Examples of the size of this spectral radius for different numerical experiments are given in Chapter 7.

3.5 Algorithm A as a linear operator

Algorithm A takes an element of S_θ, $s^* - s^{(\ell)}$, and generates, using linear operations, a new element $s^* - s^{(\ell+1)}$. Thus Algorithm A can be viewed as a linear operator form S_θ to S_θ. We employ the same notation for this operator as for the iteration matrix R_A introduced in the previous section, because they are related by the one-to-one relation between $s \in S_\theta$ and the vector of function values $\underline{s} = (s(\underline{x}_1), \ldots, s(\underline{x}_n))^T$. Specifically, we let $R_A s$, $s \in S_\theta$, be the function in S_θ whose values at the interpolation points are the components of the vector $R_A \underline{s}$. Therefore we write

$$s^* - s^{(\ell+1)} = R_A(s^* - s^{(\ell)}). \tag{3.5.1}$$

The following theorem establishes several useful properties of R_A.

Theorem 3.5.1 *The operator $R_A : S_\theta \to S_\theta$ annihilates polynomials of degree at most m. Further, it is a contraction mapping with respect to the semi-norm (2.2.23), which means that the condition*

$$\| R_A s \|_\theta \leq \| s \|_\theta \tag{3.5.2}$$

holds for all $s \in S_\theta$, with equality only if $s \in \Pi_m(\mathbb{R}^d)$. For any nonzero $s \in S_\theta$, $R_A s$ does not equal s. Also the inequality

$$(s, R_A s)_\theta < (s, s)_\theta \tag{3.5.3}$$

is achieved for all $s \in S_\theta$, $s \notin \Pi_m(\mathbb{R}^d)$.

Proof: Let s be any polynomial of degree at most m. Then the coefficients α_k, $k = 1, \ldots, n - M$ of the representation $s = \sum_{k=1}^{n-M} \alpha_k \tilde{\chi}_k + r$ vanish. Equation (3.4.6) then yields $\mathcal{Y}\underline{s} = 0$. Now the matrix R_A is the product

$(\mathcal{X} - P\hat{\mathcal{X}})(-L^{-1}U)\mathcal{Y}$. Hence $R_A\underline{s} = 0$, which proves the first assertion of the theorem.

We know by Lemma 3.2.1 with $\ell = 1$ that $\| s^* - s_k^{(1)} \|_\theta \leq \| s^* - s_{k-1}^{(1)} \|_\theta$, $k = 1, \ldots, n-q$, and that $\| s^* - s^{(2)} \|_\theta \leq \| s^* - s_{n-q}^{(1)} \|_\theta$. It follows from equation (3.5.1) that

$$\| s^* - s^{(2)} \|_\theta = \| R_A(s^* - s^{(1)}) \|_\theta \leq \| s^* - s^{(1)} \|_\theta. \tag{3.5.4}$$

Therefore, because s^* can be any function in S_θ and because $s^{(1)}$ is zero, inequality (3.5.2) is satisfied. Suppose now that we have equality. Then, for $k = 1, \ldots, n-q$, the difference between $\| s^* - s_{k-1}^{(1)} \|_\theta$ and $\| s^* - s_k^{(1)} \|_\theta$ is zero. It follows from the proof of Theorem 3.2.2 that all stages leave the current approximation unchanged and hence $s^* = s^* - s_k^{(1)}$, $k = 0, 1, \ldots, n-q$. Therefore equations (3.2.7) and (3.2.9) yield

$$(s^*, \hat{\chi}_k)_\theta = \sigma_\theta \sum_{j \in \mathcal{L}_k} \hat{\lambda}_{kj}\, s^*(\underline{x}_j) = 0, \qquad k = 1, \ldots, n-q. \tag{3.5.5}$$

Also the difference between $\| s^* - s^{(2)} \|_\theta$ and $\| s^* - s_{n-q}^{(1)} \|_\theta$ is zero, which implies that $\| \sigma^{(1)} \|_\theta$ vanishes, where $\sigma^{(1)}$ interpolates the data $s^*(\underline{x}_i) - s_{n-q}^{(1)}(\underline{x}_i) = s^*(\underline{x}_i)$, $i = n-q+1, \ldots, n$. Thus $\sigma^{(1)}$ is a polynomial of degree at most m, say p, and $s^*(\underline{x}_i) = p(\underline{x}_i)$, $i = n-q+1, \ldots, n$. Now the sum $\sum_{j \in \mathcal{L}_k} \hat{\lambda}_{kj}\, p(\underline{x}_j)$ vanishes due to (3.1.6). Hence, using (3.5.5), we can write

$$\sum_{j \in \mathcal{L}_k} \hat{\lambda}_{kj} \left(s^*(\underline{x}_j) - p(\underline{x}_j) \right) = 0. \tag{3.5.6}$$

We consider equations (3.5.6) in reverse order, remembering $s^*(\underline{x}_i) = p(\underline{x}_i)$, $i = n-q+1, \ldots, n$, and $\hat{\lambda}_{kk} \neq 0$ and that $\mathcal{L}_k$ is a subset of $\{k, \ldots, n\}$ containing k. It follows by induction on k that $s^*(\underline{x}_k) = p(\underline{x}_k)$ for $k = n-q, n-q-1, \ldots, 1$, which yields $s^* = p$. Hence condition (3.5.2) is satisfied as an equality only if $s^* = s \in \Pi_m(\mathbb{R}^d)$.

If s is not a polynomial of degree at most m, then $R_A s \neq s$, since inequality (3.5.2) is strict in this case. If, on the other hand, $s \in \Pi_m(\mathbb{R}^d)$, then $R_A s = 0$. Hence $R_A s \neq s$ for all nonzero functions $s \in S_\theta$.

The Cauchy–Schwarz inequality and condition (3.5.2) provide

$$(s, R_A s)_\theta \leq \| s \|_\theta \| R_A s \|_\theta \leq \| s \|_\theta^2 = (s, s)_\theta, \tag{3.5.7}$$

and we have found already that the second inequality is strict if $s \in S_\theta$ is not a polynomial of degree at most m. Therefore the last assertion of the theorem is also true. $\Box$

These properties of the operator R_A are important to the work of Chapter 6, where Krylov subspace methods are considered. In the next chapter we turn to another algorithm that was developed from Algorithm A by not updating the vector of residuals at intermediate stages.

Chapter 4

Algorithm B

Establishing convergence for Algorithm A was a major step forward, but unfortunately every iteration requires much computation. Specifically, since, for $k = 1, \ldots, n-q$, the k-th stage of Step 1 of Algorithm A employs the residuals $f_j - s_{k-1}^{(\ell)}(\underline{x}_j)$, $j \in \mathcal{L}_k$, and since the residuals $f_j - s_{n-q}^{(\ell)}(\underline{x}_j)$, $j = n-q+1, \ldots, n$, are interpolated in Step 2, the approximation s to the required interpolant s^* is updated frequently. Then the above residuals are calculated whenever they are required. Unfortunately the work of evaluating $s \in S_\theta$ at some point $\underline{x} \in \mathbb{R}^d$ is usually $\mathcal{O}(n)$, unless the support of θ is sufficiently small. Therefore techniques, known as fast multipole methods (Beatson and Newsam, 1992, Powell, 1993, Beatson and Light, 1997, Beatson, Cherrie and Mouat, 1999) were developed to reduce the cost of radial basis function evaluation. These methods use truncated Laurent series or Taylor series expansions to generate $s(\underline{x})$, $s \in S_\theta$, $\underline{x} \in \mathbb{R}^d$, to within a given precision ϵ. The cost to calculate all the Laurent coefficients of any s is only $\mathcal{O}(n \log n)$. Then $\mathcal{O}(\log n)$ operations are needed to estimate $s(\underline{x})$ for any $\underline{x} \in \mathbb{R}^d$. Still, wherever possible the evaluation of residuals should be avoided, since this

feature dominates the processing time. Therefore the idea of Algorithm B is to try to solve the system (3.3.9) by Jacobi iteration. It is usually more efficient than Algorithm A. Further, it converges in the majority of cases, but we will present some numerical experiments where this does not happen. A useful extension to Algorithm B that forces convergence is proposed and studied in Section 5.1 and Chapter 6.

In the following sections we give a description of Algorithm B, then analyse it and derive the iteration matrix. We conclude the chapter by viewing Algorithm B as a linear operator acting on the space S_θ.

4.1 Description

The concept of Algorithm B is to use $f_j - s^{(\ell)}(\underline{x}_j)$ instead of $f_j - s^{(\ell)}_{k-1}(\underline{x}_j)$ in formula (3.1.10) for the k-th stage, $k = 1, \ldots, n-q$, of Step 1, and also instead of $f_j - s^{(\ell)}_{n-q}(\underline{x}_j)$ in formula (3.1.11) of Step 2 of Algorithm A. Therefore the residuals do not need to be updated until the end of each iteration. Specifically, Algorithm B revises the approximations $s^{(\ell)}$ in the following way.

Step 0 Set $\ell = 1$ and $s^{(\ell)} \equiv 0$.

Step 1 For $k = 1, \ldots, n-q$, calculate the term

$$\rho_k^{(\ell)} = \frac{1}{\hat{\lambda}_{kk}} \sum_{j \in \mathcal{L}_k} \hat{\lambda}_{kj} \left[f_j - s^{(\ell)}(\underline{x}_j) \right]. \tag{4.1.1}$$

Step 2 Calculate the solution $\sigma^{(\ell)} \in T_\theta$ of the interpolation equations

$$\sigma^{(\ell)}(\underline{x}_j) = f_j - s^{(\ell)}(\underline{x}_j), \qquad j = n-q+1, \ldots, n. \tag{4.1.2}$$

Step 3 Define $s^{(\ell+1)}$ by the formula

$$s^{(\ell+1)} = s^{(\ell)} + \sum_{k=1}^{n-q} \rho_k^{(\ell)} \hat{\chi}_k + \sigma^{(\ell)}. \tag{4.1.3}$$

Step 4 Terminate if the residuals $f_j - s^{(\ell+1)}(\underline{x}_j)$, $j = 1, \ldots, n$, are sufficiently close to zero. Otherwise, increase ℓ by one and return to Step 1.

Using equation (3.2.3) with $s_{k-1}^{(\ell)}$ replaced by $s^{(\ell)}$, we deduce from (4.1.1) and (4.1.3) that

$$\begin{aligned} s^{(\ell+1)} &= s^{(\ell)} + \sum_{k=1}^{n-q} \frac{1}{\hat{\lambda}_{kk}} \sum_{j \in \mathcal{L}_k} \hat{\lambda}_{kj} \left[f_j - s^{(\ell)}(\underline{x}_j) \right] \hat{\chi}_k + \sigma^{(\ell)} \\ &= s^{(\ell)} + \sum_{k=1}^{n-q} (\sigma_\theta \hat{\lambda}_{kk})^{-1} (s^* - s^{(\ell)}, \hat{\chi}_k)_\theta \, \hat{\chi}_k + \sigma^{(\ell)} \\ &= s^{(\ell)} + \sum_{k=1}^{n-q} (s^* - s^{(\ell)}, \tilde{\chi}_k)_\theta \, \tilde{\chi}_k + \sigma^{(\ell)}, \end{aligned} \tag{4.1.4}$$

remembering $\tilde{\chi}_k = (\sigma_\theta \hat{\lambda}_{kk})^{-1/2} \hat{\chi}_k$. As before, s^* denotes the desired interpolant that is defined by the interpolation equations (1.1.1). Moreover, as in (3.3.8), $\sigma^{(\ell)}$ can be expressed as

$$\sigma^{(\ell)} = \sum_{k=n-q+1}^{n-M} (\sigma^{(\ell)}, \tilde{\chi}_k)_\theta \, \tilde{\chi}_k + q^{(\ell)} = \sum_{k=n-q+1}^{n-M} (s^* - s^{(\ell)}, \tilde{\chi}_k)_\theta \, \tilde{\chi}_k + q^{(\ell)}, \tag{4.1.5}$$

where $q^{(\ell)}$ is the unique polynomial of degree at most m which satisfies $q^{(\ell)}(\underline{x}_i) = \sigma^{(\ell)}(\underline{x}_i) = s^*(\underline{x}_i) - s^{(\ell)}(\underline{x}_i)$, $i = n - M + 1, \ldots, n$. As before, we reorder the points if necessary so that the points $\underline{x}_{n-M+1}, \ldots, \underline{x}_n$ are polynomially unisolvent. The second identity of expression (4.1.5) depends on equations (4.1.2) and (1.1.1) and on the fact that $\tilde{\chi}_k$, $k = n-q+1, \ldots, n-M$, has centres only at the points $\underline{x}_k, \ldots, \underline{x}_n$. Equations (4.1.4) and (4.1.5) yield

$$s^{(\ell+1)} = s^{(\ell)} + \sum_{k=1}^{n-M} (s^* - s^{(\ell)}, \tilde{\chi}_k)_\theta \, \tilde{\chi}_k + q^{(\ell)}. \tag{4.1.6}$$

The description of Algorithm B is complete. Next we consider some of its properties.

4.2 Analysis and the iteration matrix of Algorithm B

The following theorem is the analogue of Theorem 3.3.2. for Algorithm A.

Theorem 4.2.1 (Jacobi iteration) *Let $s^* = \sum_{k=1}^{n-M} \alpha_k^* \tilde{\chi}_k + r^*$ be the desired interpolant, and let $s^{(\ell)} = \sum_{k=1}^{n-M} \alpha_k^{(\ell)} \tilde{\chi}_k + r^{(\ell)}$ be the estimate generated by Algorithm B at the beginning of the ℓ-th iteration, starting with $s^{(1)} \equiv 0$. Then every iteration adjusts the vector of coefficients $\underline{\alpha}^{(\ell)} = (\alpha_1^{(\ell)}, \ldots, \alpha_{n-M}^{(\ell)})^T$ in a way that is analogous to performing one Jacobi iteration to solve the linear system (3.3.9), starting with $\underline{\alpha}^{(1)} = 0$. This means that $\alpha_k^{(\ell+1)}$ is given by*

$$\alpha_k^{(\ell+1)} = \left[(s^*, \tilde{\chi}_k)_\theta - \sum_{j=1, j\neq k}^{n-M} \alpha_j^{(\ell)} (\tilde{\chi}_j, \tilde{\chi}_k)_\theta \right], \qquad k = 1, \ldots, n-M. \quad (4.2.7)$$

Proof: It follows directly from equation (4.1.6) that, for $k = 1, \ldots, n-M$,

$$\begin{aligned} \alpha_k^{(\ell+1)} &= \alpha_k^{(\ell)} + (s^* - s^{(\ell)}, \tilde{\chi}_k)_\theta \\ &= \alpha_k^{(\ell)} + (s^*, \tilde{\chi}_k)_\theta - \sum_{j=1}^{n-M} \alpha_j^{(\ell)} (\tilde{\chi}_j, \tilde{\chi}_k)_\theta \\ &= (s^*, \tilde{\chi}_k)_\theta - \sum_{j=1, j\neq k}^{n-M} \alpha_j^{(\ell)} (\tilde{\chi}_j, \tilde{\chi}_k)_\theta, \end{aligned} \quad (4.2.8)$$

where we use the identity $(\tilde{\chi}_k, \tilde{\chi}_k)_\theta = 1$. □

n	q	maximum absolute value of off-diagonal elements	spectral radius
400	10	0.7302	33
	20	0.6896	9.7
	30	0.6528	6.0
	50	0.6287	3.3
900	10	0.7614	70
	20	0.7134	24
	30	0.6987	14
	50	0.6576	7.6

Table 4.1: Maximum value of the off-diagonal elements and the spectral radius of $I_{n-M} - \sigma_\theta V^T \Theta V$ for $\theta(\underline{x}) = \| \underline{x} \|_2^2 \log \| \underline{x} \|_2$, $\underline{x} \in \mathbb{R}^2$.

We recall from equation (3.3.20) that the systems (3.3.9) and (3.3.19) are the same. Therefore, since the diagonal entries of $\sigma_\theta V^T \Theta V$ take the value one due to the normalisation of the approximated Lagrange functions, it follows from Theorem 4.2.1 that

$$\underline{\alpha}^* - \underline{\alpha}^{(\ell+1)} = (I_{n-M} - \sigma_\theta V^T \Theta V)(\underline{\alpha}^* - \underline{\alpha}^{(\ell)}), \tag{4.2.9}$$

where I_{n-M} is the $(n-M) \times (n-M)$ identity matrix. Hence convergence depends on how near $\sigma_\theta V^T \Theta V$ is to the identity matrix. The smaller the off-diagonal elements are, the better the orthogonality properties and the better the performance of the algorithm, but if the off-diagonal elements are too large, divergence might occur.

It was very difficult to find an example to show failure of Algorithm B. If, however, the points $\underline{x}_1, \ldots, \underline{x}_n \in \mathbb{R}^2$ are equally spaced on one eighth of two concentric circles, half of the points being on each circle and the radii of the

circles being 1 and $1 + 10^{-5}$, then Algorithm B with $\theta(\underline{x}) = \| \underline{x} \|_2^2 \log \| \underline{x} \|_2$, $\underline{x} \in \mathbb{R}^2$, fails to converge. Table 4.1 gives the maximum absolute value of the off-diagonal elements and the spectral radius of $I_{n-M} - \sigma_\theta V^T \Theta V$ for this experiment for various choices of n and q.

The following theorem considers the situation when the functions $\hat{\chi}_k$, $k = 1, \ldots, n-q$, are orthogonal to each other and to all functions in T_θ.

Theorem 4.2.2 *If all functions $\hat{\chi}_k$, $k = 1, \ldots, n-q$, are orthogonal to each other and to all functions in T_θ with respect to the semi-inner product (2.2.22), then Algorithm B achieves convergence in at most 2 iterations.*

Proof: If the functions $\hat{\chi}_k$, $k = 1, \ldots, n-q$, are orthogonal to each other and to every function in T_θ, then $\sigma_\theta V^T \Theta V = I_{n-M}$ and it follows from (4.2.9) with $\ell = 1$ that $\underline{\alpha}^{(2)} = \underline{\alpha}^*$. Thus s^* and $s^{(2)}$ differ only by a polynomial of degree at most m. Then, using (1.1.1), (4.1.1) with $\ell = 2$ gives $\rho_k^{(2)} = 0$, $k = 1, \ldots, n-q$, due to the constraints (3.1.6), while (4.1.2) with $\ell = 2$ gives $\sigma^{(2)} = s^* - s^{(2)}$. Hence $s^{(3)}$ equals s^* by (4.1.3) with $\ell = 2$ and the method converges in at most two iterations. □

Returning to the case where $\hat{\chi}_k$, $k = 1, \ldots, n-q$, are not necessarily orthogonal to each other and to functions in T_θ, we let $s^{(\ell)} = \sum_{k=1}^{n-M} \alpha_k^{(\ell)} \tilde{\chi}_k + r^{(\ell)}$ and analyse the convergence properties of the polynomial part $r^{(\ell)}$ of $s^{(\ell)}$. Recalling (4.1.6), the difference between $r^{(\ell+1)}$ and $r^{(\ell)}$ is $q^{(\ell)}$. Hence, for $i = n - M + 1, \ldots, n$, we have

$$\begin{aligned} r^{(\ell+1)}(\underline{x}_i) - r^{(\ell)}(\underline{x}_i) &= q^{(\ell)}(\underline{x}_i) = s^*(\underline{x}_i) - s^{(\ell)}(\underline{x}_i) \\ &= \sum_{k=1}^{n-M} (\alpha_k^* - \alpha_k^{(\ell)})\, \tilde{\chi}_k(\underline{x}_i) + r^*(\underline{x}_i) - r^{(\ell)}(\underline{x}_i), \end{aligned} \tag{4.2.10}$$

remembering that $q^{(\ell)}$ is the unique polynomial of degree at most m interpolating $s^* - s^{(\ell)}$ at $\underline{x}_{n-M+1}, \ldots, \underline{x}_n$. Therefore, choosing $p_1, \ldots, p_M$ as the

basis of $\Pi_m(\mathbb{R}^d)$ such that $p_i(\underline{x}_{n-M+j}) = \delta_{ij}$ for integers $i, j \in [1, M]$, we can write

$$r^* - r^{(\ell+1)} = \sum_{i=1}^{M} \sum_{k=1}^{n-M} (\alpha_k^{(\ell)} - \alpha_k^*)\, \tilde{\chi}_k(\underline{x}_{n-M+i})\, p_i. \tag{4.2.11}$$

Thus, if $\underline{\alpha}^{(\ell)}$ converges to $\underline{\alpha}^*$, then $r^{(\ell+1)}$ converges to r^*.

The previous results facilitate the derivation of the matrix, R_B say, such that, for $\ell = 1, 2, 3, \ldots$, the vector of residuals $\underline{s}^* - \underline{s}^{(\ell+1)}$ at the end of the ℓ-th iteration is R_B acting on the vector of residuals $\underline{s}^* - \underline{s}^{(\ell)}$ at the beginning of the iteration. The analysis is very similar to the derivation of the iteration matrix for Algorithm A in Section 3.4.

Using equation (4.2.11), we obtain

$$\begin{aligned} s^* - s^{(\ell+1)} &= \sum_{k=1}^{n-M} (\alpha_k^* - \alpha_k^{(\ell+1)})\, \tilde{\chi}_k + r^* - r^{(\ell+1)} \qquad (4.2.12) \\ &= \sum_{k=1}^{n-M} (\alpha_k^* - \alpha_k^{(\ell+1)})\, \tilde{\chi}_k - \sum_{i=1}^{M} \sum_{k=1}^{n-M} (\alpha_k^* - \alpha_k^{(\ell)})\, \tilde{\chi}_k(\underline{x}_{n-M+i})\, p_i. \end{aligned}$$

Employing the matrices $\mathcal{X}$ and $\hat{\mathcal{X}}$ defined in Section 3.4, and remembering that P is the $n \times M$ matrix whose i-th row is $(p_1(\underline{x}_i) \cdots p_M(\underline{x}_i))$, it follows from equation (4.2.12) that

$$\underline{f} - \underline{s}^{(\ell+1)} = \mathcal{X}(\underline{\alpha}^* - \underline{\alpha}^{(\ell+1)}) - P\hat{\mathcal{X}}(\underline{\alpha}^* - \underline{\alpha}^{(\ell)}), \tag{4.2.13}$$

where $\underline{s}^{(\ell+1)} = (s^{(\ell+1)}(\underline{x}_1), \ldots, s^{(\ell+1)}(\underline{x}_n))^T$.

Now, remembering equation (4.2.9) and the matrix $\mathcal{Y}$ introduced in (3.4.6), we deduce

$$\underline{f} - \underline{s}^{(\ell+1)} = \left(\mathcal{X}\left(I_{n-M} - \sigma_\theta V^T \Theta V\right) - P\hat{\mathcal{X}}\right) \mathcal{Y}\, (\underline{f} - \underline{s}^{(\ell)}). \tag{4.2.14}$$

Therefore the matrix $(\mathcal{X}(I_{n-M} - \sigma_\theta V^T \Theta V) - P\hat{\mathcal{X}})\mathcal{Y}$ is the iteration matrix R_B. Further, employing (4.2.9) and the identity $\underline{\alpha}^{(1)} = 0$, we derive

$$\underline{f} - \underline{s}^{(\ell+1)} = \left(\mathcal{X}\left(I_{n-M} - \sigma_\theta V^T \Theta V\right) - P\hat{\mathcal{X}}\right) (\underline{\alpha}^* - \underline{\alpha}^{(\ell)}) \tag{4.2.15}$$

$$= \left(\mathcal{X}\left(I_{n-M} - \sigma_\theta V^T \Theta V\right) - P\hat{\mathcal{X}}\right)\left(I_{n-M} - \sigma_\theta V^T \Theta V\right)^{\ell-1} \underline{\alpha}^*$$
$$= \left(\mathcal{X}\left(I_{n-M} - \sigma_\theta V^T \Theta V\right) - P\hat{\mathcal{X}}\right)\left(I_{n-M} - \sigma_\theta V^T \Theta V\right)^{\ell-1} \mathcal{Y}\,\underline{f}.$$

Hence the spectral radius of the $(n-M)\times(n-M)$ matrix $I_{n-M} - \sigma_\theta V^T \Theta V$ determines the convergence properties, some examples being given in Table 4.1. Chapter 7 presents some more numerical examples of the value of this spectral radius. The changes made to Algorithm A, which gave rise to Algorithm B, led to divergence in certain cases as shown in Table 4.1. If, however, the functions $\hat{\chi}_k$, $k = 1, \ldots, n-q$, are sufficiently close to being orthogonal, then Algorithm B converges and is usually a highly efficient alternative to Algorithm A.

4.3 Algorithm B as a linear operator

Algorithm B can also be viewed as a linear operator acting on S_θ, since it takes the element $s^* - s^{(\ell)}$ of S_θ and generates a new element $s^* - s^{(\ell+1)}$, using linear operations. Again our notation for this operator is the same as for the iteration matrix R_B derived in the previous section. Indeed, $R_B s$, $s \in S_\theta$, is the function in S_θ whose values at the data points are the components of the vector $R_B \underline{s}$, so we write

$$s^* - s^{(\ell+1)} = R_B(s^* - s^{(\ell)}). \tag{4.3.1}$$

Therefore, equation (4.1.6) provides the explicit expression

$$R_B s = s - \sum_{k=1}^{n-M} (s, \tilde{\chi}_k)_\theta \, \tilde{\chi}_k - p, \tag{4.3.2}$$

where, in order to satisfy equations (4.1.2) and (4.1.5), p is the unique polynomial of degree at most m that interpolates $s(\underline{x}_i)$, $i = n - M + 1, \ldots, n$. The following theorem establishes some useful properties of the operator R_B.

Theorem 4.3.1 *The operator R_B from S_θ to S_θ defined by (4.3.2) annihilates polynomials of degree at most m. Further, it is self-adjoint with respect to the semi-inner product (2.2.22). The strict inequality*

$$(s, R_B s)_\theta < (s, s)_\theta \tag{4.3.3}$$

is achieved for all $s \in S_\theta$, $s \notin \Pi_m(\mathbb{R}^d)$. Also $R_B s \neq s$ holds for all nonzero functions $s \in S_\theta$.

Proof: Let $s \in \Pi_m(\mathbb{R}^d)$. Then the sum in (4.3.2) is zero, since the semi-inner product vanishes if one of the arguments lies in $\Pi_m(\mathbb{R}^d)$. On the other hand, $p = s$, since $p(\underline{x}_i) = s(\underline{x}_i)$, $i = n - M + 1, \ldots, n$, and since the last M data points are polynomially unisolvent. Hence it follows that $R_B s = 0$.

Let $s, t \in S_\theta$. We write $R_B t = t - \sum_{k=1}^{n-M} (t, \tilde{\chi}_k)_\theta \tilde{\chi}_k - r$, where $r \in \Pi_m(\mathbb{R}^d)$. Using expression (4.3.2), we then deduce

$$\begin{aligned} (R_B s, t)_\theta &= (s, t)_\theta - \sum_{k=1}^{n-M} (s, \tilde{\chi}_k)_\theta (t, \tilde{\chi}_k)_\theta - (p, t)_\theta \\ &= (s, t)_\theta - \sum_{k=1}^{n-M} (s, \tilde{\chi}_k)_\theta (t, \tilde{\chi}_k)_\theta - (s, r)_\theta = (s, R_B t)_\theta, \end{aligned} \tag{4.3.4}$$

since $(p, t)_\theta = (s, r)_\theta = 0$ due to $p, r \in \Pi_m(\mathbb{R}^d)$. Thus the self-adjointness of R_B is established.

Next we derive from equation (4.3.2)

$$(s, R_B s)_\theta = (s, s)_\theta - \sum_{k=1}^{n-M} (s, \tilde{\chi}_k)_\theta^2 \leq (s, s)_\theta. \tag{4.3.5}$$

Consider the equality case. Then $(s, \tilde{\chi}_k)_\theta$ vanishes for all $k = 1, \ldots, n - M$, since all terms of the sum are nonnegative. By Lemma 3.3.1, the functions $\tilde{\chi}_k$, $k = 1, \ldots, n - M$, form, together with a basis of $\Pi_m(\mathbb{R}^d)$, a basis for S_θ. It follows that s is orthogonal to all functions in S_θ and hence s is a

polynomial of degree at most m. Thus inequality (4.3.5) is strict for all $s \in S_\theta$, $s \notin \Pi_m(\mathbb{R}^d)$.

It follows directly from (4.3.3) that $R_B s \neq s$ for any $s \in S_\theta$ which is not a polynomial of degree at most m. On the other hand, $R_B p = 0$ for all $p \in \Pi_m(\mathbb{R}^d)$. Hence $R_B s \neq s$ for all nonzero $s \in S_\theta$ and the proof of the theorem is complete. □

The self-adjointness of the operator R_B is highly useful when the Krylov subspace method described in Chapter 6 is applied. Algorithm B has the advantage over Algorithm A that it needs a smaller number of operations per iteration, but it has the disadvantage that it diverges in certain cases as seen in Table 4.1, while convergence has been established for Algorithm A. In the first section of the next chapter and in Chapter 6 we will present the useful techniques that enforce convergence for Algorithm B.

Chapter 5

Other methods

The last two chapters described two algorithms and their properties in detail. This chapter presents two more techniques which were developed from insights gained from the analysis of Algorithm A. The convergence analysis in Section 3.2 shows the important role of the minimization of the semi-inner product. The first section of this chapter takes this concept further and introduces a line search. It is demonstrated in Section 3.3 that Algorithm A is equivalent to solving the positive definite system of equations given by (3.3.19) using Gauss–Seidel iteration. The second technique described in this chapter is a conjugate gradient method for solving (3.3.19). This method is more efficient than the Gauss–Seidel iteration for large $n-M$ (Iserles, 1996). We describe a different implementation of the conjugate gradient technique using the matrix $W = VV^T$ following Powell (1996), where V is defined after the proof of Theorem 3.3.2 in Section 3.3. We establish some properties of W, in order to justify further the preconditioning of Θ by $\sigma_\theta V^T$ from the left and by V from the right.

5.1 Line search

It is shown in Section 3.2 that the key to understanding the convergence of Algorithm A is that the sequence $(s^*-s^{(\ell)}, s^*-s^{(\ell)})_\theta$ decreases monotonically. Therefore, instead of letting the new approximation at the end of the ℓ-th iteration be $s^{(\ell)} + t^{(\ell)}$, we define

$$s^{(\ell+1)} = s^{(\ell)} + \omega^{(\ell)} t^{(\ell)}, \tag{5.1.1}$$

where $\omega^{(\ell)}$ is chosen to minimize $(s^* - s^{(\ell)} - \omega\, t^{(\ell)}, s^* - s^{(\ell)} - \omega\, t^{(\ell)})_\theta$, $\omega \in \mathbb{R}$. When $(t^{(\ell)}, t^{(\ell)})_\theta > 0$, this choice gives $\omega^{(\ell)}$ the value

$$\omega^{(\ell)} = \frac{(s^* - s^{(\ell)}, t^{(\ell)})_\theta}{(t^{(\ell)}, t^{(\ell)})_\theta}, \tag{5.1.2}$$

and can be regarded as a line search along $t^{(\ell)}$.

The semi-inner product $(t^{(\ell)}, t^{(\ell)})_\theta$ is zero if and only if $t^{(\ell)} \in \Pi_m(\mathbb{R}^d)$. Therefore, attention has to be paid to the case when $t^{(\ell)}$ is some polynomial of degree at most m, because then $(s^* - s^{(\ell)} - \omega\, t^{(\ell)}, s^* - s^{(\ell)} - \omega\, t^{(\ell)})_\theta$ takes the value $(s^* - s^{(\ell)}, s^* - s^{(\ell)})_\theta$ for all $\omega \in \mathbb{R}$. Therefore we will only consider algorithms where

$$t^{(\ell)} \in \Pi_m(\mathbb{R}^d) \text{ implies } s^* = s^{(\ell)} + t^{(\ell)}. \tag{5.1.3}$$

For Algorithms A and B we deduce from equations (3.5.1) and (4.3.1) that

$$\begin{aligned} s_A^{(\ell+1)} &= s_A^{(\ell)} + (I - R_A)(s^* - s_A^{(\ell)}), \\ s_B^{(\ell+1)} &= s_B^{(\ell)} + (I - R_B)(s^* - s_B^{(\ell)}), \end{aligned} \tag{5.1.4}$$

where the subscript denotes the algorithm which generates the sequence. Specifically, if Algorithm A is the underlying algorithm, then $t^{(\ell)} = (I - R_A)(s^* - s^{(\ell)})$ and thus $s^* = s^{(\ell)} + t^{(\ell)} + R_A(s^* - s^{(\ell)})$, but if Algorithm B

is used, then $t^{(\ell)} = (I - R_B)(s^* - s^{(\ell)})$ and hence $s^* = s^{(\ell)} + t^{(\ell)} + R_B(s^* - s^{(\ell)})$. If $t^{(\ell)} \in \Pi_m(\mathbb{R}^d)$, then

$$(s^* - s^{(\ell)} - t^{(\ell)}, s^* - s^{(\ell)})_\theta = (s^* - s^{(\ell)}, s^* - s^{(\ell)})_\theta, \tag{5.1.5}$$

since the semi-inner product vanishes on $\Pi_m(\mathbb{R}^d)$. Now for Algorithm A, $s^* - s^{(\ell)} - t^{(\ell)} = R_A(s^* - s^{(\ell)})$ and for Algorithm B, $s^* - s^{(\ell)} - t^{(\ell)} = R_B(s^* - s^{(\ell)})$. It follows in both cases from equations (3.5.3), (4.3.3) and (5.1.5) that $s^* - s^{(\ell)} \in \Pi_m(\mathbb{R}^d)$. By Theorems 3.5.1 and 4.3.1, both operators R_A and R_B annihilate polynomials of degree at most m and therefore $s^* = s^{(\ell)} + t^{(\ell)}$, if $t^{(\ell)} \in \Pi_m(\mathbb{R}^d)$.

We avoid the zero dominator in formula (5.1.2) by including the following stopping criterion. If there exists $\omega \in \mathbb{R}$ such that

$$\max\{|s^*(\underline{x}_i) - s^{(\ell)}(\underline{x}_i) - \omega\, t^{(\ell)}(\underline{x}_i)| : i = 1, \ldots, n\} \leq \text{TOL}, \tag{5.1.6}$$

where TOL is the specified accuracy, then the final approximation $s^{(\ell+1)}$ is set to $s^{(\ell)} + \omega\, t^{(\ell)}$ and the algorithm terminates, the scalar ω being chosen as follows. For all $i = 1, \ldots, n$ such that $t^{(\ell)}(\underline{x}_i) = 0$, it is checked that the inequality $|s^*(\underline{x}_i) - s^{(\ell)}(\underline{x}_i)| \leq \text{TOL}$ holds. If this test is successful, then we seek the borders of the interval $[a, b]$ in which ω should lie. Let j be the least integer such that $t^{(\ell)}(\underline{x}_j) \neq 0$. It is not possible that $t^{(\ell)}(\underline{x}_i) = 0$ for all $i = 1, \ldots, n$, because this implies by (5.1.3) $s^{(\ell)}(\underline{x}_i) = s^*(\underline{x}_i)$, $i = 1, \ldots, n$, and the method would have terminated on the previous iteration. We let a be the smaller and b be the larger of the two ratios

$$\frac{s^*(\underline{x}_j) - s^{(\ell)}(\underline{x}_j) - \text{TOL}}{t^{(\ell)}(\underline{x}_j)} \quad \text{and} \quad \frac{s^*(\underline{x}_j) - s^{(\ell)}(\underline{x}_j) + \text{TOL}}{t^{(\ell)}(\underline{x}_j)}. \tag{5.1.7}$$

Similarly, for $i = j+1, \ldots, n$ such that $t^{(\ell)}(\underline{x}_i) \neq 0$, we calculate the interval $[\hat{a}, \hat{b}]$, where $\hat{a}$ is the smaller and $\hat{b}$ is the larger of the two ratios

$$\frac{s^*(\underline{x}_i) - s^{(\ell)}(\underline{x}_i) - \text{TOL}}{t^{(\ell)}(\underline{x}_i)} \quad \text{and} \quad \frac{s^*(\underline{x}_i) - s^{(\ell)}(\underline{x}_i) + \text{TOL}}{t^{(\ell)}(\underline{x}_i)}. \tag{5.1.8}$$

We then replace the interval $[a, b]$ by the intersection of $[a, b]$ with $[\hat{a}, \hat{b}]$, if it exists. Otherwise there is no $\omega \in \mathbb{R}$ such that (5.1.6) holds, so the technique sets $\omega^{(\ell)}$ to the value (5.1.2) and lets $s^{(\ell+1)}$ be the function (5.1.1). When there exists a final interval $[a, b]$, the scalar ω is set to its midpoint, i.e. $(a + b)/2$.

If the specified accuracy TOL is greater than zero, then this may allow termination when $s^* \neq s^{(\ell)} + t^{(\ell)}$. If, however, $s^* = s^{(\ell)} + t^{(\ell)}$, then the technique calculates the intersection of the intervals with boundaries $1 - \text{TOL}/t^{(\ell)}(\underline{x}_j)$ and $1 + \text{TOL}/t^{(\ell)}(\underline{x}_j)$ for $j = 1, \ldots, n$ such that $t^{(\ell)}(\underline{x}_j) \neq 0$, where the upper boundary is the larger of the two values. The midpoint of the intersection is 1 and thus ω is set to 1 and the algorithm terminates with $s^{(\ell+1)} = s^{(\ell)} + t^{(\ell)} = s^*$.

The numerical experiments of Chapter 7 show that Algorithm B may diverge. The following theorem, however, proves that the given line search enforces convergence.

Theorem 5.1.1 *The algorithm described in this section with TOL > 0, which generates a sequence of estimates $s^{(\ell)}$, $\ell = 1, 2, \ldots$, following rules (5.1.1) and (5.1.2), starting with $s_1 \equiv 0$ and letting $t^{(\ell)} = (I - R_B)\,(s^* - s^{(\ell)})$, achieves the stopping criterion (5.1.6) for some $\omega \in \mathbb{R}$ after a finite number of iterations.*

Proof: Suppose the algorithm fails to terminate. Each choice of $\omega = \omega^{(\ell)}$ minimizes $(s^* - s^{(\ell)} - \omega t^{(\ell)}, s^* - s^{(\ell)} - \omega t^{(\ell)})_\theta$, $\omega \in \mathbb{R}$. Therefore the sequence $(s^* - s^{(\ell)}, s^* - s^{(\ell)})_\theta$, $\ell = 1, 2, \ldots$, is infinite and is monotonically decreasing. It is also bounded below by zero and thus converges to a limit as ℓ tends to infinity. This implies that the difference between two elements in the sequence tends to zero as $\ell \to \infty$. The definitions of $s^{(\ell+1)}$ in (5.1.1) and $\omega^{(\ell)}$

in (5.1.2) provide the formula

$$(s^* - s^{(\ell)}, s^* - s^{(\ell)})_\theta - (s^* - s^{(\ell+1)}, s^* - s^{(\ell+1)})_\theta = \frac{(s^* - s^{(\ell)}, t^{(\ell)})_\theta^2}{(t^{(\ell)}, t^{(\ell)})_\theta}, \quad (5.1.9)$$

for this expression that tends to zero as ℓ tends to infinity.

Using the definition $t^{(\ell)} = (I - R_B)(s^* - s^{(\ell)})$ and equation (4.3.2) with $s = s^* - s^{(\ell)}$, the square root of the numerator in (5.1.9) can be written as

$$\begin{aligned} (s^* - s^{(\ell)}, t^{(\ell)})_\theta &= \left(s^* - s^{(\ell)}, (I - R_B)(s^* - s^{(\ell)})\right)_\theta \\ &= \sum_{k=1}^{n-M} (s^* - s^{(\ell)}, \tilde{\chi}_k)_\theta^2. \end{aligned} \quad (5.1.10)$$

The denominator on the other hand is

$$\begin{aligned} (t^{(\ell)}, t^{(\ell)})_\theta &= \left((I - R_B)(s^* - s^{(\ell)}), (I - R_B)(s^* - s^{(\ell)})\right)_\theta \\ &= \sum_{i,j=1}^{n-M} (s^* - s^{(\ell)}, \tilde{\chi}_i)_\theta (s^* - s^{(\ell)}, \tilde{\chi}_j)_\theta (\tilde{\chi}_i, \tilde{\chi}_j)_\theta \\ &\leq C \sum_{i=1}^{n-M} (s^* - s^{(\ell)}, \tilde{\chi}_i)_\theta^2, \end{aligned} \quad (5.1.11)$$

where C is a positive number that is independent of ℓ. It follows that expression (5.1.9) is bounded below by $C^{-1} \sum_{k=1}^{n-M} (s^* - s^{(\ell)}, \tilde{\chi}_k)_\theta^2$. Therefore, for $k = 1, \ldots, n - M$, we deduce

$$(s^* - s^{(\ell)}, \tilde{\chi}_k)_\theta \to 0 \text{ as } \ell \to \infty. \quad (5.1.12)$$

Let $p^{(\ell)}$ be the unique polynomial of degree at most m that interpolates $s^* - s^{(\ell)}$ at $\underline{x}_{n-M+1}, \ldots, \underline{x}_n$. Since the semi-inner product vanishes on the space $\Pi_m(\mathbb{R}^d)$, $(s^* - s^{(\ell)} - p^{(\ell)}, \tilde{\chi}_k)_\theta$ equals $(s^* - s^{(\ell)}, \tilde{\chi}_k)_\theta$, $k = 1, \ldots, n - M$. Hence $(s^* - s^{(\ell)} - p^{(\ell)}, \tilde{\chi}_k)_\theta$ also tends to zero as ℓ tends to infinity.

Now the definition (2.2.22) of the semi-inner product and equation (3.3.2) give the condition

$$(s^* - s^{(\ell)} - p^{(\ell)}, \tilde{\chi}_k)_\theta = \sigma_\theta(\sigma_\theta \hat{\lambda}_{kk})^{-1/2} \sum_{j \in \mathcal{L}_k} \hat{\lambda}_{kj} \left[s^*(\underline{x}_j) - s^{(\ell)}(\underline{x}_j) - p^{(\ell)}(\underline{x}_j) \right], \tag{5.1.13}$$

so we can write the conclusion of the last paragraph in the form

$$\lim_{\ell \to \infty} \left\{ \sum_{j=k}^{n} \hat{\lambda}_{kj} \left[s^*(\underline{x}_j) - s^{(\ell)}(\underline{x}_j) - p^{(\ell)}(\underline{x}_j) \right] \right\} = 0, \quad k = 1, \ldots, n - M, \tag{5.1.14}$$

where $\hat{\lambda}_{kj}$ is defined to be zero for $j \notin \mathcal{L}_k$ and where we drop the constant, nonzero factor $\sigma_\theta(\sigma_\theta \hat{\lambda}_{kk})^{-1/2}$. As in the proof of convergence for Algorithm A in Section 3.2, we consider the equations (5.1.14) in reverse order, remembering the identity

$$s^*(\underline{x}_j) - s^{(\ell)}(\underline{x}_j) - p^{(\ell)}(\underline{x}_j) = 0, \qquad j = n - M + 1, \ldots, n, \tag{5.1.15}$$

and that each $\hat{\lambda}_{kk}$ is nonzero. It follows by induction on k that

$$\lim_{\ell \to \infty} \left\{ s^*(\underline{x}_k) - s^{(\ell)}(\underline{x}_k) - p^{(\ell)}(\underline{x}_k) \right\} = 0, \qquad k = n - M, n - M - 1, \ldots, 1. \tag{5.1.16}$$

On the other hand, using equation (4.3.2) with $s = s^* - s^{(\ell)}$, we find that the definition $t^{(\ell)} = (I - R_B)(s^* - s^{(\ell)})$ yields $t^{(\ell)} = \sum_{k=1}^{n-M} (s^* - s^{(\ell)}, \tilde{\chi}_k)_\theta \, \tilde{\chi}_k + p^{(\ell)}$, since $p^{(\ell)}(\underline{x}_i) = s^*(\underline{x}_i) - s^{(\ell)}(\underline{x}_i)$, $i = n - M + 1, \ldots, n$. Thus

$$\lim_{\ell \to \infty} \left\{ t^{(\ell)}(\underline{x}_i) - p^{(\ell)}(\underline{x}_i) \right\} = \lim_{\ell \to \infty} \left\{ \sum_{k=1}^{n-M} (s^* - s^{(\ell)}, \tilde{\chi}_k)_\theta \, \tilde{\chi}_k(\underline{x}_i) \right\} = 0, \tag{5.1.17}$$

the last identity being due to the property (5.1.12). Equations (5.1.15), (5.1.16) and (5.1.17) imply

$$\lim_{\ell \to \infty} \max \left\{ |s^*(\underline{x}_i) - s^{(\ell)}(\underline{x}_i) - t^{(\ell)}(\underline{x}_i) : i = 1, \ldots, n \right\} = 0. \tag{5.1.18}$$

Therefore, for sufficiently large ℓ, the choice $\omega = 1$ satisfies the stopping criterion (5.1.6). It follows that the theorem is true $\Box$

The Krylov subspace method presented in Chapter 6 extends the idea of line searches even further. There the search directions are modified so that they are mutually orthogonal with respect to the semi-inner product (2.2.22). The self-adjointness of operator R_B makes this construction easy, but more effort is needed when using operator R_A.

Each iteration of Algorithm B with a line search requires far fewer operations than an iteration of Algorithm A, and we have proved the convergence for both methods. Thus, if q is suitably large, Algorithm B with an added line search is usually much more efficient than Algorithm A. The next section considers another version of these algorithms.

5.2 Conjugate gradients

By now we have encountered several iterative techniques in our efforts to calculate the solution

$$\begin{aligned} s^*(\underline{x}) &= \sum_{j=1}^{n} \lambda_j^* \, \theta(\underline{x} - \underline{x}_j) + p^*(\underline{x}) \\ &= \sum_{k=1}^{n-M} \alpha_k^* \, \tilde{\chi}_k(\underline{x}) + r^*(\underline{x}), \qquad \underline{x} \in \mathbb{R}^d, \end{aligned} \tag{5.2.1}$$

of the interpolation equations (1.1.1). The method described in this section was developed from the results presented in Sections 3.3 and 4.2. There we noted that Algorithms A and B calculate approximations to the vector of coefficients $\underline{\alpha}^* = (\alpha_1^*, \ldots, \alpha_{n-M}^*)^T$ by applying Gauss–Seidel and Jacobi iterations respectively to the system of equations

$$\sigma_\theta \, V^T \Theta V \underline{\alpha}^* = \sigma_\theta V^T \underline{f}, \tag{5.2.2}$$

where the $n \times (n-M)$ matrix V is specified in Section 3.3. By the interpolation conditions, r^* is the unique polynomial of degree at most m which interpolates $s^* - \sum_{k=1}^{n-M} \alpha_k^* \tilde{\chi}_k$ at the last M data points, where as before we assume a reordering if necessary such that $\underline{x}_{n-M+1}, \ldots, \underline{x}_n$ are polynomially unisolvent. The vector $\underline{\lambda}^* = (\lambda_1^*, \ldots, \lambda_n^*)^T$ of radial basis function coefficients is given by $\underline{\lambda}^* = V\underline{\alpha}^*$. The polynomial p^* can be calculated by adding to r^*, for $k = 1, \ldots, n-M$, the polynomial part of $\tilde{\chi}_k$ multiplied by α_k^*.

It was shown in Lemma 3.3.3 that the $(n-M)\times(n-M)$ matrix $\sigma_\theta V^T \Theta V$ is positive definite, since Θ is conditionally definite. It follows that the solution $\underline{\alpha}^*$ of the system (5.2.2) can be calculated by the conjugate gradient method. The details of this approach are as follows.

Let F be the quadratic function

$$F(\underline{\alpha}) = \sigma_\theta \, \underline{\alpha}^T V^T \underline{f} - \sigma_\theta \tfrac{1}{2} \, \underline{\alpha}^T V^T \Theta V \underline{\alpha}, \qquad \underline{\alpha} \in \mathbb{R}^{n-M}, \tag{5.2.3}$$

which is concave and bounded above. The conjugate gradient method finds the unique maximizer of F, because equation (5.2.2) is equivalent to the condition $\underline{\nabla} F(\underline{\alpha}^*) = 0$.

We let $\underline{\alpha}^{(1)} = 0$ initially and $\underline{\alpha}^{(\ell)}$ denotes the estimate of $\underline{\alpha}^*$ at the beginning of the ℓ-th iteration. Termination occurs if the gradient vector

$$\underline{\gamma}^{(\ell)} = \underline{\nabla} F(\underline{\alpha}^{(\ell)}) = \sigma_\theta \, V^T \underline{f} - \sigma_\theta \, V^T \Theta V \underline{\alpha}^{(\ell)} \tag{5.2.4}$$

is zero. Then $\underline{\alpha}^{(\ell)}$ is the maximizer of F and it is returned as the solution of system (5.2.2). Otherwise, a search direction $\underline{\beta}^{(\ell)}$ is chosen. For $\ell = 1$, $\underline{\beta}^{(1)}$ is chosen to be the steepest ascent direction $\underline{\gamma}^{(1)} = \sigma_\theta V^T \underline{f}$. The idea of the conjugate gradient method is to construct a sequence of search directions that are conjugate with respect to $\sigma_\theta V^T \Theta V$, which means

$$\sigma_\theta \, \underline{\beta}^{(i)T} V^T \Theta V \underline{\beta}^{(j)} = 0, \qquad i \neq j. \tag{5.2.5}$$

This property with line searches ensures that the conjugate gradient method finds $\underline{\alpha}^*$ after at most $n-M$ iterations (Fletcher, 1987), except for the effects of computer rounding errors. The exact choice of $\underline{\beta}^{(\ell)}$ is described below.

We let $\underline{\alpha}^{(\ell+1)}$ be the vector

$$\underline{\alpha}^{(\ell+1)} = \underline{\alpha}^{(\ell)} + \varpi^{(\ell)} \underline{\beta}^{(\ell)}, \tag{5.2.6}$$

where $\varpi^{(\ell)}$ is chosen to maximize

$$F(\underline{\alpha}^{(\ell)} + \varpi \underline{\beta}^{(\ell)}) = F(\underline{\alpha}^{(\ell)}) + \varpi\, \underline{\beta}^{(\ell)T} \underline{\gamma}^{(\ell)} - \sigma_\theta \tfrac{1}{2}\, \varpi^2\, \underline{\beta}^{(\ell)T} V^T \Theta V \underline{\beta}^{(\ell)}, \quad \varpi \in \mathbb{R}. \tag{5.2.7}$$

Remembering $\sigma_\theta^2 = 1$, we deduce the value

$$\varpi^{(\ell)} = \sigma_\theta \, \frac{\underline{\beta}^{(\ell)T} \underline{\gamma}^{(\ell)}}{\underline{\beta}^{(\ell)T} V^T \Theta V \underline{\beta}^{(\ell)}}. \tag{5.2.8}$$

Therefore we obtain the relation

$$F(\underline{\alpha}^{(\ell+1)}) = F(\underline{\alpha}^{(\ell)}) + \sigma_\theta \, \tfrac{1}{2} \, \frac{(\underline{\beta}^{(\ell)T} \underline{\gamma}^{(\ell)})^2}{\underline{\beta}^{(\ell)T} V^T \Theta V \underline{\beta}^{(\ell)}}, \tag{5.2.9}$$

which is a strict increase if $\underline{\beta}^{(\ell)T} \underline{\gamma}^{(\ell)} \neq 0$, since $\sigma_\theta V^T \Theta V$ is positive definite. Thus we are moving towards the extremum. The new gradient

$$\underline{\gamma}^{(\ell+1)} = \underline{\gamma}^{(\ell)} - \sigma_\theta \, \varpi^{(\ell)} \, V^T \Theta V \underline{\beta}^{(\ell)} \tag{5.2.10}$$

is orthogonal to $\underline{\beta}^{(\ell)}$ by this choice of $\varpi^{(\ell)}$.

In practice, the technique terminates if $\| \underline{\gamma}^{(\ell)} \|_2$ is sufficiently small. Otherwise, the method causes the search directions to be conjugate with respect to the matrix $\sigma_\theta V^T \Theta V$ by applying the formula

$$\underline{\beta}^{(\ell)} = \underline{\gamma}^{(\ell)} - \frac{\underline{\beta}^{(\ell-1)T} V^T \Theta V \underline{\gamma}^{(\ell)}}{\underline{\beta}^{(\ell-1)T} V^T \Theta V \underline{\beta}^{(\ell-1)}} \, \underline{\beta}^{(\ell-1)}, \tag{5.2.11}$$

for $\ell \geq 2$ and $\underline{\gamma}^{(\ell)} \neq 0$.

The description of the basic algorithm is now complete, but the product $W = VV^T$ allows a different implementation, which is important to techniques such as those developed by Dyn *et al.* (1983, 1986).

Instead of $\underline{\alpha}^{(\ell)}$, $\underline{\beta}^{(\ell)}$ and $\underline{\gamma}^{(\ell)}$ in $\mathbb{R}^{n-M}$, we work with the vectors $\underline{\lambda}^{(\ell)} = V\underline{\alpha}^{(\ell)}$, $\underline{\mu}^{(\ell)} = V\underline{\beta}^{(\ell)}$ and $\underline{\nu}^{(\ell)} = V\underline{\gamma}^{(\ell)}$ in $\mathbb{R}^n$. The previous choice $\underline{\alpha}^{(1)} = 0$ gives the values $\underline{\lambda}^{(1)} = 0$ and

$$\underline{\nu}^{(1)} = V\underline{\gamma}^{(1)} = \sigma_\theta W\underline{f}, \tag{5.2.12}$$

since $\underline{\gamma}^{(1)} = \sigma_\theta V^T\underline{f}$, for the beginning of the first iteration. Further the gradient (5.2.4) is zero if and only if

$$\underline{\nu}^{(\ell)} = V\underline{\gamma}^{(\ell)} = \sigma_\theta\, W\underline{f} - \sigma_\theta\, W\Theta\underline{\lambda}^{(\ell)} \tag{5.2.13}$$

is zero, where the last expression follows from (5.2.4). Therefore in practice the iterations are usually stopped when $\| \underline{\nu}^{(\ell)} \|_2$ is sufficiently small, and then we let $\underline{\lambda}^{(\ell)}$ be the calculated estimate of $\underline{\lambda}^*$. Alternatively, when the search direction for the first iteration is required, it is

$$\underline{\mu}^{(1)} = V\underline{\beta}^{(1)} = \sigma_\theta W\underline{f}. \tag{5.2.14}$$

Further, for $\ell \geq 2$, equation (5.2.11) implies that we should pick the vector

$$\underline{\mu}^{(\ell)} = \underline{\nu}^{(\ell)} - \frac{\underline{\mu}^{(\ell-1)T}\Theta\underline{\nu}^{(\ell)}}{\underline{\mu}^{(\ell-1)T}\Theta\underline{\mu}^{(\ell-1)}}\ \underline{\mu}^{(\ell-1)} \tag{5.2.15}$$

as the new search direction. Substituting formula (5.2.4) into the numerator of expression (5.2.8), we see that the step length of the ℓ-th iteration has the value

$$\varpi^{(\ell)} = \frac{\underline{f}^T\underline{\mu}^{(\ell)} - \underline{\lambda}^{(\ell)T}\Theta\underline{\mu}^{(\ell)}}{\underline{\mu}^{(\ell)T}\Theta\underline{\mu}^{(\ell)}}. \tag{5.2.16}$$

Finally, by analogy with equations (5.2.6) and (5.2.10), the iteration calculates

$$\underline{\lambda}^{(\ell+1)} = \underline{\lambda}^{(\ell)} + \varpi^{(\ell)}\underline{\mu}^{(\ell)} \quad \text{and} \quad \underline{\nu}^{(\ell+1)} = \underline{\nu}^{(\ell)} - \sigma_\theta\,\varpi^{(\ell)}\,W\Theta\underline{\mu}^{(\ell)}. \tag{5.2.17}$$

In the following paragraph we consider another choice for W. By the definition of the semi-inner product (2.2.22), $(s^*,s^*)_\theta$ equals $\sigma_\theta\sum_{j=1}^n\lambda_j^*s^*(\underline{x}_j) = \sigma_\theta\underline{f}^T\underline{\lambda}^*$, using the interpolation equations (1.1.1). Thus, employing expression (2.1.21), we deduce

$$(s^*,s^*)_\theta = \sigma_\theta\,\underline{f}^T\Lambda\underline{f}. \tag{5.2.18}$$

If W satisfies $\underline{f}^TW\underline{f} = (s^*,s^*)_\theta$ for all $\underline{f}\in\mathbb{R}^n$, then $W = \sigma_\theta\,\Lambda$. Further, if $W = \sigma_\theta\,\Lambda$, then the first search direction is, using $\sigma_\theta^2 = 1$, $\underline{\mu}^{(1)} = \sigma_\theta\,W\underline{f} = \Lambda\underline{f} = \underline{\lambda}^*$ by (2.1.21) and the algorithm terminates within one iteration. One of the key ideas of the highly useful method of Dyn et al (1986) in the case of thin plate splines in two dimensions ($\sigma_\theta = 1$, $m = 3$) is to use the relation (5.2.18) to generate a choice of W that is a reasonable approximation to Λ and to use it in the conjugate gradient method. The aim is to find an approximation such that

$$(s^*,s^*) \approx \underline{f}^TW\underline{f}. \tag{5.2.19}$$

Dyn et al (1986) estimate the integral (s^*,s^*), defined by (2.2.3), by a quadrature rule with positive coefficients using a triangulation with vertices at the data points. The values of the second derivatives required are approximated on each triangle by a linear combination of the data f_i, $i = 1,\ldots,n$, which vanishes, if the relevant data can be interpolated by a linear polynomial. Due to the squares of second derivatives in (s^*,s^*) and the positive coefficients of the quadrature rule, this estimate of $\underline{f}^T\Lambda\underline{f}$ is nonnegative and its dependence on $\underline{f}$ takes the form $\underline{f}^TW\underline{f}$ for a particular symmetric $n\times n$ matrix W that

can be calculated. If we ensure that at least one of the approximations of the second derivatives is nonzero whenever $\underline{f}$ does not lie in the column space of P, then this estimate is zero if and only if all the data can be interpolated by a constant or linear polynomial. Hence W has $n-3$ positive and three zero eigenvalues. Now the element W_{ij} is nonzero only if at least one of the second derivative approximations involves both f_i and f_j. Therefore, since each approximation is usually derived from a local cluster of interpolation points, W is generally sparse. These properties make this choice of W highly useful. In general, every symmetric $n \times n$ matrix W with $n-M$ positive and M zero eigenvalues satisfying $WP = 0$ is suitable for the conjugate gradient method (Powell, 1996).

The following theorem gives some of the properties of our choice $W = VV^T$, which suggest that $\sigma_\theta V^T$ and V are very suitable as left and right preconditioners of Θ.

Theorem 5.2.1 *The matrix $W = VV^T$ satisfies $WP = 0$. It is positive semi-definite with $n-M$ positive and M zero eigenvalues (exactly as the matrix $\sigma_\theta \Lambda$). If the functions $\hat{\chi}_k$, $k = 1, \ldots, n-q$, are orthogonal to each other and to every function in T_θ with respect to the semi-inner product (2.2.22), then the estimate $\underline{f}^T W \underline{f}$ for $(s^*, s^*)_\theta$ is exact for every $\underline{f} \in \mathbb{R}^n$.*

Proof: It follows from equation (3.3.17) that $V^T P = 0$ and therefore $WP = VV^T P = 0$. The matrix V also has maximal rank $n-M$, since it is lower triangular and its diagonal elements $V_{ii} = \sigma_\theta \sqrt{\sigma_\theta \hat{\lambda}_{ii}}$ are nonzero. Hence, since $W = VV^T$, it follows that W has $n-M$ positive and M zero eigenvalues.

Lemma 3.3.1 states that if $\hat{\chi}_k$, $k = 1, \ldots, n-q$, are orthogonal to each other and to every function in T_θ, then $s^* = \sum_{k=1}^{n-M} \alpha_k^* \tilde{\chi}_k + r^*$ with $\alpha_k^* =$

$(s^*, \tilde{\chi}_k)_\theta$. Hence in this case, using $\sigma_\theta^2 = 1$,

$$\begin{aligned} \underline{f}^T W \underline{f} &= (\sigma_\theta V^T \underline{f})^T (\sigma_\theta V^T \underline{f}) = \sum_{k=1}^{n-M} (s^*, \tilde{\chi}_k)_\theta^2 = \sum_{k=1}^{n-M} (s^*, (s^*, \tilde{\chi}_k)_\theta \tilde{\chi}_k)_\theta \\ &= (s^*, \sum_{k=1}^{n-M} \alpha_k^* \tilde{\chi}_k)_\theta = (s^*, s^* - r^*)_\theta = (s^*, s^*)_\theta, \end{aligned} \tag{5.2.20}$$

where the second equation is derived from the fact that the k-th row of V^T contains the vector of coefficients of $\tilde{\chi}_k$, $k = 1, \ldots, n - M$, and the last equation follows, since the semi-inner product vanishes on the space $\Pi_m(\mathbb{R}^d)$. Therefore the last statement of the theorem is also true. □

The conjugate gradient method described here is actually equivalent to the Krylov subspace method of the next chapter applied to Algorithm B. Therefore it will receive further attention in Section 6.4.

Chapter 6

Krylov subspace methods

The technique described in this chapter is the culmination of the work that is presented in this thesis. The idea of line searches, given in Section 5.1, will be extended by constructing search directions that are mutually orthogonal with respect to the semi-inner product (2.2.22). The estimate $s^{(\ell+1)}$ at the end of the ℓ-th iteration is then the best approximation to s^* from a certain ℓ-dimensional subspace of S_θ. We will see that this property ensures convergence within at most $n - M + 1$ iterations. In practice, this method is very successful. The number of iterations needed to achieve an accuracy of say 10^{-8} is usually far less than $n - M + 1$, the actual number being below ten in most cases.

A description of the method is given in the first section, followed by a section where we analyse the properties of the technique. In the third section we describe how the mutually orthogonal search directions are chosen. Then it is shown in Section 6.4 that the Krylov subspace method applied to Algorithm B is equivalent to the conjugate gradient method of Section 5.2.

6.1 Description

An algorithm to which the Krylov subspace method can be applied lets the new approximation at the end of the ℓ-th iteration be

$$s^{(\ell)} + L(s^* - s^{(\ell)}), \tag{6.1.1}$$

where L is a linear operator from S_θ to S_θ which has the following properties

$$\left.\begin{array}{lcll} s \in S_\theta \text{ and } s \neq 0 & \Rightarrow & Ls \neq 0 & \text{(nonsingularity)} \\ s \in \Pi_m(\mathbb{R}^d) & \Rightarrow & Ls = s & \text{(polynomial reproduction)} \\ s \in S_\theta \text{ and } s \notin \Pi_m(\mathbb{R}^d) & \Rightarrow & (s, Ls)_\theta > 0 & \text{(ellipticity)} \end{array}\right\}. \tag{6.1.2}$$

An additional requirement is that Ls, $s \in S_\theta$, depends only on values of s at the data points $\underline{x}_1, \ldots, \underline{x}_n$.

Employing equation (5.1.4), if Algorithm A is considered, we let $L = I - R_A$, whereas if Algorithm B is the underlying algorithm, we let $L = I - R_B$. It follows directly from the results of Theorem 3.5.1 and Theorem 4.3.1 that both operators $I - R_A$ and $I - R_B$ satisfy the requirements (6.1.2). In particular, $(I - R_A)s \neq 0$ and $(I - R_B)s \neq 0$ for $s \in S_\theta$, $s \neq 0$, follows from $R_A s \neq s$ and $R_B s \neq s$. The polynomial reproduction property is ensured by $R_A p = 0$ and $R_B p = 0$ for all $p \in \Pi_m(\mathbb{R}^d)$. The ellipticity of $I - R_A$ and $I - R_B$ follows from $(s, R_A s)_\theta < (s, s)_\theta$ and $(s, R_B s)_\theta < (s, s)_\theta$ for all $s \in S_\theta$, $s \notin \Pi_m(\mathbb{R}^d)$. Both Algorithms A and B employ only function values at the data points. Thus the Krylov subspace method can be applied to both of these algorithms.

We denote the current approximation at the beginning of the ℓ-th iteration by $s^{(\ell)}$. First $s^{(1)}$ is set to be identically zero. If $f_i = s^*(\underline{x}_i) = 0$ for $i = 1, \ldots, n$, which implies $s^* \equiv 0$, the algorithm terminates. In the ℓ-th iteration, $\ell \geq 1$, a search direction $t^{(\ell)}$ is generated in the following way. First

$u^{(\ell)} = L(s^* - s^{(\ell)})$ is calculated. There is no problem in forming Ls^*, since we only need to know the function values $s^*(\underline{x}_j) = f_j$, $j = 1, \ldots, n$. If $\ell = 1$ or if, for $\ell > 1$, $(u^{(\ell)}, t^{(j)})_\theta = 0$ holds for $j = 1, \ldots, \ell - 1$, $t^{(\ell)}$ is chosen to be $u^{(\ell)}$. Otherwise $t^{(\ell)}$ is set to a linear combination of $u^{(\ell)}$ and all previous search directions $t^{(j)}$, $j = 1, \ldots, \ell - 1$, with a nonzero contribution from $u^{(\ell)}$, such that $t^{(\ell)}$ satisfies

$$(t^{(\ell)}, t^{(j)})_\theta = 0, \qquad j = 1, \ldots, \ell - 1. \tag{6.1.3}$$

This defines $t^{(\ell)}$ uniquely up to a polynomial of degree at most m apart from a scaling factor. In Section 6.3 we will see in detail, how the mutual orthogonality of the search directions is achieved.

As in Section 5.1, we take care of the difficulty arising from a search direction that is a polynomial of degree at most m by including the following stopping criterion. If there exists $\omega \in \mathbb{R}$ such that

$$\max\{|s^*(\underline{x}_i) - s^{(\ell)}(\underline{x}_i) - \omega\, t^{(\ell)}(\underline{x}_i)| : i = 1, \ldots, n\} \leq \text{TOL}, \tag{6.1.4}$$

where TOL is the specified accuracy, then the final approximation $s^{(\ell+1)}$ is set to $s^{(\ell)} + \omega\, t^{(\ell)}$ and the algorithm terminates. We will see in Theorem 6.2.1 of the next section that such an ω exists if $t^{(\ell)}$ lies in $\Pi_m(\mathbb{R}^d)$. The choice of the scalar ω is described in Section 5.1.

Otherwise, if inequality (6.1.4) does not hold for any $\omega \in \mathbb{R}$, the technique finds $\omega^{(\ell)}$ by minimizing

$$(s^* - s^{(\ell)} - \omega\, t^{(\ell)}, s^* - s^{(\ell)} - \omega\, t^{(\ell)})_\theta, \qquad \omega \in \mathbb{R}. \tag{6.1.5}$$

Thus $\omega^{(\ell)}$ takes the value

$$\omega^{(\ell)} = \frac{(s^* - s^{(\ell)}, t^{(\ell)})_\theta}{(t^{(\ell)}, t^{(\ell)})_\theta}. \tag{6.1.6}$$

The possibility $(t^{(\ell)}, t^{(\ell)})_\theta = 0$ does not occur here, since in this case $t^{(\ell)} \in \Pi_m(\mathbb{R}^d)$, so the stopping criterion (6.1.4) would cause termination. The scalar $\omega^{(\ell)}$ is nonzero, because it is shown in Theorem 6.2.1 below that if $(s^* - s^{(\ell)}, t^{(\ell)})_\theta = 0$, then termination occurs. The Krylov subspace technique lets the new approximation $s^{(\ell+1)}$ be $s^{(\ell)} + \omega^{(\ell)} t^{(\ell)}$. The choice (6.1.6) of $\omega^{(\ell)}$ causes $s^* - s^{(\ell+1)}$ to be orthogonal to $t^{(\ell)}$ with respect to the semi-inner product (2.2.22).

This concludes the description of the Krylov subspace method. The next section presents the properties which provide its success and gives an analysis of convergence.

6.2 Analysis

We are going to justify the term "Krylov subspace method". Let L_ℓ be the Krylov subspace of S_θ which is the span of the functions $L^j s^*$, $j = 1, \ldots, \ell$. Thus $L_1 \subset L_2 \subset L_3 \subset \cdots$ holds. It can be seen by induction that both $t^{(\ell)}$ and $s^{(\ell+1)}$ are in L_ℓ, the argument being as follows. For $\ell = 1$, we have $t^{(1)} = u^{(1)} = L(s^* - s^{(1)}) = Ls^* \in L_1$ and $s^{(2)} = s^{(1)} + \omega^{(1)} t^{(1)} = \omega^{(1)} Ls^* \in L_1$, since $s^{(1)} \equiv 0$. For $\ell > 1$, $t^{(\ell)}$ is a linear combination of $L(s^* - s^{(\ell)})$ and the previous search directions. Now $Ls^* \in L_1 \subset L_\ell$, $Ls^{(\ell)} \in L_\ell$, since $s^{(\ell)} \in L_{\ell-1}$, and the previous search directions $t^{(j)} \in L_j \subset L_\ell$, $j = 1, \ldots, \ell - 1$. Thus $t^{(\ell)}$ is an element of L_ℓ and the same follows for $s^{(\ell+1)} = s^{(\ell)} + \omega^{(\ell)} t^{(\ell)}$. Other properties of the Krylov subspace technique are stated in the following theorem.

Theorem 6.2.1 *For $\ell \geq 1$, let the algorithm specified in Section 6.1 generate the approximations $s^{(1)}, \ldots, s^{(\ell)}$ by line searches along mutually orthogonal search directions $t^{(1)}, \ldots, t^{(\ell)}$. Then the following statements are true.*

(a) If $(t^{(\ell)}, t^{(\ell)})_\theta = 0$, *then* $(s^* - s^{(\ell)}, t^{(\ell)})_\theta = 0$. *If* $(s^* - s^{(\ell)}, t^{(\ell)})_\theta = 0$, *then there exists* $\omega \in \mathbb{R}$ *such that inequality (6.1.4) holds, so the algorithm terminates.*

If the algorithm does not terminate in the ℓ*-th iteration, then*

(b) the subspace L_ℓ *of* S_θ *is spanned by* $t^{(1)}, \ldots, t^{(\ell)}$, *and*

(c) the difference $s^* - s^{(\ell+1)}$ *is orthogonal to every element in* L_ℓ *with respect to the semi-inner product (2.2.22). (Thus* $s^{(\ell+1)}$ *is the function* s *that minimizes* $(s^* - s, s^* - s)_\theta$, $s \in L_\ell$.*)*

Proof: The theorem is proved by induction on ℓ. For $\ell = 1$, we suppose $(t^{(1)}, t^{(1)})_\theta = 0$ in statement (a). Thus $t^{(1)} \in \Pi_m(\mathbb{R}^d)$ and hence $(s^* - s^{(1)}, t^{(1)})_\theta = 0$. If $(s^* - s^{(1)}, t^{(1)})_\theta = 0$, then $(s^*, Ls^*)_\theta = 0$ follows, since $s^{(1)} \equiv 0$ and since $t^{(1)} = Ls^*$. Further, the ellipticity of L implies that s^* is a polynomial of degree at most m. Then $t^{(1)}$ equals s^*, since L reproduces polynomials of degree at most m. We do not have $t^{(1)}(\underline{x}_i) = s^*(\underline{x}_i) = 0$ for $i = 1, \ldots, n$, because this would imply $s^* \equiv 0$ and the algorithm would have terminated before the first iteration. In this case, the technique described in Section 5.1 calculates the intersection of the intervals with boundaries $1 - \mathrm{TOL}/s^*(\underline{x}_j)$ and $1 + \mathrm{TOL}/s^*(\underline{x}_j)$ for $j = 1, \ldots, n$ such that $s^*(\underline{x}_j) \neq 0$, where the upper boundary is the larger of the two values. The scalar ω is then chosen to be the midpoint of the final interval. Hence ω is chosen to be 1, inequality (6.1.4) holds for $\ell = 1$, and the algorithm terminates.

Alternatively, if there is no termination for $\ell = 1$, then the space L_1 is spanned by Ls^*, and the first search direction $t^{(1)}$ is chosen to be Ls^*. Thus statement (b) is true for $\ell = 1$. Turning to statement (c), if the algorithm does not terminate, then the semi-inner product $(s^* - s^{(2)}, t^{(1)})_\theta$ is zero by the choice of $\omega^{(1)}$. Thus $s^* - s^{(2)}$ is orthogonal to all elements in L_1.

We now suppose that statements (a), (b) and (c) hold for any positive integer ℓ and deduce that they remain true if ℓ is increased by one.

Considering statement (a), if $(t^{(\ell+1)}, t^{(\ell+1)})_\theta = 0$, then $t^{(\ell+1)} \in \Pi_m(\mathbb{R}^d)$ and thus $(s^* - s^{(\ell+1)}, t^{(\ell+1)})_\theta = 0$. Now $t^{(\ell+1)}$ is a linear combination of $u^{(\ell+1)}$ and $t^{(j)}$, $j = 1, \ldots, \ell$, with a nonzero contribution from $u^{(\ell+1)}$. Therefore, if $(s^* - s^{(\ell+1)}, t^{(\ell+1)})_\theta = 0$, then the difference $s^* - s^{(\ell+1)}$ is orthogonal to $u^{(\ell+1)}$, since $s^* - s^{(\ell+1)}$ is orthogonal to all elements in L_ℓ, including $t^{(1)}, \ldots, t^{(\ell)}$. Further, $u^{(\ell+1)}$ equals $L(s^* - s^{(\ell+1)})$ and thus $(s^* - s^{(\ell+1)}, L(s^* - s^{(\ell+1)}))_\theta = 0$, so the ellipticity property (6.1.2) of L implies that $s^* - s^{(\ell+1)}$ is a polynomial of degree at most m, say p. Since L reproduces polynomials, $u^{(\ell+1)} = L(s^* - s^{(\ell+1)}) = s^* - s^{(\ell+1)} = p$ follows, and $(u^{(\ell+1)}, t^{(j)})_\theta = 0$ holds for $j = 1, \ldots, \ell$, because $u^{(\ell+1)}$ is a polynomial. Therefore $t^{(\ell+1)}$ is chosen to be $u^{(\ell+1)} = p$. It is not possible for $t^{(\ell+1)}(\underline{x}_i) = p(\underline{x}_i) = s^*(\underline{x}_i) - s^{(\ell+1)}(\underline{x}_i) = 0$ to occur for $i = 1, \ldots, n$, because then the algorithm would have terminated in the previous iteration. Hence the technique described in Section 5.1 sets the scalar ω to the midpoint 1 of the intersections of the intervals with boundaries $1 - \mathrm{TOL}/p(\underline{x}_j)$ and $1 + \mathrm{TOL}/p(\underline{x}_j)$ for $j = 1, \ldots, n$ such that $p(\underline{x}_j) \neq 0$. Thus inequality (6.1.4) is achieved and the algorithm terminates, so statement (a) is true.

Statement (b) and the inductive hypothesis imply that the subspace L_ℓ, which is defined to be spanned by $L^j s^*$, $j = 1, \ldots, \ell$, is spanned by $t^{(1)}, \ldots, t^{(\ell)}$. Therefore it is necessary to prove that $L^{\ell+1} s^*$ is a linear combination of $t^{(\ell+1)}$ and elements of L_ℓ. Now $t^{(\ell+1)}$ is a linear combination of $u^{(\ell+1)}$ and $t^{(1)}, \ldots, t^{(\ell)} \in L_\ell$ with a nonzero contribution from $u^{(\ell+1)}$. We have $u^{(\ell+1)} \in L_{\ell+1}$, since $u^{(\ell+1)} = L(s^* - s^{(\ell+1)})$, $Ls^* \in L_1$ and $Ls^{(\ell+1)} \in L_{\ell+1}$, due to $s^{(\ell+1)} \in L_\ell$. If $u^{(\ell+1)}$ were in L_ℓ, then $(s^* - s^{(\ell+1)}, u^{(\ell+1)})_\theta = 0$ would hold, since $s^* - s^{(\ell+1)}$ is orthogonal to all elements in L_ℓ. We have seen in the pre-

vious paragraph, however, that convergence occurs in this case. Hence, if the algorithm does not terminate, then $u^{(\ell+1)}$ lies in $L_{\ell+1}$, but not in L_ℓ. Thus $t^{(\ell+1)}$ is a linear combination of $L^{\ell+1}s^*$ and elements of L_ℓ, where the term involving $L^{\ell+1}s^*$ does not vanish. The assertion of statement (b) follows.

To prove statement (c), it is sufficient to prove that $(s^*-s^{(\ell+2)},t^{(j)})_\theta=0$ holds for $j=1,\ldots,\ell+1$, since $L_{\ell+1}$ is spanned by $t^{(1)},\ldots,t^{(\ell+1)}$. When the algorithm does not terminate, the choice of $\omega^{(\ell+1)}$ ensures $(s^*-s^{(\ell+2)},t^{(\ell+1)})_\theta=0$. For $j=1,\ldots,\ell$, we have the identity $(s^*-s^{(\ell+2)},t^{(j)})_\theta=(s^*-s^{(\ell+1)},t^{(j)})_\theta-\omega^{(\ell+1)}(t^{(\ell+1)},t^{(j)})_\theta$. The first term is zero, because $s^*-s^{(\ell+1)}$ is orthogonal to all elements in L_ℓ, and the second term vanishes, since $t^{(\ell+1)}$ was chosen to be orthogonal to all previous search directions. Thus statement (c) also remains true if ℓ is increased by one, which concludes the proof. $\Box$

The Krylov subspace technique can be viewed as a conjugate direction method for minimizing the quadratic form $(s^*-s,s^*-s)_\theta$, the variables being the coefficients of the basis functions subject to the constraints (2.1.12) and the coefficients of the polynomial of degree at most m. The expression conjugate is equivalent to orthogonality with respect to the semi-inner product (2.2.22). By statement (c) of Theorem 6.2.1, the approximant $s^{(\ell+1)}$ gives indeed the least value of the semi-inner product $(s^*-s,s^*-s)_\theta$ for all $s\in L_\ell$. Further, the application of the operator L can be viewed as a preconditioner.

We now prove that the given method terminates in exact arithmetic. The search directions $t^{(\ell)}$, $\ell=1,2,\ldots$, are mutually orthogonal to each other with respect to the semi-inner product (2.2.22). If the algorithm does not terminate within ℓ iterations, then statement (a) of Theorem 6.2.1 shows that $(t^{(j)},t^{(j)})_\theta\neq 0$, $j=1,\ldots,\ell$, so $t^{(\ell)}$ is not in $\Pi_m(\mathbb{R}^d)$. In order to establish the linear independence of the search directions so far, we let ρ_j, $j=1,\ldots,\ell$, be coefficients such that $s=\sum_{j=1}^{\ell}\rho_j t^{(j)}+p$ is the zero element of S_θ, where p is

a suitable polynomial of degree at most m. The mutual orthogonality of the search directions yields

$$(s,s)_\theta = \sum_{j=1}^{\ell} \rho_j^2 (t^{(j)}, t^{(j)})_\theta = 0. \tag{6.2.1}$$

Since $t^{(j)} \notin \Pi_m(\mathbb{R}^d)$, we have $(t^{(j)}, t^{(j)})_\theta > 0$ and hence every ρ_j is zero. It follows from $s \equiv 0$ and the polynomial unisolvency that p is the zero polynomial. Thus $t^{(1)}, \ldots, t^{(\ell)}$ and basis elements $p_1, \ldots, p_M$ of $\Pi_m(\mathbb{R}^d)$ are $\ell+M$ linearly independent elements of S_θ, which implies $\ell+M \leq \dim(S_\theta) = n$ for any integer ℓ such that termination does not occur in the ℓ-th iteration. Thus termination must occur on an iteration whose number is at most $n - M + 1$. We present this result as the main theorem of this chapter.

Theorem 6.2.2 *Let the technique specified in Section 6.1 generate the sequence of functions $s^{(\ell)}$, $\ell = 1, 2, \ldots$, using a nonsingular linear operator L which is elliptic and reproduces $\Pi_m(\mathbb{R}^d)$. Then inequality (6.1.4) holds for some $\ell \leq n - M + 1$, so the algorithm terminates within at most $n - M + 1$ iterations.*

As we will see in Chapter 7 which considers numerical experiments, the number of iterations in practice is generally far less than $n - M + 1$. To achieve an accuracy of TOL $= 10^{-8}$, for example, the number of iterations needed is less then ten in most cases for large enough q, when we apply the Krylov subspace technique to Algorithms A and B.

6.3 The choice of search directions

This section will describe in detail how mutually orthogonal search directions are chosen. The section is divided into two parts. First we consider a general

nonsingular, elliptic operator $L : S_\theta \to S_\theta$ which reproduces polynomials of degree at most m. Next, we examine the case where L enjoys the additional property of self-adjointness.

6.3.1 The general choice of search directions

Starting with $s^{(1)} \equiv 0$, the first search direction is $t^{(1)} = Ls^*$. Suppose the approximations $s^{(1)}, \ldots, s^{(k)}$ to s^* and the search directions $t^{(1)}, \ldots, t^{(k-1)}$ have already been constructed for $k \geq 2$ such that the search directions are mutually orthogonal with respect to the semi-inner product (2.2.22), and suppose termination did not occur in the $(k-1)$-th iteration.

The next search direction $t^{(k)}$ is then generated in the following way. First $u^{(k)} = L(s^* - s^{(k)})$ is calculated. If $(u^{(k)}, t^{(j)})_\theta = 0$ holds for $j = 1, \ldots, k-1$, then $t^{(k)}$ is set to $u^{(k)}$. Otherwise we let $u_0^{(k)} = u^{(k)}$. For every integer j in $[1, k-1]$, we calculate

$$u_j^{(k)} = u_{j-1}^{(k)} + \rho_j t^{(j)}, \tag{6.3.1}$$

where ρ_j is chosen to minimize

$$(u_{j-1}^{(k)} + \rho\, t^{(j)} - s^* + s^{(k)}, u_{j-1}^{(k)} + \rho\, t^{(j)} - s^* + s^{(k)})_\theta, \qquad \rho \in \mathbb{R}. \tag{6.3.2}$$

This gives ρ_j the value

$$\rho_j = \frac{(s^* - s^{(k)} - u_{j-1}^{(k)}, t^{(j)})_\theta}{(t^{(j)}, t^{(j)})_\theta}, \tag{6.3.3}$$

the denominator $(t^{(j)}, t^{(j)})_\theta$ being nonzero, because, if $t^{(j)}$ is a polynomial of degree at most m, convergence would have occurred already. In other words, we conduct a line search along $t^{(j)}$ for $j = 1, \ldots, k-1$ such that $u_j^{(k)}$ lies as near as possible to the difference $s^* - s^{(k)}$. The choice (6.3.3) of ρ_j causes the

difference between $u_j^{(k)}$ and $s^* - s^{(k)}$ to be orthogonal to $t^{(j)}$. Further, $u_{k-1}^{(k)}$ satisfies

$$(u_{k-1}^{(k)} - s^* + s^{(k)}, t^{(j)})_\theta = 0, \qquad j = 1, \ldots, k-1, \tag{6.3.4}$$

because the directions $t^{(1)}, \ldots, t^{(k-1)}$ are mutually orthogonal. Also $(s^* - s^{(k)}, t^{(j)})_\theta$, $j = 1, \ldots, k-1$, vanishes by statement (c) of Theorem 6.2.1 with $\ell = k-1$, since $t^{(1)}, \ldots, t^{(k-1)} \in L_{k-1}$. Thus $(u_{k-1}^{(k)}, t^{(j)})_\theta = 0$ holds for $j = 1, \ldots, k-1$. Next the new search direction is defined by $t^{(k)} = u_{k-1}^{(k)}$ and it enjoys the required orthogonality properties.

As can be seen in Chapter 7, the Krylov subspace technique applied to Algorithm A, where $L = I - R_A$, results in a very low number of iterations, but the operational cost per iteration is very high. In the next section we describe how the search directions are chosen if L is also self-adjoint. We will find that the self-adjointness property is very useful.

6.3.2 The choice of search directions for a self-adjoint operator

Constructing mutually orthogonal search directions, when using a linear operator L with the properties (6.1.2) and the self-adjointness property

$$(s, Lt)_\theta = (Ls, t)_\theta, \qquad s, t \in S_\theta, \tag{6.3.5}$$

is straightforward. The search direction $t^{(1)}$ is chosen to be Ls^*. Suppose the approximations $s^{(1)}, \ldots, s^{(k)}$ and the mutually orthogonal search directions $t^{(1)}, \ldots, t^{(k-1)}$ have already been constructed for $k \geq 2$, and suppose the algorithm did not terminate in the $(k-1)$-th iteration. It is enough to let

$$t^{(k)} = u^{(k)} - \frac{(u^{(k)}, t^{(k-1)})_\theta}{(t^{(k-1)}, t^{(k-1)})_\theta} t^{(k-1)}, \tag{6.3.6}$$

where $u^{(k)} = L(s^* - s^{(k)})$. We see that $(t^{(k)}, t^{(k-1)})_\theta = 0$ holds, so it remains to prove that $(t^{(k)}, t^{(j)})_\theta = 0$ is achieved for $j = 1, \ldots, k-2$. Since $(t^{(k-1)}, t^{(j)})_\theta = 0$ holds for $j = 1, \ldots, k-2$, the self-adjointness described by (6.3.5), the definition $u^{(k)} = L(s^* - s^{(k)})$ and equation (6.3.6) yield

$$\begin{aligned}(t^{(k)}, t^{(j)})_\theta &= \left(L(s^* - s^{(k)}) - \frac{(u^{(k)}, t^{(k-1)})_\theta}{(t^{(k-1)}, t^{(k-1)})_\theta} t^{(k-1)},\, t^{(j)} \right)_\theta \\ &= (s^* - s^{(k)}, Lt^{(j)})_\theta, \qquad j = 1, \ldots, k-2. \end{aligned} \tag{6.3.7}$$

The search direction $t^{(j)}$, $j = 1, \ldots, k-2$, lies in $L_j \subset L_{k-2}$, which implies $Lt^{(j)} \in L_{k-1}$. By statement (c) of Theorem 6.2.1 with $\ell = k-1$, $s^* - s^{(k)}$ is orthogonal to every element in L_{k-1}. Hence the semi-inner product in (6.3.7) is zero, so the search direction $t^{(k)}$ has the required orthogonality properties.

This construction of search directions is used when the Krylov subspace method is applied to Algorithm B, since by Theorem 4.3.1 R_B and thus $I - R_B$ is self-adjoint. The next section considers the Krylov subspace technique applied to Algorithm B more closely.

6.4 The Krylov subspace technique applied to Algorithm B

This section shows that the Krylov subspace method applied to Algorithm B with operator $L = I - R_B$ is equivalent to the conjugate gradient method of Section 5.2. The radial basis function coefficients of $s^{(\ell)}$, $t^{(\ell)}$ and $u^{(\ell)}$ in the Krylov subspace method are given explicitly in the expressions

$$\begin{aligned} s^{(\ell)}(\underline{x}) &= \sum_{i=1}^{n} \lambda_i^{(\ell)} \theta(\underline{x} - \underline{x}_i) + \text{ polynomial}, \\ t^{(\ell)}(\underline{x}) &= \sum_{i=1}^{n} \mu_i^{(\ell)} \theta(\underline{x} - \underline{x}_i) + \text{ polynomial}, \end{aligned}$$

$$u^{(\ell)}(\underline{x}) \;=\; \sum_{i=1}^{n} \nu_i^{(\ell)} \theta(\underline{x} - \underline{x}_i) + \text{ polynomial.} \tag{6.4.1}$$

Equations (5.2.12), (5.2.13), (5.2.14), (5.2.15) and (5.2.17) on the other hand give initial values and formulae for the re-estimation of the radial basis function coefficients in the conjugate gradient method.

Theorem 6.4.1 *The Krylov subspace method applied to Algorithm B ($L = I - R_B$) revises the vectors $\underline{\lambda}^{(\ell)} = (\lambda_1^{(\ell)}, \ldots, \lambda_n^{(\ell)})^T$, $\underline{\mu}^{(\ell)} = (\mu_1^{(\ell)}, \ldots, \mu_n^{(\ell)})^T$ and $\underline{\nu}^{(\ell)} = (\nu_1^{(\ell)}, \ldots, \nu_n^{(\ell)})^T$ of $s^{(\ell)}$, $t^{(\ell)}$ and $u^{(\ell)}$ in the same way as the conjugate gradient method of Section 5.2.*

Proof: Firstly, since $s^{(1)} \equiv 0$, both methods pick $\underline{\lambda}^{(1)} = 0$ initially. Then the Krylov subspace technique lets the first search direction be the function

$$\begin{aligned} t^{(1)}(\underline{x}) \;&=\; u^{(1)}(\underline{x}) = (I - R_B) s^*(\underline{x}) = \sum_{k=1}^{n-M} (s^*, \tilde{\chi}_k)_\theta \tilde{\chi}_k(\underline{x}) + \text{polynomial} \\ &=\; \sum_{k=1}^{n-M} \sum_{i \in \mathcal{L}_k} \sum_{j \in \mathcal{L}_k} \frac{\hat{\lambda}_{ki} \hat{\lambda}_{kj}}{\hat{\lambda}_{kk}} s^*(\underline{x}_j) \theta(\underline{x} - \underline{x}_i) + \text{polynomial} \\ &=\; \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_\theta W_{ij} f_j \, \theta(\underline{x} - \underline{x}_i) + \text{polynomial}, \end{aligned} \tag{6.4.2}$$

using (4.3.2), (3.3.1) and (3.3.2). The last identity depends on the interpolation equations (1.1.1) and on the fact that W_{ij} equals $\sum_{k=1}^{n-M} (\hat{\lambda}_{ki} \hat{\lambda}_{kj})/(\sigma_\theta \hat{\lambda}_{kk})$, defining $\hat{\lambda}_{kl}$ to be zero for $l \notin \mathcal{L}_k$, $l = 1, \ldots, n$. Hence $\underline{\mu}^{(1)} = \underline{\nu}^{(1)} = \sigma_\theta W \underline{f}$ holds. Comparing this with (5.2.12) and (5.2.14), it follows that both techniques start with the same initial vectors.

We are now going to show that the ℓ-th iterations of both methods generate identical vectors $\underline{\nu}^{(\ell)}$, $\underline{\mu}^{(\ell)}$ and $\underline{\lambda}^{(\ell+1)}$ given $\underline{\mu}^{(\ell-1)}$ and $\underline{\lambda}^{(\ell)}$. First the Krylov subspace method calculates

$$u^{(\ell)}(\underline{x}) \;=\; (I - R_B)(s^* - s^{(\ell)})(\underline{x}) = \sum_{k=1}^{n-M} (s^* - s^{(\ell)}, \tilde{\chi}_k)_\theta \tilde{\chi}_k(\underline{x}) + \text{ polynomial}$$

$$= \sum_{k=1}^{n-M} \sum_{i\in\mathcal{L}_k} \sum_{j\in\mathcal{L}_k} \frac{\hat{\lambda}_{ki}\hat{\lambda}_{kj}}{\hat{\lambda}_{kk}} \left(s^*(\underline{x}_j) - s^{(\ell)}(\underline{x}_j)\right) \theta(\underline{x} - \underline{x}_i) + \text{ polynomial}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} \sigma_\theta W_{ij} \left(f_j - s^{(\ell)}(\underline{x}_j)\right) \theta(\underline{x} - \underline{x}_i) + \text{ polynomial.} \qquad (6.4.3)$$

Thus $\underline{\nu}^{(\ell)}$ is the matrix $\sigma_\theta W$ times the vector whose components are $f_j - s^{(\ell)}(\underline{x}_j)$, $j = 1, \ldots, n$. We can neglect the polynomial part of $s^{(\ell)}$ because Theorem 5.2.1 states that W satisfies $WP = 0$. The values of $s^{(\ell)}$ at the data points without the polynomial part are the components of the vector $\Theta\underline{\lambda}^{(\ell)}$, so

$$\underline{\nu}^{(\ell)} = \sigma_\theta W \underline{f} - \sigma_\theta W \Theta \underline{\lambda}^{(\ell)}, \qquad (6.4.4)$$

which is just expression (5.2.13).

Theorem 4.3.1 states that R_B is self-adjoint, so $I - R_B$ is also self-adjoint. Thus, when the Krylov subspace technique is applied to Algorithm B, the search directions are chosen according to (6.3.6). Using the definition of the semi-inner product (2.2.22), we deduce for the Krylov subspace technique

$$\underline{\mu}^{(\ell)} = \underline{\nu}^{(\ell)} - \frac{(u^{(\ell)}, t^{(\ell-1)})_\theta}{(t^{(\ell-1)}, t^{(\ell-1)})_\theta} \underline{\mu}^{(\ell-1)} = \underline{\nu}^{(\ell)} - \frac{\underline{\nu}^{(\ell)T}\Theta\underline{\mu}^{(\ell-1)}}{\underline{\mu}^{(\ell-1)T}\Theta\underline{\mu}^{(\ell-1)}} \underline{\mu}^{(\ell-1)}, \qquad (6.4.5)$$

which agrees with equation (5.2.15). Hence both methods calculate $\underline{\nu}^{(\ell)}$ and $\underline{\mu}^{(\ell)}$ in the same way.

Further, the Krylov subspace method chooses the step length (6.1.6), which we write in the form

$$\omega^{(\ell)} = \frac{(s^*, t^{(\ell)})_\theta - (s^{(\ell)}, t^{(\ell)})_\theta}{(t^{(\ell)}, t^{(\ell)})_\theta} = \frac{\underline{f}^T\underline{\mu}^{(\ell)} - \underline{\lambda}^{(\ell)T}\Theta\underline{\mu}^{(\ell)}}{\underline{\mu}^{(\ell)T}\Theta\underline{\mu}^{(\ell)}}. \qquad (6.4.6)$$

This is exactly the value of $\varpi^{(\ell)}$ given by equation (5.2.16). Thus the step lengths $\varpi^{(\ell)}$ and $\omega^{(\ell)}$ are the same in both methods. Since $s^{(\ell+1)} = s^{(\ell)} + \omega^{(\ell)}t^{(\ell)}$, the identity $\underline{\lambda}^{(\ell+1)} = \underline{\lambda}^{(\ell)} + \varpi^{(\ell)}\underline{\mu}^{(\ell)}$ follows, which is the first part

of equation (5.2.17). Using this recurrence relation for $\underline{\lambda}^{(\ell+1)}$ and equation (6.4.4) with ℓ increased by one, we deduce that the Krylov subspace method sets

$$\underline{\nu}^{(\ell+1)} = \sigma_\theta W \underline{f} - \sigma_\theta W \Theta(\underline{\lambda}^{(\ell)} + \varpi^{(\ell)} \underline{\mu}^{(\ell)}) = \underline{\nu}^{(\ell)} - \sigma_\theta \varpi^{(\ell)} W \Theta \underline{\mu}^{(\ell)}, \quad (6.4.7)$$

which is the second part of equation (5.2.17). The proof is complete. □

The only two differences between the two techniques are that (1) the conjugate gradient method stops if $\| \underline{\nu}^{(\ell)} \|_2$ is sufficiently small, while the Krylov subspace method terminates if (6.1.4) holds for a suitable step length ω, and (2) the Krylov subspace method automatically revises the polynomial term on every iteration, while the conjugate gradient method calculates the polynomial term after a good approximation to the vector of radial basis function coefficients has been found.

The numerical experiments in the next chapter show that the Krylov subspace technique applied to Algorithm B is very successful, having a small number of iterations and requiring little time per iteration.

Chapter 7

Numerical examples

This chapter will compare the performances of four algorithms in numerical experiments, namely Algorithms A and B as described in Chapters 3 and 4 and Algorithms A and B with the Krylov subspace method of Chapter 6 added. All four methods are tested in two and in three dimensions and we consider two choices of radial basis functions, namely the thin plate spline basis function $\theta(\underline{x}) = \phi(\| \underline{x} \|_2) = \| \underline{x} \|_2^2 \log \| \underline{x} \|_2$ and the linear basis function $\theta(\underline{x}) = \phi(\| \underline{x} \|_2) = \| \underline{x} \|_2$, $\underline{x} \in \mathbb{R}^d$. It will become clear that the Krylov subspace technique is highly successful.

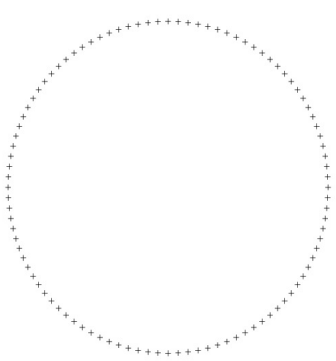
7.1 Two dimensions

We consider four kinds of distributions of the interpolation points $\underline{x}_i$, $i = 1, \ldots, n$, in two dimensions, which are taken from Faul and Powell (1998). In Problem I the points are equally spaced on the unit circle $\{ \underline{x} \in \mathbb{R}^2 : \| \underline{x} \|_2 = 1 \}$. The points of Problem II form a square grid in $\mathbb{R}^2$ that covers the unit square $[0, 1] \times [0, 1]$. In Problem III the data points are chosen randomly from

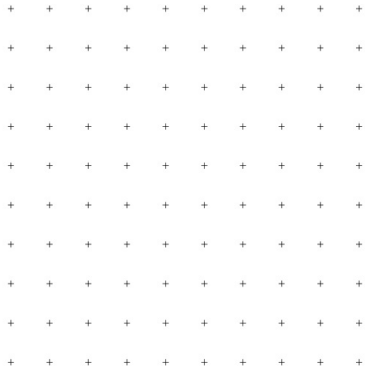
the uniform distribution on the unit disc $\{\underline{x} \in \mathbb{R}^2 : \|\underline{x}\|_2 \leq 1\}$. Problem IV was found by seeking a case where the convergence of Algorithm A is very slow. Here the points are equally spaced on one eighth of two concentric circles, half of the points being on each circle, and the radii of the circles being 1 and $1 + 10^{-5}$. These distributions are displayed in Figure 7.1.

In all problems, the right hand sides f_i, $i = 1, \ldots, n$, are independent random numbers from the uniform distribution on $[-1, 1]$. The calculation is terminated when all the moduli of the residuals $f_i - s^{(\ell)}(\underline{x}_i)$, $i = 1, \ldots, n$, are less than 10^{-8}, where $s^{(\ell)}$ denotes the approximation to s^* at the beginning of the ℓ-th iteration. The values of n and q that are employed are stated in Tables 7.1 to 7.9. The calculations are performed in double precision Fortran on a Sparc 10 workstation.

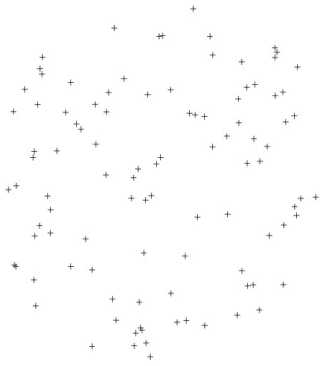
Tables 7.1 and 7.2 give the average time in seconds for the preliminary work when thin plate splines, $\phi(r) = r^2 \log r$, and the linear basis function, $\phi(r) = r$, $r = \|\underline{x}\|_2$, are used, respectively. This includes ordering the data points, constructing the sets $\mathcal{L}_k$ and calculating the coefficients $\hat{\lambda}_{kj}$, $j \in \mathcal{L}_k$, $k = 1, \ldots, n-q$. To ensure non-collinearity for thin plates splines, we reorder the data points if necessary such that the last three points $\underline{x}_{n-2}$, $\underline{x}_{n-1}$ and $\underline{x}_n$ are the data points with greatest and least x-coordinate and the data point whose perpendicular distance from the infinite straight line through these two points is the greatest. Then every set $\mathcal{L}_k$, $k = 1, \ldots, n-q$, contains these three special points. Apart from these three points, it contains the data point $\underline{x}_k$ and the $q-4$ data points in $\{\underline{x}_i : n-2 > i > k\}$ that are nearest to $\underline{x}_k$. In the case of the linear basis function, the set $\mathcal{L}_k$ contains $\underline{x}_k$ and the $q-1$ data points in $\{\underline{x}_i : i > k\}$ nearest to $\underline{x}_k$. We use a procedure described by Goodsell (2000) to construct sets of nearest neighbours. In these experiments the data points were ordered so that, if the data points are



Problem I



Problem II



Problem III

Problem IV

Figure 7.1: Distribution of the data points in the test problems.

	q			
n	10	20	30	50
400	0.22	0.68	1.54	4.62
900	0.50	1.59	3.59	11.29

Table 7.1: Average time for preliminary work for $\phi(r) = r^2 \log r$.

	q			
n	10	20	30	50
400	0.17	0.48	1.12	3.74
900	0.37	1.19	2.60	9.04

Table 7.2: Average time for preliminary work for $\phi(r) = r$.

removed in sequence from the beginning of the set $X = \{\underline{x}_1, \ldots, \underline{x}_n\}$, then the remaining points provide good coverage of the original set. Therefore, for $j = 1, \ldots, n-q$, a point which has the minimum distance to its nearest neighbour is removed next from the set of the remaining data points. Any ties are broken at random. This can be done efficiently in two dimensions by Dirichlet tessellations, but more work is required for higher dimensions. This ordering was chosen when Algorithm A was first developed and has been retained since.

It was noted by Faul and Powell (1998) and in Theorem 3.2.3 that convergence would occur in one iteration of Algorithm A, if the functions $\hat{\chi}_k$, $k = 1, \ldots, n-q$, were orthogonal to each other and to every function in T_θ (we recall that T_θ contains the functions in S_θ with centres at $\underline{x}_{n-q+1}, \ldots, \underline{x}_n$) with respect to the semi-inner product (2.2.22). In this case, Algorithm B converges in at most two iterations as shown in Theorem 4.2.2. Therefore

n	q	I	II	III	IV
400	10	0.0373	0.3174	0.2941	0.7302
	20	0.0009	0.0797	0.0886	0.6896
	30	0.0003	0.0651	0.0585	0.6528
	50	0.0001	0.0193	0.0149	0.6287
900	10	0.0377	0.2789	0.3684	0.7614
	20	0.0011	0.0890	0.0941	0.7134
	30	0.0004	0.0716	0.0376	0.6987
	50	0.0001	0.0127	0.0144	0.6576

Table 7.3: Values of expression (7.1.1) for $\phi(r) = r^2 \log r$.

we expect a correlation between performance and deviations from these orthogonality properties.

Table 7.3 indicates the size of these deviations for the thin plate spline basis function $\theta(\underline{x}) = \phi(\| \underline{x} \|_2) = \| \underline{x} \|_2^2 \log \| \underline{x} \|_2$, $\underline{x} \in \mathbb{R}^d$. Each entry in Table 7.3 states the quantity

$$\max \{ |(\tilde{\chi}_i, \tilde{\chi}_j)| : 1 \leq i < j \leq n - M \}, \tag{7.1.1}$$

which is at most one due to the identities $(\tilde{\chi}_i, \tilde{\chi}_i) = 1$, $i = 1, \ldots, n - M$, and the Cauchy–Schwarz inequality. We see that this quantity is the maximum modulus of the off-diagonal elements of the matrix of the system (3.3.9). We recall from Sections 3.4 and 4.2 that Algorithms A and B are equivalent to solving that system of equations by Gauss–Seidel and Jacobi iteration respectively. The convergence of Gauss–Seidel and Jacobi depends on the spectral radii of the matrices $L^{-1}U$ and $I_{n-M} - \sigma_\theta V^T \Theta V$ specified in Sections 3.4 and 4.2, where L is lower triangular and U is strictly upper triangular

n	q	I	II	III	IV
	10	6.7×10^{-3}	0.49	0.51	0.99
	20	1.7×10^{-4}	0.15	5.5×10^{-2}	0.94
400	30	4.0×10^{-5}	7.8×10^{-2}	1.5×10^{-2}	0.88
	50	6.1×10^{-6}	9.3×10^{-3}	6.2×10^{-3}	0.81
	10	8.5×10^{-3}	0.45	0.59	0.999
	20	2.0×10^{-4}	9.4×10^{-2}	9.2×10^{-2}	0.99
900	30	4.9×10^{-5}	2.6×10^{-2}	2.3×10^{-2}	0.96
	50	9.5×10^{-6}	1.4×10^{-2}	1.0×10^{-2}	0.89

Table 7.4: The spectral radius of $L^{-1}U$ for $\phi(r) = r^2 \log r$.

n	q	I	II	III	IV
	10	5.3×10^{-2}	0.85	1.9	33
	20	5.8×10^{-3}	0.46	0.27	9.7
400	30	2.5×10^{-3}	0.27	0.13	6.0
	50	9.0×10^{-4}	0.10	7.2×10^{-2}	3.3
	10	6.4×10^{-2}	3.0	2.6	70
	20	6.5×10^{-3}	0.40	0.35	24
900	30	2.9×10^{-3}	0.19	0.16	14
	50	1.1×10^{-3}	0.12	9.0×10^{-2}	7.6

Table 7.5: The spectral radius of $I_{n-M} - \sigma_\theta V^T \Theta V$ for $\phi(r) = r^2 \log r$.

n	q	I	II	III	IV	average time per iteration
400	10	4/4	28/14	29/18	$\star(9.0 \times 10^{-4})/64$	0.77/0.77
	20	3/3	10/7	7/7	294/32	1.40/1.41
	30	3/3	8/6	5/5	145/24	2.05/2.14
	50	2/2	5/5	4/4	85/16	3.47/3.67
900	10	5/5	24/15	36/20	$\star(5.2 \times 10^{-2})/111$	3.89/4.11
	20	3/3	8/7	8/7	$\star(5.9 \times 10^{-7})/53$	7.09/7.17
	30	3/3	6/5	6/5	443/39	10.34/10.46
	50	2/2	5/5	5/4	155/28	18.51/19.37

Table 7.6: Iteration counts of Algorithm A for $\phi(r) = r^2 \log r$.

and where $L + U = \sigma_\theta V^T \Theta V$. These spectral radii are given in Tables 7.4 and 7.5.

Each pair of numbers in Tables 7.6, 7.7, 7.8 and 7.9 gives the number of iterations needed to achieve the accuracy of 10^{-8} and the average time per iteration in seconds, without/with the Krylov subspace method for thin plate splines, $\phi(r) = r^2 \log r$, and for the linear basis function, $\phi(r) = r$, respectively. In the cases marked with $\star$ the calculations were stopped after 1000 iterations, and the maximum modulus of the final residuals $f_i - s^{(\ell)}(\underline{x}_i)$, $i = 1, \ldots, n$, is given in brackets. The cases where divergence occurred are denoted by div.

As expected, the number of iterations is small when the normalised functions $\tilde{\chi}_i$, $i = 1, \ldots, n-M$, have good orthogonality properties. Indeed, when $\phi(r) = r^2 \log r$ is employed, the algorithms perform well when expression (7.1.1) is less than 0.1, this condition being satisfied for $q \geq 20$ in Problems

n	q	I	II	III	IV	average time per iteration
400	10	8/7	112/27	div/33	div/70	0.10/0.12
	20	5/5	24/14	16/12	div/56	0.10/0.12
	30	4/4	15/10	11/8	div/42	0.11/0.12
	50	4/4	10/8	9/7	div/29	0.12/0.14
900	10	8/7	div/28	div/42	div/87	0.52/0.57
	20	5/5	21/12	18/13	div/68	0.52/0.58
	30	4/4	13/10	12/10	div/61	0.54/0.61
	50	4/4	10/8	9/8	div/51	0.56/0.63

Table 7.7: Iteration counts of Algorithm B for $\phi(r) = r^2 \log r$.

I, II and III. Expression (7.1.1) exceeds 0.2, however, for $q = 10$ and $n = 400$ in Problem III, for $q = 10$ and $n = 900$ in Problems II and III and for every q in Problem IV. Then Algorithm B diverges and also the rate of convergence of Algorithm A is slow. That bad behaviour was found before the results of Table 7.8 and 7.9 were computed, so the excellent performance for the linear radial basis function $\phi(r) = r$ was unexpected. The advantage of the linear basis function is that precautions are not taken to ensure that the points $\{\underline{x}_j : j \in \mathcal{L}_k, j \neq k\}$ are not collinear.

The results also show that the introduction of the Krylov subspace method is highly useful, especially for Problem IV when $\phi(r) = r^2 \log r$. We see, however, that the theoretical termination of the Krylov subspace method within at most $n - M + 1$ iterations is irrelevant to practical computation. Indeed, convergence occurs in practice after far fewer iterations.

The number of operations per iteration is much smaller when the linear

n	q	I	II	III	IV	average time per iteration
400	10	4/4	16/9	13/8	3/3	1.18/1.18
	20	3/3	6/6	6/5	3/3	2.17/2.17
	30	3/3	5/5	5/4	3/3	3.16/3.17
	50	3/3	4/4	4/4	3/3	5.14/5.26
900	10	3/3	9/9	9/8	4/4	6.01/6.01
	20	3/3	6/5	5/5	4/4	11.03/11.05
	30	3/3	5/5	5/4	4/4	16.09/16.10
	50	3/3	4/4	4/4	4/3	26.34/27.01

Table 7.8: Iteration counts of Algorithm A for $\phi(r) = r$.

n	q	I	II	III	IV	average time per iteration
400	10	5/5	53/15	26/13	5/5	0.17/0.18
	20	4/4	11/9	9/8	5/4	0.17/0.18
	30	4/4	9/9	7/6	5/4	0.19/0.19
	50	4/4	7/7	6/5	5/4	0.20/0.20
900	10	6/5	47/15	45/14	6/5	0.87/0.93
	20	4/4	13/9	11/8	5/4	0.88/0.94
	30	4/4	9/7	9/7	5/4	0.90/0.97
	50	4/4	8/7	7/7	5/4	0.92/0.98

Table 7.9: Iteration counts of Algorithm B for $\phi(r) = r$.

operator R_B is used instead of R_A, because function values have to be calculated less often. Indeed, R_A requires the residuals $f_j - s_{k-1}^{(\ell)}(\underline{x}_j)$, $j \in \mathcal{L}_k$, for $k = 1, \ldots, n-q$ and the residuals $f_j - s_{n-q}^{(\ell)}(\underline{x}_j)$, $j = n-q+1, \ldots, n$, while R_B requires only the residuals $f_j - s^{(\ell)}(\underline{x}_j)$, $j = 1, \ldots, n$. The average time per iteration illustrates this difference in efficiency. Thus Algorithm B combined with the Krylov subspace technique is a very good choice of an iterative technique for radial basis function interpolation.

7.2 Three dimensions

In three dimensions only one distribution of the data points $\underline{x}_i \in \mathbb{R}^3$, $i = 1, \ldots, n$, is considered. The data points are randomly distributed in the unit ball $\{\underline{x} \in \mathbb{R}^3 : \| \underline{x} \|_2 \leq 1\}$. The function values f_i, $i = 1, \ldots, n$, are independent random numbers from the uniform distribution on $[-1, 1]$. The different values of n and q are stated in Tables 7.10 to 7.12. The calculation is terminated either when all the moduli of the residuals $f_i - s^{(\ell)}(\underline{x}_i)$, $i = 1, \ldots, n$, are less than 10^{-8} or after 1000 iterations, where $s^{(\ell)}$ denotes the current approximation to s^*. The latter case is denoted by $\star$ and the maximum modulus of the residuals at termination is given in brackets. If divergence occurred, it is denoted by div as before.

Table 7.10 shows the average time for the preliminary work when the thin plate spline basis function $\phi(r) = r^2 \log r$ and the linear basis function $\phi(r) = r$ are employed. This includes constructing the sets $\mathcal{L}_k$, $k = 1, \ldots, n-q$, and determining the coefficients $\hat{\lambda}_{kj}$, $j \in \mathcal{L}_k$, $k = 1, \ldots, n-q$. In the case of the linear basis function, $\mathcal{L}_k$, $k = 1, \ldots, n-q$, contains the data point $\underline{x}_k$ itself and the $q-1$ data points in $\{\underline{x}_i : i > k\}$ nearest to $\underline{x}_k$. For thin plate splines we ensure non-planarity by reordering the data points if necessary so that

n	q	$\phi(r) = r^2 \log r$	$\phi(r) = r$
400	10	0.241	0.138
	20	0.790	0.483
	30	1.76	1.17
	50	5.29	3.86
	70	11.4	8.77
900	10	0.640	0.453
	20	1.98	1.28
	30	4.33	2.92
	50	13.1	9.66
	70	29.1	22.7
2000	10	2.10	1.60
	20	5.05	3.54
	30	10.6	7.35
	50	31.3	23.3
	70	69.4	54.7
3000	10	3.90	3.20
	20	8.59	6.23
	30	16.9	12.1
	50	48.4	36.4
	70	107	84.6
5000	10	9.17	8.14
	20	17.4	13.3
	30	31.1	23.7
	50	84.8	65.0
	70	185	148

Table 7.10: Average time for preliminary work in $\mathbb{R}^3$.

the last four points $\underline{x}_{n-3}$, $\underline{x}_{n-2}$, $\underline{x}_{n-1}$ and $\underline{x}_n$ are the points with the least and the greatest x-coordinate, the point whose perpendicular distance from the infinite straight line through these two points is greatest and the point whose perpendicular distance from the plane through these three points is greatest. Every set $\mathcal{L}_k$, $k = 1, \ldots, n-q$, contains these four points. The remaining $q-4$ points are chosen to be the nearest neighbours of $\underline{x}_k$ in $\{\underline{x}_i : n-4 \geq i \geq k\}$, including $\underline{x}_k$ itself. It is more time consuming to construct the sets $\mathcal{L}_k$, $k = 1, \ldots, n-q$, in three dimensions than in two dimensions. Here a standard search for nearest neighbours was used.

Each pair of numbers in Tables 7.11 and 7.12 gives the number of iterations needed to achieve the accuracy of 10^{-8}, and the average time per iteration. Four iterative methods were used, namely Algorithms A and B without and with the Krylov subspace technique. We see that the number of iterations needed to achieve the specified accuracy is larger for thin plate splines than for the linear radial basis function, which might be due to the inclusion of the last four points in each set $\mathcal{L}_k$, $k = 1, \ldots, n-q$, in the thin plate spline case.

If Algorithm A is used, $q = 30$ is large enough to ensure convergence within 20 iterations and $q = 50$ is large enough such that convergence occurs within 10 iterations. For Algorithm B, $q = 50$ is usually necessary to ensure convergence at all and then the number of iterations needed to achieve the specified accuracy is much larger, but the average time per iteration is much smaller than in the case of Algorithm A. If the Krylov subspace method is included, then $q = 50$ is large enough to ensure convergence within 20 iterations if thin plate splines are used, while for the linear basis function $q = 30$ already achieves this. Thus the Krylov subspace method should always be added. As in the two-dimensional case, Algorithm B requires fewer

n	q	Algorithm A without Krylov	Algorithm A with Krylov	Algorithm B without Krylov	Algorithm B with Krylov
400	10	107/0.904	42/0.914	div/0.107	79/0.109
	20	18/1.67	13/1.68	div/0.110	25/0.111
	30	12/2.45	9/2.47	78/0.121	17/0.119
	50	7/4.40	7/4.45	17/0.129	12/0.130
	70	6/6.42	6/6.43	12/0.139	9/0.140
900	10	465/4.63	79/4.65	div/0.552	154/0.564
	20	23/8.41	11/8.53	div/0.558	32/0.568
	30	14/12.4	11/12.5	div/0.579	15/0.590
	50	9/22.8	8/23.2	30/0.601	14/0.627
	70	7/33.1	7/33.5	16/0.641	12/0.650
2000	10	722/22.8	118/23.0	div/2.74	242/2.80
	20	33/42.0	20/42.4	div/2.76	42/2.83
	30	16/61.1	12/62.0	div/2.80	17/2.86
	50	10/117	9/119	52/2.85	15/2.90
	70	8/165	7/167	26/2.90	13/3.03
3000	10	$\star(4.2 \times 10^{-7})/50.6$	169/52.9	div/6.14	341/6.27
	20	36/94.1	23/95.3	div/6.16	48/6.16
	30	16/137	13/139	div/6.24	19/6.53
	50	9/264	9/266	78/6.31	17/6.64
	70	8/373	7/375	31/6.40	13/6.65
5000	10	$\star(6.4 \times 10^{-3})/144$	249/144	div/17.1	527/17.7
	20	48/263	27/268	div/17.4	57/17.8
	30	20/385	15/388	div/17.4	31/17.8
	50	10/739	9/745	402/17.5	18/17.8
	70	9/1006	8/1057	44/17.8	15/17.9

Table 7.11: Iteration counts and times for $\phi(r) = r^2 \log r$.

n	q	Algorithm A without Krylov	Algorithm A with Krylov	Algorithm B without Krylov	Algorithm B with Krylov
400	10	21/1.33	15/1.34	div/0.181	25/0.184
	20	11/2.46	9/2.47	380/0.182	16/0.186
	30	7/3.60	7/3.63	27/0.192	12/0.196
	50	6/5.90	6/6.30	11/0.202	9/0.205
	70	5/9.05	5/9.00	9/0.211	7/0.213
900	10	30/6.78	20/6.79	div/0.927	33/0.934
	20	15/12.6	11/12.6	div/0.937	19/0.937
	30	10/18.4	9/18.4	95/0.950	15/0.952
	50	7/32.0	6/32.7	16/0.974	11/0.975
	70	6/46.8	5/46.8	11/0.997	9/0.997
2000	10	41/33.5	23/35.2	div/4.60	41/4.60
	20	16/61.9	14/62.0	div/4.61	23/4.61
	30	10/90.9	9/91.0	div/4.66	17/4.68
	50	7/163	7/163	20/4.71	11/4.71
	70	6/231	6/232	13/4.77	9/4.86
3000	10	66/75.5	28/75.5	div/10.3	48/10.6
	20	18/139	13/141	div/10.3	25/10.6
	30	11/203	10/213	div/10.4	19/10.6
	50	7/370	7/378	26/10.5	13/10.6
	70	6/522	6/529	15/10.6	10/10.7
5000	10	82/212	32/212	div/28.8	55/29.7
	20	23/389	14/394	div/28.9	27/29.9
	30	13/573	11/578	div/29.0	20/30.0
	50	8/1048	7/1065	35/29.1	14/30.1
	70	6/1484	6/1550	19/29.2	10/30.1

Table 7.12: Iteration counts and times for $\phi(r) = r$.

operations per iteration than Algorithm A. This is illustrated by the iteration times. Hence Algorithm B combined with the Krylov subspace technique is the best choice.

7.3 Final remarks

We have found that Algorithm B with the Krylov subspace technique is the best choice of the iterative methods presented here. Unfortunately, however, the average time for the preliminary work is high compared to the average time per iteration for the algorithm.

The preliminary work includes constructing the sets $\mathcal{L}_k$, $k = 1, \ldots, n-q$. Goodsell (2000) presents a fast procedure for this task in two dimensions, but efficient ways to compute $\mathcal{L}_k$, $k = 1, \ldots, n-q$, in higher dimensions have to be found. For the experiments presented in the previous section, the construction of the sets $\mathcal{L}_k$, $k = 1, \ldots, n-q$, required $\mathcal{O}(nq(n-q))$ operations. There exist better algorithms for this task using tree structures.

The coefficients $\hat{\lambda}_{kj}$, $j \in \mathcal{L}_k$, $k = 1, \ldots, n-q$, are calculated before the first iteration by solving the interpolation problem on $\mathcal{L}_k$ by direct methods, which require $\mathcal{O}(q^3)$ operations for every set $\mathcal{L}_k$. Thus this work involves $\mathcal{O}(q^3(n-q))$ operations. It might be better to generate the coefficients $\hat{\lambda}_{kj}$, $j \in \mathcal{L}_k$, $k = 1, \ldots, n-q$, by an updating procedure. Thus one might take advantage of the fact that, if the $n \times n$ matrix Λ introduced in (2.1.20) is known for an interpolation problem on n points, then adding a point and extending the matrix Λ for this new interpolation problem can be done in $\mathcal{O}(n^2)$ operations.

Experiments with random orderings of the data points (retaining the last three or four points in the case of thin plate spline interpolation) have been

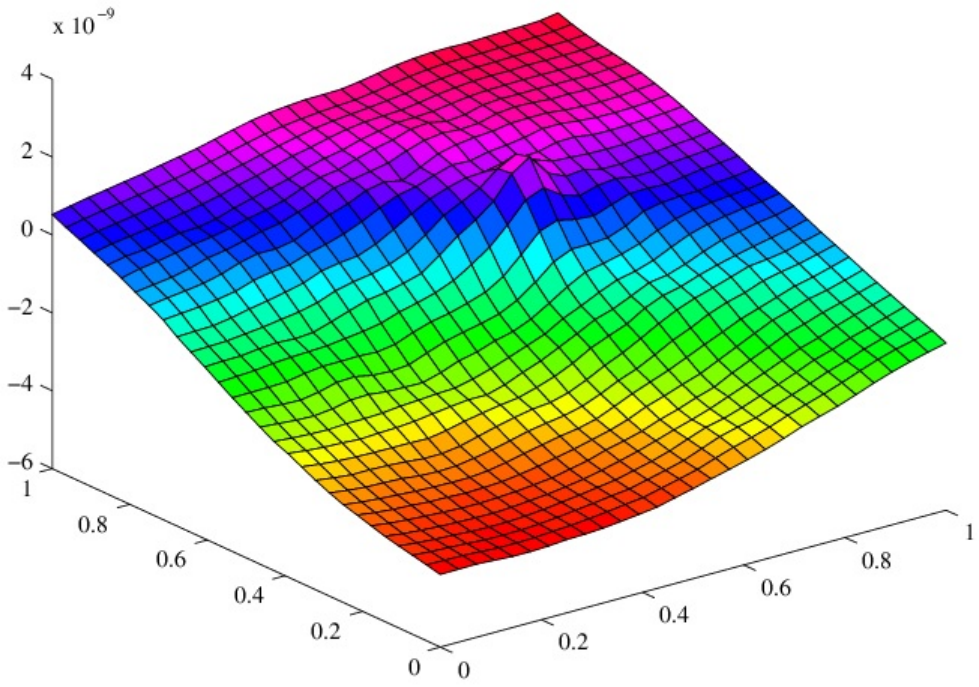


Figure 7.2: Residuals on a square grid after 11 iterations when Algorithm A with the multiquadric as basis function was used.

tried. No significant changes to the iteration counts were found. Therefore the work of ordering the data points may not be necessary.

Some preliminary numerical experiments with multiquadric functions in two dimensions have been attempted, and they were very successful. Figure 7.2 displays the residuals when Algorithm A with the multiquadric as basis functions was applied to Problem II. It only took 11 iterations to achieve an accuracy of 10^{-9}. With multiquadrics the success of the iterative method depends on the choice of the undetermined positive constant c. If c is set to

the minimal nearest neighbour distance, the algorithms perform very well, but the user might want to choose a larger constant depending on the interpolation problem. Therefore the multiquadric radial basis function was not included in the main stream of numerical experiments.

It is hoped that the importance of the semi-inner product that has been shown will assist future research on choosing the functions $\hat{\chi}_k$, $k = 1, \ldots, n - q$, for the algorithms. A different choice of functions might have better orthogonality properties. It will be interesting to see how all the given algorithms will perform with radial basis functions that are different from thin plate splines and the linear basis function.

Chapter 8

Conclusions

Since their inception (Hardy, 1971), radial basis functions have succeeded in many areas of science, including geophysics, signal processing, meteorology, orthopaedics, pattern recognition and computational fluid dynamics (Hardy, 1990). Unfortunately, a drawback is that the direct computation of the coefficients of a radial basis function interpolant to n data may require $\mathcal{O}(n^3)$ operations. It has been shown, however, that radial basis functions have a key property which makes the construction of fast iterative techniques successful. It is that they have a native vector space equipped with a semi-inner product. This key feature is the essential ingredient of the convergence analysis of the first algorithm presented here, Algorithm A. Every step either reduces the semi-norm of the difference between the required interpolant and the current approximation or leaves it unchanged. Indeed, Algorithm A constructs a set of linearly independent functions with good orthogonality properties, and then every step changes the current approximation by a multiple of one of these functions, such that the new semi-norm is minimized. To put Algorithm A into a well-known mathematical context, we have shown that it

is equivalent to solving a certain symmetric and positive definite system of equations by Gauss–Seidel iteration. This system was derived from the original system of interpolation equations by preconditioning it from the left and from the right by certain matrices. The transformed matrix contains the semi-inner products between the chosen basis functions. Thus good orthogonality properties are important to the speed of convergence.

Iterations that apply the Jacobi method are an alternative to the Gauss–Seidel technique, which brings us to Algorithm B. Unfortunately, divergence occurs in certain cases when Algorithm B is used, but this can be avoided by including a line search at the end of each iteration such that the semi-norm of the difference between the required interpolant and the current approximation is reduced.

The symmetric and positive definite system of equations can also be solved by the conjugate gradient method. Dyn *et al.* (1983) suggest a different and very successful choice of preconditioners for this interpolation problem.

We have demonstrated that the conjugate gradient method described here is equivalent to the Krylov subspace technique applied to Algorithm B. The Krylov subspace technique builds up a subspace of radial basis functions, the current approximation being the best approximation from this subspace. Each iteration enlarges this subspace by adding a radial basis function which is orthogonal to the current subspace with respect to the semi-inner product (2.2.22). The numerical experiments show that the Krylov subspace method applied to Algorithm B is very successful, giving a low number of iterations, while at the same time requiring a low number of operations per iteration. Therefore we recommend this method to compute radial basis function interpolants, but some further research should be conducted to make the pre-

liminary calculations more efficient.

It is hoped that interpolation to hundreds of thousands of data will become a routine calculation. Thus the range of applications which can take advantage of the smoothness and accuracy properties of interpolation with radial basis functions will be increased.

References

R.K. Beatson, J.B. Cherrie and C.T. Mouat (1999), "Fast fitting of radial basis functions: Methods based on GMRES iteration", *Advances in Computational Mathematics*, Vol. 11, pp. 253–270.

R.K. Beatson, G. Goodsell and M.J.D. Powell (1995), "On multigrid techniques for thin plate spline interpolation in two dimensions", *Lectures in Applied Mathematics*, Vol. 32, pp. 77–97.

R.K. Beatson and W.A. Light (1997), "Fast evaluation of radial basis functions: methods for two-dimensional polyharmonic splines", *IMA Journal of Numerical Analysis*, Vol. 17, pp. 343–372.

R.K. Beatson, W.A. Light and S. Billings (1999), "Fast solution of the radial basis function interpolation equations: Domain decomposition methods", Technical Report No. 2000/21, Department of Mathematics and Computer Science, University of Leicester.

R.K. Beatson and G.N. Newsam (1992), "Fast evaluation of radial basis functions: I", *Computers Math. Applic.*, Vol. 24, No. 12, pp. 7–19.

R.K. Beatson and M.J.D. Powell (1994), "An iterative method for thin-plate spline interpolation that employs approximations to the Lagrange functions", *Numerical Analysis 1993*, eds. D.F. Griffiths and G.A. Watson,

Longmans (Harlow).

M.D. Buhmann (1990), "Multivariate cardinal interpolation with radial basis functions", *Constr. Approx.*, Vol. 6, pp. 225–256.

J. Duchon (1975), "Fonctions-spline du type plaque mince dimension 2", Technical Report 231, Universitè de Grenoble.

J. Duchon (1976), "Fonctions-spline á energie invariante par rotation", Technical Report 27, Universitè de Grenoble.

J. Duchon (1977), "Splines minimizing rotation-invariant seminorms in Sobolev spaces", in *Constructive Theory of Functions of Several Variables, Lecture Notes in Mathematics 571*, eds. W. Schempp and K. Zeller, Springer-Verlag (Berlin), pp. 85–100.

N. Dyn (1987), "Interpolation of scattered data by radial functions", *Topics in multivariate approximation*, eds. C.K. Chui, L.L. Schumaker and F. Utreras, Academic Press (New York), pp. 47–61.

N. Dyn and D. Levin (1981), "Bell shaped basis functions for surface fitting", *Approximation Theory and Applications*, ed. Z. Ziegler, Academic Press (New York), pp. 113–129.

N. Dyn and D. Levin (1983), "Iterative solution of systems originating from integral equations and surface interpolation", *SIAM J. Numer. Anal.*, Vol. 20, pp. 377–390.

N. Dyn, D. Levin and S. Rippa (1986), "Numerical procedures for surface fitting of scattered data by radial functions", *SIAM J. Sci. Statist. Comput.*, Vol. 7, pp. 639-659.

A.C. Faul and M.J.D. Powell (1998), "Proof of convergence of an iterative technique for thin plate spline interpolation in two dimensions", *Advances in Computational Mathematics*, Vol. 11, pp. 183–192.

A.C. Faul and M.J.D. Powell (1999), "Krylov subspace methods for radial basis function interpolation", *Numerical Analysis 1999*, eds. D.F. Griffiths and G.A. Watson, CRC Press LLC, pp. 115-141.

R. Fletcher (1987), "Practical methods of optimization", John Wiley & Sons (Chichester).

R. Franke (1982), "Scattered data interpolation: tests of some methods", *Math. of Comp.*, Vol. 38, pp. 181-200.

G. Goodsell (1997), "A multigrid type method for thin plate spline interpolation on a circle", *IMA J. Numer. Anal.*, Vol. 17, pp. 321-327.

G. Goodsell (2000), "On finding the p-th nearest neighbours of scattered points in two dimensions for small p", *Computer Aided Geometric Design*, Vol. 17, pp. 387-392.

R.L. Hardy (1971), "Multiquadric equations of topography and other irregular surfaces", *J. Geophysics Res.*, Vol. 76(8), pp. 1905-1915.

R.L. Hardy (1990), "Theory and applications of the multiquadric-biharmonic method", *Comput. Math. Applic.*, Vol. 19, pp. 163–208.

A. Iserles (1996), "A First Course in the Numerical Analysis of Differential Equations", Cambridge University Press.

W. Madych and S. Nelson (1988), "Multivariate interpolation and conditionally positive definite functions", *Approx. Theory Appl.*, Vol. 4, pp. 77–89.

J. Meinguet (1979), "Multivariate interpolation at arbitrary points made simple", *Journal of Applied Mathematics and Physics*, Vol. 30, pp. 292–304.

C.A. Micchelli (1986), "Interpolation of scattered data: distance matrices and conditionally positive definite functions", *Constr. Approx.*, Vol. 2, pp. 11–22.

J. Levesley and W.Light (1999), "Direct form seminorms arising in the theory of interpolation by translates of a basis function", *Advances in Computational Mathematics*, Vol. 11, pp. 161-182.

M.J.D. Powell (1992), "The theory of radial basis function approximation in 1990", in *Advances in Numerical Analysis, Volume II: Wavelets, Subdivision Algorithms, and Radial Basis Functions*, ed. W.A. Light, Clarendon Press (Oxford), pp. 105–210.

M.J.D. Powell (1993), "Truncated Laurent expansions for the fast evaluation of thin plate splines", *Numerical Algorithms*, Vol. 5, pp. 99–120.

M.J.D. Powell (1994), "Some algorithms for thin plate spline interpolation to functions of two variables", *Advances in Computational Mathematics: New Delhi, India*, eds. H.P. Dikshit and C.A. Micchelli, World Scientific Publishing Co. (Singapore), pp. 303-319.

M.J.D. Powell (1996), "A review of methods for multivariable interpolation at scattered data points", *The State of Art in Numerical Analysis*, eds. I.S. Duff and G.A. Watson, Clarendon Press (Oxford), pp. 283–309.

M.J.D. Powell (1996), "A review of algorithms for thin plate spline interpolation in two dimensions", *Advanced Topics in Multivariate Interpola-*

tion, eds. F. Fontanella, K. Jetter and P.-J. Laurent, World Scientific Publishing Co. (Singapore), pp. 1-20.

M.J.D. Powell (1997), "A new iterative algorithm for thin plate spline interpolation in two dimensions", *Annals of Numerical Mathematics*, Vol. 4, pp. 519–527.

R. Schaback (1993), "Comparison of radial basis function interpolants", in *Multivariate Approximations: From CAGD to Wavelets*, eds. K. Jetter and F. Utreras, World Scientific (Singapore), pp. 293–305.

R. Schaback (1999), "Native Hilbert spaces for radial basis functions I", *International Series of Numerical Mathematics*, Vol. 132, pp. 255-282.

R. Sibson and G. Stone (1991), "Computation of thin-plate splines", *SIAM Journal on Scientific and Statistical Computing*, Vol. 12, pp. 1304–1313.

List of symbols

$\mathbb{R}$	the real numbers,
$\mathbb{R}^d$	the d-dimensional space of real numbers,
$\Pi_m(\mathbb{R}^d)$	all polynomials in d variables of total degree at most m,
S	the linear space of thin plate splines,
$(\cdot\,,\cdot)$	the semi-inner product on S,
$\|\cdot\|$	the semi-norm on S arising from the semi-inner product $(\cdot,\cdot)$
S_θ	the linear space spanned by functions which are translates of a conditionally definite function θ, and $\Pi_m(\mathbb{R}^d)$ for a suitable integer m,
$(\cdot\,,\cdot)_\theta$	the semi-inner product on S_θ,
$\|\cdot\|_\theta$	the semi-norm on S_θ arising from the semi-inner product $(\cdot,\cdot)_\theta$,
σ_θ	takes the value $+1$ if θ is conditionally positive definite, and -1 if θ is conditionally negative definite,
T_θ	subspace of S_θ consisting of functions with centres at the last q data points $\underline{x}_{n-q+1}\ldots,\underline{x}_n$,
I	identity map on S_θ,
I_k	$k\times k$ identity matrix,
δ_{ij}	Kronecker delta.